Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут»

**В.Я. Жуйков, Т.О. Терещенко,
Ю.С. Ямненко, А.В. Заграничний**

МІКРОПРОЦЕСОРНА ТЕХНІКА

*Рекомендовано Методичною радою НТУУ «КПІ»
як електронний підручник для студентів спеціальності
«Електроніка»*

Київ
2016

УДК 681.3:621.38
ББК 32.973.26-04
Ж59

Гриф надано Міністерством освіти і науки України
(Лист №1/11-1794 від 04.06.2002 р.)

Рецензенти: *М.М.Юрченко*, д-р техн. наук, проф. (Інститут електродинаміки НАН України)
В.В. Каплун, д-р. техн. наук, проф. (Київський національний університет технологій та дизайну)

Мікропроцесорна техніка: Електронний підручник / В.Я. Жуйков, Ж59 Т.О. Терещенко, Ю.С. Ямненко, А.В.Заграничний ; відп. ред. О.В. Борисов. 2016. – 440 с.

ISBN 966–622–135–7

Розглянуто системи числення і кодування інформації; наведено загальні принципи побудови мікропроцесорів, мікроконтролерів та мікропроцесорних систем; розглянуто особливості архітектури і функціональні можливості 8-, 16-, 32- та 64-розрядних процесорів; способи організації та побудову модулів пам'яті і роботу інтерфейсів пристроїв уведення-виведення. Подано матеріал про архітектуру однокристальних мікроконтролерів з *CISC*- і *RISC*-архітектурою, сигнальних процесорів та нейропроцесорів. Наведено приклади складання програм та проектування мікропроцесорних систем.

Для студентів бакалаврату «Електроніка», «Електротехніка» та інших бакалаврів, програмою яких передбачено вивчення дисципліни «Мікропроцесорна техніка». Може бути корисним для аспірантів і спеціалістів, що працюють у галузі електроніки.

УДК 681.3:621.38
ББК 32.973.26-04

ISBN 966–622–135–7

Зміст

Умовні скорочення

АЛП – арифметико-логічний пристрій

АЦП – аналого-цифровий перетворювач

БВВ – буфер вибірки з випередженням

БКОПК – блок конвеєрних обчислень з плавучою комою

БПАП – блок передбачення адреси переходу

БШ – буфер шини

ВІС – велика інтегральна схема

ДДК – двійково-десятковий код

ДШК – дешифратор команд

ЕЗЛ – емітернозв'язана логіка

ЕП – елемент пам'яті

ЕОМ – електронна обчислювальна машина

ЗГ – задавальний генератор

ЗЗП – зовнішній запам'ятовувальний пристрій

ЗП – запам'ятовувальний пристрій

ІВВ – інтерфейс введення-виведення

КЗ – керувальний затвор

КМДН – комплементарна технологія метал-діелектрик-напівпровідник

КПДП – контролер прямого доступу до пам'яті

МДН – метал-діелектрик-напівпровідник

МП – мікропроцесор

МПК – мікропроцесорний комплект

МПС – мікропроцесорна система

НОЗП – надоперативний запам'ятовувальний пристрій

ОЗП – оперативний запам'ятовувальний пристрій

ОМК – однокристальний мікроконтролер

ПВВ – пристрій введення-виведення

ПВЗ – пристрій вибірки/зберігання

ПДП – прямий доступ до пам’яті

ПЕ – процесорний елемент

ПЗ – плавучий затвор

ПЗП – постійний запам’ятовувальний пристрій

ПЗПМ – ПЗП, програмовний маскою

ПК – пристрій керування

ПКП – програмовний контролер переривань

ПЛМ – програмовна логічна матриця

ППЗП – програмовний ПЗП

ППІ – програмовний паралельний інтерфейс

ПТ – програмовний таймер

РЗП – регістр загального призначення

РК – регістр команд

РПД – резидентна пам’ять даних

РПЗП – репрограмовний ПЗП

РПП – резидентна пам’ять програм

СІД – схема інкремента/декремента

СПВВ – спеціалізований процесор введення-виведення

ТТЛ – транзисторно-транзисторна логіка

УСАПП – універсальний синхронно-асинхронний приймач-передавач

ЦАП – цифро-аналоговий перетворювач

ЦП – центральний процесор

ШІМ – широтно-імпульсний модулятор

Передмова

Мета підручника – отримання студентами теоретичних знань та практичних навиків побудови і програмування мікропроцесорних систем керування, проектування інтерфейсів введення-виведення, спеціалізованих пристроїв на МП різних серій та мікро-ЕОМ. Підручник складено за матеріалами лекцій з дисциплін, пов'язаних з мікропроцесорною технікою та включених у навчальний процес Національного технічного університету України «Київський політехнічний інститут» і Національного технічного університету «Харківський політехнічний інститут»

У першому розділі розглянуто системи числення і кодування інформації. Особливу увагу приділено формам подання чисел у мікропроцесорах.

У другому розділі подано основні поняття та визначення мікропроцесорної техніки, класифікацію мікропроцесорів, принципи побудови мікропроцесорних систем, основи програмування мовою асемблер.

У третьому розділі розглянуто архітектуру і функціональні можливості 8- і 16-розрядних мікропроцесорів, виконання команд за машинними тактами і циклами. Закінчується розділ прикладом побудови модуля центрального процесора.

У четвертому розділі йдеться про особливості архітектури 32- та 64-розрядних мікропроцесорів: сегментну і сторінкову організацію пам'яті, захист за привілеями, перемикання задач тощо.

П'ятий розділ присвячено проектуванню системи пам'яті мікропроцесорних систем – модулів постійних та оперативних запам'ятовувальних пристроїв. Розглянуто принципи дії та структурні схеми основних типів кеш-пам'яті, стекову організацію пам'яті.

Шостий розділ розкриває питання побудови інтерфейсів пристроїв введення-виведення. Розглянуто організацію каналу прямого доступу до пам'яті та систему переривань. Розділ закінчується прикладом розроблення мікропроцесорної системи керування.

У сьомому розділі розглянуто архітектуру і функціональні можливості однокристальних мікроконтролерів з *CISC*-архітектурою, а у восьмому – з *RISC*-архітектурою.

Дев'ятий розділ присвячено сигнальним процесорам обробки даних у форматах із фіксованою та плавучою комою.

У десятому розділі наведено відомості про нейронні обчислювачі, основи побудови алгоритмів навчання нейронних мереж та апаратну реалізацію нейронних обчислювачів.

Відгуки і побажання прохання надсилати на адресу:

03056, Київ-56, вул. Політехнічна, 16, к. 307
Тел. (044) 441-13-53
Факс (044) 236-96-76
E-mail: Petergerya@edd.ntu-kpi.kiev.ua

Вступ

Важливе місце у схемотехніці електронних систем посідають системи керування з мікропроцесорами та мікроконтролерами, які дозволяють реалізувати складні закони керування електронними пристроями. Знання схемотехніки аналогових та цифрових систем створює базу для вивчення принципів побудови мікропроцесорних систем керування. Перевага мікропроцесорних систем керування – їх гнучкість: систему, розроблену для виконання конкретного завдання керування, легко пристосувати для вирішення інших завдань зміною програмного забезпечення.

Перший мікропроцесор (МП) *Intel* 4004 з'явився 1971 року. Він працював на частоті 750 кГц, уміщував 2300 транзисторів і мав 4-розрядну шину даних. Цей винахід визнано одним з найбільших досягнень XX сторіччя. Сучасні мікропроцесорні великі інтегральні схеми (ВІС), наприклад однокристальні мікроконтролери (ОМК), містять усі складові ЕОМ – МП, пам'ять даних, пам'ять програм, інтерфейсні схеми – та ефективно використовуються в системах керування промислового та побутового обладнання.

Розширення функцій мікропроцесорних систем (МПС) потребувало вдосконалення знань спеціалістів різних профілів у цьому напрямі. Тому вивчення основ побудови та програмування мікропроцесорів є неодмінною складовою підготовки спеціалістів вищих навчальних закладів. Незважаючи на велику різноманітність типів МП та функцій, що вони виконують, логіка побудови систем і створення програмного забезпечення залишається незмінною. Вивчення загальних принципів побудови, особливостей архітектури, використання різних видів пам'яті та програмування мікропроцесорних комплектів дає теоретичну базу для розробки і використання мікропроцесорних систем різних типів.

Розділ 1

СИСТЕМИ ЧИСЛЕННЯ І КОДУВАННЯ ІНФОРМАЦІЇ

1.1. Типи систем числення

Найчастіше в обчисленнях використовується десяткова система числення, повна назва якої – однорідна десяткова позиційна система числення з безпосереднім поданням чисел.

Число – це величина, яка виражає кількість однорідних об'єктів. *Системою числення* називають сукупність правил запису чисел обмеженою кількістю символів, названих *цифрами*. Системи числення поділяють на позиційні і непозиційні. Серед позиційних систем розрізняють системи з безпосереднім або кодованим поданням чисел. *Розрядом* цифри називається місце (або позиція) цифри в позиційній системі. *Основою p* системи числення називають кількість різних цифр, які застосовуються для написання чисел: 0, 1,..., p – 1. На рис. 1.1 наведено класифікацію систем числення. Курсивом виділено подання десяткового числа 11 у різних системах числення.

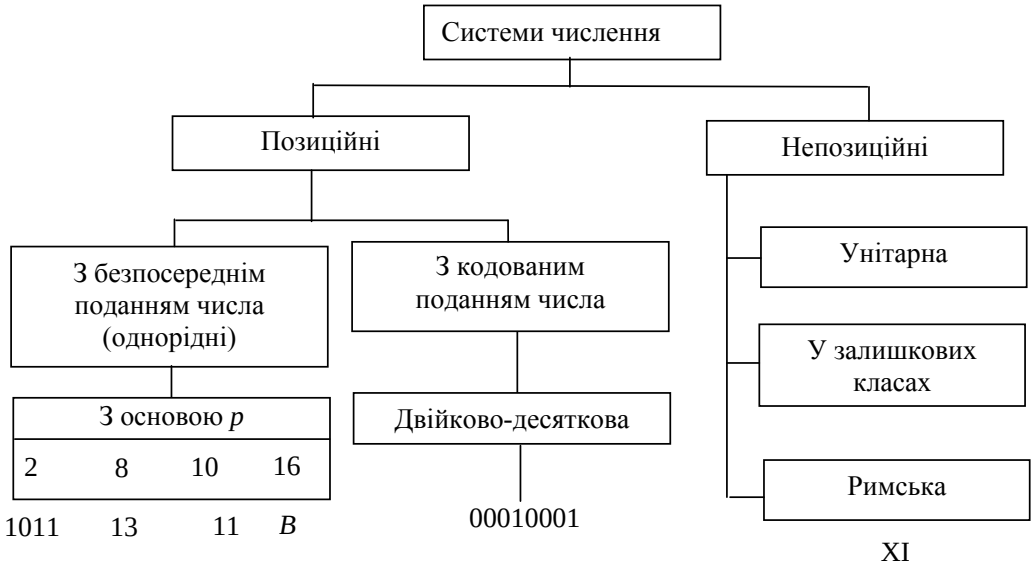


Рис. 1.1. Класифікація систем числення

До непозиційних систем числення належать: римська, унітарна і система залишкових класів.

У римській системі числення використовують такі цифри: $I = 1$, $V = 5$, $X = 10$, $L = 50$, $C = 100$, $D = 500$, $M = 1000$. У цій системі число 2003 записується як *MMIII*. В унітарній системі число подають загальною сукупністю однорідних об'єктів. У системі залишкових класів числа подають остачами від ділення на прості числа. У цій системі всі операції можуть виконуватися окремо для цифр кожного розряду.

До позиційних систем числення відносять системи, у яких кожна цифра займає певне положення (розряд або позицію) у ряду цифр, що зображують число. Щоб одержати значення числа, потрібно кожну цифру розряду помножити на число, яке називається *вагою розряду*. Ваги окремих розрядів являють собою геометричну прогресію зі знаменником, що дорівнює основі системи числення p. Наприклад, розряди десяткового числа 1327,45 мають ваги: $10^3 = 1000$; $10^2 = 100$; $10^1 = 10$; $10^0 = 1$; $10^{-1} = 0{,}1$; $10^2 = 0{,}01$. Позиційні системи числення, в яких цифри всіх розрядів набувають значення 0,1,…, p–1, а основа p є однаковою для всіх розрядів, називають *однорідними*. Подання числа X в однорідній позиційній системі числення з основою p має вигляд:

$$X = \sum_{s=1}^{n} x^{(s)} p^{n-s} = x^{(1)} p^{n-1} + x^{(2)} p^{n-2} + \ldots + x^{(n)} p^0 = x^{(1)} x^{(2)} \ldots x^{(n)}. \quad (1.1)$$

Число X, що містить n розрядів цілої частини і k розрядів дробу, можна виразити формулою

$$X = \sum_{s=1}^{n+k} x^{(s)} p^{n-s} = x^{(1)} p^{n-1} + x^{(2)} p^{n-2} + \ldots + x^{(n)} p^0 + x^{(n+1)} p^{-1} + \\ + x^{(n+2)} p^{-2} + \ldots + x^{(n+k)} p^{-k} = x^{(1)} x^{(2)} \ldots x^{(n)}, \; x^{(n+1)} x^{(n+2)} \ldots x^{(n+k)}. \quad (1.2)$$

Десяткова система або система з основою 10 ($p = 10$) оперує з 10 цифрами (від 0 до 9). У системах числення з основою більше 10 використовують 10 цифр для молодших значень цифр розрядів і латинські літери *A*, *B*, *C*… – для старших.

Якщо необхідно позначити основу системи числення, то використовують числові індекси або латинські літери: для двійкового числа індекс 2 або літера *B (Binary),* для десяткового – індекс 10 або літера *D (Decimal),* для шістнадцяткового – індекс 16 або літера *H (Hexadecimal).*

Приклад 1.1. Записати число 11_{10} у двійковій системі числення.

У двійковій системі числення основа $p = 2$, а цифри розрядів можуть набувати значень 0 або 1. Згідно з формулою (1.1) число 11_{10} у двійковій системі числення записують як

$$X = \sum_{s=1}^{4} x^{(s)} 2^{n-s} = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 1011_2.$$

Приклад 1.2. Записати число 11_{10} у вісімковій системі числення.

У вісімковій системі числення основа $p = 8$, а розрядні компоненти можуть набувати значень {0, 1, 2, 3, 4, 5, 6, 7}. Згідно з формулою (1.1) число 11_{10} у вісімковій системі числення можна записати так:

$$X = \sum_{s=1}^{2} x^{(s)} 8^{n-s} = 1 \cdot 8^1 + 3 \cdot 8^0 = 13_8 .$$

Приклад 1.3. Записати число 11_{10} у шістнадцятковій системі числення.

Для шістнадцяткової системи $p = 16$, $x^{(i)} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$. Згідно з формулою (1.1)

$$X = B \cdot 16^1 = 0B_{16} = 0BH .$$

Приклад 1.4. Подати число $1327{,}45_{10}$ у вигляді полінома (1.2).

Значення ваг та цифр розрядів числа 1327,45 у десятковій системі числення ілюструє табл. 1.1 згідно з формулою (1.2):

$$1327{,}45 = 1 \cdot 10^3 + 3 \cdot 10^2 + 2 \cdot 10^1 + 7 \cdot 10^0 + 4 \cdot 10^{-1} + 5 \cdot 10^{-2}.$$

Таблиця 1.1. **Значення ваг та цифр розрядів у десятковій системі**

Степінь основи	3	2	1	0	–1	–2	
Значення ваг позицій	1000	100	10	1	0,1	0,01	
Цифра розряду	1	3	2	7	4	5	
Число	1000	+ 300	+ 20	+ 7	+ 0,4	+ 0,05	$= 1327{,}45_{10}$

Розглянуті у прикладах системи, у яких кожній цифрі розряду відповідає окремий символ, називаються *системою з безпосереднім поданням чисел*.

Систему, в якій кількість символів менша, ніж кількість цифр, а кожну цифру кодують певною комбінацією кількох символів, називають *системою з кодованим поданням чисел*. Такою є, наприклад, двійково-десяткова система числення, яка містить десять цифр, але тільки два символи, причому кожна з десяти цифр кодується числами двійкової системи: 0 = 0000, 1 = 0001,…, 9 = 1001.

Приклад 1.5. Записати число 11_{10} у двійково-десятковій системі числення.

Для кожної цифри десяткового числа запишемо її двійковий еквівалент, який займає 4 розряди ($1_{10} = 0001_2$). Отже, число 11_{10} у двійково-десятковій системі буде подано таким чином:

$$11_{10} = 0001\ 0001_{2\text{-}10}.$$

Контрольні запитання

1. Дайте визначення системи числення.
2. Наведіть приклади позиційних систем числення з основою 2, 10, 16.
3. Наведіть приклад системи числення з кодованим поданням чисел.

1.2. Двійкова система числення

Двійкова система числення, або система з основою 2, використовує цифри 0 і 1. Такі цифри називаються *бітами* (***Bi**nary Digit**s***). Фізично в цифрових електронних системах значення 0 відповідає напрузі низького рівня (*L*-рівня), а значення 1 – напрузі високого рівня (*H*-рівня).

У табл. 1.2 наведено значення ваг перших чотирьох двійкових позицій і показано відповідність між двійковим числом 1001_2 та його десятковим еквівалентом 9_{10}. Розряд, якому відповідає значення ваги позиції 1, називається *молодшим бітом*, а розряд, якому відповідає найбільше значення ваги позиції (у цьому разі 8), – *старшим бітом*.

Таблиця 1.2. **Значення позицій двійкових чисел**

Степінь основи	3	2	1	0
Значення ваг позицій	8	4	2	1
Двійкове число	Старший біт 1	0	0	Молодший біт 1
Десяткове число	8	+ 0	+ 0	+ 1 = 9_{10}

У табл. 1.3 подано десяткові числа від 0 до 15 та їх двійкові еквіваленти.

Таблиця 1.3. **Двійкові еквіваленти десяткових чисел від 0 до 15**

Числа											
десяткові		двійкові				десяткові		двійкові			
Значення ваг позицій											
10^1	10^0	2^3	2^2	2^1	2^0	10^1	10^0	8	4	2	1
0	0	0	0	0	0	0	8	1	0	0	0
0	1	0	0	0	1	0	9	1	0	0	1
0	2	0	0	1	0	1	0	1	0	1	0
0	3	0	0	1	1	1	1	1	0	1	1
0	4	0	1	0	0	1	2	1	1	0	0
0	5	0	1	0	1	1	3	1	1	0	1
0	6	0	1	1	0	1	4	1	1	1	0
0	7	0	1	1	1	1	5	1	1	1	1

Перетворення двійкового числа на десятковий еквівалент. Під кожним бітом двійкового числа записують десяткові значення кожної позиції. Десяткові числа підсумовують. Приклад перетворення двійкового числа $1011\ 0110_2$ на десятковий еквівалент наведено у табл. 1.4.

Таблиця 1.4. **Перетворення двійкового числа на десятковий еквівалент**

Степінь основи	7	6	5	4	3	2	1	0	
Значення ваг позицій	128	64	32	16	8	4	2	1	
Двійкове число	1	0	1	1	0	1	1	0	
Десяткове число	128	+ 0	+ 32	+ 16	+ 0	+ 4	+ 2	+ 0	= 182_{10}

Перетворення десяткового числа на двійковий еквівалент. Десяткове число ділять на 2. Остачу у вигляді 0 або 1 записують у молодший

розряд двійкового числа. Частку від ділення знов ділять на 2, остачу (0 або 1) записують у наступний після молодшого розряд. Ці дії виконують доти, доки частка від чергового ділення не дорівнюватиме 1. Одиницю записують у старший розряд двійкового числа. Приклад перетворення десяткового числа 155_{10} на двійковий еквівалент 10011011_2 наведено на рис. 1.2.

Для перетворення цілого числа X, записаного в системі числення з основою p, на його еквівалент у системі числення з основою q слід ділити X на q до отримання цілої остачі, меншої від q (метод послідовного ділення). Число X у системі числення з основою q подається послідовністю остач ділення в порядку, зворотному їхньому одержанню, причому старшу цифру в числі X дає остання остача.

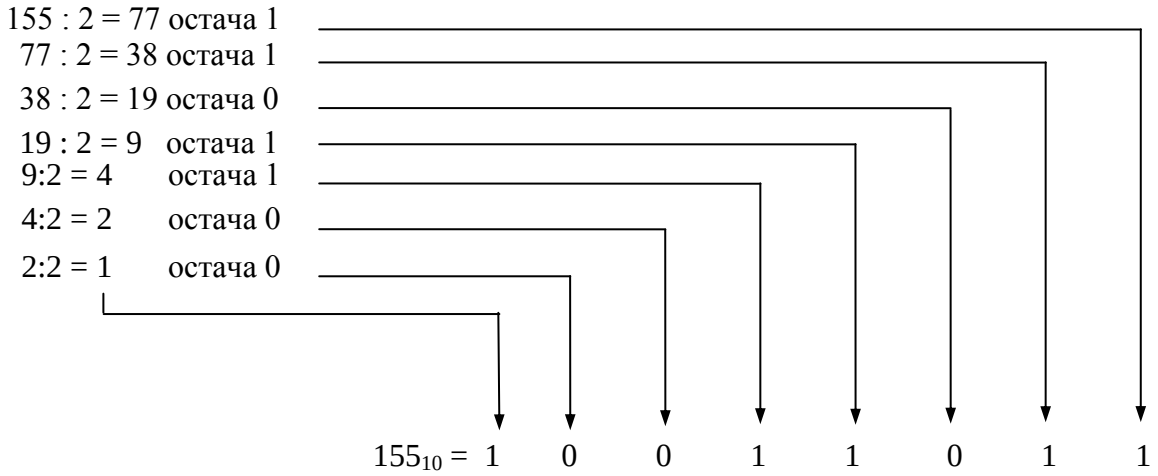


Рис. 1.2. Перетворення десяткового числа 155_{10} на двійковий еквівалент

Для чисел, що мають як цілу, так і дробову частину, переведення з однієї системи числення в іншу здійснюється окремо для цілої і дробової частин. Для перетворення правильного дробу X, записаного у системі числення з основою p, на його еквівалент в системі числення з основою q слід послідовно множити X на q, причому множити слід тільки дробові частини (метод послідовного множення). Еквівалент X у системі числення з основою q подається у вигляді послідовності цілих частин результатів множення у порядку їхнього одержання, причому старший розряд є цілою частиною першого результату. Якщо необхідно виконати перетворення з точністю q^{-k}, то кількість послідовних множень дорівнює k.

Приклад 1.6. Записати десятковий дріб $0{,}366_{10}$ з точністю 2^{-8} у двійковій системі числення.

Дробові частини заданого числа та чисел, що утворюються у результаті множення, 8 разів послідовно множимо на 2 (рис. 1.3).

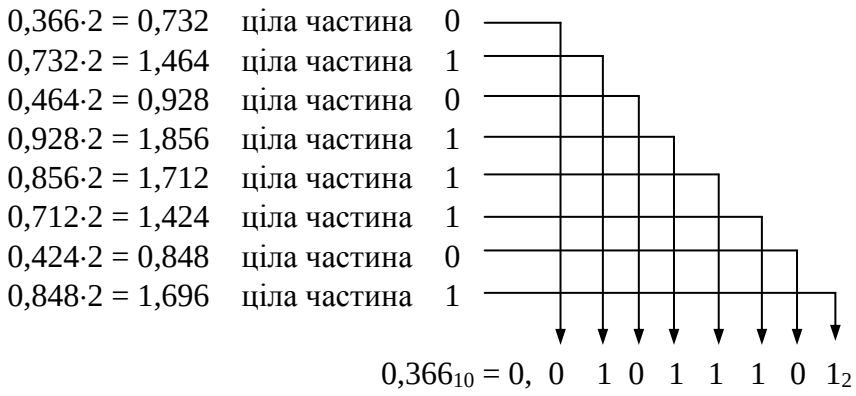


Рис. 1.3. Перетворення десяткового числа $0{,}366_{10}$ на двійковий еквівалент з точністю 2^{-8}

У табл. 1.5 наведено перетворення одержаного двійкового числа на десятковий еквівалент, де під кожним бітом двійкового числа записані десяткові значення кожної позиції, які підсумовують.

Таблиця 1.5. **Перетворення двійкового числа $0{,}01011101_2$ на десятковий еквівалент**

Степінь основи	0	–1	–2	–3	–4	–5	–6	–7	–8
Значення ваг позицій	1	0,5	0,25	0,125	0,0625	0,03125	0,015625	0,0078125	0,00390625
Двійкове число	0	0	1	0	1	1	1	0	1
Десяткове число	0+0+0,25+0+0,0625+0,03125+0,015625+0+0,00390625 = 0,36328125								

Отримане число $0{,}36328125_{10}$ наближує задане число 0,366 з точністю $2^{-8} = 0{,}0039$.

Контрольні запитання

1. Що означає біт у двійковій системі?
2. Перетворіть на десятковий код такі двійкові числа: а) 0001; б) 0101; в) 1000; г) 1011; д) 1111; е) 0111.
3. Перетворіть на двійковий код наступні десяткові числа з точністю 2^{-6}: а) 23,55; б) 39,41.

1.3. Двійкова арифметика

Додавання, віднімання або множення двійкових чисел виконують за такими самими правилами, як і в арифметиці десяткових чисел. Правила додавання однорозрядних двійкових чисел наведено у табл. 1.6. Правила 1 та 2 очевидні; правило 3 ілюструє перенесення одиниці у старший розряд: 1 + 1 = 10. Правило 4 показує, що результатом додавання трьох 1 є число 11.

Таблиця 1.6. **Правила двійкового додавання**

Номер правила	1	2	3	4
1-й доданок	0	0	1	1 (Перенесення з молодшого розряду) + 1
2-й доданок	+ 0	+ 1	+ 1	1
Сума	0	1	10	11
			Перенесення 1 до старшого розряду	Перенесення 1 до старшого розряду

Приклад 1.7. Додати два 8-розрядних двійкових числа: 10100010_2 і 01110101_2.

Виконуємо додавання двох чисел, використавши правила двійкового додавання (табл. 1.6) для кожного з розрядів. У цьому прикладі результат додавання 8-розрядних чисел є 8-розрядним:

```
  1   111   – рядок перенесень
  10100011
+ 00110101
----------
  11011000
```

Правильність додавання перевіримо в десятковій системі (табл. 1.7).

Таблиця 1.7. **Додавання у двійковій та десятковій системах**

Значення ваг розрядів / Числа		2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1-й доданок	Двійкове	1	0	1	0	0	0	1	1
	Десяткове	128		+ 32				+ 2	+ 1 = 163_{10}
2-й доданок	Двійкове	0	0	1	1	0	1	0	1
	Десяткове			32	+ 16		+ 4		+ 1 = 53_{10}
Результат	Двійкове	1	1	0	1	1	0	0	0
	Десяткове	128	+ 64		+ 16	+ 8			= 216_{10}

Приклад 1.8. Додати два двійкових числа: 10100011_2 та 01110101_2.

Виконаємо додавання у двійковому вигляді. Результат є 9-розрядним числом:

```
  1 1 1           1 1
    1 0 1 0 0 0 1 1
+   0 1 1 1 0 1 0 1
-----------------
1 0 0 0 1 1 0 0 0
```

Результат перевіримо додаванням у десятковому вигляді:

```
+ 163
  117
-----
  280
```

Якщо для позначення кожного двійкового числа відводиться вісім розрядів, то одержання 9-розрядного результату додавання двох чисел призводить до некоректної роботи мікропроцесора. У прикладі 1.8 у восьми молодших розрядах результату додавання у двійковому вигляді знаходиться число $00011000_2 = 24_{10}$. При цьому відбувається перенесення 1 у дев'ятий розряд, вага якого $2^8 = 256$. Це перенесення можна врахувати програмно і скоректувати неправильний результат, наприклад, розміщенням перенесення у додатковому регістрі. Тоді двобайтове число буде містити 256 + 24 = 280, тобто правильний результат.

У табл. 1.8 наведено правила віднімання однорозрядних двійкових чисел. Правила 1 – 3 аналогічні правилам десяткового віднімання. Правило 4 потребує позики зі старшого розряду так, що зменшуваним є число 10, від'ємником – 1 і різницею – 1.

***Таблиця 1.8*. Правила двійкового віднімання**

Номер правила	1	2	3	4
Зменшуване	0	1	1	1 0 (Позика зі старшого розряду) 0
–	–	–	–	
Від'ємник	0	0	1	1
Різниця	0	1	0	1

Приклад 1.9. Відняти від двійкового числа 10100011_2 число 00110101_2.

Виконуємо віднімання двох чисел, використавши правила двійкового віднімання (див. табл. 1.8):

$$\begin{array}{rll} 11111 & \text{– рядок позик} & \\ 10100011 & = & 163_{10} \\ -\ 00110101 & = & 53_{10} \\ \hline 01101110 & = & 110_{10} \end{array}$$

Відзначимо, що віднімання більшого числа від меншого приводить до некоректного результату, оскільки є потреба у позиці з дев'ятого розряду. Результат віднімання у цьому випадку треба скоректувати програмно.

У табл. 1.9 подано правила множення однорозрядних двійкових чисел.

***Таблиця 1.9*. Правила двійкового множення**

Номер правила	1	2	3	4
Множник	0	1	0	1
	×	×	×	×
Множник	0	0	1	1
Добуток	0	0	0	1

У перших трьох випадках принаймні один множник дорівнює 0, тому добуток дорівнює 0. У четвертому випадку множниками є одиниці, тому добутком є 1.

Приклад 1.10. Перемножити два двійкових числа: 1101_2 і 101_2.

Знайдемо добуток, використавши правила (див. табл. 1.9):

$$\begin{array}{r} \times \quad 1101 \\ \quad\quad 101 \\ \hline \quad\quad 1101 \\ + \quad 0000 \\ \quad\quad 1101 \\ \hline \quad 1000001 \end{array}$$

Для перевірки подамо множники у десятковому вигляді: $1101_2 = 13_{10}$, $101_2 = 5_{10}$, результат множення $13 \times 5 = 65_{10} = 1000001_2$.

Відзначимо, що розрядність добутку дорівнює сумі розрядів множників.

Контрольні запитання

1. Виконайте додавання чисел 10110011_2 та 11110010_2 і перевірте результат у десятковій системі.
2. Виконайте віднімання чисел 132_{10} та 77_{10} у двійковому вигляді.
3. Виконайте множення чисел 110110_2 та 111_2 і перевірте результат у десятковій системі.
4. Визначте розрядність результату множення чисел 11011100_2 та 11101111_2.

1.4. Інші поширені системи числення

Трійкова система числення. Існують три канонічні трійкові системи числення: дві зміщені, цифри розрядів яких набувають значень {0, 1, 2} та {–2, –1, 0}, і одна симетрична, цифри розрядів якої набувають значень {–1, 0, 1}. У симетричній системі числення використовуються трійкові вагові коефіцієнти розрядів: 3^0, 3^1, 3^2, …, а значення –1, 0, 1 відповідають трьом логічним рівням напруги. Подання чисел у симетричній системі має вигляд:

$$X = \sum_{s=1}^{n} x^{(s)} 3^{n-s} = x^{(1)} 3^{n-1} + x^{(2)} 3^{n-2} + \ldots + x^{(n)} 3^0,$$

де $x^{(i)} \in \{-1, 0, 1\}$.

У табл. 1.10 наведено еквіваленти дев’яти додатних і від’ємних десяткових чисел у симетричній трійковій системі числення.

Таблиця 1.10. **Десяткові числа та їх еквіваленти у симетричній трійковій системі**

Числа							
десяткові додатні	симетричні трійкові			десяткові від'ємні	симетричні трійкові		
Значення ваг позицій							
10^0	3^2	3^1	3^0	10^0	3^2	3^1	3^0
1	0	0	1	–1	0	0	–1
2	0	1	–1	–2	0	–1	1
3	0	1	0	–3	0	–1	0
4	0	1	1	–4	0	–1	–1
5	1	–1	–1	–5	–1	1	1
6	1	–1	0	–6	–1	1	0
7	1	–1	1	–7	–1	1	–1
8	1	0	–1	–8	–1	0	1
9	1	0	0	–9	–1	0	0

Правила алгебричного додавання однорозрядних чисел у симетричній трійковій системі наведено у табл. 1.11.

Приклад 1.11. Додати два трійкових багаторозрядних числа: -1101_3 та $01-10_3$.

Знайдемо результат алгебричного додавання двох чисел, використавши правила (див. табл. 1.11):

$$\begin{array}{rcccc} & 1 & & & \\ & -1 & 1 & 0 & 1 \\ & 0 & 1 & -1 & 0 \\ + & & & & \\ \hline & 0 & -1 & -1 & 1 \end{array}$$

Подамо числа у десятковій системі числення та виконаємо їх додавання:

$$-1101_3 = -17_{10};$$

$$01-10_3 = 6_{10};$$

$$-17_{10} + 6_{10} = -11_{10}.$$

Таблиця 1.11. **Правила додавання у симетричній трійковій системі**

Номер правила	1	2	3	4	5	6
1-й доданок	0	0	0	1	1	1
	+	+	+	+	+	+
2-й доданок	0	1	–1	1	–1	–1
Сума	0	1	–1	–1	0	1

Правила множення однорозрядних чисел у симетричній трійковій системі наведено у табл. 1.12.

***Таблиця 1.12*. Правила множення у симетричній трійковій системі**

Номер правила	1	2	3	4	5	6
Множник	0	0	0	1	1	–1
	×	×	×	×	×	×
Множник	0	1	–1	1	–1	–1
Добуток	0	0	0	1	–1	1

Приклад 1.12. Перемножити два трійкових числа: -1011_3 і -11_3.
Результат множення визначимо згідно з правилами табл. 1.12:

$$\begin{array}{rcccc} & -1 & 0 & 1 & 1 \\ \times & & & -1 & 1 \\ \hline & -1 & 0 & 1 & 1 \\ 1 & 0 & -1 & -1 & \\ \hline 1 & -1 & -1 & 0 & 1 \end{array}$$

Результат множення у десятковому вигляді:

$$1-1-101 = 1\cdot 3^4 + (-1)\cdot 3^3 + (-1)\cdot 3^2 + 0\cdot 3^1 + 1\cdot 3^0 = 81-27-9+1 = 46.$$

Дійсно,

$$-23_{10} \times -2_{10} = 46_{10}.$$

Скоротити довжину запису двійкових чисел можна, застосовуючи системи числення з кратною основою, тобто системи, основи яких p_1 і p_2 зв'язані співвідношенням $p_2 = p_1^k$, де k – ціле додатне число**.** Прикладами таких систем є двійкова, вісімкова і шістнадцяткова системи ($2^3 = 8$; $2^4 = 16$). Вісімкову і шістнадцяткову системи числення, як правило, використовують як допоміжні для скорочення трудомісткості обробки інформації.

Вісімкова система числення. Вісімкова система – це система з основою 8, яка містить вісім цифр від 0 до 7. У табл. 1.13 подано десяткові, вісімкові та двійкові еквіваленти 16 перших чисел десяткової системи.

***Таблиця 1.13*. Десяткові, вісімкові і двійкові еквіваленти чисел $0–15_{10}$**

Десятковий еквівалент	Вісімковий еквівалент	Двійковий еквівалент			
		Значення ваг позицій			
		8	4	2	1
0	0	0	0	0	0
1	1	0	0	0	1
2	2	0	0	1	0
3	3	0	0	1	1
4	4	0	1	0	0
5	5	0	1	0	1
6	6	0	1	1	0
7	7	0	1	1	1

Продовження табл. 1.13

Десятковий еквівалент	Вісімковий еквівалент	Двійковий еквівалент			
		Значення ваг позицій			
		8	4	2	1
8	10	1	0	0	0
9	11	1	0	0	1
10	12	1	0	1	0
11	13	1	0	1	1
12	14	1	1	0	0
13	15	1	1	0	1
14	16	1	1	1	0
15	17	1	1	1	1

Для того щоб перетворити двійкове число на вісімковий еквівалент, його ділять на тріади – групи по три біти, починаючи з молодшого біта. Якщо кількість бітів не кратна 3, то двійкове число доповнюється нулями зліва. Потім кожну тріаду замінюють на еквівалентну вісімкову цифру відповідно до табл. 1.13. Наприклад, вісімковий еквівалент числа 11111000100_2 – це число 3704_8:

Двійкове число	011	111	000	100
Вісімкове число	3	7	0	4

Приклад 1.13. Перетворити вісімкове число 6521_8 на двійковий еквівалент.

Для перетворення вісімкового числа кожну його цифру замінюють двійковою тріадою:

Вісімкове число	6	5	2	1
Двійкове число	110	101	010	001

Отже, $6521_8 = 110101010001_2$.

Приклад 1.14. Перетворити вісімкове число 2357_8 на десятковий еквівалент.

Перетворення виконують за табл. 1.14, у другому рядку якої вказано ваги чотирьох перших позицій вісімкового числа. Отже, $2357_{10} = 1263_{10}$.

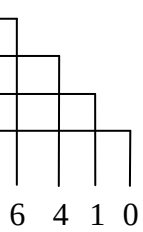
***Таблиця 1.14*. Вісімково-десяткове перетворення**

Степені числа 8	8^3	8^2	8^1	8^0
Значення ваг позицій	512	64	8	1
Вісімкове число 2357_8	2	3	5	7
Десяткове число	512×2	$+ 64 \times 3$	$+ 8 \times 5$	$+1 \times 7 = 1263_{10}$

Приклад 1.15. Перетворити десяткове число 3336_{10} на вісімковий еквівалент.

При перетворенні десяткове число 3336_{10} ділять на 8, що дає частку 417_{10} і остачу $0_{10} = 0_8$. Таким чином, молодший розряд вісімкового числа має значення 0. Частка 417_{10} стає діленим і її знову ділять на 8, що дає частку 52_{10} і остачу $1_{10} = 1_8$, яка стає значенням другого розряду вісімкового числа. Ділення 52_{10} на 8 дає частку 6_{10} і остачу $4_{10} = 4_8$. Останню частку 6_{10} ділять на 8 із часткою 0 і остачею $6_{10} = 6_8$. Оскільки остання частка дорівнює 0, то цифра 6_8 стає значенням старшого розряду вісімкового числа:

$3336_{10} : 8 = 417_{10}$ остача $0_{10} = 0_8$
$417_{10} : 8 = 52_{10}$ остача $1_{10} = 1_8$
$52_{10} : 8 = 6_{10}$ остача $4_{10} = 4_8$
$6_{10} : 8 = 0$ остача $6_{10} = 6_8$



Отже, $3336_{10} = 6410_8$.

Шістнадцяткова система числення є системою з основою 16 та містить 16 символів: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *A*, *B*, *C*, *D*, *E*, *F*. У табл. 1.15 наведено двійкові та шістнадцяткові еквіваленти 16 перших десяткових чисел. Кожну шістнадцяткову цифру подають єдиною комбінацією чотирьох двійкових цифр. Так, шістнадцятковим еквівалентом двійкового числа 10011110_2 є число $9E_{16}$. Це означає, що старшу тетраду (4 старші розряди) 1001 двійкового числа записують як 9_{16}, а молодшу тетраду 1110 – як E_{16}.

Таблиця 1.15. **Двійкові та шістнадцяткові еквіваленти десяткових чисел**

Десяткове число		Шістнадцятковий еквівалент	Двійковий еквівалент			
Значення ваг позицій						
10^1	10^0	16^0	2^3	2^2	2^1	2^0
0	0	0	0	0	0	0
0	1	1	0	0	0	1
0	2	2	0	0	1	0
0	3	3	0	0	1	1
0	4	4	0	1	0	0
0	5	5	0	1	0	1
0	6	6	0	1	1	0
0	7	7	0	1	1	1
0	8	8	1	0	0	0
0	9	9	1	0	0	1
1	0	*A*	1	0	1	0
1	1	*B*	1	0	1	1
1	2	*C*	1	1	0	0
1	3	*D*	1	1	0	1
1	4	*E*	1	1	1	0
1	5	*F*	1	1	1	1

Приклад 1.16. Перетворити двійкове число 111010_2 на шістнадцятковий еквівалент.

Для перетворення двійкового числа на шістнадцятковий еквівалент треба поділити його на тетради, починаючи з наймолодшого розряду, а потім кожну тетраду замінити еквівалентною шістнадцятковою цифрою. Значення молодшої тетради $1010_2 = A_{16}$, старшої – $0011_2 = 3_{16}$. Отже, $111010_2 = 3A_{16}$.

Приклад 1.17. Перетворити шістнадцяткове число $7F_{16}$ на двійковий еквівалент.

Для перетворення шістнадцяткового числа на двійковий еквівалент кожну шістнадцяткову цифру слід замінити на двійковий еквівалент – тетраду (див. табл. 1.15).

Еквівалентом шістнадцяткової цифри 7_{16} є двійкове число 0111_2, а цифри F_{16} – число 11112_2. Отже, $7F_{16} = 11110111_2$.

Приклад 1.18. Перетворити шістнадцяткове число $2C6E_{16}$ на десятковий еквівалент.

Перетворення виконують згідно з табл. 1.16. Кожну цифру шістнадцяткового числа множать на відповідну вагу позиції. Сума цих добутків дає десяткове число 11374_{10}.

***Таблиця 1.16*. Перетворення шістнадцяткового числа на десяткове**

Значення позицій	16^3	16^2	16^1	16^0
Значення ваг позицій	4096	256	16	1
Шістнадцяткове число	2	3	6	*E*
Десяткове число	2×4096	$+3 \times 256$	$+ 6 \times 16$	$+1 \times 14 = 11374_{10}$

Приклад 1.19. Перетворити десяткове число 15797 на шістнадцятковий еквівалент.

При перетворенні десяткове число 15797_{10} ділять на 16, що дає частку 987_{10} і остачу $5_{10} = 5_{16}$. Таким чином, молодший розряд шістнадцяткового числа має значення 5. Частка 987_{10} стає діленим і її знову ділять на 16, що дає частку 61_{10} і остачу $11_{10} = B_{16}$, яка стає значенням другого розряду шістнадцяткового числа. Ділення 61_{10} на 16 дає частку 3_{10} і остачу $13_{10} = D_{16}$. Ділення 3_{10} на 16 дає частку 0 і остачу $3_{10} = 3_{16}$. Оскільки остання частка дорівнює 0, то цифра 3_{16} стає значенням старшого розряду шістнадцяткового числа:

$15797_{10} : 16 = 987_{10}$ остача $5_{10} = 5_8$
$987_{10} : 16 = 61_{10}$ остача $11_{10} = B_{16}$
$61_{10} : 16 = 3_{10}$ остача $13_{10} = D_{16}$
$3_{10} : 16 = 0$ остача $3_{10} = 3_{16}$

3 *D B* 5

Отже, 1579710 = 3DB516.

Двійково-десяткова система числення – система, у якій кожну десяткову цифру від 0 до 9 подають 4-розрядним двійковим еквівалентом. Така система числення дозволяє скоротити програмні та апаратні витрати при перетворенні двійкових чисел, які використовують під час обробки інформації у процесорі, на десяткові, що виводять на пристрої відображення. Ця система є ефективною і при перетворенні десяткових чисел на двійкові. Двійково-десяткові числа записують з індексом 2-10 або ДДК (двійково-десятковий код), наприклад $01001001_{2\text{-}10}$, $0100_{\text{ДДК}}$.

Приклад 1.20. Перетворити десяткове число 3691_{10} на двійково-десятковий код.

При перетворенні кожну цифру десяткового числа перетворюють на двійковий 4-розрядний еквівалент:

Десяткове число	3	6	9	1
Двійково-десяткове число	0011	0110	1001	0001

Отже, $3691_{10} = 0011\ 0110\ 1001\ 0001_{2\text{-}10}$.

Приклад 1.21. Перетворити двійково-десяткове число $1000000001110010_{2\text{-}10}$ на десятковий еквівалент.

Кожна тетрада двійково-десяткового числа перетворюється на десятковий еквівалент:

Двійково-десяткове число	1000	0000	0111	0010
Десяткове число	8	0	7	2

Отже, $1000\ 0000\ 0111\ 0010_{2\text{-}10} = 8072_{10}$.

Контрольні запитання

1. Подайте числа: -18_{10}, 25_{10}, 32_{10} у симетричній трійковій системі числення.
2. Знайдіть десятковий еквівалент чисел у трійковій симетричній системі: а) $(110)_3$; б) $(-1-1)_3$.
3. Виконайте додавання трійкових чисел $(1-10-10)_3$ та $(1101-1)_3$ і перевірте результат у десятковій системі.
4. Виконайте множення трійкових чисел $(110)_3$ та $(1-10)_3$ і перевірте результат у десятковій системі.
5. Перетворіть на двійковий еквівалент числа: а) 3_8; б) 7_8; в) 7642_8; г) 2105_8.
6. Перетворіть на вісімковий еквівалент числа: а) 101_2; б) 110_2; в) 010_2; г) 111000101010_2.
7. Перетворіть на двійковий еквівалент числа: а) C_{16}; б) 6_{16}; в) F_{16}; г) E_{16}.
8. Перетворіть на шістнадцятковий еквівалент числа: а) 1001_2; б) 1100_2; в) 01111110_2; г) 11011011_2.
9. Перетворіть на двійково-десятковий еквівалент такі десяткові числа: а) 39_{10}; б) 65_{10}; в) 40_{10}; г) 17_{10}.
10. Перетворіть на десятковий еквівалент двійково-десяткові числа: а) $1000\ 0000_{2\text{-}10}$; б) $0000\ 0001_{2\text{-}10}$; в) $1001\ 0010_{2\text{-}10}$; г) $0111\ 0110_{2\text{-}10}$.

1.5. Подання чисел у мікропроцесорах

У регістрах або комірках пам’яті МП інформацію розміщено у вигляді двійкових чисел, причому для кожного розряду числа відведено окрему комірку, що зберігає один біт інформації. Сукупність комірок, призначених для розміщення одного двійкового числа, називають *розрядною сіткою*. Кількість комірок у розрядній сітці обмежена і залежить від конструктивних особливостей МП.

Спосіб розміщення розрядів числа в розрядній сітці визначається формою подання двійкових чисел: із фіксованою або плавучою комою.

Подання чисел у формі з фіксованою комою. Для розміщення двійкового числа, що містить цілу і дробову частини (без урахування знака) у n-розрядній сітці k комірок приділяють для розміщення цілої частини та $n–k$ комірок – для розміщення дробової. При такому поданні двійкових чисел положення коми у розрядній сітці фіксовано. Якщо кількість

розрядів у дробовій частині перевищує *n–k*, то молодші розряди, які знаходяться за межами розрядної сітки, не сприймаються мікропроцесором. Будь-яке двійкове число, менше ніж 2^{n-k}, сприймається як нульове і називається *машинним нулем*. У результаті відкидання молодших розрядів дробової частини числа, розташованої за межами розрядної сітки, виникає похибка подання. Максимальне значення абсолютної похибки подання Δ_1 не перевищує одиниці молодшого розряду сітки:

$$\Delta_1 = 2^{-(n-k)}.$$

При такій формі подання чисел мінімальне число $m = 2^{-(n-k)}$, максимальне $M = 2^k - 2^{-(n-k)}$. Тоді відносне значення похибки подання δ_1 деякого числа *N*, $m \le N < M$, дорівнює

$$\delta_1 = \frac{\Delta_1}{N} 100\ \%.$$

Мінімальне значення відносної похибки має місце з поданням максимального числа *M*:

$$\delta_M = \frac{\Delta_1}{M} 100\ \% = \frac{2^{-(n-k)}}{2^k - 2^{-(n-k)}} 100\ \% = \frac{1}{2^n - 1} 100\ \%,$$

а максимальне значення відносної похибки – з поданням мінімального числа *m*:

$$\delta_m = \frac{\Delta_1}{m} 100\ \% = \frac{2^{-(n-k)}}{2^{-(n-k)}} 100\ \% = 100\ \%.$$

Подання чисел у формі з плавучою комою. Форму з плавучою комою застосовують для розширення діапазону і зменшення відносної похибки подання чисел у МП.

Число *N* зображують у вигляді добутку. Першим множником є правильний дріб *a*, який називається *мантисою числа*. Другим множником є основа 2, піднесена до степеня *p*, який називається *порядком числа*:

$$N = \pm a \cdot 2^{\pm p}.$$

Мантиса і порядок є знаковими числами. Для зазначення знаків у розрядній сітці відводяться 2 додаткові розряди. З такою формою подання існують різні варіанти запису одного і того самого числа. Наприклад, число $11{,}01_2$ можна записати як $0{,}01101_2 \cdot 2^{11_2}$ або як $0{,}1101_2 \cdot 2^{10_2}$. Таким чином, кома у мантисі може зсуватися (плавати), а мантиса може набувати різних значень, менших від одиниці, при відповідних значеннях порядку. Форма подання числа, в якому старший розряд мантиси не дорівнює 0, називається *нормалізованою*. Усі інші форми подання є ненормалізованими.

У мікропроцесорних системах, у яких реалізовано подання чисел у формі з плавучою комою, числа зберігають у нормалізованому вигляді, при цьому більшу кількість розрядів використовують для зберігання дробової частини, внаслідок чого підвищується точність обчислень. Якщо після виконання арифметичних операцій (наприклад, віднімання) результат виявляється ненормалізованим, то перед занесенням числа в пам'ять виконують його нормалізацію, тобто зсув мантиси ліворуч на відповідну кількість розрядів, і зменшення порядку числа на відповідну кількість одиниць.

Записуючи двійкове число у формі з плавучою комою у $(n + 2)$-розрядній сітці, k комірок приділяють для розміщення мантиси, $n–k$ комірок – для розміщення порядку, а 2 розряди – для зазначення знаків (рис. 1.4).

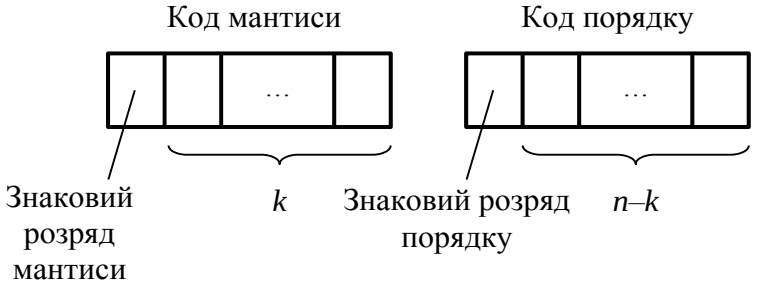


Рис. 1.4. Подання чисел у формі з плавучою комою

У нормалізованій формі значення мантиси завжди більші або дорівнюють $1/2$, але не перевищують одиниці. Вага молодшого розряду мантиси дорівнює 2^{-k}, а вага старшого розряду – 2^{-1}. Максимальне значення мантиси становить $1 - 2^{-k}$ і зі збільшенням k наближається до одиниці. Максимальне значення числа, що визначає порядок, дорівнює $2^{n-k} - 1$.

За максимальних значень мантиси і порядку значення M поданого числа є максимальним:

$$M = \left[1 - 2^{-k}\right] 2^{[2^{n-k}-1]}.$$

За мінімального значення мантиси і максимального за модулем від'ємного значення порядку значення m поданого числа є мінімальним:

$$m = 2^{-1} \cdot 2^{-[2^{n-k}-1]}.$$

Абсолютна похибка Δ_2 подання двійкових чисел визначається вагою молодшого розряду мантиси й істотно різна для великих (порядок максимальний) і малих чисел (порядок від'ємний і за модулем максимальний). Максимальне значення абсолютної похибки визначається таким чином:

$$\Delta_2 = 2^{-k} \cdot 2^{\pm[2^{n-k}-1]},$$

де знак «+» стосується похибки подання великих чисел, а знак «–» – малих чисел.

Відносну похибку подання двійкового числа N визначають як $\delta_2 = \frac{\Delta_2}{N} 100\,\%$. При поданні максимального числа значення відносної похибки є мінімальним:

$$\delta_M = \frac{\Delta_2}{M} 100\,\% = \frac{2^{-k}}{1 - 2^{-k}} 100\,\% = \frac{1}{2^k - 1} 100\,\%.$$

При поданні мінімального числа значення відносної похибки є максимальним:

$$\delta_m = \frac{\Delta_2}{m} 100\,\% = \frac{2^{-k}}{2^{-1}} 100\,\% = 2^{-(k-1)} 100\,\%.$$

Відносна похибка подання чисел у формі з плавучою комою незначно змінюється у всьому діапазоні і мала навіть для малих чисел.

Діапазон чисел у формі з плавучою комою істотно ширший, ніж із фіксованою комою. Наприклад, при 16-розрядній сітці (n = 16) діапазон подання чисел у формі з фіксованою комою визначається 16 двійковими розрядами. У формі з плавучою комою, у якій для розміщення мантиси відведено k = 10 розрядів з урахуванням знакових розрядів, діапазон $\frac{M}{m} = 2^{127}$ визначається 127 розрядами. Максимальна відносна похибка в першому випадку дорівнює $\delta_m = 100\,\%$, а в другому – $\delta_m = 2^{-9} 100\,\% = 0{,}1953\,\%$.

Форму подання двійкових чисел обирають залежно від типу задачі, потрібної швидкодії та точності виконання арифметичних операцій та діапазону зміни значень величин, якими оперує МП.

Контрольні запитання

1. Запишіть числа 1110110, 011_2 у нормалізованій та ненормалізованій формах із плавучою комою.
2. Розрахуйте максимальну абсолютну похибку подання числа у формі з фіксованою комою при n = 12, k = 9.
3. Обчисліть максимальну абсолютну похибку подання числа у формі з плавучою комою при n = 18, k = 5.
4. Розрахуйте максимальну відносну похибку подання числа у формі з плавучою комою при n = 11, k = 6.
5. Знайдіть мінімальну відносну похибку подання числа у формі з фіксованою комою при n = 9, k = 3.
6. Визначте діапазон подання чисел у формі з плавучою комою при n = 20, k = 8.

1.6. Елементи кодування інформації

Залежно від способу обробки бітів, розміщених у розрядній сітці, розрізняють два види кодів: паралельний, коли в кожний момент часу всі розряди сітки доступні для обробки, і послідовний, коли в кожний момент часу доступний один розряд сітки. Числа, подані паралельним кодом, доступні за один такт, а числа, подані послідовним кодом, – за n тактів, де n – розрядність сітки. Якщо розрядність числа перевищує довжину сітки, то його обробка ведеться по частинах.

Натуральним кодом називають подання числа як цілого беззнакового у двійковій системі числення. Діапазон чисел у натуральному коді для n-розрядної сітки становить від 0 до $2^n - 1$, тобто для 8-розрядної сітки діапазон чисел у натуральному коді – від 0 до 255. Наприклад, натуральний код числа 53_{10} у 8-розрядній сітці наведено на рис. 1.5.

Розряди	*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
Вага розрядів	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
53_{10}	0	0	1	1	0	1	0	1

Рис. 1.5. Натуральний код числа 53_{10} у 8-розрядній сітці

Для подання цілих знакових чисел використовують прямий, обернений і додатковий коди. Старший розряд сітки є знаковим. Значення цього розряду дорівнює 0 для додатних чисел і 1 – для від'ємних. В інших розрядах розміщується модуль числа. Додатні числа подаються однаково у всіх трьох кодах. Так, додатне число $+53_{10}$ має вигляд, поданий на рис. 1.5. Але оскільки старший розряд є знаковим, діапазон додатних чисел становить від 0 до $2^{n-1} - 1$. Наприклад, для 8-розрядної сітки діапазон додатних чисел – від 0 до +127. Подання від'ємних чисел залежить від використання того чи іншого коду.

Подання від'ємного числа *у прямому коді* здійснюється так. У старшому, знаковому, розряді розміщується 1, а в інших розрядах – модуль числа. Діапазон від'ємних чисел у прямому коді становить від 0 до $-(2^{n-1} - 1)$. Для прикладу на рис. 1.6 наведено прямий код числа -53_{10} у 8-розрядній сітці, для якої діапазон від'ємних чисел – від 0 до –127 = 11111111.

Розряди	*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
Вага розрядів	–	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Знак числа	0	1	1	0	1	0	1	
1								

Рис. 1.6. Прямий код числа -53_{10} у 8-розрядній сітці

Перевагами прямого коду є простота виконання арифметичних операцій та однаковий діапазон значень додатних і від'ємних чисел.

Недоліками прямого коду є те, що операції додавання і віднімання чисел з різними знаками потребують додаткових операцій для визначення більшого за модулем числа та знака результату. Крім того, наявність знака при поданні числа 0 (+0 = 000000 і –0 = 1000000) потребує виконання зайвих операцій.

Подання від'ємного числа *у оберненому коді* здійснюється обчисленням числа, яке доповнює додатне число з тим самим модулем до найбільшого беззнакового числа, яке може бути розташоване в даній розрядній сітці. Одержання оберненого коду від'ємного числа зводиться до порозрядного інвертування розрядів додатного числа, включаючи знаковий розряд. Діапазон від'ємних чисел у оберненому коді становить від 0 до $-(2^{n-1}-1)$. Як приклад на рис. 1.7 показано обернений код числа -53_{10} у 8-розрядній сітці, для якої діапазон від'ємних чисел – від 0 до -127_{10}, при цьому оберненим кодом найменшого числа -127_{10} є число 10000000_2.

Розряди	*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
Ваги розрядів	–	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	Знак числа	Модуль числа						
	1	1	0	0	1	0	1	0

Рис. 1.7. Обернений код числа -53_{10} у 8-розрядній сітці

Розглянемо віднімання двох чисел в оберненому коді на прикладі чисел $+163_{10}$ і $+53_{10}$. Подамо різницю у вигляді:

$$125_{10} - 53_{10} = 125_{10} + (255_{10} - 53_{10}) - 255_{10}. \qquad (1.3)$$

Вираз $(255_{10} - 53_{10})$ є оберненим кодом від'ємного числа -53_{10}, тобто доповненням до найбільшого числа $255_{10} = 11111111_2$, що можна розмістити у 8-розрядній сітці. З виразу (1.3) випливає, що операцію віднімання можна замінити операцією додавання в оберненому коді з подальшим коригуванням результату (відніманням 255_{10}). Виконання операції додавання числа 125_{10} та числа -53_{10}, поданого в оберненому коді, показано на рис. 1.8.

Рядок перенесень		1	1	1	1	1			
$+125_{10}$	+	0	1	1	1	1	1	0	1
$255_{10}-53_{10}$		1	1	0	0	1	0	1	0
$+71_{10}$		0	1	0	0	0	1	1	1

Рис. 1.8. Виконання операції додавання числа 125_{10} та числа -53_{10}, поданого в оберненому коді

Під час додавання виникає перенесення у неіснуючий дев'ятий розряд, тобто переповнення розрядної сітки. Це зменшує отриманий результат на 256_{10}. Оскільки згідно з виразом (1.3) результат треба зменшити на 255_{10}, для правильного результату потрібно до одержаної суми додати одиницю. Після такого коригування відповідь становитиме $+71_{10} + 1_{10} = +72_{10} = +01001000_2$, що відповідає виразу (1.3). Коригування результату досягається апаратними засобами – реалізацією додавання до молодшого біта значення перенесення, яке виходить за розрядну сітку. У цьому прикладі має місце перенесення одиниці в неіснуючий дев'ятий розряд. Нульове значення знакового розряду результату означає, що різниця чисел додатна.

Перевагами оберненого коду є простота операцій одержання та додавання чисел із різними знаками, недоліками – два подання нуля: +0 = 00000000 і –0 = 111111111, а також потреба в апаратній реалізації коригування результату.

Подання від'ємного числа *у додатковому коді* здійснюється обчисленням числа, яке доповнює додатне число з тим самим модулем до найбільшого беззнакового числа, з подальшим додаванням 1 до результату. Інакше кажучи, додатковий код отримують додаванням 1 до оберненого коду.

Додатковий код можна одержати за таким формальним правилом: цифри прямого коду додатного числа необхідно інвертувати послідовно зліва направо до останньої одиниці, не включаючи її. Останню праву одиницю і наступні за нею (праворуч) нулі слід залишити без зміни. Діапазон від'ємних чисел у додатковому коді становить від 0 до -2^{n-1}. Як приклад на рис. 1.9 показано додатковий код числа -53_{10} у 8-розрядній сітці, для якої діапазон від'ємних чисел – від 0 до -128_{10}, при цьому додатковим кодом найменшого числа -128_{10} є число 10000000_2.

Розряди	*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
Ваги розрядів	–	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	Знак числа	Модуль числа						
	1	1	0	0	1	0	1	1

Рис. 1.9. Додатковий код числа -53_{10} у 8-розрядній сітці

Розглянемо віднімання чисел у додатковому коді. Подамо різницю у вигляді:

$$125_{10} - 53_{10} = 125_{10} + (256_{10} - 53_{10}) - 256_{10}. \qquad (1.4)$$

Вираз $(256_{10} - 53_{10})$ є додатковим кодом від'ємного числа -53_{10}. З виразу (1.4) випливає, що операцію віднімання можна замінити операцією додавання у додатковому коді. Виконання операції додавання числа 125_{10} та числа -53_{10}, поданого у додатковому коді, ілюструє рис. 1.10.

Рядок перенесень	1	1	1	1	1	1	1	1	
$+125_{10}$		0	1	1	1	1	1	0	1
+									
$256_{10} - 53_{10}$		1	1	0	0	1	0	1	1
$+72_{10}$		0	1	0	0	1	0	0	0

Рис. 1.10. Виконання операції додавання числа 125_{10} та числа -53_{10}, поданого у додатковому коді

Унаслідок додавання відбулося переповнення, тобто перенесення у неіснуючий дев'ятий розряд, що зменшило результат на 256_{10}. Отже, відповідно до формули (1.4) отримана сума і буде остаточним результатом.

Перевагами додаткового коду є простота операцій одержання та додавання чисел із різними знаками, а також те, що нуль має єдине подання: 0 = 00000000. Завдяки цим перевагам додатковий код використовують найчастіше.

Кодування, захищене від завад. Унаслідок дій завад під час передачі, обробки і збереження двійкових кодів у мікропроцесорних системах можуть статися помилки, наприклад прийом 1 замість 0 або навпаки. Це може призвести до неправильного результату роботи мікропроцесорної системи.

Неправильний прийом, обробка або збереження одного або декількох бітів називається *помилкою*. Кількість неправильно прийнятих, оброблених або збережених бітів називається *кратністю помилки*. Для визначення діапазону послідовності бітів, який містить неправильні біти, використовується поняття *пакет помилок*. Довжина пакета помилок визначається кількістю символів між першим і останнім неправильним бітом, включаючи ці біти. При цьому на кількість правильних бітів, розміщених між неправильними, накладається обмеження.

Одним із найбільш ефективних шляхів захисту інформації у мікропроцесорних системах є кодування, стійке до завад, яке здійснюється введенням у кодові комбінації додаткових бітів, призначених або для виявлення і виправлення помилок, або тільки для виявлення помилок. Відповідно до цього коди, стійкі до завад, поділяють на коректувальні коди, які виявляють і виправляють помилки, та коди, які тільки виявляють помилки.

Можливість виявлення помилок за наявності додаткових бітів обумовлена тим, що для передачі інформації використовуються не всі можливі комбінації в кількості $N = 2^n$, а лише частина з них – $N_0 < N$. Комбінації з безпомилковою інформацією є дозволеними, інші $N - N_0$ комбінацій є забороненими. Поява заборонених комбінацій розглядається як помилка. Можливі такі помилки, за яких одна дозволена комбінація переходить у іншу. У цьому разі помилки не виявляються. Загальна кількість

можливих помилок визначається добутком $N \cdot N_0$. З цієї кількості помилок буде виявлено $N_0(N - N_0)$. Відношення кількості виявлених до загальної кількості можливих помилок називають *коефіцієнтом виявлення коду*:

$$K_{\text{вияв}} = \frac{N_0(N - N_0)}{N \cdot N_0} = 1 - \frac{N_0}{N}.$$

Розглянемо можливість виправлення помилок під час використання коду з додатковими бітами. Для забезпечення виправлення помилок множина кодових комбінацій N розбивається на N_0 підмножин, що не перетинаються. Кожній з цих підмножин відповідає одна з дозволених комбінацій. При цьому виправляються помилки, що не переводять передану комбінацію в інші підмножини. Помилка буде виправлена у $N - N_0$ випадках, тоді як загальна кількість помилок – $N \cdot N_0$. Відношення кількості виправлених помилок до кількості виявлених називають *коефіцієнтом виправлення коду*:

$$K_{\text{випр}} = \frac{N - N_0}{N_0(N - N_0)} = \frac{1}{N_0}.$$

Ступінь відмінності двох кодових комбінацій характеризується кодовою відстанню d, тобто кількістю бітів, які є різними для двох комбінацій. Кодова відстань може набувати значень від 1 до n, де n – довжина кодової комбінації. Кодову відстань можна визначити, обчисливши суму двох комбінацій за модулем 2 і підрахувавши кількість одиниць у цій сумі. Якщо $d = 1$, то всі кодові комбінації є дозволеними, а виявлення і виправлення помилок неможливе. При $d = 2$ одноразові помилки переводять дозволені комбінації в заборонені, тому такий код може виявляти всі одноразові помилки, а також помилки непарної кратності (помилки одночасно у трьох, п’яти, семи і так далі розрядах). Помилки парної кратності (помилки одночасно у двох, чотирьох, шести і так далі розрядах) таким кодом не виявляються. Якщо необхідно виявляти помилки кратності r, то мінімальна кодова відстань $d_{\min}$ між комбінаціями коду має бути хоча б на одиницю більше від кратності помилки $d_{\min} \geq r + 1$.

Коректувальна здатність коду забезпечується введенням k додаткових перевірних бітів. Тоді загальна довжина кодової комбінації буде $n + k$, а загальна кількість можливих комбінацій $N = 2^{n+k}$, що дозволяє визначити необхідну кількість перевірних бітів для побудови кодів, стійких до завад.

Одним з поширених кодів, стійких до завад, які використовуються у мікропроцесорній техніці, є *код з контролем на парність*. У цьому коді до інформаційних бітів праворуч додається один контрольний біт ($k = 1$). Якщо кількість одиниць в інформаційних бітах є парною, то значення контрольного біта дорівнює 0, в протилежному випадку – 1. Отже, у будь-

якому випадку кількість одиниць у повній послідовності $n + 1$ біт є парною. Якщо при перевірці після передачі кількість одиниць є непарною, то це означає, що має місце помилка. Код із контролем на парність дозволяє виявляти всі помилки непарної кратності і не дозволяє виявляти помилки парної кратності.

Приклад 1.22. Знайти значення контрольного біта k в кодах із контролем на парність: 1) 11000; 2) 11100.

Значення контрольного біта визначимо з умови парності кількості одиниць у послідовності із заданого коду і контрольного біта. Таким чином, у прикладі

n	k
11000	0
11100	1

Іншим поширеним кодом є *код Хеммінга*, що виявляє і виправляє одноразові помилки. Кожній з $2^n - 1$ ненульових комбінацій n-розрядного числа відповідає комбінація з $n + k$ бітів. Значення перевірних бітів отримуються в результаті додавання за модулем 2 значень бітів у деяких визначених інформаційних розрядах. Із загальної кількості $2^{n+k} - 1$ можливих помилок код Хеммінга може виявити та виправити $2^k - 1$ помилок. Припустімо, що треба передати або обробити 15 різних послідовностей з одиниць та нулів. Без кодування для цього достатньо чотирьох інформаційних бітів ($n = 4$). Потрібну кількість додаткових перевірних бітів k обчислюють з формули $2^k - 1 = n + k$. Звідки визначають кількість перевірних розрядів та кількість одноразових помилок, які можуть бути виявлені та виправлені. У цьому випадку кількість додаткових розрядів $k = 3$, а кількість одноразових помилок – $2^k - 1 = 7$.

Контрольні біти k_i розташовують у послідовності інформаційних бітів u_j на позиціях із номерами 2^{i-1} (табл. 1.17).

Таблиця 1.17. **Послідовність контрольних та інформаційних бітів у коді Хеммінга**

1	2	3	4	5	6	7	8	9	10	…	15
k_1	k_2	u_1	k_3	u_2	u_3	u_4	k_4	u_5	u_6	…	u_{11}

Значення перевірних бітів k_i обчислюється додаванням за модулем 2 значень бітів, у двійковому виразі номерів яких наявна одиниця в i-му розряді.

Для обчислення значення k_1 потрібно додати за модулем 2 значення бітів із непарними номерами. Це біти на позиціях з номерами 3, 5, 7, 9, 11, 13, 15:

$$k_1 = u_1 \oplus u_2 \oplus u_4 \oplus u_5 \oplus u_7 \oplus u_9 \oplus u_{11} .$$

Для визначення k_2 треба додати за модулем 2 біти, у двійковому виразі номерів яких наявна одиниця у другому розряді, тобто 3, 6, 7, 10, 11, 14, 15:

$$k_2 = u_1 \oplus u_3 \oplus u_4 \oplus u_6 \oplus u_7 \oplus u_{10} \oplus u_{11}.$$

Контрольний біт k_3 визначається додаванням за модулем 2 бітів, у двійковому виразі номерів яких наявна одиниця у третьому розряді, тобто 5, 6, 7, 12, 13, 14, 15:

$$k_3 = u_2 \oplus u_3 \oplus u_4 \oplus u_8 \oplus u_9 \oplus u_{10} \oplus u_{11}.$$

Аналогічно визначається k_4 через біти з номерами: 9, 10, 11, 12, 13, 14, 15:

$$k_4 = u_5 \oplus u_6 \oplus u_7 \oplus u_8 \oplus u_9 \oplus u_{10} \oplus u_{11}.$$

Визначення та виправлення помилок здійснюється k перевірками. При кожній перевірці додаються за модулем 2 біти послідовності інформаційних та контрольних бітів, двійкові номери яких мають одиницю в першому, другому, третьому і так далі розрядах. Якщо під час передавання не було збою, то результати перевірок дорівнюють нулю. Якщо збій відбувся, то хоча б одна перевірка не дорівнює нулю. У цьому випадку треба сформувати двійковий код з результатів перевірок, який вкаже на розряд, де відбувся збій. Молодший розряд коду результату перевірок формує перша перевірка, старший – остання. Інверсія біта в розряді з одержаним номером виправить помилку.

Приклад 1.23. Сформувати код Хеммінга, що виявляє і виправляє одноразову помилку у послідовності:

1	1	0	0
u_1	u_2	u_3	u_4

Для формування коду Хеммінга необхідно $k = 3$ контрольні біти, які мають займати позиції з номерами 1, 2 і 4.

Послідовність з контрольними символами має вигляд:

1	2	3	4	5	6	7
k_1	k_2	u_1	k_3	u_2	u_3	u_4

Значення контрольних бітів визначимо як:

$$k_1 = u_1 \oplus u_2 \oplus u_4 = 1 \oplus 1 \oplus 0 = 0;$$
$$k_2 = u_1 \oplus u_3 \oplus u_4 = 1 \oplus 0 \oplus 0 = 1;$$
$$k_3 = u_2 \oplus u_3 \oplus u_4 = 1 \oplus 0 \oplus 0 = 1.$$

Отже, послідовність, закодована за кодом Хеммінга, буде такою: 0111100.

Нехай після передачі відбувся збій в одному розряді і прийнята послідовність 0110100.

Перша та друга перевірки дадуть значення 0, а третя – 1:

1. $k_1 \oplus u_1 \oplus u_2 \oplus u_4 = 0 \oplus 1 \oplus 1 \oplus 0 = 0.$
2. $k_2 \oplus u_1 \oplus u_3 \oplus u_4 = 1 \oplus 1 \oplus 0 \oplus 0 = 0.$
3. $k_3 \oplus u_2 \oplus u_3 \oplus u_4 = 0 \oplus 1 \oplus 0 \oplus 0 = 1.$

Код 100, який створюють результати перевірок, вказує, що відбувся збій у четвертому розряді. Якщо проінвертувати четвертий розряд, то одержимо виправлену послідовність 0111100.

Контрольні запитання

1. Подайте число -75_{10}: а) у прямому коді; б) в оберненому коді; в) у додатковому коді.
2. Поясніть, чому при додаванні чисел в оберненому коді потрібне коригування результату, а при додаванні в додатковому не потрібне.
3. Укажіть діапазон значень додатних і від'ємних чисел у прямому, оберненому та додатковому кодах при $n = 16$.
4. Назвіть переваги і недоліки прямого, оберненого і додаткового кодів.
5. Поясніть зміст понять коефіцієнта виявлення помилок, коефіцієнта виправлення помилок та кодової відстані.
6. Знайдіть значення контрольного біта в кодах з контролем на парність у кодових комбінаціях: а) 0010 1101; б) 0110 1110; в) 1110 0000.
7. Знайдіть коди Хеммінга, якщо інформаційні послідовності дорівнюють: а) 0010 ; б) 0110; в) 1110 .
8. Доведіть, що утворені в п. 7 коди Хеммінга коректують одноразові помилки.

Розділ 2

ЗАГАЛЬНІ ПРИНЦИПИ ПОБУДОВИ МІКРОПРОЦЕСОРНИХ СИСТЕМ

2.1. Основні поняття і визначення

Мікропроцесор – це пристрій, який здійснює прийом, обробку і видачу інформації. Конструктивно МП містить одну або декілька інтегральних схем і виконує дії за програмою, записаною в пам'яті.

Мікропроцесорна система – обчислювальна, контрольно-вимірювальна або керувальна система, в якій основним пристроєм обробки інформації є МП. Мікропроцесорна система будується з набору мікропроцесорних ВІС.

Мультимікропроцесорна (або мультипроцесорна) система – система, яка утворюється об'єднанням деякої кількості універсальних або спеціалізованих МП, завдяки чому забезпечується паралельна обробка інформації і розподілене керування.

Мікропроцесорний комплект (МПК) – сукупність інтегральних схем, сумісних за електричними, інформаційними та конструктивними параметрами і призначених для побудови електронно-обчислювальної апаратури та мікропроцесорних систем керування. Зазвичай МПК містить: ВІС МП (один чи кілька корпусів інтегральних схем); ВІС оперативних запам'ятовувальних пристроїв (ОЗП); ВІС постійних запам'ятовувальних пристроїв (ПЗП); інтерфейси або контролери зовнішніх пристроїв; службові ВІС (тактовий генератор, регістри, шинні формувачі, контролери шин, арбітри шин).

Мікропроцесори та МПК класифікують за такими ознаками: призначенням; кількістю ВІС; способом керування; за типом архітектури; за типом системи команд.

За призначенням МП поділяють на універсальні та спеціалізовані.

Універсальними мікропроцесорами є МП загального призначення, які розв'язують широкий клас задач обчислення, обробки та керування.

Спеціалізовані мікропроцесори призначені для розв'язання задач лише певного класу. До спеціалізованих МП належать: сигнальні; медійні та мультимедійні; трансп'ютери.

Сигнальні процесори призначені для цифрової обробки сигналів у реальному масштабі часу (наприклад, фільтрація сигналів, обчислення згортки, обчислення кореляційної функції, підсилення, обмеження і трансформація сигналу, пряме та зворотне перетворення Фур'є). До сигнальних процесорів належать процесори фірм *Texas Instruments – TMS*320*C*80, *Analog Devices – ADSP*2106*x*, *Motorola – DSP*560*xx* та *DSP*9600*x*.

Медійні і *мультимедійні процесори* призначені для обробки аудіосигналів, графічної інформації, відеозображень, а також для розв'язування ряду задач у мультимедіакомп'ютерах, іграшкових приставках, побутовій техніці. До медійних і мультимедійних процесорів належать процесори фірм *MicroUnity – Mediaprocessor*, *Philips – Trimedia*, *Cromatic Reserch – Mpact Media Engine*, *Nvidia – NV*1, *Cyrix – MediaGX*.

Трансп'ютери призначені для масових паралельних обчислень і роботи у мультипроцесорних системах. Для них характерним є наявність внутрішньої пам'яті та вбудованого міжпроцесорного інтерфейсу, тобто каналів зв'язку з іншими ВІС МП. До трансп'ютерів належать процесори фірм *Inmos – T*-2, *T*-4, *T*-8, *T*9000.

За кількістю ВІС у МПК розрізняють багатокристальні МПК і однокристальні мікроконтролери. До багатокристальних комплектів відносять МПК з однокристальними і секційними МП.

Однокристальний мікропроцесор є конструктивно завершеним виробом у вигляді однієї ВІС. Інша назва однокристальних МП – мікропроцесори із фіксованою розрядністю даних. До цього типу належать процесори фірм *Intel – Pentium* (*P*5, *P*6, *P*7), *AMD – K*5, *K*6, *Cyrix* – 6*x*86, *Diqital Equipment – Alpha* 21064, 21164*A*, *Silicon Graphics – MIPS R*10000, *Motorola – Power PC* 603, 604, 620, *Hewlett Packard – PA*-8000, *Sun Microsystems – Ultra SPARC* II.

У секційних мікропроцесорах в одній ВІС реалізується лише деяка функціональна частина (секція) процесора. Інша назва секційних МП – розрядно-модульні мікропроцесори або мікропроцесори з нарощенням розрядності. Секційність ВІС МП зумовлює значну гнучкість МПС, можливість нарощення розрядності даних, створення специфічних технологічних команд із набору мікрокоманд. До секційних належать МП серій *К*589, *К*1804.

Однокристальний мікроконтролер являє собою пристрій, виконаний конструктивно в одному корпусі ВІС і містить основні складові частини МПК. До таких мікроконтролерів належать ОМК фірм *Intel – MCS*-196/296, *MicroChip – PIC*17*C*4*x*, *PIC*17C75*x*, *Mitsubishi Electric – M*3820, *Motorola – MC*33035, *MC*33039.

За способом керування розрізняють МП зі схемним та МП з мікропрограмним керуванням.

Мікропроцесори зі схемним керуванням мають фіксований набір команд, розроблений фірмою-виробником, який не може змінювати

користувач. У мікропроцесорах з мікропрограмним керуванням систему команд розробляють при проектуванні конкретного МПК на базі набору найпростіших мікрокоманд з урахуванням класу задач, для розв'язання яких призначений МПК.

За типом архітектури, або принципом побудови розрізняють МП з фоннеймановською архітектурою і МП з гарвардською архітектурою.

За типом системи команд розрізняють *CISC* (*Complete Instruction Set Computing*) – процесори з повним набором команд, і *RISC* (*Reduced Instruction Set Computing*) – процесори зі зменшеним набором команд.

Слід зазначити, що багато МПК підпадають під різні класифікаційні ознаки, оскільки здатні вирішувати задачі різних класів. Так, існують універсальні МП з мультимедійним розширенням наборів команд, наприклад, *Pentium MMX*, *Pentium* II, *Cyrix* 6*x*86*MX*, *AMD K*6, *Ultra SPARC*. У *CISC*-процесорах *Pentium PRO* реалізовано ядро з *RISC*-архітектурою.

Контрольні запитання

1. Назвіть складові частини МПК.
2. За якими класифікаційними ознаками поділяють МП і МПК?
3. На які задачі орієнтовано спеціалізовані МП?
4. Які переваги і недоліки мають секційні МП порівняно з однокристальними?

2.2. Організація шин

Шина – це інформаційний канал, який об'єднує всі функціональні блоки МПС і забезпечує обмін даними у вигляді двійкових чисел. Конструктивно шина являє собою *n* провідників та один спільний провідник (земля). Дані про шину передаються у вигляді слів, що є групою бітів.

У паралельній шині *n* бітів передаються по окремих лініях одночасно, у послідовній шині – по єдиній лінії послідовно у часі. Паралельні шини виконують у вигляді плоского кабелю, а послідовні – у вигляді коаксіального або волоконно-оптичного кабелю. Коаксіальний кабель використовують для передачі даних на відстань до 100 метрів, узгоджуючи передавальні та приймальні каскади із хвильовим опором лінії. Волоконно-оптичний кабель використовують для передачі на більші відстані.

Усі основні блоки МПС з'єднують з єдиною паралельною шиною, яка називається *системною шиною SB* (*System Bus*). Системна шина містить три шини: адреси, даних і керування.

Шина адреси AB (*Address Bus*) є однонапрямленою. Вона призначена для передавання адреси комірки пам'яті або пристрою введення-виведення (ПВВ). Напрям передавання по шині адреси – від МП до зовнішніх

пристроїв. Варіанти умовних позначень однонапрямленої паралельної шини показано на рис. 2.1, на якому стрілка вказує напрям передавання.

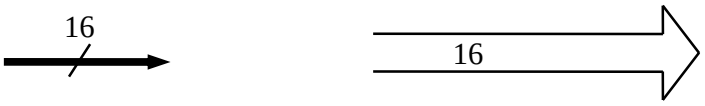


Рис. 2.1. Варіанти умовних позначень однонапрямленої паралельної 16-розрядної шини

Число 16 на рис. 2.1 позначає розрядність шини. Зазначимо, що допускається позначення шин і без наведення розрядності.

Шина даних DB (*Data Bus*) є двонапрямленою. Вона призначена для передавання даних між блоками МПС. Інформація по одних і тих самих лініях *DB* може передаватися у двох напрямах – як до МП, так і від нього. Варіанти умовних позначень двонапрямленої шини показано на рис. 2.2.

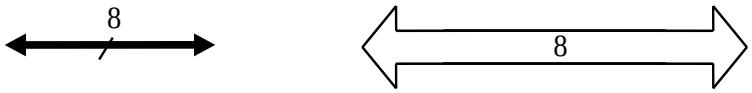


Рис. 2.2. Варіанти умовних позначень двонапрямленої паралельної 8-розрядної шини

Шина керування CB (*Control Bus*) призначена для передавання керувальних сигналів. Хоча напрям керувальних сигналів може бути різним, однак шина керування не є двонапрямленою, оскільки для сигналів різного напряму використовуються окремі лінії. Позначається ця шина так само, як і однонапрямлена (див. рис. 2.1).

Як приклад на рис. 2.3 показано структурну схему передавання інформації між *m* регістрами по внутрішній *n*-розрядній шині даних з урахуванням прийнятих позначень, а на рис. 2.4 – розширену структурну схему.

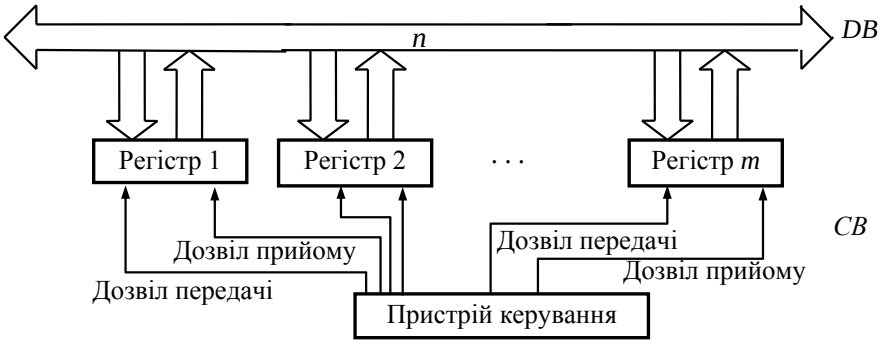


Рис. 2.3. Структурна схема передавання інформації між *m* регістрами по внутрішній *n*-розрядній шині даних

Дані по шині з *n* ліній передаються в режимі розподілу часу. Пристрій керування в кожний момент часу визначає адресу регістра, який передає інформацію, та регістра, який приймає інформацію. Для цього пристрій керування генерує сигнали *Дозвіл передачі* і *Дозвіл прийому*, що передаються по лініях шини керування *CB*. Лінії шини і сигнали керування мають назви *Дозвіл передачі* і *Дозвіл прийому*. У кожний момент часу передавати інформацію в шину може тільки один регістр. Це означає, що у

разі надходження сигналу *Дозвіл передачі* до шини приєднується тільки один модуль (рис. 2.4).

Вхідні лінії регістрів з'єднані безпосередньо з відповідними лініями шини. Тому під час подання сигналу *Дозвіл прийому*, який надходить по окремій лінії для кожного регістра, дані передаються по шині у відповідний регістр. Вихідні лінії регістрів з'єднуються з відповідними лініями шини через ключі *S*, що допускають монтажну логіку. Сигнал *Дозвіл передачі* надходить на ключі від пристрою керування по окремій для кожного регістра лінії.

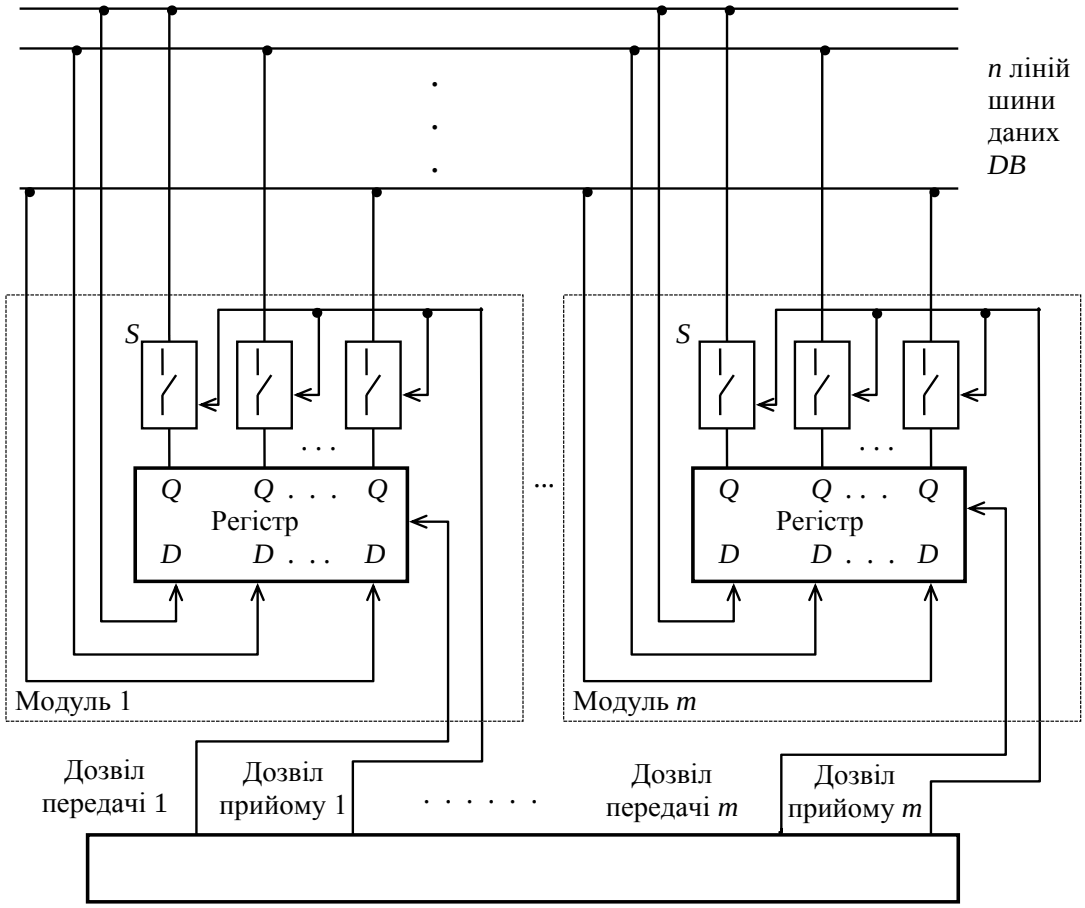


Рис. 2.4. Розширена структурна схема передавання інформації між *m* регістрами по внутрішній *n*-розрядній шині даних

Модуль, який братиме участь в обміні інформацією, визначається за одним з таких способів: 1) через відповідні лінії шини керування, окремі для кожного модуля (див. рис. 2.4); 2) за допомогою k ($k = \log_2 m$) ліній шини адреси, по яких передається ідентифікаційний код, що ставиться у відповідність кожному модулю й однозначно його визначає; 3) використанням одних і тих самих ліній шини даних для передавання адрес і даних.

Дані по шині можуть передаватися у двох режимах: синхронному й асинхронному. У синхронному режимі пристрій керування визначає модулі, що беруть участь в обміні інформацією, синхронізує роботу модулів та керує процесом обміну, виробляючи відповідні сигнали керування і синхронізації.

В асинхронному режимі модулі, готові до обміну, ініціюють процес передавання та прийняття інформації, виробляючи відповідні сигнали готовності.

Контрольні запитання

1. Яке призначення та які складові частини системної шини?
2. Вкажіть принципи передачі інформації по шинах: адреси; даних; керування.
3. Як передається інформація по паралельним та послідовним шинам?

2.3. Принципи побудови мікропроцесорних систем

В основу побудови МПС систем покладено три принципи: магістральності; модульності; мікропрограмного керування.

Принцип магістральності визначає характер зв'язків між функціональними блоками МПС – усі блоки з'єднуються з єдиною системною шиною.

Принцип модульності полягає в тому, що система будується на основі обмеженої кількості типів конструктивно і функціонально завершених модулів. Кожний модуль МПС системи має вхід керування третім (високоімпедансним) станом. Цей вхід називається $\overline{CS}$ (*Chip Select*) – вибір кристала або $\overline{OE}$ (*Output Enable*) – дозвіл виходу.

Дію сигналу $\overline{CS}$ для тригера показано на рис. 2.5. Вихідний сигнал тригера Q з'явиться на виводі лише при активному (у цьому випадку – нульовому) рівні сигналу $\overline{CS}$. Якщо $\overline{CS}$ = 1, тригер переводиться у високоімпедансний стан. Вихід тригера є тристабільним, тобто може знаходитися в одному з трьох станів: логічної одиниці, логічного нуля або у високоімпедансному. У кожний момент часу до системної шини МПС приєднано лише два модулі – той, що приймає, і той, що передає інформацію. Інші знаходяться у високоімпедансному стані.

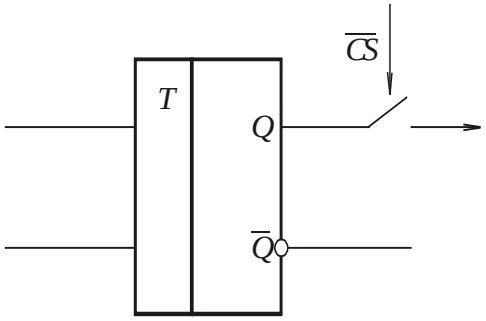


Рис. 2.5. Дія сигналу $\overline{CS}$ для тригера

Принципи магістральності та модульності дозволяють нарощувати керувальні й обчислювальні можливості МП через приєднання інших модулів.

Принцип мікропрограмного керування полягає у можливості здійснення елементарних операцій – мікрокоманд (зсуву, пересилання інформації, логічних операцій). Певною комбінацією мікрокоманд можна створити набір команд, який максимально відповідатиме призначенню системи, тобто створити технологічну мову. У секційних процесорах набір мікрокоманд можна змінити, використовуючи інші мікросхеми пам'яті мікрокоманд.

Узагальнену структурну схему МПС показано на рис. 2.6. До складу МПС входять: центральний процесор (ЦП), ПЗП, ОЗП; система переривань, таймер, ПВВ. Пристрої введення-виведення приєднані до системної шини через інтерфейси введення-виведення.

Постійний та оперативний запам'ятовувальні пристрої складають систему пам'яті, призначену для збереження інформації у вигляді двійкових чисел. Постійний запам'ятовувальний пристрій призначений для збереження програм керування таблиць, констант, ОЗП – для збереження проміжних результатів обчислень. Пам'ять організовано у вигляді масиву комірок, кожна з яких має свою адресу і містить байт або слово. Байтом називається група із 8 біт, а слово може мати будь-яку довжину в бітах. Найчастіше під словом розуміють двійкове число довжиною 2 байт.

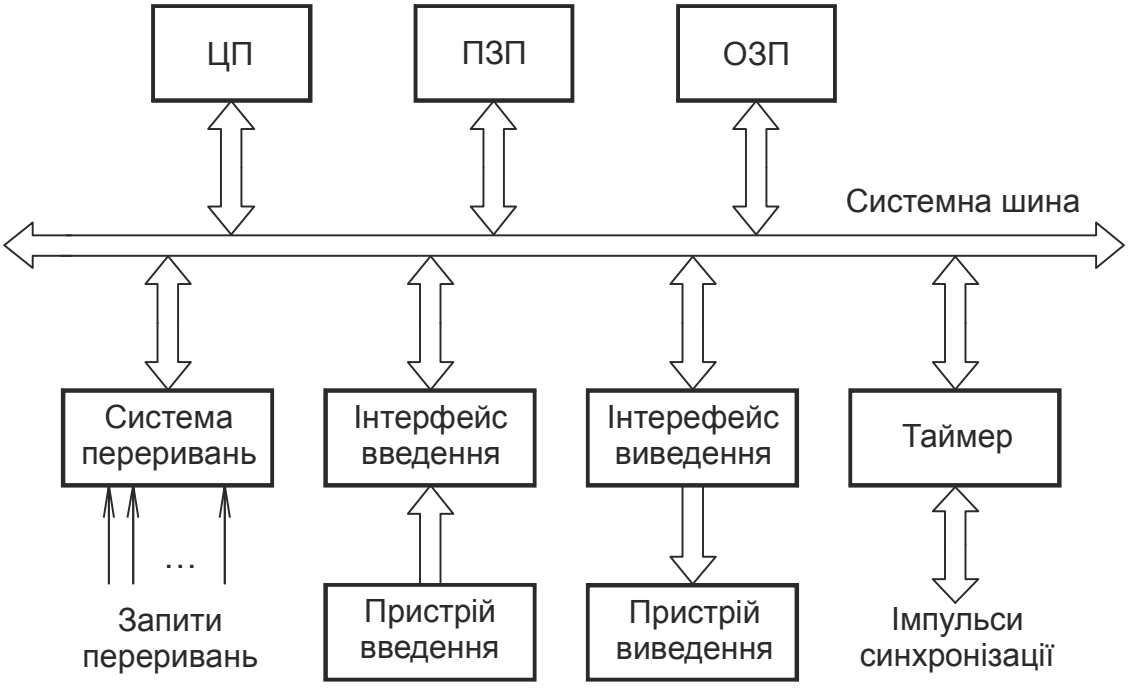


Рис. 2.6. Узагальнена структурна схема мікропроцесорної системи керування

Для звернення до комірки пам'яті треба видати її адресу на шину адреси. На рис. 2.7 зображено структуру пам'яті з 8 однобайтових комірок, де кожній адресі відповідає певний вміст комірки. Так, комірка з адресою 000 має вміст $01011111_2 = 5F_{16}$.

Адреса	Дані
000	01011111
001	00010011
010	01110111
011	00001100
100	00000000
101	11111111
110	10101010
111	11110000

Рис. 2.7. Структура пам’яті з 8 однобайтових комірок

Сегментом називається область пам’яті, яка складається з суміжних комірок. У МП *i*8086 сегмент завжди починається з адреси, кратної 16, і займає до 64 кбайт. Існують три основні сегменти: кодів, даних, стека.

Сегмент кодів містить коди команд, які адресуються сегментним регістром кодів *CS* та регістром – вказівником команд *IP*. Регістр *CS* визначає початкову адресу сегмента кодів, а регістр *IP* – зміщення в сегменті (відстань від початку сегмента до комірки, в якій знаходиться адреса команди).

Сегмент даних містить дані, константи та робочі області, потрібні для виконання програми. Регістр *DS* має початкову адресу сегмента даних, а зміщення в сегменті задається в команді.

Сегмент стека містить адреси повернення з підпрограм та дані. Регістр *SS* має початкову адресу сегмента стека, а регістр *SP* – зміщення в сегменті.

Деякі операції використовують додатковий сегмент даних, початкова адреса якого задається регістром *ES*, а зміщення в сегменті визначається командою.

Двобайтове зміщення (16 біт) може знаходитися в межах від 0000_{16} до $FFFF_{16}$. Для звернення до будь-якої адреси у програмі виконується додавання адрес, які знаходяться в регістрі сегмента та зміщення. Наприклад, перший байт у сегменті кодів має зміщення нуль, другий байт – одиницю і так далі до $FFFF_{16}$.

Конкретна адреса команди (для сегмента кодів), комірки пам’яті (для сегмента даних та додаткового сегмента) або комірки стека (для сегмента стека) визначається результатом додавання адреси сегмента, яка міститься у відповідному сегментному регістрі, та зміщення.

Модуль центрального процесора обробляє дані та керує всіма іншими модулями системи. Центральний процесор, крім ВІС МП, містить схеми синхронізації та інтерфейсу із системною шиною. Він вибирає коди команд з пам’яті, дешифрує їх і виконує. Протягом часу виконання команди – командного циклу ЦП виконує такі дії:

– виставляє адресу команди на шину адреси *AB*;
– отримує код команди з пам’яті та дешифрує його;
– обчислює адреси операнда і зчитує дані;
– виконує операцію, визначену командою;
– сприймає зовнішні керувальні сигнали, (наприклад, запити переривань);
– генерує сигнали стану і керування, потрібні для роботи пам’яті та ПВВ.

Пристрої введення-виведення або зовнішні пристрої – це пристрої, призначені для введення інформації у МП або виведення інформації з нього.

Прикладами ПВВ є дисплеї, друкувальні пристрої, клавіатура, цифро-аналоговий та аналого-цифровий пристрої, реле, комутатори. Для з'єднання ПВВ із системною шиною їх сигнали мають відповідати певним стандартам. Це досягається за допомогою інтерфейсів введення-виведення.

Інтерфейси введення-виведення виконують функцію узгодження сигналів ПВВ із сигналами системної шини МП. Їх називають також контролерами або адаптерами. Мікропроцесор звертається до інтерфейсів за допомогою спеціальних команд введення-виведення. При цьому МП виставляє на шину адреси *AB* адресу інтерфейсу, а по шині даних *DB* зчитує дані з пристрою введення або записує у пристрій виведення. На рис. 2.4 показано один інтерфейс введення і один інтерфейс виведення.

Система переривань дозволяє МПС реагувати на зовнішні сигнали – запити переривань, джерелами яких можуть бути: сигнали готовності від зовнішніх пристроїв, сигнали від генераторів, сигнали з виходів датчиків. Із появою запиту переривання ЦП перериває основну програму і переходить до виконання підпрограми обслуговування запиту переривання. Для побудови системи переривань МПК містять ВІС спеціальних програмовних контролерів переривань.

Таймер призначений для реалізації функцій, пов'язаних з відліком часу. Після того, як МП завантажує в таймер число, яке задає частоту, затримку або коефіцієнт ділення, таймер реалізує потрібну функцію самостійно.

Контрольні запитання

1. Назвіть принципи побудови МПС і охарактеризуйте їх.
2. Наведіть типову структуру МПС і поясніть призначення функціональних модулів.
3. Поясніть призначення входу керування третім станом.

2.4. Архітектура мікропроцесорів

Поняття архітектури мікропроцесора визначає його складові частини, а також зв'язки та взаємодію між ними. Архітектура містить: 1) структурну схему самого МП; 2) програмну модель МП (опис функцій регістрів); 3) інформацію про організацію пам'яті (ємність пам'яті та способи її адресації); 4) опис організації процедур введення-виведення.

Існують два основні типи архітектури – фоннеймановська та гарвардська. *Фоннеймановську архітектуру* (рис. 2.8, *а*) запропонував 1945 року американський математик Джо фон Нейман.

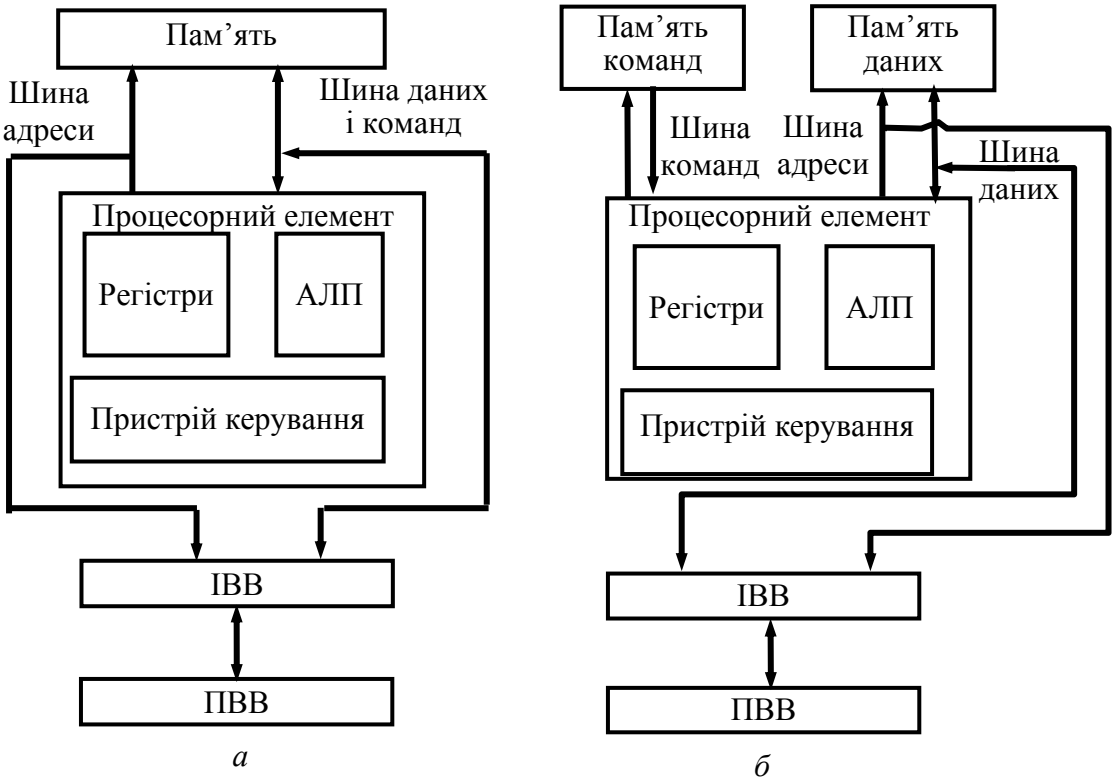


Рис. 2.8. Основні типи архітектури: *а* – фоннеймановська; *б* – гарвардська

Особливістю цієї архітектури є те, що програма і дані знаходяться у спільній пам’яті, доступ до якої здійснюється по одній шині даних і команд.

Гарвардську архітектуру вперше реалізовано 1944 року в релейній обчислювальній машині Гарвардського університету (США). Особливістю цієї архітектури є те, що пам’ять даних і пам’ять програм розділені та мають окремі шину даних і шину команд (рис. 2.8, *б*), що дозволяє підвищити швидкодію МП системи.

Структурні схеми обох архітектур містять: процесорний елемент, пам’ять, інтерфейси введення-виведення (ІВВ) і ПВВ. Пам’ять і ІВВ для різних типів МП можуть бути як внутрішніми, тобто розміщуватися на тому ж кристалі, що і процесорний елемент, так і зовнішніми. Процесорний елемент містить регістри, арифметико-логічний пристрій (АЛП), пристрій керування і виконує функції обробки даних та керування процесами обміну інформацією. Пам’ять забезпечує зберігання кодів команд програми і даних. Інтерфейси призначені для зв’язку з ПВВ (наприклад, з клавіатурою, дисплеєм, друкувальними пристроями, датчиками інформації). Усі елементи структурної схеми з’єднані за допомогою шин.

Розширену структурну схему з процесором фоннеймановської архітектури показано на рис. 2.9.

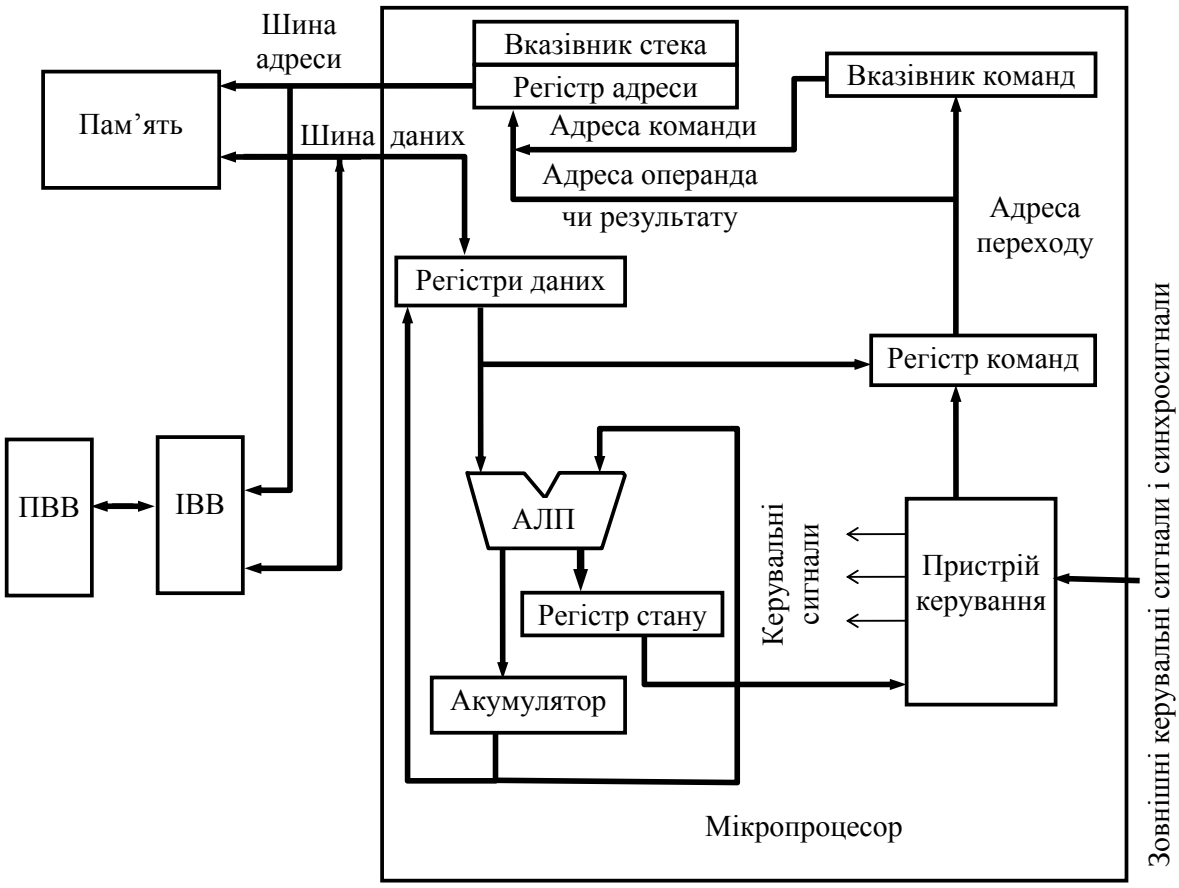


Рис. 2.9. Структурна схема з процесором фоннеймановської архітектури

Схема процесора містить пристрій керування, АЛП і регістри: адреси, даних, команд, стану, а також акумулятор, лічильник команд та вказівник стека.

Пристрій керування відповідно до кодів команд та зовнішніх керувальних сигналів і сигналів синхронізації виробляє керувальні сигнали для всіх блоків структурної схеми МП, а також керує обміном інформацією між МП, пам’яттю і ПВВ. Пристрій керування реалізує такі функції:

1. Функція початкового встановлення МП. Зовнішній сигнал початкового встановлення процесора *RESET* формується при ввімкненні джерела живлення МП або при натисканні кнопки *RESET*. У разі появи цього сигналу пристрій керування забезпечує завантаження нульового значення у програмний лічильник, що ініціює вибирання з пам’яті байта команди з нульовою адресою. Наприкінці вибирання вміст лічильника команд збільшується на одиницю і вибирається байт команд з наступною адресою. Таким чином виконується вся записана у пам’яті програма.

2. Функція синхронізації. Згідно із зовнішніми керувальними сигналами і сигналами синхросигналізації пристрій керування синхронізує роботу всіх блоків МП.

3. Функція переривань. Із надходженням сигналу переривання пристрій керування ініціює роботу підпрограми обробки відповідного пере-

ривання. Потреба у реалізації функцій переривань виникає тоді, коли під час виконання основної програми треба перевести МП на розв'язання іншої задачі, наприклад, обробки аварійної ситуації або роботи з ПВВ.

4. Функція узгодження швидкодії модулів мікропроцесорної системи. Під час обслуговування пам'яті та ПВВ із значно меншою швидкодією, ніж МП, узгодження швидкодії вирішується генерацією тактів очікування МП, а під час обслуговування пристроїв з більшою швидкодією, ніж МП, використовується режим безпосереднього доступу до пам'яті.

Арифметико-логічний пристрій являє собою комбінаційну схему на основі суматора, який сигналами з виходів пристрою керування налагоджується на виконання певної арифметичної або логічної операції: додавання, віднімання, ЛОГІЧНЕ І, ЛОГІЧНЕ АБО, ЛОГІЧНЕ НІ, ВИКЛЮЧНОГО АБО, зсуву, порівняння, десяткової корекції. Отже, АЛП виконує арифметичні або логічні операції над операндами, які пересилаються з пам'яті і(або) регістрів МП.

Операнд – це об'єкт у вигляді значення даних, вмісту регістрів або вмісту комірки пам'яті, з яким оперує команда, наприклад, у команді додавання операндами є доданки. Операнд може задаватися у команді у вигляді числа або знаходитися в регістрі чи комірці пам'яті. Одержаний після виконання команди в АЛП результат пересилається в регістр або комірку пам'яті.

Регістри призначені для зберігання *n*-розрядного двійкового числа. Вони являють собою *n* тригерів зі схемами керування читанням/записом та вибірки. Регістри створюють внутрішню пам'ять МП і використовуються для зберігання проміжних результатів обчислень.

Акумулятор – це регістр, у якому зберігається один з операндів. Після виконання команди в акумуляторі замість операнда розміщується результат операції. У 8-розрядних процесорах акумулятор бере участь в усіх операціях АЛП. У 16-розрядних МП більшість команд виконуються без участі акумулятора, але в деяких командах (введення, виведення, множення, ділення) акумулятор діє так само, як і у 8-розрядних МП, тобто зберігає один з операндів, а після виконання команди – результат операції.

Вказівник команд, або *програмний лічильник*[*], призначений для зберігання адреси комірки пам'яті, яка містить код наступної команди. Програму дій МП записано в пам'яті у вигляді послідовності кодів команд. Для переходу до наступної команди вміст лічильника збільшується на одиницю у момент вибирання команди з пам'яті. Наприкінці виконання команди в лічильнику команд зберігається адреса наступної команди.

[*]У технічній літературі застосовують обидві назви цього регістра. Зазвичай у МП використовують назву *IP* (*Instruction Pointer*) – вказівник команд, а в однокристальних мікро-ЕОМ та мікроконтролерах – *PC (Program Counter)* – програмний лічильник.

Вказівник стека – це регістр, який зберігає адресу останньої зайнятої комірки стека. Стеком або стековою пам'яттю називається область пам'яті, організованої за принципом «останній прийшов – перший пішов».

Регістр команд зберігає код команди протягом усього часу виконання команди.

Регістр адреси і регістри даних призначені для зберігання адрес і даних, використовуваних під час виконання поточної команди у МП.

Регістр стану або регістр прапорців (ознак) призначений для зберігання інформації про результат операції в АЛП і являє собою декілька тригерів, які набувають одиничних або нульових значень. Наприклад, прапорець нуля встановлюється в одиницю при нульовому результаті операції.

Контрольні запитання

1. Дайте визначення архітектури МП.
2. Яка відмінність між гарвардською та фоннейманівською архітектурами?
3. Які функції виконує пристрій керування?
4. Що визначає вміст вказівника команд? Як він змінюється?
5. Чим відрізняється акумулятор від інших регістрів МП?

2.5. Основи програмування мовою асемблер

Програма являє собою послідовність команд, виконання яких приводить до розв'язку задачі.

Команда визначає операцію, яку має виконати МП над даними. Команда містить у явній або неявній формах інформацію про те, де буде розміщений результат операції, а також про адресу наступної команди. Код команди складається з декількох частин, які називаються *полями*. Склад, призначення і розташування полів називається *форматом команди*. У загальному випадку формат команди містить операційну та адресну частини. Операційна частина містить код операції (наприклад, додавання, множення, передача даних). Адресна частина складається з декількох полів і містить інформацію про адреси операндів, результату операції та наступної команди. Формат команди, у якому адресна частина складається з двох полів (ознаки адресації та адреси операндів) показано на рис. 2.10.

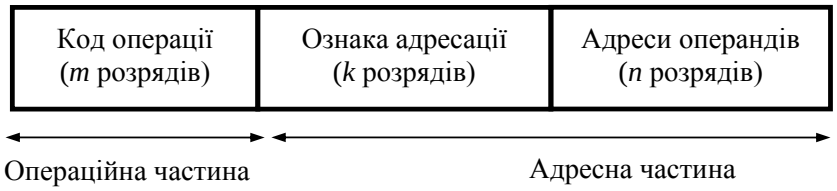


Рис. 2.10. Формат команди

Поле «Ознака адресації» визначає спосіб адресації операнда. Біти полів «Ознака адресації» та «Адреси операндів» разом визначають комірки пам'яті, в яких зберігаються операнди.

Розрізняють такі групи команд: 1) команди передачі даних; 2) команди операцій введення-виведення; 3) команди обробки інформації (арифметичні, логічні, зсуву, порівняння операндів, десяткової корекції); 4) команди керування порядком виконання програми (переходів, викликів підпрограм, повернень з підпрограм, переривань); 5) команди задання режимів роботи МП.

Загальна кількість бітів у коді команди називається *довжиною формату*. Кількість двійкових розрядів m у полі «Код операції» забезпечує можливість подання всіх операцій, які виконує МП. Якщо МП виконує M різних операцій, то кількість розрядів визначається так:

$$m \geq \log_2 M.$$

Якщо пам'ять містить S комірок, то потрібна для запису адреси одного операнда кількість розрядів у полі «Адреси операндів» становить:

$$n \geq \log_2 S.$$

Довжина формату команди визначає швидкість виконання команди і залежить від способу адресації операндів. Існують такі способи адресації: пряма, непряма, безпосередня, автоінкрементна (автодекрементна), сторінкова, індексна, відносна.

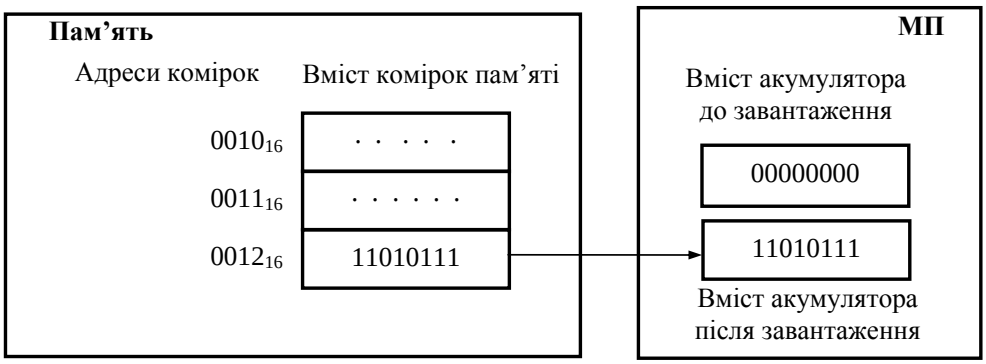
Пряма адресація. При такій адресації адреса операнда вказана безпосередньо в команді. Як приклад розглянемо команду МП К580ВМ80А (*i*8080) прямого завантаження акумулятора вмістом комірки пам'яті, розміщеної за адресою 0012_{16}. Формат та схему виконання цієї команди показано на рис. 2.11.

У байті 1 команди (рис. 2.11, *а*) міститься код операції пересилання даних в акумулятор із комірки пам'яті, а в байтах 2 і 3 – адреса комірки пам'яті. У байті 2 розташований молодший (12_{16}), а в байті 3 – старший (00_{16}) байт адреси.

На рис. 2.11, *б* комірка пам'яті з адресою 0012_{16} має вміст 11010111_2. Вміст акумулятора до операції становить 00000000_2. Після виконання команди значення вмісту комірки пам'яті копіюється в акумулятор.

а

Рис. 2.11. Команда прямого завантаження в акумулятор вмісту комірки пам'яті: *а* – формат команди; *б* – схема виконання (Див. також с. 51.)

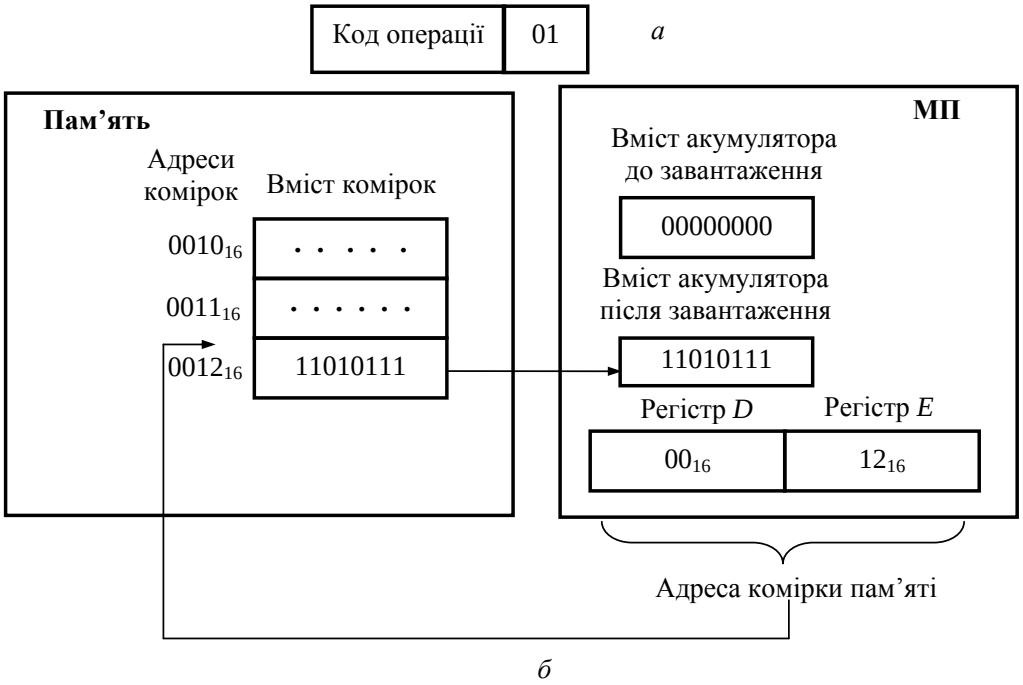


б

Рис. 2.11. Закінчення

Непряма адресація. При такій адресації у форматі команди вказується номер регістра, у якому зберігається адреса комірки пам'яті, яка містить операнд. Для збереження 16-розрядної адреси у 8-розрядному процесорі 8-розрядні регістри об'єднуються у регістрові пари. У першому регістрі регістрової пари зберігається старший байт адреси, а в другому – молодший. Номер регістрової пари, в якій зберігається адреса, є 2-розрядним двійковим числом, тому він розміщується в однобайтовій команді разом з кодом операції.

Як приклад виконання команди МП К580ВМ80А непрямого завантаження в акумулятор вмісту комірки пам'яті з адресою 0012_{16}, яка зберігається в регістровій парі *DE*, показано на рис. 2.12.



б

Рис. 2.12. Команда непрямого завантаження акумулятора:
а – формат команди; *б* – схема виконання

Команда непрямого завантаження акумулятора є однобайтовою і, крім коду операції, містить номер 01 регістрової пари *DE*. Старша частина адреси комірки пам'яті (00_{16}) зберігається у регістрі *D*, а молодша частина (12_{16}) – у регістрі *E*.

Вміст регістрової пари передається в регістр адреси МП, у результаті чого вміст 11010111_2 комірки з адресою 0012_{16} копіюється в акумулятор.

Безпосередня адресація. У першому байті команди з безпосередньою адресацією розміщується код операції. Значення операндів заносяться в команду під час програмування і знаходяться у другому або другому і третьому байтах.

Цими значеннями зазвичай є деякі константи, заздалегідь відомі програмісту. У процесі виконання програми значення операндів залишаються незмінними, оскільки вони разом із командою розміщуються в ПЗП. Використання такого способу не потребує адреси операндів. Як приклад на рис. 2.13 зображено формат і схему виконання команди безпосереднього завантаження акумулятора значенням 11010111_2, яке зберігається у другому байті команди. Після виконання команди це число копіюється в акумулятор. При кожному черговому зверненні до цієї команди в акумулятор записується таке саме число.

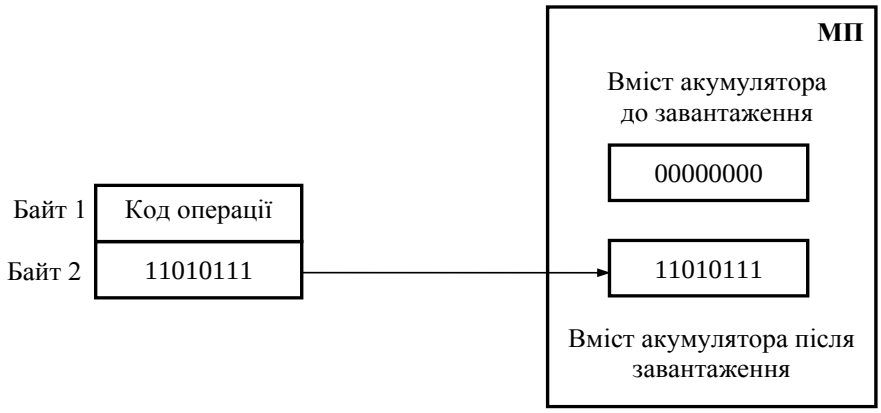


Рис. 2.13. Команда безпосереднього завантаження акумулятора

Автоінкрементна (автодекрементна) адресація. При автоінкрементній адресації адреса операнда обчислюється так само, як і при непрямій адресації, а потім здійснюється збільшення вмісту регістра: на один – для звернення до наступного байта, на два – для звернення до наступного слова. Розмір операнда визначається кодом операції.

Сторінкова адресація. Під час використання сторінкової адресації пам'ять поділяється на ряд сторінок однакової довжини. Адресація сторінок здійснюється або з програмного лічильника, або з окремого регістра сторінок. Адресація пам'яті всередині сторінок здійснюється адресою, що міститься в команді.

Індексна адресація. Для утворення адреси операнда до значення адресного поля команди додається значення вмісту індексного регістра, яке називається індексом.

Відносна адресація. При відносній адресації адреса операнда визначається додаванням вмісту програмного лічильника або іншого регістра із зазначеним у команді числом. Вміст програмного лічильника або іншого регістра називається *базовою адресою*. Для збереження базових адрес у МП можуть бути передбачені базові регістри або спеціально виділені комірки пам'яті. Тоді в адресному полі команди вказується номер базового регістра.

У МПС використовується програмування мовою асемблер. *Асемблером* називається і мова програмування у мнемокодах команд, і спеціальна програма-транслятор, що переводить (транслює) мнемокоди у машинні коди, які зчитуються мікропроцесором із пам'яті програм, дешифруються і виконуються. Процес переведення у машинні коди називається *асемблюванням.*

Програма на мові асемблер містить два типи виразів:

– команди, що транслюються в машинні коди;

– директиви, що керують ходом трансляції.

Вираз має вигляд:

```
{<мітка>}: <мнемокод> {<операнд>}{,}{< операнд >}{; коментар}.
```

У фігурних дужках наведено елементи виразу, яких може не бути у деяких командах. Мітка, мнемокод і операнди відокремлюються хоча б одним пробілом або табуляцією. Максимальна довжина рядка становить 132 символи, проте найчастіше використовуються рядки з 80 символів, що відповідає довжині екрана.

Прикладами команд асемблера є:

```
Мітка   Мнемокод  Операнд(и)  Коментар
        MOV       AX, 0       ; Команда, два операнди
M1:     ADD       AX, BX      ; Мітка, команда, два операнди
DELAY:  MOV       CX, 1234    ; Мітка, команда, два операнди
```

Прикладом директиви є:

```
Мітка   Мнемокод  Операнд(и)  Коментар
COUNT:  DB        1           ; Мітка, команда, один операнд
```

Мітка у мові асемблера є символічною адресою команди. Мітками позначаються не всі команди, а лише ті, до яких треба виконувати перехід за допомогою команд переходів або викликів підпрограм. У командах переходів або викликів підпрограм позначення мітки використовується як операнд – символічна адреса переходу, наприклад:

```
Мітка  Мнемокод  Операнд(и)  Коментар
       JMP       M1          ; Перехід до команди з міткою M1
       CALL      DELAY       ; Виклик підпрограми з міткою
                             ; DELAY
```

Після мітки ставиться двокрапка. Першим символом у мітці має бути літера або один із спеціальних символів: знак питання «?»;

крапка «.»; знак амперсанд «@»; підкреслювання «_»; знак долара «$». Знак питання і крапка можуть займати тільки перше місце. Максимальна довжина мітки – 31 символ. Приклади міток: *COUNT*, *PAGE*25, $*E*10. Рекомендується використовувати описові та смислові мітки. Усі мітки у програмі мають бути унікальними, тобто не може бути декількох команд з однаковими мітками. Як мітки не можна використовувати зарезервовані асемблером слова, до яких належать коди команд, директиви, імена регістрів. Наприклад, імена *AX, DI* та *AL* є зарезервованими і використовуються тільки для зазначення відповідних регістрів.

Мнемокод ідентифікує команду асемблера. Для мнемокодів використовують скорочені або повні англійські слова, які передають значення основної функції команди: *ADD* – додати, *SUB* (*SUBtract*) – відняти, *XCHG* (*eXCHanGe*) – поміняти.

Операнди відокремлюються комами. Якщо задано два операнди, то перший з них завжди є приймачем, а другий – джерелом інформації. Команда може містити різну кількість операндів різних типів, наприклад:

```
Мітка  Мнемокод  Операнд(и)  Коментар
       RET       M1          ; Повернутися (операнди не вказані)
       INC       CX          ; Збільшити CX (один операнд)
       ADD       AX, 12H     ; Додати 12H до вмісту AX
                             ; (два операнди)
       MOV       BX, [SI]    ; Занести до регістра BX число
                             ; з комірки пам'яті з адресою
                             ; DS:SI (два операнди)
```

Коментарі ігноруються у процесі трансляції і використовуються для документування і кращого розуміння змісту програми. Коментар завжди починається із символу «**;**» і може містити будь-які символи. Коментар може займати увесь рядок або бути розташованим за командою в одному рядку, наприклад:

```
Мітка  Мнемокод  Операнд(и)  Коментар
                             ; Цей рядок є коментарем
       ADD       AX, BX      ; Команда і коментар в одному рядку
```

Оскільки коментарі не транслюються у машинні коди, то їх кількість не впливає на ефективність виконання програми.

Програма мовою асемблер називається *початковою програмою* або *початковим програмним модулем*. Асемблювання або переведення початкової програми у машинні коди виконує програма-транслятор, наприклад *TASM.COM*. Залежно від установок, які задає користувач, програма переводить початковий модуль в один із двох програмних модулів: командний модуль (файл з розширенням *.COM*) або об'єктний модуль (файл з розширенням *.OBJ*).

Командний модуль містить машинні коди команд з абсолютними адресами і виконується МП. Командний модуль доцільно використовувати у тих випадках, коли ємність програми не перевищує розміру одного сегмента (64 кбайт). Першим оператором командного модуля є директива *ORG* 100*H* (*ORIGIN* – початок), яка розміщує першу команду програми у сегменті кодів зі зміщенням 100*H*. Закінчуватися програма має або командою *RET*, або стандартною процедурою коректного виходу до *MS DOS*:

```
Мітка  Мнемокод  Операнд(и)  Коментар
       MOV       AH, 4CH     ; Занести у AH число 4CH
                             ; (значення параметра переривання
                             ; INT 21H)
       INT       21H         ; Викликати стандартну процедуру
                             ; переривання 21H - коректного
                             ; виходу до MS DOS
```

Останнім записом програми має бути директива *END*.

Об'єктний модуль містить машинні коди команд з відносними адресами. Об'єктний модуль виконується МП після заміни відносних адрес на абсолютні за допомогою програми-укладача, наприклад *LINK.EXE*, яка генерує модуль з розширенням *.EXE* (*EXE*-файл або *EXE*-програму); *EXE*-файл, на відміну від командного модуля, може перевищувати обсяг одного сегмента. Однак у цьому разі обов'язковим є визначення сегментів за допомогою директив асемблера. Закінчується *EXE*-файл стандартною процедурою коректного виходу до *MS DOS*.

Програма-укладач має ще одне призначення – вона об'єднує об'єктний модуль з бібліотечними модулями або кілька окремих об'єктних модулів в один *EXE*-файл. *Бібліотечними модулями* називають об'єктні файли, які містять найбільш поширені підпрограми. Бібліотечні модулі розміщуються у спеціальному системному файлі – бібліотеці (*LIBRARY*).

При асемблюванні програма-транслятор генерує лістинг і файл лістингу програми. *Лістинг* – це відображення на дисплеї або папері текстів початкового програмного модуля, програмного модуля (*.COM* або *.OBJ*) та повідомлень, які вказують на помилки програмування, зумовлені порушенням правил запису виразів (наприклад, немає операнда або неправильний мнемокод команди).

Директиви призначені для керування процесом асемблювання і формування лістингу. Вони діють тільки у процесі асемблювання програми і не переводяться в машинні коди. Мова асемблер містить такі основні директиви:

- початку і кінця сегмента *SEGMENT* та *ENDS*;
- початку і кінця процедури *PROC* та *ENDP*;
- призначення сегментів *ASSUME*;
- початку *ORG*;
- розподілу та ініціювання пам'яті *DB*, *DW*, *DD*;
- завершення програми *END*;
- відзначення *LABEL*.

Директиви початку і кінця сегмента *SEGMENT* та *ENDS* призначені для опису сегментів, які використовує програма. Директиви початку і кінця сегмента використовуються разом, наприклад:

```
Назва    Мнемокод  Операнд(и)
DATASG   SEGMENT   {<параметри>}
           .
           .       Інші команди або директиви сегмента
           .
DATASG   ENDS
```

Обидві директиви *SEGMENT* і *ENDS* повинні мати однакові назви. Директива *SEGMENT* може містити три типи параметрів: вирівнювання, об'єднання і класу.

Параметр вирівнювання визначає початкову адресу або межу сегмента, наприклад:

PAGE = *xxx*00,

PARA = *xxxx*0 (межа за замовчуванням),

WORD = *xxxxe* (парна межа),

BYTE = *xxxxx*,

де *x* – будь-яка шістнадцяткова цифра; *e* – парна шістнадцяткова цифра. Якщо немає параметра вирівнювання за замовчуванням, береться параметр *PARA*, який вказує на те, що сегмент розміщується на початку параграфа, а початкова адреса сегмента є кратною 16. *Параграфом* називається область пам'яті розміром 16 байт, початкова адреса якої кратна 16, тобто має праворуч чотири нульові розряди.

Параметр об'єднання вказує на спосіб обробки сегмента при компонуванні декількох програмних модулів:

NONE: значення за замовчуванням. Сегмент має бути логічно відокремленим від інших сегментів, хоча фізично він може розміщуватися поряд. Передбачається, що сегмент має власну базову адресу.

PUBLIC: усі *PUBLIC* – сегменти з однаковими назвою та класом завантажуються у суміжні області та мають одну базову адресу.

STACK: призначення аналогічне параметру *PUBLIC*. У будь-якій програмі має бути визначений принаймні один сегмент *STACK*. Якщо визначено більше одного стека, то вказівник стека *SP* (*Stack Pointer*) встановлюється на початок першого стека.

COMMON: для сегментів *COMMON* з однаковими назвами та класами встановлюється одна спільна базова адреса. Під час виконання програми здійснюється накладання другого сегмента на перший. Розмір загальної області визначається найдовшим сегментом.

AT-параграф: параграф слід визначати заздалегідь. Цей параметр забезпечує визначення міток та змінних за фіксованими адресами у фіксованих областях пам'яті.

'Клас': цей параметр може мати будь-яку правильну назву, яка розміщується в одинарних лапках. Параметр використовується для обробки

сегментів, які мають однакові назви та класи. Типовими є класи ’*STACK*’ та ‘*CODE’*, наприклад:

```
Назва     Мнемокод  Операнд
STACKSG  SEGMENT    PARA STACK 'STACK'
```

У випадку, якщо програма не має об’єднуватися з іншими програмами, параметр об’єднання не вказується.

Директиви початку і кінця процедури *PROC* та *ENDP* використовуються для визначення підпрограм у сегменті кодів і мають такий формат:

```
<Назва> PROC  {<тип процедури>}.
```

Можливі два типи процедур:

– *NEAR* – процедура знаходиться в тому самому сегменті, що і команди, які її викликають;

– *FAR* – процедура знаходиться за межами сегмента.

За замовчуванням береться тип процедури *NEAR*.

Сегмент кодів може містити декілька процедур. Описання сегмента кодів, що містить тільки одну процедуру, має такий вигляд:

```
Назва             Мнемокод  Операнд
Ім'я_сегмента     SEGMENT   PARA
Ім'я_процедури    PROC      FAR
                            RET
Ім'я_процедури    ENDP
Ім'я_сегмента     ENDS
```

Ім’я процедури має бути обов’язково і збігатися з іменем у директиві *ENDP*, яка визначає кінець процедури.

Директива призначення сегментів *ASSUME* використовується для встановлення відповідності між сегментами та сегментними регістрами і має такий формат:

```
ASSUME    <сегментний регістр>:<ім'я>{,}{…}.
```

Наприклад, запис *SS*:ім_стек вказує, що ім’я стека визначається вмістом регістра *SS*. Одна директива *ASSUME* може призначати до чотирьох сегментних регістрів у будь-якій послідовності, наприклад:

```
Мнемокод  Операнд(и)
ASSUME    SS:ім_стек, DS:ім_дані, CS:ім_код, ES:ім_додаткові_дані
```

Для скасування будь-якого призначеного раніше у директиві *ASSUME* сегментного регістра треба використовувати слово *NOTHING*:

```
Мнемокод  Операнд(и)
ASSUME    ES: NOTHING
```

Якщо програма не використовує якийсь сегмент, то відповідний йому операнд можна пропустити або вказати слово *NOTHING*.

Директива *ORG* використовується для зміни вмісту програмного лічильника без команд умовного чи безумовного переходу. Найчастіше цю ди-

рективу використовують для встановлення початкової адреси програми, наприклад, директива *ORG* 100*H* встановлює програмний лічильник на зміщення 100*H* відносно початку сегмента кодів. Операнд зі знаком долара «$» має поточне значення програмного лічильника. Наприклад, директива *ORG* $ + 10*H* збільшує адресу, завантажену у програмний лічильник, на 10*H*.

Директиви розподілу та ініціювання пам'яті використовуються для визначення вмісту та резервування комірок пам'яті.

Директива має формат:

{ <ім'я> } *Dn* {кількість повторень *DUP*}<вираз>,

де мнемокод $Dn = \begin{Bmatrix} DB \\ DW \\ DD \\ DQ \\ DT \end{Bmatrix}$ вказує на довжину даних: *DB* – байт; *DW* – слово (2 байт); *DD* – подвійне слово; *DQ* – чотири слова; *DT* – 10 байт. Якщо у форматі наявне ім'я, то далі у програмі воно може використовуватися для позначення комірки пам'яті.

<Вираз> у форматі директиви містить одну або кілька констант для задання початкових значень вмісту комірок пам'яті або знак «?» для невизначеного значення вмісту. Наприклад, директива *ALPHA DB* 34 означає, що комірка пам'яті з іменем *ALPHA* містить число 34. У ході виконання програми вміст комірки може бути змінений. Директива *BETA DW* ? визначає, що комірка з іменем *BETA* має розрядність 16, але вміст комірки є невизначеним. Директива може містити декілька констант, розділених комами й обмежених лише довжиною рядка. Наприклад, вираз

ARRAY DB 01, 02, 11, 12, 21, 22

визначає 6 констант у вигляді послідовності суміжних байтів. Посилання на комірку з іменем *ARRAY* вказує на першу константу (01), з іменем *ARRAY* + 1 – на другу (02), з іменем *ARRAY* + 2 – на третю (11) і т. д. Запис

MOV AL, ARRAY + 4

завантажує у регістр *AL* значення 21.

Одна директива може визначити декілька комірок пам'яті. У цьому разі директива має вигляд:

{ <ім'я> } *Dn* {кількість повторень} *DUP* <вираз>.

Наприклад, директива, що визначає 5 байт, які містять число 21, записується так:

DB 5 *DUP* (21).

Директива завершення програми *END* є останньою у програмі та має такий формат:

END {<стартова адреса>}.

Параметр директиви <стартова адреса> використовується лише при створенні *EXE*-файлів.

Директива мітки *LABEL* призначена для встановлення відповідності між іменем і типом змінних. Вона має такий формат:

```
<ім'я> LABEL {<тип>}.
```

Як тип можна використовувати слова *BYTE, WORD, DWORD*, що визначають довжину даних: байт, слово або подвійне слово. Директива *LABEL* перевизначає параметри процедур *NEAR* або *FAR*. Наприклад, директива

TOS LABEL WORD

присвоює комірці пам'яті ім'я *TOS* і зазначає, що її вміст є словом.

Приклади написання простих програм. Прості програми доцільно оформляти у вигляді командних файлів. Першою директивою таких програм є директива *ORG* 100*H*, останньою – *END*.

Приклад 2.1. Написати програму додавання вмісту двох 8-розрядних комірок пам'яті, що знаходяться в сегменті даних *DS* із зміщеннями 1000*H* і 1001*H*. Результат розмістити у комірці пам'яті з адресою *DS*:1002*H*.

У цьому прикладі для простоти не будемо враховувати можливість виникнення перенесення. Програма має вигляд:

```
Мнемокод  Операнд(и)      Коментар
ORG       100H            ; Початок програми
MOV       AL, [1000H]     ; AL ← DS:[1000H]
                          ; Переслати у 8-розрядний регістр AL
                          ; вміст комірки
                          ; пам'яті з адресою DS:1000H
ADD       AL, [1001H]     ; AL ← AL + DS:[1001H]
                          ; Додати до вмісту AL вміст комірки
                          ; DS:[1001H]
MOV       [1002H], AL     ; DS:[1002H] ← AL
                          ; Переслати вміст AL у комірку
                          ; DS:[1002H]
END                       ; Завершення програми
```

Відзначимо, що запис *MOV AL*, [1000*H*] рівнозначний запису *MOV AL, DS*:[1000*H*], оскільки сегмент *DS* прийнятий за замовчуванням.

Приклад 2.2. Написати програму, яка забезпечує розділення вмісту 16-розрядної комірки пам'яті з адресою *ES*:[2000*H*] на чотири тетради. Тетради мають бути записані в молодші частини чотирьох послідовних 8-розрядних комірок пам'яті, починаючи з адреси *DS*:1000*H*, причому старша тетрада має бути записана у комірку зі старшою адресою.

У цьому прикладі для запису результату зручно використовувати непряму адресацію. Програма має вигляд:

```
Мнемокод  Операнд(и)      Коментар
ORG       100H            ; AX ← ES:[2000H]
MOV       AX, ES:[2000H]  ; Переслати вміст 16-розрядної
                          ; комірки ES:[2000H]
                          ; у 16-розрядний регістр AX
```

```
MOV     DX, AX          ; DX ← AX, зберегти початкове число
                        ; в DX
AND     AX, 000FH       ; AX←AX ∧ 0000 0000 0000 1111
                        ; Виділити молодшу тетраду (скинути
                        ; всі розряди AX, крім чотирьох
                        ; молодших)
MOV     SI, 1000H       ; SI ← 1000H
                        ; Записати в SI початкову адресу
                        ; результату
MOV     [SI], AL        ; DS:[SI] ← AL
                        ; Переслати у комірку пам'яті
                        ; з адресою DS:SI молодшу
                        ; 8-розрядну частину AL регістра AX
MOV     AX, DX          ; AX ← DX
                        ; Переслати початкове число з DX у AX
AND     AX, 00F0H       ; AX←AX ∧ 0000 0000 1111 0000
                        ; Виділити другу тетраду
MOV     CL, 4           ; Завантажити у CL число розрядів
                        ; зсуву
ROR     AL, CL          ; Циклічний зсув AL на чотири
                        ; розряди праворуч, у результаті
                        ; виділене чотирирозрядне число
                        ; переміститься в AL
INC     SI              ; SI ← SI + 1
                        ; Збільшити SI для запису
                        ; другого числа
MOV     [SI], AL        ; DS:[SI] ← AL, запам'ятати другу
                        ; тетраду у комірці DS:[SI]
MOV     AX, DX          ; AX ← DX, переслати початкове число
                        ; з DX у AX
AND     AX, 0F00H       ; AX ← AX ∧ 0000 1111 0000 0000,
                        ; Виділити третю тетраду
INC     SI              ; SI ← SI + 1, збільшити адресу
                        ; результату DS:[SI] ← AH,
MOV     [SI], AH        ; запам'ятати третю тетраду
MOV     AX, DX          ; AX←DX, переслати вихідне число
                        ; з DX у AX
AND     AX, 0F000H      ; AX ← AX ∧ 1111 0000 0000 0000
                        ; Виділити четверту тетраду
INC     SI              ; Збільшити адресу результату
MOV     CL, 4           ; Завантажити у CL число розрядів
ROR     AH, CL          ; Циклічний зсув AL на чотири
                        ; розряди праворуч
MOV     [SI], AH        ; Запам'ятати четверту тетраду
END
```

Щоб зменшити громіздкість програми, доцільно зводити її до однотипних кроків і використовувати циклічні операції. Розглянутий приклад можна спростити, якщо виконати зсув 16-розрядного числа так, щоб тетрада, яка виділяється, завжди була молодшою. Алгоритм такої програми разом з командами показано на рис. 2.14.

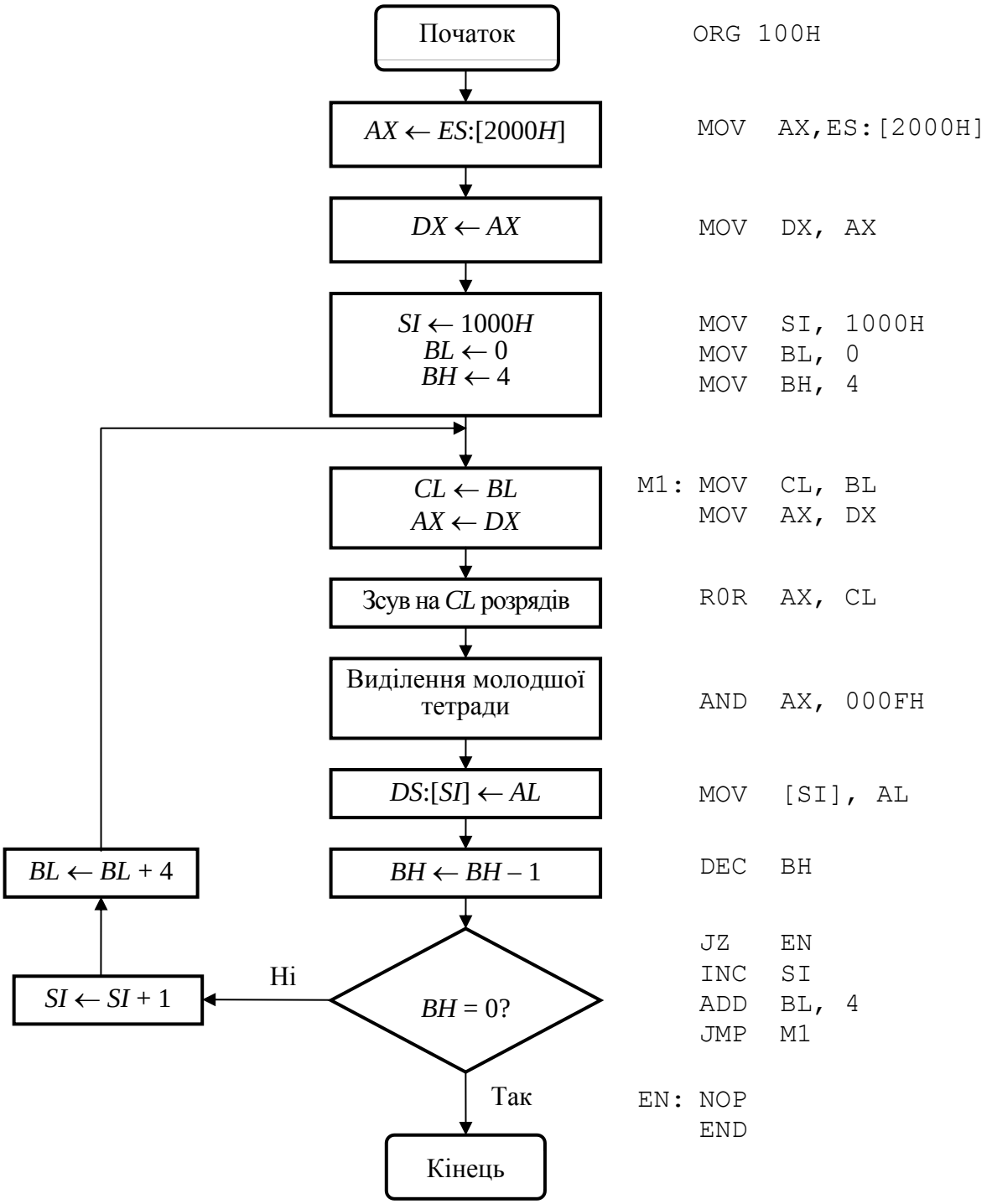


Рис. 2.14. Алгоритм розв’язання задачі прикл. 2.2

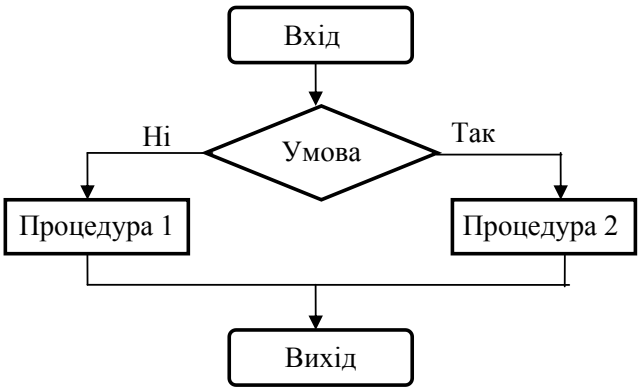


Рис. 2.15. Алгоритм процедури ЯКЩО-ТО-ІНАКШЕ

Типові обчислювальні процедури. Алгоритм типової обчислювальної процедури ЯКЩО-ТО-ІНАКШЕ показано на рис. 2.15.

Ця процедура застосовується тоді, коли треба реалізувати перехід до однієї з двох обчислювальних процедур залежно від умови. Написання програм мовою асемблер виконуються командами переходів за умовами встановлення (скидання) прапорців.

Приклад 2.3. Написати програму ділення вмісту *AX* на вміст *BL*. Результат помістити у 8-розрядну комірку пам'яті з адресою *DS*:1000*H*. Остачею від ділення знехтувати. Якщо вміст *BL* = 0, то ділення не виконувати, на місце результату помістити число 0*FFH*.

Програма має вигляд:

```
Мітка   Мнемокод   Операнд(и)     Коментар
        ORG        100H
        CMP        BL,0           ; Порівняти вміст BL з нулем; результат
                                  ; команди впливає на встановлення
                                  ; прапорця нуля Z
        JZ         M1             ; Якщо Z = 1 (BL = 0), то перехід на
                                  ; мітку M1,
        DIV        BL             ; інакше виконати ділення AL ← AX : BL,
                                  ; остача → AH
        JMP        M2             ; Безумовний перехід на мітку M2
M1:     MOV        AL,0FFH        ; Занести число 0FFH у AL
M2:     MOV        [1000H],AL     ; Запам'ятати результат у комірці
                                  ; DS : [1000H]
        END
```

Процедура ЯКЩО-ТО (рис. 2.16) є частковим випадком процедури ЯКЩО-ТО-ІНАКШЕ і використовується в тому випадку, коли треба реалізувати одну обчислювальну процедуру залежно від умови.

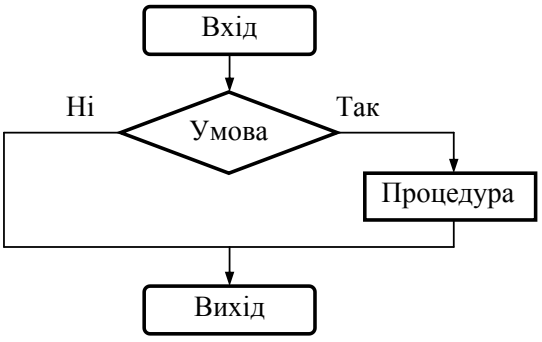


Рис. 2.16. Алгоритм процедури ЯКЩО-ТО

Процедура РОБИ-ПОКИ (рис 2.17) використовується для повторення однотипних дій до моменту виконання умови закінчення циклу.

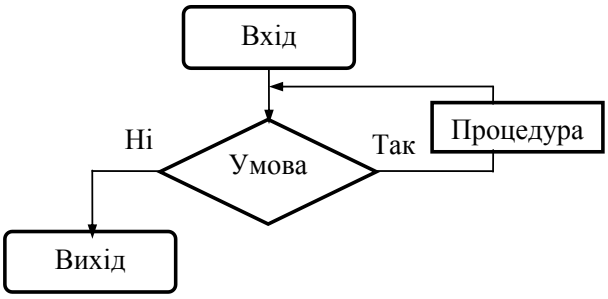


Рис. 2.17. Алгоритм процедури РОБИ-ПОКИ

Приклад 2.4. Написати програму додавання за модулем 256 масиву з 100*H* байт, розташованих за початковою адресою 7000*H*:3000*H*. Результат у вигляді одного байта записати в комірку з адресою 7000*H*:5000*H*.

Програма має вигляд:

```
Мітка   Мнемокод    Операнд(и)      Коментар
        ORG         100H            ; Початок
        MOV         AX,7000H        ; Завантажити в AX адресу сегмента
        MOV         DS,AX           ; Завантажити в DS адресу сегмента
        MOV         SI,3000H        ; Завантажити в SI зміщення першого
                                    ; елемента масиву
        MOV         CX,101H         ; Завантажити у лічильник CX довжину
                                    ; масиву +1
        MOV         AL,[SI]         ; Завантажити у AL перший елемент масиву
M1:     LOOP        M0              ; Зменшити вміст CX на 1, якщо CX≠0,
                                    ; то перейти на мітку M0,
        MOV         [5000H],AL      ; інакше запам'ятати результат
                                    ; у DS:[5000H]
        JMP         EXIT            ; Перехід на вихід
M0:     INC         SI              ; SI←SI+1 – адреса наступного
                                    ; елемента
        ADD         AL,[SI]         ; Додати вміст DS:SI до попередньої
                                    ; суми в акумулятори AL
        JMP         M1              ; Перейти на мітку M1 для перевірки
                                    ; умови виходу із циклу
        EXIT:NOP                    ; Вихід
        END
```

Процедура ПОВТОРЮЙ-ДО-ТОГО-ЯК (рис. 2.18) аналогічна попередній, але однотипні дії виконуються перед перевіркою умови.

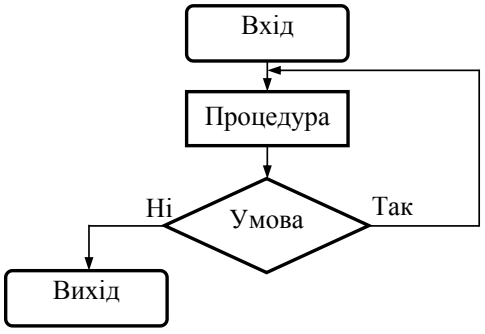


Рис. 2.18. Алгоритм процедури ПОВТОРЮЙ-ДО-ТОГО-ЯК

Програма виконання завдання прикл. 2.4 згідно з алгоритмом (рис. 2.18) має вигляд:

```
Мітка  Мнемокод  Операнд(и)     Коментар
       ORG       100H           ; Початок
       MOV       AX,7000H       ; Завантажити у AX адресу сегмента
       MOV       DS,AX          ; Завантажити в DS адресу сегмента
       MOV       SI,3000H       ; Завантажити в SI зміщення першого
                                ; елемента масиву
       MOV       CX,100H        ; Завантажити в лічильник CX довжину
                                ; масиву
       MOV       AL,[SI]        ; Завантажити у AL перший елемент
                                ; масиву
M0:    INC       SI             ; SI←SI+1 – адреса наступного елемента
       ADD       AL,[SI]        ; Додати вміст DS:SI до попередньої
                                ; суми у AL
       LOOP      M0             ; Зменшити вміст CX на одиницю,
                                ; якщо CX≠0 , то перейти до мітки
                                M0,
       MOV       [5000H],AL     ; інакше запам'ятати результат
                                ; у DS:[5000H]
       JMP       EXIT           ; Перехід на вихід
       END
```

Написання *EXE*-програм. Написання *EXE*-програм має виконуватися за таких умов:

1) зазначення відповідності між сегментами та сегментними регістрами;
2) зберігання вмісту *DS* у стеку;
3) записування числа 0 у стек;
4) завантаження адреси сегмента даних у регістр *DS*.

Перша вимога виконується за допомогою директиви *ASSUME*, інші – за допомогою відповідних команд асемблера.

Приклад 2.5. Написати *EXE*-програму знаходження максимального числа у масиві 8-розрядних беззнакових чисел. Результат записати в регістр *DL*.

Програма має вигляд:

```
Мітка       Мнемокод  Операнд(и)              Коментар
(або ім'я)
DATASG      SEGMENT   PARA 'DATA'             ; Визначити сегмент даних
MASSIV      DB        01,02,03,45,56,         ; Визначити у сегменті
                      67,78,89,0FE,10         ; даних 10 значень масиву
                                              ; з іменем MASSIV
DATASG      ENDS
STACKSG     SEGMENT PARA STACK 'Stack'        ; Визначити сегмент стека
            DW 100 DUP (?)                    ; Визначити 100 слів
TOS         LABEL     WORD                    ; Визначити ім'я і формат
                                              ; вершини стека
STACKSG ENDS
CODESG      SEGMENT   PARA 'CODE'             ; Визначити сегмент кодів
```

```
        BEGIN   PROC FAR            ; Початок процедури
        ASSUME  SS: STACKSG, DS:DATASG, CS: CODESG
        PUSH    DS                  ; Завантажити вміст DS у стек
        SUB     AX,AX               ; Встановити нульовий
                                    ; вміст у AX
        PUSH    AX                  ; Записати нуль у стек
        MOV     AX,DATASG           ; Завантажити адресу
                                    ; DATASG у AX
        MOV     DS,AX               ; Записати адресу DATASG
                                    ; у регістр DS
        LEA     BX,MASSIV           ; Завантажити у регістр BX
                                    ; адресу першого елемента
                                    ; масиву
        MOV     CX,10               ; Завантажити у CX довжину
                                    ; масиву
        MOV     DL,[BX]             ; DL←DS:[BX]
COMP:   MOV     AL,[BX]             ; AL←DS:[BX]
        CMP     AL,[BX+1]           ; Порівняти два сусідні
                                    ; елементи масиву
        JAE     NEXT                ; Якщо вміст попереднього
                                    ; елемента масиву [BX]
                                    ; більший або дорівнює
                                    ; вмісту наступного [BX +1],
                                    ; то перейти на мітку NEXT,
        MOV     DL,[BX+1]           ; інакше завантажити у DL
                                    ; значення [BX + 1]
NEXT:   INC     BX                  ; Збільшити BX для адресації
                                    ; наступного елемента масиву
        LOOP    COMP                ; Перевірка умови виходу
                                    ; з циклу
        RET                         ; повернення
        BEGIN   ENDP                ; Кінець процедури BEGIN
CODESG  ENDS                        ; Кінець сегмента кодів
        END                         ; Кінець програми
```

У розглянутому прикладі вихід *у MS DOS* здійснюється командою *RET* з використанням для цього адреси, записаної у стек на початку програми командою *PUSH DS*. Інакше можна завершити програму командою *INT* 20*H*.

Контрольні запитання

1. Наведіть основні характеристики та структурні схеми процесорів з фоннеймановською та гарвардською архітектурами.
2. Вкажіть призначення регістрів МП з фоннеймановською архітектурою.
3. Охарактеризуйте функції пристрою керування.
4. Наведіть визначення та призначення акумулятора.
5. Поясніть роботу програмного лічильника.
6. Вкажіть місцезнаходження операнда з прямою адресацією.

7. Поясніть, яким чином визначається адреса операнда з непрямою адресацією.
8. Поясніть, як визначається значення операнда з безпосередньою адресацією.
9. Поясніть, яким чином визначається адреса операнда з відносною адресацією.
10. Напишіть програму пересилання вмісту 8-розрядної комірки пам’яті з адресою 7000*H*:1000*H* у 8-розрядний регістр *AL*.
11. Напишіть програму віднімання вмісту двох послідовних комірок пам’яті з адресами *DS*:35*A*0*H* і 35*A*1*H* із записом результату в комірку з адресою 35*A*2*H*.
12. Напишіть програму додавання масивів байтів з адресами 8350:4735*H* і 3660:2200*H* за правилом «перший з першим, другий з другим і т. д.». Занести в масив 6250:2400*H* адреси тих пар доданків, сума яких дорівнює нулю. Довжина масиву 100*H*.
13. Виконайте ділення масиву з 25*H* слів 5*B*00:3000*H* на масив з 25*H* байт 5*C*00:4000*H* за правилом «перший на перший, другий на другий і т. д.». Результати занести в масив 6000:5000*H*. За потреби ділення на 0 ділення не виконувати, а байти результату завантажити числом 1*AH*.

Розділ 3

ОДНОКРИСТАЛЬНІ МІКРОПРОЦЕСОРИ

3.1. Однокристальний 8-розрядний мікропроцесор

Структурну схему узагальненого 8-розрядного однокристального МП показано на рис. 3.1. Схема має єдину внутрішню 8-розрядну шину, по якій передаються дані, коди команд та адреси.

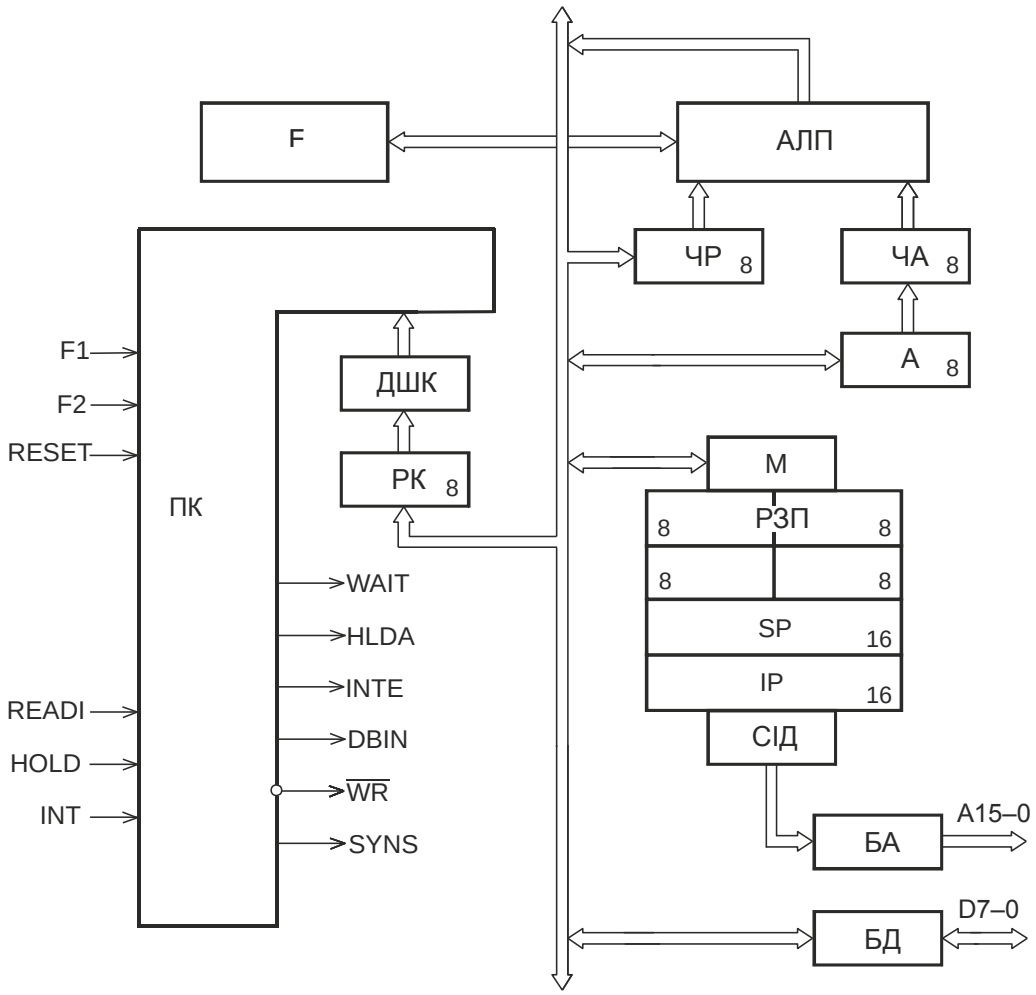


Рис. 3.1. Структурна схема 8-розрядного мікропроцесора:
ПК – пристрій керування; ДШК – дешифратор команд; РК – регістр команд; АЛП – арифметико-логічний пристрій; *A* – акумулятор; ЧА – часовий акумулятор; ЧР – часовий регістр; *F* – регістр прапорців; РЗП – регістри загального призначення; М – мультиплексор; *SP* (*Stack Pointer*) – вказівник стека; *IP* – (*Instruction Pointer*) – вказівник команд; БА – буферний регістр адреси; БД – буферний регістр даних; СІД – схема інкремента/декремента

Пристрій керування відповідно до дешифрованих кодів команд та зовнішніх керувальних сигналів генерує керувальні сигнали для всіх блоків структурної схеми.

Дешифратор команд формує сигнали для пристрою керування згідно з дешифрованим кодом команди.

У 8-розрядному регістрі команд зберігається машинний код команди (один байт).

Арифметико-логічний пристрій являє собою комбінаційну схему на основі суматора і логічних елементів, який сигналами з виходів пристрою керування налагоджується на ту чи іншу арифметичну або логічну операцію – додавання, віднімання, І, АБО, ВИКЛЮЧНЕ АБО, НІ, зсув.

Акумулятор є 8-розрядним регістром, в якому зберігається один з операндів у двоопераднних командах, а також результат операції. Наприклад, у команді додавання

$$ADD\ B;\ A+B \rightarrow A$$

явно вказаний лише один операнд – 8-розрядний регістр *B*. Регістр *B* – один з РЗП. У деяких МП РЗП позначаються літерами латинського алфавіту: *B, C, D, H, L*, в інших – *R*0, *R*1, *R*2,… . Другим операндом є акумулятор. Результат додавання вмісту акумулятора та регістра *B* переноситься в акумулятор, що символічно записується в коментарі до команди.

Часовий акумулятор та часовий регістр являють собою 8-розрядні буферні регістри, які дозволять відокремити входи АЛП від його виходу, тобто уникнути гонки сигналів.

Регістр прапорців *F* (від англ. *flags* – прапорці) або ознак являє собою декілька тригерів (п’ять або шість), які встановлюються в одиничний (або скидаються в нульовий) стан залежно від результату операції в АЛП.

Регістри загального призначення – блок 8-розрядних РЗП, у яких зберігаються дані та проміжні результати. Цей блок можна розглядати як ОЗП, що має найбільшу швидкодію серед ОЗП різноманітних типів, оскільки він розміщений безпосередньо на кристалі ВІС МП. Деякі типи 8-розрядних процесорів, крім 8-розрядних РЗП, містять 16-розрядні індексні регістри для організації непрямої адресації, інші – припускають звернення до пари 8-розрядних регістрів як до одного 16-розрядного.

Мультиплексор – пристрій, що з’єднує один з РЗП із внутрішньою шиною МП.

Вказівник стека *SP (Stack Pointer)* – 16-розрядний регістр, у якому зберігається адреса останньої зайнятої комірки стека.

Вказівник команд *IP (Instruction Pointer)* – 16-розрядний регістр, у якому зберігається адреса команди, яка виконується. Після вибірки з пам’яті програм кожного байта команди вміст *IP* збільшується на одиницю.

Буферний регістр адреси та буферний регістр даних – регістри з трьома станами виходу, призначені для формування сигналів на лініях шин адреси і даних відповідно.

Схема інкремента/декремента – пристрій, який дозволяє без участі АЛП збільшити або зменшити на одиницю вміст одного з регістрів РЗП, *IP* або *SP*.

Конструктивно ВІС 8-розрядного процесора виконано в корпусі з 40 виводами, з яких 16 припадає на шину адреси, 8 – на шину даних, 2 (4) – на ввімкнення живлення, а інші – на лінії шини керування. Основні лінії шини керування показано на рис. 3.1:

– *F*1, *F*2 – вхід двох послідовностей імпульсів синхронізації, що не перекриваються (рис. 3.2);

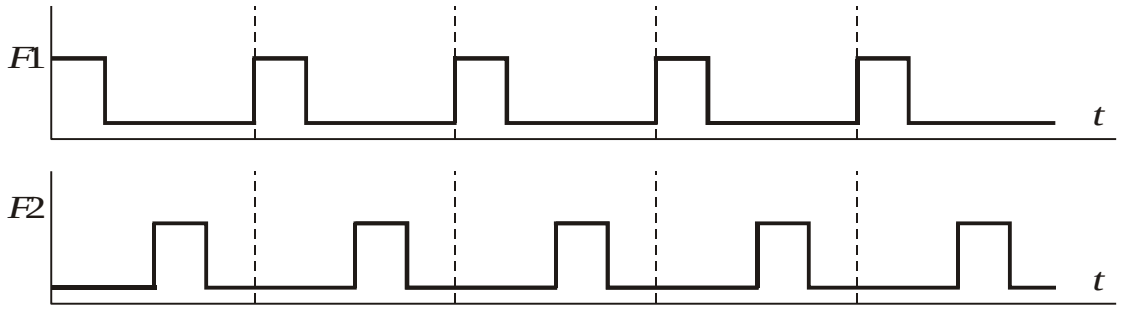


Рис. 3.2. Часові діаграми імпульсів синхронізації *F*1 і *F*2

– *RESET* – вхід сигналу початкового встановлення (скидання);
– *READY* – вхід сигналу готовності зовнішнього пристрою або пам'яті до обміну; використовується для організації обміну з менш швидкодіючими (порівняно з МП) пристроями;
– *WAIT* – вихід сигналу підтвердження очікування; активний рівень сигналу свідчить про те, що процесор перейшов у режим очікування і виконує холості такти;
– *HOLD* – вхід сигналу запиту прямого доступу до пам'яті (ПДП), або запит захоплення шин; використовується для організації обміну з пристроями, швидкодія яких вища від швидкодії процесора;
– *HLDA* (*HoLD Acknowledge*) – вихід сигналу підтвердження ПДП; активний рівень цього сигналу свідчить про те, що процесор перевів свої шини адреси, даних та керування у високоімпедансний стан;
– *INT* (*INTerrupt*) – вхід сигналу запиту переривання;
– *INTE* (*INTerrupt Enable*) – вихід сигналу дозволу переривання;
– *DBIN* (*Data Bus IN*) – вихід сигналу читання; *H*-рівень цього сигналу свідчить про те, що двонапрямлена шина даних знаходиться у режимі прийому інформації;
– $\overline{WR}$ (*WRite*) – вихід сигналу запису; *L*-рівень цього сигналу свідчить про те, що двонапрямлена шина даних знаходиться у режимі видачі інформації;

– *SYNC* (*SYNChronization*) – вихід сигналу синхронізації; *H*-рівень цього сигналу свідчить про те, що на шині даних передається байт стану, який використовується для формування деяких керувальних сигналів.

Схеми конкретних МП відрізняються кількістю та позначенням регістрів, а також деякими керувальними сигналами. Наприклад, у МП *i*8085 замість двох сигналів *F*1 і *F*2 використовується один сигнал синхронізації *CLK* (*CLocK*); замість сигналу *DBIN* – сигнал читання $\overline{RD}$ (*ReaD*). Нульовий рівень цього сигналу свідчить про те, що двонапрямлена шина даних знаходиться у режимі прийому інформації. У МП *i*8085 є додатковий сигнал $M / \overline{IO}$ (*Memory/Input-Output*) – ознака звернення до пам'яті (логічна одиниця) або до пристрою введення-виведення (логічний нуль), але немає сигналу *SYNC*.

Схема (див. рис. 3.1) працює таким чином. У разі ввімкнення живлення або формування сигналу початкового встановлення *RESET* вміст вказівника команд *IP* набуває нульового значення і починається машинний цикл вибірки команди з пам'яті. Вміст комірки пам'яті за нульовою адресою через буферний регістр даних та внутрішню шину МП надходить у регістр команд, після цього – у дешифратор команд. Відповідно до дешифрованих кодів команд і зовнішніх сигналів синхронізації та керування пристрій керування формує керувальні імпульси для кожної мікрооперації команди.

Програмною моделлю МП називається сукупність програмно доступних регістрів, тобто тих регістрів, вміст яких можна зчитати або змінити за допомогою команд. Програмну модель МП складають акумулятор, РЗП, регістр прапорців, вказівник стека та вказівник команд.

Організація пам'яті. Максимально можлива ємність пам'яті з прямою адресацією визначається кількістю розрядів шини адреси. Більшість 8-розрядних процесорів (*i*8080, *i*8085, *Z*80, *Motorola* 6800) мають 16-розрядну шину адрес, тобто дозволяють адресувати 2^{16} = 64 кбайт пам'яті.

Мікропроцесори з 8-розрядною шиною даних мають чотири режими адресації операндів:

1. *Пряма адресація.* У цьому режимі другий та третій байти команд містять адресу операнда.
2. *Регістрова адресація.* У мнемоніці команди вказується позначення РЗП, у якому знаходиться операнд.
3. *Безпосередня адресація.* У цьому режимі в команді вказується 8- або 16-бітовий операнд у другому або у другому та третьому байтах команди. Операнд у цьому випадку знаходиться у пам'яті програм.
4. *Непряма регістрова адресація*. У команді вказується регістр (або пара регістрів), який містить адресу комірки пам'яті.

Організація введення-виведення. 8-розрядні МП мають можливість передати або прийняти дані із зовнішніх ПВВ. Пристрої введення-виведення з'єднуються із системною шиною МП системи за допомогою портів введення-виведення, які являють собою 8-розрядні регістри зі схемами вибірки та керуванням читанням/записом. Кількість таких пристроїв визначається можливим діапазоном 8-розрядних адрес портів, тобто $2^8 = 256$ портів введення і 256 портів виведення. Як порти введення можуть бути використані буферні регістри, наприклад: КР580ИР82, КР589ИР12 або інтерфейс введення-виведення паралельної інформації КР580ВВ55.

Введення або виведення даних може здійснюватися двома способами: з використанням окремого адресного простору ПВВ; з використанням спільного з пам'яттю адресного простору, тобто з відображенням на пам'ять.

Перший спосіб дозволяє виконувати введення і виведення даних за командами введення *IN* та виведення *OUT*. Використання другого способу передбачає розміщення адрес портів у спільному з пам'яттю адресному просторі. При цьому операції звернення до портів не відрізняються від операцій звернення до пам'яті.

Виконання команд у МП *i*8080. Кожна команда у МП виконується протягом командного циклу. Командний цикл складається із циклу вибірки команди та циклу виконання команди (рис. 3.3).

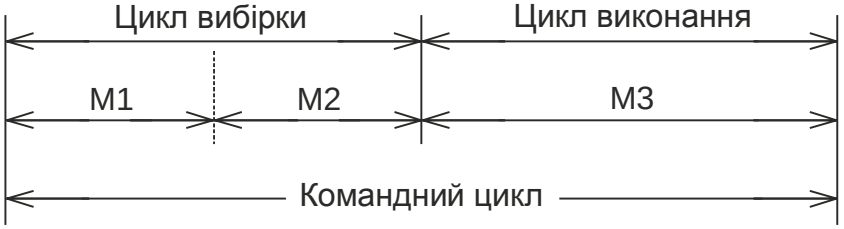


Рис. 3.3. Приклад командного циклу 8-розрядного МП

Тривалість циклу вибірки команди залежить від формату команди (кількості байтів у машинному коді команди). Команди займають від одного до трьох байтів у програмній пам'яті. Багатобайтові команди зберігаються в сусідніх комірках пам'яті. Для вибірки однобайтової команди (наприклад, додавання акумулятора *A* і регістра *B* – *ADD B*) потрібне одне звернення до пам'яті, для вибірки трибайтової команди (наприклад, виклику підпрограми за адресою *ADDR* – *CALL ADDR*) – три звернення. Тривалість циклу виконання команди залежить від засобу адресації операндів. Так, при виконанні команд з регістровою адресацією не потрібне додаткове звернення до пам'яті для читання операнда, у командах з непрямою адресацією таке звернення необхідне. Тому тривалість командного циклу в МП *i*8080 є різною для різних команд і визначається кількістю звернень до пам'яті або до зовнішнього пристрою.

Інтервал, протягом якого здійснюється одне звернення процесора до пам'яті чи до зовнішнього пристрою, визначається як *машинний цикл M*. Отже, командний цикл процесора складається з деякої кількості машинних циклів (залежно від типу команди). У наведеному на рис. 3.3 прикладі цикл вибірки складається з двох машинних циклів (*M*1 і *M*2), а цикл виконання – з одного машинного циклу (*M*3). У команді може бути від одного (для однобайтових команд з регістровою адресацією) до п'яти (для трибайтових складних команд) машинних циклів.

Машинний цикл, у свою чергу, розбивається на деяку кількість машинних тактів *T*, протягом кожного з яких виконується елементарна дія (мікрооперація) у процесорі. Кількість тактів у циклі визначається кодом команди і знаходиться у межах від 3 до 5. Тривалість такту задається періодом імпульсів синхронізації і визначається як інтервал часу між фронтами двох сусідніх імпульсів послідовності *F*1, яка формується зовнішніми ланцюгами. Отже, командний цикл МП *i*8080 складається з деякої кількості машинних циклів, а кожний машинний цикл – з визначеної кількості тактів, протягом яких виконуються ті чи інші елементарні дії у процесорі.

Для синхронізації процесора з пам'яттю та зовнішніми пристроями, які характеризуються меншою швидкодією, для організації роботи в режимі ПДП та зупину процесора передбачено три особливі режими: очікування, захоплення шин, зупину, тривалість яких має довільне, але завжди кратне тривалості такту *T* значення.

Розрізняють такі типи машинних циклів залежно від дій, що виконує МП:

ВИБІРКА (читання першого байта команди);

ЧИТАННЯ ПАМ'ЯТІ (читання другого та третього байтів команди, читання операнда);

ЗАПИС У ПАМ'ЯТЬ;

ЧИТАННЯ СТЕКА;

ЗАПИС У СТЕК;

ВВЕДЕННЯ даних із зовнішнього пристрою;

ВИВЕДЕННЯ даних на зовнішній пристрій;

ПЕРЕРИВАННЯ;

ЗУПИН;

ПЕРЕРИВАННЯ ПРИ ЗУПИНІ.

Першим машинним циклом команди завжди є цикл ВИБІРКА, протягом якого здійснюється вибірка з пам'яті байта коду команди за адресою, що визначається вмістом вказівника команд. Вміст вказівника у циклі збільшується на одиницю. Однобайтові команди з регістровою адресацією потребують для виконання лише одного циклу ВИБІРКА.

Для вибірки дво- або трибайтових команд, крім циклу ВИБІРКА, потрібні ще один або два машинні цикли для читання другого або другого і третього байтів команди. Ці цикли визначаються як цикли ЧИТАННЯ ПАМ'ЯТІ. Цикл ЧИТАННЯ ПАМ'ЯТІ потрібний також для вибірки операнда при виконанні команд з непрямою або прямою адресацією. Для запису операндів або зберігання адрес при виконанні відповідних команд пересилок потрібні машинні цикли ЗАПИС У ПАМ'ЯТЬ.

У командах із зверненням до стека виконуються цикли ЧИТАННЯ СТЕКА та ЗАПИС У СТЕК. Адреси пам'яті визначаються вказівником стека *SP*. Для виконання команд введення-виведення виконуються машинні цикли ВВЕДЕННЯ та ВИВЕДЕННЯ; для організації переривань програми та зупину процесора – цикли ПЕРЕРИВАННЯ, ЗУПИН, ПЕРЕРИВАННЯ ПРИ ЗУПИНІ.

Кожний машинний цикл процесора ідентифікується байтом, який називається *байтом стану*. Байт стану видається на шину даних на початку кожного машинного циклу і супроводжується одночасним видаванням сигналу *SYNC* на однойменний контакт ВІС МП. Байт стану несе інформацію про наступні дії процесора. Його можна запам'ятати за сигналом *SYNC* і сформувати такі керувальні сигнали, які не вдалося вивести у явному вигляді на контакти ВІС МП (через обмежену кількість контактів мікросхеми). Байти стану для процесора *i*8080 наведено в табл. 3.1.

Таблиця 3.1. **Байти стану для різних типів машинних циклів**

Розряд шини даних	**Тип машинного циклу**									
	ВИБІРКА	ЧИТАННЯ ПАМ'ЯТІ	ЗАПИС У ПАМ'ЯТЬ	ЧИТАННЯ СТЕКА	ЗАПИС У СТЕК	ВВЕДЕННЯ	ВИВЕДЕННЯ	ПЕРЕРИВАННЯ	ЗУПИН	ПЕРЕРИВАННЯ ПРИ ЗУПИНІ
*D*0	0	0	0	0	0	0	0	1	0	1
*D*1	1	1	0	1	0	1	0	1	1	1
*D*2	0	0	0	1	1	0	0	0	0	0
*D*3	0	0	0	0	0	0	0	0	1	1
*D*4	0	0	0	0	0	0	1	0	0	0
*D*5	1	0	0	0	0	0	0	1	0	1
*D*6	0	0	0	0	0	1	0	0	0	0
*D*7	1	1	0	1	0	0	0	0	1	0

Часову діаграму машинного циклу ВИБІРКА (ЧИТАННЯ ПАМ'ЯТІ) для процесора *i*8080 показано на рис. 3.4, а циклу ЗАПИС У ПАМ'ЯТЬ – на рис. 3.5. Такти підраховуються за передніми фронтами послідов-

ності *F*1, а мікрооперації в кожному такті визначаються переднім фронтом сигналу *F*2. При тактовій частоті 2 МГц тривалість такту дорівнює 0,5 мкс. Сигнали на лініях шин *A*15–*A*0 (*D*7–*D*0) зображено на рис. 3.4 на одній часовій діаграмі у вигляді ліній *L*- та *H*-рівнів одно-часно. Отже, на діаграмі вказано лише тип інформації, наявної на шині, наприклад, встановлення на шині значення адреси або даних. Пунктирною лінією позначено високоімпедансний стан ліній шин (*z*-стан).

У першому машинному такті *T*1 на шину адреси (лінії *A*15–*A*0) ви-дається адреса – вміст вказівника команд *IP* (якщо виконується цикл ВИБІРКА) або вміст вказівника адреси (якщо виконується цикл ЧИТАННЯ ПАМ'ЯТІ). Водночас на шину даних (лінії *D*7–*D*0) видається байт стану, а також формується сигнал *SYNC* на однойменному контакті ВІС МП.

У другому такті *T*2 закінчується надходження байта стану і сигналу *SYNC*, тривалість яких дорівнює одному такту. У машинному циклі ВИБІРКА вміст *IP* збільшується для адресації наступного байта команди або наступної команди. У цьому ж такті пристрій керування МП здійснює аналіз сигналів на входах *READY* і *HOLD*, а також контроль виконання команди зупину *HLT*. Якщо пам'ять або зовнішній пристрій не готові до обміну (*READY* = 0), оскільки надійшов запит ПДП (*HOLD* = 1) або виконується команда зупину *HLT*, то обмін даними здійснюватися не може, і процесор переходить в один із режимів – очікування, захоплення шин або зупину. У цих режимах здійснюється очікування сигналу протягом декількох тактів очікування T_w, кількість яких визначається зовнішніми сигналами. На рис. 3.4 у такті *T*2 сигнал *READY* дорівнює логічному нулю, а у такті T_w – логічній одиниці.

У такті *T*3 залежно від типу машинного циклу здійснюється звернення до пам'яті, стека або зовнішнього пристрою. У результаті в МП вво-диться (рис. 3.4) або з нього виводиться (рис. 3.5) байт команди, адреси або даних. Залежно від типу команди машинний цикл може містити такти *T*4 і *T*5 (наприклад, якщо для виконання команди потрібна обробка операндів). В останньому такті команди (*T*3, *T*4 або *T*5) аналізується наявність сигналу запиту переривання *INT*. Якщо переривання дозволено, то процесор переходить до машинного циклу ПЕРЕРИВАННЯ.

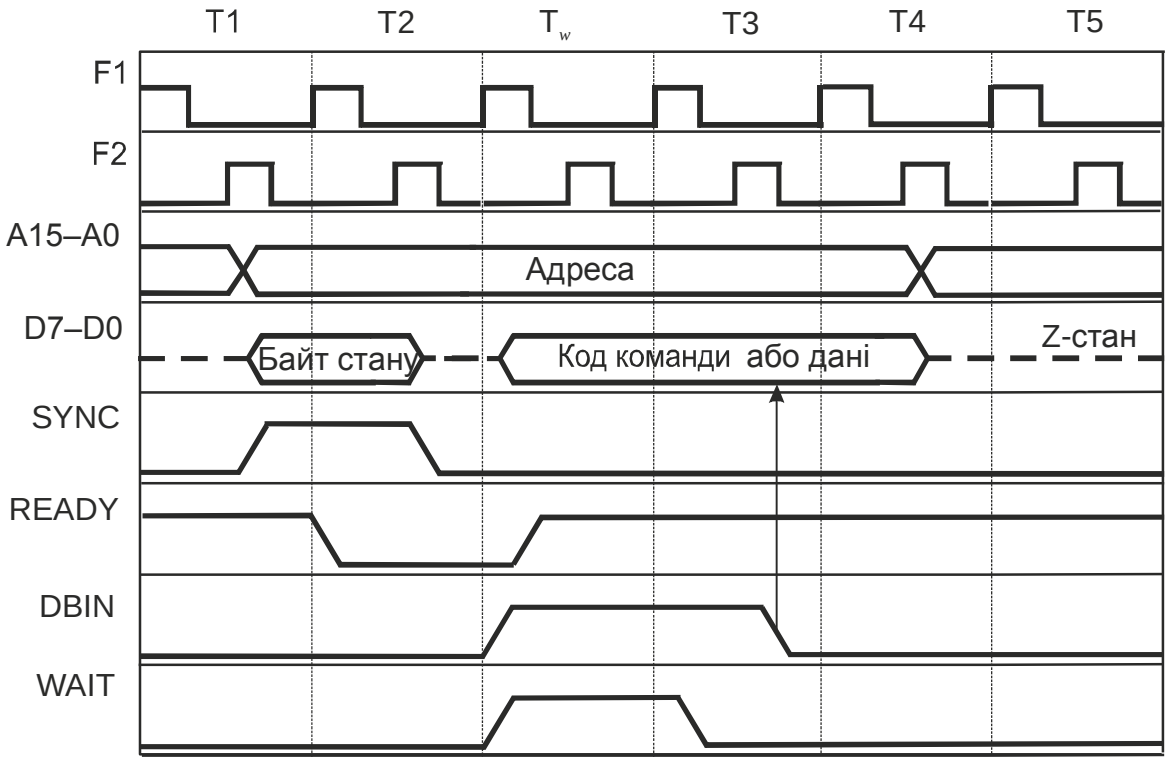


Рис. 3.4. Цикл ВИБІРКА та ЧИТАННЯ ПАМ’ЯТІ

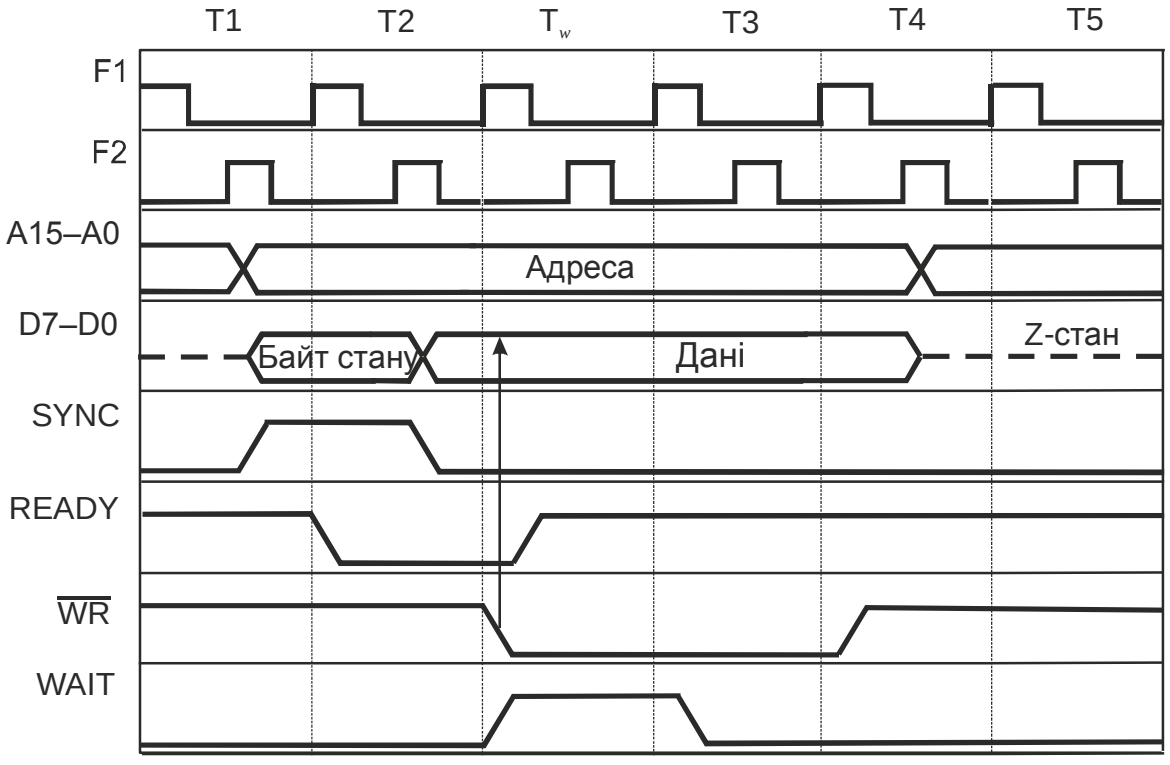


Рис. 3.5. Цикл ЗАПИС У ПАМ’ЯТЬ

Як приклад розглянемо виконання команди *ADD B* за мікроопераціями. Команда містить один машинний цикл ВИБІРКА, що вико-

нується за чотири такти, а потрібна для виконання команди мікроопе-рація п'ятого такту виконується в такті *Т*2 наступної команди. На рис. 3.6, *а – д* показано дії МП у кожному такті машинного циклу. У такті *Т*1 (рис. 3.6, *а*) на шину адреси видається вміст вказівника команд *IP*, який у такті *Т*2 (рис. 3.6, *б*) збільшується на одиницю для адресації наступної команди. Команда вибирається з пам'яті у такті *Т*3 (рис. 3.6, *в*). У такті *Т*4 (рис. 3.6, *г*) здійснюється підготовка операндів до додавання: вміст регістра *В* по внутрішній шині пересилається в часовий регістр, а вміст акумулятора *А* – у регістр у часовий акумулятор. У п'ятому такті, суміщеному з тактом *Т*2 наступної команди для збільшення швидкодії, виконується додавання операндів. Результат додавання запам'ятовується в акумуляторі.

Особливі режими роботи МП *i*8080. Мікропроцесор *i*8080 має такі особливі режими роботи: переривання, очікування, захоплення шин при ПДП, що ініціюються зовнішніми сигналами керування, зупин (перехід до цього режиму здійснюється програмно).

Переривання. У мікропроцесорі *i*8080 є засоби обробки запитів переривань восьми рівнів. Якщо один із зовнішніх пристроїв, з'єднаних із системою переривання МП ініціює запит переривання (див. рис. 3.6), то система формує сигнал на виводі *INT* МП у вигляді сигналу високого рівня. Одночасно на шину даних система переривання посилає код команди *RST V* (переривання за вектором *V*). Вектор *V* являє собою код, який вказує на адресу початкової команди підпрограми обслуговування даного запиту переривання

Послідовність дій МП у режимі переривання така:

1) приймання запиту переривання та блокування входу запиту переривання;
2) приймання команди *RST V*;
3) зберігання адреси повернення (вмісту вказівника команд) у стеку;
4) формування адреси підпрограми обслуговування джерела запиту.

Запити переривання МП *i*8080 приймаються із входу *INT* МП тригером переривань, яким керує тригер дозволу переривань. У свою чергу, тригер дозволу переривань можна програмним способом установити в логічну одиницю або нуль. Стан логічної одиниці тригера дозволу переривань дозволяє приймання переривання із входу *INT*, стан нуля – забороняє.

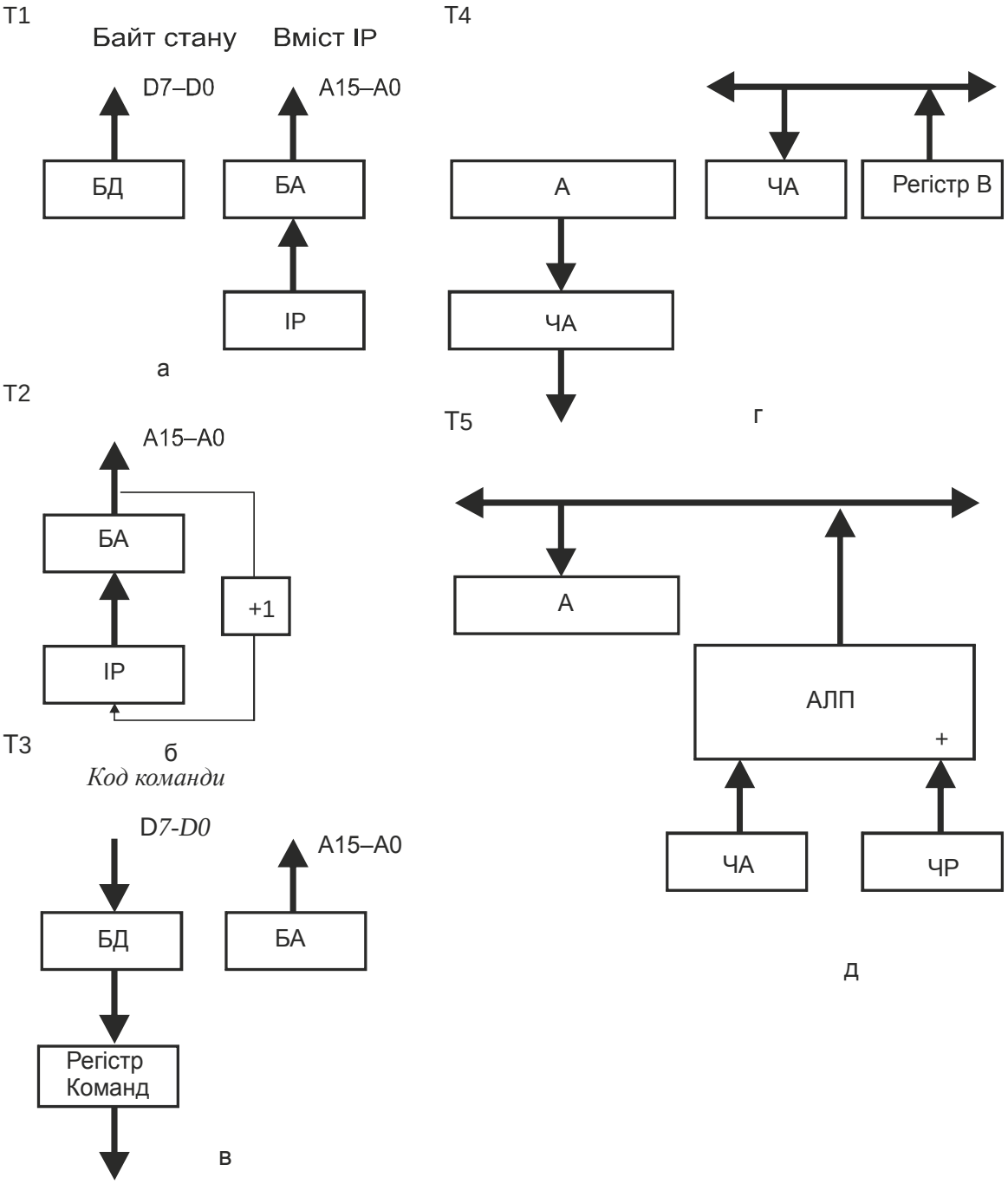


Рис. 3.6. Виконання команди *ADD B* за мікрооперaціями:
*а, б, в, г, д – T*1, *T*2, *T*3, *T*4, *T*5 відповідно

Система переривання може встановити активний рівень сигналу на лінії *INT* у будь-який момент виконання програми, однак приймання його синхронізується таким чином. За наявності сигналу дозволу переривання (тригер дозволу переривань установлений в одиницю) тригер переривань установлюється в останньому такті останнього машинного циклу команди, протягом виконання якої надійшов запит. Це дає можливість процесору завершити виконання команди перед тим, як почнеться

обробка переривання. Якщо переривання не дозволено, тобто тригер дозволу переривань скинуто до нуля, запит переривань із входу *INT* ігнорується. Часову діаграму роботи процесора показано на рис. 3.7, *а, б*.

Після прийняття сигналу запиту переривання процесор переходить до виконання циклу ПЕРЕРИВАННЯ, що складається з трьох машинних циклів. Перший з них *M*1 (рис. 3.7, *а*) призначений для приймання команди *RST V*, а два інші – *M*2 і *M*3 – для зберігання адреси повернення у стек (рис. 3.7, *б*). У циклі *M*1 у першому такті *T*1 у байті стану формується сигнал підтвердження переривання, який використовується для керування читанням команди *RST V*. У такті *T*3 процесор приймає по шині даних байт команди *RST V*, що формується системою переривання. У тактах *T*4, *T*5 циклу ПЕРЕРИВАННЯ здійснюється формування адреси першої комірки стека, відведеної для зберігання адреси повернення з підпрограми обслуговування запиту переривання. У циклах *M*2 та *M*3 здійснюється завантаження адреси повернення (вмісту вказівника команд) у стек. У наступному циклі витягається перша команда підпрограми обслуговування переривання за адресою, вказаною в команді *RST V*. Наступне керування МП покладається на підпрограму.

В окремому випадку за допомогою підпрограми здійснюється зберігання вмісту основних робочих регістрів процесора, керування тригером дозволу переривання, відтворення вмісту основних робочих регістрів та повернення до основної програми (відновлення вмісту вказівника команд).

Захоплення шин. Режим захоплення шин використовують для організації виконання операцій ПДП. Для цього процесор має вхідний вивід *HOLD* запиту захоплення шин та вихідний вивід *HLDA* підтвердження захоплення. Зовнішній пристрій запитує режим ПДП сигналом високого рівня на лінії *HOLD*. При цьому процесор зупиняє виконання операцій і від’єднується від зовнішніх шин даних та адреси. Лінії шин переходять у високоімпедансний стан.

Процесор підтверджує прийняття запиту ПДП встановленням високого потенціалу на виході підтвердження захоплення *HLDA*. Доки діє сигнал на вході *HOLD*, шини процесора знаходяться в розпорядженні зовнішнього пристрою, який надіслав запит ПДП. Сигнали керування обміном інформацією між зовнішнім пристроєм та пам’яттю формуються спеціальною ВІС – контролером ПДП.

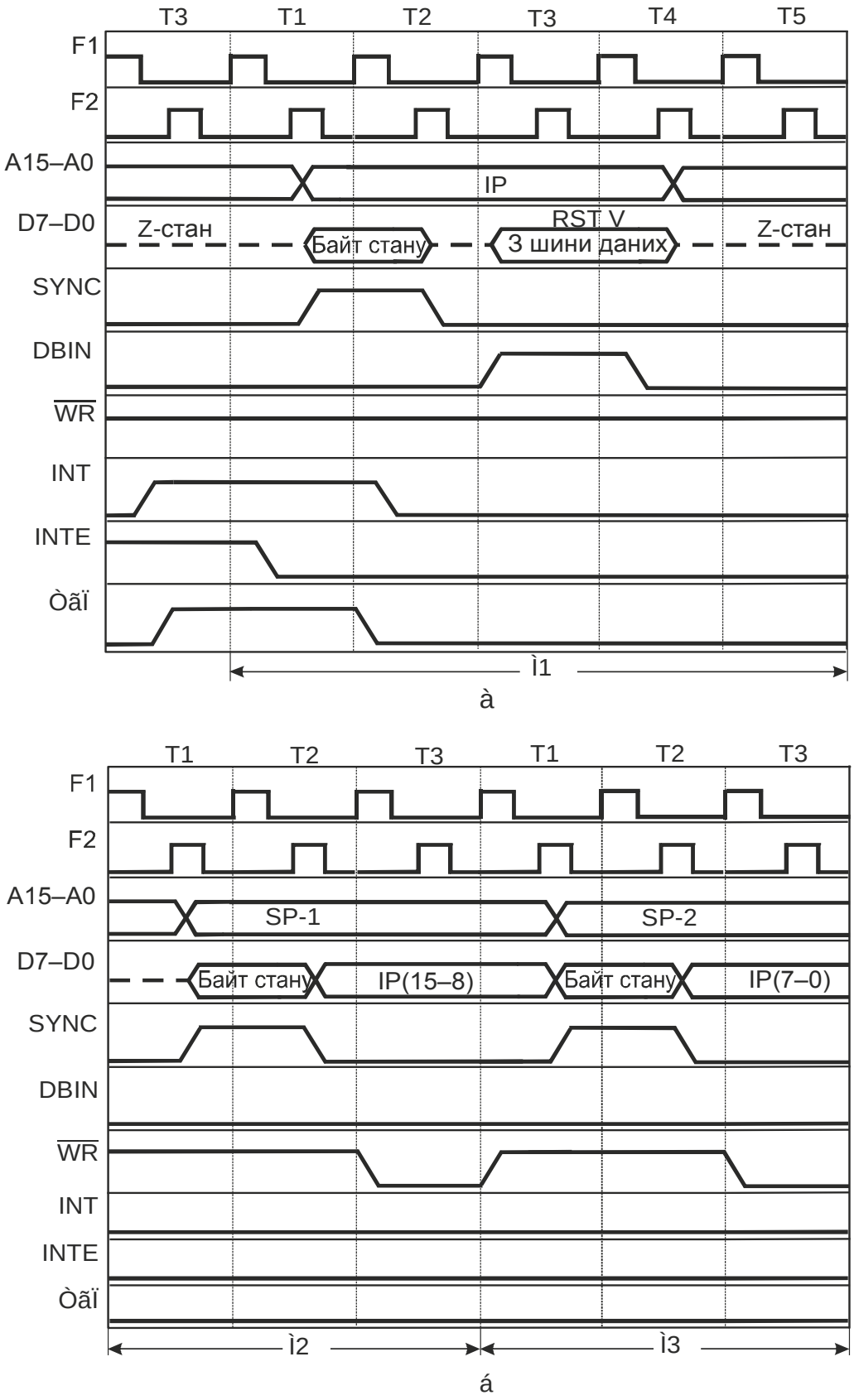


Рис. 3.7. Часові діаграми циклу ПЕРЕРИВАННЯ:
а – машинний цикл *М*1; *б* – машинні цикли *М*2 і *М*3; ТгП – тригер переривань

Часову діаграму роботи процесора у режимі захоплення шин у циклі ЧИТАННЯ ПАМ'ЯТІ зображено на рис. 3.8. Сигнал *HOLD* сприймається процесором у такті *T*2. За наявності сигналу готовності зовнішнього пристрою на вході *READY* встановлюється високий рівень на вході внутрішнього тригера захоплення, завдяки чому по фронту наступного імпульсу *F*1 вихідний сигнал МП *HLDA* перемикається у стан логічної одиниці. Під час виконання циклів читання або введення процесор підтверджує захоплення на початку такту *T*3 по закінченні читання. У циклах записування та виведення це здійснюється в такті, наступному за *T*3, по закінченні запису. В обох випадках шини процесора переводяться у високоімпедансний стан по фронту імпульсу *F*2, наступному за імпульсом *F*1, за яким виконувалося перемикання виходу *HLDA*.

Процесор виходить з режиму захоплення таким чином. Після закінчення асинхронного сигналу запиту захоплення на вході *HOLD* імпульсом *F*2 скидається тригер захоплення, завдяки чому по передньому фронту імпульсу *F*1 на виході *HLDA* підтвердження захоплення формується сигнал низького потенціалу. Процесор переходить до виконання наступного машинного циклу.

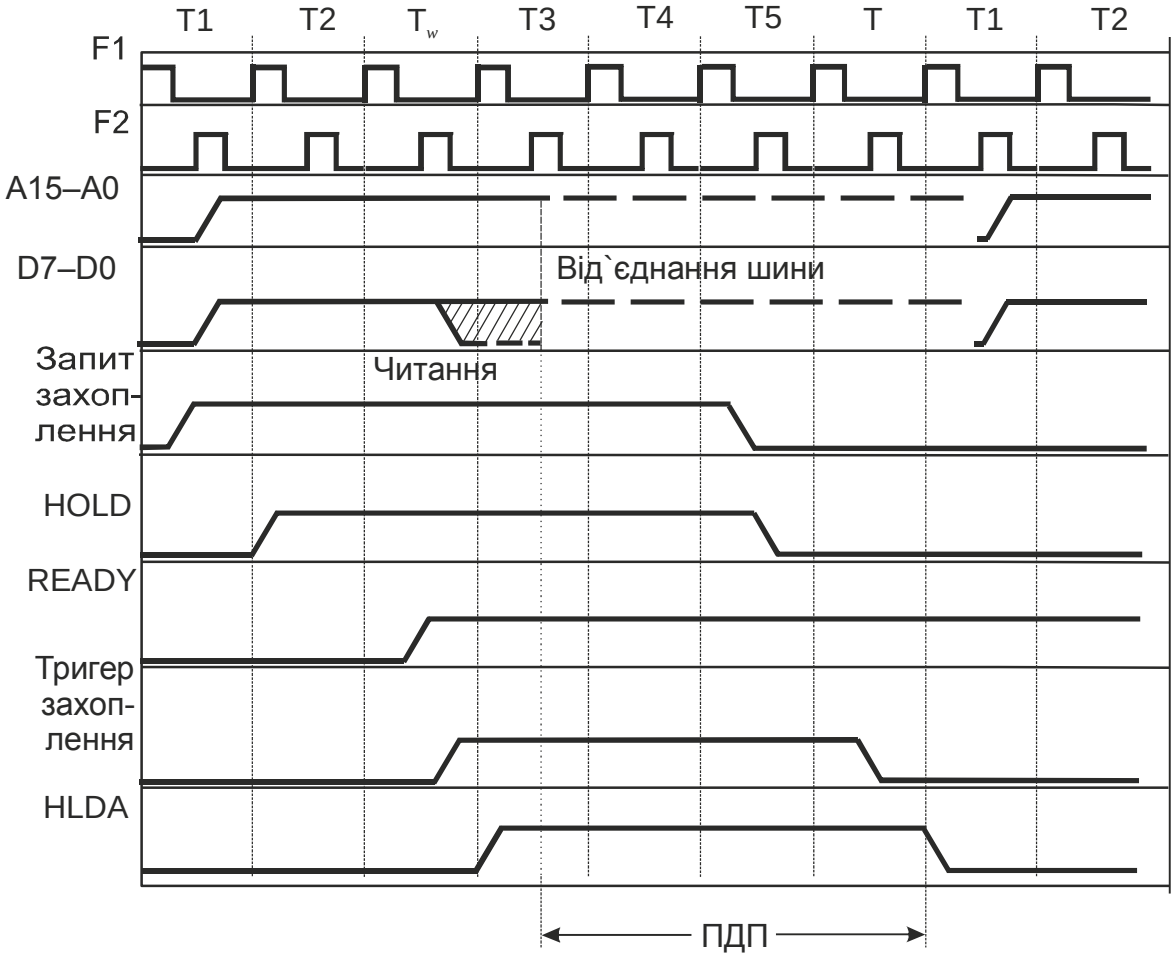


Рис. 3.8. Цикл ЧИТАННЯ ПАМ'ЯТІ в режимі ПДП

Зупин. Процесор входить у режим зупину при виконанні команди зупину *HLT*. Виконання дій МП ілюструється часовими діаграмами (рис. 3.9). Перехід у режим зупину виконується за два машинні цикли. У першому машинному циклі ВИБІРКА здійснюється зчитування з пам'яті першого байта команди *HLT*. У другому машинному циклі ЗУПИН після закінчення такту *T*2 процесор переходить у режим зупину, при якому шини даних і адреси переходять у високоімпедансний стан, а процесор виконує такти очікування T_w. Режим зупину підтверджується бітом *D*3 у байті стану, який видається також у такті *T*2 (див. табл. 3.1).

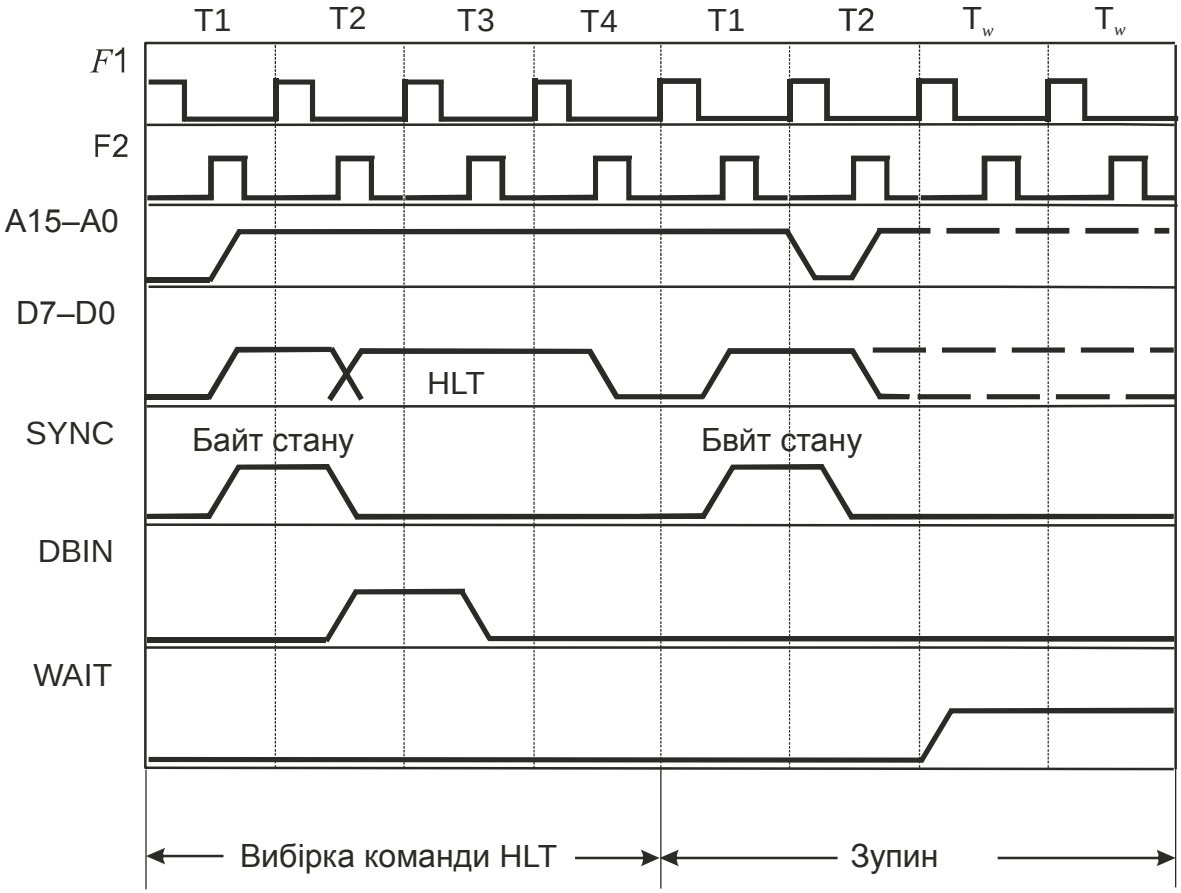


Рис. 3.9. Цикл ЗУПИН

Вийти із режиму зупину можна трьома способами:

– поданням сигналу на лінію *RESET* (при цьому вказівник команд набуває нульового значення і процесор переходить до машинного циклу ВИБІРКА команди за першою адресою);
– поданням сигналу на вхід *HOLD*, завдяки чому процесор переходить до виконання циклу ЗАХОПЛЕННЯ. Після закінчення цього сигналу процесор входить у режим зупину за переднім фронтом імпульсу *F*1;
– поданням сигналу переривання (за наявності сигналу дозволу переривань на виході *INTE*), завдяки чому процесор по фронту імпульсу *F*1 переходить у режим *T*1 машинного циклу ПЕРЕРИВАННЯ.

Для реалізації цієї можливості треба перед початком режиму зупину забезпечити встановлення тригера дозволу переривань виконанням команди дозволу переривання *E*I.

Обробку запитів захоплення шин і переривання при зупині ілюструє рис. 3.10. Нехай процесор знаходиться в режимі, тоді лінії шин адрес і даних знаходяться у високоімпедансному стані та процесор виконує такти T_w.

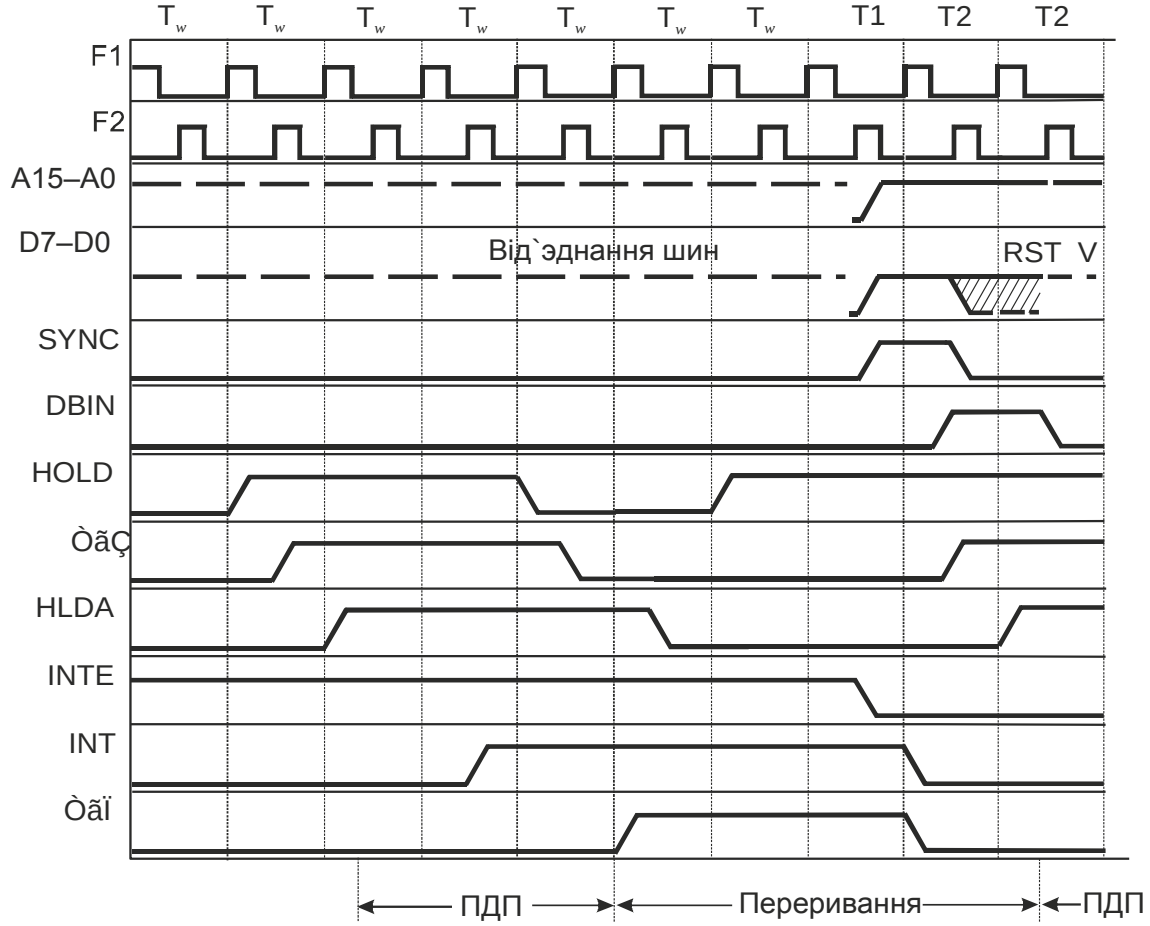


Рис. 3.10. Цикли ЗАХОПЛЕННЯ шин та ПЕРЕРИВАННЯ ПРИ ЗУПИНІ

Із надходженням запиту захоплення шин *HOLD* у наступному такті встановлюється тригер захоплення (ТгЗ). Потім процесор видає сигнал підтвердження захоплення *HLDA* і здійснює ПДП. Якщо виконується послідовність дій у режимі ПДП, то запити переривання не сприймаються процесором до закінчення обміну. Якщо переривання дозволено і надійшов запит переривання *INT*, то після закінчення режиму ПДП МП переходить до машинного циклу ПЕРЕРИВАННЯ ПРИ ЗУПИНІ. Якщо в цьому циклі надійшов запит захоплення *HOLD*, він ігнорується МП до завершення виконання циклу читання команди *RST*. Потім МП переходить до режиму ПДП. Послідовність дій МП у режимі переривання завершується по закінченні режиму ПДП.

Увімкнення мікропроцесора. З поданням напруги живлення процесор починає функціонувати. Напругу живлення МП необхідно вмикати разом з поданням сигналу на лінію *RESET*. Тривалість цього сигналу має бути не менше трьох періодів імпульсів синхронізації. За сигналом *RESET* вміст вказівника команд набуває нульового значення, і процесор починає виконувати дії, що відповідають машинному циклу ВИБІРКА. У результаті МП починає виконувати команду, код якої розміщено у нульовій комірці пам'яті. Отже, запуск програми починається за сигналом на лінії *RESET*. Першою командою програми має бути команда безумовного переходу *JMP ADR*, яка здійснює перехід до команди, що знаходиться у довільному місці пам'яті.

Вміст регістрів загального призначення і регістра прапорців залишається невизначеним доти, доки його не встановлять команди програми.

Контрольні запитання

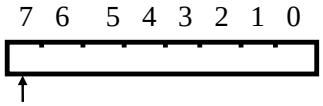
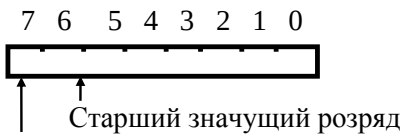
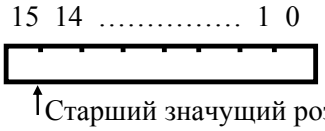
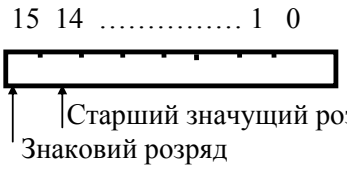
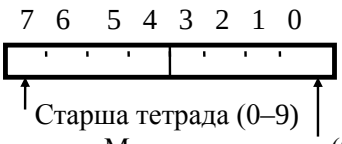
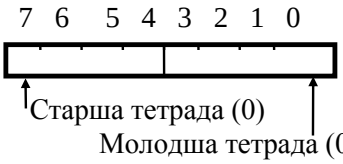
1. Вкажіть призначення: регістра команд; акумулятора; блока РЗП.
2. Вкажіть призначення прапорців у МП.
3. Які керувальні сигнали використовує МП для забезпечення роботи з пристроями, швидкодія яких значно менша від швидкодії самого МП?
4. Які керувальні сигнали використовує МП для забезпечення роботи з пристроями, швидкодія яких більше від швидкодії МП?
5. Які дії виконує процесор за сигналом *RESET*?
6. Які режими адресації використовуються у 8-розрядних МП?
7. Яку інформацію несе байт стану?
8. Які дії відбуваються в тактах *T*1 і *T*2 будь-якого машинного циклу?
9. Назвіть можливі типи машинних циклів.
10. Назвіть послідовність дій процесора в машинному циклі ПІДТВЕРДЖЕННЯ ПЕРЕРИВАННЯ.
11. Які дії виконує МП при надходженні запиту переривання в режимі переривання?
12. Які дії виконує МП після увімкання напруги живлення? Назвіть можливі способи виходу з режиму зупину.

3.2. Однокристальні 16-розрядні мікропроцесори

До 16-розрядних МП першого покоління належать МП *i*8086/*i*8088 та *i*80186/*i*80188, до МП другого покоління – *i*80286. Велика інтегральна схема МП *i*8086 геометричними розмірами 5,5×5,5 мм має 40 контактів, містить близько 29000 транзисторів і споживає 1,7 Вт від джерела живлення +5 В, тактова частота – 5, 8 або 10 МГц.

Мікропроцесор виконує операції над 8- та 16-розрядними даними, наведеними у двійковому або двійково-десятковому вигляді, може обробляти окремі біти, а також рядки або масиви даних. Він має вбудовані апаратні засоби множення та ділення. Формати даних і виконувані операції наведено в табл. 3.2.

Таблиця 3.2. **Формати даних і операцій, виконуваних МП *i*8086**

Тип даних	Формат	Діапазон	Операції
Байт без знака		0–255	Додавання, віднімання, множення, ділення
Байт зі знаком		Від –128 до +127	Те саме
Слово без знака		0–65 535	“
Слово зі знаком		Від –32 768 до +32 767	“
Упаковане двійково-десяткове число		0–99	Додавання, віднімання з корекцією
Розпаковане двійково-десяткове число		0–9	Додавання, віднімання, множення, ділення з корекцією

Примітка. Знакові числа подають у додатковому коді.

Мікропроцесор має внутрішній надоперативний запам’ятовувальний пристрій (НОЗП) ємністю 14×16 байт. Шина адреси 20-розрядна, що дозволяє безпосередньо адресувати до $2^{20} = 1048576$ комірок пам’яті (1 Мбайт).

Простір адрес введення-виведення становить 64 кбайт. У ВІС *i*8086 реалізовано багаторівневу векторну систему переривань з кількістю векторів до 256. Передбачено також організацію ПДП, за яким МП припиняє роботу і переводить у третій стан шини адреси, даних та керування.

Середній час виконання команди займає 12 тактів. Особливістю МП *i*8086 є можливість часткової реконфігурації апаратної частини для забезпечення роботи в двох режимах – мінімальному і максимальному. Режими роботи задаються апаратно. У мінімальному режимі, використовуваному для побудови однопроцесорних систем, МП самостійно формує всі сигнали керування внутрішнім системним інтерфейсом. У максимальному режимі, використовуваному для побудови мультипроцесорних систем, МП формує на лініях стану двійковий код, який залежить від типу циклу шини. Відповідно до цього коду системний контролер К1810ВГ88 формує сигнали керування шиною. Контакти, які звільнилися у результаті кодування інформації, використовуються для керування мультипроцесорним режимом. При використанні арифметичного співпроцесора потрібно обирати максимальний режим.

Структурна схема. У МП *i*8086 застосовано конвеєрну архітектуру, що дозволяє суміщувати у часі цикли виконання команди та вибірки з пам'яті кодів наступних команд. Це досягається паралельною роботою двох порівняно незалежних пристроїв – операційного пристрою та шинного інтерфейсу. Структурну схему МП *i*8086 показано на рис. 3.11. Операційний пристрій виконує команду, а шинний інтерфейс здійснює взаємодію із зовнішньою шиною: виставляє адреси, зчитує коди команд, операнди, записує результати обчислень у пам'ять або пристрої введення-виведення.

Операційний пристрій складається з РЗП, призначених для зберігання проміжних результатів – даних та адрес; АЛП з буферними регістрами; регістра прапорців; блока керування та синхронізації (БК та С), який дешифрує коди команд і генерує керувальні сигнали для всіх блоків схеми МП. Шинний інтерфейс складається з шестибайтової регістрової пам'яті, яка називається чергою команд, чотирьох сегментних регістрів: *CS*, *DS*, *ES*, *SS*, вказівника команд *IP*, суматора, а також допоміжних регістрів зв'язку і буфера шин (БШ) адреси/даних. Черга команд працює за принципом *FIFO* (*First Input – First Output*, тобто перший прийшов – перший пішов) і зберігає на виході порядок надходження команд. Довжина черги 6 байт.

Коли операційний пристрій зайнятий виконанням команди, шинний інтерфейс самостійно ініціює випереджаючу вибірку кодів команд з пам'яті у чергу команд. Вибирання з пам'яті чергового командного слова здійснюється тоді, коли в черзі виявляється два вільні байти. Черга збільшує швидкодію процесора у випадку послідовного виконання команд. У разі вибирання команд переходів, викликів і повернень з підпрограм та обробленні запитів переривань черга команд скидається і вибирання починається з нового місця програмної пам'яті.

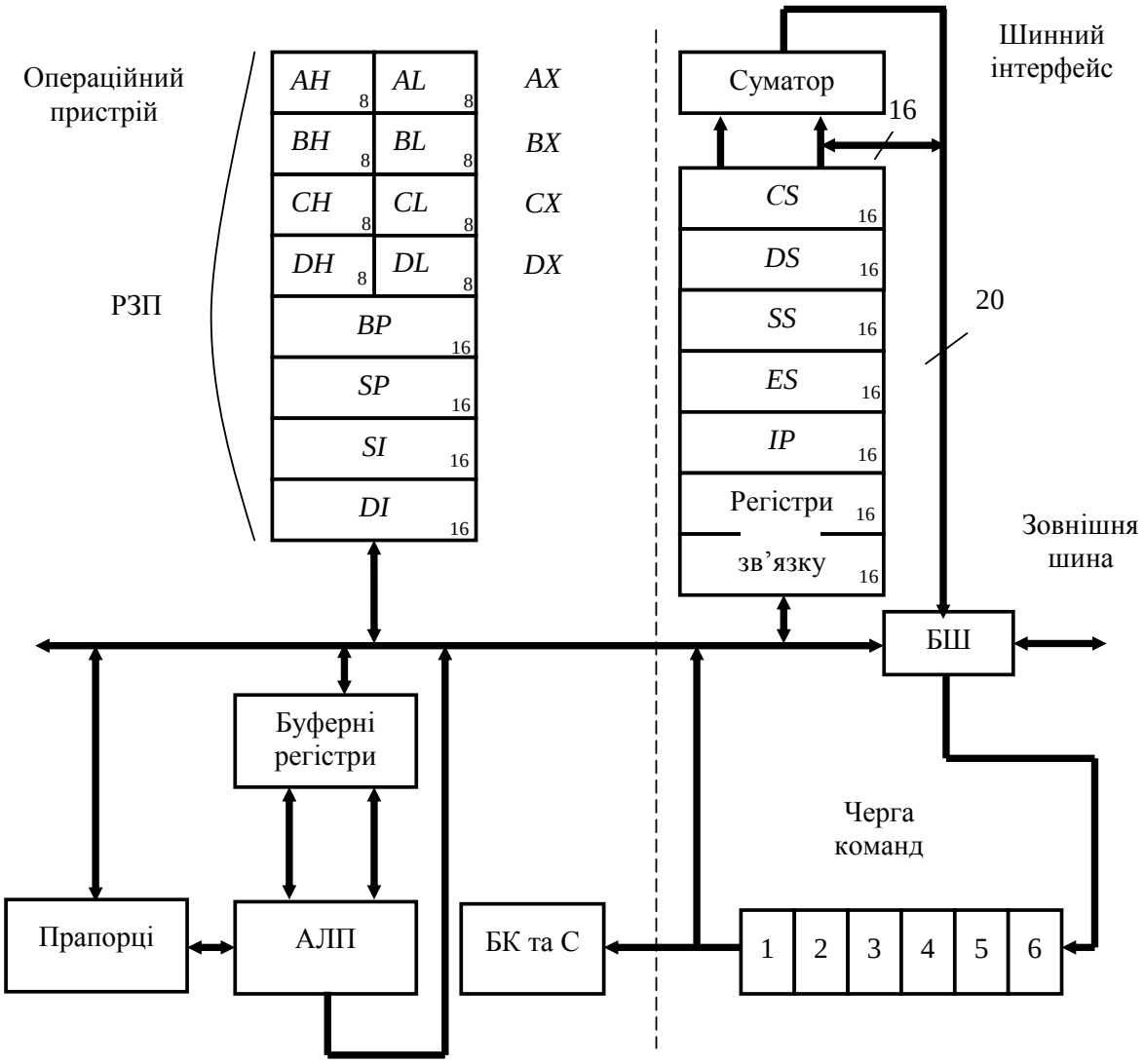


Рис. 3.11. Структурна схема мікропроцесора *i*8086

Ще одним із завдань шинного інтерфейсу є формування фізичної 20-розрядної адреси із двох 16-розрядних слів. Першим словом є вміст одного з сегментних регістрів *CS*, *SS*, *DS*, *ES*, а друге слово залежить від типу адресації операнда або коду команди. Складання 16-розрядних слів відбувається зі зміщенням на 4 розряди і здійснюється за допомогою суматора, що входить до складу шинного інтерфейсу.

Призначення контактів. Умовне графічне зображення МП *i*8086 ілюструє рис. 3.12.

Призначення контактів ВІС залежить від режиму роботи МП. Вісім контактів має подвійне позначення, причому позначення в дужках відповідають максимальному режиму.

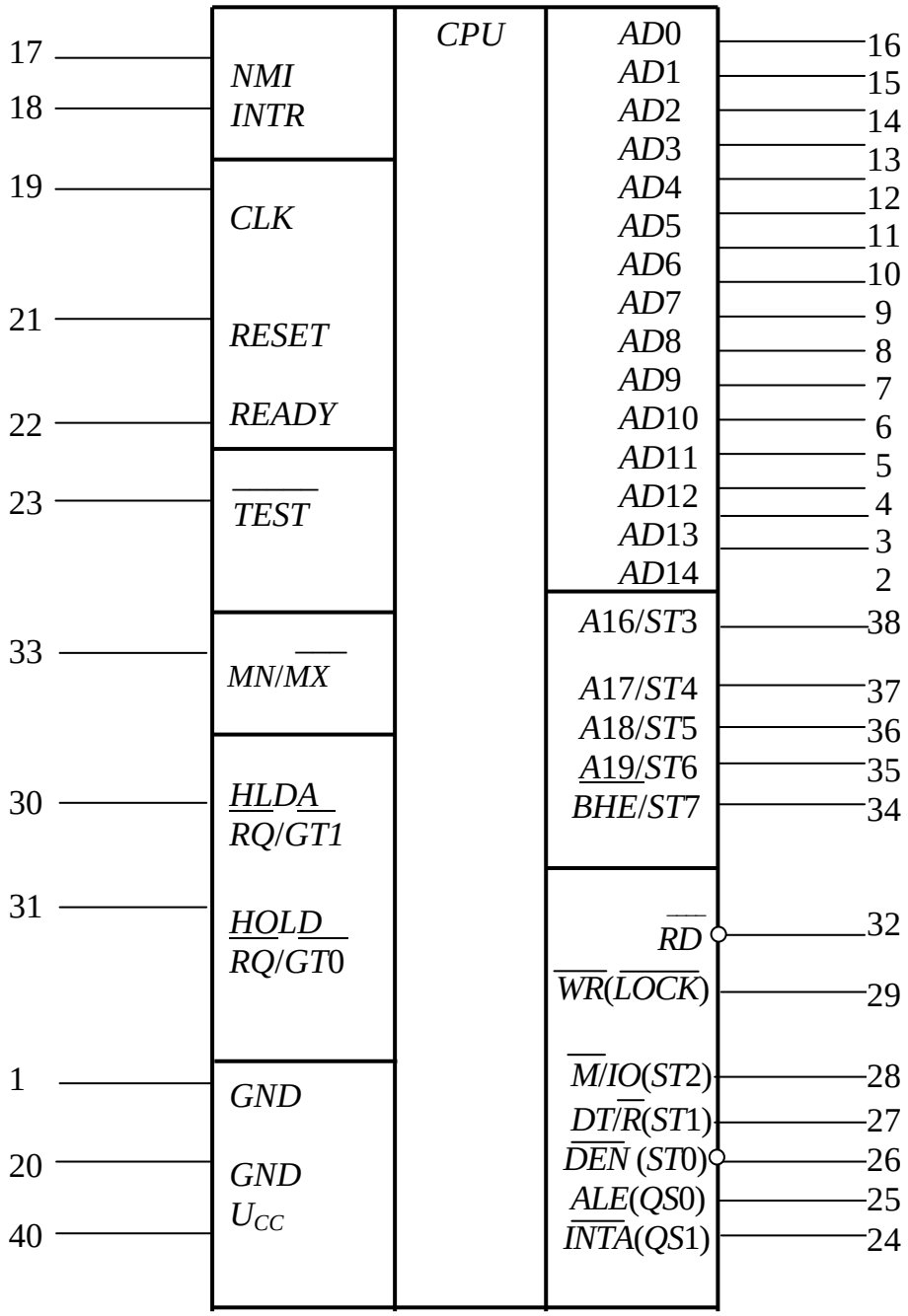


Рис. 3.12. Графічне зображення ВІС МП *i*8086

У табл. 3.3 наведено призначення контактів МП, однакових для обох режимів, у табл. 3.4 – призначення контактів, які використовуються тільки в мінімальному режимі, у табл. 3.5 – призначення контактів, які використовуються тільки в максимальному режимі. Літерою *Z* позначено тристабільні виходи, що переводяться у третій високоімпедансний стан при переході МП у режим захоплення; у дужках наведено альтернативні позначення контактів, які трапляються в літературі.

***Таблиця 3.3*. Призначення контактів МП *i*8086 для мінімального і максимального режимів**

Позначення	Призначення	Тип
1	2	3
*AD*15–*AD*0	*Address/data* – мультиплексна двонапрямлена шина адреси/даних (*ADB – Address Data Bus*), за якою з розподілом у часі передаються адреси і дані. Адреси передаються в першому такті циклу шини і супроводжуються сигналом *ALE*, а дані – у другій половині циклу шини і супроводжуються сигналом *DEN*	Вхід/ вихід (*Z*)
$\overline{BHE}$ /*ST*7	*Byte High Enable/Status* 7 – вихідний сигнал дозволу старшого байта/сигнал стану. У першому такті циклу водночас з адресною інформацією передається сигнал $\overline{BHE}$. Активний (*L*) рівень $\overline{BHE}$ означає, що по старшій половині *AD*15–*AD*8 шини адреси/даних передаються 8-розрядні дані. Сигнал $\overline{BHE}$ використовується для дозволу доступу до старшого банка пам'яті або до зовнішнього пристрою з байтовою організацією, підключеного до старшої половини шини даних. В інших тактах формується сигнал стану *ST*7	Вихід (Z)
$\overline{RD}$	*Read* – вихідний сигнал читання. Вказує на те, що МП виконує цикл читання	Вихід (*Z*)
READY	*Ready* – вхідний сигнал готовності, який підтверджує, що комірка пам'яті або пристрій введення-виведення, який адресується у команді, готовий до взаємодії з МП при передачі даних	Вхід
INTR	*Interrupt Request* – вхідний сигнал запиту (при *H*-рівні) маскованого переривання. Якщо переривання дозволено, МП переходить до підпрограми обробки переривання. В іншому випадку МП ігнорує цей сигнал	Вхід
RESET (*CLR*)	*Reset* (*Clear*) – сигнал апаратного скидання (при *H*-рівні). Переводить МП у початковий стан, при якому скинуті сегментні регістри (крім *CS*, усі розряди якого встановлюються в одиничний стан), вказівник команд *IP*, усі прапорці, регістри черги команд і всі внутрішні тригери у пристрої керування. Сигнал *RESET* не впливає на стан загальних регістрів. Під час дії сигналу *RESET* усі виходи, що мають три стани, переводяться у третій стан; виходи, що мають два стани, стають пасивними. Мінімальна тривалість сигналу *RESET* при першому ввімкненні МП становить 50 мкс, а при повторному запуску – 4 такти синхронізації, тобто 0,8 мкс при тактовій частоті 5 МГц. Після закінчення сигналу *RESET* починається цикл вибірки команди з пам'яті з адресою 0*FFFFH*:0000	Вхід
CLK, (*CLC*)	*Clock* – вхідні тактові імпульси, які забезпечують синхронізацію роботи МП	Вхід
MN/ $\overline{MX}$	*Minimum/maximum* – вхід сигналу вибору мінімального або максимального режимів. Сигнал на цьому вході визначає режим роботи МП: 1 – мінімальний, 0 – максимальний	Вхід

Продовження табл. 3.3

1	2	3
$\overline{TEST}$	*Test* – вхідний сигнал перевірки. Сигнал використовується разом з командою очікування *WAIT*, виконуючи яку, МП перевіряє рівень сигналу $\overline{TEST}$. Якщо $\overline{TEST} = 0$, МП переходить до виконання наступної після *WAIT* команди. Якщо $\overline{TEST} = 1$, МП знаходиться у стані очікування, виконує холості такти і періодично, з інтервалом $5T_{CLK}$, перевіряє значення сигналу $\overline{TEST}$	Вхід

Таблиця 3.4. **Призначення контактів МП *i*8086 у мінімальному режимі**

Позначення	Призначення	Тип
$\overline{INTA}$	*Interrupt Acknowledge* – вихідний сигнал підтвердження переривання, що визначає читання вектора переривання	Вихід
ALE	*Address Latch Enable* – вихідний сигнал дозволу фіксації адреси; видається на початку кожного циклу шини і використовується для запису адреси в регістр-фіксатор	Вихід
$\overline{DEN}$ ($\overline{DE}$)	*Data Enable* – вихідний сигнал дозволу даних, що визначає появу даних на шині адреси/даних	Вихід (*Z*)
$DT / \overline{R}$ ($OP / \overline{IP}$)	*Data Transmit/Receive (Output-Input)* – вихідний сигнал передавання/приймання даних; визначає напрям передачі даних по *ADB*. Призначений для керування шинними формувачами і діє протягом усього циклу шини	Вихід (*Z*)
$M / \overline{IO}$	*Memory/Input-Output* – вихідний сигнал ознаки звернення до пам'яті ($M/\overline{IO} = 1$) або зовнішнього пристрою ($M/\overline{IO} = 0$). Використовується для розподілу адресного простору пам'яті та введення-виведення	Вихід (*Z*)
$\overline{WR}$	*Write* – вихід сигналу запису. Строб, який вказує на те, що МП виконує цикл запису в пам'ять або зовнішній пристрій і супроводжує дані, що видаються МП на шину даних	Вихід (*Z*)
HOLD	*Hold* – вхід сигналу запиту захоплення шин від зовнішнього пристрою або контролера ПДП	Вхід
HLDA	*Hold Acknowledge* – вихідний сигнал підтвердження захоплення. Сигнал вказує на те, що МП перевів свої шини адреси/даних, адреси/стану і керування у *Z*-стан	Вихід

Таблиця 3.5. **Призначення контактів МП *i*8086 у максимальному режимі**

Позначення	Призначення	Тип
1	2	3
*ST*2, *ST*1, *ST*0 (*S*2–*S*0)	Вихідні сигнали ліній стану; характеризують тип виконуваного циклу шини; використовуються для формування керувальних сигналів	Вихід (*Z*)
$\overline{RQ} / \overline{GT}0$ $\overline{RQ} / \overline{GT}1$ ($\overline{RQ} / \overline{E}0$) ($\overline{RQ} / \overline{E}1$)	*Request*/*Grant* (*Request/Enable)* – два вхідні/вихідні сигнали запиту/надання локальної шини; використовуються для зв'язку з іншими процесорами, а саме з арифметичним співпроцесором. Лінія *RQ/GT*1 має менший пріоритет	Вхід/ вихід

Продовження табл. 3.5

1	2	3
LOCK	*Lock* – вихідний сигнал блокування (зайнятості) шини, сигнал монополізації керування шиною; формується під час виконання команди за префіксом *LOCK* й інформує інші процесори і пристрої про те, що вони не повинні запрошувати системну шину	Вихід
*QS*1, *QS*0	*Queue Status* – два вихідні сигнали стану черги; ідентифікують стан внутрішньої шестибайтової черги команд МП і діють протягом такту синхронізації після виконання операції над чергою. Сигнали *QS*1, *QS*0 призначені для співпроцесора, що контролює шину адреси/даних, фіксує момент, коли з програмної пам'яті вибирається призначена для нього команда з префіксом *ESC*, а після цього стежить за чергою команд і визначає момент, коли ця команда має виконуватися	Вихід

Лінії стану. Лінії *ST*2–*ST*0 – виходи сигналів стану – ідентифікують тип циклу шини, що виконується згідно з табл. 3.6. *Циклом шини* називають звернення до комірки пам'яті або зовнішнього пристрою. Це визначення збігається з визначенням машинного циклу для 8-розрядних процесорів (див. підрозд. 3.1). Однак у 16-розрядних процесорах цикл шини може ініціювати не тільки МП, але й арифметичний співпроцесор або спеціалізований процесор введення-виведення.

Початок циклу визначається переходом ліній стану *ST*2–*ST*0 з пасивного (111) в активний стан, а кінець – зворотним переходом у пасивний стан. Сигнали *ST*2–*ST*0 подаються на входи контролера шини *i*8288, що дешифрує їх і формує сигнали керування системною шиною $\overline{IOR}$, $\overline{IOW}$, $\overline{MEMR}$, $\overline{MEMW}$, $\overline{INTA}$, *ALE*, $\overline{DEN}$.

***Таблиця 3.6*. Ідентифікація типу циклу шини**

Лінії стану			Тип циклу шини
*ST*2	*ST*1	*ST*0	
0	0	0	ПІДТВЕРДЖЕННЯ ПЕРЕРИВАННЯ ($\overline{INTA}$)
0	0	1	ВВЕДЕННЯ (читання зовнішнього пристрою)
0	1	0	ВИВЕДЕННЯ (запис у зовнішній пристрій)
0	1	1	ЗУПИН
1	0	0	ВИБІРКА команди
1	0	1	ЧИТАННЯ ПАМ'ЯТІ
1	1	0	ЗАПИС У ПАМ'ЯТЬ
1	1	1	Циклу шини немає

Сигнал *ST*2 є логічним еквівалентом сигналу $M/\overline{IO}$, а *ST*1 – сигналу $DT/\overline{R}$ (див. підрозд. 3.1). Сигнали *ST*4, *ST*3 вказують, який сегментний регістр використовується у даному циклі (табл. 3.7) і можуть використовуватися для розширення адресного простору системи. У цьому випадку окремий банк пам'яті ємністю 1 Мбайт виділяється кожному із чотирьох сегментів. До виводів МП *S*4, *S*3 підключають дешифратор, що вибирає

відповідний банк пам'яті. Таке приймання забезпечує розширення адресної пам'яті до 4 Мбайт і захист від помилкового запису у сегмент, що перекривається з іншими сегментами.

Таблиця 3.7. **Ідентифікація сегментного регістра**

*ST*4	*ST*3	Сегментний регістр
0	0	*ES*
0	1	*SS*
1	0	*CS*
1	1	*DS*

Сигнал *ST*5 відповідає стану прапорця *IF* дозволу переривань: 0 – переривання заборонено, 1 – переривання дозволено. Сигнали *ST*6, *ST*7 не використовуються і зарезервовані для наступних моделей МП.

Ідентифікація стану черги команд здіснюється за допомогою сигналів *QS*1, *QS*2 (див. табл. 3.5). Значення цих ліній визначає операцію над чергою команд відповідно до табл. 3.8.

Таблиця 3.8. **Ідентифікація стану черги команд**

*QS*1	*QS*0	Операція над чергою
0	0	Операції немає, в останньому такті не було вибірки із черги
0	1	З черги вибрано перший байт команд
1	0	Черга порожня; була спустошена командою передачі керування
1	1	З черги вибрано наступний байт команди

Лінії запиту/надання локальної шини. Двонапрямлені лінії $\overline{RQ}/\overline{GT}0, \overline{RQ}/\overline{GT}1$ використовуються для передачі імпульсних сигналів запиту/дозволу доступу до локальної шини (каналу). Процес доступу до шини здійснюється в такому порядку: спочатку пристрій, увімкнений до локальної шини, який потребує доступу до загальних ресурсів, формує імпульс тривалістю 1 такт; після цього наприкінці поточного циклу МП видає відповідний імпульс, що підтверджує можливість доступу до локальної шини. У наступному такті МП переводить шини адреси/даних і керування у високоімпедансний стан і відключається від каналу. По закінченні роботи з каналом пристрій видає на ту ж лінію третій імпульс, що вказує на закінчення захоплення каналу. У наступному такті МП відновлює керування шиною і продовжує обчислення. Усі три імпульси мають однакову тривалість і низький активний рівень. Сигнали на лініях незалежні, однак лінія $\overline{RQ}/\overline{GT}0$ має вищий пріоритет, ніж лінія $\overline{RQ}/\overline{GT}1$, при одночасному надходженні запитів. Кожна з двох розглянутих ліній використовується для встановлення режиму захоплення шин і еквівалентна парі *HOLD* і *HLDA* МП *i*8086 у мінімальному режимі (див. підрозд. 3.1).

Організація пам'яті. Пам'ять являє собою масив ємністю 1 Мбайт, тобто 2^{20} 8-розрядних комірок (рис. 3.13). У пам'яті зберігаються як байти, так і двобайтові слова. Слова розміщуються у двох сусідніх комірках

пам’яті: старший байт зберігається у комірці зі старшою адресою, молодший – з молодшою. Адресою слова вважається адреса його молодшого байта. На рис. 3.13 показано приклад, коли з адресою 00000 зберігається байт 35*H*, а з адресою 00001 – слово 784*AH*. Початкові (00000*H*–003*FFH*) і кінцеві (*FFFF*0*H*–*FFFFFH*) адреси зарезервовані для системи переривань та початкового встановлення відповідно.

Організація пам’яті, коли кожній адресі відповідає вміст однієї комірки пам’яті (рис. 3.13), називається *лінійною.* У МП *і*8086 застосовано сегментну організацію пам’яті, яка характеризується тим, що програмно доступною є не вся пам’ять, а лише деякі сегменти, тобто області пам’яті. Усередині сегмента використовують лінійну адресацію.

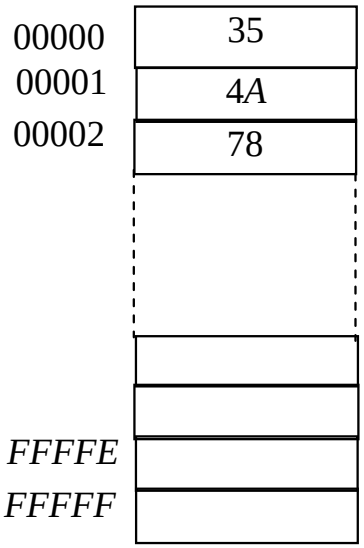


Рис. 3.13. Програмна модель пам’яті

Упровадження сегментної організації можна пояснити таким чином. Мікропроцесор *і*8086 являє собою 16-розрядний процесор, тобто він має 16-розрядну внутрішню шину, 16-розрядні регістри і суматори. Прагнення розробників ВІС адресувати якомога більший масив пам’яті зумовило використання 20-розрядної шини даних. Для порівняння: 16-розрядна шина адреси дозволяє адресувати 2^{16} = 64 кбайт; 20-розрядна – 2^{20}= 1 Мбайт.

Для формування 20-розрядної адреси у 16-розрядному процесорі використовують інформацію двох 16-розрядних регістрів. У МП *і*8086 20-розрядна адреса формується з двох 16-розрядних адрес, які називають *логічними.* Перша логічна адреса, доповнена праворуч чотирма нулями, являє собою початкову адресу сегмента ємністю 64 кбайт. Друга логічна адреса визначає зміщення у сегменті, тобто відстань від початку сегмента до адресованої комірки. Якщо вона дорівнює 0000, то адресується перша комірка сегмента, якщо *FFFFH* – то остання. Отже, логічний адресний простір розподілено на блоки суміжних адрес розміром 64 кбайт, тобто сегменти.

Такий підхід до організації пам’яті зручний ще й тому, що пам’ять зазвичай логічно поділяється на області коду (програмної пам’яті), даних і стека. Фізична 20-розрядна адреса комірки пам’яті формується з двох 16-розрядних адрес – адреси сегмента *Seg* і виконавчої адреси *EA* (*Executive Address*), які додаються зі зміщенням на чотири розряди (рис. 3.14).

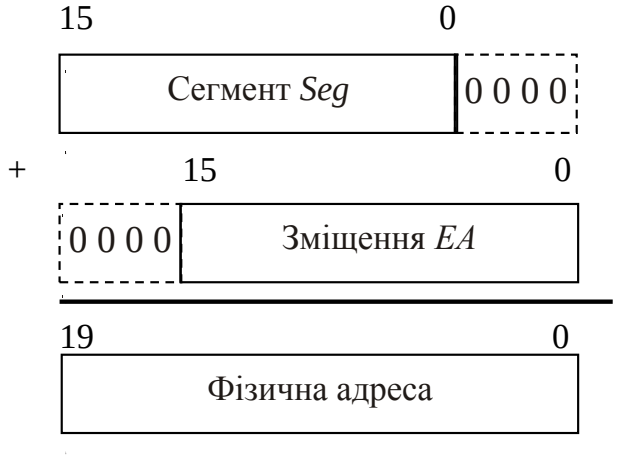


Рис. 3.14. Формування фізичної адреси

Зміщення адреси сегмента на 4 розряди ліворуч еквівалентне його множенню на 2^4. Тоді фізична адреса дорівнює $16 \times Seg + EA$. Як перша логічна адреса *Seg* використовується вміст одного із чотирьох сегментних регістрів: *CS* (*Code Segment* – сегмент кодів), *DS* (*Data Segment* – сегмент даних), *ES* (*Extended Segment* – додатковий сегмент даних), *SS* (*Stack Segment* – сегмент стека). Друга логічна адреса *EA* або зміщення залежить від сегмента. Так, у сегменті кодів як *EA* використовується вміст лічильника інструкцій *IP*, у сегментах даних значення *EA* залежить від засобу адресації операнда, у сегменті стека використовуються регістри *SP* або *BP*.

Перетворення логічних адрес на фізичні завжди однозначне, тобто парі *Seg* і *EA* відповідає єдина фізична адреса. Зворотне перетворення не є однозначним: фізичну адресу можна подати за допомогою 4096 пар логічних адрес. У подальшому будемо позначати фізичну адресу у вигляді *Seg*:*EA*, де як *Seg* і *EA* можуть використовуватися і позначення регістрів, і 16-розрядні дані.

Приклад 3.1. Знайти значення фізичної адреси за двома значеннями логічних адрес *CS*:*IP*.

Нехай вмістом сегментного регістра *CS* є число 2002*H*, вмістом вказівника команд *IP* – 3175*H*. Додамо до значення *CS* чотири нулі праворуч:

$$CS(0000) = 0010\ 0000\ 0000\ 0010\ 0000B = 20020H.$$

Виконавши операцію додавання цієї величини до вмісту регістра *IP*, отримаємо фізичну адресу:

$$\begin{array}{r} 0010\quad 0000\quad 0000\quad 0010\quad 0000 \\ +\quad\quad\; 0011\quad 0001\quad 0111\quad 0101 \\ \hline 0010\quad 0011\quad 0001\quad 1001\quad 0101 \end{array} = 23195H$$

Отже, запис *CS*:*IP* при *CS* = 2002*H*, *IP* = 3175*H* відповідає фізичній адресі 23195 *H*.

Приклад 3.2. Знайти значення двох логічних адрес, які б відповідали фізичній адресі 23195*H* і не дорівнювали логічним адресам прикл. 3.1.

Значення фізичної адреси 23195*H* можна одержати додаванням двох інших логічних адрес 2100*H*:2195*H*:

$$\begin{array}{r} 0010\quad 0001\quad 0000\quad 0000\quad 0000 \\ +\quad\quad\; 0010\quad 0001\quad 1001\quad 0101 \\ \hline 0010\quad 0011\quad 0001\quad 1001\quad 0101 \end{array} = 23195H$$

Ємність пам’яті 1 Мбайт, починаючи з нульової адреси, розбивається на параграфи по 16 байт. Сегмент може починатися тільки на межі параграфа, тобто в адресі сегмента молодші чотири біти адреси – нульові. Розміщення сегментів у пам’яті довільне: сегменти можуть частково або повністю перекриватися або не мати загальних частин. Змінюючи значення як першої, так і другої логічних адрес, можна адресувати будь-яку комірку із загальної пам’яті ємністю 1 Мбайт.

На рис. 3.15, *а* показано розміщення у просторі 1 Мбайт чотирьох сегментів по 64 кбайт без перекриття. Початкові адреси сегментів визначаються вмістом 16-розрядних сегментних регістрів, які доповнено праворуч чотирма нульовими бітами. Зміщення в сегменті кодів визначається вмістом регістра *IP*; зміщення в сегменті даних і додатковому сегменті даних – ефективною адресою *EA*, яка наводиться в команді; у сегменті стека – вмістом регістра *SP*.

У сегментах кодів розміщено коди команд, тобто програму у машинних кодах; у решті сегментів – дані. Програма може звертатися тільки до даних у сегментах (рис. 3.15), позначених заштрихованими областями.

Змінюючи вміст сегментних регістрів, можна пересувати сегменти в межах усієї пам’яті 1 Мбайт. На рис. 3.15, *б* показано розташування сегментів кодів, даних, стека та додаткового сегмента із частковим перекриттям. Такий випадок виникає тоді, коли вміст сегментних регістрів відрізняється менш ніж на 64 кбайт/16 = 4096 байт.

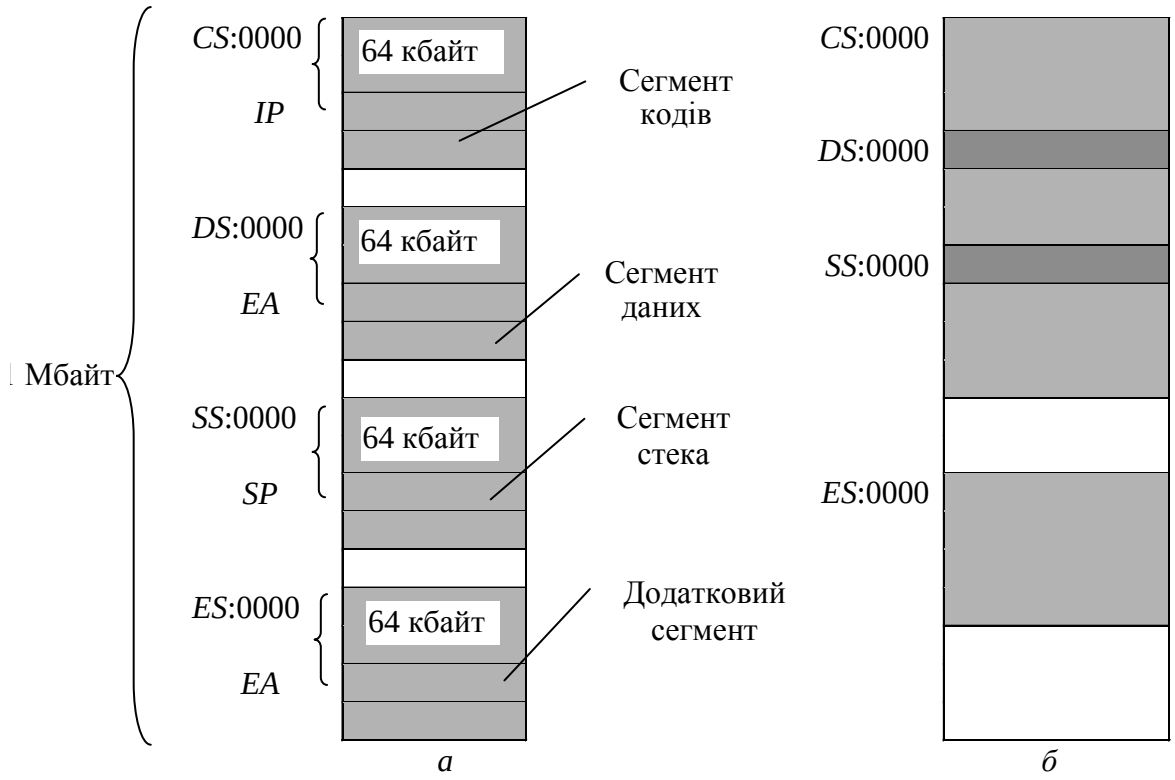


Рис. 3.15. Розміщення сегментів у просторі пам’яті 1 Мбайт:
а – без перекриття; *б* – з частковим перекриттям

Програмна модель. Програмна модель МП *i*8086 (рис. 3.16) складається з РЗП, сегментних регістрів, вказівника команд і регістра прапорців.

Регістри загального призначення поділяються на регістри даних і регістри-вказівники. До регістрів даних відносять чотири 16-розрядні регістри: *AX, BX, CX, DX*. Кожний із цих регістрів складається з двох 8-розрядних регістрів, які можна незалежно адресувати за символічними іменами *AH, BH, CH, DH* (старші байти – *High*) та *AL, BL, CL, DL* (молодші байти – *Low*). Регістри-вказівники *SP* (*Stack Pointer* – вказівник стека), *BP* (*Base Pointer* – базовий регістр), *SI* (*Source Index* – індекс джерела), *DI* (*Destination Index* – індекс призначення) є 16-розрядними. Усі РЗП можна використати для зберігання даних, але в деяких командах допускається використання певного регістра за замовчуванням: *AX* – при множенні, діленні, введенні та виведенні слів; *AL* – при множенні, діленні, введенні та виведенні байтів, десятковій корекції, перетворенні байтів (команда *XLAT*); *AH* – при множенні і діленні байтів; *BX* – при трансляції; *CX* – як лічильник циклів і вказівник довжини рядків у рядкових командах; *CL* – для зберігання зміщення з указанням змінної; *DX* – при множенні та діленні слів, введенні та виведенні з непрямою адресацією; *SP* – при операціях зі стеком;

SI, *DI* – при рядкових операціях. На відміну від 8-розрядних МП регістр *SP* зберігає зміщення останньої зайнятої комірки стека відносно початку сегмента стека, а повна адреса стека визначається як *SS*:*SP*.

Регістри даних

	15 … 8	7 … 0
AX	*AH*	*AL*
BX	*BH*	*BL*
CX	*CH*	*CL*
DX	*DH*	*DL*

Регістри-вказівники

15 … 0
SP
BP
SI
DI

Сегментні регістри

15 … 0
CS
DS
ES
SS

Вказівник команд

15 … 0
IP

Регістр прапорців *F*

X	*X*	*X*	*X*	*OF*	*DF*	*IF*	*TF*	*SF*	*ZF*	*X*	*AF*	*X*	*PF*	*X*	*CF*

Рис. 3.16. Програмна модель МП *i*8086

Сегментні регістри *CS*, *DS*, *ES*, *SS* визначають початкові адреси чотирьох сегментів пам’яті. Використання сегментних регістрів визначається типом звернення до пам’яті (табл. 3.9).

Для деяких типів звернень допускається заміна сегментного регістра за замовчуванням на альтернативний, яка реалізується префіксами команд *CS*:, *DS*:, *SS*:, *ES*:.

Таблиця 3.9. **Використання регістрів при адресації пам'яті**

Тип звернення до пам'яті	Сегментний регістр		Зміщення
	за замовчуванням	альтернативний	
Вибірка команд	*CS*	Немає	*IP*
Стекові операції	*SS*	Немає	*SP*
Адресація змінної	*DS*	*CS, ES, SS*	*EA*
Рядок-джерело	*DS*	*CS, ES, SS*	*SI*
Рядок-приймач	*ES*	Немає	*DI*
Використання *BP* при зверненні до стека у разі читання/запису даних	*SS*	*CS, ES, DS*	*EA*

Примітка. Рядок-джерело і рядок-приймач – це рядки даних (масиви), які беруть участь у рядкових командах.

Приклад 3.3. Переслати вміст комірки пам'яті з адресою *DS*:1000*H* у регістр-акумулятор *AL*.

Для того щоб переслати вміст комірки пам'яті в акумулятор, треба використати команду пересилки *MOV dst, src*, де операндом призначення *dst* (*Destination*) є регістр *AL*, а операндом джерела інформації *src* (*Source*) – комірка пам'яті. Комірка пам'яті позначається квадратними дужками, всередині яких записується зміщення у сегменті, тобто друга логічна адреса. Перша логічна адреса за замовчуванням є вмістом регістра *DS*.

Після запису мнемоніки команди пересилки *MOV* записується операнд-призначення, а потім через кому – операнд-джерело. Після операндів через крапку з комою записується коментар до команди. Отже, за командою

```
MOV AL, [1000H] ; AL←DS:[1000H]
```

у регістр *AL* пересилається байт з комірки пам'яті з адресою *DS*:1000*H*.

Зазначимо, що перед використанням цієї команди вміст регістра *DS* має бути визначеним.

Приклад 3.4. Переслати вміст комірки пам'яті з адресою *ES*:1000*H* у регістр-акумулятор *AL*.

За командою з префіксом *ES*

```
MOV AL, ES: [1000H] ; AL ← ES:[1000H]
```

у *AL* пересилається вміст комірки пам'яті з адресою *ES*:1000*H*.

На відміну від 8-розрядних МП вказівник команд *IP* зберігає зміщення в сегменті кодів поточної команди.

Регістр прапорців зберігає ознаки результатів виконання арифметичних і логічних операцій і керувальні ознаки, які можна встановити або скинути програмно. Типи прапорців подано в табл. 3.10.

***Таблиця 3.10*. Призначення прапорців**

Позначення	Призначення	Розрядність операнда	
		8	16
AF	*Auxiliary Flag* – прапорець допоміжного перенесення/позики з молодшої тетради в старшу (з розряду *D*3 у розряд *D*4). Використовується при десятковій арифметиці	+	–
CF	*Carry Flag* – прапорець перенесення/позики. Установлюється при виході результату додавання (віднімання) беззнакових операндів за межу діапазону. У командах зсуву прапорець *CF* фіксує значення старшого біта	+	+
OF	*Overflow Flag* – прапорець переповнення. Установлюється при виході знакового результату за межі діапазону	+	+
SF	*Sign Flag* – прапорець знака. Дублює значення старшого біта результату; *SF* = 0 для додатних чисел і *SF* = 1 – для від'ємних	+	+
PF	*Parity Flag* – прапорець паритету (парності). Установлюється при парному числі одиниць у результаті	+	–
ZF	Zero *Flag* – прапорець нульового результату. Установлюється при отриманні нульового результату операції	+	+
DF	*Direction Flag* – прапорець керування напрямом у рядкових операціях. При *DF* = 1 індексні регістри *SI, DI*, що беруть участь у рядкових операціях, автоматично декрементуються на кількість байтів операнда, при *DF* = 0 – інкрементуються	–	–
IF	*Interrupt-enable Flag* – прапорець дозволу переривань. При *IF* = 1 дозволяється виконання маскованих апаратних переривань	–	–
TF	*Trap Flag* – прапорець трасування (покрокового режиму). При його встановленні після виконання кожної команди викликається внутрішнє переривання 1 (*INT* 1)	–	–

Приклад 3.5. Визначити значення прапорців після виконання команди *ADD AL, BL* (додавання вмісту 8-розрядних регістрів *AL, BL*; результат передається в *AL*), якщо в регістрі *AL* міститься число 49*H*, а в регістрі *BL* – 68*H*.

Після виконання команди додавання прапорці встановляться таким чином (рис. 3.17).

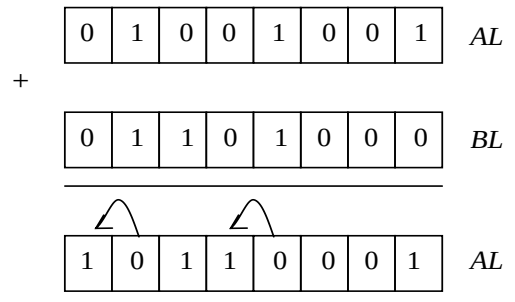


Рис. 3.17. Установлення прапорців після виконання команди *ADD AL, BL*

Пояснимо встановлення прапорців. Результат операції додавання не є нульовим, тому прапорець *ZF* скинуто, тобто *ZF* = 0. Старший розряд результату дорівнює одиниці, тому *SF* = 1. Кількість одиниць у результаті 4, тобто парне число, тому *PF* = 1. У результаті додавання виникло переповнення із молодшої тетради у старшу (*AF* = 1), у знаковий розряд (*OF* = 1). Переповнення розрядної сітки не відбулося, тому *CF* = 0.

Адресація портів введення-виведення. Простір адрес портів введення-виведення несегментований, займає 64 кбайт і адресується 16 молод-

шими розрядами 20-розрядної шини адреси. Порти можуть бути як 8-, так і 16-розрядними. Будь-які два суміжні 8-розрядні порти можна вважати 16-розрядним портом аналогічно слову у пам'яті. При цьому для обміну з 8-розрядними портами використовується регістр *AL*, а з 16-розрядними – регістр *AX*. Перші 256 портів (з номерами 0–0*FFH*) можна адресувати за допомогою прямої адресації.

Приклад 3.6. Ввести інформацію з 8-розрядного порту з адресою 56*H* у регістр-акумулятор *AL*.

Для того щоб ввести інформацію з 8-розрядного порту з адресою 56*H* в акумулятор *AL*, треба виконати команду введення *IN*. Першим операндом команди є позначення акумулятора *AL*, якщо вводиться байт інформації, або *AX*, якщо вводиться слово. У цьому випадку треба використати операнд *AL*. Другим операндом є номер порту 56*H*. Отже, за командою

```
IN AL, 56H      ; AL ← P8(56H)
```

відбудеться введення інформації з 8-розрядного порту з адресою 56*H* до акумулятора.

Зазначимо, що допускається позначення номера порту у квадратних дужках:

```
IN AL, [56H]      ; AL ← P8(56H).
```

Приклад 3.7. Вивести інформацію з регістра-акумулятора *AX* у 16-розрядний порт з адресою 34*H*.

Для того щоб вивести інформацію з акумулятора *AX* до 16-розрядного порту з адресою 34*H*, треба виконати команду виведення *OUT*. Першим операндом команди є номер порту 34*H*, другим – позначення акумулятора *AL*, якщо виводиться байт інформації, або *AX*, якщо виводиться слово. У цьому разі треба використати операнд *AX*. Отже, за командою

```
OUT 34H, AX       ; AX → P16(34H)
```

відбудеться виведення інформації з регістра *AX* на 16-розрядний порт з адресою 34*H*.

Усі 64 кбайт портів адресуються непрямо – за допомогою регістра *DX*.

Приклад 3.8. Ввести інформацію з 8-розрядного порту з адресою, що знаходиться у регістрі *DX*, у регістр-акумулятор *AL*.

Для того щоб ввести інформацію, треба виконати команду введення *IN*, першим операндом якої є позначення акумулятора *AL*, а другим – позначення регістра *DX*.

Отже, за командою

```
IN AL, DX         ; AL ← P8(DX)
```

відбудеться введення інформації в акумулятор *AL* з 8-розрядного порту з адресою, що знаходиться в регістрі *DX*. Вміст регістра *DX* має бути визначений до моменту виконання команди введення.

Допускається запис команди з непрямою адресацією у вигляді

```
IN AL, [DX].
```

Типи адресації операндів. У МП *i*8086 використовуються такі самі основні типи адресації (пряма, регістрова, безпосередня та непряма), що і

для 8-розрядних процесорів (див. підрозд. 3.1), однак непряма адресація має такі різновиди: базова, індексна, базово-індексна.

Базова адресація. Ефективна адреса операнда *EA* обчислюється складанням вмісту базових регістрів *BX* або *BP* і зміщенням (8- або 16-розрядного знакового числа). В окремому випадку зміщення може не бути.

Приклад 3.9. Переслати в регістр-акумулятор *AX* вміст комірки пам'яті, розміщеної у сегменті даних і має ефективну адресу (зміщення у сегменті), яка дорівнює сумі вмісту регістра *BX* і числа 2000*H*.

Для того щоб переслати вміст комірки пам'яті в акумулятор, треба використати команду пересилання *MOV dst, src*, де операндом призначення *dst* (*destination*) є регістр *AX*, а операндом джерела інформації *src* (*source*) – комірка пам'яті. Комірка пам'яті позначається квадратними дужками, всередині записується значення ефективної адреси, тобто *BX* + 2000*H*.

Отже, за командою

```
MOV AX, [BX + 2000H]      ; AX ← DS:[BX + 2000H]
```

у регістр *AX* пересилається байт з комірки пам'яті з адресою *DS*: *BX* + 2000*H*.

Зазначимо, що перед використанням цієї команди вмісти регістрів *DS* і *BX* визначаються заздалегідь.

Індексна адресація. При індексній адресації як адреси зміщення використовуються вмісти індексних регістрів *SI* або *DI* та зміщення у вигляді числа.

Приклад 3.10. Переслати у регістр-акумулятор *AX* вміст комірки пам'яті, розміщеної у сегменті даних з ефективною адресою (зміщення у сегменті), яка дорівнює сумі вмісту регістра *SI* і числа 5000*H*.

За командою

```
MOV AX, [SI + 5000H]      ; AX ← DS:[SI + 5000H]
```

у регістр *AX* пересилається байт з комірки пам'яті з адресою *DS*:*SI* + 5000*H*.

Перед використанням цієї команди вмісти регістрів *DS* і *SI* визначаються заздалегідь.

Базово-індексна адресація. Ефективна адреса операнда *EA* дорівнює сумі вмісту базових регістрів *BX* або *BP*, індексних регістрів *SI* або *DI* та зміщення. Зазначимо, що числового зміщення може не бути.

Приклад 3.11. Переслати у регістр-акумулятор *AX* вміст комірки пам'яті, що розміщена у сегменті даних і має ефективну адресу, яка дорівнює сумі вмістів двох регістрів *SI* і *BX*.

За командою

```
MOV AX, [SI + BX]    ; AX ← DS:[SI + BX]
```

у регістр *AX* пересилається байт з комірки пам'яті з адресою *DS*:*SI* + *BX*.

Перед використанням цієї команди вмісти регістрів *DS*, *SI* і *BX* визначаються заздалегідь.

Базова та індексна адресації застосовуються для звернення до елементів одновимірного масиву, базово-індексна – до двовимірного масиву.

Цикли шини процесора. Протягом циклу шини МП виставляє адресу комірки пам'яті або ПВВ на шину адреси, формує керувальні сигнали читання/запису, а після цього зчитує або записує дані. Крім циклів ЧИТАННЯ і ЗАПИС ПАМ'ЯТІ або ПВВ, існують цикли ПІДТВЕРДЖЕННЯ ПЕРЕРИВАННЯ і ЗАХОПЛЕННЯ шин. Цикл шини може ініціювати не тільки незалежний процесор *i*8086, але й арифметичний співпроцесор або співпроцесор введення-виведення. Розрізняють цикли шини в мінімальному і максимальному режимах. Часові діаграми циклів ЧИТАННЯ та ЗАПИС у мінімальному режимі показано на рис. 3.18.

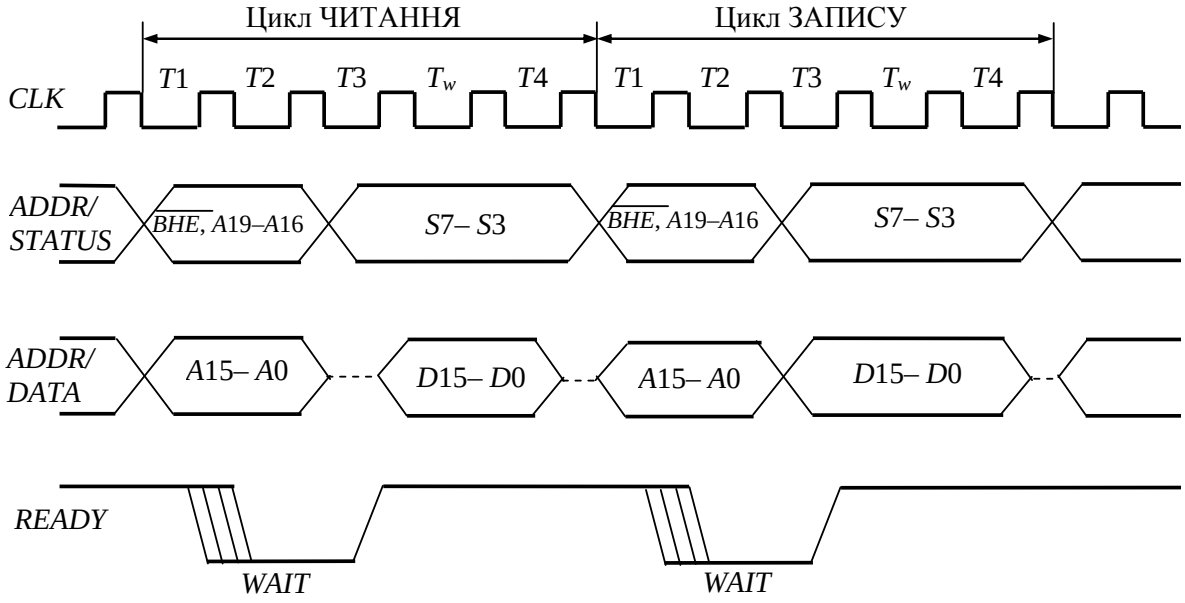


Рис. 3.18. Цикли ЧИТАННЯ і ЗАПИС МП *i*8086 у мінімальному режимі

Цикл шини складається, як мінімум, із чотирьох тактів. Такт визначається як проміжок часу між задніми фронтами двох сусідніх імпульсів *CLK*. Будь-який цикл шини може бути необмежено розтягнуто за допомогою сигналу готовності *READY*; при цьому процесор вводить необов'язкові такти очікування T_w.

Цикли звернення до порту відрізняються від циклів пам'яті тим, що старші розряди шини адреси мають нульове значення (при непрямій адресації за допомогою *DX* сигнали на лініях *A*19–*A*16 набувають значень *L*-рівня, при прямій адресації сигнали на лініях *A*19–*A*8 є нульовими).

Цикл ПІДТВЕРДЖЕННЯ ПЕРЕРИВАННЯ формується аналогічно циклу ЧИТАННЯ порту, але замість активного сигналу $I\overline{OR}$ активним є сигнал $\overline{INTA}$, а шина адреси процесором не керується.

Типи переривань. Процесор *i*8086 може обробляти до 256 типів переривань. Кожному перериванню відповідає свій вектор – подвійне слово, що містить адресу *CS*:*IP* підпрограми, що викликається. Під вектори переривань у загальному просторі адрес пам'яті відводиться 1 кбайт починаючи з нульової адреси (рис. 3.19).

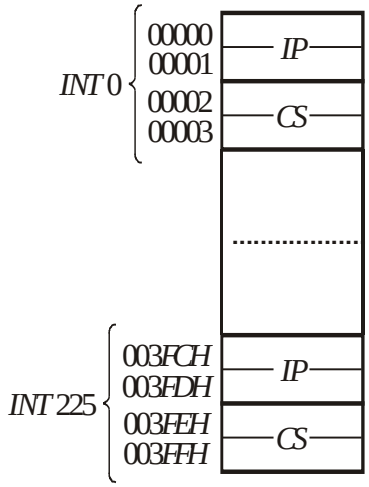


Рис. 3.19. Карта векторів переривань

У разі переходу на підпрограму обробки переривань *INT n* (*n* – тип переривання) процесор переміщує у стек вміст регістрів *IP*, *CS*, регістр прапорців *F* і скидає прапорець дозволу переривання *IF*; обчислює адресу 4×*n* і перше слово за цією адресою переміщує у *IP*, друге – у *CS*. Послідовність цих дій еквівалентна командам:

```
PUSHF                ; Запам'ятовування у стеку прапорців
CALL FAR i_proc_4n   ; Далекий виклик підпрограми обробки
                     ; переривання
```

Скидання прапорця переривання *IF* не дозволяє перервати виконання підпрограми обробки переривання до її завершення або виконання команди дозволу *STI*. Останньою командою підпрограми обробки переривання є команда *IRET*. За цією командою процесор вибирає зі стека адресу повернення (адресу команди, наступної за командою *INT*) і вміст регістра прапорців.

Типи переривань показано на рис. 3.20. Переривання поділяються на зовнішні апаратні та внутрішні. Запити *IRQ* і зовнішніх апаратних переривань надходять до системи переривань або на вивід немаскованого переривання *NMI* МП. Система переривання формує сигнал *INTR* маскованого переривання МП. Зазначимо, що масковане переривання відрізняється від немаскованого тим, що перше може бути заборонено програмно – командою скидання прапорця дозволу переривань *IF*. У цьому разі при надходженні запитів переривання вони будуть ігноруватися. Внутрішні переривання процесора поділяють на програмні й апаратні.

Джерелами внутрішніх апаратних переривань (рис. 3.20) є: помилка ділення (тип 0); покроковий режим (тип 1); команда *INTO* (тип 4).

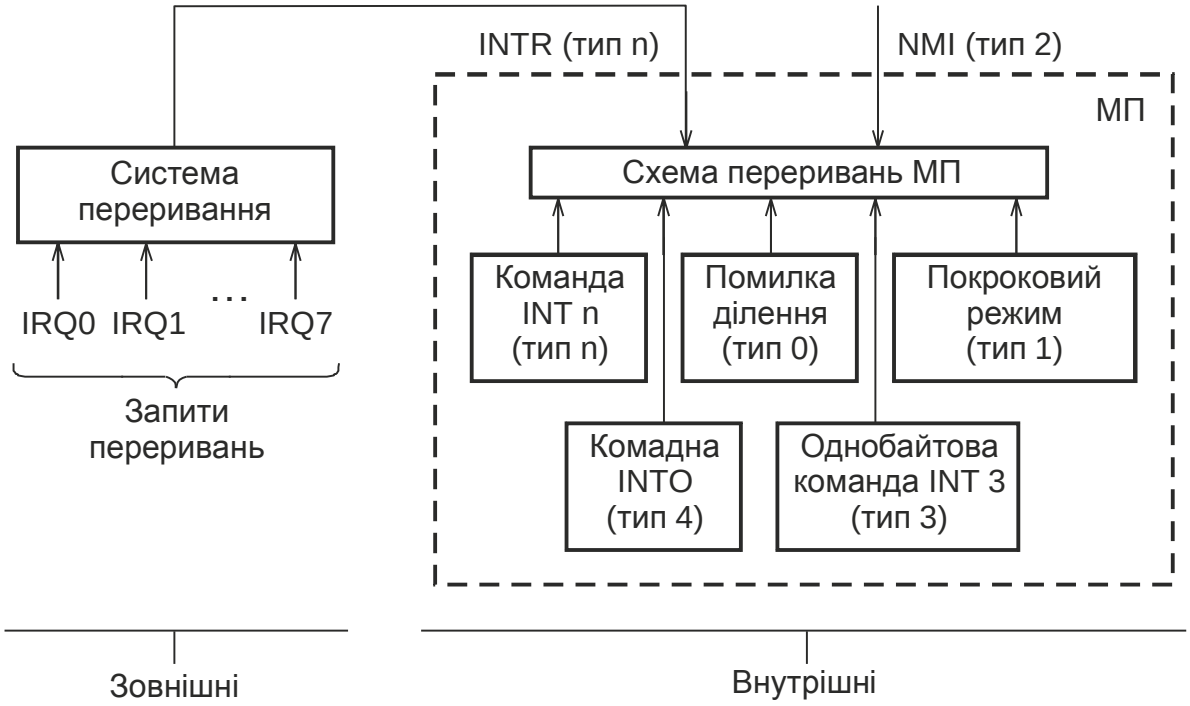


Рис. 3.20. Типи переривань

Внутрішні програмні переривання *INT n* та *INT* 3 виконуються за командами переривання і дозволяють викликати підпрограми обробки переривань (наприклад, сервісні підпрограми *BIOS* і *DOS*) без застосування дальніх викликів. На відміну від *INT n* переривання *INT* 3 є однобайтовою командою і зазвичай використовується для передачі керування підпрограмі-налагоджувачу. Виконання програмних переривань не залежить від прапорця дозволу переривань *IF*.

Внутрішні апаратні переривання процесора виникають у таких особливих випадках:

– при діленні на нуль (тип 0);
– при встановленому прапорці трасування (тип 1). У цьому разі переривання відбувається після виконання кожної команди;
– після команди *INTO* (тип 4), якщо встановлено прапорець переповнення *OF*.

Апаратні переривання виникають при активному рівні сигналів на контактах МП *NMI* (немасковане переривання – тип 2) і *INTR* (масковані, типи 5–255). Масковані переривання виконуються при встановленому прапорці *IF*. При переході до підпрограми обробки апаратного переривання процесор формує два цикли підтвердження переривання один за одним, у яких генерується сигнал $\overline{INTA}$. За другим імпульсом $\overline{INTA}$

контролер переривань передає по шині даних номер вектора переривання *n*. Далі дії процесора аналогічні виконанню програмного переривання. Обробка поточного переривання може бути перервана немаскованим перериванням або іншим маскованим перериванням вищого пріоритету у тому разі, якщо підпрограма-обробник встановить прапорець дозволу переривання *IF*. Немасковане переривання виконується незалежно від стану прапорця *IF*.

Мікропроцесор i8088 відрізняється від МП *i*8086 тим, що має зовнішню 8-розрядну шину даних при внутрішній 16-розрядній шині. Зменшення розрядності шини даних спрощує побудову блоків пам'яті інтерфейсу із зовнішніми пристроями, але продуктивність процесора знижується на 20–30 %. Структурна схема *i*8088 аналогічна схемі МП *i*8086, однак довжина черги команд скорочена до 4 байт, а попередня вибірка виконується за наявності одного вільного байта. Ці властивості оптимізують конвеєр з урахуванням розрядності шини. Програмно процесори ідентичні, їхня система команд і набір регістрів однакові. Так само, як і МП *i*8086, МП *i*8088 виконує 8- і 16-розрядні логічні й арифметичні операції, включаючи множення та ділення в двійковому та двійково-десятковому кодах, операції з рядками, підтримує режими переривання, ПДП, операції з портами. Розташування контактів МП *i*8088 й *i*8086 збігається, за винятком того, що лінії *AD*15–*AD*8 використовуються тільки для адреси, і лінія $\overline{BHE}$ замінена лінією стану *ST*0. Сигнали *ST*0, *DT*/$\overline{R}$ і *IO*/$\overline{M}$ можуть бути використані для ідентифікації циклу шини згідно з табл. 3.6.

Мікропроцесори i80186(i80188) є програмно сумісними з МП *i*8086. Розрядність шини адреси 20, шини даних – 16 і 8 відповідно. Процесори мають вбудовані периферійні контролери переривань, ПДП, триканальний таймер і тактовий генератор. Мікропроцесори *i*80C186/*i*80C188 мають засоби керування енергоспоживанням. Є модифікації з вбудованим послідовним портом та контролерами динамічної пам'яті.

Контрольні запитання

1. Укажіть характерні особливості мінімального та максимального режимів роботи МП *i*8086.
2. Укажіть існуючі формати даних МП *i*8086.
3. Наведіть приклади упакованого і розпакованого двійково-десяткових чисел.
4. Яким чином у МП подаються від'ємні числа?
5. Поясніть принцип конвеєрної архітектури.
6. Укажіть функції операційного пристрою та шинного інтерфейсу.

7. Яким чином визначаються тип, початок і кінець циклу шини за допомогою ліній стану?
8. Які блоки МП беруть участь у формуванні 20-розрядної фізичної адреси?
9. Обчисліть 20-розрядну фізичну адресу *DS*:*SI*, якщо *DS* = 1234*H*, *SI* = 5678*H*.
10. Підберіть дві пари 16-розрядних логічних адрес, які є еквівалентними фізичній адресі 12008*H*.
11. Які групи регістрів входять до програмної моделі МП?
12. Які сегментні регістри за замовчуванням адресують початок сегментів кодів, стека, даних?
13. Укажіть призначення регістра прапорців.
14. Наведіть приклад виконання команди, після якої встановлюється прапорець знака.
15. Наведіть приклад виконання команди, після якої встановлюється прапорець паритету.
16. Наведіть приклад виконання команди, після якої встановлюється прапорець нульового результату.
17. Наведіть приклади команд введення-виведення з прямою адресацією 8-розрядного, 16-розрядного портів.
18. Які існують типи адресації операндів у пам’яті?
19. Яким чином обчислюється ефективна адреса операнда при різних типах адресації?
20. Назвіть та охарактеризуйте існуючі типи циклів шини.
21. Дайте визначення вектора переривань і карти векторів переривань.
22. Які дії виконує МП при переході на підпрограму обробки переривань?
23. Назвіть та охарактеризуйте існуючі типи переривань МП *i*8086.

3.3. Система команд МП *i*8086

Система команд МП *i*8086 (табл. 3.11) містить 91 мнемокод. Усі команди МП можна розподілити на п’ять груп:

1) команди передачі інформації (команди пересилання, роботи зі стеком, введення-виведення);
2) команди обробки інформації (арифметичні, логічні, команди зсуву);
3) рядкові команди;
4) команди передачі керування, включаючи команди переривань;
5) команди керування станом МП.

У табл. 3.11 вжито такі позначення:

src – операнд-джерело;

dest – операнд-призначення;
reg – 8- /16-розрядний РЗП;
*reg*8 – 8-розрядний РЗП;
*reg*16 – 16-розрядний РЗП;
sr – сегментний регістр;
mem – 8-/16-розрядна комірка пам'яті;
*mem*8 – 8-розрядна комірка пам'яті;
*mem*16 – 16-розрядна комірка пам'яті;
r/m – 8-/16-розрядний регістр або комірка пам'яті;
r/m/i – 8-/16-розрядний регістр, комірка пам'яті або безпосередній операнд;
immed – безпосередній операнд;
disp – 8-/16-розрядне зміщення при заданні адреси;
*disp*8 – 8-розрядне зміщення;
*disp*16 – 16-розрядне зміщення;
target – мітка, до якої здійснюється перехід;
seg target – перша логічна адреса (сегментний адрес) мітки *target*;
offset target – друга логічна адреса (зміщення у сегменті) мітки *target*;
a – акумулятор *AL* або *AX*;
m[*disp*] – комірка пам'яті з ефективною адресою *EA* = *disp*.

***Таблиця 3.11*. Система команд мікропроцесора *i*8086**

Мнемокод команди	Опис команди	Алгоритм команди	Кількість байт	Кількість тактів
1	2	3	4	5
КОМАНДИ ПЕРЕДАЧІ ІНФОРМАЦІЇ				
Команди пересилання				
MOV dest, src	Пересилання даних з регістра, комірки пам'яті або безпосереднього операнда у регістр або пам'ять	*reg* ← *reg*	2	2
		sr ← *reg*	2	2
		reg ← *sr*	2	2
		mem ← *reg*	2–4[1)]	9 + *EA*
		reg ← *mem*	2–4[1)]	8 + *EA*
		mem ← *sr*	2–4[1)]	9 + *EA*
		sr ← *mem*	2–4[1)]	8 + *EA*
		a ← *mem*	3	11
		mem ← *a*	3	11
		*mem*8 ← *immed*	3–5[2)]	10 + *EA*
		*mem*16 ← *immed*	4–6[3)]	10 + *EA*
		*reg*8 ← *immed*	2	4
		*reg*16 ← *immed*	3	4

Продовження табл. 3.11

1	2	3	4	5
XCHG r/m, reg	Обмін даними між регістрами або регістром і пам'яттю	*reg* ←→ *reg* *mem* ←→ *reg* *A* ←→ *reg*	2 2–4[1)] 1	4 17 + *EA* 3
XLAT	Перекодування вмісту *AL* в значення байта пам'яті з адресою *ES*:[*BX* + (*AL*)]	*AL* ← *ES*:[*BX* + + (*AL*)]	1	11
*LEA reg*16, *mem*	Завантаження ефективної адреси комірки пам'яті *mem* у регістр	*reg* ← *EA*	2–4[1)]	2 + *EA*
*LDS reg*16, *mem*	Завантаження у регістр reg16 слова із комірки пам'яті за адресою [*mem*], у *DS* – наступного слова з комірки за адресою [*mem* + 2]	*reg* ← [*mem*] *DS* ← [*mem*+2]	2–4[1)]	16 + *EA*
*LES reg*16, *mem*	Завантаження в регістр *reg*16 слова із комірки пам'яті за адресою [*mem*], в *ES* – наступного слова з комірки за адресою [*mem* + 2]	*reg* ← [*mem*] *ES* ← [*mem* + 2]	2–4[1)]	16 – *EA*
LAHF	Завантаження молодшого байта регістра прапорців *FL* в *AH*	*AH* ← *FL*	1	4
SAHF	Збереження *AH* у молодшому байті регістра прапорців *FL*	*FL* ← *AH*	1	4
Команди роботи зі стеком				
PUSH r/m/sr	Пересилання слова з регістра або з пам'яті у стек	*SP* ← *SP* – 2 [*SS*:*SP*] ← *r/m*	2–4[1)]	11/(16 + *EA*)
		SP ← *SP* – 2 [*SS*:*SP*] ← *sr*	1	10
PUSHF	Пересилання у стек вмісту регістра прапорців	*SP* ← *SP* – 2 [*SS*:*SP*] ← *F*	1	10
POP r/m/sr	Пересилання слова даних зі стека у регістр або пам'ять	*r/m* ← [*SS*:*SP*] *SP* ← *SP* + 2	2–4[1)]	8/(16 + *EA*)
		sr ← [*SS*:*SP*] *SP* ← *SP* + 2	1	8
POPF	Пересилання даних зі стека у регістр прапорців	*F* ← [*SS*:*SP*] *SP* ← *SP* + 2	1	8

Продовження табл. 3.11

1	2	3	4	5
Команди введення-виведення				
*IN AL, P*8 *IN AL, DX*	Уведення в акумулятор *AL* байта з 8-розрядного порту з адресою *P*8 або з адресою, що зберігається в *DX*	*AL* ← Порт (*P*8) *AL* ← Порт (*DX*)	2 1	10 8
*IN AX, P*8 *IN AX, DX*	Уведення в акумулятор *AX* слова з 16-розрядного порту з адресою *P*8 або з адресою, що зберігається в *DX*	*AX* ← Порт (*P*8) *AX* ← Порт (*DX*)	2 1	10 8
*OUT P*8, *AL* *OUT DX, AL*	Виведення байта з акумулятора *AL* у 8-розрядний порт з адресою *P*8 або з адресою, що зберігається в *DX*	Порт (*P*8) ← *AL* Порт (*DX*) ← *AL*	2 1	10 8
*OUT P*8, *AX* *OUT DX, AX*	Виведення слова з акумулятора *AX* у 16-розрядний порт з адресою *P*8 або з адресою, що зберігається в *DX*	Порт (*P*8) ← *AX* Порт (*DX*) ← *AX*	2 1	10 8
КОМАНДИ ОБРОБКИ ІНФОРМАЦІЇ				
Арифметичні команди				
ADD r/m, *r/m/i*	Додавання двох операндів	*r/reg* ← *r/reg* + *reg* *reg* ← *reg* + *r/m* *reg*8 ← *reg*8 + *immed* *reg*16 ← *reg*16 + *immed* *mem*8 ← *mem*8 + *immed* *mem*16 ← *mem*16 + *immed* *AL* ← *AL* + *immed* *AX* ← *AX* + *immed*	2 2–4[1)] 2–4[1)] 3 4 3–5[2)] 2 3	3 16 + *EA* 9 + *EA* 4 4 17 + *EA* 4 4

Продовження табл. 3.11

1	2	3	4	5
ADC r/m, r/m/i	Додавання двох операндів і прапорця перенесення *CF* від попередньої операції. Прапорець перенесення додається до молодшого біта результату	*reg* ← *reg* + *reg* + *CF*	2	3
		mem ← *mem* + *reg* + *CF*	2–4[1)]	16 + *EA*
		reg ← *reg* + *mem* + *CF*	2–4[1)]	9 + *EA*
		*reg*8 ← *reg*8 + *immed* + *CF*	3	4
		*reg*16 ← *reg*16 + *immed* + *CF*	4	4
		*mem*8 ← *mem*8 + *immed* + *CF*	3–5[2)]	17 + *EA*
		*mem*16 ← *mem*16 + *immed* + *CF*	4–6[3)]	17 + *EA*
		AL ← *AL* + *immed* + *CF*	2	4
		AX ← *AX* + *immed* + *CF*	3	4
INC r/m	Інкремент (додавання з одиницею). Команда не діє на прапорець *CF*	*reg*8 ← *reg*8 + 1	2	3
		*reg*16 ← *reg*16 + 1	1	2
		mem ← *mem* +1	2–4[1)]	15 + *EA*
AAA	Корекція після додавання розпакованих двійково-десяткових чисел		1	4
DAA	Корекція після додавання упакованих двійково-десяткових чисел		1	4
SUB r/m, r/m/i	Віднімання двох операндів	*reg* ← *reg* – *reg*	2	3
		mem ← *me* – *reg*	2–4[1)]	16 + *EA*
		reg ← *reg* – *mem*	2–4[1)]	9 + *EA*
		*reg*8 ← *reg* – *immed*	3	4
		*reg*16 ← *reg*1 – *immed*	4	4
		*mem*8 ← *mem* – *immed*	3–5[2)]	17 + *EA*
		*mem*16 ← *mem*1 – *immed*	4–6[3)]	17 + *EA*
		AL ← *A* – *immed*	2	4
		AX ← *A* – *immed*	3	4
SBB r/m, r/m/i	Віднімання байта та прапорця позики *CF* від попередньої операції. Прапорець позики віднімається від молодшого біта результату	*reg* ← *reg* – *reg* – *CF*	2	3
		mem ← *mem* – *reg* – *CF*	2–4[1)]	16 + *EA*
		reg ← *reg* – *mem* – *CF*	2–4[1)]	9 + *EA*
		*reg*8 ← *reg*8 – *immed* – *CF*	3	4
		*reg*16 ← *reg*16 – *immed* – *CF*	4	4
		*mem*8 ← *mem*8 – *immed* – *CF*	3–5[2)]	17 + *EA*
		*mem*16 ← *mem*16 – *immed* – *CF*	4–6[3)]	17 + *EA*
		AL ← *AL* – *immed* – *CF*	2	4
		AX ← *AX* – *immed* – *CF*	3	4

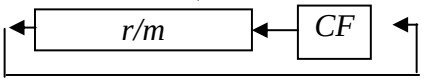
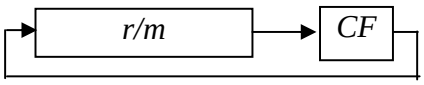
Продовження табл. 3.11

1	2	3	4	5
DEC r/m	Декремент (віднімання одиниці). Команда не діє на прапорець *CF*	*reg*8 ← *reg*8 – 1 *reg*16 ← *reg*16 – 1 *mem* ← *mem* – 1	2 1 2–4[1)]	3 2 15 + *EA*
NEG r/m	Зміна знака операнда	*reg* ← – *reg* *mem* ← – *mem*	3 2–4[1)]	3 16 + *EA*
CMP r/m, r/m/i	Порівняння двох операндів – встановлення вмісту регістра прапорців *F* за результатом віднімання (без збереження результату віднімання)	*F* ⇐ *reg* – *reg* *F* ⇐ *mem* – *reg* *F* ⇐ *reg* – *mem* *F* ⇐ *reg*8 – *immed* *F* ⇐ *reg*16 – *immed* *F* ⇐ *mem*8 – *immed* *F* ⇐ *mem*16 – *immed* *F* ⇐ *AL* – *immed* *F* ⇐ *AX* – *immed*	2 2–4[1)] 2–4[1)] 3 4 3–5[2)] 4–6[3)] 2 3	3 9 + *EA* 9 + *EA* 4 4 1 + *EA* 1 + *EA* 4 4
AAS	Корекція після віднімання розпакованих двійково-десяткових чисел		1	4
DAS	Корекція після віднімання упакованих двійково-десяткових чисел		1	4
MUL r/m	Множення *AL* (*AX*) на беззнакове значення *r/m*	*AX* ←*AL* × *reg*8 (*DX*, *AX*) ←*AX* × *reg*16 *AX* ← *AL* × *mem*8 (*DX*, *AX*) ← *AX* × *mem*16	2 2 2–4[1)] 2–4[1)]	70 – 77 118 – 133 76 – 83 + *EA* 124 – 139 + + *EA*
IMUL r/m	Множення *AL* (*AX*) на знакове значення *r/m*	*AX* ← *AL* × *reg*8 (*DX*, *AX*) ←*AX* × *reg*16 *AX* ← *AL* × *mem*8 (*DX*, *AX*) ← *AX* × *mem*16	2 2 2–4[1)] 2–4[1)]	80 – 98 128 – 154 96 – 104 + *EA* 134 – 160 + *EA*
AAM	Корекція після множення розпакованих двійково-десяткових чисел		2	83
DIV r/m	Ділення акумулятора на беззнакове число (ділення на нуль викликає переривання *INT* 0)	*AX* : *reg*8 → *AL* (остача в *AH*) (*DX*, *AX*) : *reg*16 → *AX* (остача в *DX*) *AX* : *mem*8 → *AL* (остача в *AH*) (*DX*, *AX*) : *mem*16 → *AX* (остача в *DX*)	2 2 2–4[1)] 2–4[1)]	80 – 90 144 – 162 86 – 96 + *EA* 150 – 168 + *EA*

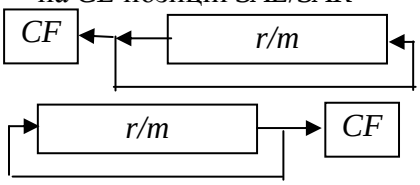
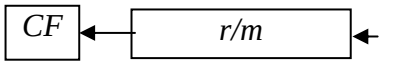
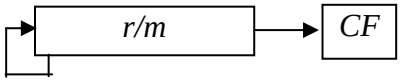
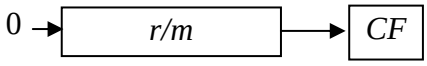
Продовження табл. 3.11

1	2	3	4	5
IDIV r/m	Ділення акумулятора на ціле 8- або 16-розрядне число. (Ділення на нуль викликає переривання *INT* 0.)	$AX : reg8 \to AL$ (остача в *AH*)	2	101 – 112
		$(DX, AX) : reg16 \to AX$ (остача в *DX*)	2	165 – 184
		$AX : mem8 \to AL$ (остача в *AH*)	2–4[1)]	144 – 168 + + *EA*
		$DX, AX : mem16 \to AX$ (остача в *DX*)	2–4[1)]	166 – 190 + + *EA*
AAD	Корекція перед діленням розпакованих двійково-десяткових чисел		2	60
CBW	Перетворення байта *AL* на слово *AX* (повторення вмісту знакового розряду (*AL*.7) регістра *AL* в усіх розрядах регістра *AH*)	$AH \leftarrow (AL7)$	1	2
CWD	Перетворення слова *AX* на подвійне слово *DX*, *AX* (повторення вмісту знакового розряду (*AX*.15) регістра *AX* в усіх розрядах регістра *DX*)	$DX \leftarrow (AX15)$	1	5
Логічні команди				
NOT r/m	Інверсія (інверсія всіх бітів операнда)	$reg \leftarrow \overline{reg}$	2	3
		$mem \leftarrow \overline{mem}$	2–4[1)]	16 + *EA*
AND r/m, r/m/i	ЛОГІЧНЕ І двох операндів	$reg \leftarrow reg \wedge reg$	2	3
		$mem \leftarrow mem \wedge reg$	2–4[1)]	16 + *EA*
		$reg \leftarrow reg \wedge mem$	2–4[1)]	9 + *EA*
		$reg8 \leftarrow reg8 \wedge immed$	3	4
		$reg16 \leftarrow reg16 \wedge immed$	4	4
		$mem8 \leftarrow mem8 \wedge immed$	3–5[2)]	17 + *EA*
		$mem16 \leftarrow mem16 \wedge immed$	4–6[3)]	17 + *EA*
		$AL \leftarrow AL \wedge immed$	2	4
		$AX \leftarrow AX \wedge immed$	3	4
OR r/m, r/m/i	ЛОГІЧНЕ АБО двох операндів	$reg \leftarrow reg \vee reg$	2	3
		$mem \leftarrow mem \vee reg$	2–4[1)]	16 + *EA*
		$reg \leftarrow reg \vee mem$	2–4[1)]	9 + *EA*
		$reg8 \leftarrow reg8 \vee immed$	3	4
		$reg16 \leftarrow reg16 \vee immed$	4	4

Продовження табл. 3.11

1	2	3	4	5
		*mem*8 ← *mem*8 ∨ *immed*	3–5[2)]	17 + EA
		*mem*16 ← *mem*16 ∨ *immed*	4–6[3)]	17 + EA
		AL ← *AL* ∨ *immed*	2	4
		AX ← *AX* ∨ *immed*	3	4
XOR r/m, r/m/i	ВИКЛЮЧНЕ АБО двох операндів	*reg* ← *reg* ⊕ *reg*	2	3
		mem ← *mem* ⊕ *reg*	2–4[1)]	16 + *EA*
		reg ← *reg* ⊕ *mem*	2–4[1)]	9 + *EA*
		*reg*8 ← *reg*8 ⊕ *immed*	3	4
		*reg*16 ← *reg*16 ⊕ *immed*	4	4
		*mem*8 ← *mem*8 ⊕ *immed*	3–5[2)]	17 + *EA*
		*mem*16 ← *mem*16 ⊕ *immed*	4–6[3)]	17 + *EA*
		AL ← *AL* ⊕ *immed*	2	4
		AX ← *AX* ⊕ *immed*	3	4
TEST r/m, r/m/i	Перевірка (ЛОГІЧНЕ І без запису результату і встановлення прапорців відповідно до результату)	*F* ⇐ *reg* ∧ *reg*	2	3
		F ⇐ *mem* ∧ *reg*	2–4[1)]	9 + *EA*
		F ⇐ *reg* ∧ *mem*	2–4[1)]	9 + *EA*
		F ⇐ *reg*8 ∧ *immed*	3	4
		F ⇐ reg16 ∧ *immed*	4	4
		F ⇐ *mem*8 ∧ *immed*	3–5[2)]	10 + *EA*
		F ⇐ *mem*16 ∧ *immed*	4–6[3)]	10 + *EA*
		F ⇐ *AL* ∧ *immed*	2	4
		F ⇐ *AX* ∧ *immed*	3	4
Команди зсуву				
RCL/*RCR r*, 1 *RCL*/*RCR m*, 1 *RCL*/*RCR r, CL* *RCL*/*RCR m, CL*	Циклічний зсув ліворуч/праворуч через біт *CF*: – на одну позицію – на *CL* позицій  ліворуч  праворуч		2 2–4[1)] 2 2–4[1)]	2 15 + *EA* 8 + 4*CL* 20 + *EA* + + 4*CL*

Продовження табл. 3.11

1	2	3	4	5
ROL/ROR r, 1 *ROL/ROR m*, 1 *ROL/ROR r, CL* *ROL/ROR m, CL*	Циклічний зсув ліворуч/ праворуч: – на одну позицію – на *CL* позицій *SAL*/*SAR*  ліворуч праворуч		2 2–4[1)] 2 2–4[1)]	2 15 + *EA* 8 + 4*CL* 20 + *EA* + + 4*CL*
SAL/SAR r, 1 *SAL/SAR m*, 1 *SAL/SAR r, CL* *SAL/SAR m, CL*	Зсув арифметичний ліворуч/ праворуч: – на одну позицію – на *CL* позицій  ліворуч  праворуч При зсуві праворуч значення біта у старшому розряді залишається незмінним		2 2–4[1)] 2 2–4[1)]	2 15 + *EA* 8 + 4*CL* 20 + *EA* + + 4*CL*
SHR r, 1 *SHR m*, 1 *SHR r, CL* *SHR m, CL*	Зсув логічний праворуч [4)] – на одну позицію – на *CL* позицій  		2 2–4[1)] 2 2–4[1)]	2 15 + *EA* 8 + 4*CL* 20 + *EA* + + 4*CL*
Рядкові команди				
REP	Префікс повторення рядкових операцій до онулення *CX* (*CX* декрементується після виконання кожної рядкової операції)		1	
REPE (REPZ)	Префікс умовного повторення – повторення при *ZF* = 1 або до обнуління *CF*		1	
REPNE (*REPNZ*)	Префікс умовного повторення – повторення при *ZF* = 0 або до обнуління *CF*		1	

Продовження табл. 3.11

1	2	3	4	5
MOVSB	Копіювання байта з комірки пам'яті з адресою *DS*:[*SI*] у комірку *ES*:[*DI*]	*ES*:[*DI*]← *DS*:[*SI*] *SI*← *SI* ± 1[7)] *DI* ← *DI* ± 1[7)]	1	18[5)] 9 + 17 *CX*[6)]
MOVSW	Копіювання слова з *DS*:[*SI*] у *ES*:[*DI*]	*ES*:[*DI*] ← *DS*:[*SI*] *SI* ← *SI* ± 2[7)] *DI* ← *DI* ± 2[7)]	1	18[5)] 9 + 17 *CX*[6)]
LODSB	Копіювання байта з *DS*:[*SI*] в *AL*	*AL* ← *DS*:[*SI*] *SI*← *SI* ± 1[7)]	1	11[5)] 9 + 10 *CX*[6)]
LODSW	Копіювання слова з *DS*:[*SI*] в *AX*	*AX* ← *DS*:[*SI*] *SI* ← *SI* ± 2[7)]	1	11[5)] 9 +10 *CX*[6)]
STOSB	Запис байта з *AL* у *ES*:[*DI*]	*ES*:[*DI*] ← *AL* *DI* ← *DI* ± 1[7)]	1	11[5)] 9 + 10 *CX*[6)]
STOSW	Запис слова з *AX* у *ES*: [*DI*]	*ES*:[*DI*] ← *AX* *DI* ← *DI* ± 2[7)]	1	11[5)] 9 + 10 *CX*[6)]
CMPSB	Порівняння байтів *DS*:[*SI*] і *ES*:[*DI*] із записом результату порівняння у регістр прапорців	*F* ⇐ *ES*:[*DI*] – *DS*:[*SI*] *SI* ← *SI* ± 1[7)] *DI* ← *DI* ± 1[7)]	1	22[5)] 9 + 22 *CX*[6)]
CMPSW	Порівняння слів *DS*:[*SI*] і *ES*:[*DI*] із записом результату порівняння у регістр прапорців	*F* ⇐ *ES*:[*DI*] – *DS*:[*SI*] *SI* ← *SI* ± 2[7)] *DI* ← *DI* ± 2[7)]	1	22[5)] 9 + 22 *CX*[6)]
SCASB	Порівняння байта *DS*:[*SI*] і *AL* із записом результату порівняння у регістр прапорців	*F* ⇐ [*DS*:*SI*] –*AL* *SI* ← *SI* ± 1[7)]	1	15[5)] 9 + 15 *CX*[6)]
SCASW	Порівняння слова [*DS*:*SI*] і *AX* із записом результату порівняння у регістр прапорців	*F* ⇐ [*DS*:*SI*] – *AX* *SI* ← *SI* ± 2[7)]	1	15[5)] 9 + 15 *CX*[6)]
КОМАНДИ ПЕРЕДАЧІ КЕРУВАННЯ				
*JMP target*16 *JMP NEAR target*8 *JMP reg* *JMP mem*	Внутрішньосегментний безумовний перехід до цільової адреси *target* (перехід у межах сегмента довжиною 64 кбайт)	*IP* ← *IP* + *disp*16 *IP* ← *IP* + *disp*8 *IP* ← *IP* + *reg* *IP* ← *IP* + *mem*	3 2 2 2–4[1)]	15 15 2 18 + *EA*
JMP FAR target	Міжсегментний безумовний перехід до цільової адреси *target* (перехід у межах усієї ємності пам'яті 1 Мбайт)	*IP* ← *offset target* *CS* ← *seg target*	5	15
JMP FAR mem		*IP* ← [*mem*] *CS* ← [*mem* + 2]	2–4[1)]	24 + EA

Продовження табл. 3.11

1	2	3	4	5
JCX target	Перехід, якщо *CX* = 0		2	18/6[8)]
LOOP target	Цикл: *CX* ← *CX* – 1 і перехід, якщо *CX* ≠ 0		2	16/4[8)]
LOOPE (*LOOPZ*) *target*	*CX* ← *CX* – 1 і перехід, якщо *CX* ≠ 0 і *ZF* = 1		2	18/6[8)]
LOOPNE (*LOOPNZ*) *target*	*CX* ← *CX* – 1 і перехід, якщо *CX* ≠ 0 і *ZF* = 0		2	19/5[8)]
JA (*JNBE*) *target*	Перехід, якщо перший беззнаковий операнд більший за другий (*CF* = *ZF* = 0)		2	16/4[8)]
JAE (*JNB*) *target*	Перехід, якщо перше беззнакове число не менше за друге (*CF* = 0)		2	16/4[8)]
JB (*JC*) *target*	Перехід, якщо перше беззнакове число менше за друге (*CF* = 1)		2	16/4[8)]
JE (*JZ*) *target*	Перехід, якщо числа рівні (*ZF* = 1)		2	16/4[8)]
JG (*JNLE*) *target*	Перехід, якщо перше знакове число більше за друге (*SF* = (*ZF*&*OF*))		2	16/4[8)]
JGE (*JNL*) *target*	Перехід, якщо перше знакове число більше або дорівнює другому (*SF* = *OF*)		2	16/4[8)]
JL (*JNGE*) *target*	Перехід, якщо перше знакове число менше за друге (*SF* ≠ *OF*)		2	16/4[8)]
JLE (*JNG*) *target*	Перехід, якщо перше знакове число менше або дорівнює другому (*SF* ≠ *OF* або *ZF* = 0)		2	16/4[8)]
JNC (*JAE/JNB*) *target*	Перехід, якщо немає переносу (*CF* = 0)		2	16/4[8)]
JNE (*JNZ*) *target*	Перехід, якщо числа не рівні (*ZF* = 0)		2	16/4[8)]
JNO target	Перехід, якщо немає переповнення (*OF* = 0)		2	16/4[8)]
JNP (JPO) target	Перехід, якщо паритет непарний (*PF* = 0)		2	16/4[8)]
JNS target	Перехід, якщо додатний результат (*SF* = 0)		2	16/4[8)]
JO target	Перехід, якщо є переповнення (*OF* = 1)		2	16/4[8)]
JP (*JPE*) *target*	Перехід, якщо паритет парний (*PF* = 1)		2	16/4[8)]
JS target	Перехід, якщо від'ємний результат (*SF* = 1)		2	16/4[8)]
CALL NEAR target	Внутрішньосегментний виклик процедури (виклик у межах сегмента довжиною 64 кбайт)	*SP* ← *SP* – 2 [*SS*:*SP*] ← *IP* *IP* ← *target*	3	19
CALL NEAR reg		*IP* ← *reg*	2	16
CALL NEAR mem		*IP* ← *mem*	2–4[1)]	21 + *EA*

Продовження табл. 3.11

1	2	3	4	5
CALL FAR target	Міжсегментний виклик процедури (виклик у межах усієї ємності пам'яті 1 Мбайт)	*SP* ← *SP* – 2 [*SS*:*SP*] ← *CS* *SP* ← *SP* – 2 [*SS*:*SP*] ← *IP* *CS*, *IP* ← *target*	5	28
CALL FAR mem		*IP* ← [*mem*] *CS* ← [*mem* + 2]	2–4[1)]	37 + *EA*
RET *RET NEAR*	Повернення з внутрішньосегментної процедури. Необов'язковий параметр *n* задає корекцію значення вказівника стека	*IP* ← [*SS*:*SP*] *SP* ← *SP* + 2	1	8
RET (*n*) *RET NEAR* (*n*)		*SP* ←*SP* + *n*	3	12
RET FAR	Повернення з міжсегментної процедури. Необов'язковий параметр *n* задає корекцію значення вказівника стека	*IP* ← [*SS*:*SP*] *SP* ←*SP* + 2 *CS* ← [*SS*:*SP*]	1 3	18 17
RET FAR (*n*)		*SP* ← *SP* + *n*		
Команди переривань				
INT n	Виконання програмного переривання	*SP* ← *SP* – 2 [*SS*:*SP*] ← *F* *SP*← *SP* – 2 [*SS*:*SP*] ← *CS* *SP* ← *SP* – 2 [*SS*:*SP*] ← *IP* *IP* ← [0000:4*n*] *CS* ← [0000:4*n* + 2]	2	51
INT 3	Виконання програмного переривання 3		1	52
INTO	Виконання програмного переривання 4, якщо прапорець *OF* = 1		1	53/4[9)]
IRET	Повернення з переривання	*IP* ← [*SS*:*SP*] SP ← *SP*+2 *CS* ← [*SS*:*SP*] *SP* ← *SP* + 2 *F* ← [*SS*:*SP*] *SP* ← *SP* + 2	1	24
КОМАНДИ КЕРУВАННЯ СТАНОМ МП				
CLC	Скидання прапорця перенесення	*CF* ← 0	1	2
CMC	Інверсія прапорця перенесення	*CF* ← $\overline{CF}$	1	2
STC	Установлення прапорця перенесення	*CF* ← 1	1	2
CLD	Скидання прапорця напряму	*DF* ← 0	1	2

Закінчення табл. 3.11

1	2	3	4	5
STD	Установлення прапорця напряму	*DF* ← 1	1	2
CLI	Заборона маскованих апаратних переривань	*IF* ← 0	1	2
STI	Дозвіл маскованих апаратних переривань	*IF* ← 1	1	2
HLT	Зупин процесора		1	2
WAIT	Очікування сигналу на лінії *TEST*		1	3
ESC msk/mem	Передача коду команди *msk* або коду й операнда *mem* арифметичному співпроцесору		2 2–4[1)]	2 8 + *EA*
LOCK	Префікс блокування шини на час виконання наступної команди у максимальному режимі		1	2
NOP	Немає операцій		1	2

[1)] Команда займає 2 байт, якщо при адресації комірки пам'яті не використовується зміщення, тобто *disp* = 0, наприклад, позначення комірки пам'яті *mem* відповідає позначенню [*SI*]; команда займає 3 байт, якщо використовується 8-розрядне зміщення *disp*8, наприклад, [*SI* + 25*H*]; команда займає 4 байт, якщо зміщення16-розрядне – *disp*16, наприклад, [*SI* + 1000*H*].

[2)] Команда займає 3 байт при *disp* = 0, 4 байт при *disp*8 і 5 байт при *disp*16.

[3)] Команда займає 4 байт при *disp* = 0, 5 байт при *disp*8 і 6 байт при *disp*16.

[4)] Логічний зсув ліворуч збігається з арифметичним зсувом ліворуч. Допускається замість позначення *SAL* використовувати позначення *SHL*.

[5)] Наведено час виконання рядкової команди без префікса повторення.

[6)] Наведено час виконання рядкової команди із префіксом повторення. У регістрі *CX* записано кількість повторень.

[7)] При встановленому прапорці напряму (*DF* = 1) – операція додавання, інакше – віднімання.

[8)] *m*/*n* – при виконанні переходу команда виконується за *m* тактів, інакше – за *n* тактів.

[9)] При встановленому прапорці (*OF* = 1) команда виконується за 53 такти, у протилежному випадку – за 4 такти.

Значення кількості тактів *EA*, потрібне для обчислення ефективної адреси, залежить від способу адресації операнда (табл. 3.12).

Таблиця 3.12*. Обчислення ефективної адреси *EA

Адресація	Спосіб обчислення	Кількість тактів
Пряма	[*disp*]	6
Непряма	[*BX*], [*BP*], [*SI*], [*DI*]	5
Базова або індексна	[*BX* + *disp*], [*BP* + *disp*], [*SI* + *disp*], [*DI* + *disp*]	9
Базово-індексна без зміщення	[*BP* + *DI*], [*BX* + *SI*], [*BP* + *SI*], [*BX* + *DI*]	7 8
Базово-індексна зі зміщенням	[*BP* + *DI* + *disp*], [*BX* + *SI* + *disp*], [*BP* + *SI* + *disp*], [*BX* + *DI* + *disp*]	11 11 12 12

Вплив команд на значення прапорців ілюструє табл. 3.13, у якій позначено: «+» – команда впливає на прапорець; «–» – не впливає; «1» – встановлює прапорець в одиницю; «0» – скидає прапорець у нуль; «?» – стан невизначений (залежить від конкретних значень операндів).

***Таблиця 3.13*. Установлення прапорців**

Операція	Команди	Прапорці								
		OF	*CF*	*AF*	*SF*	*ZF*	*PF*	*DF*	*IF*	*TF*
Додавання, віднімання	*ADD, ADC, SUB, SBB*	+	+	+	+	+	+	–	–	–
	CMP, NEG, CMPS, SCAS	+	+	+	+	+	+	–	–	–
	INC, DEC	+	–	+	+	+	+	–	–	–
Множення	*MUL, IMUL*	+	+	?	?	?	?	–	–	–
Ділення	*DIV, IDIV*	?	?	?	?	?	?	–	–	–
Десяткова корекція	*DAA, DAS,*	?	+	+	+	+	+	–	–	–
	AAA, AAS	?	+	+	?	?	?	–	–	–
	AAM, AAD	?	?	?	+	+	+	–	–	–
Логічні команди	*AND, OR, XOR, TEST*	0	0	?	+	+	+	–	–	–
Зсув	*RCL, RCR, ROL, ROR dest*	+	+	?	–	–	–	–	–	–
	RCL, RCR, ROL, ROR dest, CL	?	+	?	–	–	–	–	–	–
	SHL, SHR dest	+	+	?	+	+	+	–	–	–
	SHL, SHR dest, CL	?	+	?	+	+	+	–	–	–
	SAR	0	+	?	+	+	+	–	–	–
Відновлення прапорців	*POPF, IRET*	+	+	+	+	+	+	+	+	+
	SAHF	–	+	+	+	+	+	–	–	–
Переривання	*INT, INTO*	–	–	–	–	–	–	–	0	0
Керування прапорцями	*STC*	–	1	–	–	–	–	–	–	–
	CLC	–	0	–	–	–	–	–	–	–
	CMC	–	$\overline{CF}$	–	–	–	–	–	–	–
	STD	–	–	–	–	–	–	1	–	–
	CLD	–	–	–	–	–	–	0	–	–
	STI	–	–	–	–	–	–	–	1	–
	CLI	–	–	–	–	–	–	–	0	–

Приклади виконання команд. Розглянемо приклади виконання групи команд передачі інформації. До цієї групи команд належать команди пересилання даних із регістрів у регістри, з регістрів у пам’ять, із пам’яті у регістри, з пам’яті до пам’яті, у тому числі у стек і зі стека.

Приклад 3.12. Передати дані з регістра *CL* у регістр *BL*. До виконання команди регістр *BL* містить число 101011111, а *CL* – 00001111.

За командою

```
MOV BL,CL          ; BL ← CL
```

(*MOVe* – переслати) вміст регістра *CL* пересилається у регістр *BL*. Після виконання команди вміст регістра *BL* дорівнює 00001111, вміст *CL* не зміниться, тобто в МП вміст двох регістрів стане однаковим:

BL = 00001111, *CL* = 00001111.

Приклад 3.13. Переслати вміст комірки пам'яті *DS*:[100*EH*] у регістр *CX*.

За командою

```
MOV CX, [100EH]  ; CX ← DS: [100EH]
```

вміст 16-розрядної комірки пам'яті з адресою *DS*:[100*EH*] пересилається у 16-розрядний регістр *CX*. Нехай до виконання команди вміст регістра *CX* дорівнював 1234*H*, тобто 0001001000110100_2, а в пам'яті за вказаною адресою знаходиться слово 5678*H*, причому молодший байт слова 78*H* розміщений з адресою *DS*:[100*EH*], старший байт 56*H* – з адресою *DS*:[100*FH*]. Після виконання команди молодший байт слова в пам'яті перепишеться у молодшу частину регістра *CX*, тобто в *CL*, а старший байт – у старшу частину, тобто в *CH*. У результаті вміст *CX* буде 5678*H*, а інформація у пам'яті залишиться незмінною (рис. 3.21).

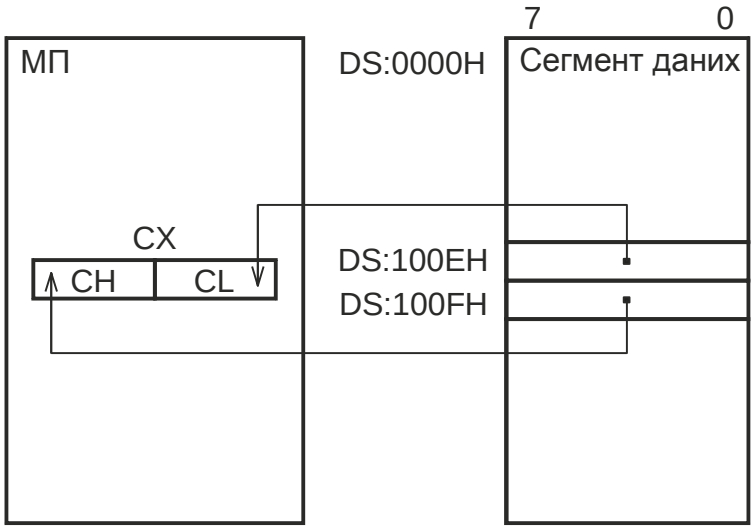


Рис. 3.21. Виконання команди *MOV* [100*EH*], *CX*

Зазначимо, що у двохоперандних командах типу

MOV dest, src

один з операндів має бути вмістом регістра:

MOV r1, r2 ; r1 ← r2

MOV m, r ; m ← r

MOV r, m ; r ← m

Приклад 3.14. Завантажити сегментний регістр *DS* початковим значенням 4000*H*.

Оскільки команди *MOV sr, immed* не існує, то для того, щоб завантажити в сегментний регістр якесь значення, треба записати його у РЗП, а потім передати в сегментний регістр, тобто використати дві команди:

```
MOV AX, 4000H    ; AX ← 4000H
MOV DS, AX       ; DS ← AX
```

Приклад 3.15. Здійснити обмін вмістів двох регістрів *BL* і *CL*.

За командою

```
XCNG BL,CL       ; BL ←→ CL
```

(*eXChaNGe* – обмін) вмісти регістрів *BL* і *CL* міняються місцями.

Нехай значення регістрів *BL* = 10101111, *CL* = 00001111. Після виконання команди вміст

BL = 00001111;

CL = 10101111.

Зазначимо, що при виконанні цієї команди, крім регістрів, які беруть участь у команді, використовуються буферні регістри МП, які зберігають проміжні дані.

Приклад 3.16. Нехай у таблиці з початковою адресою *ES*:*BX* послідовно розміщено байти семисегментного коду цифр від 0 до 9 (рис. 3.22). Замінити значення вмісту регістра *AL* на відповідний семисегментний код.

Для вирішення завдання скористаємося командою

```
XLAT                ; AL ← ES:[BX + AL]
```

де *XLAT* (*indeX Load Accumulator from Table*) – індексне (за непрямою адресацію) завантаження акумулятора даними з таблиці (рис. 3.22). Команда замінює вміст *AL* на вміст комірки пам'яті, розміщеної у сегменті *ES* зі зміщенням *BX* + *AL*.

ES:*BX*	«0»
ES:*BX* + 1	«1»
ES:*BX* + 2	«2»
ES:*BX* + 3	«3»
…	…
ES:*BX* + 9	«9»

Рис. 3.22. Таблиця семисегментних кодів у пам'яті

Ця команда використовується для перекодування символу, який знаходиться в *AL*, на байт з таблиці перекодування. Для коректної роботи команди потрібно, щоб таблиця була розміщена за певною адресою, а саме за початковою адресою *ES*:*BX*. Довжина таблиці не має перебільшувати 256 байт.

Якщо в регістрі *AL* до виконання команди знаходиться, наприклад, число 3, то після виконання в *AL* буде семисегментний код числа 3.

Команду *XLAT* доцільно використовувати для заміни аргументу значенням функції, а значення функцій заздалегідь записувати в таблиці.

Приклад 3.17. Запам'ятати в стеку вміст 16-розрядного регістра загального призначення.

За командою

```
PUSH AX             ; SP ← SP – 2, AX ← SS:[SP]
```

(*PUSH* – занести) вміст вказівника стека *SP* зменшується на два, тобто *SP* ← *SP* – 2, для адресування верхньої вільної комірки стека. Після цього вміст регістра *AX* запам'ятовується у 16-розрядній комірці стека з адресою *SS*:*SP*. Виконання команди проілюстровано на рис. 3.23.

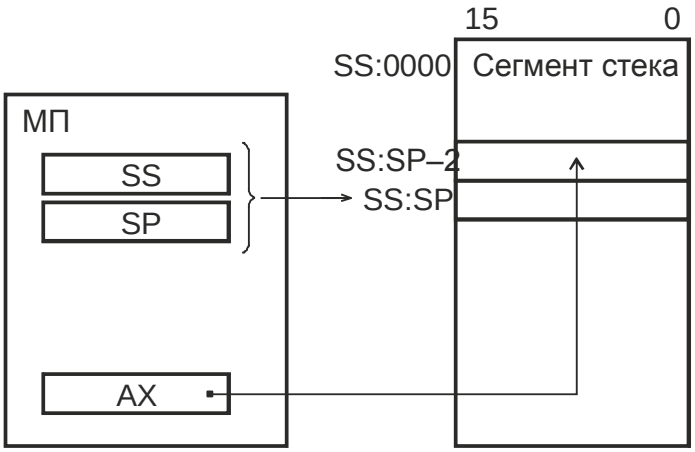


Рис. 3.23. Виконання команди *PUSH AX*

Команди запису у стек та зчитування зі стека можуть оперувати лише з 16-розрядними операндами.

Операції зі стеком є більш швидкодіючими порівняно зі зверненнями до пам'яті з довільною вибіркою завдяки тому, що адреса комірки стека формується автоматично. Програміст повинен лише ініціювати вершину стека на початку програми, тобто записати в регістри *SS* і *SP* початкові значення:

```
MOV AX, 7000H    ; AX ← 7000H
MOV SS, AX       ; SS ← AX
MOV SP, 4000H    ; SP ← 4000H
```

Початковою адресою стека в цьому прикладі є 7000*H*:4000*H*.

Розглянемо приклади виконання арифметичних, логічних команд і команд зсувів.

Приклад 3.18. Додати вміст двох регістрів: *CL* і *DL*. До виконання команди вміст регістрів: *CL* = 10011100, *DL* = 11000101.

За командою

```
ADD CL,DL        ; CL ← CL + DL
```

(*ADD* – додати) додаються два операнди; при цьому результат записується на місце першого операнда. Після виконання команди у регістрі *CL* буде значення 01100001:

1			1	1				– рядок перенесень
	1	0	0	1	1	1	0	0
+	1	1	0	0	0	1	0	1
	0	1	1	0	0	0	0	1

Усі арифметичні команди впливають на прапорці. Так, після виконання цієї команди встановилися:

– в одиничний стан:
  AF – прапорець перенесення з розряду *D*3 у розряд *D*4;
  CF – прапорець перенесення з розряду *D*7;
  OF – прапорець переповнення (у результаті додавання двох знакових від'ємних чисел отримано додатний результат);
– у нульовий стан:
  ZF – прапорець нуля (результат ненульовий);

PF – прапорець паритету (число одиниць у результаті непарне);
SF – прапорець знака (знаковий розряд у результаті дорівнює нулю).

Приклад 3.19. Виконати додавання чотирибайтових операндів, які містяться в сусідніх комірках пам'яті з початковими адресами *DS*:1000*H* і *DS*:2000*H*. Результат запам'ятати у регістрах *CX*, *DX*.

Для виконання цього завдання треба переслати вміст молодшого слова першого доданка в регістр *CX* і додати до молодшого слова другого доданка за допомогою команди *ADD*:

```
MOV CX,[1000H]   ; CX ← DS:[1000H]
ADD CX, [2000H]  ; CX ← CX + DS:[2000H]
```

Для того щоб урахувати перенесення, яке може виникнути при додаванні молодших слів, слід виконати додавання наступних слів операндів за допомогою команди *ADC*, що передбачає додавання значення біта *CF* до отриманої суми. Якщо *CF* = 0, сума не змінюється.

Отже, слід переслати вміст старшого слова першого доданка в регістр *DX* і додати до старшого слова другого доданка:

```
MOV DX,[1002H]   ; DX ← DS:[1002H]
ADC DX, [2002H]  ; DX ← DX + DS:[2002H] + біт CF
```

Зауважимо, що, оскільки слова займають у пам'яті дві сусідні комірки, для адресації старшого слова доданка адресу потрібно збільшити на 2.

Алгоритм команди додавання з урахуванням перенесення *ADC DX,* [2002*H*] (*ADC – Add with Carry –* додати з перенесенням) показано на рис. 3.24.

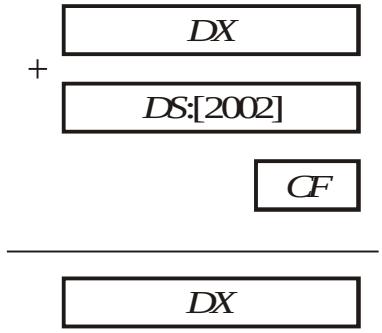


Рис. 3.24. Виконання команди *ADC DX*, [2002*H*]

Додавання багатобайтових операндів виконується так: молодші байти операндів додаються за командою *ADD*, а всі інші байти – за командою *ADC*.

Приклад 3.20. Виконати віднімання від вмісту регістра *AX* числа 5000*H*.

За командою

```
SUB AX,5000H     ; AX ← AX – 5000H
```

(*SUB – SUBtract –* відняти) різниця *AX* – 5000*H* пересилається в акумулятор *AX*.

Приклад 3.21. Виконати віднімання вмісту комірки пам'яті з адресою *DS*:1000*H* та числа 35*H* з урахуванням позики від попередньої операції.

Для того щоб урахувати позику, потрібно виконати віднімання за допомогою команди *SBB*, яка передбачає віднімання біта *CF* від отриманої різниці [*DS*:1000*H*] – 35*H*. Якщо *CF* = 0, то результат віднімання залишиться незмінним:

```
SBB [1000H],35H  ; [DS:1000H] ← [DS:1000H] – 35H – біт CF
```

Команду *SBB* використовують для віднімання багатобайтових операндів з урахуванням позики з попередніх операцій. Віднімання багатобайтових операндів здійснюється таким чином: молодші байти віднімаються за командою *SUB*, а всі інші – за командою *SBB*.

Приклад 3.22. Визначити додатковий код операнда, який міститься в регістрі *BX*.

За командою

```
NEG BX           ; BX ← BX + 1
```

(*NEG – NEGative*) виконується переведення числа в додатковий код, тобто спочатку здійснюється його побітова інверсія, а потім додавання одиниці.

Приклад 3.23. Порівняти значення вмісту акумулятора і регістра *CX*.

За командою

```
CMP AX, CX       ; AX – CX ⇒ F
```

(*CoMPare* – порівняти) від вмісту *AX* віднімається вміст *CX*. При цьому різниця ніде не фіксується, але згідно з нею встановлюються прапорці. Так, прапорець нуля *ZF* установлюється в одиницю, якщо числа тотожні. Якщо значення *AX* за модулем менше ніж *CX*, установлюється прапорець позики *CF*.

Приклад 3.24. Знайти добуток двох беззнакових 8-розрядних операндів, які знаходяться в регістрах *DL* і *AL*.

У команді множення *MUL* (*MULtiplay* – перемножити) беззнакових чисел вказується лише один операнд – *DL*, оскільки другий множник за замовчуванням знаходиться в акумуляторі. Результат множення байтів розміщується у 16-розрядному регістрі *AX*:

```
MUL DL           ; AL × DL → AX
```

Приклад 3.25. Знайти добуток двох беззнакових 16-розрядних операндів, які знаходяться в регістрах *CX* і *AX*.

При множенні двобайтових операндів у мнемоніці команди *MUL* вказується лише один операнд *CX*. Другий операнд за замовчуванням знаходиться в акумуляторі *AX*. Молодша частина добутку зберігається у *AX*, старша – у *DX*. Результат розміщується у регістрах *AX* та *DX*:

```
MUL CX           ; AX × CX → (DX, AX)
```

Приклад 3.26. Знайти результат ділення вмісту акумулятора *AX* на вміст регістра *CL*.

За командою

```
DIV CL           ; AX:CL → AL, остача → AH
```

(*DIVide* – поділити) вміст *AX* ділиться на вміст *CL*, результат записується у *AL*, остача від ділення – у *AH*.

Приклад 3.27. Виконати команду ділення подвійного слова, що зберігається у регістрах *DX*, *AX*, на слово в регістрі *CX*.

При діленні подвійного слова на слово у мнемоніці команди *DIV* вказується лише дільник, який знаходиться в регістрі *CX*. Молодша частина діленого за замовчуванням знаходиться в регістрі *AX*, старша частина – у регістрі *DX*. Результат розміщується в регістрах *AX* та *DX*:

```
DIV CX              ;(DX, AX):CX → AX, остача → DX
```

Приклад 3.28. Виконати команду десяткової корекції результату додавання двох упакованих двійково-десяткових чисел, які знаходяться в регістрах $AL = 25H$, $BL = 37H$. Команду додавання записати так, щоб після її виконання результат був розміщений у регістрі *AL*.

Команда

```
DAA
```

(*Decimal correction of Accumulator at Addition* – десяткова корекція акумулятора при додаванні) використовується після команди додавання і перетворює результат додавання двійково-десяткових чисел на двійково-десяткове число. Корекція полягає в узгодженні перенесень при додаванні десяткових і шістнадцяткових чисел. Така корекція потрібна, оскільки при порозрядному додаванні десяткових чисел перенесення виконується, якщо результат перевищує 9, а при порозрядному додаванні шістнадцяткових чисел – якщо результат перевищує $F = 15_{10}$. Алгоритм десяткової корекції такий:

1. Якщо прапорець $AF = 1$ або молодша тетрада $AL > 9$, то $AL \leftarrow AL + 06$, $AF \leftarrow 1$.
2. Якщо прапорець $CF = 1$ або старша тетрада $AL > 9$, то $AL \leftarrow AL + 60H$, $CF \leftarrow 1$.

Після виконання команди

```
ADD AL,BL
```

вміст регістра *AL* визначається так:

$$\begin{array}{r} 0\ 0\ 1\ 0\ 0\ 1\ 0\ 1 \\ +0\ 0\ 1\ 1\ 0\ 1\ 1\ 1 \\ \hline 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0 \end{array} = 5CH$$

Це число містить літери, тобто не є двійково-десятковим числом, тому після команди *ADD* виконується команда *DAA*, у результаті виконання якої результат додавання змінюється. Значення молодшої тетради результату більше 9. Згідно з алгоритмом

$$\begin{array}{r} 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0 \\ +\qquad\qquad\quad 1\ 1\ 0 \\ \hline 0\ 1\ 1\ 0\ 0\ 0\ 1\ 0 \end{array}_{2-10} = 62$$

Дійсно, 25 + 37 = 62. Результат додавання є двійково-десятковим числом.

Приклад 3.29. Здійснити порозрядну операцію ЛОГІЧНЕ АБО над числами, які містяться в регістрах *BL* = 11011010 та *CL* = 10001001.

Команда

```
OR BL, CL           ; BL ← BL v CL
```

(*OR* – або) виконує операцію ЛОГІЧНЕ АБО, результат якої розміщується на місці першого операнда. Після виконання команди *BL* = 11011011.

Так само порозрядно діють усі логічні операції: *AND, XOR, NOT*.

Логічні команди діють на прапорці таким чином: прапорці *CF* і *OF* скидаються у нуль, значення прапорця *AF* не визначено, прапорці *ZF*, *SF*, *PF* встановлюються відповідно до результату (див. табл. 3.10). Після виконання команди прапорці набувають таких значень: *ZF* = 0, *SF* = 1, *PF* = 1.

Приклад 3.30. Установити прапорці відповідно до результату операції ЛОГІЧНЕ І над числом 38*H*, яке міститься в регістрі *CL*, та числом 35*H*.

За командою

```
TEST CL,35H       ; CL ∧ 35H ⇒ F
```

(*TEST* – перевірка) здійснюється кон'юнкція операндів, за результатом установлюються прапорці, але результат операції не фіксується. Після виконання команди

$$\begin{array}{rcccccccc} & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ \wedge & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ \hline & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{array}$$

прапорці набувають значень відповідно до результату операції: *ZF* = 0, *PF* = 1, *SF* = 0, *CF* = 0, *OF* = 0, стан прапорця *AF* не визначений.

Приклад 3.31. Виконати циклічний зсув вмісту регістра *BX* ліворуч на один розряд.

За командою

```
ROL BX, 1
```

або

```
ROL BX, CL
```

(*ROL – ROtate shift Left* – циклічний зсув ліворуч) здійснюється циклічний зсув вмісту регістра *BX* ліворуч відповідно на один або *CL* розрядів (рис. 3.25). За результатами операції змінюються значення прапорців *CF*, *SF*, *ZF*, *PF*. Команда зсуву на один розряд змінює також прапорець *OF* – він установлюється в одиницю при відповідній зміні знакового розряду.

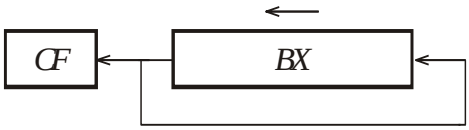


Рис. 3.25. Виконання команди *ROL BX*,1

Розглянемо приклади виконання групи рядкових команд. Рядкові команди оперують з масивами даних – рядками. За замовчуванням один з масивів – джерело інформації – знаходиться в сегменті даних *DS* з початковою адресою, яка визначається вмістом регістра *SI*, другий масив – приймач інформації – знаходиться у додатковому сегменті даних *ES* з початковою адресою, яка визначається вмістом регістра *DI*. Елементами рядків є слова або байти. Напрямок обробки інформації у рядках визначається прапорцем напряму *DF*. При *DF* = 0 обробка інформації починається з молодших адрес, при *DF* = 1 – із старших адрес.

Приклад 3.32. Передати вміст 8-розрядної комірки пам'яті з адресою *DS*:*SI* у комірку пам'яті з адресою *ES*:*DI*.

Для виконання цього завдання використовується команда *MOVSB* (*MOVe Sequence Byte* – перемістити послідовність байтів), яка пересилає байт або послідовність байтів з однієї області пам'яті в іншу. Якщо у мнемоніці команди використовується префікс *REP*, то пересилається послідовність байтів, при цьому кількість байтів у послідовності задається вмістом регістра *CX*. Якщо у мнемоніці команди префікса *REP* немає, то пересилається вміст однієї комірки пам'яті. За замовчуванням при виконанні команди *MOVSB* пересилається вміст комірки пам'яті з адресою *DS*:*SI* у комірку пам'яті з адресою *ES*:*DI*.

Приклад 3.33. Переслати слово, молодший байт якого зберігається в комірці пам'яті з адресою *DS*:*SI*, а старший байт – у комірці з адресою *DS*:*SI* + 1, у комірку пам'яті з адресою *ES*:*DI*.

За командою

```
MOVSW
```

(*MOVe Serial Word* – перемістити рядок слів) виконується пересилання слова (за замовчуванням використовуються ті самі адреси комірок, що і в прикл. 3.32).

Приклад 3.34. Переслати масив, що складається зі 100 слів, з області пам'яті, яка має початкову адресу *DS*:*SI* (*DS* = 2000*H*, *SI* = 1000*H*), в область пам'яті, яка має початкову адресу *ES*:*DI* (*ES* = 3000*H*, *DI* = 4000*H*).

За командою

```
REP MOVSW
```

(*REPeat MOVSW*) виконується пересилання слів, кількість яких зберігається за замовчуванням у регістрі *CX*.

Пересилання масиву здійснюється виконанням такої послідовності команд:

```
MOV   CX,100    ; Занести у CX значення довжини масиву
CLD             ; Скинути прапорець напрямку для автоінкрементації
                ; адрес масивів
MOV   AX,2000H  ; Занести в регістри DS і SI адреси сегмента та
MOV   DS,AX     ; зміщення джерела даних
MOV   SI,1000H
MOV   AX,3000H  ; Занести в регістри ES і DI адреси сегмента та
MOV   ES,AX     ; зміщення регістра-приймача даних
MOV   DI,1000H
REP   MOVSW     ; Переслати рядок слів
```

Розглянемо приклади виконання групи команд передачі керування. Команди передачі керування зазвичай змінюють вміст вказівника команд *IP*, а деякі і вміст регістра сегмента кодів *CS*. За допомогою цих команд можна змінити послідовність виконання команд у програмі, оскільки регістр *CS* містить базову адресу поточного сегмента кодів, з якого вибираються команди, а регістр *IP* – адресу, яка задає зміщення команди відносно початку сегмента кодів. Після виконання команди передачі керування пристрій керування МП, використовуючи новий вміст регістрів *CS* і *IP*, вибирає з пам'яті наступну команду. У такий спосіб виконується переоформлення черги команд, які надходять у блок операційного пристрою (див. рис. 3.11).

До цієї групи команд належать: команди безумовного й умовного переходів; команди викликів підпрограм і повернень з підпрограм; команди циклів; команди переривань.

Приклад 3.35. Виконати безумовний внутрішньосегментний перехід.

За командою

```
JMP NEAR LABEL
```

або

```
JMP LABEL
```

(*JuMP NEAR* – стрибок неподалік) здійснюється перехід до виконання команд, перша з яких позначена міткою *LABEL*, яка знаходиться у поточному сегменті кодів, тобто в межах 64 кбайт. Якщо тип переходу не вказано, за замовчуванням виконується тип *NEAR*. При переході у межах сегмента змінюється вміст програмного лічильника *IP*←*'LABEL'*, де *LABEL* – символічна адреса або мітка, а *'LABEL'* – зміщення у сегменті кодів цієї мітки.

Приклад 3.36. Виконати безумовний міжсегментний перехід.

За командою

```
JMP FAR LABEL
```

(*JuMP FAR* – стрибок далеко) здійснюється перехід до виконання команд, першу з яких позначено міткою *LABEL*. Ця мітка знаходиться в межах усієї пам'яті ємністю 1 Мбайт. При міжсегментному переході змінюється як вміст програмного лічильника *IP*, так і вміст сегментного регістра кодів *CS* згідно з міткою *LABEL*.

Приклад 3.37. Виконати умовний перехід залежно від стану прапорця *CF*.

За командою

```
JC LABEL
```

(*Jump if Carry* – стрибок, якщо є перенесення) здійснюється перехід виконання команд на мітку *LABEL* лише у тому разі, якщо прапорець *CF* = 1. В іншому випадку, якщо *CF* = 0, здійснюється перехід до виконання команди, наступної після команди умовного переходу.

Приклад 3.38. Викликати підпрограму, розміщену в деякому місці пам'яті ємністю 1 Мбайт із символічною адресою *NAME*.

Для переходу до виконання підпрограми використовують команду *CALL*. При виконанні команди *CALL* змінюються значення регістрів *CS* і *IP*. Перед виконанням команди запам'ятовуються значення цих регістрів, щоб після виконання підпрограми можна було повернутися до виконання основної програми. Запам'ятовування значень регістрів відбувається у стеку.

За командою

```
CALL FAR NAME
```

(*CALL* – виклик) викликається підпрограма з адресою *NAME*:

- вміст *SP* зменшується на 2;
- у комірку пам'яті з адресою *SS*:*SP* пересилається вміст регістра *CS*;
- вміст *SP* зменшується на 2;
- у комірку пам'яті з адресою *SS*:*SP* пересилається вміст регістра *IP*;
- в *IP* і *CS* завантажуються нові значення, які відповідають символічній адресі *NAME*.

У результаті цих дій у стеку запам'ятовується вміст регістрів *CS* і *IP*, тобто повна адреса *CS*:*IP* тієї команди, яку треба буде виконати після закінчення підпрограми *NAME*. Останньою командою підпрограми, що викликається, є команда *RET FAR*. За цією командою зі стека витягуються два слова, які були записані при виклику підпрограми, і заносяться у регістри *CS* і *IP*, тобто відновлюється поточна адреса *CS*:*IP* команди основної програми.

Приклад 3.39. Виконати послідовність команд 100 разів.

Для повторення виконання послідовності команд 100 разів потрібно в регістрі *CX* задати кількість повторень

```
MOV CX,100
```

Потім записується послідовність команд, перша з яких позначається міткою:

```
M1: <послідовність команд>
```

та виконується команда

```
LOOP M1
```

(*LOOP* – петля). Команда *LOOP M*1 зменшує вміст *CX* на одиницю, а потім порівнює його з нулем. Якщо $CX \neq 0$, то здійснюється перехід до виконання команди з міткою *M*1. Якщо $CX = 0$, то виконується команда, наступна за *LOOP M*1 (вихід із циклу).

До команд циклів належать:

```
LOOPE M1
```

(*LOOP if Equal* – петля, якщо дорівнює) та

LOOPNE M1

(*LOOP if Not Equal* – петля, якщо не дорівнює). Ці команди реалізують вихід із циклу, якщо або $CX = 0$, або виконується додаткова умова: значення прапорця $ZF = 1$ (для команди *LOOPE*) або $ZF = 0$ (для команди LOOPNE). Прапорець *ZF* може бути встановлений або скинутий у результаті виконання однієї з команд циклу.

Приклад 3.40. Перейти до підпрограми обробки переривання типу $n = 8$.

Мікропроцесор *i*8086 може обробляти 256 типів переривань. У спеціальній області пам'яті, розміщеній за початковою адресою 0000:0000, яка називається *картою векторів переривань*, записано точки входів у підпрограми обробки переривань. За адресою 0000:4×<номер переривання> зберігаються значення *IP*, за адресою 0000:4 ×<номер переривання> + 2 зберігаються значення *CS*. Нові значення *CS*:*IP* визначають адресу першої команди підпрограми обробки переривання.

Команда

```
INT 8
```

ініціює таку послідовність дій:

- скидаються прапорці *IF* і *TF*, що забороняє переривання та покрокову роботу МП;
- вміст *SP* зменшується на 2;
- у комірку пам'яті з адресою *SS*:*SP* пересилається вміст регістра прапорців *F*;
- вміст *SP* зменшується на 2;
- у комірку пам'яті з адресою *SS*:*SP* пересилається вміст регістра *CS*;
- вміст *SP* зменшується на 2;
- у комірку пам'яті з адресою *SS*:*SP* пересилається вміст регістра *IP*;

– номер переривання множиться на 4 (4·8 = 20*H*);
– в *IP* і *CS* завантажуються нові значення з карти векторів переривань (початкова адреса 0000*H*):

$$IP \leftarrow [0000{:}0020H],$$

$$CS \leftarrow [0000{:}0022H].$$

У результаті цих дій здійснюється міжсегментний непрямий виклик підпрограми обробки переривання, причому адреса підпрограми однозначно визначається номером переривання.

Отже, за командою *INT* 8 у стеку буде записано вміст регістрів *IP*, *CS* і *F*, а потім у регістри *IP* і *CS* запишуться нові значення з карти векторів переривань; МП перейде до виконання підпрограми обробки переривання з номером 8.

Виконання команди *INT n* може бути ініційоване і програмно і апаратно. У першому випадку машинний код команди *INT* зчитується з програмної пам'яті, у другому – формує на шині даних систему переривань.

Розглянемо приклади виконання групи команд керування станом МП. До цієї групи належать команди: скидання, встановлення, інверсії прапорців; команда *ESC* (перемикання на співпроцесор); *LOCK* (захоплення шини); *NOP* (немає операції); *HLT* (зупин); *WAIT* (очікування).

Приклад 3.41. Скинути прапорець напряму *DF* у стан логічного нуля.
За командою

```
CLD                ; DF ← 0
```

(*CLear DF* – очищення *DF*) прапорець *DF* скидається у стан логічного нуля.

Приклад 3.42. Установити прапорець напряму *DF* у стан логічної одиниці.
За командою

```
STD                ; DF ← 1
```

(*SeT DF* – встановлення *DF*) прапорець *DF* встановлюється у стан логічної одиниці.

Контрольні запитання

1. На які групи поділяють команди МП *i*8086?
2. Які групи команд не впливають на прапорці?
3. Укажіть значення прапорців, які встановлюються при додаванні чисел 25*H* і 97*H*?
4. Які дії виконує МП при виклику процедури типу *FAR*; типу *NEAR*?
5. Які прапорці змінюються при викликах переривань?

3.4. Побудова модуля центрального процесора на основі *i*8086

Для побудови модуля ЦП потрібно забезпечити синхронізацію роботи системи та узгодження роботи ЦП із системною шиною.

Схема синхронізації. Для синхронізації використовують генератор тактових імпульсів *i*8284, який генерує сигнали синхронізації для ЦП і

периферійних пристроїв, а також синхронізує зовнішні сигнали готовності *READY* і початкового встановлення *RESET* з тактовими сигналами МП. Генератор тактових імпульсів (рис. 3.26) містить подільники частоти на 3 і на 2 та логіку керування сигналами скидання і готовності. Робота задавального генератора (ЗГ) стабілізується кварцовим резонатором, який приєднується до входів *X*1, *X*2. Вхід *TANK* використовується для додаткового приєднання паралельного резонансного *LC*-контура, що дозволяє працювати на вищих гармоніках кварцового резонатора. При цьому опорна частота ЗГ визначається параметрами контура і дорівнює $\frac{1}{2\pi\sqrt{LC}}$. Вхід $F/\overline{C}$ дозволяє обрати зовнішній ($F/\overline{C}$ = 1) або внутрішній ($F/\overline{C}$ = 0) генератор. У випадку обрання зовнішнього генератора із частотою імпульсів F_{EFI} його підключають до входу *EFI*.

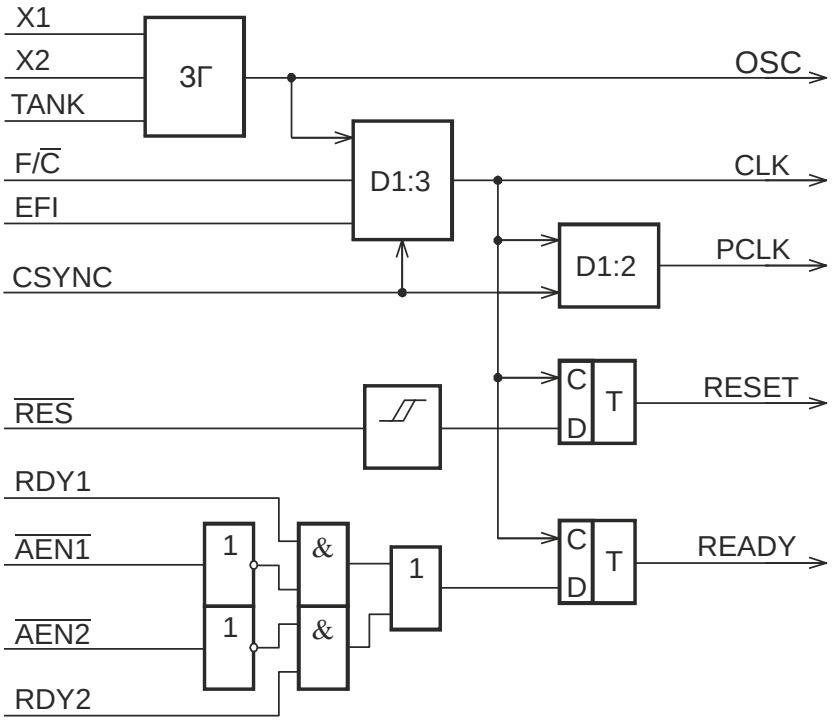


Рис. 3.26. Структурна схема ВІС генератора *i*8284

Схема формування тактових імпульсів формує сигнали: *CLK* – тактової частоти F_{CLK} для ЦП, *PCLK* – тактової частоти F_{PCLK} для керування периферійними ВІС, *OSC* – тактової частоти ЗГ, які потрібні для керування пристроями та контролю частоти. Частоти цих сигналів зв'язані співвідношеннями: $F_{OSC} = 3F_{CLK} = 6F_{PCLK}$ у режимі внутрішнього генератора та $F_{EFI} = 3F_{CLK} = 6F_{PCLK}$ у режимі зовнішнього генератора. Вихідний сигнал *CLK* формується одним з трьох способів: 1) з коливань основної частоти кварцового резонатора, приєднаного до входів *X*1 та *X*2; 2) з третьої гармоніки кварцового резонатора, що виділяється *LC*-фільтром,

з'єднаним із входом *TANK*; 3) від зовнішнього генератора, підключеного до входу *EFI*. Схема формування тактових імпульсів має вхід зовнішньої синхронізації *CSYNC*, за допомогою якого можна синхронізувати роботу декількох генераторів тактових імпульсів, які входять до складу системи. Сигнал *CSYNC* впливає також на подільник частоти на 2: при *CSYNC* = 0 робота подільника зупиняється, при *CSYNC* = 1 – відновлюється.

Вихідний сигнал *READY* використовується для підтвердження готовності до обміну. Високий рівень цього сигналу вказує на наявність даних на *DB*. Схема формування сигналу *READY* побудована так, щоб спростити вмикання системи в інтерфейсну шину стандарту *Multibus*, і містить дві ідентичні пари сигналів *RDY*1, $\overline{AEN1}$ та *RDY*2, $\overline{AEN2}$, об'єднаних схемою АБО. Сигнали *RDY*1 та *RDY*2 формуються елементами, які входять до складу системи, і свідчать про їхню готовність до обміну. Сигнали $\overline{AEN1}$ та $\overline{AEN2}$ дозволяють формування сигналу *READY* за сигналами *RDY*1 та *RDY*2, підтверджуючи адресацію елементів.

Схема формування вихідного сигналу скидання *RESET* має на вході тригер Шмітта, а на виході – тригер, що формує фронт сигналу *RESET* по зрізу сигналу *CLK*. Зазвичай до входу $\overline{RES}$ приєднується *RC*-коло, що забезпечує автоматичне формування сигналу при вмиканні джерела живлення.

Графічне позначення генератора ілюструє рис. 3.27.

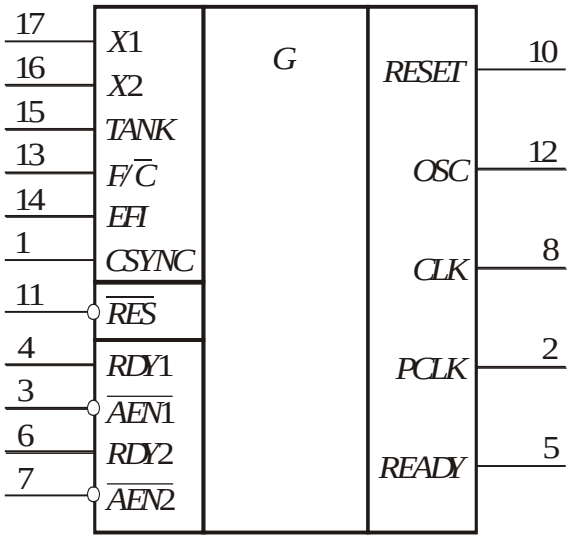


Рис. 3.27. Графічне позначення ВІС генератора *i*8284

Інтерфейс ЦП із системною шиною виконує такі функції:

1) демультиплексування шини адреси/даних (розподіл її на шину адреси *AB* та шину даних *DB*);
2) буферизація шин (збільшення навантажувальної здатності ліній шин, забезпечення можливості їхнього переходу в *Z*-стан);
3) формування сигналів керування.

Виконання першої функції здійснюється за допомогою регістрів-фіксаторів, наприклад, буферних регістрів *i*8282, *i*8283. Узагальнена структурна схема регістра-фіксатора (рис. 3.28) містить вісім *D*-тригерів з вихідними схемами *SW*, які мають три стани. Сигнали запису інформації *STB* і дозволу вибірки $\overline{OE}$ спільні для всіх тригерів ВІС. У буферному регістрі *i*8282 (рис. 3.29) до схем *SW* приєднуються прямі виходи *D*-тригерів, а в буферному регістрі *i*8283 (рис. 3.30) – інверсні виходи. Якщо сигнал має високий рівень, то на вході *STB* стан вхідних ліній *DI*7–*DI*0 передається на вихідні лінії *DO*7–*DO*0. Запис інформації в *D*-тригерах здійснюється по зрізу сигналу *STB*. Малий вхідний і досить великий вихідний струми дозволяють використовувати ВІС буферних регістрів як регістри-фіксатори або шинні формувачі. При використанні буферних регістрів як шинних формувачів вхід *STB* з'єднується з виводом живлення +5 В через резистор опором 1 кОм, а вхід $\overline{OE}$ – із спільною шиною.

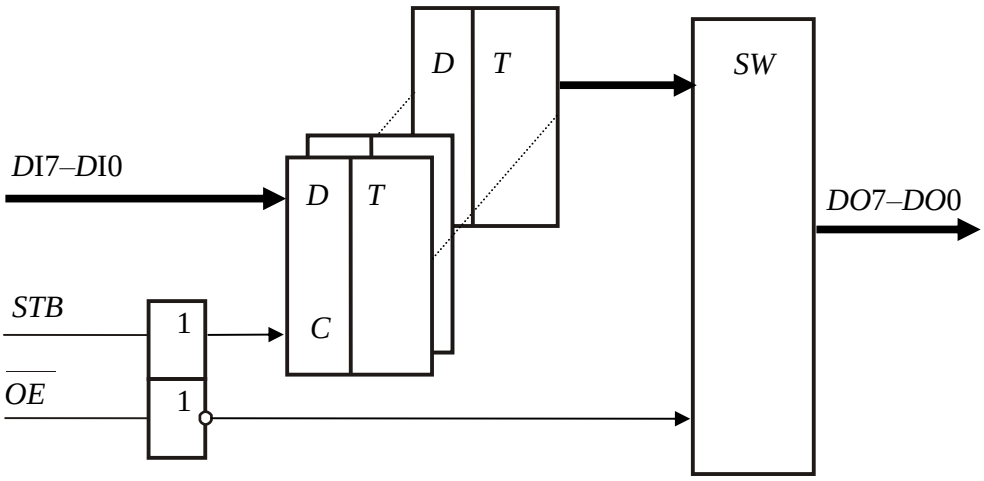


Рис. 3.28. Структурна схема буферного регістра

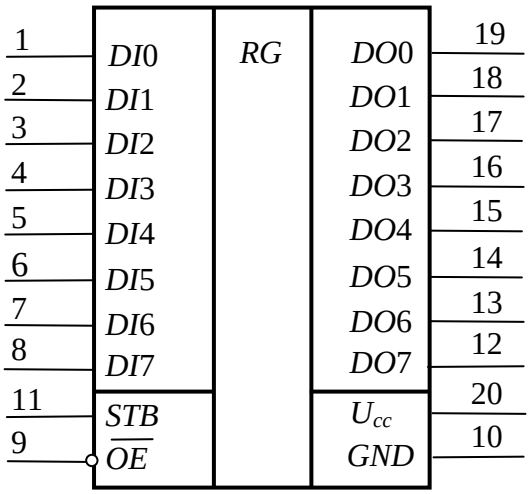


Рис. 3.29. Графічне позначення буферного регістра *i*8282

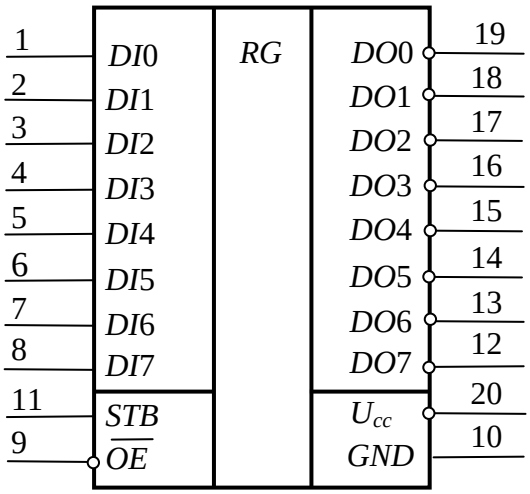


Рис. 3.30. Графічне позначення буферного регістра *i*8283

Для збільшення навантажувальної здатності (друга функція) двонапрямленої шини даних використовують 8-розрядні шинні формувачі *i*8286, *i*8287. Формувач *i*8286 не інвертує дані, а *i*8287 інвертує.

Структурна схема шинного формувача (рис. 3.31) містить вісім однакових функціональних блоків із трьома станами і спільними сигналом керування напрямом передачі T та сигналом дозволу передачі $\overline{OE}$.

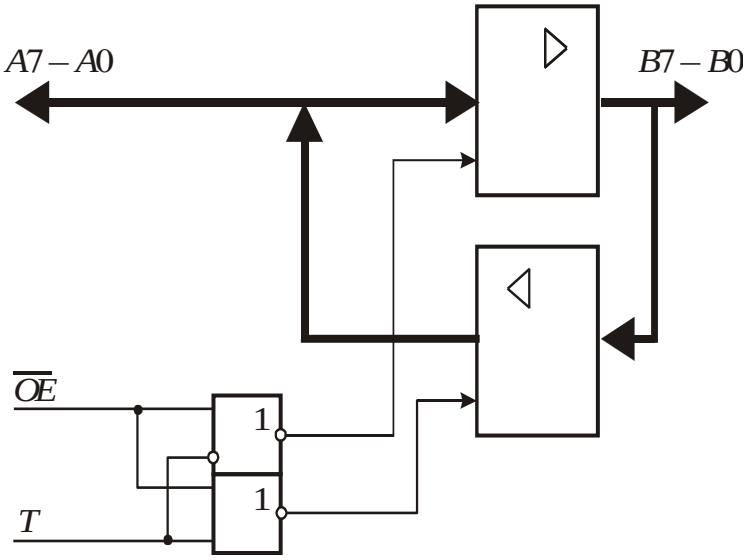


Рис. 3.31. Структурна схема шинного формувача

При низькому рівні сигналу T ($T = 0$) здійснюється передача даних з ліній *B*7–*B*0 на лінії *A*7–A0, при високому рівні сигналу ($T = 1$) – передача з ліній *A*7–*A*0 на лінії *B*7–*B*0. При $\overline{OE} = 0$ передачу дозволено, при $\overline{OE} = 1$ – заборонено.

Графічні позначення формувачів *i*8286, *i*8287 показано на рис. 3.32 і 3.33 відповідно.

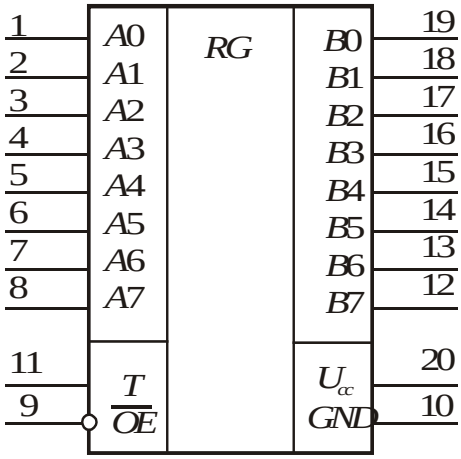


Рис. 3.32. Графічне позначення шинного формувача *i*8286

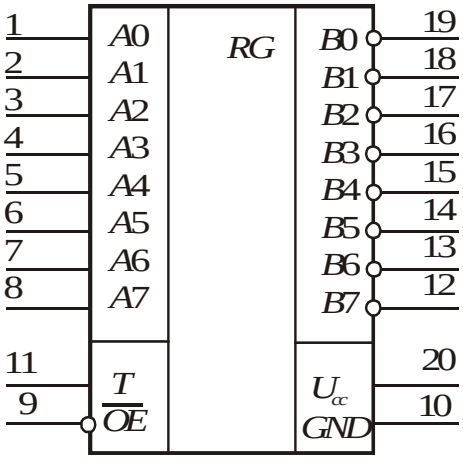


Рис. 3.33. Графічне позначення шинного формувача *i*8287

Третя функція реалізується за допомогою додаткових логічних елементів, які формують сигнали шини керування з вихідних сигналів ВІС МП.

На рис. 3.34 показано приклад функціональної схеми модуля ЦП для однопроцесорних систем. Мікропроцесор *i*8086 увімкнений у мінімальному режимі. Схема синхронізації реалізована на базі ВІС тактового генератора *i*8284, на вхід *RDY*1 якої подається сигнал готовності зовнішніх пристроїв або пам'яті до обміну. У мінімальному режимі використовується одна шина, тому вхід *RDY*2 з'єднаний через резистор з виводом живлення. Демультиплексування шини адреси/даних і шини адреси/стану на дві шини здійснюється за допомогою трьох регістрів-фіксаторів *i*8282.

Відзначимо, що сигнал дозволу старшого байта $\overline{BHE}$, що з'являється водночас з дійсною адресою, також фіксується в одному з розрядів регістрів-фіксаторів. Сигнали $\overline{BHE}$ і *A*0 використовуються для вибірки банків системи пам'яті. Формувач 16-розрядної шини даних виконано на двох ВІС шинних формувачів *i*8286.

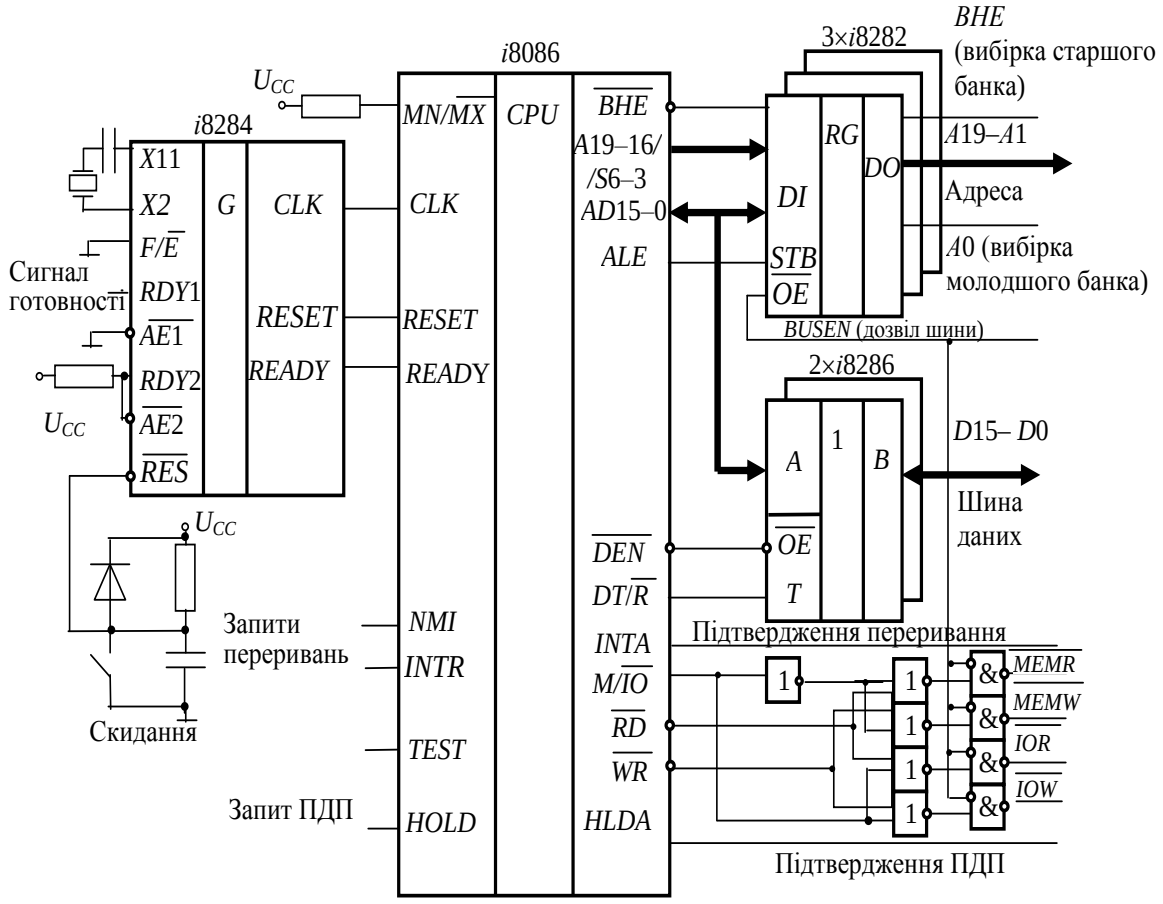


Рис. 3.34. Функціональна схема модуля центрального процесора

У мінімальному режимі процесор формує керувальні сигнали шин формувачів і регістрів-фіксаторів, а також сигнали $M / \overline{IO}$, $\overline{RD}$, $\overline{WR}$, з яких за

допомогою логічних елементів формуються чотири сигнали керування читанням/записом для пам’яті і ПВВ. Шини адреси, даних і керування переводяться у *Z*-стан сигналом $\overline{BUSEN}$, що формує контролер ПДП.

Контрольні запитання

1. Поясніть призначення вхідних та вихідних сигналів схеми синхронізації.
2. Опишіть функції ВІС генератора *i*8284 та інтерфейсу ЦП із системною шиною.
3. Поясніть принцип функціонування схеми буферного регістра.
4. Поясніть принцип функціонування шинного формувача.

Розділ 4

ОДНОКРИСТАЛЬНІ УНІВЕРСАЛЬНІ МІКРОПРОЦЕСОРИ (СТАРШІ МОДЕЛІ)

4.1. Мікропроцесор *i*80286

Загальні відомості. Мікропроцесор *i*80286 належить до другого покоління 16-розрядних МП. Він виконаний за технологією 1,5 мкм, містить 134 000 транзисторів і працює з тактовою частотою 12,5 МГц. За рахунок удосконалення архітектури швидкодія МП *i*80286 у шість разів вища, ніж МП *i*8086 з тактовою частотою 5 МГц. Розрядність регістрів дорівнює 16. Шина адреси 24-розрядна, що дозволяє адресувати $2^{24} = 16$ Мбайт пам'яті. Простір адрес введення-виведення становить 64 кбайт. Система команд містить усі команди *i*8086, декілька нових команд загального призначення і групу команд керування захистом. Мікропроцесор *i*80286 має спеціальні засоби для роботи у системах з багатьма користувачами та багатозадачних режимах. Його найсуттєвішою відмінністю від МП серії *i*8086/88 є механізм керування адресацією пам'яті, що забезпечує чотирирівневу систему захисту та підтримку віртуальної пам'яті. Спеціальні засоби призначено для підтримки механізму перемикання задач. Мікропроцесор *i*80286 має засоби контролю за переходом через межу сегмента.

Мікропроцесор може працювати в двох режимах:

1. 8086 *Real Address Mode* (або *Real Mode*) – режим реальної адресації або реальний режим. У цьому режимі МП *i*80286 фактично являє собою високошвидкісний МП *i*8086 і адресує 1 Мбайт пам'яті.

2. *Protected Virtual Address Mode* (або *Protected Mode*) – захищений режим віртуальної адресації, або просто захищений режим. У цьому режимі МП адресує до 16 Мбайт пам'яті, а при використанні механізму сторінкової адресації до 1 Гбайт віртуальної пам'яті кожної задачі.

Перемикання в захищений режим здійснюється швидко – однією командою (із заздалегідь підготовленими таблицями дескрипторів), а в режим реальної адресації – повільно лише через апаратне скидання процесора. У *MS DOS* використовується реальний режим. Захищений режим використовується в операційних системах типу *XENIX*, *UNIX*, *OS*/2, *NetWare*286, *MS Windows*.

Для процесора *i*80286 можливі 256 різноманітних типів переривань. Система переривань відрізняється від системи МП *i*8086 наявністю переривання при виникненні особливих умов у ході виконання команд (наприклад, при розміщенні двобайтового операнда в останній комірці сегмента даних зі зміщенням *FFFFH*). Таке переривання називається *особливим випадком* або *винятком.* На відміну від переривань після обробки винятків (крім винятку 9, що стосується співпроцесора) керування передається знову тій самій команді (включаючи всі префікси), що викликала переривання. Після усунення умов, які викликали виняток, відбувається повторне виконання команди.

Організація пам'яті. У реальному режимі адресація пам'яті переважно така ж, як і у МП *i*8086. Відмінність полягає у можливості використання додаткового блока пам'яті ємністю 64 кбайт. Якщо в ході виконання команди при обчисленні адреси комірки пам'яті виникає перенесення у двадцятий розряд шини адреси *A*20, процесор починає працювати з комірками пам'яті, адреси яких лежать у діапазоні 100000*H*–10*FFFFH*.

Приклад 4.1. Знайти значення фізичної адреси операнда в реальному режимі при виконанні команди *MOV AL*, [*SI*]; пересилання в регістр *AL* вмісту комірки пам'яті з адресою *DS*:*SI*, якщо вмістом сегментного регістра *DS* є число 0*F*802*H*, вмістом регістра *SI* – 0*B*175*H*.

Для обчислення фізичної адреси додамо до значення *DS* чотири нулі праворуч:

$$DS(0000) = 1111\ 1000\ 0000\ 0010\ 0000B = 0F8020H.$$

Виконавши операцію додавання отриманої величини з вмістом регістра *SI*, отримаємо фізичну адресу:

$$\begin{array}{l} 1111 \qquad\qquad 11 \qquad \text{– рядок перенесень} \\ +\,11111000\ 0000\ 0010\ 0000 \\ \qquad 1011\ 0001\ 0111\ 0101 \\ \hline 1\ 0000\ 0011\ 0001\ 1001\ 0101 = 103195H \end{array}$$

Отже, при обчисленні фізичної адреси отримане одиничне значення розряду *A*20 означає, що операнд буде розміщено у другому мегабайті фізичної пам'яті.

Ще однією особливістю реального режиму *i*80286 є можливість контролю за переходом за межі сегмента. При адресації слова зі зміщенням 0*FFFFH* генерується виняток 13 (*Segment Overrun Exception*). У разі спроби виконання команди *ESC* з операндом пам'яті, що не вміщається у сегменті, генерується виняток 9 – *Processor Extension Segment Overrun Interrupt*.

У захищеному режимі також використовується сегментна адресація; кількість сегментів може бути від 1 до 16 Мбайт, довжина сегментів задається і може варіюватися від 1 до 64 кбайт, задаються атрибути або права доступу до сегмента (дозвіл запису або тільки читання, рівні привілеїв тощо). Кожний сегмент характеризується 8-розрядною структурою даних – *дескриптором сегмента* – що містить інформацію про базову адресу сегмента, його межу й атрибути. Дескриптори розміщуються у

спеціальних таблицях – глобальній дескрипторній таблиці *GDT* або локальній дескрипторній таблиці *LDT*, які зберігаються в ОЗП. Незалежно від рівня привілею програма не може звертатися до сегмента доти, доки він не описаний у дескрипторній таблиці.

У захищеному режимі вміст сегментних регістрів називається *селекторами сегментів.* Вони використовуються для пошуку базової (початкової) адреси сегмента в одній з дескрипторних таблиць. Формат селектора показано на рис. 4.1.

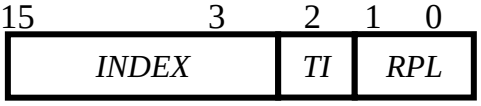


Рис. 4.1. Формат селектора сегмента

Селектори, що завантажуються у 16-розрядні сегментні регістри *CS, DS, SS, ES*, мають три поля:

– *RPL* (*Requested Privilege Level*) (біти 0 і 1) – запрошений рівень привілеїв сегмента;
– *TI* (*Table Indicator*) (біт 2) – індикатор використання таблиці дескрипторів (при *TI* = 1 використовується глобальна таблиця, при 0 – локальна);
– *INDEX* (біти 3–15) – номер дескриптора в таблиці.

Глобальна дескрипторна таблиця єдина. Вона містить дескриптори для всіх задач, які виконує МП у багатозадачному режимі. Локальних дескрипторних таблиць може бути кілька – для кожної задачі можна задати свою локальну дескрипторну таблицю.

Фізична 24-розрядна адреса операнда знаходиться додаванням початкової 24-розрядної адреси сегмента із дескрипторної таблиці й адреси-зміщення, яку вказано у команді. Формування фізичної 24-розрядної адреси у захищеному режимі ілюструє рис. 4.2.

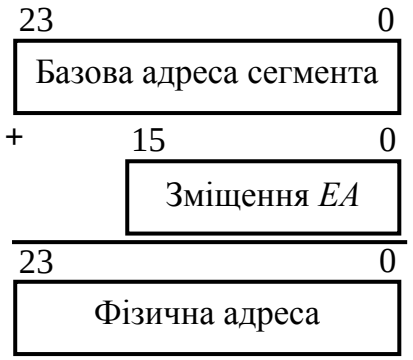


Рис. 4.2. Формування фізичної 24-розрядної адреси у захищеному режимі

Приклад 4.2. Знайти значення фізичної адреси комірки пам'яті [*DS*:*SI*], якщо базова адреса сегмента даних дорівнює 456789*H*, а вміст регістра *SI* – 1234*H*.

Виконавши операцію додавання 24-розрядної базової адреси з адресою-зміщенням, тобто з вмістом регістра *SI*, отримаємо фізичну адресу:

$$\begin{array}{r} 0100\ 0101\ 0110\ 0111\ 1000\ 1001 \\ + \quad 0001\ 0010\ 0011\ 0100 \\ \hline 0100\ 0101\ 0111\ 1001\ 1011\ 1101 = 4579BBH \end{array}$$

Отже, фізична адреса дорівнює 4579*BBH*.

Програмна модель. До програмної моделі МП *i*80286 (рис. 4.3) входять 19 програмно доступних регістрів і 6 недоступних (тіньових). Із 19 програмно доступних регістрів МП *i*80286 14 повторюють регістри процесора *i*8086 (підрозд. 4.5).

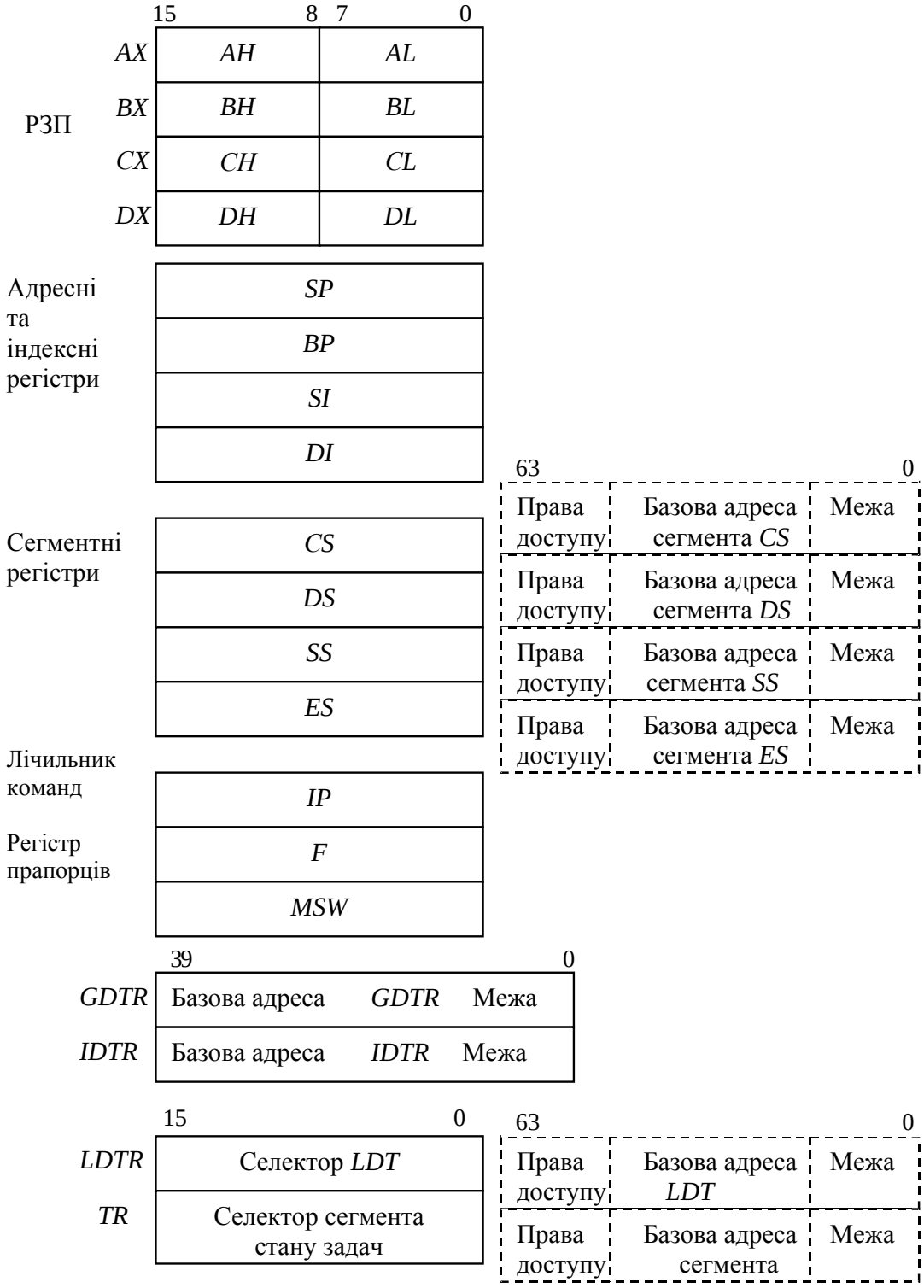


Рис. 4.3. Програмна модель МП *i*80286

Нові регістри такі:

1. 40-розрядний регістр *GDTR* (*Global Descriptor Table Register*) – регістр глобальної дескрипторної таблиці. Призначений для задання розміщення глобальної дескрипторної таблиці в пам'яті. Регістр *GDTR* безпосередньо містить базову адресу і значення межі таблиці, яке задає її розмір. Розмір таблиці більший від значення межі на одиницю.

2. 16-розрядний регістр *LDTR* (*Local Descriptor Table Register*) – регістр-селектор локальної дескрипторної таблиці. Регістр є 16-розрядним на відміну від 40-розрядного регістра *GDTR*. Він не містить безпосередньо всієї інформації про сегмент, як *GDTR*. Але його вміст вказує, де в глобальній дескрипторній таблиці знаходиться інформація про початкову адресу, межу і права доступу до локальної таблиці. Ця інформація переписується у тіньовий програмно недоступний регістр.

3. 40-розрядний регістр *IDTR* (*Interrupt Descriptor Table Register*) – регістр дескрипторної таблиці переривань. Завдяки цьому регістру в МП *i*80286 з'явилася можливість розміщувати таблицю векторів переривань у довільному місці ОЗП, а не з нульової адреси, як у МП *i*8086. Регістр *IDTR* за структурою аналогічний регістру *GDTR*.

4. 16-розрядний регістр задачі *TR* (*Task Register*) – регістр-селектор сегмента стану поточної задачі *TSS* (*Task State Segment*). Такі сегменти асоціюються з кожною задачею. Їх призначено для збереження контексту задачі під час перемикання задач.

5. 16-розрядний регістр стану машини *MSW* (*Mashine State Word*), що керує режимом процесора і містить такі біти (рис. 4.4):

MSW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													TS	*EM*	*MP*	*PE*

Рис. 4.4. Регістр стану машини *MSW*

1) *PE* (*Protecrion Enable*) – дозвіл захисту. Установлення цього прапорця переводить МП у захищений режим. Повернення до реального режиму можливе тільки за сигналом *RESET*.

2) *MP* (*Monitor Processor Extension*) – наявність співпроцесора. Якщо арифметичний співпроцесор *i*80287 з'єднаний з процесором *i*80286, то при ініціалізації операційна система має встановити цей біт у стан логічної одиниці. Тоді при виконанні команди *WAIT*, а також при виконанні команди з префіксом *ESC* і значенні *TS* = 1 МП буде генерувати виняток 7.

3) *EM* (*Processor Extension Emulated*) – емуляція співпроцесора. Установлення цього біта викликає виняток 7 при виконанні кожної команди арифметичного співпроцесора, що дозволяє здійснювати його програмну емуляцію.

4) *TS* (*Task Switch*) – перемикання задач. Зі встановленням цього біта команда, яка належить до співпроцесора, викличе виняток 7. Це дозволяє програмно визначити, чи відноситься контекст співпроцесора до поточної задачі.

Тіньові регістри відіграють роль надоперативної пам'яті. Їх призначено для підвищення швидкодії роботи МП. На рис. 4.3 тіньові регістри показано пунктиром.

Крім того, у програмній моделі на відміну від МП *i*8086 додано нові біти у регістрі прапорців і змінено використання сегментних регістрів у захищеному режимі. У регістрі прапорців біт 14 визначений як *NT* (*Nested Task Flag* – прапорець вкладеної задачі), а біти 13–12 – як *IOPL* (*Input*/*Output Privilege Level* – двобітове поле рівня привілеїв введення-виведення). Ці прапорці діють тільки в захищеному режимі. Прапорець *NT* встановлюється в одиницю при перемиканні задач за допомогою команди *CALL*. При виконанні команди *IRET* перевіряється стан прапорця *NT*, і якщо *NT* = 1, здійснюється перемикання задач, якщо ні, то виконується звичайне повернення з переривання. Поле *IOPL* вказує рівень привілею поточної задачі, за якого дозволяється виконання певних операцій.

Сегментні регістри *CS, SS, ES* і *DS* визначають початкові адреси сегментів. У реальному режимі 20-розрядна початкова адреса сегмента визначається як вміст 16-розрядного сегментного регістра, доповненого праворуч чотирма нульовими бітами. У захищеному режимі початкова 24-розрядна базова адреса сегмента знаходиться у дескрипторній таблиці в ОЗП, а вміст сегментних регістрів є селекторами, які вказують на тип таблиці та номер запису в таблиці (див. рис. 4.1).

Під час завантаження нового значення селектора дескриптори зчитуються з ОЗП і запам'ятовуються у внутрішніх програмно недоступних або тіньових регістрах процесора. Це дозволяє підвищити швидкодію процесора, оскільки значення базових адрес сегментів змінюються порівняно рідко.

Регістри *GDTR, LDTR, IDTR* задають розміщення дескрипторних таблиць у пам'яті (рис. 4.5).

На рис. 4.5 показано, що глобальна дескрипторна таблиця містить $N + 1$ дескрипторів, локальна – $K + 1$, дескрипторна таблиця переривань – $M + 1$. Початкова адреса глобальної дескрипторної таблиці визначається бітами 39–16 регістра *GDTR*, кінцева - обчислюється додаванням значень початкової адреси і межі таблиці (біти 15–0). Аналогічно початкова і кінцева адреси таблиці переривань визначаються вмістом регістра *IDTR*. Для адресації дескрипторів локальної таблиці використовують вміст регістра *LDTR*, який є селектором. Він вказує на номер дескриптора у глобальній де скрипторній таблиці. Цей дескриптор завантажується у тіньовий регістр (на рис. 4.5 показано пунктиром). Перша та остання адреси локальної дескрипторної таблиці визначаються вмістом цього регістра.

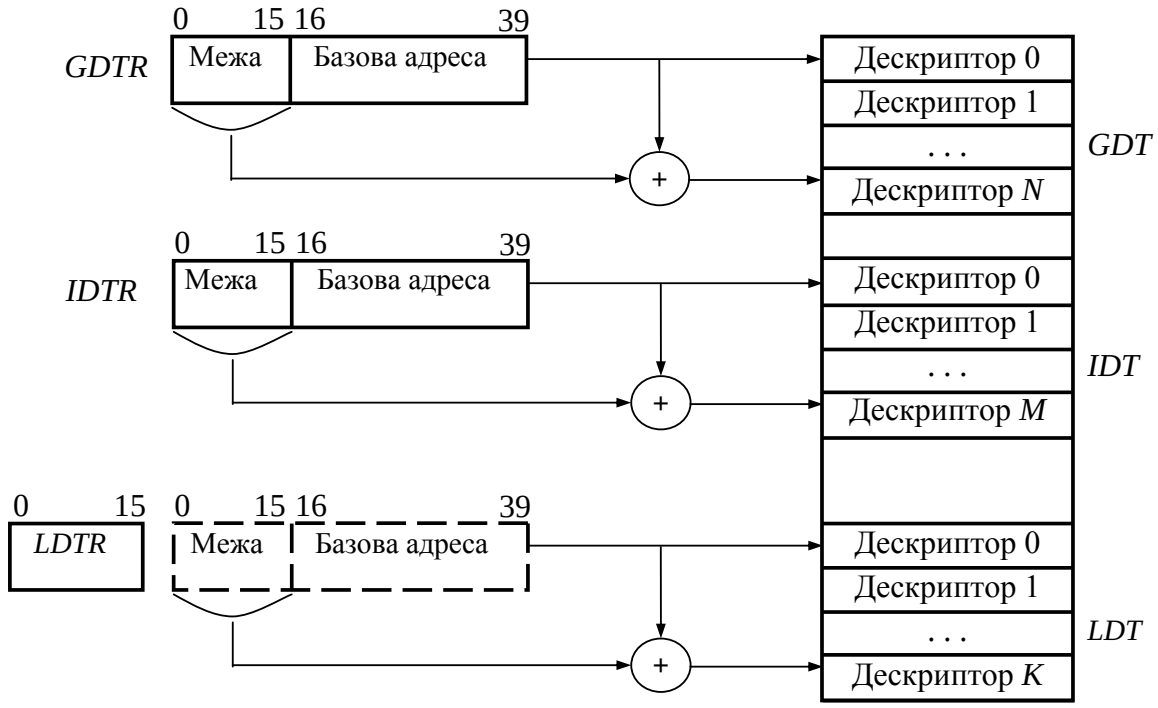


Рис. 4.5. Розміщення дескрипторних таблиць у пам’яті

Приклад 4.3. Знайти значення, яке треба завантажити у регістр *GDTR*, щоб задати у пам’яті глобальну дескрипторну таблицю з 21 дескриптора і з початковою адресою 000100*H*.

Біти 39–16 регістра *GDTR* задають початкову 24-розрядну адресу, отже, мають дорівнювати 000100*H*. Кожний запис у таблиці займає 8 байт, отже, адреса 21 дескриптора визначається як $000100H + 20 \cdot 8 = 0001A0H$. У бітах 15–0 регістра *GDTR* треба розмістити число *A*0*H*. Отже, вміст регістра *GDTR* дорівнює 000100*A*0*H*.

Зазначимо, що команди завантаження регістрів таблиць *GDTR, LDTR, IDTR, LGDT, LLDT, LIDT* є привілейованими і виконуються у програмах з вищим рівнем пріоритету.

Адресний простір портів введення-виведення. Адресний простір портів МП *i*80286 такий самий, як і для МП *i*8086, тобто становить $64 \cdot 2^{10}$ однобайтових або $32 \cdot 2^{10}$ двобайтових портів. Додаткові рядкові команди *REP INSB*/*INSW*, *REP OUTSB*/*OUTSW* (табл. 4.1) забезпечують блокову обробку зі швидкістю, що перевищує аналогічні операції в режимі ПДП. У захищеному режимі команди є привілейованими, тобто вони можуть виконуватися тільки з певним рівнем привілею, що визначається полем *IOPL* регістра прапорців. Інакше викликається виняток 13 – порушення захисту.

Система команд. До системи команд МП *i*80286 входять усі команди *i*8086 і ряд додаткових (табл. 4.1). У табл. 4.1 використано ті самі позначення, що і у табл. 3.11.

Таблиця 4.1. **Додаткові команди МП *i*80286**

Мнемокод	Опис
1	2
Команди роботи зі стеком	
PUSH immed	Переміщення безпосередніх даних *immed* у стек
PUSHA	Переміщення в стек вмісту регістрів *AX, BX, CX, DX, SI, DI, BP, SP*
POPA	Переміщення даних зі стека в регістри *AX, BX, CX, DX, SI, DI, BP, SP*
Арифметичні команди	
*IMUL reg*16, *r/m*	Множення вмісту *reg*16 на вміст *r/m* (16 біт)
*IMUL reg*16, *r/m, immed*	Множення вмісту *r/m* на 16-бітовий безпосередній операнд і переміщення результату в *reg*16
Рядкові команди	
[*REP*] *INSB* ([*REP*] *INSW*)	Введення байта *INSB* або слова *INSW* в комірку пам'яті *ES*:[*DI*] із порту з адресою *DX* з автоінкрементуванням (при *DF* = 0) або автодекрементуванням (при *DF* = 1) адреси. При використанні префікса *REP* операція повторюється *CX* разів
[*REP*] *OUTSB* ([*REP*] *OUTSW*)	Виведення байта *OUTSB* або слова *OUTSW* з комірки пам'яті *DS*: [*SI*] у порт з адресою *DX* з автоінкрементуванням (при *DF* = 0) або автодекрементуванням (при *DF* = 1) адреси. При використанні префікса *REP* операція повторюється *CX* разів
Команди переривань	
*BOUND reg*16, *lmts*	Перевірка меж масиву – якщо знакове число у *reg*16 не знаходиться в заданих межах, виконується *INT* 5. Межі задаються у двох суміжних словах пам'яті за адресою *lmts*
Команди підтримки процедур	
ENTER frmsiz, frms	Підготовка блока параметрів процедур (*frmsiz* – кількість байтів для змінних процедури, *frms* – рівень вкладення процедур)
LEAVE	Відміна чинності дії команди *ENTER* (відновлює значення вмісту регістрів *SP* і *BP*)
Команди керування станом МП	
CLTS	Скидання прапорця перемикання задач
SEG	Префікс заміни сегмента
Команди керування захистом	
LGDT src	Завантаження *GDTR* з пам'яті (6 байт)*
SGDT dest	Збереження вмісту *GDTR* у пам'яті (6 байт)*
LIDT src	Завантаження *IDTR* з пам'яті (6 байт)*
SIDT dest	Збереження вмісту *IDTR* у пам'яті (6 байт)*
LLDT src	Завантаження *LDTR* з регістра або пам'яті *reg*/*mem*16
SLDT dest	Збереження вмісту *LDTR* у *reg*/*mem*16
LMSW src	Завантаження *MSW* з регістра або пам'яті *reg*/*mem*16
SMSW dest	Збереження *MSW* у *reg*/*mem*16
LTR src	Завантаження регістра задачі з *reg*/*mem*16
STR dest	Збереження вмісту регістра задачі в *reg*/*mem*16
LAR dest, src	Завантаження старшого байта *dest* байтом прав доступу дескриптора *src*

Продовження табл. 4.1

1	2
LSL dest, src	Завантаження *dest* межею сегмента, дескриптор якого заданий *src*
ARPL reg/*mem*16, *reg*16	Вирівнювання *RPL* у селекторі до найбільшого значення двох чисел: поточного рівня привілеїв та заданого операндом
VERR seg	Верифікація читання: встановлення прапорця *ZF* у стан логічної одиниці для дозволу читання у сегменті seg
VERW seg	Верифікація запису: встановлення *ZF* у стан логічної одиниці для дозволу запису в сегмент seg

* Команди *LGDT, LIDT* завантажують у 40-розрядні регістри *GDTR, IDTR* п'ять байтів, але при виконанні команди передаються шість байтів: перші п'ять з них пересилаються у регістри, а шостий (з найбільшою адресою) ігнорується. Однак для сумісності з майбутніми розробками МП він повинен мати нульове значення. Команди *SGDT, SIDT* передають вміст регістрів *GDTR, IDTR* у пам'ять, при цьому передаються шість байтів: у п'яти знаходиться вміст регістра, а шостий – невизначений.

Приклад 4.4. Написати програму переведення МП у захищений режим адресації.

Для переведення МП у захищений режим адресації треба встановити біт *PE* – молодший біт регістра *MSW*. Однак перед установленням цього біта потрібно задати початкові значення регістрів таблиць *GDTR, LDTR, IDTR*, а також значення елементів дескрипторних таблиць в ОЗП. Зазвичай таблиці переписують із ПЗП у ОЗП. Зазначимо, що таблиці мають знаходитися в ОЗП, оскільки при виконанні програми їх вміст може модифікуватися.

Програма переведення МП має такий вигляд:

```
    SMSW   BX          ; Збереження MSW у BX
    OR     BX, 0001H   ; Установлення молодшого біта у стан
                       ; логічної одиниці
    LMSW   BX          ; Завантаження MSW
    JMP    M1          ; Скидання черги команд
M1:                    ; Початок роботи у захищеному режимі
```

Короткий перехід *JMP M*1 на наступну команду потрібний для скидання черги команд. У МП *i*80286 існує випереджаюча вибірка команд. Якщо МП виконав команду *LMSW,* яка перевела його у захищений режим, а в черзі команд залишилися команди, які мали виконуватися у реальному режимі, то після переведення процесора у захищений режим команди реального режиму будуть декодовані неправильно.

Цикли шини. Мікропроцесор *i*80286 містить шинний інтерфейс з 6-байтовою чергою команд, що забезпечує конвеєрну адресацію (*Pipelined Addressing*). Це забезпечує вибірку кодів команд або даних з пам'яті з випередженням, що дає змогу почати фазу ідентифікації або адресації нового циклу, не дочекавшись закінчення попереднього.

Розрізняють шість типів циклів шини:

ЧИТАННЯ ПАМ'ЯТІ;
ЗАПИС У ПАМ'ЯТЬ;
ЧИТАННЯ ПОРТУ;
ЗАПИС У ПОРТ;

ПІДТВЕРДЖЕННЯ ПЕРЕРИВАННЯ;
СТОП/ЗУПИН.

Відзначимо, що слово з непарною адресою передається за два цикли шини, з парною – за один. На рис. 4.6 показано цикли шини ЧИТАННЯ і ЗАПИС слова у комірку пам'яті з парною адресою. Зчитування слова з ОЗП здійснюється як мінімум за чотири періоди тактових імпульсів *CLK* або за два стани процесора (без урахування сигналу готовності *READY*). Кожний стан МП триває два такти *CLK*. Під час першого стану, позначеного через T_S, процесор виставляє на шину адреси значення адреси комірки пам'яті, з якої буде зчитуватися слово. Завдяки конвеєрній адресації адреса комірки виставляється з деяким випередженням, однак вона не зберігається на шині адреси протягом циклу. Для сумісності з шиною *ISA* сигнали шини адреси запам'ятовуються у регістрах-фіксаторах за стробом *ALE*. Керувальні сигнали читання пам'яті $\overline{MRDC}$ і адресний строб *ALE* формуються системним контролером 82288 на початку другого стану T_C. Сигнали керування й адреси обробляються схемою керування пам'яттю, у результаті чого починаючи із середини другого стану T_C на шині даних з'являється слово з ОЗП і процесор зчитує його із шини даних.

Тривалість циклу ЧИТАННЯ з парною адресою становить час двох станів.

При зчитуванні слова з непарною адресою в першому циклі передається байт по старшій половині шини даних *D*15–*D*8, у другому – передається другий байт по молодшій половині *D*7–*D*0, тобто цикл шини у цьому випадку вдвічі довший.

Цикл шини ЗАПИС У ПАМ'ЯТЬ з парною адресою також, як і при читанні, дорівнює тривалості двох станів, а з непарною – чотирьох. У першій половині циклу T_S виставляється адреса і дані, у другій – відбувається запис в ОЗП.

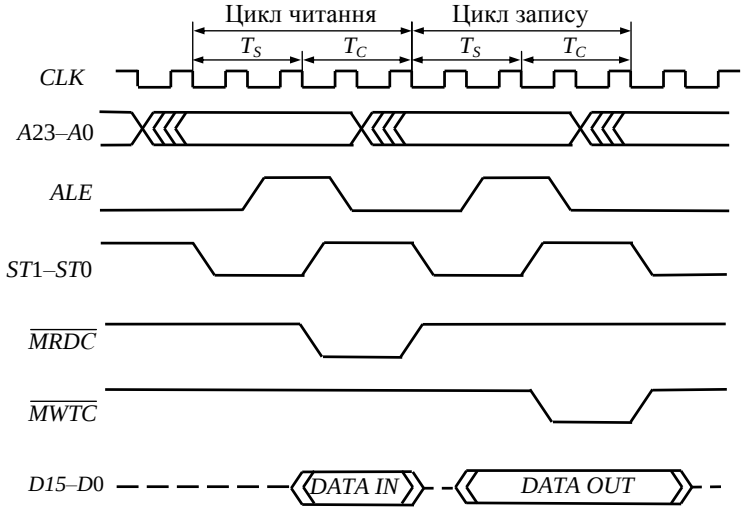


Рис. 4.6. Цикли шини ЧИТАННЯ і ЗАПИС слова у комірку пам'яті з парною адресою

Виконання циклів шини ЧИТАННЯ ПОРТУ та ЗАПИС У ПОРТ аналогічне розглянутим вище циклам ЧИТАННЯ ПАМ'ЯТІ та ЗАПИС У ПАМ'ЯТЬ, однак у цьому випадку на шину *A*15–*A*0 виставляється адреса порту і замість сигналів читання або запису пам'яті генеруються сигнали $\overline{IOR}$ – для читання портів введення-виведення або $\overline{IOW}$ – для запису в порти введення-виведення.

Цикл шини ПІДТВЕРДЖЕННЯ ПЕРЕРИВАННЯ (рис. 4.7) виконується при виникненні апаратного переривання (наявності активного рівня на виводі *INTR*). Цикл складається з двох процесорних циклів, поділених трьома тактами очікування T_W.

Кожний процесорний цикл складається з трьох станів – T_S, T_C, T_C. Це треба для того, щоб продовжити дію сигналу *INTA*, що посилається на контролер переривань 8259*A*. Перший цикл дозволяє ведучому контролеру визначити, який з ведених контролерів викликав переривання. У другому циклі МП зчитує із шини даних вектор переривання, що використовується для знаходження адреси в таблиці векторів переривань.

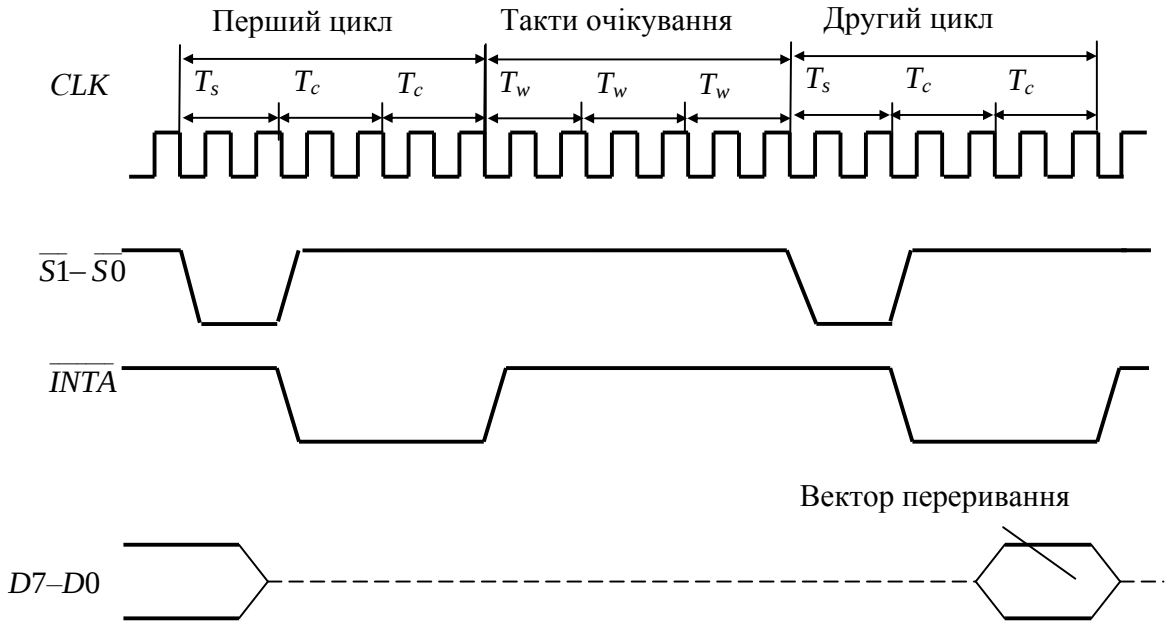


Рис. 4.7. Цикл шини ПІДТВЕРДЖЕННЯ ПЕРЕРИВАННЯ

Цикл шини СТОП/ЗУПИН (*Halt*/*Shutdown*) виникає або при виконанні команди *HLT*, або при обробці особливого випадку 8 (табл. 4.2) у захищеному режимі.

Типи переривань. Переривання і винятки в реальному і захищеному режимах наведено в табл. 4.2.

Таблиця 4.2. **Переривання і винятки МП *i*80286**

Номер переривання	Адреса повернення	Функції	
		Реальний режим	Захищений режим
1	2	3	4
0	Перший байт команди	Помилка ділення – виникає, якщо частка занадто велика або дільник дорівнює нулю	
1	Наступна команда	Переривання покрокової роботи	
2	Немає	Немасковане переривання	
3	Наступна команда	Контрольний зупин (*INT* 3)	
4	Наступна команда	Особливий випадок переповнення (*INTO*)	
5	Перший байт команди	Особливий випадок виходу з діапазону – виникає при виконанні команди *BOUND*	
6	Перший байт команди	Неприпустимий код операції – виникає, якщо трапляється недійсний код операції або команда довжиною понад 10 байт	
7	Перший байт команди	Співпроцесор недоступний	Співпроцесор недоступний або відбулося перемикання задач
8	Перший байт команди	–	Подвійна відмова – ситуація, коли МП виявляє два незалежні винятки при відпрацюванні однієї команди. Якщо під час його обслуговування станеться виняток, то МП вимкнеться (цикл *Shutdown*). У цьому стані процесор припиняє свої дії і виводиться з нього сигналами скидання або немаскованого переривання
9	Невизначена	Порушення межі сегмента співпроцесором – виникає, якщо операнд співпроцесора не поміщається у сегмент, наприклад, 16-розрядний операнд має зміщення 0*FFFFH*	
10	Перший байт команди	–	Неприпустимий сегмент стану задачі
11	Перший байт команди	–	Сегмента немає
12	Перший байт команди	Порушення межі сегмента стека	Порушення межі сегмента стека або стека немає

Продовження табл. 4.2

1	2	3	4
13	Перший байт команди	Порушення межі сегмента даних або коду – виникає, якщо операнд або код операції не вміщається в сегменті	Порушення захисту – виникає у таких випадках: при виході за межі таблиці дескрипторів; з порушенням привілеїв; при завантаженні недійсного дескриптора або типу сегмента; при спробі запису в сегмент коду або сегмент даних, призначених лише для читання; при читанні тільки з виконуваного сегмента кодів; при спробі виконати привілейовані команди, дозволені лише при певних рівнях *CPL* і *IOPL*
14	Зарезервовано		
15	Зарезервовано		
16	Наступна команда *ESC* або *WAIT*	Особливий випадок співпроцесора – виникає в МП *i*80286 при будь-якому немаскованому особливому випадку у співпроцесорі	
17	Зарезервовано		
...	...		
31	Зарезервовано		

Кожному номеру переривання відповідає елемент у таблиці *IDT* дескрипторів переривань, яка містить вектори переривань. Після скидання МП, а також під час роботи МП у реальному режимі таблиця векторів *IDT* розміститься починаючи з нульової адреси, однак командою завантаження регістра *IDTR* можна змінити місце її розміщення у межах першого мегабайта і зменшити розмір таблиці.

Приклад 4.5. Визначити, чи виникне переривання під час роботи МП у реальному режимі при виконанні команди

```
MOV AX, [0FFFFH]
```

При виконанні цієї команди в акумулятор *AX* пересилається слово з сегмента даних, причому молодший байт слова має зміщення 0*FFFFH*, а старший – 0000*H*. Цей випадок є порушенням межі сегмента і виникає переривання 13.

Контрольні запитання

1. Назвіть основні характерні ознаки режиму реальної адресації та захищеного режиму.
2. Укажіть кількість ліній шини адреси у реальному та захищеному режимах.
3. Укажіть максимально можливу кількість дескрипторних таблиць, які можуть бути задані.
4. Чим відрізняються тіньові регістри від програмно доступних?

5. Яку інформацію містять дескрипторні таблиці?
6. Укажіть призначення регістра *IDTR*.
7. Які команди треба виконати перед переходом у захищений режим роботи?
8. Дайте характеристику існуючих типів циклів шини.
9. Охарактеризуйте поняття «переривання» та «виняток».
10. Чим відрізняється обробка переривання від обробки винятку?

4.2. Архітектура 32-розрядних мікропроцесорів

Існуючі 32-розрядні МП *i*386, *i*486, *Pentium*, *Pentium Pro* і *Pentium* II мають розрядність регістрів та шини адреси, яка дорівнює 32. Шина даних для процесорів *i*386, *i*486 є 32-розрядною, а для процесорів *Pentium*, *Pentium Pro* і *Pentium* II – 64-розрядною. Вони дозволяють адресувати 4 Гбайт пам'яті, мають засоби підтримки сегментної та сторінкової адресації пам'яті. Процесори мають чотирирівневу систему захисту пам'яті та портів введення-виведення, можуть працювати у багатозадачному режимі. До режимів роботи МП *i*80286 доданий *Virtual Real Mode* – режим віртуального процесора *i*8086. Мікропроцесори допускають паралельну роботу декількох віртуальних процесорів *i*8086 під керуванням операційної системи типу *Windows*, *OS*/2, *Unix*. Процесори оперують з бітами, полями бітів, 8-, 16- та 32-бітовими операндами, рядками бітів, байтів, слів (16-розрядних даних) і подвійних слів (32-розрядних даних). В архітектуру процесорів введено засоби налагодження і тестування.

Програмна модель. Програмну модель 32-розрядного процесора зображено на рис. 4.8. Вона містить такі групи регістрів: регістри загального призначення, лічильник команд, регістр прапорців, сегментні регістри, регістри керування, системні адресні регістри, регістри налагодження, регістри тестування.

Регістри загального призначення містять усі регістри даних і регістри-вказівники МП *i*8086 та *i*80286 і стільки ж додаткових 32-розрядних регістрів. У позначенні 32-розрядних регістрів використовується початкова літера *E* (*Expanded* – розширений).

Вказівник команд EIP містить зміщення наступної виконуваної команди в сегменті кодів. При 16-розрядних адресах використовуються молодші 16 розрядів (*IP*).

Рис. 4.8. Програмна модель 32-розрядного процесора

Регістр прапорців EF розширено до 32 розрядів. Молодші 16 розрядів регістра *EF* створюють регістр прапорців *F* 16-розрядного процесора. У регістр *EF* додано нові прапорці:

- *ID* (*Identification Flag*) – прапорець дозволу команди ідентифікації CPUID *(Central Processor Unit Identification)* – для *Pentium*+[*] і деяких процесорів типу 486;
- *VIP* (*Virtual Interrupt Pending*) – віртуальний запит переривання – для *Pentium*+;
- *VIF* (*Virtual Interrupt Flag*) – віртуальна версія прапорця дозволу переривання *IF* для багатозадачних систем – для *Pentium*+;
- *AC* (*Alignment Check*) – прапорець контролю вирівнювання. Використовується тільки на рівні привілеїв 3. Якщо *AC* = 1 і *AM* = 1 (*AM* – біт у регістрі керування *CR*0), то у разі звернення до операнда, не вирівняного за відповідною межею (2, 4, 8 байт)[**], відбудеться виняток 17 (для *i*486+);
- *VM* (*Virtual* 8086 *Mode*) – у захищеному режимі вмикає режим віртуального процесора 8086. Спроба використання привілейованих команд у цьому випадку призведе до винятку 13;
- *RF* (*Resume Flag*) – прапорець поновлення. У режимі налагодження одиничне значення *RF* дозволяє здійснити рестарт команди після особливого випадку налагодження. Використовується спільно з регістрами точок зупинів.

Сегментні регістри. Окрім сегментних регістрів МП *i*8086 та *i*80286 (*DS*, *CS*, *SS*, *ES*), програмна модель містить два додаткові сегментні регістри даних: *FS* і *GS*. З кожним із шести сегментних регістрів пов'язані тіньові регістри дескрипторів. У тіньові регістри у захищеному режимі переписуються 32-розрядна базова адреса сегмента, 20-розрядна межа й атрибути (права доступу) з дескрипторних таблиць.

*Керувальні регістри CR*0–*CR*3 (*Control Register*) зберігають ознаки стану процесора, спільні для всіх задач. Молодші чотири розряди регістра *CR*0 містять біти регістра *MSW* МП *i*80286 і деякі інші біти керування. Регістр *CR*1 зарезервовано; *CR*2 зберігає 32-розрядну лінійну адресу, за якою отримано відмову сторінки пам'яті; *CR*3 у старших 20 розрядах зберігає фізичну базову адресу таблиці каталогу сторінок і біти керування кеш-пам'яттю. Регістр *CR*4 (*Pentium*+) містить біти дозволів архітектурних розширень МП.

Системні адресні регістри. Системні регістри-вказівники глобальної дескрипторной таблиці *GDTR* і таблиці переривань *IDTR* зберігають відповідно 32-розрядні базові адреси і 16-розрядні межі таблиць. Системні сегментні регістри задач *TR* і локальної дескрипторної таблиці *LDTR* є 16-розрядними селекторами. Їм відповідають тіньові регістри дескрипторів,

[*] Далі позначення *i*386+, *i*486+, *Pentium*+ означають, що наведені дані справедливі для вказаної моделі МП і всіх старших моделей.

[**] Вирівнювання операнда по межі 2, 4, 8 означає, що адреса операнда є кратною відповідно 2, 4, 8.

які містять 32-розрядну базову адресу сегмента, 20-розрядну межу і права доступу.

*Регістри налагодження DR*0–*DR*3 (*Debug Register*) зберігають 32-розрядні адреси точок зупину в режимі налагодження; *DR*4–*DR*5 зарезервовані і не використовуються; *DR*6 відображає стан контрольної точки; *DR*7 керує розміщенням у програмі контрольних точок.

Регістри тестування TR (*Test Register*) входять до групи модельно-специфічних регістрів, їх склад і кількість залежать від типу процесора: в МП *i*386 використовувалися два регістри: *TR*6 і *TR*7, у *Pentium*-1 – *TR*1–*TR*12. Ця група регістрів зберігає результати тестування МП і кеш-пам'яті.

Сегментна організація пам'яті. У 32-розрядних МП розрізняють три адресні простори пам'яті – логічний, лінійний і фізичний. Логічна адреса (або віртуальна) складається із селектора і зміщення *EA*. Лінійна адреса утворюється додаванням базової адреси сегмента до ефективної адреси. Фізична адреса пам'яті створюється після перетворення лінійної адреси блоком сторінкової переадресації.

Організація пам'яті залежить від режиму роботи МП. У реальному і віртуальному режимах *i*8086 адресація пам'яті така сама, як у МП *i*8086. У захищеному режимі використовується сегментна і сторінкова організація пам'яті. Сегментна організація використовується на прикладному рівні, а сторінкова – на системному. Формування адреси комірки пам'яті у захищеному режимі показано на рис. 4.9. Блок сегментації перетворює простір логічних адрес на простір лінійних адрес. Вихідними даними для блока сегментації є зміщення *EA* у сегменті та сегментний регістр, які задаються в команді. Вміст сегментного регістра у захищеному режимі є селектором. Він містить інформацію про тип дескрипторної таблиці (глобальної або локальної) та індекс дескриптора (див. рис. 4.1). Індекс є номером дескриптора у таблиці. Дескриптор містить базову адресу сегмента. Лінійна адреса створюється додаванням базової та ефективної адрес згідно з рис. 4.2.

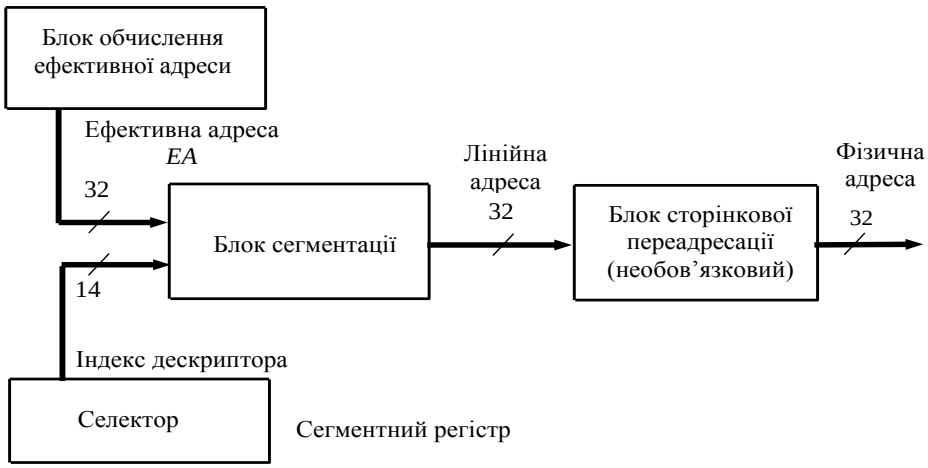


Рис. 4.9. Формування адреси комірки пам'яті в захищеному режимі

Блок сторінкової переадресації формує фізичну адресу пам’яті. При вимкненому блоці лінійна адреса збігається із фізичною. Блок обчислення ефективної адреси обчислює зміщення операнда у сегменті за одним із типів адресації, наведених у табл. 4.3. Алгоритм обчислення адреси комірки пам’яті для різних типів адресації показано на рис. 4.10.

Регістри загального призначення МП можуть виконувати функції таких регістрів: базового *Base*, індексного *Index*, масштабування множника *Scale* і зміщення *Disp*. У табл. 4.4 подано використання цих регістрів залежно від режимів адресації: 16-розрядної або 32-розрядної. Масштабовані типи адресації можливі тільки в 32-розрядному режимі адресації.

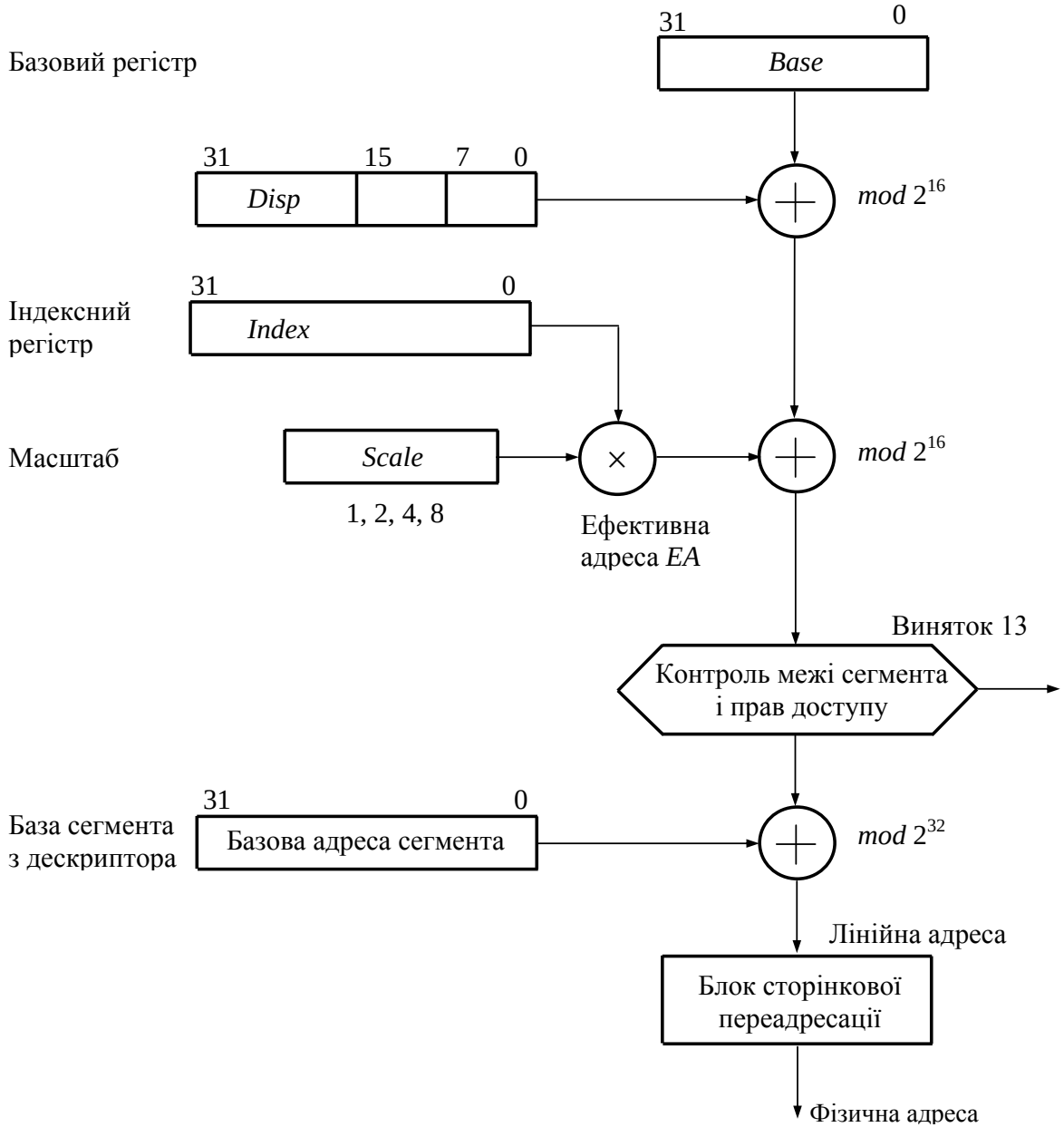


Рис. 4.10. Алгоритм обчислення адреси комірки пам’яті для різних типів адресації

***Таблиця 4.3*. Типи адресації у 32-розрядних процесорах**

Тип адресації	Обчислення *EA*
Регістрова	*EA* = вміст РЗП
Пряма	*EA* = *Disp*
Непряма регістрова	*EA* = *Base*
Базова	*EA* = *Base* + *Disp*
Індексна	*EA* = *Index* + *Disp*
Масштабована індексна	*EA* = *Scale* × *Index* + *Disp*
Базова індексна	*EA* = *Base* + *Index*
Масштабована базова індексна	*EA* = *Base* + *Index* × *Scale*
Базова індексна зі зміщенням	*EA* = *Base* + *Index* +*Disp*
Масштабована базова індексна зі зміщенням	*EA* = *Base* + *Index* × *Scale* + *Disp*

***Таблиця 4.4.* Використання РЗП при обчисленні ефективної адреси**

Компонент	16-розрядна адресація	32-розрядна адресація
Базовий регістр (*Base*)	*BX* або *BP*	Будь-який 32-розрядний РЗП
Індексний регістр (*Index*)	*SI* або *DI*	Будь-який 32-розрядний РЗП, крім *ESP*
Масштаб (*Scale*)	1	1, 2, 4 або 8
Зміщення (*Disp*)	0, 8 або 16 біт	0, 8 або 32 біт

У реальному режимі за замовчуванням використовується 16-розрядна адресація, але за допомогою префікса зміни розрядності адреси можна перемкнути на 32-розрядний режим. У захищеному режимі тип адресації залежить від біта *D* у дескрипторі кодового сегмента (при *D* = 0 використовується 16-розряднаа адресація, при *D* = 1 – 32-розрядна).

Використання сегментних регістрів при адресації пам’яті визначається типом звернення до пам’яті (табл. 4.5). Для деяких типів звернень допускається заміна сегментного регістра, що вводиться застосуванням префіксів команд *CS*:, *DS*:, *SS*:, *ES*:, *FS*:, *GS*:, наприклад:

```
ADD FS:[ESI],EAX; [FS:ESI] ← [FS:ESI] + EAX
```

***Таблиця 4.5.* Використання сегментних регістрів для адресації пам’яті**

Тип звернення до пам’яті	Сегментний регістр		Зміщення
	за замовчуванням	альтернативний	
Вибірка команд	*CS*	Немає	*IP, EIP*
Стекові операції	*SS*	Немає	*SP, ESP*
Адресація змінної	*DS*	*CS, ES, SS, FS, GS*	*EA*
Рядок-джерело	*DS*	*CS, ES, SS, FS, GS*	*SI*
Рядок-приймач	*ES*	Немає	*DI*
Використання *BP, EBP* або *ESP* як базового регістра	*SS*	*CS, ES, DS, FS, GS*	*EA*

Приклад 4.6. Знайти значення фізичної адреси операнда в команді пересилання у регістр *AL* вмісту комірки пам’яті

```
MOV AL, [BX + 4 · SI + 1000H]
```

якщо базова адреса сегмента даних дорівнює 12456789*H*, а вміст регістрів *BX* = 0120H, *SI* = 1234*H*. Блока сторінкової переадресації немає.

Ефективна адреса комірки пам'яті

$$EA = 0120H + 4 \cdot 1234H + 1000H = 59F0H.$$

Виконавши операцію додавання 32-розрядної базової адреси з ефективною адресою *EA*, отримаємо лінійну адресу, яка збігається і з фізичною адресою:

$$12456789H + 59F0H = 1245C179H.$$

Отже, фізична адреса дорівнює 1245*C*179*H*.

Формування базової адреси сегмента пояснено на рис. 4.11. Поле *TI* селектора сегмента визначає робочу дескрипторну таблицю (глобальну або локальну), де знаходиться початкова адреса сегмента.

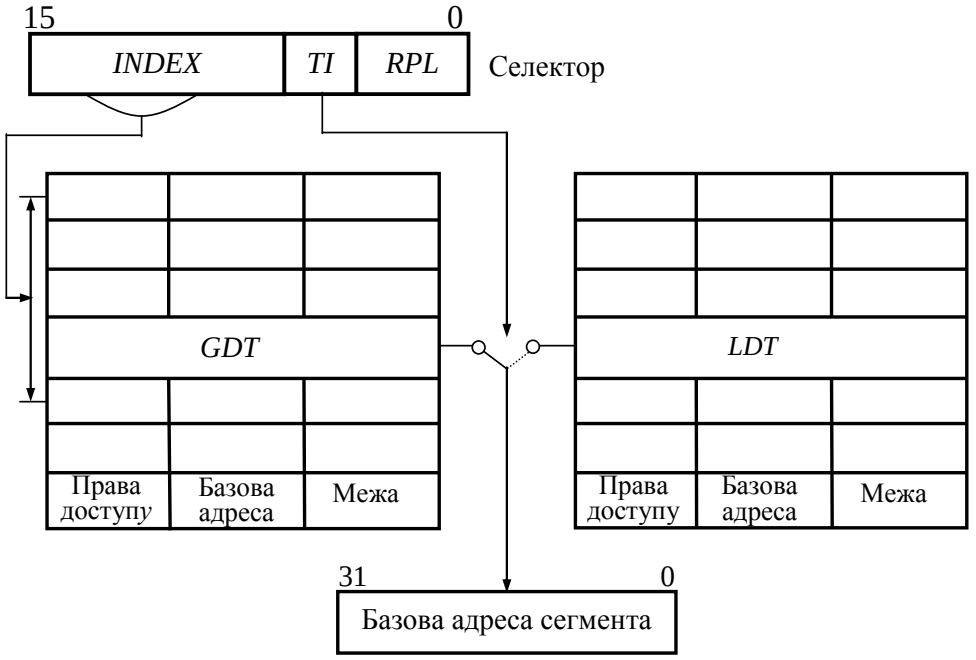


Рис. 4.11. Формування базової адреси сегмента

Поле *RPL* визначає запрошений рівень привілею сегмента. Поле *INDEX* визначає зміщення від початкової адреси таблиці. Зазначимо, що початкова адреса таблиці зберігається або в регістрі *GDTR* для глобальної таблиці, або у тіньовому регістрі. В останньому випадку регістр *LDTR* є, у свою чергу, селектором і вказує, де в глобальній дескрипторній таблиці знаходиться інформація про початкову адресу локальної таблиці. Ця інформація переписується в тіньовий регістр.

Формат дескриптора для 32-розрядних процесорів показано на рис. 4.12. Дескриптор МП *i*80286 містить нуль у бітах 63–48, а поля базової адреси і межі займають 24 і 16 біт відповідно. У 32-розрядному МП поле базової адреси займають другий, третій, четвертий і сьомий байти дескриптора. У ході виконання команди ці байти об'єднуються в одну 32-розрядну базову адресу.

Рис. 4.12. Формат дескриптора для 32-розрядних процесорів

Поле межі займає байти з номерами 0, 1 та молодші чотири біти шостого байта дескриптора. Межа задає максимальне зміщення в сегменті або останню одиницю, що адресується в сегменті. При 20-розрядній межі максимальне значення елементів, що адресуються, 2^{20}. Проте, оскільки елементом сегмента може бути не тільки байт, але і сторінка 4 кбайт, сегмент може містити від 1 байт до 4 Гбайт. Байт з номером 5 дескриптора *AR* (*Access Rights*) містить право доступу, зокрема, такі біти керування: *P* (*Present*) – біт наявності; *DPL* (*Descripter Privilege Level*) – поле рівня привілеїв сегмента; *S* (*System*) – системний біт; *Type* – поле типу сегмента; *A* (*Accessed*) – біт звернення.

Біт наявності P дорівнює одиниці, якщо сегмент знаходиться у фізичній пам'яті (ОЗП). У системі віртуальної пам'яті операційна система може передавати вміст деяких сегментів на диск, при цьому вона скидає біт *P* у нуль у дескрипторі цього сегмента. Якщо програма після цього знову звертається до сегмента, виникає особливий випадок відсутності сегмента. Операційна система шукає вільну область фізичної пам'яті (при цьому, можливо, відправляє на диск деякий інший сегмент), копіює вміст запрошеного сегмента з диска у пам'ять, записує в його дескриптор нову базову адресу, і здійснює рестарт команди, яка викликала особливий випадок 11 відсутності сегмента. Описаний процес називається *свопінгом* (*swapping*) або *довантаженням*.

Поле рівня привілеїв сегмента DPL містить 2 біти. Найвищому рівню привілею відповідає значення 0, найнижчому – значення 3.

Системний біт S має нульове значення (*S* = 0) у дескрипторах сегмента кодів, системних сегментів для зберігання локальних таблиць дескрипторів, станів задач *TSS* (*Task State Segment*) та у дескрипторах, що називаються *вентилями* (*Gate*) або *шлюзами*. В інших випадках *S* = 1.

Вентиль містить інформацію про логічну адресу входу до деякої системної програми і займає 8 байт. Формат вентилів показано на рис. 4.13.

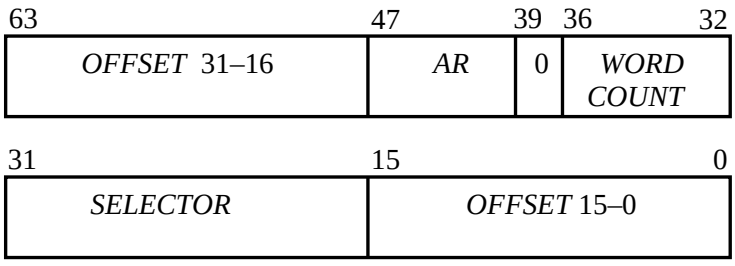


Рис. 4.13. Формат вентилів

Вентилі призначені для передачі керування і містять логічну адресу переходу у вигляді селектора *SELECTOR* і 32-розрядного зміщення *OFFSET*. Вентилі виклику використовуються для викликів процедур зі зміною рівня привілеїв, вентилі задач – для перемикання задач, вентилі переривань і вентилі пастки – для переходу до процедур обслуговування переривань, при цьому вентилі переривань забороняють переривання (скидають прапорець *IF*), а вентилі пастки – не забороняють.

Поле *WORD COUNT* (рис. 4.13) у вентилях виклику визначає кількість слів, які копіюються зі стека однієї процедури у стек іншої процедури. Для інших вентилів поле *WORD COUNT* містить нульові значення.

Поле типу сегмента Type займає три розряди та визначає тип сегмента згідно з табл. 4.6.

Таблиця 4.6. **Типи сегментів і системних об'єктів**

Type			*A*	Тип сегмента	Дозволені операції
Сегменти кодів і даних					
0	0	0	*A*	Даних	Тільки зчитування
0	0	1	*A*	Даних	Зчитування і запис
0	1	0	*A*	Стека*	Тільки зчитування
0	1	1	*A*	Стека	Зчитування і запис
1	0	0	*A*	Коду	Тільки виконання
1	0	1	*A*	Коду	Виконання і зчитування
1	1	0	*A*	Підлеглий сегмент коду	Тільки виконання
1	1	1	*A*	Підлеглий сегмент коду	Виконання і зчитування
Системні сегменти**					
0	0	0	1	Доступний сегмент *TSS* стану задачі *i*80286	
0	0	1	0	Таблиця локальних дескрипторів *LDT*	
0	0	1	1	Зайнятий сегмент *TSS* стану задачі *i*80286	
1	0	0	1	Доступний сегмент *TSS* стану задачі *i*386+	
1	0	1	0	Зарезервовано	
1	0	1	1	Зайнятий сегмент *TSS* стану задачі *i*386+	
Вентилі***					
0	1	0	0	Вентиль виклику *i*80286 (*Call Gate*)	
0	1	0	1	Вентиль задачі *i*80286 (*Task Gate*)	
0	1	1	0	Вентиль переривання *i*80286 (*Interrupt Gate*)	
0	1	1	1	Вентиль пастки *i*80286 (*Trap Gate*)	
1	1	0	0	Вентиль виклику *i*386 + (*Call Gate*)	
1	1	0	1	Вентиль задачі *i*386 + (*Task Gate*)	
1	1	1	0	Вентиль переривання *i*386 + (*Interrupt Gate*)	
1	1	1	1	Вентиль пастки *i*386 + (*Trap Gate*)	

***** На практиці такі сегменти стека не використовуються.

****** Інші стани полів *Type* та *A* не використовуються.

Біт звернення A. У сегментах коду і даних $A = 0$ означає, що до сегмента не було звернень. У системних об'єктах поле *Type* разом з бітом *A* визначає тип системного об'єкта згідно з табл. 4.6.

У старшій тетраді шостого байта дескриптора знаходяться такі біти керування:

– *G* (*Granularity*) – біт гранулярності. При $G = 0$ одиницею пам'яті в сегменті є байт, при $G = 1$ – сторінка довжиною 4 кбайт;
– *D* (*Default size*) – біт розміру. При $D = 0$ операнди у пам'яті вважаються 16-розрядними, $D = 1$ – 32-розрядними. Застосовується для сумісності з МП *i*80286;
– *U* (*User*) – біт користувача. Може бути встановлений або скинутий програмно.

Як видно з опису дескриптора, 32-розрядний МП дозволяє створення сегментів, у яких можуть виконуватися операції зчитування, читання/записування, виконання або виконання/зчитування. Для створення характерних для МП *i*8086 сегментів, у яких виконуються одночасно всі перераховані операції, використовують перекриття сегментів пам'яті, тобто початкова адреса одного сегмента є адресою іншого.

Сторінкова організація пам'яті. Цей тип організації застосовують здебільшого у системах віртуальної пам'яті, що дозволяє програмісту використовувати більший простір адрес, ніж існуюча фізична пам'ять. Ураховуючи властивість просторової локальності кодів і даних (близького розміщення потрібних комірок пам'яті), доцільно оперувати не байтами, а деякими невеликими модулями пам'яті – сторінками. При сторінковому перетворенні весь лінійний адресний простір 32-розрядного МП ємністю 4 Гбайт розбивається на 2^{20} сторінок по 4 кбайт. Фізичний простір пам'яті МПС також розбивається на сторінки, причому в фізичній пам'яті сторінок значно менше 2^{20}. Наприклад, при ємності пам'яті 4 Мбайт кількість фізичних сторінок (їх ще називають сторінковими кадрами або *page frame*) дорівнює 2^{10}. Відсутні сегменти у фізичній пам'яті зберігаються у зовнішній пам'яті (нагромаджувачі на твердому магнітному диску) і за потреби завантажуються у фізичну пам'ять, тобто відбувається процес свопінгу. Прикладні програми можуть розпоряджатися усім простором віртуальної пам'яті 4 Гбайт. Процес сторінкового перетворення адреси ілюструє рис. 4.14.

У процесі перетворення старші 20 біт 32-розрядної лінійної адреси замінюються іншим 20-розрядним значенням – номером фізичної сторінки згідно з механізмом перетворення адреси (рис. 4.14). Регістр керування *CR*3 *PDBR* (*Page Directory Base Register*) (підрозд. 4.2) містить фізичну базову адресу каталогу сторінок. *Каталог сторінок* знаходиться у фізичній пам'яті постійно і не бере участі у свопінгу. Він містить 1024 32-розрядних адреси *PDE* (*Page Directory Entry*). Кожна з них є

початковою адресою таблиць сторінок. *Таблиця сторінок PTE* (*Page Table Entry*) містить адреси сторінкових кадрів у фізичній пам'яті.

Фізична базова адреса каталогу сторінок формується із значення рядка таблиці сторінок *PTE* і 12 розрядів зміщення лінійної адреси.

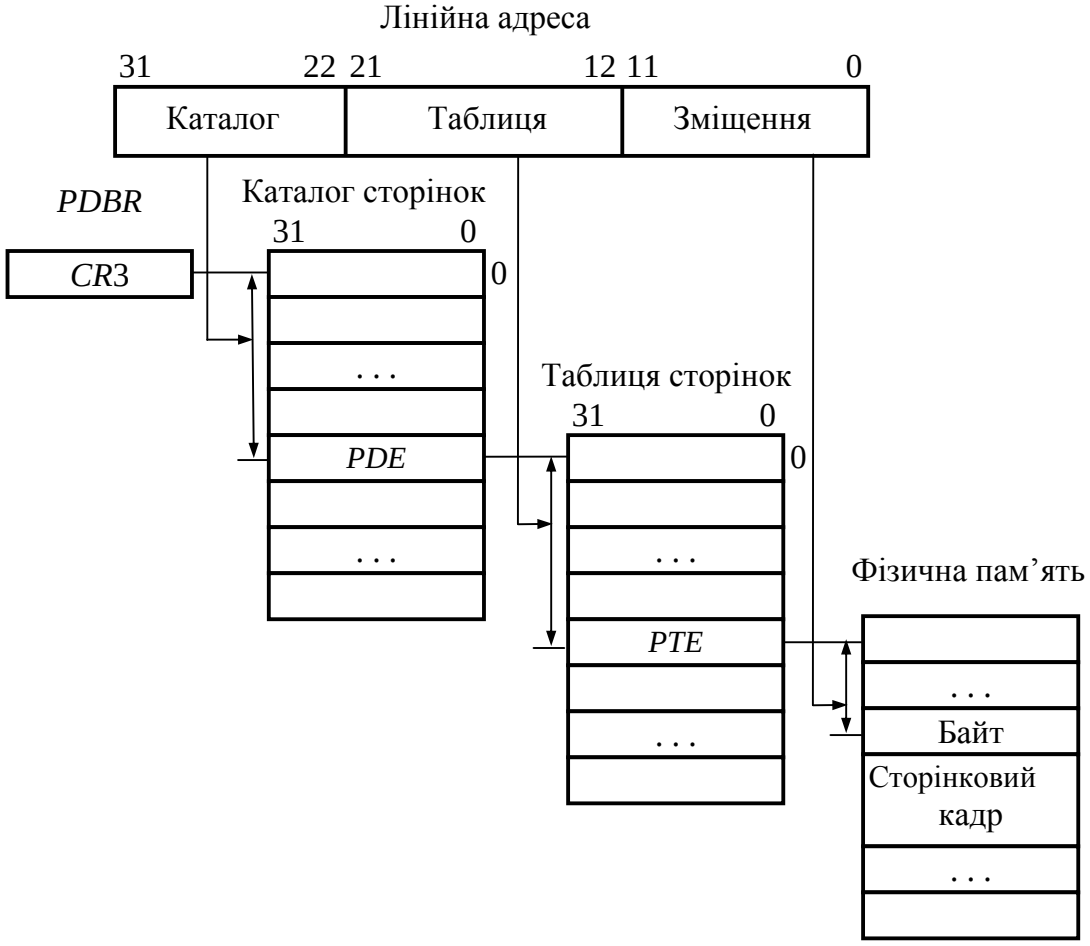


Рис. 4.14. Сторінкове перетворення адреси

Захист за привілеями. Систему привілеїв призначено для запобігання недозволеним взаємодіям користувачів, несанкціонованому доступу до даних, пошкодженню програм і даних. Частково ці задачі вирішуються організацією захищеного режиму пам'яті, частково – захистом за привілеями; 32-розрядні процесори підтримують чотири рівні привілеїв 0–3, причому рівень 0 є найбільш привілейованим. Рівень 0 зазвичай присвоюється ядру операційної системи, рівень 1 – системним сервісам, рівень 2 – розширенням операційної системи і рівень 3 – прикладним програмам користувача. Під час виконання програми контролюється, чи може програма:

- виконувати привілейовані команди;
- звертатися до даних інших програм;
- передавати керування іншій програмі командами передачі керування типу *FAR*.

До привілейованих команд належать команди, що змінюють сегментацію, впливають на механізм захисту, модифікують прапорець дозволу переривань *IF*. При спробі виконати ці команди на рівнях привілеїв 1, 2 або 3 генерується виняток 13.

Для контролю звернення програми до даних інших програм використовуються поля *CPL* (*Current Privilege Level* або *Code Privilege Level* – поточний рівень привілеїв; задається полем *RPL* селектора *CS*) і дескриптора даних *DPL*. Доступ до даних дозволяється при $CPL \le DPL$.

Передача керування програмам різних рівнів привілеїв здійснюється за допомогою використання:

– підлеглих сегментів коду (див. табл. 4.6);
– дескрипторів вентилів викликів (шлюзів).

У підлеглих сегментах виконання команд можливе, якщо поточний рівень привілеїв (*CPL*) не нижчий від рівня привілеїв дескриптора (*DPL*) підлеглого сегмента, у непідлеглих – керування сегмента передається при $CPL = DPL$. Зазвичай у підлеглих сегментах кодів розміщують бібліотеки, до яких можуть звертатися програми різних рівнів привілеїв. Використання підлеглих кодових сегментів не змінює поточний рівень привілеїв. Єдиним засобом зміни рівня привілею є використання вентилів викликів. Вентилі ідентифікують дозволені точки входу у кодові сегменти з більшим рівнем привілею. У дескрипторі вентиля задається повна адреса точки входу (селектор: зміщення) тієї процедури, якій передається керування.

Перемикання задач. У багатозадачних системах і системах з великою кількістю користувачів МП виконує деяку частину команд однієї задачі (програм), після цього перемикається на виконання іншої задачі і так продовжується доти, доки знов не повертається до першої задачі. Для підтримки багатозадачного режиму в МП є такі засоби:

– сегмент стану задачі *TSS*;
– дескриптор сегмента стану задачі;
– регістр задачі *TR*;
– вентиль задачі.

Дескриптор сегмента стану задачі вказує на сегмент, що містить повний стан задачі, а вентиль задачі містить селектор, що вказує на дескриптор *TSS*. Регістр *TR* є селектором сегмента *TSS* поточної задачі. Кожна задача має свій сегмент стану. У сегменті *TSS* міститься інформація про стан процесора на час перемикання задач – вміст майже всіх регістрів МП, включаючи регістр прапорців, роздільні вказівники стеків для рівнів привілеїв 0, 1, 2 і посилання на селектор *TSS* задачі, що її викликала.

Перемикання задач здійснюється або за командами міжсегментних переходів *JMP FAR* чи викликів підпрограм *CALL FAR*, або за апаратними чи програмними перериваннями і винятками. У першому випадку про-

грама має посилатися на сегмент стану задачі *TSS* або на дескриптори вентиля задачі в *GDT*(*LDT*), у другому – відповідний перериванню дескриптор у таблиці переривань *IDT* має бути дескриптором вентиля задачі.

Під час передачі керування викликаній задачі за командою *IRET* перевіряється прапорець вкладеної задачі *NT* (*Nested Task*). Якщо *NT* = 0, команда *IRET* працює у звичайному режимі, залишаючись у поточній задачі. Якщо *NT* = 1, команда *IRET* виконує перемикання на попередню задачу.

Контрольні запитання

1. Укажіть призначення регістрів, які входять до програмної моделі 32-розрядного процесора.
2. Укажіть призначення прапорців, які входять до програмної моделі 32-розрядного процесора.
3. Укажіть призначення регістрів керування та регістрів тестування.
4. Укажіть призначення та існуючі типи дескрипторних таблиць.
5. Як значення системного біта визначає тип дескриптора?
6. Дайте визначення процесу свопінгу і поясніть, як він відбувається.
7. Поясніть принцип сторінкової організації пам’яті.
8. Поясніть необхідність та принцип функціонування механізму захисту за привілеями.
9. За яких умов дозволяється зчитувати (записувати) дані певного сегмента?
10. Яка інформація міститься у вентилях викликів?
11. Поясніть особливості багатозадачного режиму роботи.
12. Укажіть призначення регістра *TR*.

4.3. Особливості архітектури мікропроцесорів *i*386 та *i*486

Мікропроцесор *i*386. Перший 32-розрядний процесор *i*386 з’явився 1985 року. Він виконаний за 1,5 мкм-технологією і містить 275 тис. транзисторів. Розрядність регістрів, шини даних та адреси дорівнює чотирьом. Ємність пам’яті, що прямо адресується, 4 Гбайт. Процесор може працювати у трьох режимах – реальному, захищеному та режимі віртуального процесора *i*8086 – *V*86. Припускає паралельну роботу декількох віртуальних процесорів *i*8086 під керуванням операційної системи типу *Windows*, *OS*/2, *Unix*. Перемикання режимів відбувається швидше, ніж у МП *i*80286. Має механізми сторінкової адресації, які істотно підвищують ефективність роботи з пам’яттю понад 1 Мбайт навіть у межах *DOS*. Черга команд – 16 байт. Мікропроцесор *i*80386 має модифікації: *DX* – регістри, шини даних та адреси є 32-розрядними; *SX* – із зовнішньою 16-розрядною шиною даних та 24-розрядною шиною адреси;

SL – відрізняється від модифікації *SX* зниженим енергоспоживанням та вбудованим контролером зовнішньої кеш-пам'яті на 16–64 кбайт. До комплекту 386*SL* входить мікросхема 82360*SL* – набір периферійних контролерів для ноутбуків. Основні характеристики процесорів 80*x*86 зведено у табл. 4.7.

Таблиця 4.7*. Основні характеристики процесорів 80*x*86 фірм *Intel* та *IBM

Процесор	Розрядність			Ємність кеш-пам'яті, кбайт	Наявність співпроцесора	Частота процесора, МГц
	регістрів	шини даних	шини адреси			
8088	16	8	20	–	–	4,77–8
8086	16	16	20	–	–	5
286	16	16	24	–	–	6–25
386*SX*	32	16	24	–	–	16–23
386*SL*	32	16	24	–	–	25
386*SLC*	32	16	24	8	–	25–40
486*SLC*	32	16	24	16	–	25–40
486*SLC*2	32	16	24	16	–	40–66
486*SLC*3	32	16	24	16	–	75
386*DX*	32	32	32	–	–	25–40
486*DLC*	32	32	32	16	–	25–40
486*SX*	32	32	32	8	–	16–33
486*BL*2	32	32	32	16	–	40–66
486*BL*3	32	32	32	16	–	75–100
486*DX*	32	32	32	8	+	25–50
487*SX*	32	32	32	8	+	25–50
486*SL*	32	32	32	8	+	25–50
486*DX*2	32	32	32	8/16	+	40–80
486*DX*4	32	32	32	16	+	75–120

Мікропроцесор *i*486. Процесор з'явився 1989 року. Характеризується значно вищою швидкодією порівняно з *i*8086. Його виконано за 1 мкм-технологією; містить 1,2 млн транзисторів. Основні особливості МП *i*486: наявність внутрішньої кеш-пам'яті, вбудованого математичного співпроцесора, сумісного за командами арифметичного співпроцесора *i*387. У МП *i*486 збільшено чергу команд, прискорено виконання операцій як у цілочисловому АЛП, так і в блоці математичного співпроцесора за рахунок архітектури, уведено множення тактової частоти системної плати. У модифікаціях 486*DX*2 внутрішня частота дорівнює подвоєній зовнішній, а в процесорах 486*DX*4 кратність може бути 2; 2,5; 3. Математичного співпроцесора у модифікаціях *SX* і в деяких модифікаціях *SL* немає. Процесори *DX*4 залежно від модифікації можуть працювати при живленні 5 В та 3,3 В і мають режим *SMM* (*System Management Mode*), що дозволяє керувати енергоспоживанням.

Крім розглянутих вище процесорів фірми *Intel* (див. табл. 4.7), існують аналогічні за технічними характеристиками процесори, які виготовляють інші фірми – *IBM, AMD, Cyrix, Texas Instruments*. Так, процесор 386*SLC* являє собою поліпшений варіант 386*SX*. Він має внутрішню кеш-пам'ять і характеризується прискореним виконанням операцій. Мікропроцесор 486*SLC* – це варіант процесора *i*486*SX*, а процесори *SLC*2/*SLC*3 здійснюють подвоєння (потроєння) зовнішньої частоти.

Процесори *SCL*2 і *SL* характеризуються значенням напруги живлення 3,3 В і зниженим енергоспоживанням. Більшість процесорів фірми *AMD* мають знижене споживання (літера *L* у позначенні ВІС) і напругу живлення 3,3 В (на знижену напругу живлення вказує літера *V* у позначенні МП), наприклад, процесор *AMD* 5*X*-133*ADV* являє собою варіант *i*486 зі збільшеною частотою у чотири рази і зниженою напругою живлення.

Внутрішня кеш-пам'ять. Починаючи з МП *i*486, застосовується внутрішнє розділене кешування команд і даних (докладно принципи організації кеш-пам'яті розглянуто в підрозд. 5.5). Якщо адресовану область відображено у кеш-пам'яті (випадок потрапляння – *cache hit*), то запит на читання обслуговується тільки кеш-пам'яттю без звернення до основної пам'яті. Під час запиту на запис спочатку модифікується інформація в кеш-пам'яті, а після цього, залежно від типу кеш-пам'яті, й основна пам'ять.

У перших процесорах *i*486 використовувався режим наскрізного запису *Write Through*, коли інформація одночасно записується як у буфер, так і в ОЗП. Більш пізні модифікації використовують режим зворотного запису *Write Back*, який полягає в тому, що копія блоку записується в ОЗП лише в тому разі, якщо його вміст змінювався.

Заповнення рядка кеш-пам'яті процесор намагається виконати найбільш швидким засобом – пакетним циклом з 32-бітовими передачами. Вибір рядка для заміщення новими даними здійснюється на підставі аналізу біта *LRU* (*Least Recently Used*), який виконується тільки для кеш-промахів читання. За наявності кеш-промахів запису заповнення рядків не виконується.

Пакетний режим передачі даних. Пакетний режим передачі даних (*Burst Mode*) призначено для швидких операцій з рядками кеш-пам'яті. При цьому вміст чотирьох 32-розрядних комірок основної пам'яті пересилається в один рядок кеш-пам'яті або навпаки – вміст одного рядка кеш-пам'яті пересилається у чотири 32-розрядні комірки основної пам'яті. Оскільки рядок кеш-пам'яті процесора *i*486 має довжину 16 байт (128 біт), то для його пересилання потрібно чотири 32-розрядні цикли шини, протягом кожного з яких відбувається пересилання

вмісту однієї 32-розрядної комірки з основної пам’яті або 32 біт з кеш-пам’яті.

Розглянемо випадок пересилання вмісту чотирьох 32-розрядних комірок основної пам’яті в один рядок кеш-пам’яті (рис. 4.15).

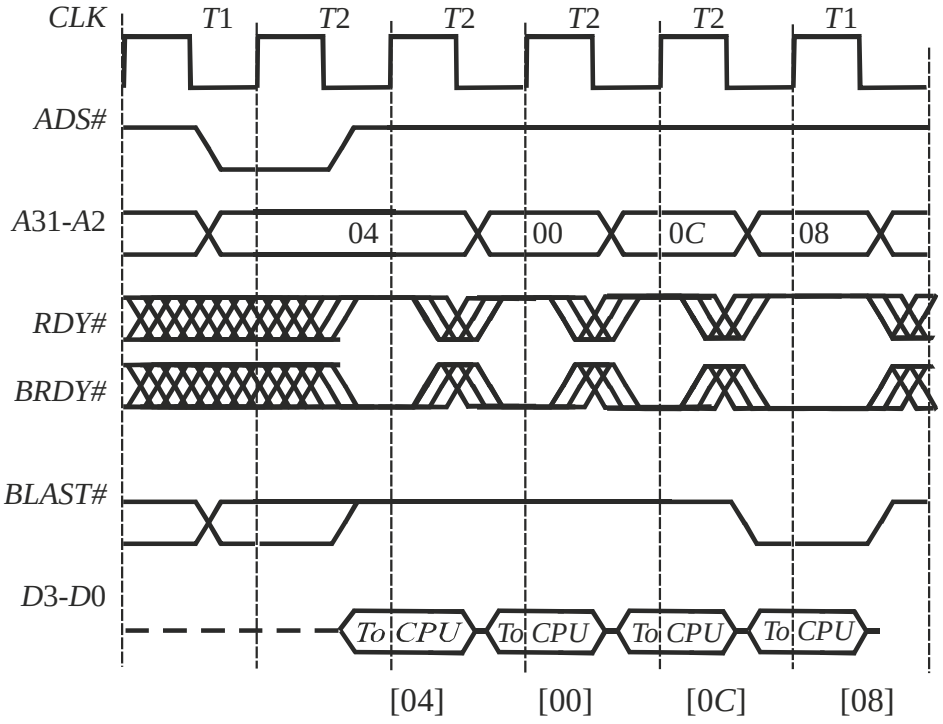


Рис. 4.15. Діаграма пересилання вмісту чотирьох 32-розрядних комірок пам’яті

Під час зчитування вмісту 32-розрядної комірки розряди *A*0 і *A*1 не беруть участі у формуванні адрес, оскільки вони означають положення кожного із чотирьох байтів у 32-розрядній комірці. Тому на рис. 4.15 показано значення розрядів *A*31–*A*2.

У першому такті *T*1 встановлюється адреса *A*31–*A*2, сигнали ідентифікації типу циклу і формується строб *ADS#*. Цей такт виконується процесором так само, як і за звичайної передачі даних.

У другий такт *T*2 передається перше 32-розрядне слово. При цьому сигнал *BLAST#* має значення *H*-рівня. Про завершення пакетного циклу процесор повідомляє зовнішньому пристрою сигналом *BLAST#*, що набуває значення *L*-рівня в останньому такті передачі пакета. Якщо зовнішній пристрій підтримує пакетний режим, він генерує сигнал готовності до пакетної передачі *BRDY#* (замість сигналу *RDY#*). У цьому разі процесор продовжить цикл як пакетний, не вводячи такти *T*1 адресації – ідентифікації (з сигналом *ADS#*), а відразу переходячи до передавання наступного 32-розрядного слова. Формуванням сигналу *RDY#* замість *BRDY#* зовнішній пристрій може в будь-

який момент перервати пакетне передавання, і процесор продовжить його звичайними циклами. Для передачі 16 байт у пакетному режимі потрібно п’ять тактів шини (без тактів чекання) замість восьми тактів за звичайного режиму передачі.

Пояснимо чергування адрес 32-розрядних слів у пакетній посилці. Подамо адресу слова у вигляді таблиці (табл. 4.8).

Таблиця 4.8. **Формат адреси слова**

*A*31–*A*4	*A*3	*A*2	*A*1	*A*0
Довільні значення	Номер 32-розрядного слова у пакетній посилці		Номер байта у 32-розрядному слові	

Розряди *A*1 та *A*0 у цій адресі дорівнюють нулю. Розряди *A*3, *A*2 визначають положення кожної із чотирьох 32-розрядних комірок у рядку кеш-пам’яті завдовжки 128 біт. Розряди *A*31–*A*4 визначають адресу 32-розрядного слова в основній пам’яті, яке пересилається у кеш-пам’ять.

Рядок кеш-пам’яті вирівняно по межі 128-розрядних даних, його адреса має нульові значення розрядів *A*3–*A*0. У пакетній передачі завжди зчитуються дані, що відповідають одному рядку кеш-пам’яті, адреса якого кратна 128. Адреса першого 32-розрядного слова пакетної посилки може бути не кратною 128. За нульових значень розрядів *A*31–*A*4 та *A*1, *A*0 адреса першого слова визначається розрядами *A*3, *A*2 (табл. 4.9).

Таблиця 4.9. **Адреси слів у пакетній посилиці**

*A*31–*A*4	*A*3	*A*2	*A*1	*A*0	Адреса першого слова у пакетній посилці
0...0	0	0	0	0	00
0...0	0	1	0	0	04
0...0	1	0	0	0	08
0...0	1	1	0	0	0*C*

Чергування адрес 32-розрядних слів у пакетній посилці залежно від адреси першого слова наведено в табл. 4.10.

Таблиця 4.10. **Послідовність зміни адрес у пакетному циклі**

Адреса			
першого слова	другого слова	третього слова	четвертого слова
00	04	08	0*C*
04	00	0*C*	08
08	0*C*	00	04
0*C*	08	04	00

Приклад, коли перша адреса блоку не збігається з межею рядка, а дорівнює 04*H* показано на рис. 4.15. Порядок чергування адрес при

цьому відповідає другому рядку табл. 4.10, тобто дорівнює 04*H*, 00*H*, 0*CH*, 08*H*. Наведений порядок чергування адрес у пакетній посилці характерний для всіх процесорів *Intel* та сумісних з ними, починаючи з *i*486.

Буфери відкладеного запису. Буфери відкладеного запису призначені для запам'ятовування даних у буфері, коли зовнішня шина зайнята. Процесор *i*486 має чотири буфери відкладеного запису. Інформація у буфер записується за один такт. Після закінчення поточного циклу шини інформація з буферів передається у пам'ять або ПВВ. Зовнішні операції запису з буферів виконуються у тому самому порядку, у якому надійшли запити на запис. Але, якщо при незвільнених буферах усі запити на запис у пам'ять пов'язані з кеш-потраплянням, а запит на читання пов'язаний з кеш-промахом, то операція читання може виконатися раніше, ніж операції запису. Але не можна змінювати порядок виконання команд у МП більше, ніж один раз, оскільки нові прочитані дані можуть замінити модифікований рядок кеш-пам'яті, з якого поновлена інформація очікує у буфері черги на запис в основну пам'ять. У такому разі друга спроба зміни послідовності команд може порушити цілісність даних. Для операцій введення-виведення зміна послідовності команд не допускається, бо це може порушити протокол обміну.

***OverDrive*-процесори.** Вони призначені для модернізації МПС або ПЕОМ. Модернізацію здійснюють заміною початкової моделі МП на нові моделі, так звані *OverDrive*-процесори, які мають вищі техніко-економічні показники. Такими моделями для *i*486 є *Intel DX*2 *OverDrive*, *Intel DX*4 *OverDrive*, *Pentium OverDrive* 63 та 83 МГц.

Для модернізації МПС на системній платі, крім уже встановленого процесора, у додатковий сокет (роз'єднувач), позначений як *OverDrive*, встановлюється *OverDrive*-процесор у корпусі *PGA*-169. *OverDrive*-процесор спеціальним вихідним сигналом від'єднує основний процесор, що залишається на платі.

Режим системного керування 32-розрядних мікропроцесорів. Деякі модифікації процесорів *i*386 і *i*486, крім перерахованих режимів (реального, захищеного і віртуального *V*86), мають режим системного керування *SMM* (*System Management Mode*), який призначено для керування енергоспоживанням або виконання програм, повністю ізольованих як від прикладного програмного забезпечення, так і від операційної системи. Перемикання в режим *SMM* здійснюють або апаратно – подачею нульового потенціалу на контакт $\overline{SMI}$ мікросхеми процесора, або, у деяких моделях, програмно – за прийняттям повідомлення по шині *APIC*. Переходячи у режим *SMM*, МП виставляє сигнал

підтвердження на контакті $\overline{SMIACT}$. Після цього процесор зберігає свій стан – вміст майже всіх регістрів – у спеціальній області пам'яті *SMRAM*. Якщо режим *SMM* використовують для вимкнення живлення процесора з можливістю швидкого ввімкнення, пам'ять *SMRAM* має бути енергонезалежною. У цій області пам'яті знаходиться підпрограма обробки переривання *SMI*. Доступ до пам'яті дозволено тільки за наявності сигналу $\overline{SMIACT}$. Повернення з режиму *SMM* здійснюють як програмно, так і за перериванням.

Контрольні запитання

1. Укажіть призначення пакетного режиму.
2. Поясніть процес пакетного передавання даних.
3. Укажіть призначення буферів відкладеного запису.
4. У яких випадках може порушуватися порядок обслуговування запитів на запис і читання у буферах відкладеного запису?
5. Укажіть призначення *OverDrive*-процесорів.

4.4. Особливості архітектури мікропроцесорів *Pentium*

Мікропроцесор *Pentium* являє собою високопродуктивний 32-розрядний процесор з внутрішньою 64-розрядною шиною даних. Процесор є продовженням розробок процесорів *i*80*x*86 і програмно-сумісний з ними, але має ряд особливостей. У МП *Pentium* уперше застосовано 0,8 мкм *BiCMOS*-технологію. *BiCMOS*-технологія поєднує переваги двох технологій – швидкодію біполярної і мале енергоспоживання *CMOS*. Використання субмікронної технології дозволило збільшити кількість транзисторів до 3,1 млн. Для порівняння: процесор *i*8086 містить 29 тис. транзисторів, а найближчий до *Pentium* процесор *i*486 – 1,2 млн транзисторів. Збільшення кількості транзисторів (більше ніж удвічі) дозволило розмістити в одній мікросхемі компоненти, що раніше розташовувалися в інших мікросхемах. Це зменшило час доступу і збільшило продуктивність процесора. Висока тактова частота, суперскалярна архітектура, розділена кеш-пам'ять для програм і даних та інші вдосконалення дозволили досягти більшої продуктивності та сумісності з програмним забезпеченням, розробленим для мікропроцесорів фірми *Intel*. Мікропроцесор *Pentium* дозволяє використовувати такі операційні системи, як *UNIX*, *Windows NT*, *OS*/2, *Solaris* і *NEXTstep*. Розглянемо особливості архітектури.

Структурна схема та характеристики. Узагальнену структурну схему МП *Pentium* зображено на рис. 4.16

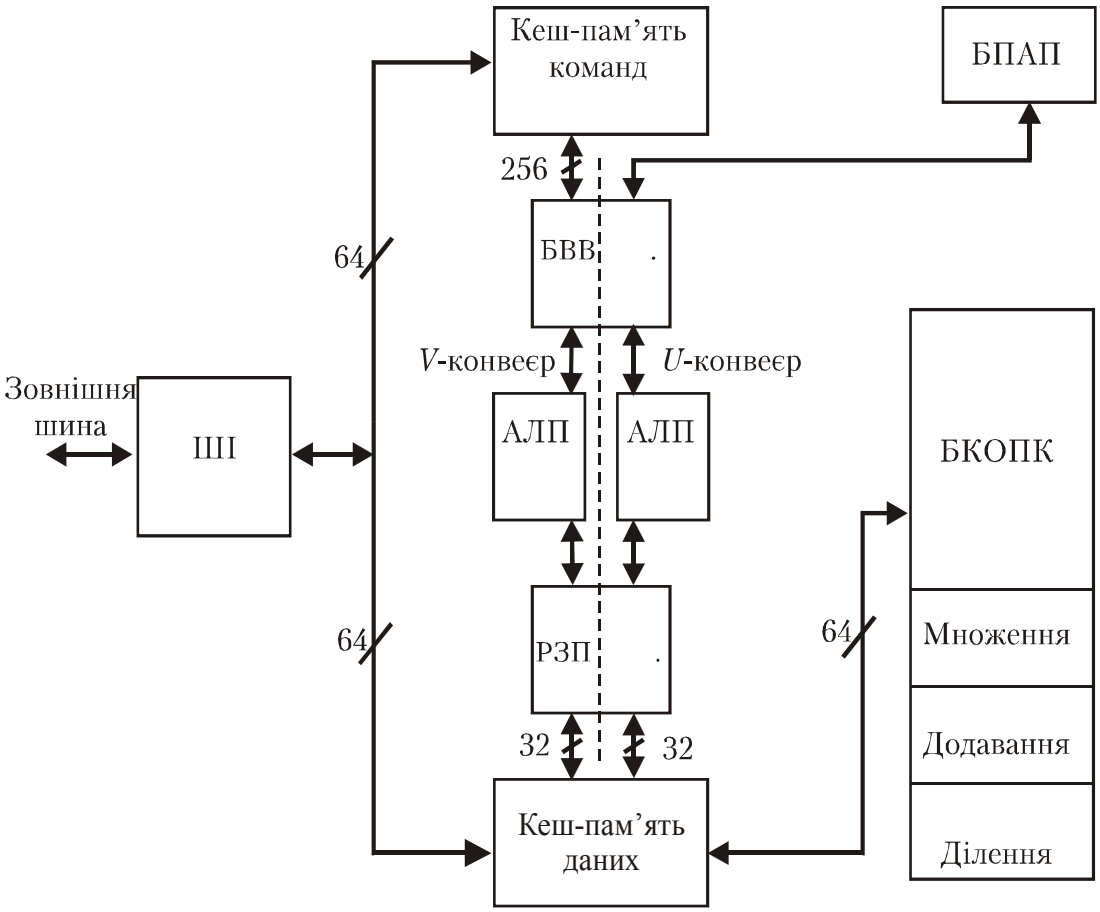


Рис. 4.16. Узагальнена структурна схема мікропроцесора *Pentium*

Подана на рис. 4.16. схема містить:

– ШІ – 64-розрядний шинний інтерфейс, призначений для спряження внутрішньої шини процесора із зовнішньою шиною;
– два 32-розрядні цілочислові АЛП;
– кеш-пам'ять команд;
– кеш-пам'ять даних;
– РЗП;
– буфери вибірки з випередженням (ББВ);
– блок передбачення адреси переходу (БПАП);
– блок конвеєрних обчислень з плавучою комою (БКОПК).

Розширена 64-розрядна шина даних. Завдяки цьому МП *Pentium* підтримує декілька типів циклів шини, включаючи і пакетний режим, при якому частина даних з 256 біт передається у кеш-пам'ять даних за один цикл. Це істотно підвищує швидкість передачі порівняно з процесором *i*486 *DX*. Наприклад, МП *Pentium* із частотою шини 66 МГц має швидкість передачі 528 Мбайт/с, МП *i*486 *DX* із частотою шини 50 МГц – 160 Мбайт/с. Розширена шина даних забезпечує конвеєризацію циклів шини, що збільшує пропускну здатність шини і дозволяє другому циклу починатися раніше, ніж завершився перший.

Суперскалярна архітектура. Термін «суперскалярна» означає мікропроцесорну архітектуру, що містить більше ніж один обчислювальний блок. Процесор *Pentium* має два конвеєри, які можуть виконувати дві команди одночасно – *U*-конвеєр з повним набором команд і *V*-конвеєр з обмеженим набором команд. На рис. 4.16 конвеєри спрощено подані двома цілочисловими АЛП, РЗП і БВВ. Як і у випадку єдиного конвеєра процесора *i*486, подвійний конвеєр процесора *Pentium* виконує цілочислові команди у п'ять етапів (рис. 4.17):

1) вибірка з випередженням команди з пам'яті (передвибірка) *PF* (*PreFetch*);
2) декодування команди *D*1 (стадія 1);
3) декодування команди *D*2 (стадія 2);
4) виконання команди *EX*;
5) запам'ятовування результату в буфері відкладеного запису *WB* (див. підрозд. 4.3).

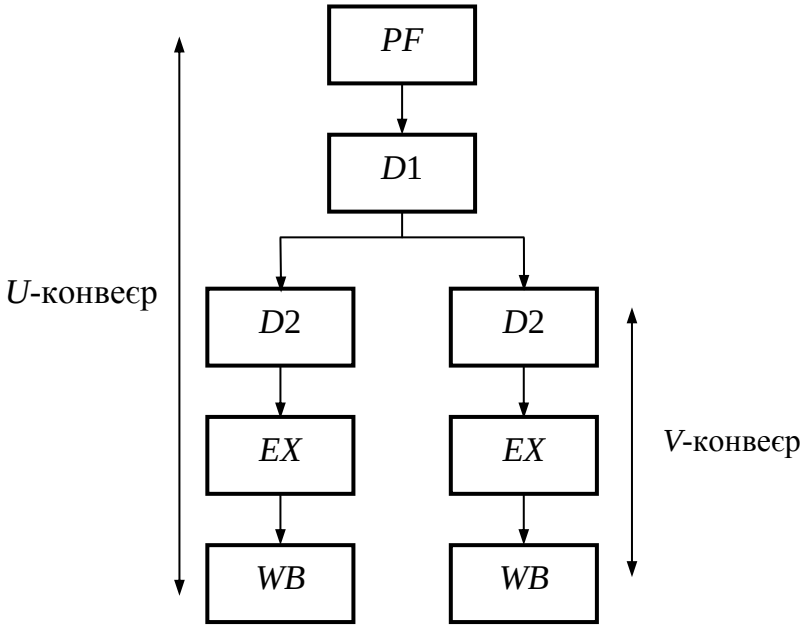


Рис. 4.17. Етапи виконання цілочислової команди у процесорі *Pentium*

Перший етап виконується блоком БВВ, який має чотири 32-розрядні буфери. Дві незалежні пари буферів вибірки працюють сумісно з БПАП, який передбачає, буде перехід чи ні. Якщо перехід не передбачається, продовжується вибірка, якщо передбачається, то дозволяється робота іншого буфера, і він починає передвибірку з точки переходу. Якщо передбачений перехід не здійснився, конвеєри команд очищаються, і передвибірка починається знову. На другій стадії декодування команди формуються адреси операндів пам'яті.

Кожний конвеєр має свій 64-розрядний буфер відкладеного запису, який може заповнюватися за один такт, наприклад, при одночасних кеш-промахах запису на обох конвеєрах. Ніякі запити на читання не порушують порядку запитів на запис, які вже знаходяться у буфері. *Pentium* підтримує строгий порядок запису.

Високопродуктивний математичний співпроцесор БКОПК містить восьмитактовий конвеєр і апаратні засоби реалізації арифметичних операцій – множення, додавання, ділення. Більша частина команд операцій з плавучою комою можуть виконуватися в одному цілочисловому конвеєрі, після чого вони надходять у конвеєр обчислень з плавучою комою. Продуктивність вбудованого арифметичного співпроцесора *Pentium* переважає продуктивність математичного співпроцесора *FPU*-486 (*Floating-Point Unit*) у 2–10 разів.

Використання подвійного конвеєра дозволяє декільком командам знаходитися в різних стадіях виконання і додатково збільшувати продуктивність МП повним заповненням конвеєрів командами. У процесорі *Pentium* використовується апаратне виконання команд, що також підвищує продуктивність процесора.

Розділені кеш-пам'яті команд і даних. Мікропроцесор *Pentium* має розділені кеш-пам'яті команд і даних. Це дозволяє уникнути конфліктів між процесом вибірки для однієї команди і доступом до даних для іншої, які можуть виникати, наприклад, у процесорі *i*486. При реалізації розділеного кешування для команд і даних обидві команди можуть виконуватися одночасно. Ємність кеш-пам'яті команд і кеш-пам'яті даних у процесорі *Pentium* однакова і становить 8 кбайт. Кеш-пам'ять команд і даних виконано за схемою двоканальної асоціативної кеш-пам'яті (див. підрозд. 5.8). Кеш-пам'ять даних має два інтерфейси (по одному для кожного конвеєра), що дозволяє забезпечувати даними дві окремі команди протягом одного машинного циклу.

Кеш-пам'ять даних працює з відкладеним (до звільнення зовнішньої шини) записом і налагоджується у режим наскрізного або зворотного запису. В останньому випадку дані зчитуються з кеш-пам'яті, а після цього записуються в основну пам'ять. Такий спосіб кешування дозволяє збільшувати продуктивність порівняно з простим кешуванням з безпосереднім записом, при якому процесор записує дані одночасно в кеш-пам'ять і основну пам'ять. Кеш-пам'ять даних підтримує протокол *MESI*. Цей протокол забезпечує роботу з урахуванням можливості звернення інших процесорів до кеш-пам'яті даних. Назва протоколу *MESI* складається з назв станів рядка кеш-пам'яті: *M* (*Modified*), *E* (*Exclusive*), *S* (*Shared*) *I* (*Invalid*). Стани рядка кеш-пам'яті визначають таким чином.

1. *M*-стан – рядок, наявний тільки у кеш-пам'яті процесора. Рядок модифікований, тобто відрізняється від умісту основної пам'яті. Запис у

нього можливий без генерації зовнішнього (відносно локальної шини) циклу звернення.

2. *E*-стан – рядок, наявний тільки у кеш-пам'яті процесора, але він не модифікований. Запис у рядок можливий без генерації зовнішнього циклу звернення. Під час запису в рядок він перейде у *M*-стан.

3. *S*-стан – рядок, наявний у кеш-пам'яті розглядуваного процесора; потенційно може бути наявним у кеш-пам'яті інших процесорів. Його читання можливе без генерації зовнішнього циклу, а запис має супроводжуватися наскрізним записом в основну пам'ять, що спричинить анулювання відповідних рядків у кеш-пам'яті інших процесорів.

4. *I*-стан – рядок, якого немає у кеш-пам'яті, його читання з основної пам'яті може привести до генерації циклу заповнення рядка кеш-пам'яті. Запис у рядок кеш-пам'яті наскрізний з використанням зовнішньої шини.

Підтримка мультипроцесорного режиму роботи. Архітектура *Pentium* дозволяє працювати двом і більше *Pentium*-процесорам у мультипроцесорних системах. Реалізовано інтерфейс побудови двопроцесорних систем із симетричною архітектурою (починаючи з другого покоління *Pentium*).

Засоби задання розміру сторінки пам'яті. *Pentium*-процесор має опцію (спеціальний біт керування) для вибору розміру сторінок пам'яті – традиційну (4 кбайт) і розширену (4 Мбайт). Збільшення розміру сторінки доцільне при використанні громіздких графічних додатків.

Засоби виявлення помилок і тестування за допомогою функціональної надмірності. З метою підвищення надійності у процесорі *Pentium* передбачено внутрішнє виявлення помилок внутрішніх пристроїв (внутрішній контроль паритету) та зовнішнього шинного інтерфейсу, контроль паритету шини адреси й тестування за допомогою функціональної надмірності. Внутрішнє визначення помилок полягає у доповненні кодів команд і даних бітом парності, що дозволяє визначати помилки непомітно як для системи, так і для користувача. Тестування за допомогою функціональної надмірності використовують у програмних додатках, особливо критичних до достовірності результатів. Тестування за допомогою функціональної надмірності базується на роботі двох *Pentium*-процесорів у конфігурації *головний/контрольований* (*master/checker*). У такій конфігурації основний процесор працює у звичайному однопроцесорному режимі. Контрольований процесор виконує ті самі операції, але не керує шиною, і порівнює вихідні сигнали основного процесора з тими сигналами, які він генерує сам. У разі розбіжності отриманих результатів формується сигнал помилки, який може оброблятися системою як переривання. Такий спосіб дозволяє виявляти понад 99 % помилок. Крім того, засоби тестування передбачають можливість виконання вбудованого теста *BIST* (*Built In Self Test*), що забезпечує виявлення помилок мнемокодів, програмовних логічних матриць, тестування

кеш-пам’яті команд і даних, адресних буферів і ПЗП. У цілому самотестування охоплює понад 70 % вузлів процесора. Усі процесори мають стандартний тестовий порт *IEEE* 1149.1 для самотестування за допомогою стандартного інтерфейсу *JTAG*.

Особливості процесорів *Pentium*:

– уведення декількох нових команд, у тому числі розпізнавання моделі процесора;
– уведення засобів керування енергоспоживанням;
– застосування конвеєрної адресації шинних циклів;
– скорочений час (кількість тактів) виконання команд;
– трасування команд і моніторинг продуктивності;
– розширення можливостей віртуального режиму – впровадження віртуалізації прапорця переривань.

Реалізовано нові додаткові засоби налагодження:

– зондовий режим (*Probe Mode*), що забезпечує доступ до внутрішніх регістрів, ПВВ і системної пам’яті процесора. Цей режим дозволяє перевіряти і змінювати стан процесора за допомогою засобів налагодження програм з можливостями, подібними до можливостей внутрішньосхемних емуляторів;
– розширені налагодження *DE* (*Debug Extensions*), які дозволяють ставити контрольні точки за адресами введення-виведення;
– внутрішні лічильники, які використовують для поточного контролю продуктивності та обліку кількості подій;
– покрокове виконання за допомогою команди *CPUID*.

Розширення архітектури. Додатково до базової архітектури 32-розрядних процесорів (див. підрозд. 4.2) *Pentium* має набір регістрів, специфічних для моделі *MSR* (*Model Specific Registers*). Склад регістрів *MSR* може бути різним у різних моделях МП (*Pentium* і *Pentium Pro*), яке призводить до їхньої можливої несумісності. Програмне забезпечення, що використовує регістри *MSR*, має використовувати відомості про процесор, отримані за допомогою команди *CPUID*.

До складу регістрів *MSR* входять:

– тестові регістри *TR*1–*TR*12 (див. підрозд. 4.2);
– засоби моніторингу продуктивності;
– регістри-фіксатори адреси і даних циклу, який викликав спрацювання контролю машинної помилки.

Тестові регістри дозволяють керувати більшістю функціональних вузлів процесора, забезпечуючи можливість докладного тестування їх працездатності. За допомогою бітів регістра *TR*12 можна заборонити нові архітектурні властивості (передбачення і трасування розгалужень, паралельне виконання команд), а також роботу кеш-пам’яті.

Засоби моніторингу продуктивності дозволяють оптимізувати апаратне та програмне забезпечення завдяки виявленню у програмному коді потенційно «вузьких місць». Розробник може спостерігати та підраховувати такти внутрішніх процесорних подій, які впливають на продуктивність операцій читання і запису, вдалі та невдалі звернення до кеш-пам'яті, переривання, використання шини. Це дозволяє оцінювати ефективність програмного коду і досконало налагоджувати програмні додатки або системи для отримання максимальної продуктивності. Засобами моніторингу продуктивності є таймер реального часу і лічильники подій. Таймер *TSC* (*Time Stamp Counter*) виконано на основі 64-розрядного лічильника, уміст якого інкрементується з кожним тактом роботи ядра процесора. Для читання його вмісту призначено команду *RDTSC*. 40-розрядні лічильники подій *CTR*0, *CTR*1 програмуються на підраховування подій різних класів, пов'язаних із шинними операціями, виконанням команд, роботою конвеєрів, кеш-пам'яті, контролем точок зупину тощо. Шестибітові поля типів подій дозволяють кожному з лічильників незалежно підраховувати події з великого списку. Стан лічильників можна попередньо встановлювати і зчитувати програмно. Крім того, існують зовнішні лінії *PM*1–*PM*0, які програмуються на зазначення фактів спрацьовування або переповнення відповідних лічильників. Оскільки ці сигнали можуть змінювати своє значення із частотою, що не перевищує частоту системної шини, через внутрішнє множення частоти кожна поява цих сигналів може означати і декілька (до значення коефіцієнта множення) фактів спрацювання лічильників.

Назва регістрів-фіксаторів адреси і даних циклу, який викликав спрацювання контролю машинної помилки, указує на їх можливу несумісність для різних класів (Pentium та Pentium Pro) або навіть для різних моделей процесорів. Програма, що їх використовує, має звернутися до відомостей про процесор за командою CPUID.

Процесори *Pentium* мають можливість зменшувати енергоспоживання у неробочому режимі. За сигналом *STOPCLK#* процесор вивантажує буфери відкладеного запису і входить у режим *Stop Grant*, у якому припиняється тактування більшості вузлів процесора, що спричинює зниження споживання приблизно в 10 разів. У цьому стані МП припиняє виконання команд і не обслуговує переривання, однак продовжує спостереження за шиною даних. Із цього стану процесор виходить по закінченні сигналу *STOPCLK#*. Керування сигналом *STOPCLK#* разом з використанням режиму *SMM* реалізує механізм розширеного керування живленням *APM (Advanced Power Management)*. Для уповільнення процесора із пропорційним зниженням споживаної потужності сигнал *STOPCLK#* має бути періодичним імпульсним. Скважність імпульсів визначає коефіцієнт простою процесора і його продуктивність.

У стан зниженого споживання *Auto HALT PowerDown* процесор переходить під час виконання команди *HALT*. У цьому стані процесор реагує на всі переривання і також продовжує спостереження за шиною. У режимі припинення зовнішньої синхронізації процесор споживає мінімальну потужність, але не виконує ніяких функцій. Наступна подача сигналу синхронізації супроводжується сигналом апаратного скидання *RESET*.

Основні технічні характеристики процесорів *Pentium* наведено в табл. 4.11.

Процесори Pentium P5 з тактовою частотою 60 та 66 МГц мають напругу живлення 5 В і потребують примусового охолодження. Ці процесори випускаються у корпусах *PGA*-273 (матриця 21×21).

Процесори Pentium P54 з тактовою частотою 75–200 МГц мають напругу живлення 3,3 В, що знижує потужність тепловиділення. Процесори містять розширені засоби керування енергоспоживанням *SMM*. Засоби внутрішнього множення частоти дозволяють реалізувати роботи зовнішньої системної шини на частотах 50, 60 і 66,6 МГц, а ядра процесора – на частотах 75, 90, 100, 120, 133, 150, 166, 200 МГц.

Процесори можуть працювати у двопроцесорних системах, підтримуючи режим *SMP* (*Symmetric Multi-Processing*) або тестування за допомогою функціональної надмірності *FRC* (*Functional Redundancy Checking*). У режимі *SMP* кожний процесор виконує свою задачу, але обидва використовують спільні ресурси комп'ютера, включаючи пам'ять і зовнішні пристрої. У кожний момент часу шиною може керувати тільки один з двох процесорів, але вони можуть мінятися ролями. Для реалізації двопроцесорного режиму введено розширений програмовний контролер переривань *APIC* (*Advanced Programmable Interruption Controller*).

Цей контролер має зовнішні сигнали локальних переривань *LINT* [1:0] та трипровідну інтерфейсну шину (*PICD* [1:0] і *PICCLK*), по якій обидва процесори з'єднуються з контролером *APIC* системної плати. Запити локальних переривань обслуговуються лише тим процесором, на виводи якого (*LINT*0, *LINT*1) надходять їхні сигнали. Режим обробки апаратних переривань *APIC* дозволяється сигналом *APICEN*; надалі він може бути заборонений програмно.

Pentium OverDrive-процесори (розд. 4.3) із частотами 120 і 133 МГц – варіанти процесорів *Pentium* другого покоління зі зниженим енергоспоживанням і подвоєнням частоти. Ці процесори дорожчі від звичайних *Pentium* 120 або 133 МГц. Їхнє застосування доцільне лише тоді, коли з якихось причин неможливо замінити системну плату, а продуктивності *Pentium* 60 або 66 МГц недостатньо. *Pentium OverDrive* 125, 150 та 166 МГц призначені для заміни процесорів першого покоління із частотами 75, 90 та 100 МГц.

Таблиця 4.11. **Основні технічні характеристики процесорів *Pentium***

Процесор	Розрядність			Ємність кеш-пам'яті, кбайт	Передбачення переходів	Підтримка *MMX*	Частота, МГц	
	регістрів	шини даних	шини адреси				шини	процесорного ядра
1	2	3	4	5	6	7	8	9
Pentium (1 покоління)	32	64	32	*L*1: 2 × 8	+	–	60 66	60 66
Pentium (2 покоління)	32	64	32	*L*1: 2 × 8	+	–	 50 60 66	75 90 100 120, 133, 150, 166, 180, 200
Pentium OverDrive	32	64	32	*L*1: 2 × 8	+	–	60 66	120 133
Pentium MMX	32	64	32	*L*1: 2 × 16	+	+	66	166, 200, 233
Pentium Pro	32	64	36	*L*1: 2 × 8 *L*2: 256 (512)	+	–	50 60 66	150 166 180, 200
Pentium II *OverDrive*	32	64	36	*L*1: 2 × 16 *L*2: 512	+	+	66	333
Pentium II	32 64*	64	36	*L*1: 2 × 16 *L*2: 512	+	+	66 100	233 266 300 350, 400, 450

Продовження табл. 4.11

1	2	3	4	5	6	7	8	9
Pentium III	32 64*	64	36	*L*1 2 × 16, *L*2 256 (512)	+	+	100 (133)	450 500 533, 550, 600, 650, 667, 700, 733, 750, 800, 850, 866, 933, 1000, 1130
Pentium II *Xeon*	32 64*	64	36	*L*1: 2 × 16, L2: 512 (1024)	+	+	66 100	233 266 300 350, 400
Pentium III *Xeon*	32 64*	64	36	*L*1: 2 × 16, *L*2: 256 (1 Мбайт) (2 Мбайт)	+	+	100 (133)	550 600 650 667, 700, 733, 750, 800, 866, 933, 1000
Celeron	32 64*	64	36	*L*1: 2 × 16, *L*2: 128**	+	+	66	233 266, 300, 350, 400, 450, 500, 533, 566, 600, 633, 667, 700

Продовження табл. 4.11

1	2	3	4	5	6	7	8	9
AMD K5	32	64	32	*L*1: 16 /8	+	–	50 60 66	75 90 100, 120, 133, 166
AMD K6 MMX	32	64	36	*L*1: 2 × 32, *L*2: 256	+	+ 3*DNow!*	66	166 200, 233, 266, 300
AMD K6-2	32	64	36	*L*1: 2 × 32	+	+ 3*DNow!*	66 100	200 300, 333, 350, 366, 400, 450, 475, 500, 550
AMD K6-2+	32	64	36	*L*1: 2 × 32, *L*2: 128	+	+ 3*DNow!*	66 100	450–550
AMD K6-III	32	64	36	*L*1: 2 × 32, *L*2: 256	+	+ 3*DNow!*	66 100	350–475
Cyrix 6x86MX	32 64*	64	36	*L*1: 16	+	+	60 66 75 83	166 200 233 266

Закінчення табл. 4.11

1	2	3	4	5	6	7	8	9
Cyrix 6*x*86*M*II*xxxGP*	32 64*	64	36	*L*1: 16	+	+	66 75 83	300–433
VIA Cyrix III	32 64*	64	36	*L*1: 2 × 64	+	+ 3*DNow!*	66 100 133	500–600

* Розрядність 64 мають тільки регістри *MMX*.

** *Celeron* 266 і 300 кеш-пам'яті рівня *L*2 не мають.

Примітка. *L*1 – перший рівень – розділена кеш-пам'ять команд і даних; *L*2 – другий рівень – спільна кеш-пам'ять команд і даних.

Процесор Pentium Pro (P6). У цьому процесорі застосоване динамічне виконання команд, тобто комбінація засобів передбачення множинних відгалужень, аналізу проходження даних і віртуального виконання, при якому в процесорі команди можуть виконуватися не в такому порядку, як передбачено програмним кодом. При цьому команди, що не залежать від результатів попередніх операцій, можуть виконуватись у зміненому порядку, але послідовність вивантаження результатів у пам'ять і порти буде відповідати початковому програмному коду. Можливість виконання команд з випередженням (спекулятивне виконання), перевпорядкування команд у разі, якщо команду в одному конвеєрі буде виконано швидше, ніж попередню у другому конвеєрі, і передбачення переходів при динамічному виконанні підвищує продуктивність обчислень.

Процесор підтримує режим тестування за допомогою функціональної надмірності *FRC*. Архітектура процесора дозволяє об'єднати у симетричну мультипроцесорну систему до чотирьох процесорів на одній шині. Сумарну пропускну здатність підвищено за рахунок подвійної незалежної шини. Одна шина – системна – виконує функцію взаємодії ядра процесора з основною пам'яттю та інтерфейсними пристроями, друга – виключно функцію обміну із вторинною кеш-пам'яттю. Перша шина працює на тактовій частоті процесора, друга – на частоті системи. Таке розділення шин дозволяє втричі прискорити обмін процесора з пам'яттю. Завдяки цьому відпадає потреба в окремій зовнішній кеш-пам'яті. Мікропроцесор містить розділені кеш-пам'яті першого рівня (*L*1) для даних і команд ємністю 8 кбайт кожна та об'єднану кеш-пам'ять другого рівня (*L*2). Кеш-пам'ять даних першого рівня підтримує одну операцію завантаження та одну операцію запису за такт. Інтерфейс кеш-пам'яті другого рівня працює з тактовою частотою процесора і може передавати 64 біт за один такт. Ємність кеш-пам'яті другого рівня становить 256 кбайт (при технології 0,6 мкм) та 512 кбайт (при технології 0,35 мкм).

*Процесори Pentium MMX (P*55*C***)** – це процесори, орієнтовані на мультимедійне, 2*D*- і 3*D*-графічне та комунікаційне застосування. В архітектуру *Pentium MMX* додатково введено:

- вісім 64-розрядних *MMX*-регістрів, розташованих у 64 молодших розрядах стека 80-розрядних регістрів блока обчислень з плавучою комою;
- чотири нові типи даних (упаковані байти – 8 байт у 64-розрядному пакеті; упаковані слова – чотири 16-розрядних слова у 64-розрядному пакеті; упаковані подвійні слова – два 32-розрядних подвійних слова у 64-розрядному пакеті; 64-розрядні слова);
- 57 додаткових команд, які поділяються на групи: обміну даними між регістрами *MMX* і цілочисловими регістрами або пам'яттю, арифметичні (додавання та віднімання у різних режимах, множення, комбіна-

ції множення та додавання), порівняння, перетворення форматів, логічні, зсуву (логічного та арифметичного), очищення регістрів *MMX*;
- подвоєні ємності кеш-пам'яті команд та кеш-пам'яті даних (16 кбайт);
- поліпшену логіку передбачення переходів;
- розширену конвеєризацію;
- більш глибоку буферизацію пам'яті (подвоєний розмір буфера відкладеного запису).

Арифметичні та логічні операції у процесорі *Pentium MMX* виконуються паралельно над кожним байтом слова або подвійного слова, що міститься у 64-розрядному *MMX*-регістрі. Таким чином реалізується *SIMD*-модель обробки (*Single Instruction – Multiple Data*). У *SIMD*-командах обробляється одночасно 64-розрядне слово, яке залежно від *MMX*-команди трактується як вісім однобайтових операндів, чотири двобайтові, два чотирибайтові або один восьмибайтовий операнд. *SIMD*-обробка суттєво прискорює виконання мультимедійних алгоритмів, для яких є характерним здійснення ідентичних операцій над великими масивами однотипних даних (наприклад, 16-розрядними відліками оцифрованого звука, 8-розрядними кодами кольору пікселя та ін.). Використання *MMX*-команд дозволяє підвищити швидкість виконання мультимедійних операцій на 60 % порівняно з процесором *Pentium* першого покоління за однакових тактових частот. В інших командах забезпечується сумісність з *Pentium*. Команди *MMX* не впливають на прапорці, не викликають переривань і винятків, доступні для будь-якого режиму роботи процесора.

Поліпшена логіка передбачення переходів забезпечується збільшеною кількістю буферів вибірки команд з випередженням. *Pentium MMX* має чотири 16-розрядних БВВ.

Розширена конвеєризація забезпечується збільшенням етапів виконання цілочислових програм у конвеєрі (див. рис. 4.16) до шести за рахунок додання етапу вибірки (*F*) між етапами передвибірки (*PF*) та декодування команди (*D*1). На етапі вибірки виконується декодування довжини команди, яке у попередніх моделях *Pentium* проводиться на етапі *D*1.

Pentium MMX має подвоєний порівняно з *Pentium* розмір обох частин кеш-пам'яті *L*1 і *L*2. Але у процесорі *Pentium MMX* немає режиму тестування за допомогою функціональної надмірності *FRC*. У двопроцесорних системах підтримується лише режим *SMP*.

Pentium® OverDrive® MMX процесори – варіант процесорів *Pentium MMX* з тактовими частотами 150, 166, 180 і 200 МГц для заміни звичайних (без розширення *MMX*) процесорів *Pentium* 75–200 МГц. Вони вирізняються фіксованим коефіцієнтом множення частот, який дорівнює трьом, і тим, що двопроцесорних режимів немає.

Процесори Pentium для мобільних застосувань (блокнотних персональних комп'ютерів) мають знижене енергоспоживання внаслідок зниження напруги живлення ядра процесора. Вони характеризуються вищою допустимою температурою, що дає змогу використовувати їх у більш жорстких умовах експлуатації. Ці процесори не підтримують двопроцесорний режим, не мають розширеного програмовного контролера переривань *APIC* і відповідних зовнішніх виводів.

Процесор Pentium II має 36-розрядну шину адреси, що дозволяє адресувати фізичну пам'ять ємністю 64 Гбайт. Частота ядра процесора становить 233, 266, 300 та 450 МГц, частота шини – 66,66 та 100 МГц. Висока продуктивність процесора досягається застосуванням у ньому динамічного виконання команд, розширення *MMX* та подвійної незалежної шини.

Кеш-пам'ять першого рівня (*L*1) збільшено до 32 кбайт (16 кбайт – кеш-пам'ять команд, 16 кбайт – кеш-пам'ять даних). Кеш-пам'ять другого рівня (*L*2) розташовано на одній платі з процесором.

Процесор Pemtium III виконано за технологією *SSE* (*Streaming SIMD Extensions*). У *Pentium* III реалізовано 70 нових *SIMD*-команд, що оперують із 128-розрядними регістрами *XMM*0–*XMM*7. Таким чином, виконуючи операцію над двома регістрами, *SSE* фактично оперує чотирма парами чисел. Завдяки цьому процесор може виконувати до чотирьох операцій одночасно, що є корисним у таких застосуваннях:

- тривимірна графіка та моделювання, які використовують обчислення у форматі з плавучою комою;
- обробка сигналів і моделювання процесів із широким діапазоном зміни параметрів;
- генерація тривимірних зображень у програмах реального часу, що не використовують цілочисловий код;
- алгоритми кодування і декодування відеосигналів із блочною обробкою;
- числові алгоритми цифрової фільтрації, що працюють з потоками даних.

Крім розглянутих процесорів фірми *Intel*, існують процесори, аналогічні за продуктивністю *Pentium*, які випускають інші фірми – *AMD* (*AMD K*5 *PR* 75/90/100, *AMD K*6), *CYRIX* (*Cx*86 (*M*1), *CYRIX* 6*x*86, *P*120+, *P*133+, *P*150+, *P*166+, *P*200+, *CYRIX* 6*x*86*MX*, *CYRIX MediaGX*), *Hewlett Packard* (*Merced* (*P*7), *PA RISC*, *PA*-8000), *DEC* (*Alpha* 21068, 21164, 21264).

*Процесори AMD K*5 *PR*75/90/100/120/133/166 виконані за гібридною *CISC*/*RISC*-архітектурою і мають ускладнений конвеєр із п'ятьма блоками обробки, що функціонують паралельно. На відміну від *Pentium,* у цих процесорах можуть одночасно виконуватися команди з плавучою комою, завантаження (зберігання) та переходу. Великий набір і можливість

динамічного перейменування регістрів і наявність блоку завантаження (зберігання) дозволяють виконувати дві операції за один цикл вибірки з пам'яті. У процесорі використовується передбачення розгалужень і зміна порядку виконання команд. Кеш-пам'ять команд має ємність 16 кбайт; кеш-пам'ять даних – 8 кбайт.

*Процесори AMD-K*6 *MMX* мають розділену кеш-пам'ять даних і команд першого рівня ємністю по 32 кбайт. Кеш-пам'ять даних є двопортовою і підтримує зворотний запис. Кеш-пам'ять команд має додаткову область для попередньо декодованих команд. Кожна команда має біти передекодування, які вказують на зміщення початку наступної команди у кеш-пам'яті. Внутрішньої вторинної кеш-пам'яті немає. Процесор підтримує логіку передбачення розгалужень, використовуючи таблицю історії розгалужень з 8 192 елементів; кеш-пам'ять адрес переходу та стек повернення, які забезпечують імовірність правильного передбачення переходу понад 95 %.

На відміну від процесорів *Intel P*54 і *P*55, процесор *AMD-K*6 *MMX* не має вбудованих засобів підтримки мультипроцесорних систем, включаючи *APIC*. У ньому немає сигналу перевірки шинних операцій (*BUSCHK*) і зондового режиму, а також не виводяться сигнали точок зупину та моніторингу продуктивності. У процесорі використовується спекулятивне виконання зі зміною послідовності команд, попереднє посилання даних, перейменування регістрів.

*Процесори AMD-K*6-2 *з технологією* 3*DNow!* виконано за 0,25 мкм-технологією. Вони характеризуються швидкодіючою вбудованою кеш-пам'яттю другого рівня ємністю 256 кбайт, кеш-пам'яттю першого рівня ємністю 64 кбайт. Процесор має ефективну *RISC* архітектуру і поліпшений блок обчислень з плавучою комою. Частота ядра процесора має значення від 300 до 400 МГц. Частота шини – 100 МГц.

Процесори Cyrix 6*x*86 (*M*1) мають можливість динамічного перейменування регістрів, зміни порядку виконання команд, спекулятивне виконання, передбачення переходів. Процесори містять дві кеш-пам'яті: уніфіковану первинну кеш-пам'ять ємністю 16 кбайт, що використовується як для команд, так і для даних, і додаткову 256-байтову кеш-пам'ять команд. Виділена кеш-пам'ять команд дозволяє уникати конфліктів під час звернення до даних і команд у спільній кеш-пам'яті. Процесор здатний одночасно виконувати цілочислові команди та команди з плавучою комою. Етапи циклу виконання команди:

1) вибірка команди *F*;
2) декодування команди *D*1 (стадія 1);
3) декодування команди *D*2 (стадія 2);
4) обчислення адреси *AC*1 (стадія 1);
5) обчислення адреси *AC*2 (стадія 2);
6) виконання *EX*;
7) запис результату *WB*.

Етапи декодування та обчислення адреси конвеєризовано, причому забезпечено можливість перевпорядкування команд на етапах *EX* і *WB*.

Існує версія низьковольтних процесорів для мобільних застосувань.

Процесори Cyrix 6*x*86*MX* – удосконалений варіант процесора *M*l, у який додатково введено можливість виконання набору з 57 мультимедійних команд, сумісних із *MMX*-розширенням, та підвищено тактову частоту. Процесор містить дві кеш-пам'яті: чотиривхідну асоціативну кеш-пам'ять ємністю 64 кбайт зі зворотним записом та високошвидкісну кеш-пам'ять команд ємністю 256 байт. Для мобільних застосувань передбачено ефективну систему керування енергоспоживанням.

Контрольні запитання

1. Які операційні системи можуть використовуватися у МП *Pentium*?
2. Назвіть призначення основних блоків структурної схеми процесора *Pentium*.
3. Поясніть роботу блока передбачення адреси переходу на прикладі виконання команди *JC LABEL*.
4. Який ефект має розділення кеш-пам'яті на кеш-пам'ять команд і кеш-пам'ять даних?
5. Які засоби виявлення помилок має процесор *Pentium*?
6. У чому полягає принцип тестування за допомогою функціональної надмірності?
7. Які функції тестування має процесор *Pentium*?
8. Укажіть призначення та можливості засобів моніторингу продуктивності.
9. Які особливості регістрів-фіксаторів слід ураховувати під час розробки програмного забезпечення?
10. Укажіть призначення та принцип режиму *MMX, SMM*.
11. Як здійснюється перемикання у режим зменшеного енергоспоживання і вихід з нього?

4.5. Особливості архітектури 64-розрядних мікропроцесорів

1997 року фірми *Intel і Hewlett-Packard* розробили нову мікропроцесорну архітектуру *EPIC (Explicitly Parallel Instruction Computing* – явного паралельного обчислення команд), яку було покладено в основу 64-розрядних МП *IA*-64, *McKinley, Itanium, Itanium* 2.

Особливості архітектури *EPIC* такі:

– *велика кількість регістрів загального призначення*. Так, кількість регістрів МП *IA*-64 становить 128 64-розрядних регістрів для операцій з цілими числами і 12 880 – з дробовими;

– *пошук залежностей між командами.* Причому пошук виконує не процесор, а компілятор. Команди МП *IA*-64 компілятор групує у «низку» завдовжки в 128 розрядів. Низка містить три команди і шаблон, у якому зазначено залежності між командами (тобто визначено, чи можна з командою *к*1 запустити паралельно команду *к*2, або ж *к*2 треба виконати тільки після *к*1), а також між іншими низками (чи можна з командою *к*3 із низки *с*1 запустити паралельно команду *к*4 із низки *с*2);
– *масштабованість архітектури.* Тобто пристосування набору команд до великої кількості функціональних пристроїв. Наприклад, одна низка з трьох команд відповідає наборові з трьох функціональних пристроїв процесора. Процесори *IA*-64 можуть містити різну кількість таких функціональних пристроїв, залишаючись при цьому сумісними за кодом. Адже завдяки тому, що в шаблоні зазначено залежність і між низками, процесору з *N* однаковими блоками з трьох функціональних пристроїв буде відповідати командне слово з 3*N* команд (*N* низок);
– *предикація (Predication). Предикацією* називається спосіб обробки умовних розгалужень. Команди з різних гілок умовного розгалуження позначаються предикатними полями (полями умов) і виконуються паралельно, але їх результати не записуються, поки значення предикатних регістрів не визначено. Коли в кінці циклу визначається умова розгалуження, предикатний регістр, який відповідає «правильній» гілці, установлюється в одиницю, а другий – у нуль. Перед записом результатів процесор перевіряє предикатне поле і записує результати лише тих команд, предикатне поле яких містить одиницю;
– *завантаження за припущенням* (*Speculative loading).* Цей механізм призначений знизити простої процесора, зв'язані з чеканням виконання команд завантаження з відносно повільної основної пам'яті. Компілятор переміщає команди завантаження даних із пам'яті так, щоб вони виконалися якомога раніше. Отже, коли дані з пам'яті знадобляться якійсь команді, процесор не буде простоювати.

Процесор *Itanium 2,* виконаний за 0,18 мкм-технологією, здатний виконувати шість команд за один машинний цикл. У сукупності з підвищенням тактової частоти, а також пропускної спроможності системної шини (6,4 Гбіт/с, частота шини – 400 МГц, розрядність шини – 128), цей чинник забезпечує в 1,5–2 рази більшу продуктивність, ніж у «першому» *Itanium.* Процесор має велику ємність кеш-пам'яті третього рівня до 3 Мбайт, розташованої на кристалі, яка працює на частоті ядра.

2003 року на ринку з'явились процесори, зроблені за 0,12 мкм-технологією:

– *Deerfield,* призначений для використання у двопроцесорних системах;
– *Madison,* орієнтований на мультипроцесорні системи.

Наступний процесор, *Montecito*, буде вироблено з використанням 90 нм технології. Його поява планується 2004 року.

64-розрядні МП сімейства *Hammer,* розроблені фірмою *AMD,* базуються на архітектурі *x*86-64, яка являє собою розширення архітетури 32-розрядних процесорів *x*86-4 (рис. 4.18).

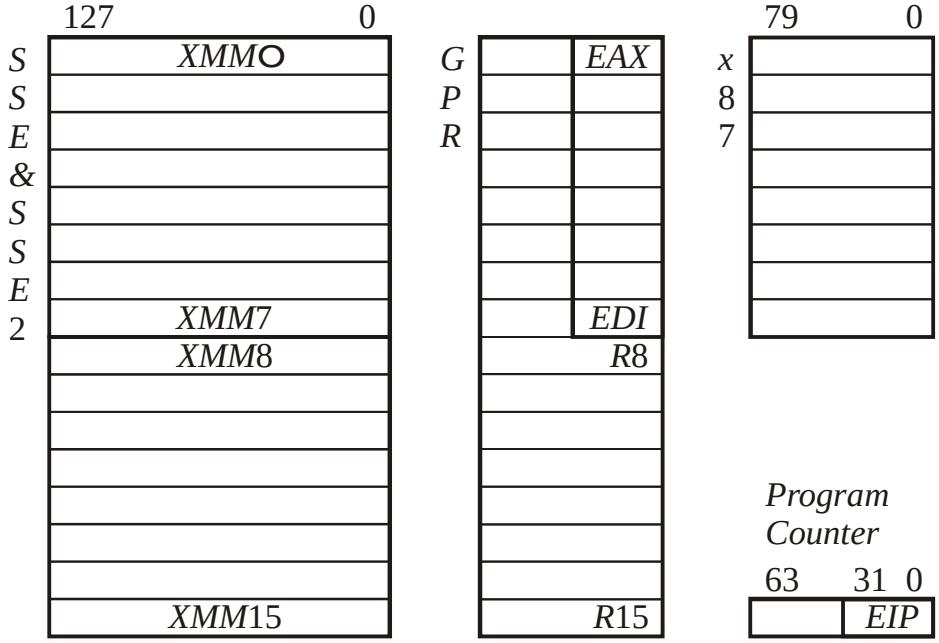


Рис. 4.18. Архітектура *x*86-64

Регістри загального призначення (*GPR*) доповнено ще 8 регістрами *R*8–*R*15, використовуваними в 64-бітному режимі, а розрядність регістрів *EAX, EBX* та інших збільшено з 32 до 64. До блоку *SSE* (*Streaming SIMD Extensions*) як доповнення до восьми 128-розрядних регістрів *XMM*0–*XMM*7 додано ще вісім нових регістрів, що забезпечить підтримку *SSE*2 збільшенням *SIMD*-команд. Розширення регістрів показано на рис. 4.19.

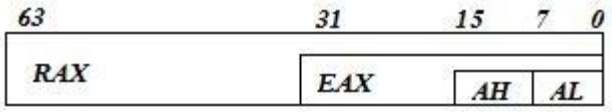


Рис. 4.19. Розширення регістрів загального призначення

Процесор, побудований на основі *x*86-64, може виконувати раніше створені 32-розрядні додатки на відміну від того ж *Intel Itanium*, де систему команд *x*86-32 доводиться емулювати.

Контрольні запитання

1. Порівняйте архітектури 64-розрядних МП.
2. Поясніть позитивний ефект предикації.
3. Які особливості архітектури *EPIC?*

Розділ 5

ПОБУДОВА МОДУЛІВ ПАМ'ЯТІ МІКРОПРОЦЕСОРНИХ СИСТЕМ

5.1. Класифікація систем пам'яті

Система пам'яті є функціональною частиною МПС, призначеною для запису, зберігання та видачі інформації. Технічні засоби, що реалізують функції пам'яті, називаються *запам'ятовувальними пристроями* (ЗП).

Однією з ознак класифікації ЗП є фізична природа середовища, яке використовується для зберігання інформації. За цією ознакою виділяють такі види ЗП: електронні, на пристроях із зарядовим зв'язком (ПЗЗ), магнітні, на циліндричних магнітних доменах, ультразвукові лінії затримки (магнітострикційні, кварцові, зі спеціальних сплавів скла), кріогенні, голографічні.

За швидкістю обміну інформацією з АЛП розрізняють такі *типи пам'яті* (рис. 5.1): регістрова пам'ять МП, внутрішня кеш-пам'ять, зовнішня кеш-пам'ять, оперативна пам'ять, постійна пам'ять, зовнішня пам'ять.

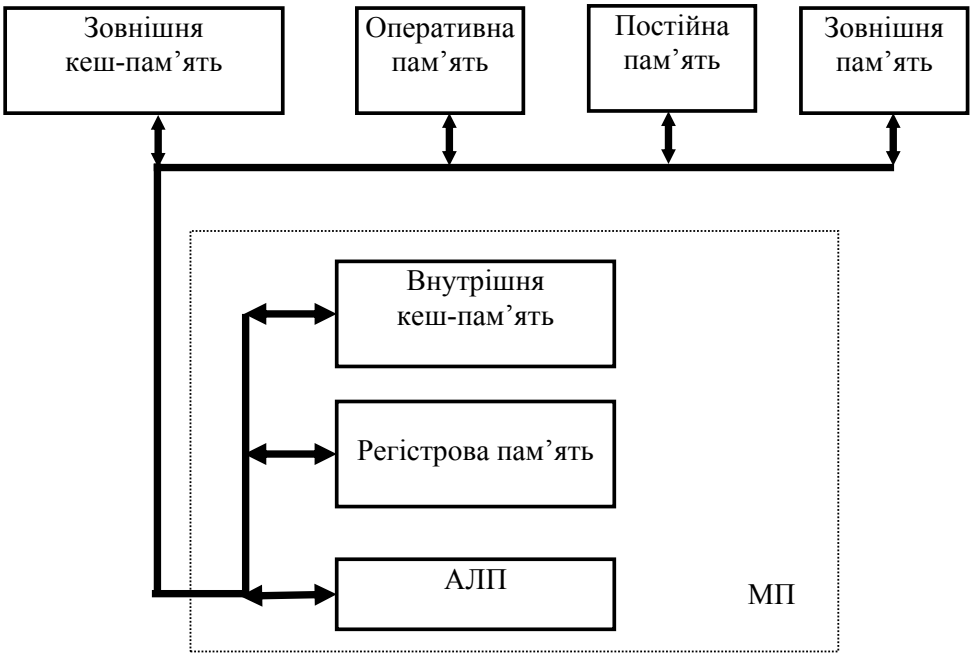


Рис. 5.1. Типи пам'яті у МПС

Надоперативний ЗП або регістрова пам'ять МП являє собою сукупність регістрів загального призначення. Звернення до НОЗП не потребує від МП виставлення адреси на шину *АВ* під час зчитування/запису інформації, тому операції з НОЗП є найбільш швидкодіючими. Час вибірки НОЗП–5–7 нс. Загальна кількість 8- або 16-розрядних регістрів у МП зазвичай становить від 16 до 64.

Внутрішня кеш-пам'ять – це оперативна пам'ять статичного типу ємністю 1–16 кбайт, яку вбудовано безпосередньо у МП. Внутрішня кеш-пам'ять працює на тактовій частоті процесора. У моделях *i*386, *i*486 кеш-пам'ять спільна для даних і команд. У МП *Pentium* кеш-пам'ять використовується окремо для команд і даних.

Зовнішня кеш-пам'ять так само, як і внутрішня, являє собою пам'ять статичного типу, однак має значно більшу ємність. Вона встановлюється на системній платі і працює на частоті шини. Зовнішню кеш-пам'ять призначено для зменшення кількості звернень до інших менш швидкодіючих пристроїв пам'яті. Ємність зовнішньої кеш-пам'яті в сучасних ПЕОМ становить зазвичай 64 кбайт – 1 Мбайт і має тенденцію до збільшення.

Оперативна пам'ять – це пам'ять статичного або динамічного типу. У мікросхемах статичного типу елементом пам'яті є тригер на біполярних транзисторах або транзисторах зі структурою *метал-діелектрик-напівпровідник* (МДН). В ОЗП динамічного типу елементом пам'яті є конденсатор. Оперативна пам'ять припускає зміну свого вмісту в ході виконання процесором обчислювальних операцій з даними і може працювати в режимі запису, зчитування та зберігання інформації. Оперативну пам'ять призначено для зберігання змінної інформації – поточних даних, результатів обчислень. Її ємність становить 1–64 Мбайт, час доступу – 70–200 нс. Оперативний запам'ятовувальний пристрій є енергозалежним, тому що інформація, записана в ньому, втрачається після вимкнення живлення.

Постійна пам'ять являє собою спеціальну мікросхему, що містить інформацію, яка не має змінюватися у процесі виконання програми. Ця інформація заноситься у ПЗП під час виготовлення або на етапі його програмування у спеціальному пристрої – програматорі, і у процесі роботи МПС може тільки зчитуватися. Постійна пам'ять у МПС працює в режимах зберігання та зчитування і використовується для зберігання таблиць, констант, кодів команд програм, стандартних підпрограм, наприклад, підпрограм *BIOS, POS*. Зазвичай, ПЗП має ємність 64–128 кбайт. Записана в ПЗП інформація зберігається після вимкнення живлення. Ця властивість ПЗП одержала назву *енергонезалежності*.

Зовнішня пам'ять реалізується у вигляді нагромаджувачів зі змінними і незмінними носіями: на твердих і гнучких магнітних дисках, стримерах, оптичних і лазерних компакт-дисках (*CD–ROM*). Обмін інформацією з пристроями пам'яті цього типу є найповільнішим, але така пам'ять найбільша за ємністю. Ємність нагромаджувачів на твердих дисках становить 1–100 Гбайт. Низьку швидкодію зовнішніх ЗП на магнітних носіях зумовлено наявністю електромеханічних пристроїв.

Наявність того чи того типу пам'яті поза мікросхемою МП (див. рис. 5.1) зумовлюється функціональним призначенням МПС.

Запам'ятовувальні пристрої характеризуються такими основними параметрами:

– *розрядністю даних* (визначається розрядністю комірки пам'яті);
– *інформаційною ємністю* (визначається кількістю одиниць інформації в бітах, яку ЗП може зберігати одночасно). Інформаційну ємність часто позначають через $N \times n$, де N – кількість n-розрядних слів; n – розрядність шини даних. Так, запис значення інформаційної ємності ЗП $2\,048 \times 8$ або $2\,К \times 8$ означає, що ЗП може зберігати 2 048 байт або 16 384 біт;
– *часом вибірки* (визначається як інтервал від моменту видачі запиту на передачу даних з пам'яті до моменту появи інформації на виході ЗП);
– *тривалістю циклу звернення* (*циклу пам'яті*) $t_{ц}$ (визначається мінімально допустимим інтервалом часу між двома послідовними зверненнями до ЗП);
– *напругою живлення* $U_{живл}$;
– *потужністю енергоспоживання* $P_{сп}$ (визначається добутком струму споживання і напруги джерела живлення). Для деяких типів ЗП наводять два значення потужності енергоспоживання – одне для режиму звернення, коли здійснюється запис або зчитування, інше – для режиму зберігання. Потужність енергоспоживання в режимі зберігання зазвичай істотно менша від потужності енергоспоживання режиму звернення;
– *питомою вартістю* (визначається відношенням вартості ЗП до його інформаційної ємності).

Контрольні запитання

1. Дайте визначення і наведіть приклади енергозалежних та енергонезалежних ЗП.
2. Назвіть типи систем пам'яті.
3. Порівняйте за швидкодією типи систем пам'яті.
4. Визначте інформаційну ємність у бітах і кількість ліній шини даних для ЗП, позначених: а) $1\,024 \times 8$; б) $4\,048 \times 16$.
5. Як визначається питома вартість ЗП?

5.2. Постійні запам'ятовувальні пристрої

Основна складова частина ПЗП – елемент пам'яті, який зберігає 1 біт інформації. Елементи пам'яті об'єднано в матрицю нагромаджувача інформації. Сукупність з n елементів пам'яті, у якій розміщується n-розрядне слово, називають *коміркою пам'яті*, при цьому величина n визначає *розрядність комірки*. Кількість комірок пам'яті дорівнює 2^m, де m – кількість адресних входів, а інформаційна ємність мікросхеми –

$2^m \times n$ біт. Кожна комірка пам'яті має свою адресу. Більшість ПЗП мають словникову організацію, тобто припускають паралельне зчитування n розрядів слова D_{n-1}–D_0.

За способом програмування, тобто за способом занесення інформації, розрізняють такі типи ПЗП: *програмовні одноразово* та *багаторазово. До* першої групи належать ПЗП, програмовні маскою, які програмує виготовлювач способом замовленого фотошаблона (маски), та одноразово програмовні користувачем способом перепалювання плавких перемичок на кристалі (ППЗП, *PROM* програмовний ПЗП). Друга група поділяється на ВІС *з ультрафіолетовим стиранням* (*EPROM*) та з *електричним стиранням*, які ще називають *флеш-пам'ять* (РПЗП, *EEPROM репрограмовний ПЗП*).

Програмовні маскою ПЗП (ПЗПМ або *ROM – Read Only Memory)*. Мікросхеми цього типу використовують для зберігання стандартних підпрограм, фізичних констант, таблиць. Інформацію заносять у ПЗП через спеціальну маску на завершальній стадії виробництва, тому мікросхеми ПЗП цього типу отримали назву програмовні маскою.

Принцип побудови в більшості мікросхем ПЗПМ однаковий і може бути поданий структурою мікросхем К155РЕ21–К155РЕ24 (рис. 5.2).

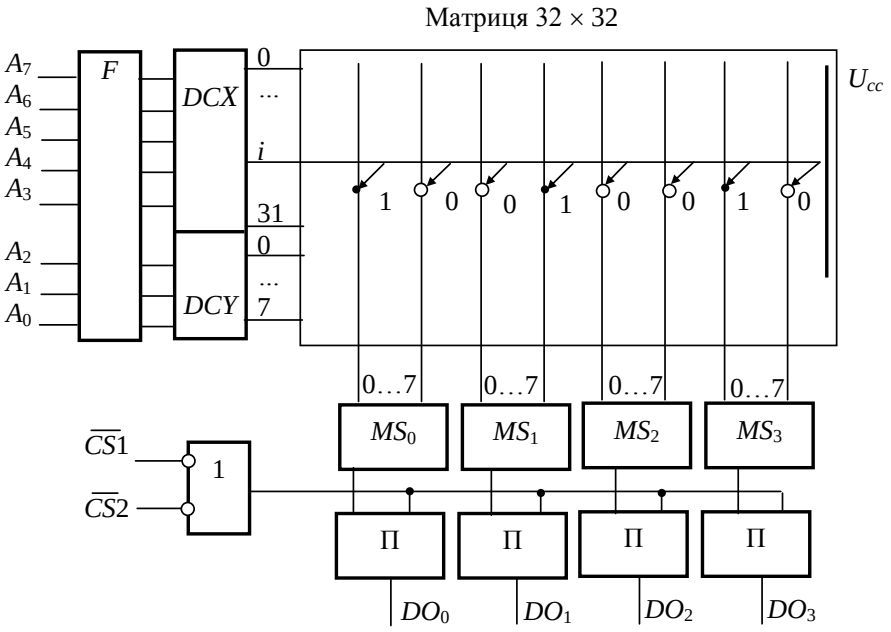


Рис. 5.2. ПЗП, програмовний маскою

Основні елементи структурної схеми: матриця елементів пам'яті (ЕП), дешифратори рядків *DCX* і стовпців *DCY*, селектори MS_0–MS_3, адресний формувач *F*, підсилювачі зчитування П. Елемент пам'яті являє собою резистивну або напівпровідникову (діодну, транзисторну) перемичку, розміщену на перетині рядка і стовпця. Перемички формують у процесі виготовлення мікросхеми в тих точках матриці, куди варто занести

логічну 1. У тих точках матриці, де має бути логічний 0, перемичку не формують.

Матриця на рис. 5.2 містить 32×32 ЕП. Вона складається з 32 транзисторів за кількістю рядків, кожний з яких має 32 емітери за кількістю стовпців. Колектори всіх транзисторів з'єднані із шиною живлення. Бази транзисторів утворюють рядки матриці. Їх підключено до виходів дешифратора адреси рядків. Емітери або мають з'єднання з розрядною шиною («1»), або не мають («0»). Розрядні шини розділено на чотири групи по вісім шин у кожній. Кожна з чотирьох груп шин замикається на селектор, який під керуванням сигналів з виходів дешифратора стовпців *DCY* вибирає з восьми шин одну і комутує її на вихід. На виходи селекторів приєднано підсилювачі зчитування з сигналами стробування $\overline{CS}_1$ і $\overline{CS}_2$. Якщо $\overline{CS}_1 = \overline{CS}_2 = 0$, то підсилювачі відкриті для зчитування інформації, за інших комбінацій цих сигналів – закриті і на їх виходах установлюються *H*-рівні. Вибірку 4-розрядного слова DO_0–DO_3 виконують 8-розрядним кодом адреси, що надходить на адресний формувач *F,* потрібний для узгодження схем на кристалі із зовнішніми колами, і потім на входи дешифраторів рядків A_7–A_3 і стовпців A_2–A_0.

На одному з виходів кожного дешифратора формуються *H*-рівні напруги, які вибирають з матриці 4-розрядне слово. На вихід мікросхеми обране слово надходить, якщо сигнали керування, що дозволяють зчитування $\overline{CS}_1 = \overline{CS}_2 = 0$.

У позначенні мікросхем ПЗПМ використовують літери РЕ, наприклад ВІС КР568РЕ2 з інформаційною ємністю 8 К×8 (рис. 5.3). У цій мікросхемі записано символи міжнародного телеграфного коду № 2. ВІС має 13 адресних входів, що дозволяє адресувати 2^{13} 8-розрядних комірок. Зчитування інформації відбувається, якщо рівень сигналу $\overline{CS}$ нульовий. Якщо $\overline{CS} = 1$, то виходи *D*0–*D*7 знаходяться у високоімпедансному стані, оскільки вхід $\overline{CS}$ інверсний.

Технічні характеристики ПЗПМ наведено в табл. 5.1.

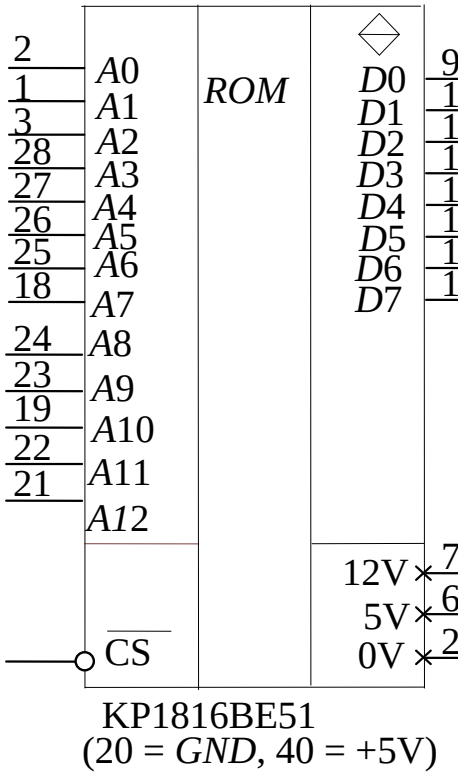


Рис. 5.3. ПЗПМ К568РЕ2

Таблиця 5.1. **Технічні характеристики ВІС ПЗПМ**

Тип мікросхеми	Ємність	$t_ц$, нс	$U_{живл}$, В	$P_{сп}$, мВт
К155РЕ21	256×4	30	5	650
К155РЕ22	256×4	30	5	650
К155РЕ23	256×4	30	5	650
К155РЕ24	256×4	30	5	650
К555РЕ4	2 К×8	110	5	850
К541РЕ1	2 К×8	150	5	1000
К596РЕ1	8 К×8	350	4	640
КА596РЕ2	64 К×16	450	5	1050
К563РЕ1	8 К×8	580	5	50
К563РЕ2	32 К×8	500	5	20
К505РЕ3	512×8	1500	5; –12	500
КР568РЕ1	2 К×8	600	5; 12	450
КР568РЕ2	8 К×8	400	5; 12	600
КР568РЕ3	16 К×8	800	5; 12	300
КМ568РЕ4	8 К×8	300	5; 12	400
К568РЕ5	128 К×8	200	5; 12	300
К1801РЕ1	4 К×16	300	5	75
К1809РЕ1	4 К×16	300	5	275
КА1603РЕ1	2 К×8	410	5	50
КР1610РЕ1	2 К×8	500	5	300
КМ1656РЕ1	2 К×8	80	5	925
КМ1656РЕ2	2 К×8	80	5	925
КМ1656РЕ3	512×8	60	5	775

Щоб знизити споживану потужність $P_{сп}$, деякі мікросхеми, наприклад К596РЕ1, допускають застосування режиму імпульсного живлення, за якого живлення на мікросхему подають тільки під час зчитування інформації.

Занесення в ПЗП інформації називають *прошиванням*. Часто мікросхеми ПЗПМ різних серій мають стандартні прошивання. Наприклад, у мікросхемах ПЗПМ К155РЕ21–К155РЕ24 записано відповідні коди букв російського (РЕ21), латинського (РЕ22) алфавітів, арифметичних знаків і цифр (РЕ23), додаткових знаків (РЕ24). У сукупності ці мікросхеми утворять генератор символів на 96 знаків формату 7×5. Одна з мікросхем серії КР555РЕ4 містить прошивання 160 символів, що відповідають 8-розрядному коду обміну інформації КОІ 2–8 з форматом знаків 7×11. Прошивання кодів алфавітно-цифрових символів містить мікросхема КМ1656РЕ2.

Дві спільно застосовувані мікросхеми К505РЕ3-002, К505РЕ3-003 містять коди букв російського та латинського алфавітів, цифр, арифметичних і додаткових знаків, їх використовують як генератор 96 символів формату 7×9 з горизонтальним розгорненням знаків. Модифікації 0059, 0060

мають те саме призначення, але генерують знаки формату 5×7. Модифікації 0040–0049 містять прошивання коефіцієнтів для швидкого перетворення Фур'є. Ряд модифікацій містить прошивання функції синуса від 0 до 90° з дискретністю 10' (0051, 0052), від 0 до 45° (0068, 0069) і від 45 до 90° (0070, 0071) з дискретністю 5'. Модифікації 0080, 0081 містять прошивання функції $Y = X^2$ при $X = 1...128$.

Модифікації мікросхеми КР568РЕ2 містять стандартні прошивання символів міжнародного телеграфного коду № 2 форматів 5×7 і 7×9 (0001), символів російського і латинського алфавітів, кодових таблиць, цифр і арифметичних знаків (0003,0011) функції синуса від 0 до 90° (0309), асемблера (0303–0306), редактора текстів (0301, 0302).

Мікросхема КР568РЕ2-0001 має прошивання міжнародних телеграфних кодів № 2 і 5, а КР568РЕ3-0002 – редактора текстів для асемблера.

Модифікації мікросхеми КР1610РЕ1-0100–КР1610РЕ1-0107 містять прошивання програмного забезпечення мікро-ЕОМ.

Названі мікросхеми ПЗПМ зі стандартними прошиваннями варто розглядати як приклади: кількість таких мікросхем і їх модифікацій постійно зростає.

Мікросхеми ПЗПМ працюють у режимах зберігання і зчитування. Для зчитування інформації треба подати код адреси і сигнали керування, що дозволяють зчитування. Сигнали керування можна подавати *H*-рівнем, якщо вхід *CS* прямий, або *L*-рівнем, якщо вхід інверсний.

Багато мікросхем мають кілька входів керування, зазвичай зв'язаних визначеним логічним оператором. У таких мікросхемах слід подавати на керувальні входи визначену комбінацію сигналів, наприклад 00 або 110, щоб сформувати умову дозволу зчитування.

Вихідні сигнали в усіх мікросхем ПЗПМ мають ТТЛ-рівні. Виходи побудовано переважно за схемою з трьома станами.

Одноразово програмовні ПЗП (ППЗП або *PROM – Programmed Read Only Memory*). Постійні ЗП цього типу використовують для зберігання налагоджених програм МПС керування різноманітного призначення, зазвичай у серійних виробах. Мікросхеми ППЗП за принципом побудови і функціонування аналогічні ПЗПМ, але істотно відмінні від них тим, що допускають програмування на місці їх застосування користувачем. Програмування або занесення інформації в одноразово програмовні ПЗП полягає в руйнуванні (перепалюванні) частини плавких перемичок на поверхні кристала імпульсами струму амплітудою 30...50 мА у тих перетинах рядків і стовпців матриці нагромаджувача, куди потрібно записати 0(1). У початковому стані ВІС ПЗП має перемички в усіх перетинах рядків і стовпців. Програмування ВІС виконують у програматорі поданням електричних імпульсів, за допомогою яких перепалюють відповідні

перемички. Тому таке програмування одноразове. Технічні засоби для виконання цієї операції досить прості, їх може побудувати сам користувач. Ця обставина та низька вартість і доступність мікросхем ППЗП зумовили їх значне поширення на практиці.

Типовий варіант реалізації мікросхеми ППЗП показано на рис. 5.4. Для конкретності розгляду наведено структуру мікросхеми К556РТ4. У всіх основних елементах вона повторює структуру ПЗПМ (див. рис. 5.2), але має додаткові пристрої F_1–F_4 для формування струму програмування.

Матриця до програмування, тобто у вихідному стані, містить однорідний масив провідних перемичок, що з'єднують рядки і стовпці в усіх точках їх перетинань. Перемички встановлюють з ніхрому (у мікросхемах серії К556 та ін.), з полікристалічного кремнію (К541), із силіциду платини (К1608) та інших матеріалів. Перемичка в матриці виконує роль ЕП. Наявність перемички кодують логічною 1, якщо підсилювач зчитування являє собою повторювач, і логічним 0, якщо підсилювач зчитування – інвертор, як на рис. 5.4. Отже, мікросхема ППЗП у вихідному стані перед програмуванням залежно від характеристики вихідного підсилювача може мати матрицю, заповнену або логічним 0, або логічною 1. Програмування мікросхеми, матрицю якої у вихідному стані заповнено 0, полягає в перепалюванні перемичок у тих ЕП, де мають зберігатися 1. Якщо матрицю у вихідному стані заповнено 1, то перепалюють перемички в ЕП, де мають зберігатися 0.

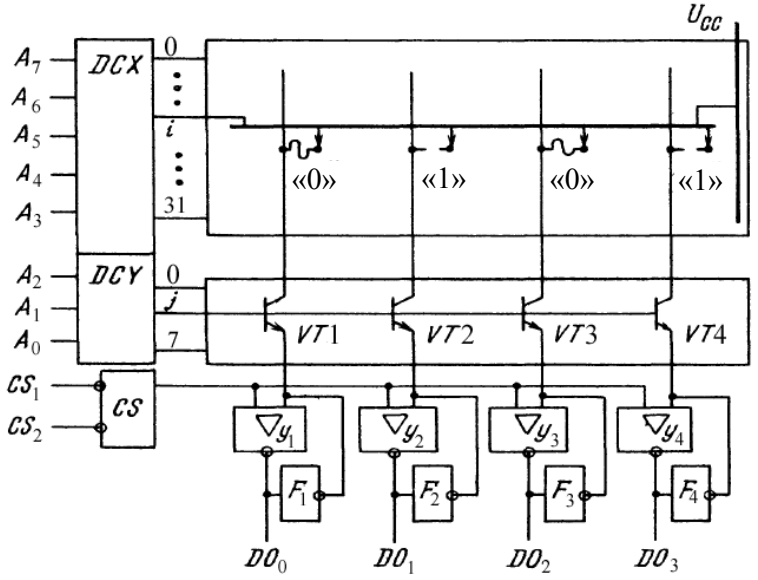


Рис. 5.4. Побудова мікросхеми ППЗП

У позначенні мікросхем одноразово програмовних ПЗП використовують літери РТ, наприклад ВІС КР556РТ16 (8 К×8) (рис. 5.5).

Мікросхеми ППЗП, що випускає вітчизняна промисловість (табл. 5.2), здебільше виготовлено за ТТЛШ-технологією, і серед них переважає серія К556. Функціональний склад серії містить мікросхеми ємністю до 64 кбіт зі словниковою 4- і 8-розрядною організацією, з часом вибірки 45–85 нс і рівнем споживаної потужності від 0,6 до 1 Вт.

Невелику частину мікросхем ППЗП виконано за іншими технологіями: ІІЛ (К541), *n*-МДН (К565), ЕСЛ (К500, К1500), КМДН (К1623). Мікросхеми серії К1623 вирізняються найнижчим рівнем енергоспоживання, але за швидкодією вони поступаються мікросхемам серії К556.

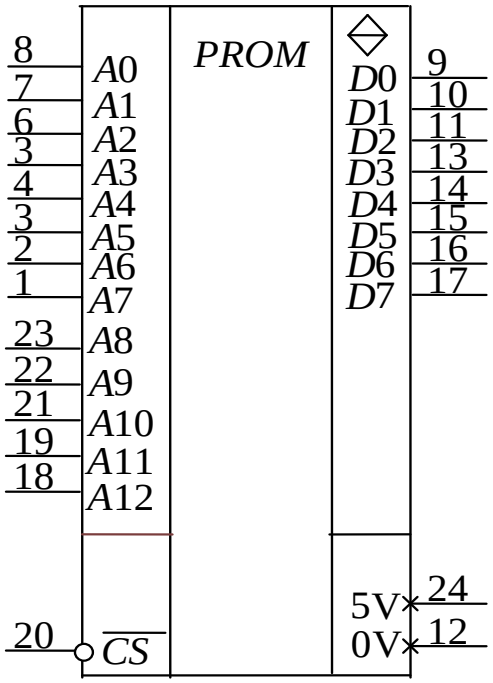


Рис. 5.5. ПЗП К556РТ16

Таблиця 5.2. **Технічні характеристики ВІС ППЗП**

Тип мікросхеми	Ємність, біт	$t_ц$, нс	$P_{сп}$, мВт	Початковий стан
КР556РТ1	ПЛМ	70	850	–
КР556РТ2	ПЛМ	80	950	–
КР556РТ4	256×4	70	690	0
КР556РТ5	512×8	80	1000	1
КР556РТ6(РТ7)	2 К×8	80	900	0
КР556РТ11	256×4	45	650	0
КР556РТ12(РТ13)	1 К×4	60	740	0
КР556РТ14(РТ15)	2 К×4	60	740	0
КР556РТ16	8 К×8	85	1000	0
КР556РТ17	512×8	50	900	1
КР556РТ18	2 К×8	60	950	0
К541РТ1	256×4	80	400	0
К541РТ2	2 К×8	100	770	0
К1608РТ2	512×8	40	920	0
К1623РТ1	2 К×8	200	–	–
К155РЕ3	32×8	70	550	0
К1500РТ1416	256×4	20	670	1

Для мікросхем ППЗП усіх серій, крім К500, К1500, К565, характерні такі властивості, як єдина напруга живлення 5 В, наявність вхідних і вихідних ТТЛ-рівнів напруги логічного 0 (0,4 В) і логічної 1 (2,4 В) і, отже,

повна сумісність мікросхем, однотипні виходи: або з трьома станами, або з відкритим колектором. Мікросхеми з виходами ТТЛ потребують до них зовнішніх резисторів і джерела напруги живлення.

ПЗП, багаторазово програмовних з ультрафіолетовим (УФ) стиранням (репрограмовні ПЗП – РПЗП-УФ або *EPROM – Erasable Programmed Read Only Memory*). Основна відмітна риса мікросхем РПЗП полягає в їх здатності до багаторазового (від 100 до 10 тис.) перепрограмування самим користувачем. Ця властивість мікросхем забезпечує застосування ЕП із властивостями «керованих перемичок», функції яких виконують транзистори *n*-МОН з використанням механізму лавинної інжекції заряду (ЛІЗМОН) з подвійним затвором.

Такий ЕП на являє собою *n*-МОН транзистор, у якого в однорідному діелектрику SiO_2 під затвором сформовано ізольовану провідну ділянку з металу або полікристалічного кремнію. Цей затвор одержав назву «плавучого» (ПЗ).

У режимі програмування на керувальний затвор, джерело і стік подають імпульс напруги 21–25 В позитивної полярності. У зворотно зміщених *p-n* переходах виникає процес лавинного розмноження носіїв заряду й інжекція частини електронів у ПЗ. У результаті нагромадження на ПЗ негативного заряду передатна характеристика транзистора зміщається в область високої граничної напруги (праворуч), що відповідає запису 0.

Для стирання інформації перед новим циклом програмування слід витіснити нагромаджений під затвором заряд. Стирають інформацію опроміненням мікросхеми ультрафіолетовим світлом через спеціальне вікно у верхній частині корпусу. У результаті посилення теплового руху за рахунок отриманої енергії від джерела УФ випромінювання електрони ЕП розсмоктуються з ПЗ у підкладку.

Це зміщує передатну характеристику транзистора в область низької граничної напруги (ліворуч), що відповідає запису 1 (рис. 5.6).

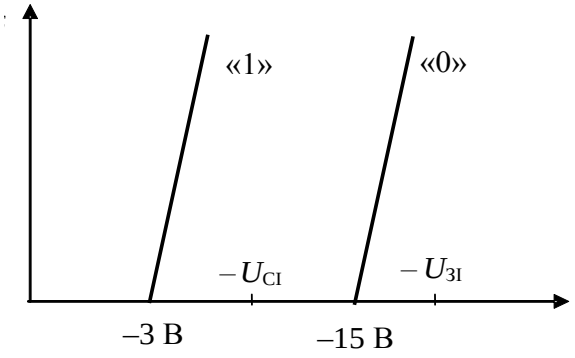


Рис. 5.6. Передатна характеристика ЕП

Стирання можна зробити, не витягаючи мікросхему з контактного пристрою, але тоді треба вимкнути напруги живлення і сигнали. Типові джерела випромінювання – дугові ртутні лампи і лампи з парами ртуті в кварцових балонах: ДРТ-220, ДРТ-375, ДБ-8, ДБ-60 та ін. Випромінювання проникає до напівпровідникового кристала РПЗП через прозоре вікно в кришці корпусу. Час стирання інформації становить 30–60 хв. Відстань від корпусу до балона лампи має бути 2,5 см.

Теоретичними розрахунками доведено можливість зберігання заряду сотні років. На практиці цей час обмежують для одних типів мікросхем декількома тисячами годин, для інших – декількома роками, наприклад, у К573РФ6 гарантійний термін збереження інформації без живлення становить п’ять років.

У позначенні мікросхем багаторазово програмованих ПЗП з ультрафіолетовим стиранням використовують літери РФ, наприклад, ВІС КР573РФ6 (8 К×8) (рис 5.7). Крім розглянутих вище виводів, ВІС ПЗП містить вивід $\overline{OE}$ (*Output Enable*) – дозвіл виходу і *PR* (*Programming*) – логічний сигнал програмування. Сигнал $\overline{OE}$ діє так само, як і сигнал $\overline{CS}$.

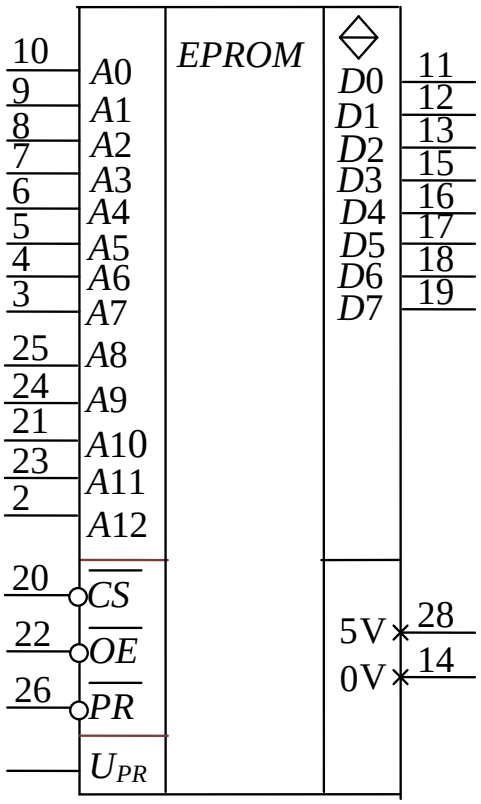


Рис. 5.7. ВІС ПЗП К5736РФ6

Характеристики мікросхем РПЗП-УФ наведено в табл. 5.3.

Таблиця 5.3. **Технічні характеристики РПЗП-УФ**

Тип мікросхеми	Ємність, біт	$t_ц$, мкс	$P_{сп}$, мВт	$U_{живл}$, В	$U_{пр}$, В	$t_{пр}$, с	$t_{ст}$, хв
К573РФ1	1 К×8	0,45	1100	±5; –12	26	300	30хв
К573РФ2	2 К×8	0,45	580	5	25	100	30хв
К573РФ3	4 К×16	0,45	450	5	18	40	30хв
К573РФ4	8 К×8	0,5	700	5; 12	25	800	30хв
К573РФ5	2 К×8	0,45	580	5	25	100	30хв
К573РФ6	8 К×8	0,3	870	5	19	400	30хв
К573РФ7	32 К×8	0,3	600	5	25	–	–
К573РФ9	128 К×8	0,35	550	5	25	–	–

Серед мікросхем серії К573 виділяється складнішою структурою і розширеними функціональними можливостями мікросхема К573РФ3 ємністю 4 К×16 біт. Її відмітна риса полягає в тому, що вона пристосована для безпосередньої роботи зі стандартною магістраллю, маючи відповідні вбудовані засоби інтерфейсу забезпечення режиму обміну з нею. Крім цього, мікросхема має вбудований програмувальний адресний пристрій, що дозволяє без додаткового устаткування поєднувати до восьми мікросхем у блок ПЗП з їх підключенням до магістралі. Зазначені можливості мікросхеми дозволяють комплектувати її з мікросхемами ОЗП К1809РУ1, ПЗП К1809РЕ1, К1801РЕ1, з якими вона цілком сумісна за розведенням і вхідними сигналами у режимах зчитування та зберігання, для створення універсальних модулів ЗП на основі стандартної магістралі.

Режими роботи мікросхем РПЗП-УФ: зберігання, зчитування, запис (програмування) – забезпечуються сигналами керування, наведеними в табл. 5.4.

Таблиця 5.4. **Режими мікросхем РПЗП-УФ**

Тип мікросхеми	Параметри	Режим роботи			
		Запис слова	Контроль запису	Зчитування	Зберігання
1	2	3	4	5	6
К573РФ1 1 К×8 100 циклів	$\overline{CS}$, В	12	–	0	5
	$\overline{PR}$, В	26*	–	0	*x*
	τ, мс	300	–	$0{,}45\cdot10^{-3}$**	–
К573РФ2 (РФ5) 2 К×8 100 циклів (РФ2) 25 циклів (РФ5)	$\overline{CS}$, рівень	*H**	*L*	*L*	*H*
	$\overline{OE}$, рівень	*H*	*L*	*L*	*x*
	U_{PR}, *B*	25	25	5	5
	τ, мс	$2\cdot10^{4}$	–	$0{,}35\cdot10^{-3}$	–

Продовження табл. 5.4

1	2	3	4	5	6
К573РФ3 4 К×16 10 циклів	$\overline{CS}$, рівень $\overline{CE}$, рівень $\overline{OE}$, рівень $\overline{PR}$, рівень U_{PR}, *В* τ, мс	*L** *H* *H* *L* 18 10	*L** *L* *H* *H* 18 10	*L* *L* *L* *H* 5 $0{,}45\cdot10^{-3}$	*H* *H* *H* *L* 5 –
К573РФ4 8 К×8 25 циклів	$\overline{CS}$, рівень $\overline{PR}$, рівень U_{PR}, *В* τ, мс	*L* *L** 25 100	– – – –	*L* *H* 5 $0{,}5\cdot10^{-3}$	*H* *x* 5 –
К573РФ6 8 К×8 10^4 циклів	$\overline{CS}$, рівень $\overline{OE}$, рівень $\overline{PR}$, рівень U_{PR}, *В* τ, мс	*L* *H* *L** 19 50	*L* *L** *H* 19 –	*L* *L* *H* 5 $0{,}3\cdot10^{-3}$	*H* *x* *x* 5 –

* Вплив у формі імпульсу тривалістю τ.

** Час вибору адреси.

Реалізуючи керування, треба мати на увазі, що об'єднані адресні й інформаційні виводи працюють у мультиплексному режимі: спочатку, якщо $\overline{CS} = 0$ і внутрішній код мікросхеми збігається з прийнятим A_{13}–A_{15}, відбувається запис у вхідний регістр коду адреси A_1–A_{12}, потім виводи переходять у режим приймання даних DI_0–DI_{15} для запису або в режим виводу зчитаних даних DO_0–DO_{15} у магістраль. Під час програмування сигнал $\overline{CE}$ на етапі приймання адреси має значення 0, потім у процесі приймання даних набуває значення 1.

У режимі зчитування після фіксації адреси на вхідному регістрі виходи переходять у третій стан, а зчитана з матриці інформація розміщається у внутрішньому вихідному регістрі. На виходах вона з'являється під час сигналу дозволу $\overline{OE}$.

Недоліки мікросхем РПЗП-УФ: мала кількість циклів перепрограмування (від 10 до 100), що зумовлено швидким старінням діелектрика під впливом УФ випромінювання, потреба вилучення з апаратури для стирання інформації, значний час стирання, потреба в спеціальному устаткуванні для стирання, висока чутливість до висвітлення та можливість випадкового стирання інформації. Разом з тим у мікросхем цієї групи є й істотні переваги: порівняно висока швидкодія, велика розмаїтість варіантів виконання щодо інформаційної ємності, невисока вартість і доступність. Ці властивості мікросхем РПЗП-УФ зумовлюють їх широке застосування в розробках.

Багаторазово програмовні з електричним стиранням ПЗП (РПЗП-ЕС або *EEPROM – Electrical Erasable Programmed Read Only Memory*) припускають тисячі циклів програмування.

Елемент пам'яті зі структурою МНОН (метал *Al* – нітрид кремнію Si_3N_4 – оксид кремнію SiO_2 – напівпровідник Si) являє собою МОН-транзистор з індукованим каналом *p*-типу або *n*-типу, що має двошаровий діелектрик під затвором. Верхній шар формують з нітриду кремнію, нижній – з оксиду кремнію, причому нижній шар значно тонший за верхній. Якщо до затвору відносно підкладки прикласти імпульс напруги позитивної полярності з амплітудою 30–40 В, то під дією сильного електричного поля між затвором і підкладкою електрони здобувають достатньо енергії, щоб пройти тонкий діелектричний шар до межі розділу двох діелектриків. Верхній шар (нітриду кремнію) має значну товщину, тому електрони подолати його не можуть.

Нагромаджений на межі розділу двох діелектричних шарів заряд електронів знижує граничну напругу і зміщує передатну характеристику транзистора вліво. Цей стан ЕП відповідає логічній 1. Логічному 0 відповідає стан транзистора без заряду електронів у діелектрику. Щоб забезпечити цей стан, на затвор подають імпульс напруги негативної полярності з амплітудою 30–40 В. При цьому електрони витісняються в підкладку. Якщо заряду електронів під затвором немає, то передатна характеристика зміщається в область високих граничних напруг (див. рис. 5.6).

Режиму стирання і програмування можна досягти за допомогою напруги однієї полярності: негативної для р-МНОН, позитивної для п-МНОН структур. Ця можливість основана на використанні явища лавинної інжекції електронів під затвор, що відбувається, якщо до джерела і стоку прикласти імпульс негативної напруги 30–40 В, а затвор і підкладку з'єднати з корпусом. У результаті електричного пробою переходів джерело – підкладка і стік – підкладка відбувається лавинне розмноження електронів і інжекція деяких з них, які мають достатню кінетичну енергію («гарячих» електронів), на границю між шарами діелектриків. Для стирання треба подати імпульс негативної напруги на затвор. У режимі зчитування на затвор подають напругу Uзч, значення якої лежить між двома граничними рівнями. Якщо в ЕП записано 1, то транзистор відкриється, якщо 0 – залишиться в закритому стані. Підсилювач зчитування трансформує стан шини в рівень напруги 1 або 0 на виході мікросхеми.

Принцип побудови і режим роботи РПЗП-ЕС розглянемо на прикладі мікросхеми КР1601РРЗ ємністю 2 К×8 з ЕП на *p*-МНОН транзисторах.

Структурна схема (рис. 5.8) містить усі елементи, потрібні для роботи мікросхеми як ПЗП: матрицю з елементами пам'яті, дешифратори коду адреси рядків і стовпців, селектор (ключі вибору стовпців), ПВВ. Крім того, у структурі передбачено функціональні вузли, що

забезпечують її роботу в режимах стирання і програмування (запису інформації) – це комутатори режимів і формувачі імпульсів напруг відповідної амплітуди і тривалості із напруги програмування U_{PR}. Порівняно з мікросхемами ПЗПМ і ППЗП систему керувальних сигналів доповнено сигналами програмування *PR* і стирання *ER*. Нагромаджувач з матричною організацією містить 128 рядків і 128 стовпців, на перетинаннях яких розміщено 16 384 елементів пам'яті. Керують нагромаджувачем сьома старшими розрядами адресного коду, який після дешифрування вибирає рядок з 128 елементами пам'яті. Сигнали, зчитані з елементів обраного рядка, надходять на входи селектора, призначення якого полягає у виборі з 128-розрядного коду на входах восьми розрядів, що далі надходять через ПВВ на виходи мікросхеми.

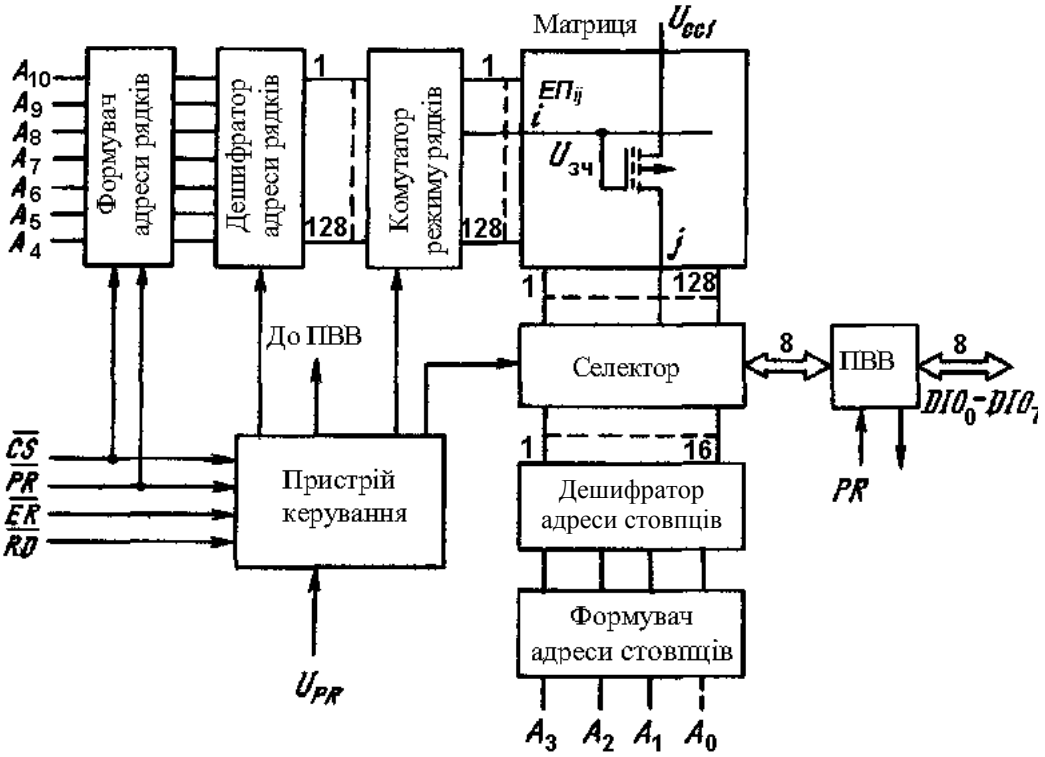


Рис. 5.8. Структурна схема мікросхеми РПЗУ-ЕС

Селектором керують чотири молодші розряди адресного коду, які після дешифрування забезпечують вибірку одного 8-розрядного слова з 16, що містяться в обраному рядку. Пристрій керування під впливом сигналів на його входах забезпечує роботу мікросхеми в одному з таких режимів: зберігання, зчитування, стирання, запису (програмування). Керувальні сигнали мають таке призначення: $\overline{CS}$ – вибір мікросхеми; $\overline{PR}$ – дозвіл режиму запису (програмування); U_{PR} – напруга програмування; $\overline{RD}$ – сигнал зчитування; $\overline{ER}$ – сигнал стирання інформації. Входи сигналів інверсні, тому значення дозволу цих сигналів – 0. Багато мікросхем групи ЕС допускають вибіркове стирання за адресою. Умови

реалізації названих режимів для мікросхем РПЗП групи ЕС наведено в табл. 5.5. Розглянемо ці умови для мікросхеми КР1601РР3.

У режимі загального стирання на керувальні входи подають сигнали, що відповідають табл. 5.5, зокрема напругу програмування U_{PR} = –36 В. Процес стирання починається з моменту подачі імпульсу $\overline{ER}$, тривалість якого має бути від 100 до 200 мс. Після стирання всі ЕП матриці переходять у стан, що відповідає логічному 0. У цьому режимі сигнали на адресних та інформаційних виводах можуть мати довільні значення.

Таблиця 5.5. **Режими мікросхем РПЗП-ЕС**

Тип мікросхеми	Параметри	Режими роботи			
		Стирання	Запис слова	Зчитування	Зберігання
КР558РР2 2 К×8 *n*-МНОН 10^4 циклів	$\overline{CS}$, рівень $\overline{OE}$, рівень U_{PR}, В τ, мс	*H* *L* 18* 10^3	*H* *H* 18* 10	*L* *L* 5 $0{,}35\cdot10^{-3}$**	*H* x 5 –
КР558РР3 8 К×8 *n*-МНОН 100 циклів	$\overline{CS}$, рівень $\overline{OE}$, рівень $\overline{ER}$, рівень U_{PR}, В τ, мс	*L** *H* *L* 18 $2\cdot10^4$	*L** *L** *H* 24 5***	*L* *L* *H* 0 $0{,}35\cdot10^{-3}$	*H* *x* *x* *x* –
КР1601РР3 2 К×8 *p*-МНОН 10^4 циклів	$\overline{CS}$, рівень $\overline{RD}$, рівень $\overline{PR}$, рівень $\overline{ER}$, рівень U_{PR}, В τ, мс	*L* *H* *L* 0* –36 200	*L* *H* *L** *H* –36 20	*L* *L* *H* *H* –12 $0{,}4\cdot10^{-3}$	*H* *x* *x* *x* *x* –
КМ1609РР1 2 К×8 ЛІЗМОН 10^4 циклів	$\overline{CS}$, рівень $\overline{OE}$, В U_{PR}, В τ, мс	*L* 12 21* 12	*L* 5 21* 12	*L* 0 5 $0{,}3\cdot10^{-3}$	*H* *x* 5 –
К573РР2 2 К×8 ЛІЗМОН 10^4 циклів	$\overline{CS}$, рівень $\overline{OE}$, В U_{PR}, В τ, мс	*L* 12 22* 50	*L* 12 22* 50	*L* 0 5 $0{,}35\cdot10^{-3}$	*H* *x* 5 –

* Вплив у формі імпульсу тривалістю τ.

** Час вибору адреси.

*** Час утримання сигналу вибору мікросхеми щодо сигналу дозволу $\overline{OE}$.

Мікросхема КР1601РР3 допускає порядкове стирання. Цей режим відрізняється від розглянутого значенням сигналу $\overline{PR}$ = 0, наявністю на всіх інформаційних виводах сигналів з рівнем 1, а на адресних входах – сигналів адреси рядка A_4–A_{10}, за якою треба стерти інформацію з усіх 128 ЕП. Час вибіркового стирання такий самий, як і загального.

У режимі запису (програмування) на виводи мікросхеми подають записуваний байт, код адреси, сигнали керування за табл. 5.5 і потім імпульс

сигналу програмування $\overline{PR}$ = 0 на час 20 мс. Для програмування в автоматичному режимі всієї мікросхеми з кількістю адрес 2 048 потрібно 41 с.

У режимі зчитування на вивід U_{PR} комутують напругу живлення мінус 12 В для зниження споживаної потужності, подають код адреси і сигнали керування згідно з табл. 5.5, причому форма сигналу зчитування $\overline{RD}$ має бути імпульсною. Через 0,4 мкс на інформаційних виходах з'являється слово, що зчитується.

Режим зберігання забезпечують сигналом $\overline{CS}$ = 1, який забороняє звертання до мікросхеми незалежно від значень сигналів на інших входах. Можливий другий варіант забезпечення режиму зберігання, якщо використовують імпульсне живлення напругою мінус 12 В. Такий режим дозволяє зменшувати споживану потужність. Коли в паузах між звертаннями до мікросхеми вимикають напругу живлення, вона переходить у режим зберігання. Керувати переключеннями живлення доцільно сигналом $\overline{CS}$.

У процесі експлуатації мікросхем РПЗП слід забезпечити відповідний порядок умикання і вимикання напруг живлення і програмування: під час умикання спочатку подають 5 В, потім мінус 12 В і останньою напругу програмування, під час вимикання послідовність міняється на зворотну. Можна всі три напруги вмикати і вимикати одночасно.

У позначенні мікросхем цього типу з паралельним уведенням і виведенням інформації використовують літери РР, наприклад ВІС К558РР3 (рис. 5.9). Інтегральна схема ПЗП має вхід сигналу керування стиранням *ER* (*Erase*).

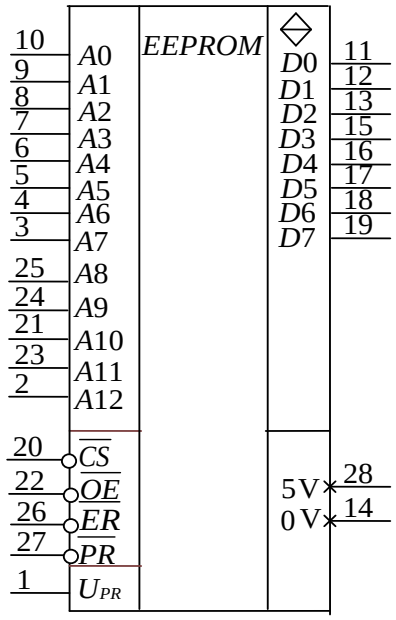


Рис. 5.9. ВІС ПЗП К558РР3

Характеристики мікросхем РПЗП-ЕС найпопулярніших серій наведено в табл. 5.6.

Таблиця 5.6. **Мікросхеми РПЗП**

Тип мікросхеми	Ємність, біт	$t_ц$, мкс	$P_{сп}$, мВт	$U_{живл}$, В	$U_{пр}$, В	$t_{пр}$, с	$t_{ст}$, с
КР558РР1	256×8	5,00	307	5; –12	–30	1	0,005
КР558РР2	2 К×8	0,35	480	5	18	20	1
КР558РР3	8 К×8	0,40	400	5	24	40	20
КР1601РР1	1 К×4	1,80	625	5; –12	–32	20	0,2
КР1601РР3	2 К×8	1,60	850	5; –12	–36	40	0,2
КМ1609РР1	2 К×8	0,35	525	5	21	24	0,012
К1609РР2	8 К×8	0,30	525	5	22	–	–
К1611РР1	8 К×8	0,30	850	5	22	–	–
К573РР2	2 К×8	0,35	620	5	22	100	0,05

Мікросхеми РПЗП з ЕП на *p*-МНОН транзисторах КР558РР1, КР1601РР1, КР1601РР3 (табл. 5.6) мають порівняно низьку швидкодію, високу напругу програмування (30–40 В) і потребують двох джерел живлення.

Щоб поліпшити характеристики РПЗП, широко застосовують технологію виготовлення ЕП на *n*-МНОН транзисторах. Такі ЕП улаштовані аналогічно розглянутим, але мають провідність підкладки *p*-типу, а джерело і стік *n*-типу. Мікросхеми з ЕП на *n*-МНОН транзисторах КР558РР2, КР558РР3, К1611РР1 мають утроє більшу швидкодію, знижену до 22 В напругу програмування і працюють від одного джерела живлення.

Перевага мікросхем РПЗП групи ЕС у можливості перепрограмування без вилучення їх із пристрою, де вони працюють. Ще одна позитивна властивість мікросхем цієї групи – значна кількість циклів перепрограмування, що досягає для більшості мікросхем 10 тис. Ця їх властивість у сполученні з енергетичною незалежністю дозволяє широко використовувати мікросхеми в апаратурі як вбудовані ПЗП зі змінюваною інформацією. Гарантійний термін зберігання інформації із вимкненим живленням становить від трьох тисяч годин до п’яти років (КМ1609РР1).

Флеш-пам’ять *(Flash*-пам’ять**).** Мікросхеми *Flash*-пам’яті вперше розробила фірма *Intel* 1988 року. Пам’ять нового типу енергонезалежна, з електричним стиранням і перепрограмуванням.

Елементи пам’яті мікросхеми *Flash*-пам’яті побудовано на одному МОН транзисторі з ПЗ, виконаним за особливою, запатентованою *Intel* технологією, названою *ETOX* (*EPROM Thin Oxide*). Напівпровідникова структура цього транзистора подібна до структури ЕП *EPROM* з перепрограмуванням та стиранням інформації ультрафіолетовим опроміненням. Він містить підкладку *p*-типу, на якій сформовано області *n*-стоку і витоку. Над проміжком між ними розміщено керувальний затвор (КЗ), відділений від підкладки шаром оксиду SiO_2. В останньому сформовано область з полікремнію, яка виконує функції другого – плавучого затвора.

У цій області може нагромаджуватися заряд електронів, електричне поле якого зміщає поріг відкривання транзистора. У результаті, якщо напруга на КЗ номінальна і в ПЗ немає заряду, то транзистор відкритий, а за наявності заряду – закритий. Ці два стани і використовують для запам'ятовування «1» та «0».

Завдяки використанню технології *ETOX* товщину шару SiO_2 між ПЗ і підкладкою транзистора вдалося зменшити порівняно з *EPROM* більш ніж утричі (до 100 Å). Унаслідок цього напруга, використовувана для запису інформації, знизилася до 12 В та з'явилася можливість електричного стирання, тобто видалення заряду з ПЗ, за рахунок тунельного ефекту при напрузі між стоком і КЗ, що дорівнює 12 В. Ці особливості дозволили забезпечити перезапис інформації у складі МПС й у багато разів збільшити кількість перезаписів.

Для організації нагромаджувача інформації в мікросхемах *Flash*-пам'яті ЕП розміщено у вигляді прямокутної матриці. У кожнім рядку КЗ транзисторів об'єднані й утворюють шини вибору слова, стоки в кожному стовпці також об'єднані й утворюють шини вибору розряду (біта), а об'єднані джерела – шини, що підключають до підсилювачів відтворення.

Така схема з'єднань нагромаджувача дуже критична до виконання стирання. Річ у тому, що під час стирання з ПЗ може бути вилучено більше електронів, ніж було забезпечено інжекцією під час програмування. У результаті в ПЗ з'явиться позитивний заряд, транзистор стане провідним незалежно від напруги на КЗ і відбудеться шунтування всього стовпця ЕП у матриці. Повернути таку мікросхему в працездатний стан уже не вдасться. Щоб уникнути цього небажаного явища, фахівці *Intel* розробили ряд заходів, що впливають на структуру й організацію роботи мікросхем *Flash*-пам'яті. До них належать:

– застосування спеціальних алгоритмів запису та стирання з контролем стану і завершенням процесу за результатами контролю;
– попереднє програмування в алгоритмі стирання, коли перед стиранням усі ЕП матриці встановлюються в стан 0;
– включення до складу мікросхеми регістра, який зберігає ідентифікатори фірми виготовлювача й типу мікросхеми, що дозволяє захистити елемент від помилок вибору алгоритму;
– вбудовування в мікросхеми кіл, що реалізують алгоритми стирання і запису. Це спрощує зовнішнє керування і захищає від помилок під час перезапису.

Можна виділити три групи мікросхем *Flash*-пам’яті:

– мікросхеми першого покоління, виконані у вигляді єдиного масиву (блоку) або інакше – що цілком стираються (*BULK-ERASE*). До цієї групи належать мікросхеми 28*F*256, 28*F*512, 28*F*010, 28*F*020;
– мікросхеми з поділом масиву пам’яті на блоки різного розміру і з різними рівнями захисту від випадкового стирання та запису, або мікросхеми зі стиранням блоків різного розміру (*BOOT-BLOCK*). До них належать 28*F*001*BX-T/B,* 28*F*002*BX-T/B*(*L*), 28*F*200*BX-T/B*(*L*), 2*BF*004*BX-T/B*(*L*), 28*F*400*BX-T/B*(*L*);
– мікросхеми третього покоління з найбільшим розміром масиву, розділеного на блоки однакового обсягу з незалежним стиранням (*FLASH-FILE*)*:* 28*F*008*SA*(*L*), 28*F*016*SA*, 28*F*032*SA*.

Мікросхеми *BULK-ERASE* можна використати замість традиційних схем *EPROM*. Їх основна перевага – можливість електричного стирання, але збереження енергетичної незалежності. Якщо під час використання звичайного ПЗП процес модифікації «зашитих» у ньому даних потребує тривалої процедури стирання, для чого мікросхему слід витягти з плати і піддати ультрафіолетовому опроміненню, то *Flash*-пам’ять можна перепрограмувати під керуванням процесора самої системи. Порівняно з *EEPROM*, що має ускладнену структуру ЕП, а отже, й обмеження на щільність їх розміщення на кристалі, *Flash*-пам’ять, яка використовує один транзистор на один ЕП, безсумнівно, виграє щодо щільності й собівартості.

Мікросхеми групи *BOOT-BLOCK* застосовують для зберігання *BIOS* у персональних комп’ютерах. Вони дозволяють об’єднати *BIOS*, який тепер можна обновити безпосередньо з дискети. Особливість *Flash*-пам’яті в тому, що її зміст не можна стерти, не подавши на спеціальний вхід мікросхеми напругу програмування плюс 12 В. Це і дозволяє зберегти важливу інформацію від випадкового або несанкціонованого стирання. Можливість електричного перепрограмування *Flash*-пам’яті істотно полегшує процес модернізації мікросхем *BIOS*.

Мікросхеми групи *FLASH-FILE* використовують для зберігання даних великого обсягу в так званих *Flash*-картах – альтернативі жорстким магнітним дискам. Оскільки обсяги виробництва мікросхем *Flash*-пам’яті незмінно зростають, очікується, що в недалекому майбутньому *Flash*-пам’ять замінить жорсткі магнітні диски в багатьох сферах застосування, наприклад у системах, які працюють в умовах сильних механічних впливів, у яких жорсткі диски не застосовують або вони швидко виходять з ладу. За часом доступу *Flash*-пам’ять у 125–250 разів «швидше» жорсткого диска, однак поступається поки йому щодо інформаційного обсягу. У *Flash*-карт, які випускають тепер, він не перевищує 40 Мбайт.

Кількість циклів стирання/запису мікросхем *Flash*-пам'яті – не менше 100 000. Для новітніх мікросхем 28*F*016*SA* і 28*F*032*SA* використовують технологію й алгоритми стирання/запису, що дозволяють поліпшити цю характеристику на порядок.

Мінімальний час читання байта/слова не перевищує 100 нс і більший у мікросхемах, які працюють при напрузі 3,3 В.

Час запису (байт/слово) становить приблизно 9 мкс, час стирання блоку (64 кбайт) – близько 1 с. У міру вироблення ресурсу в циклах стирання/запису змінюється структура оксиду між ПЗ і напівпровідником. У результаті збільшується кількість циклів, потрібна для стирання/запису інформації, тому витрати часу на ці операції можуть зрости в кілька разів. Інформаційна ємність мікросхем – від 256 кбіт до 32 Мбіт.

Напруга живлення мікросхем *Flash*-пам'яті – 5 В ± 10 %, стирання і програмування – 12 В ± 5 %. Випускають також мікросхеми, що працюють при напрузі 3,3±0,3 В (в умовній позначці є літера *L*). Для них характерна більша тривалість циклу читання байта. Мікросхеми 28*F*016*SA* і 28*F*032*SA* можуть працювати як при 5, так і при 3,3 В (робочу напругу встановлюють за рівнем напруги на відповідному виводі мікросхеми).

Споживаний струм істотно залежить від режиму роботи мікросхеми. Якщо звертань немає, то вона знаходиться в режимі чекання (*Standby*). Основну частину внутрішніх кіл у цьому разі вимкнуто, і споживаний струм значно менший, ніж в активному режимі. Під час стирання і запису споживаний струм зростає (порівняно з активним режимом) переважно в колі із джерелом плюс 12 В. Для елементів мікросхем, які стираються вроздріб, існує режим мікроспоживання *(Powerdown*), у якому струм, споживаний від джерел напруг 5 і 12 В, не перевищує частки мікроампера.

Більшість ВІС *Flash*-пам'яті характеризуються послідовним уведенням і виведенням даних по шині I^2C *(Inter Integrated Circuit Bus).*

Шина складається всього з двох двонапрямлених ліній: *SCL (Serial Clock)* і *SDA (Serial Data)* з відкритими колекторами, до яких можна підключати до 128 пристроїв, у варіанті з 10-розрядною адресою – до 1 024. Лінії навантажено резисторами, з'єднаними з джерелом живлення. Номінали резисторів і напругу живлення специфікація не встановлює. Треба лише, щоб струм короткого замикання кожної з ліній на загальний провід не перевищував 3 мА, а ємність з урахуванням підключених абонентів – 400 пф.

Один із пристроїв слугує ведучим (*master),* інші – веденими (*slave).* Перший генерує синхроімпульси *SCL* і керує всім обміном по шині. Ведені тільки у відповідь на запити ведучого приймають від нього або передають йому дані.

Типову структурну схему ВІС *Flash*-пам'яті з послідовним уведенням-виведенням інформації по шині I^2C зображено на рис. 5.10, а схему з'єдання кількох ВІС – на рис. 5.11.

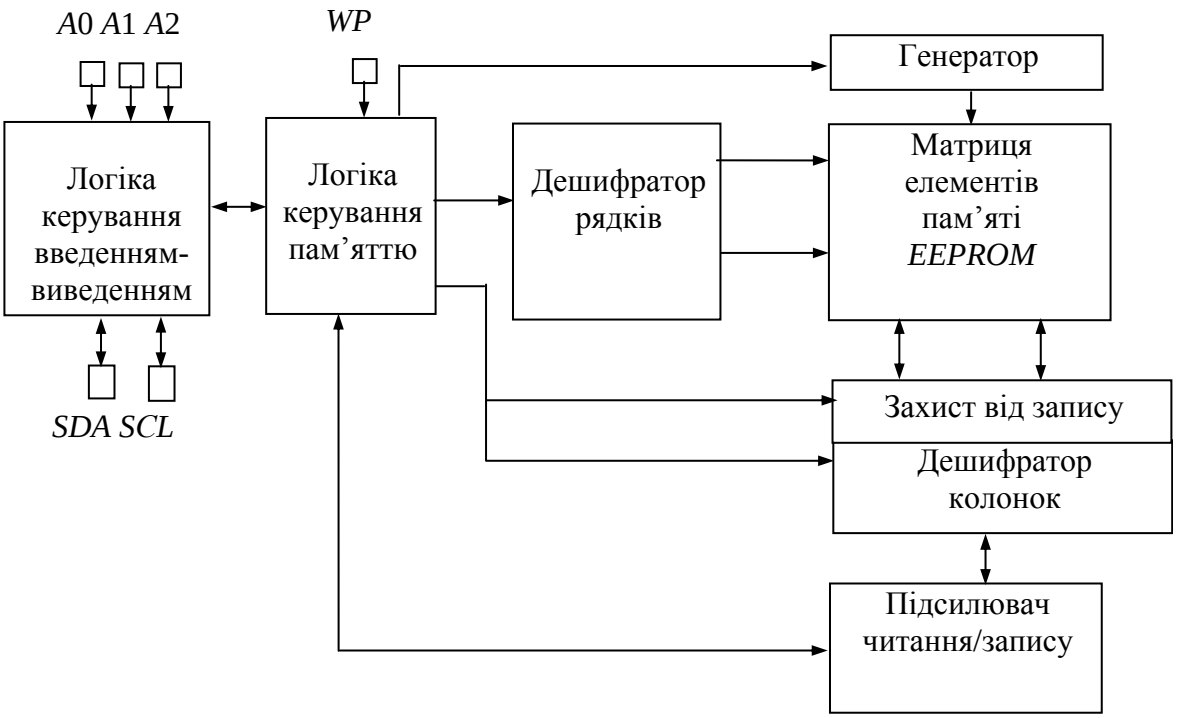


Рис. 5.10. Структурна схема ВІС *Flash*-пам'яті з послідовним уведенням-виведенням інформації по шині I^2C

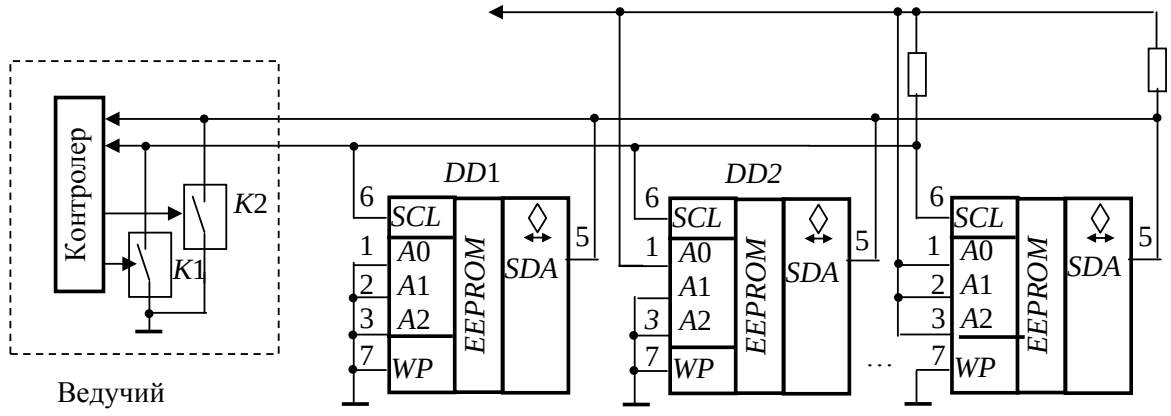


Рис. 5.11. Схема з'єднань мікросхем пам'яті з лініями шини I^2C

Загальне уявлення про порядок передачі інформації дають часові діаграми, показані на рис. 5.12. Вихідний стан шини – *H*-рівні на лініях *SDA* і *SCL*. Щоб почати сеанс обміну даними, ведучий, не змінюючи стану лінії *SCL*, установлює *L*-рівень на лінії *SDA*, а потім – такий самий рівень на лінії *SCL* (команда СТАРТ).

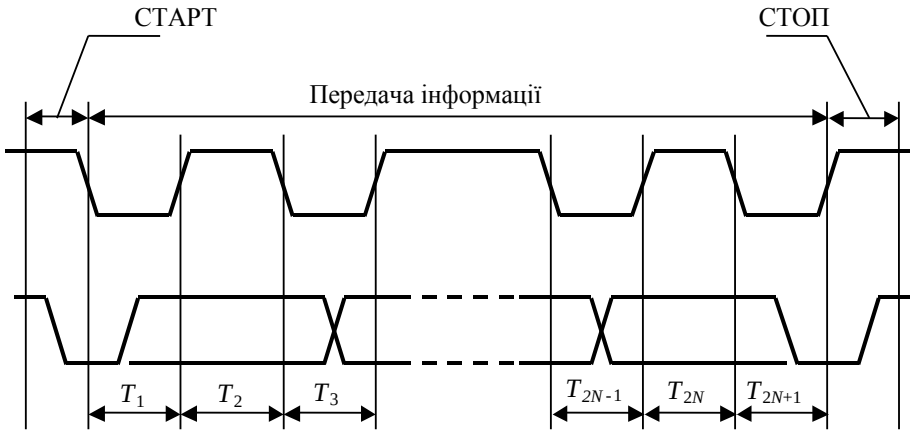


Рис. 5.12. Порядок передачі інформації шини I^2C: T_1, T_3, T_{2N+1} – рівень *SDA* дозволено міняти; T_2, T_{2N} – рівень *SDA* має бути стабільним

Передача інформації бітна. Для *L*-рівнія *SCL*, підтримуваного ведучим, передавач установлює на лінії *SDA* рівень, що відповідає значенню переданого біта (1 – *H*-рівень, 0 – *L*-рівень). Потім ведучий установлює на лінії *SCL H*-рівень, а після закінчення часу, відведеного приймачу для прийняття інформації, знову змінює його *L*. Ця процедура повторюється для кожного переданого біта. Ведучий завершує сеанс командою СТОП – зміною *L*-рівня на лінії *SDA* високим за такого самого рівня на лінії *SCL*.

У всіх ситуаціях, крім подачі команд СТАРТ і СТОП, не допускається зміна рівня на лінії *SDA* за високого рівня *SCL*. Не варто й одночасно змінювати стан цих ліній – часовий зсув має становити не менш 0,3 (0,1) мкс.

Частота повторення імпульсів *SCL* – не більше 100 (400) кГц, тривалість імпульсу або паузи – не менше 4,7 (1,2) мкс. Часові діаграми, зображені на рис. 5.13, показують, як відбувається передача байта. Їй передує команда СТАРТ або розглянутий далі біт підтвердження приймання попереднього байта.

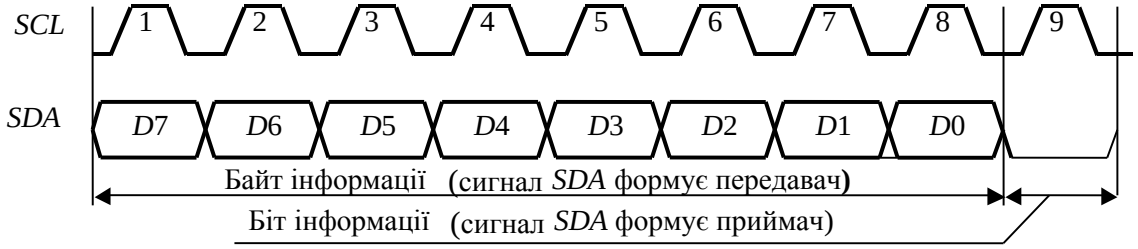


Рис. 5.13. Часові діаграми передачі інформації

Першим передають старший розряд байта, останнім – молодший, після чого пристрій, що прийняв байт, підтверджує цей факт, установивши на лінії *SDA* низький рівень і підтримуючи його до закінчення генерації ведучим на лінії *SCL* дев'ятого імпульсу високого рівня. Після підтвердження приймання чергового байта починається передача наступного або подається команда СТОП.

Сім старших бітів байта, обов'язково передані ведучим слідом за командою СТАРТ, являють собою адресу веденого, з яким він має намір

установити зв'язок. Якщо серед підключених до шини є справний пристрій з такою адресою, він має підтвердити приймання і підготував тися до подальших дій. Усі інші ведені, не впізнавши своєї адреси, відключаються до наступної команди СТАРТ.

Молодший восьмий біт першого байта – ознака напрямку передачі. Якщо його значення – логічний 0 (*L*-рівень), інформація в цьому сеансі буде записуватися/передаватися від ведучого до веденого, інакше перший має намір читати дані, передані веденим. Процес передачі всіх наступних байтів аналогічний описаному вище, їх кількість за сеанс не обмежено, однак зміни напрямку передачі до завершення сеансу не передбачено.

Логічними рівнями, поданими на входи *A*0–*A*2 (див. рис. 5.10) , зазвичай задають три молодші розряди адреси веденої мікросхеми шини I^2C. Старші чотири розряди адреси мікросхеми пам'яті завжди містять двійкову комбінацію 1010

Вхід *WP (Write Protect)* призначено для керування захистом записаних у мікросхему даних. Якщо цей вивід залишено вільним або з'єднано із загальним проводом, то можна змінювати вміст будь-яких комірок. За високого логічного рівня весь масив або його частину захищено від стирання і запису. Останнім часом випускають мікросхеми, у яких передбачено ділянки пам'яті, запис у яких можна заблокувати й іншими способами, наприклад командами ведучого.

Логічним рівнем на вході *MODE* (за його наявності) переключають режими запису даних. Якщо його залишено вільним або з'єднано із загальним проводом, то діє мультибайтний режим, інакше – сторінковий. Мікросхеми, що не мають названого входу, працюють тільки у сторінковому режимі запису.

Кількість мікросхем пам'яті з лініями шини I^2C у схемі, зображеній на рис. 5.11, може бути від однієї до восьми, причому комбінації логічних рівнів на цих входах *A*0–*A*2 мають бути різними для кожної ВІС. Зазвичай у ролі ведучого виступає мікроконтролер або спеціалізований пристрій. Ключами К1 і К2 слугують транзистори вихідних каскадів двонапрямленого порту. Кожна з ліній шини I^2C займає всього один розряд такого порту. Інакше для неї доводиться витрачати по одному розряду звичайних однонапрямлених портів введення-виведення, доповнюючи їх ключами на біполярних або польових транзисторах чи логічними елементами з відкритим колектором (стоком).

Параметри мікросхем пам'яті з інтерфейсом I^2C, які випускають фірми *Atmel, Fairchild, Philips, Seiko, ST*, наведено в табл. 5.7. Усі вони – ПЗП багаторазового репрограмування з електричним стиранням і записом даних. Напруга живлення мікросхем – від 1,8–4,5 (залежно від типу і буквеного індексу в позначенні) до 5,5 В. Найчастіше зі зменшенням напруги живлення знижуються споживаний струм і швидкодія. Навантажувальна здатність виходу *SDA* – 3–5 мА, вхідний струм і ємність будь-якого входу не перевищують відповідно 1 мкА і 4–8 пф.

Таблиця 5.7. **Параметри мікросхем пам'яті з інтерфейсом I^2C**

Тип ВІС	Ємність, кбіт	Сторінка запису,байт	Захист від запису	Максимальна кількість ВІС на шині	Формат адреси веденого							Термін зберігання даних, роки	Напруга живлення, В	Споживання струму в режимі			Час запису, мс
					*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1			спокою, мкс	читання, мА	запису, мА	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
*AT*24*C*01	1	8	+	1	1	0	1	0	*x*	*x*	*x*	100	1,8–5,5	3–18	1	3	10
*AT*24*C*01*A*	1	8	+	8	1	0	1	0	*A*2	*A*1	*A*0	100	1,8–5,5	3–18	1	3	10
*AT*24*C*21	1	8	+	1	1	0	1	0	*x*	*x*	*x*	100	2,5–5,5	4–30	1	3	10
*M*24*C*01	1	16	+	8	1	0	1	0	*A*2	*A*1	*A*0	40	2,5–5,5	0,1–1	0,8–2	0,8–2	5–10
S-24*C*01*D*	1	8	+	1	1	0	1	0	*x*	*x*	*x*	10	2,5–5,5	1	0,3–0,8	0,3–0,8	10
*AT*24*C*02	2	8	+	8	1	0	1	0	*A*2	*A*1	*A*0	100	1,8–5,5	3–18	1	3	10
*AT*24*C*02*A*	2	8	+*	8	1	0	1	0	*A*2	*A*1	*A*0	100	1,8–5,5	3–18	1	3	10
*AT*34*C*02	2	16	+	8	1	0	1	0	*A*2	*A*1	*A*0	100	1,8–5,5	3–18	1	3	10
*M*24*C*02*U*	2	16	+	8	1	0	1	0	*A*2	*A*1	*A*0	40	2,5–5,5	0,1–1	0,8-2	0,8–2	5–10
*NM*24*C*02*U*	2	16	–	8	1	0	1	0	*A*2	*A*1	*A*0	40	2,7–5,5	50	1	1	15
*NM*24*C*03*U*	2	16	+*	8	1	0	1	0	*A*2	*A*1	*A*0	40	2,7–5,5	50	1	1	15
*PCF*8522*E*	2	4	+	8	1	0	1	0	*A*2	*A*1	*A*0	10	2,7–5,5	2	1–2	1–2	10–25
*PCF*8582*E*	2	8	–	8	1	0	1	0	*A*2	*A*1	*A*0	40	4,5–6	10	0,2	2	7
S-24*C*02*B*	2	8	+*	1	1	0	1	0	*X*	*x*	*x*	10	2–5,5	1	0,3–0,8	0,3–0,8	10
*ST*14*C*02*C*	2	8	–	1	1	0	1	0	0	0	0	10	3–5,5	100	2	2	10
*AT*24*C*04	4	16	+	4	1	0	1	0	*A*2	*A*1	*B*8	100	1,8–5,5	3–18	1	3	10
*AT*24*C*04*A*	4	16	+*	4	1	0	1	0	*A*2	*A*1	*B*8	100	1,8–5,5	3–18	1	3	10
S-24*C*04*B*	4	16	+*	1	1	0	1	0	*x*	*x*	*B*8	10	1,8–5,5	0,1–1	0,8–2	0,8–2	10
*AT*24*C*08	8	16	–	2	1	0	1	0	*A*2	*B*9	*B*8	100	1,8–5,5	3–18	1	3	10
*AT*24*C*08*A*	8	16	+	2	1	0	1	0	*A*2	*B*9	*B*8	100	1,8–5,5	3–18	1	3	10
*M*24*C*08	8	16	+	2	1	0	1	0	*A*2	*B*9	*B*8	40	2,5–5,5	0,1–1	0,8–2	0,8–2	5–10

Продовження табл. 5.7

1	2	3	4	5	6	7	8	9	*10*	*11*	*12*	13	14	15	16	17	18
*AT*24*C*16	16	16	+*	1	1	0	1	0	*B*10	*B*9	*B*8	100	1,8-5,5	3–18	1	3	10
*AT*24*C*164	16	16	+	8	1	*A*2	*A*1	*A*0	*B*10	*B*9	*B*8	100	1,8–5,5	3–18	1	3	10
*M*24*C*16	16	16	+	1	1	0	1	0	*A*2	*B*9	*B*8	40	2,5–5,5	0,1–1	0,8–2	0,8–2	5–10
*ST*24*C*16	16	16	–	1	1	0	1	0	*B*10	*B*9	*B*8	40	4,5–5,5	100–300	2	2	10
*ST*24*E*16	16	16	+	8	1	0	1	0	*A*2	*A*1	*A*0	40	4,5–5,5	100–300	2	2	10
*ST*24*W*16	16	16	+	1	1	0	1	0	*B*10	*B*9	*B*8	40	4,5–5,5	100–300	2	2	10
*ST*25*C*16	16	16	–	1	1	0	1	0	*B*10	*B*9	*B*8	40	4,5–5,5	5–50	1	1	10
*ST*25*E*16	16	16	+	1	1	0	1	0	*A*2	*A*1	*A*0	40	4,5–5,5	5–50	1	1	10
*ST*25*W*16	16	16	+	1	1	0	1	0	*B*10	*B*9	*B*8	40	4,5–5,5	5–50	1	1	10
*AT*24*C*32	32	32	+**	8	1	0	1	0	*A*2	*A*1	*A*0	100	1,8–5,5	0,1–2	1	3	10
*AT*24*C*64	64	32	+**	8	1	0	1	0	*A*2	*A*1	*A*0	40	1,8–5,5	0,1–2	1	3	10
*NM*24*C*65	64	32	+*	8	1	0	1	0	*A*2	*A*1	*A*0	40	2,7–5,5	1–10	1	1	15
*AT*24*CS*128	128	64	+	4	1	0	1	0	*0*	*A*1	*A*0	40	1,8–5,5	0,2–5	2	3	10
*M*14128	128	64	+	1	1	0	1	0	0	0	0	40	2,5–5,5	2–20	1	2	10
*AT*24*CS*256	256	64	+	4	1	0	1	0	0	*A*1	*A*0	40	1,8–5,5	0,2–5	2	3	10
*M*14256	256	64	+	1	1	0	1	0	0	0	0	40	2,5–5,5	2–20	1	2	5
*AT*24*C*512	512	128	+	4	1	0	1	0	0	*A*1	*A*0	40	1,8–5,5	0,2–5	2	3	10
*AT*24*C*512*SC*	512	128	–	1	1	0	1	0	0	0	0	40	2,7–5,5	0,6–8	2	3	10

* Захист має лише старша половина масиву пам’яті.

** Захист має лише старша чверть масиву пам’яті.

Примітка. Максимальна частота *SCL* дорівнює:

100 кГц для мікросхем *AT*24*C*21, *PCF*8522*E*, *PCF*85822*E*, *ST*14*C*02*C*, *ST*24*C*16, *ST*24*W*16, *ST*25*C*16, *ST*25*W*1;

1000 кГц для мікросхем *AT*24*CS*128, *AT*24*CS*256, *AT*24*C*512, *AT*24*C*512*SC*;

400 кГц для решти мікросхем, наведених у таблиці.

Кількість циклів перезапису:

10^5 – для мікросхем *PCF*85822*E*, *AT*24*CS*128, *M*14128, *AT*24*CS*256, *M*14256, *AT*24*C*512, *AT*24*C*512*SC*;

$5 \cdot 10^5$ – для мікросхем *PCF*85882*E,*

10^6 – для решти мікросхем.

Контрольні запитання

1. У чому призначення ПЗП?
2. Як організовані схеми ПЗП і на яких ЕП їх виконують?
3. Як можна задавати 0 і 1 у матричних ПЗП?
4. У чому розходження технології виготовлення ППЗП з плавкими перемичками та із замиканням перемичок?
5. Як програмуються одноразово програмовні ПЗП?
6. На яких елементах виконують РПЗП?
7. Поясніть принцип роботи ЕП РПЗП.

5.3. Побудова модулів постійних запам'ятовувальних пристроїв

Розглянемо побудову модуля ПЗП МПС на основі 8-розрядних процесорів. Схему (рис. 5.14) модуль ПЗП має у тому разі, коли розрядність шини даних процесора збігається з розрядністю шини даних ПЗП, а інформаційна ємність ПЗП достатня для зберігання інформації.

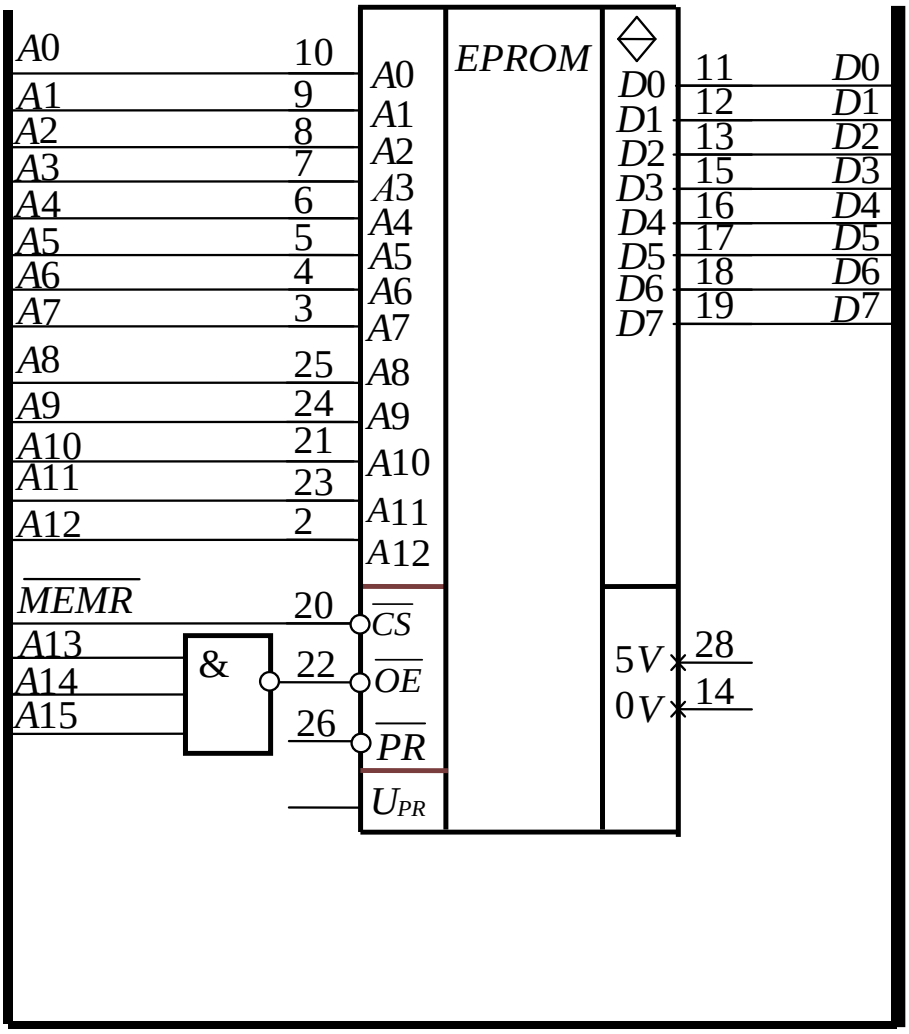


Рис. 5.14. Модуль ПЗП ємністю 8 К×8

Задачу *нарощування ємності* ПЗП розглядають у тих випадках, коли потрібна ємність модуля пам'яті перевищує ємність однієї ВІС ПЗП.

Приклад 5.1. Визначити інформаційну ємність, початкову та кінцеву адреси модуля пам'яті МПС 8-розрядного МП. Модуль складається з однієї ВІС К573РФ6, яку з'єднано із системною шиною згідно з рис. 5.14.

Оскільки ВІС має 13 адресних входів і 8 виходів даних, її інформаційна ємність $2^{13} - 8 = 8$ К×8.

Для визначення початкової та кінцевої адрес модуля пам'яті зазначимо, що зчитування інформації з ПЗП буде відбуватися за одночасної дії сигналів $\overline{CS} = 0$ і $\overline{OE} = 0$, при цьому буде зчитано вміст комірки з адресою, поданою на входи *A*12–*A*0. Сигнал $\overline{CS} = 0$ тоді, коли виконується цикл шини ЗЧИТУВАННЯ ПАМ'ЯТІ (тобто $\overline{MEMR} = 0$). Сигнал $\overline{OE} = 0$ у діапазоні адрес з одиничними значеннями розрядів *A*13, *A*14, *A*15. Отже, початкову та кінцеву адреси модуля пам'яті визначають таким чином:

$1110\ 0000\ 0000\ 0000_2 = 0E000H$ – початкова;
$1111\ 1111\ 1111\ 1111_2 = 0FFFFH$ – кінцева.

Приклад 5.2. Розробити схему модуля ПЗП з інформаційною ємністю 32 К×8 та початковою адресою 8 000*H* на основі ВІС К573РФ6. Модуль ПЗП з'єднати із системною шиною 8-розрядного МП. Вибірку окремих ВІС здійснити за допомогою дешифратора.

Для забезпечення інформаційної ємності 32 К×8 схема модуля ПЗП має містити чотири ВІС ПЗП ємністю 8 К×8 кожна (рис. 5.15). Оскільки модуль пам'яті містить чотири ВІС ПЗП, для вибірки кожної з них потрібен дешифратор *DC* із чотирма виходами *a, b, c, d*. Щоб початкова адреса модуля ПЗП дорівнювала 8 000*H*, треба забезпечити вибірку даних із модуля при одиничному значенні адресного розряду *A*15 (8 000*H* = $1000\ 0000\ 0000\ 0000_2$). Для нульового значення розряду *A*15 вибірка не виконується, тому значення вихідних сигналів дешифратора мають бути одиничними: $a = b = c = d = 1$. Значення розряду *A*15 надходить на вхід дозволу дешифратора *E*. Під час подання на вхід *E* нульового значення *A*15, не обирається жодна з ВІС ПЗП. Для адресації чотирьох ВІС ПЗП при фіксованому значенні старшого адресного розряду *A*15 потрібно використовувати ще два адресні розряди. Для цього використовують розряди *A*14 та *A*13, які надходять на адресні входи дешифратора *A*1 та *A*0. У таблиці відповідності (див. табл. 5.1) для дешифратора *DC*, що відповідає таким умовам, наведено значення вихідних сигналів *a, b, c, d*, які надходять на входи $\overline{CS}$ чотирьох ВІС для вибірки відповідної ВІС, починаючи з адреси 8 000*H*. Символом *x* у табл. 5.8 позначено будь-яке значення вхідного сигналу: 0 або 1.

Молодші 13 розрядів шини адреси (*AB*) подаються на адресні входи *A*12–*A*0 всіх ВІС ПЗП паралельно адресують комірку всередині однієї ВІС, а два старші розряди *A*14 та *A*13 обирають одну з ВІС ПЗП. Виходи ВІС *D*0–*D*7 з'єднані із шиною даних (*DB*) МПС. Таким чином нарощується ємність.

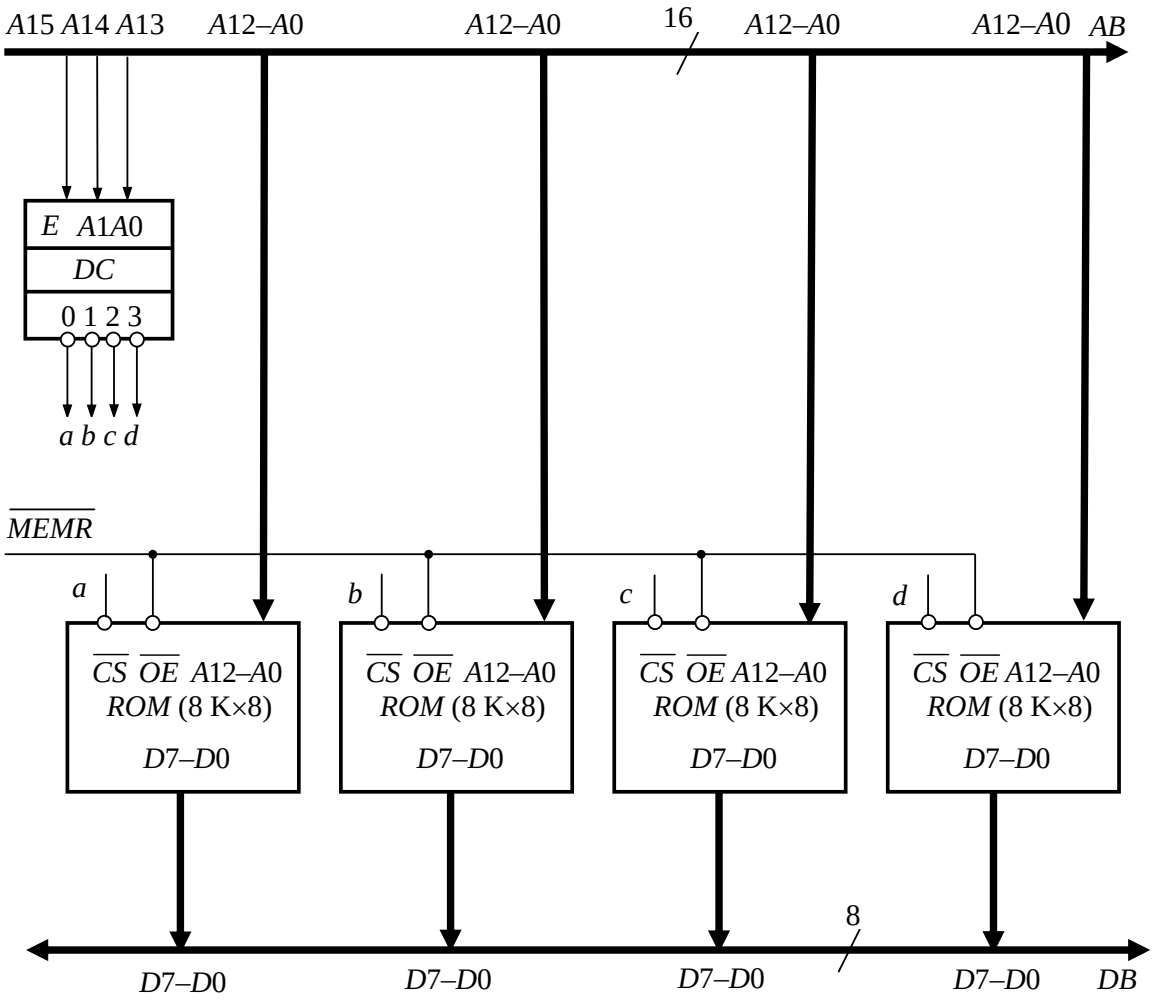


Рис. 5.15. Модуль ПЗП ємністю 32 К×8

Таблиця 5.8. **Виходи дешифратора**

*A*15	*A*14	*A*13	*a*	*b*	*c*	*d*
0	*x*	*x*	1	1	1	1
1	0	0	0	1	1	1
1	0	1	1	0	1	1
1	1	0	1	1	0	1
1	1	1	1	1	1	0

Розглянемо побудову модуля постійної пам’яті для МПС на базі 16-розрядних процесорів, які можуть оперувати як з 8-, так і з 16-розрядними комірками пам’яті. Для використання 8-розрядних ВІС у модулях пам’яті 16-розрядних процесорів, наприклад, з інформаційною ємністю 1М×8, постійну пам’ять виконують у вигляді двох *банків* по 512 кбайт кожний. Один із банків з’єднується з молодшою половиною шини даних, тобто до розрядів *D*7–*D*0, і називається *молодшим*, другий – до старшої половини шини даних (розряди *D*15–*D*8) і називається *старшим*. Молодший банк

містить байти з парними адресами ($A0 = 0$), старший – з непарними ($A0 = 1$). Для адресації байта всередині банку використовують адресні розряди $A19$–$A1$. Зчитування з ПЗП організовано таким чином, що при зверненні до ПЗП на шину даних МП завжди надходять два байти, тобто зчитується вміст обох банків одночасно. У разі потреби процесор може обирати один належний байт з двох. Систему пам'яті, виконану у вигляді двох банків, показано на рис. 5.16. Кожний з банків виконано за структурною схемою модуля ПЗП для 8-розрядних процесорів, розглянутих вище.

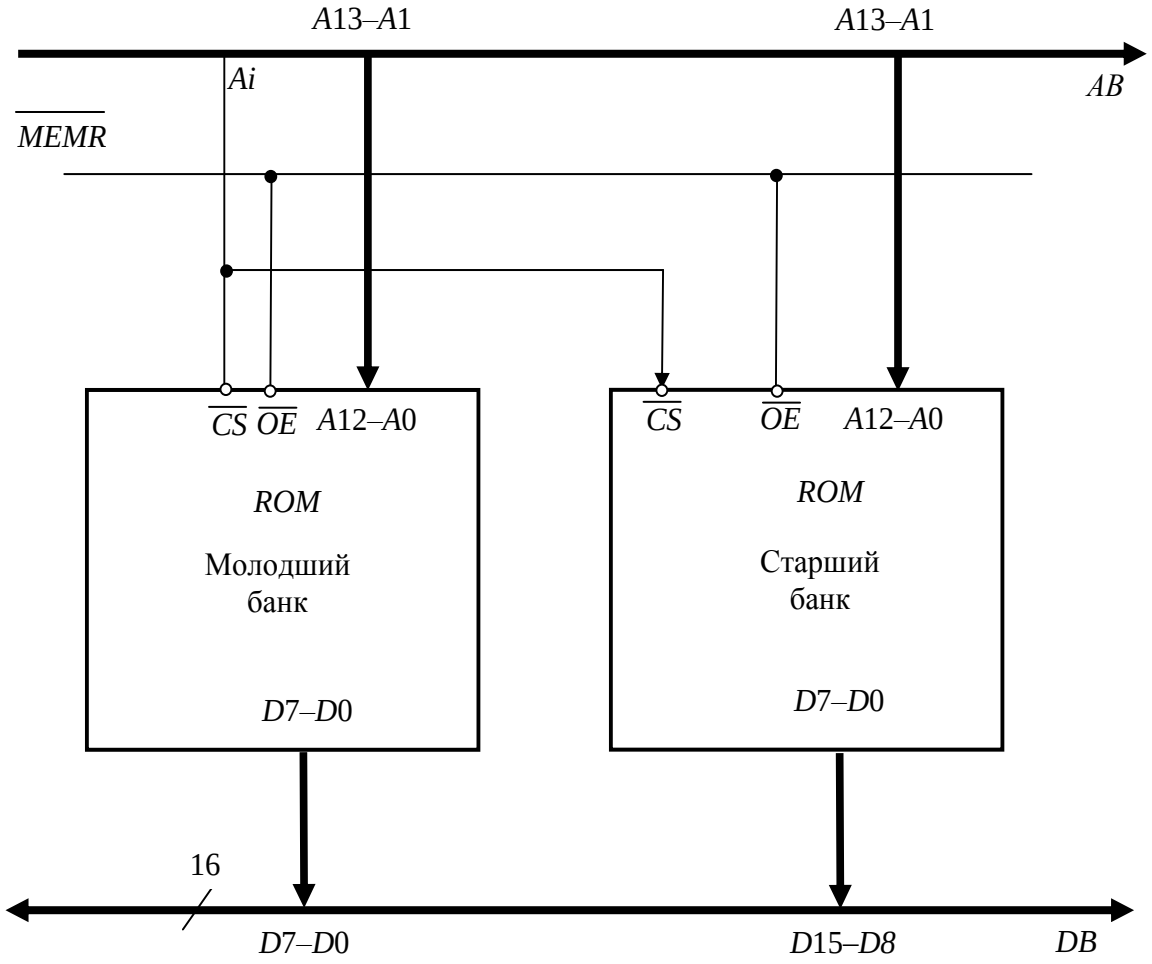


Рис. 5.16. Модуль ПЗП у 16-розрядних МПС

У МПС з 32-розрядною шиною даних модуль ПЗП виконується у вигляді чотирьох банків. Інформація зчитується одночасно з усіх чотирьох банків, після чого МП обирає одно-, дво- або чотирибайтове слово залежно від виконуваної команди.

Контрольні запитання

1. Для чого призначений ПЗП?
2. Наведіть приклад нарощування ємності ПЗП удвічі?
3. Поясніть поняття банку пам’яті.
4. Наведіть приклад побудови модуля ПЗП у 16-розрядній МПС на базі 8-розрядних ВІС ПЗП.

5.4. Оперативні запам’ятовувальні пристрої статичного типу

Оперативні запам’ятовувальні пристрої використовуються для зберігання інформації, одержуваної в процесі роботи МПС. Вони можуть бути як *статичними,* так і *динамічними*. У статичних ОЗП записана інформація постійно зберігається у виділеному для неї місці і не руйнується під час її зчитування. Руйнування інформації можливе тільки під час її примусового стирання або вимкнення напруги джерела живлення. У динамічних ОЗП інформація постійно циркулює в масиві, відведеному для її зберігання. При цьому зчитування інформації супроводжується її руйнуванням. Для зберігання інформації її треба перезаписати заново.

Для позначення на принципових електричних схемах ВІС ОЗП використовують скорочення *RAM (random access memory).*

Статичні ОЗП. Елементом пам’яті ОЗП статичного типу є тригер. Для зберігання інформації в тригері потрібне джерело живлення, тобто тригер розглянутого типу енергетично залежний. За наявності живлення тригер здатний зберігати свій стан як завгодно довго. В один із двох станів, у яких може перебувати тригер, його приводять сигнали, що надходять по лініях шини даних у режимі запису.

Розрізняють ОЗП з *однорозрядною* та *багаторозрядною* організацією.

Для ОЗП з однорозрядною організацією інформаційна ємність дорівнює $2^m \times 1$ біт, де m – кількість адресних ліній ВІС ОЗП, з багаторозрядною – $2^m \times n$ біт, де n – кількість ліній даних.

Структурну схему ОЗП з однорозрядною організацією показано на рис. 5.17.

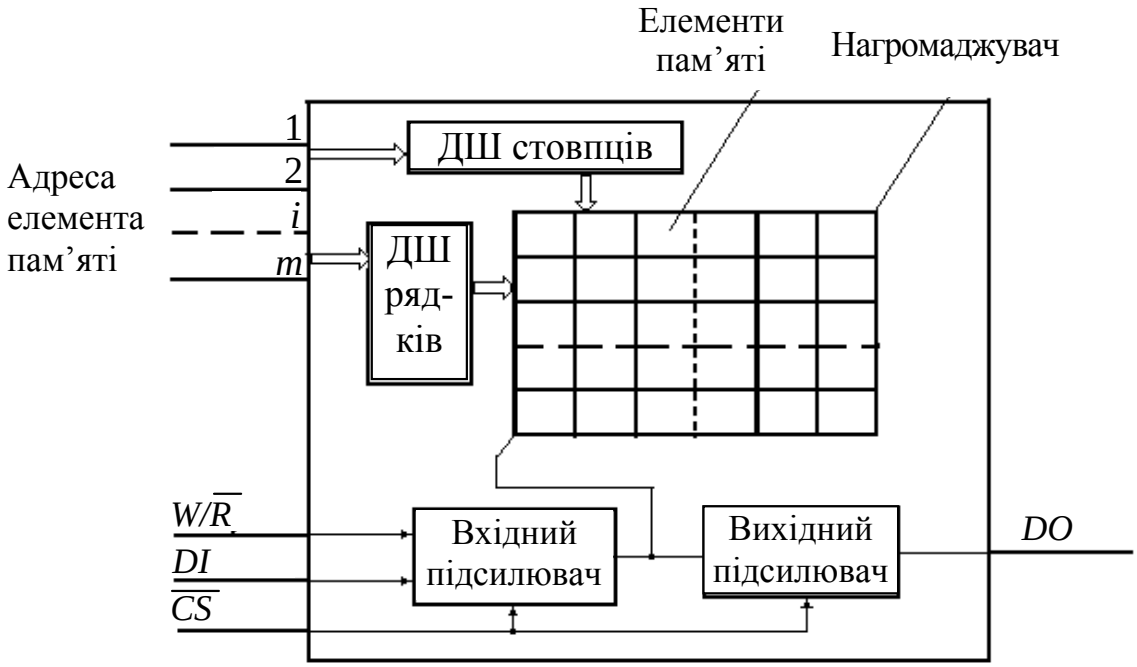


Рис. 5.17. Структурна схема ОЗП статичного типу з інформаційною ємністю $N\times1$

Кожний елемент пам’яті має свою адресу. Для того щоб зчитати інформацію з елемента пам’яті або записати в нього нове значення, його слід обрати поданням на ВІС ОЗП відповідної адреси A_{m-1}–A_0. Частина адресних розрядів надходить на дешифратор рядків, а частина – на дешифратор (ДШ) стовпців. Таким чином визначають положення елемента пам’яті в матриці нагромаджувача.

Щоб ввести інформацію в ОЗП, використовують вхід *DI* (*Data Input*), а для виведення – вихід *DO* (*Data Output*) і вихідний підсилювач. Вхідний та вихідний підсилювачі можна виконати як один підсилювач читання/запису. Керують режимами (запису, зчитування, зберігання) за допомогою сигналів $\overline{CS}$ і $W/\overline{R}$. Одиничний стан сигналу $W/\overline{R}$ визначає режим запису біта інформації в елемент пам’яті, а нульовий – режим зчитування біта інформації з елемента пам’яті.

Схематичне зображення мікросхем ОЗП КР541РУ3 з інформаційною ємністю $N\times1$ ілюструє рис. 5.18.

Роботу мікросхеми статичного ОЗП в динамічному режимі ілюструють часові діаграми, показані на рис. 5.19, які відображують послідовність подачі адресних і керувальних сигналів під час запису і зчитування, а також часові інтервали між різними сигналами і

тривалостями сигналів: час циклу запису/зчитування ($t_{ц.зп/зч}$), тривалість сигналів $\overline{CS}$ (вибору мікросхеми) $\bar{\tau}_{вм}$ і паузи між ними $\tau_{вм}$, час установлення сигналу $\overline{CS}$ щодо адреси $t_{ус.вм.а}$,час збереження адреси після сигналу $\overline{CS}$ $t_{зб.а.вм}$, час вибірки адреси $t_{в.а}$ чи час вибору мікросхеми $t_{в.\,вм}$.

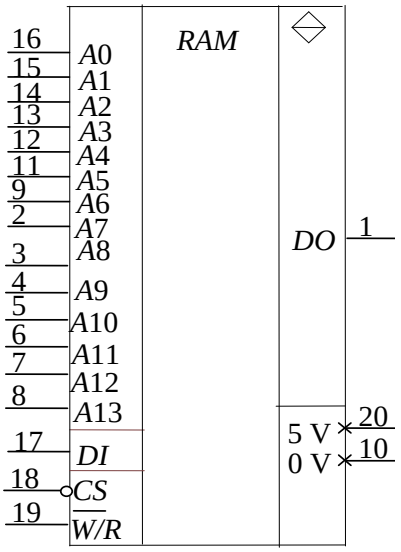


Рис. 5.18. ВІС ОЗП КР541РУ3

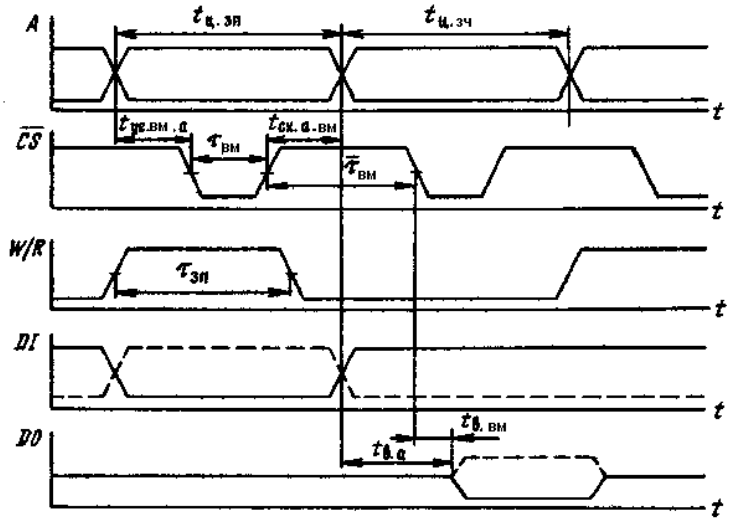


Рис. 5.19. Часові діаграми мікросхеми статичного ОЗП

Великі інтегральні схеми ОЗП з багаторозрядною організацією $2^m \times n$ мають n інформаційних входів/виходів DIO_{n-1}–DIO_0. Структурну схему ВІС ОЗП статичного типу з інформаційною ємністю $2^m \times 8$ показано на рис. 5.20. Вона має 8 входів/виходів даних DIO_7–DIO_0. Такий ОЗП припускає читання/запис 8-розрядного коду. Для запису нуля або одиниці, що надходять на входи DIO_7–DIO_0, потрібно на адресні входи подати код

адреси A_{m-1}–A_0, а на входи блока керування – сигнал дозволу *CS* та сигнал $\overline{W}/R$. Зчитування інформації відбувається аналогічно, але значення сигналу $\overline{W}/R$ протилежне. У більшості мікросхем ОЗП статичного типу підсилювач введення-виведення містить ключовий підсилювач-формувач, що може перебувати у трьох станах, два з яких відповідають нулеві або одиниці, а третій – високоімпедансний (*z*-стан). У *z*-стані не відбувається обміну інформацією з ОЗП. Перехід у *z*-стан визначається сигналом $\overline{OE}$.

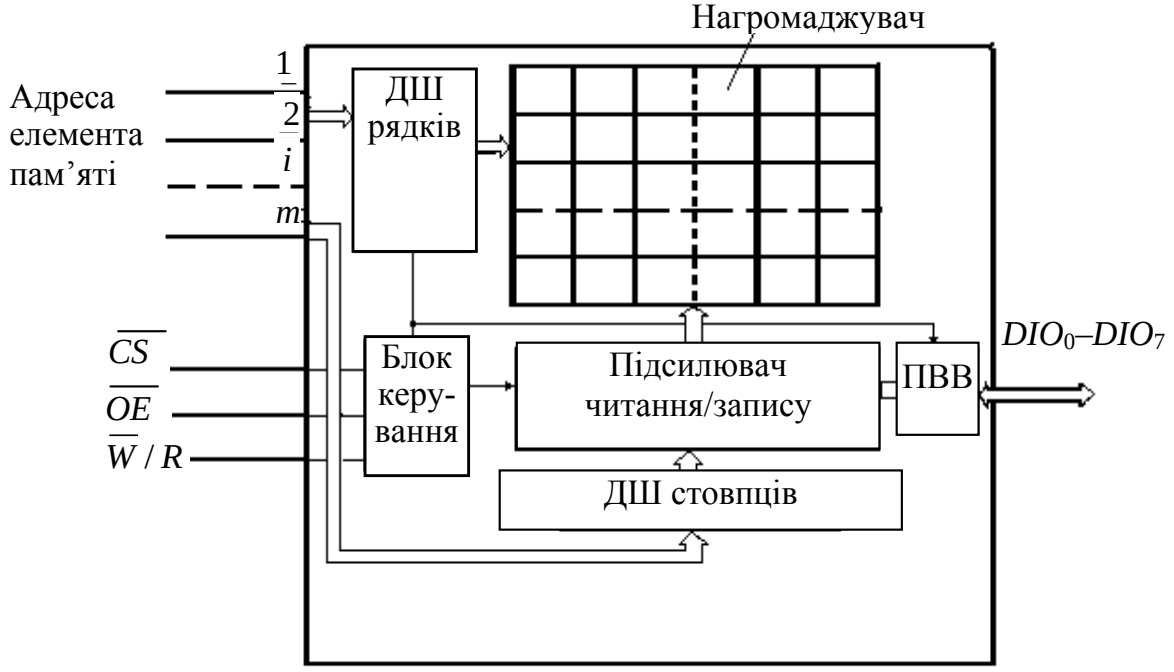


Рис. 5.20. Структурна схема ОЗП статичного типу з інформаційною ємністю *N*×8

Схематичне зображення мікросхем ОЗП з інформаційною ємністю *N*×8 ілюструє рис. 5.21.

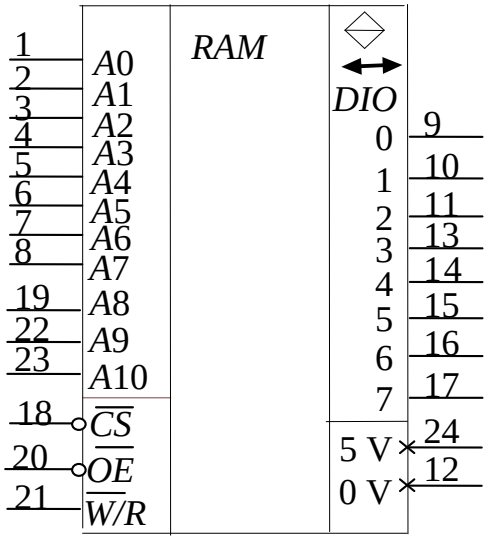


Рис. 5.21. ВІС ОЗП Р537РУ10

Технічні характеристики ОЗП статичного типу подано в табл. 5.9.

Таблиця 5.9. **Характеристика серій мікросхем статичних ОЗП**

Серія	Ємність, біт	$t_{ц.зп/зч}$, нс	$U_{живл}$, В	$P_{сп}$, Вт	Технологія
К500	16×4, 64×4 1 К×1, 4 К×1	40	–5,2	0,6–1,1	ЕЗЛ
К1500	64×4, 1 К×1 4 К×1, 16 К×1	9–45	–4,5	0,6–1,1	ЕЗЛ
К541	4 К×1, 1 К×4 8 К×1, 16 К×1	100–170	5	0,3–0,5	ІІЛ-ТТЛ
К132	1 К×1, 1 К×4 4 К×1, 16К×1, 64 К×1	55–85	5	0,4–0,9	*n*-МДН
К537	1 К×1, 4 К×1 2 К×8, 1 К×4	110–500	5	0,02–0,2	КМДН
К134	1 К×1	1000	5	0,6	ІІЛ-ТТЛ
К185	64×1, 1 К×1 256×1	300–500	5	0,4	ІІЛ-ТТЛ
К581	2 К×8, 4 К×4	120–200	5	0,4	КМДН
К155	256×1, 1 К×1	45–90	5	0,9	ТТЛ-ЕЗЛ
К561	256×1	800	6–12	0,15	КМДН
К176	256×1	900	9	0.02	КМДН
К565	1 К×1	450	5	0,4	*n*-МДН

Серед ВІС (табл. 5.9) найбільш швидкодійні мікросхеми пам’яті, виконані за технологіями ЕЗЛ (К500, К1500), *n*-МДН (К132), ТТЛ (К155) з часом циклу звертання від 9 до 90 нс. Мікросхеми за технологією КМДН (К537, К581, К561, К176) мають порівняно невелику швидкодію $t_{ц.зп/зч}$ = = 100–800 нс, але характеризуються істотно меншим рівнем споживаної потужності. Ця особливість КМДН-мікросхем зумовлює їх перспективність для застосування в пристроях з істотно обмеженим енергоресурсом, а також для побудови енергонезалежних ОЗП.

Найрозвиненішим функціональним складом із серій КМДН-мікросхем характеризується серія К537. Докладніші відомості про неї наведено в табл. 5.10.

Таблиця 5.10. **Динамічні параметри мікросхем серії К537 у діапазоні температур – 10...+70 °С**

Тип мікросхеми	$t_{ц.зп/зч}$, нс не менше	$t_{в. вм}$, нс не більше	$t_{ус. вм.а}$, нс не менше	$\tau_{вм}$, нс не менше	$\bar{\tau}_{вм}$, нс не менше	$t_{зб.а.вм}$, нс не менше
1	2	3	4	5	6	7
РУ1А	1300	900	200	900	400	200
РУ1Б	2000	1400	300	1400	600	300

Продовження табл. 5.10

1	2	3	4	5	6	7
РУ1В	4000	2800	600	2800	1200	600
РУ2А	500	390	20	390	110	90
РУ2Б	670	560	20	560	–	90
РУ3А	290	300	40	230	60	20
РУ3Б	210	200	40	150	60	20
РУ6А	340	220	20	220	120	100
РУ6Б	530	400	20	400	130	110
РУ8А	350	200	70	220	130	60
РУ8Б	530	400	70	400	130	60
РУ9А	400	220	20	220	180	160
РУ9Б	580	400	20	400	180	160
РУ10	180	170	–	300	–	–
РУ13	200	200	–	–	–	–
РУ14А	110	110	25	70	–	25
РУ14Б	180	180	35	130	–	25

Функціональний ряд серії – це більше 15 типів мікросхем, що різняться інформаційною ємністю (від 1 024 до 16 384 біт), організацією (одно- та багаторозрядною), швидкодією (більш ніж у 5 разів), рівнем споживаної потужності.

Загальні властивості мікросхем серії К537: єдина напруга живлення 5 В, ТТЛ-рівні вхідних і вихідних сигналів, вихід із трьома станами, висока завадостійкість, припустима значна ємність навантаження (200 пф і більше), невелике енергоспоживання, причому при збереженні майже на три порядки менше, ніж при звертанні, здатність зберігати записану інформацію за зниженої до 2–3 В напруги живлення.

Контрольні запитання

1. Що являє собою ЕП статичного ОЗП?
2. Що таке однорозрядна та багаторозрядна організації матриці нагромаджувача ОЗП?
3. Назвіть основні параметри ОЗП статичного типу.
4. Як співвідносяться значення потужності у режимах читання/запису та зберігання інформації?

5.5. Побудова модулів оперативного запам’ятовувального пристрою статичного типу

Розглянемо побудову модуля ОЗП статичного типу для МПС з 8-розрядними процесорами, використовуючи ВІС ОЗП з інформаційною ємністю $N\times1$. Для цього потрібно нарощувати розрядність даних.

Приклад 5.3. Для МПС з 8-розрядним МП розробити схему модуля ОЗП з інформаційною ємністю 4 К×8 на основі ВІС К537РУ14 з інформаційною ємністю 4 К×1.

Для побудови модуля пам'яті ємністю 4 К×8 потрібно вісім ВІС ОЗП К537РУ14 (рис. 5.22). Оскільки такі ВІС не мають входу сигналу *OE*, тобто, записуючи інформацію, вони не переходять у *z*-стан, слід використовувати шинний формувач. У цьому разі використаємо шинний формувач КР580ВА86, що з'єднує вихід ВІС із шиною даних під час виконання циклу ЧИТАННЯ ПАМ'ЯТІ, $\overline{MEMR} = 0$ та нульовому рівні сигналів на входах $\overline{W}/R$ і $\overline{CS}$. Кожну з ВІС ОЗП з'єднано з однією із ліній шини даних *D*7–*D*0.

Для мікросхем ОЗП, у яких під час записування інформації виходи переходять у високоімпедансний стан, шинний формувач не використовується, і виходи ВІС безпосередньо з'єднуються із шиною даних.

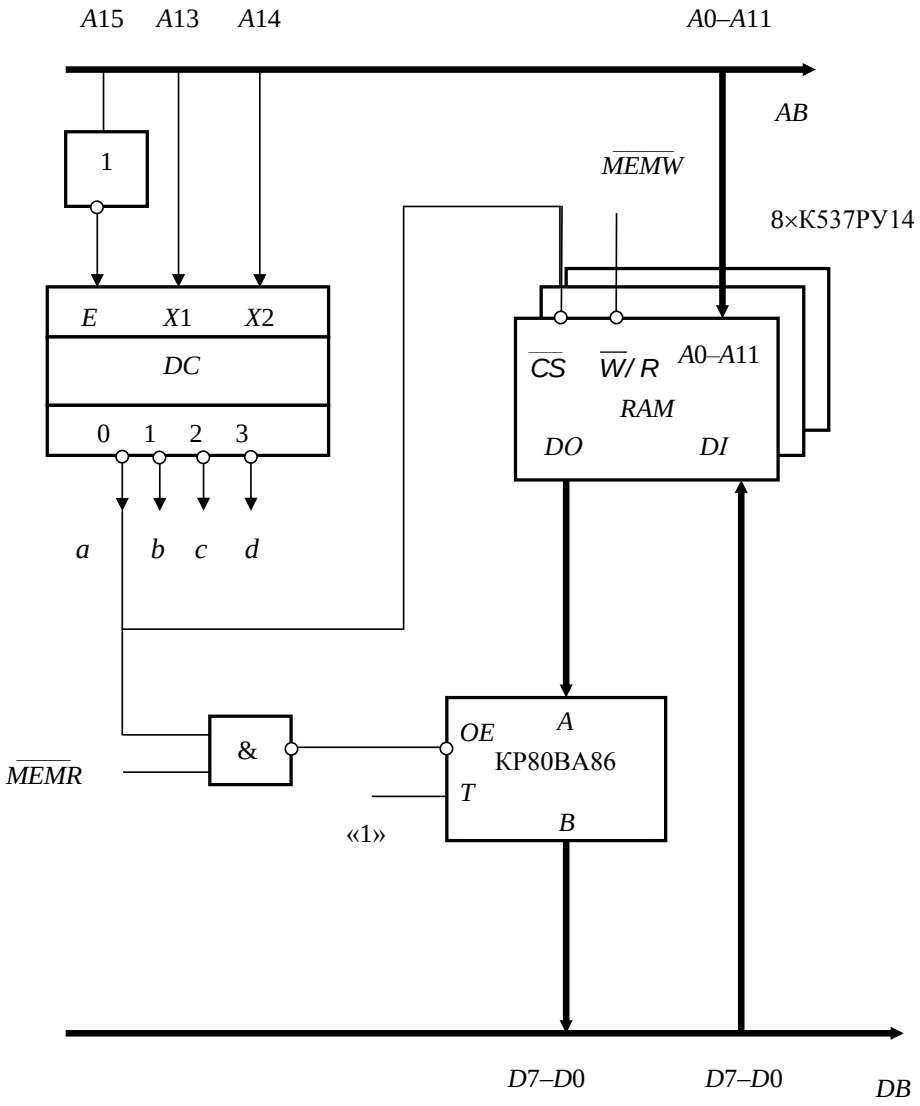


Рис. 5.22. Нарощування розрядності модуля статичного ОЗП

Отже, розрядність нарощується приєднанням однієї ВІС ОЗП до кожної лінії шини даних. При цьому адресні входи для всіх ВІС ОЗП з'єднуються паралельно. Паралельно також з'єднуються входи керування і вибірки. Нарощування ємності ОЗП здійснюється аналогічно нарощуванню ємності ПЗП. На рис. 5.22 показано один із чотирьох можливих варіантів приєднання модуля ОЗП до виходу *a* дешифратора, інших трьох модулів ОЗП – до виходів *b, c, d* дешифратора.

Розглянемо побудову модуля оперативної пам’яті для МПС на основі 16-розрядних процесорів. Модуль ОЗП так само, як і ПЗП, виконують у вигляді двох банків, приєднаних до молодшої і старшої половин шини даних відповідно.

Приклад 5.4. Розробити схему модуля ОЗП інформаційною ємністю 16 К×16 для МПС із 16-розрядним МП.

У схемній реалізації модуля ОЗП має бути забезпечене читання/запис як 8-розрядних, так і 16-розрядних даних. Модуль ОЗП складається з двох банків – молодшого та старшого (рис. 5.23).

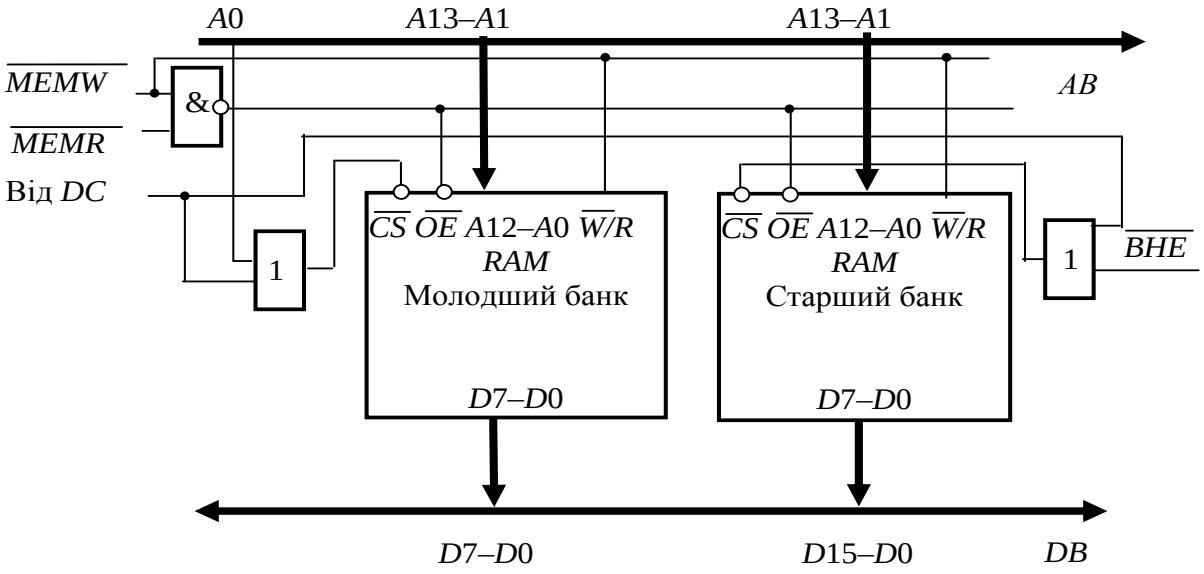


Рис. 5.23. Модуль ОЗП у 16-розрядних МПС

Адресні розряди *A*13–*A*1 з’єднані з адресними входами *A*12–*A*0 обох банків. Вибірка банків здійснюється одиничним значенням сигналу на виході дешифратора *DC*, сигналами *A*0 для вибірки молодшого банку або $\overline{BHE}$ (*Bus High Enable* – дозвіл старшого байта шини) для вибірки старшого банку. Одиничний рівень сигналу на виході *DC* з’являється під час надходження на шину адреси *AB* відповідної адреси ОЗП. Сигнал $\overline{MEMR}$ або $\overline{MEMW}$ указує на виконання циклу ЧИТАННЯ ПАМ’ЯТІ або ЗАПИС У ПАМ’ЯТЬ відповідно.

Комбінація значень сигналів *A*0 і $\overline{BHE}$ визначає чотири можливі випадки звернення до пам’яті:

– вибірку байта за парною адресою;
– вибірку байта за непарною адресою;
– вибірку слова за парною адресою;
– вибірку слова за непарною адресою.

Під час вибірки байта за парною адресою сигнали *A*0 = 0, $\overline{BHE}$ = 1. Байт із парною адресою передається по лініях *D*7–*D*0, тобто здійснюється

зчитування або запис байта. У випадку запису байта в молодший банк інформація в старшому банку захищена від стирання, тобто одиничне значення сигналу $\overline{BHE}$ забороняє звернення до старшого банку.

Під час вибірки байта за непарною адресою сигнали $A0 = 1$, $\overline{BHE} = 0$. Наприклад, за командою

```
MOV BL, BYTE PTR [1001H]
```

вміст комірки пам'яті з адресою *DS*:1001*H* пересилається в молодшу половину 16-розрядного регістра *BX*, тобто у 8-розрядний регістр *BL*. При цьому вміст комірки пам'яті записується у розряди *D*15–*D*8, тобто у старшу половину шини даних, потім, у процесі виконання команди, – на молодшу половину внутрішньої 16-розрядної шини МП, а після цього – в регістр *BL*. Цей процес, що називається *маршрутизацією байта*, відбувається автоматично і непомітно для програміста.

Під час вибірки слова за парною адресою сигнали $\overline{BHE} = 0$, $A0 = 0$. У цьому разі водночас обираються два банки і 16-розрядне слово передається по лініях *D*15–*D*0 за один цикл шини.

Якщо слово має непарну адресу, то його молодший байт розміщено у старшому банку пам'яті, старший байт – у молодшому банку. Під час вибірки слова за непарною адресою спочатку $A0 = 1$, $\overline{BHE} = 0$, по лініях шини *D*15–*D*8 передається молодший байт. Після цього генеруються сигнали $A0 = 0$, $\overline{BHE} = 1$, здійснюється інкремент (збільшення на одиницю) повної адреси *A*19–*A*0, старший байт слова передається з молодшого банку або у молодший банк по лініях шини *D*7–*D*0. Отже, вибірка слова за непарною адресою потребує два цикли шини. Тому слова доцільно розміщувати за парними адресами, особливо під час організації операцій зі стеком.

Контрольні запитання

1. Як здійснюється нарощування розрядності у модулі статичного ОЗП?
2. Які особливості має побудова модулів оперативної пам'яті для МПС на базі 16-розрядних процесорів?
3. Які сигнали використовують для вибірки банків пам'яті ОЗП?
4. Назвіть чотири можливі випадки звернення до пам'яті в 16-розрядних процесорах?
5. Що таке маршрутизація байта?
6. Які рекомендації можна дати щодо розміщення даних у стеку?

5.6. Оперативні запам'ятовувальні пристрої динамічного типу

У мікросхемах ОЗП динамічного типу елемент пам'яті – це конденсатор *p-n*-переходу МДН-транзистора. Заряджений стан конденсатора вважають станом логічної одиниці, розряджений – станом логічного нуля. Такі елементи пам'яті не можуть тривалий час зберігати свій стан і тому потребують додаткового обладнання для забезпечення періодичного відновлення (регенерації) інформації. Час вибірки для динамічного ОЗП становить 70–200 нс. Порівняно з ОЗП статичного типу ОЗП динамічного типу характеризуються більшою інформаційною ємністю, що зумовлено меншою кількістю компонент в одному елементі пам'яті; меншою швидкодією, що пов'язано з потребою заряджати і розряджати конденсатор; меншою потужністю споживання; меншою вартістю. Переважно модулі оперативної пам'яті сучасних МПС реалізуються на базі ВІС ОЗП динамічного типу.

Структурну схему динамічного ОЗП з інформаційною ємністю $N\times1$ показано на рис. 5.24.

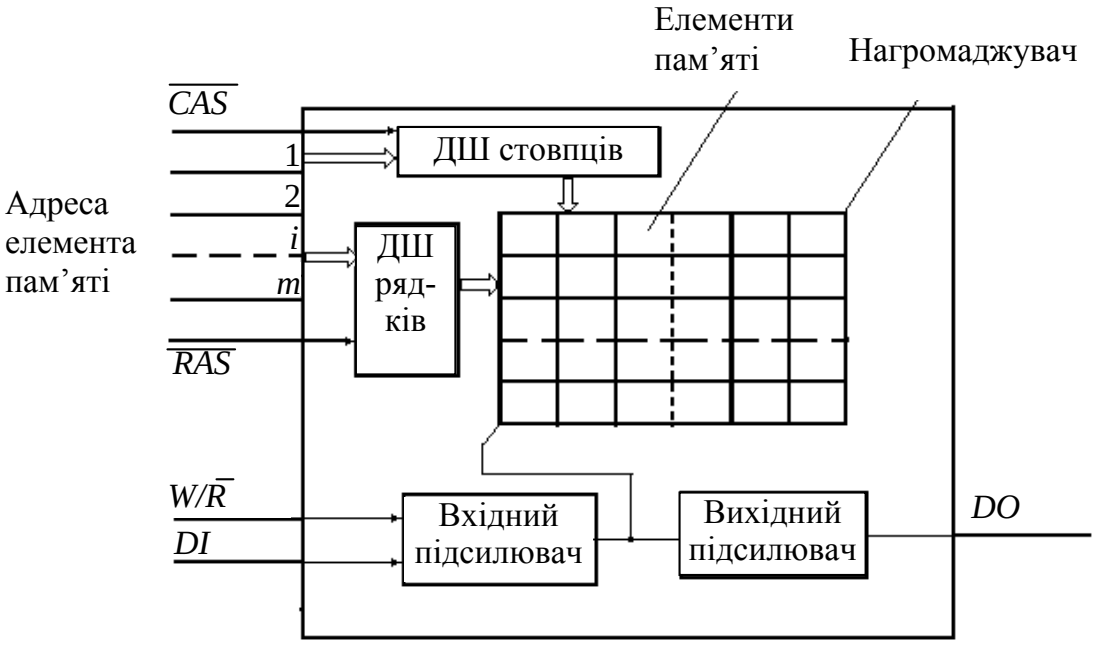


Рис. 5.24. Мікросхема динамічного ОЗП з інформаційною ємністю $N\times1$

Оперативні ЗП динамічного типу працюють у таких режимах: запису, зчитування, зчитування/модифікації/запису, сторінкового запису, сторінкового зчитування, регенерації. Для забезпечення адресування всієї ємності пам'яті використовують мультиплексування адресних сигналів у часі. Спочатку відбувається зчитування або запис даних, адреса яких визначається *m* молодшими розрядами шини адреси, що супроводжується сигналом строба $\overline{RAS}$ (*Row Address Strobe* – строб адреси рядка), який надходить на дешифратор рядків.

Після цього виконується зчитування або запис даних, адреса яких визначається *m* старшими розрядами шини адреси, що супроводжується сигналом строба $\overline{CAS}$ (*Column Address Strobe* – строб адреси стовпця), який надходить на ДШ стовпців. Для ВІС К5656РУ7 (рис. 5.25) за допомогою восьми адресних ліній *A*7–*A*0 передаються дані, адресування яких потребує 16 адресних ліній. Ця ВІС має інформаційну ємність 64 К×1 та два виводи, на які надходять сигнали стробів $\overline{RAS}$ і $\overline{CAS}$.

Зчитування інформації відбувається за заднім фронтом сигналу $\overline{CAS}$ при $\overline{W}/R = 1$, запис – за заднім фронтом сигналу $\overline{CAS}$ при $\overline{W}/R = 0$. Режим зчитування/модифікації/запису полягає у зчитуванні інформації з подальшим записом в один і той самий елементи пам'яті. Сторінкові режими запису та зчитування реалізуються зверненням до ВІС за адресою рядка з вибіркою елемента пам'яті цього рядка зміною адрес стовпців.

Регенерація інформації відбувається зверненням до кожного з рядків, при цьому формується адреса рядка і сигнал $\overline{RAS}$, а сигнал $\overline{CAS}$ дорівнює логічній одиниці. Процес регенерації припиняється при зверненні МП до ОЗП. У цьому разі обробляється вимога МП, після чого режим регенерації продовжується з тієї адреси, на якій його було припинено. Зазначені режими й умови їх реалізації стосовно мікросхем серії К565 відображено в табл. 5.11.

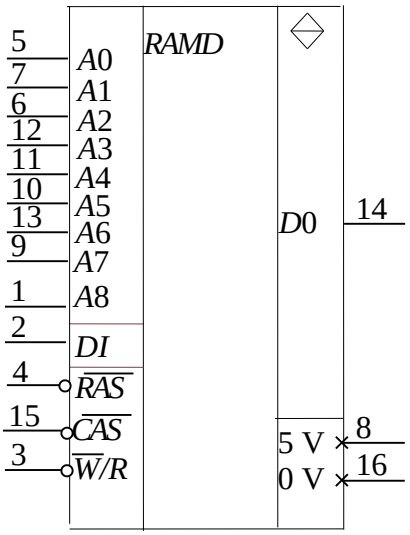


Рис. 5.25. ВІС динамічного ОЗП К565РУ7

Таблиця 5.11. **Таблиця істинності мікросхем серії К565**

$\overline{RAS}$	$\overline{CAS}$	$\overline{W}/R$	*A*	*DI*	*D*0	Режим роботи
1	1	*x*	*x*	*x*	*Z*	Зберігання
1	0	*x*	*x*	*x*	*Z*	Зберігання
0	1	*x*	*A*	*x*	*Z*	Регенерація
0	0	0	*A*	0	Z	Запис 0
0	0	0	*A*	1	*Z*	Запис 1
0	0	1	*A*	*x*	*D*	Зчитування

Часові діаграми роботи ВІС К565 у режимах запису (*а*), зчитування (*б*), регенерації (*в*) показано на рис. 5.26.

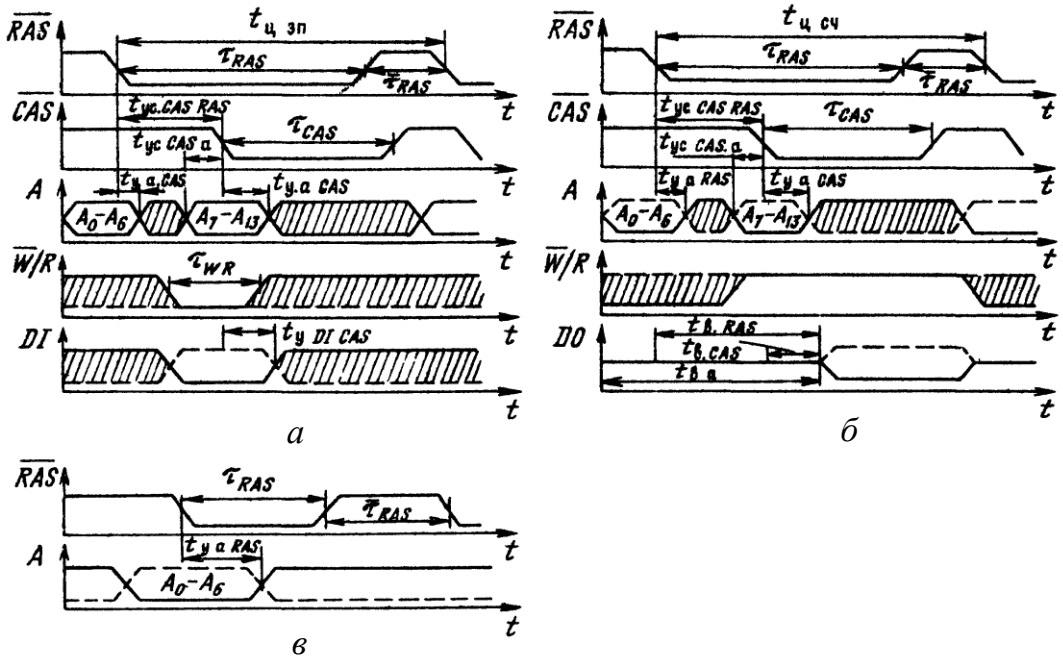


Рис. 5.26. Часові діаграми мікросхеми К565РУ3 у режимах запису (а), зчитування (б), регенерації (в)

Щоб звернутися до мікросхеми для запису і зчитування інформації, треба подати код адреси рядків A_0–A_7, одночасно з ним або з якоюсь (не нормується) затримкою сигнал $\overline{RAS}$, потім з нормованою затримкою на час утримання адреси рядків щодо сигналу $\overline{RAS}$ слід подати код адреси стовпців і через час установлення $t_{\text{ус.а}CAS}$ – сигнал $\overline{CAS}$. До моменту подачі коду адреси стовпців на вхід *DI* підводять записуваний біт інформації, який сигналом $\overline{W}/R$ за наявності $\overline{CAS} = 0$ фіксується на вхідному тригері. Сигнал запису $\overline{W}/R$ може бути поданий рівнем або імпульсом. В останньому випадку його тривалість має бути не менше визначеного параметром τ_{WR} значення(рис. 5.26, *а*). Якщо сигнал запису поданий рівнем, то фіксацію *DI* тригером виконує негативний перепад сигналу $\overline{CAS}$ (за наявності $\overline{RAS} = 0$). Після запису треба витримати паузу τ_{RAS}, яка дорівнює інтервалу між сигналами $\overline{RAS}$, для відновлення стану внутрішніх кіл мікросхеми.

В аналогічному порядку треба подати адресні та керувальні сигнали під час зчитування інформації (рис. 5.26, *б*). Сигнал $\overline{W}/R = 1$ може бути поданий рівнем або імпульсом. Час появи вихідного сигналу можна відраховувати від моменту надходження сигналів адреси $t_{\text{в.а}}$ або сигналів керування, час вибірки сигналу $\overline{RAS}$ $t_{\text{в}RAS}$, час вибірки сигналу $\overline{CAS}$ $t_{\text{в}CAS}$. Оцінюючи мікросхему за цими параметрами, варто мати на увазі, що вони взаємозалежні, і тому досить знати один з них. Більш інформативний параметр $t_{\text{в}CAS}$, оскільки

інформацію виводить з мікросхеми сигнал $\overline{CAS}$ за наявності, звичайно, сигналу зчитування $\overline{W}$ /R = 1. З рис. 5.26, *б* випливає: $t_{вRAS} = t_{вCAS} + t_{усCAS,RAS}$.

У табл. 5.12 наведено динамічні параметри мікросхем ОЗП серії К565, а самі параметри зазначено на часових діаграмах рис. 5.26.

***Таблиця 5.12*. Динамічні параметри мікросхем серії К565**

Параметр, нс	К565РУ3			К565РУ5,6				К565РУ7	
	А, Б	В	Г	Б	В	Г	Д	В	Г
$t_{ц.зп/зч}$	510	410	370	230	280	360	460	340	410
$t_{ц.зч/м/зп}$	670	520	420	310	380	460	600	410	490
$t_{ц.зп/зч}$*	370	275	225	150	180	250	320	120	140
τ_{RAS}	300	250	200	120	150	200	250	150	200
$\bar{\tau}_{RAS}$	200	150	120	100	120	150	200	180	200
τ_{CAS}	220	165	135	70	90	120	150	75	100
$t_{усCAS,RAS}$	100	85	65	30	35	55	75	50	60
$t_{ус.а.RAS}$	60	45	25	15	20	40	60	20	25
$t_{усCAS.а}$	20	10	10	0	0	0	0	–	–
$t_{ус.аCAS}$	100	75	55	25	35	45	60	–	–
τ_{WR}	120	75	55	35	45	80	120	40	65
$t_{усDI\ CAS}$	100	75	55	45	55	80	120	55	70
τ_{CAS} *	140	100	80	70	80	120	160	60	70
$\tau_{в\ CAS}$	200	165	135	70	90	120	150	50	60
$T_{рег}$, мс	2	2	2	2	2	2	1	8	8

* У сторінковому режимі.

Щоб забезпечити надійне зберігання записаної в нагромаджувачі інформації, реалізують режим примусової регенерації. Регенерація інформації в кожнім ЕП має відбуватися не рідше ніж через 2 мс (для К565РУ5Д и КР565РУ6Д через 1 мс). Регенерація автоматично виконується для всіх ЕП обраного рядка під час звертання до матриці для запису чи зчитування інформації.

Час, протягом якого треба звернутися до рядка для регенерації, визначає параметр «Період регенерації» $T_{рег}$, (див. табл. 5.12).

Цикл регенерації складається з m звертань до матриці, де m – кількість рядків, шляхом перебору адрес рядків за допомогою зовнішнього лічильника циклів звертань. Звертання до матриці для регенерації може бути організоване у кожному з режимів: запису, зчитування, зчитування/модифікації/запису, а також у спеціальному режимі регенерації – сигналом $\overline{RAS}$.

Режим роботи *зчитування/модифікація/запис* полягає в зчитуванні інформації про наступний запис у той самий ЕП. У часових діаграмах сигналів для цього режиму сполучено діаграми для зчитування

(див. рис. 5.26, *б)* і запису (див. рис. 5.26, *а*) інформації: для незмінних сигналів $\overline{RAS}$ і $\overline{CAS}$ режим зчитування переміняє режим запису даних за тією самою адресою. Модифікація режиму полягає в зміні сигналу зчитування на сигнал запису й у підведенні до входу *DI* записуваної інформації. Час циклу в цьому режимі звертання більше, ніж в інших (табл. 5.12). У всіх зазначених режимах регенерація відбувається в природному порядку як операція, що супроводжує процес звертання до мікросхеми.

Під час організації примусової регенерації найбільш доцільний і зручний для реалізації режим регенерації сигналом $\overline{RAS}$ (див. рис. 5.26, *в)*, коли виконують перебір адрес у супроводі сигналу стробування $\overline{RAS}$, якщо $\overline{CAS}$ = 1.

У мікросхемі К565РУ1 режим регенерації виконують по циклу зчитування чи *зчитування/модифікація/запис* з виконанням умови $\overline{CS}$ =1, за якої доступ до входу і виходу мікросхеми закрито. Вихід знаходиться у стані з високим опором.

У розрахунок часу регенерації варто брати час циклу за обраного режиму регенерації, помноживши його на кількість рядків. Наприклад, на регенерацію інформації в ЕП одного рядка мікросхеми К565РУ5Б в режимі *зчитування/модифікація/запис* треба (див. табл. 5.12) 310 нс, тоді для регенерації ЕП усіх 256 рядків буде потрібно 80 мкс, що становить 4 % робочого часу мікросхеми. У режимі регенерації тільки сигналом $\overline{RAS}$ загальний час регенерації зменшується до 61,5 мкс, що становить 3 % часу функціонування мікросхеми.

Сторінкові режими запису і зчитування реалізують звертанням до мікросхеми за адресою рядка з вибіркою ЕП цього рядка зміною адреси стовпців. У цих режимах значно зменшується час циклу запису (зчитування) (див. табл. 5.12), оскільки, якщо сигнали $\overline{RAS}$= 0 і код адреси рядка незмінні, то використовують лише частину повного циклу запису (зчитування), що належить до адресації стовпців.

Основні параметри мікросхем ОЗП динамічного типу наведено в табл. 5.13.

Таблиця 5.13. **Параметри ВІС ОЗП динамічного типу**

Тип мікросхеми	Інформаційна ємність, біт	$t_{ц.\ зп/зч}$, нс	$U_{живл}$, В	$P_{сп}$, мВт	
				Звернення	Зберігання
1	2	3	4	5	6
К565РУ3А-Г	16 К×1	510–370	12, ±5	460	40
К565РУ5Б-Д	64 К×1	230–460	5	250–160	21
К565РУ5Д3	16 К×1	110–500	5	160	21

Продовження табл. 5.13

1	2	3	4	5	6
К565РУ6Б-Д	16 К×1	460	5	150–120	20
К565РУ6В,Г	256 К×1	450	5	350	35
К565РУ7	256 К×1	200	5	250	25
К565РУ8	256 К×1	120	5	250	25
К565РУ8	1 М×1	120	5	350	35

До складу серії К565 входять мікросхеми з інформаційною ємністю 4К, 16К, 64К и 256К. Мікросхеми К565РУ1 і К565РУЗ мають потребу в трьох джерелах живлення. Застосовуючи ці мікросхеми, варто враховувати вимоги вмикання і вимикання джерел живлення: першим підключають джерело мінус 5 В, а відключають останнім. Ця вимога зумовлена тим, що напруга мінус 5 В подається на підкладку (кристал) і якщо її не підключити першою, то під впливом, навіть короткочасним, напруг двох інших джерел з напругою 5 і 12 В в кристалі може відбутися тепловий пробій і ушкодитися мікросхема.

Контрольні запитання

1. Що являє собою елемент пам’яті динамічного ОЗП?
2. Назвіть основні параметри ОЗП динамічного типу.
3. Порівняйте параметри ОЗП статичного і динамічного типів. Назвіть недоліки і переваги ОЗП динамічного типу.

5.7. Побудова модулів оперативних запам’ятовувальних пристроїв динамічного типу

Для керування ОЗП динамічного типу використовують контролери динамічної пам’яті, наприклад, К1810ВТ03, К1810ВТ02, *i*8207, які формують адресні та керувальні сигнали у режимах роботи і регенерації, а також здійснюють арбітраж, тобто розв’язання конфліктів між запитами на регенерацію і звернення до пам’яті. Контролер формує також сигнал готовності блоку динамічної пам’яті до обміну.

Контролер динамічної пам’яті *i*8207 (рис. 5.27) призначено для керування чотирма ВІС ОЗП динамічного типу по 512 кбайт у кожній. Він складається з адресних буферів *B*1 і *B*2, лічильника рядків регенерації (ЛРР), двох мультиплексорів *M*1 і *M*2 та системи керування. Система керування аналізує вхідні сигнали і керує всіма блоками контролера.

Виконуючи команди читання/запису пам’яті, контролер забезпечує з’єднання виходів *AO*8–*AO*0 спочатку з молодшою, а потім зі старшою половинами адреси, формування стробувальних сигналів $\overline{RAS}$ і $\overline{CAS}$ для кожної із чотирьох ВІС пам’яті, а також сигналу дозволу запису. У режимі регенерації контролер забезпечує з’єднання з виводами *AO*8–*AO*0 виходів лічильника рядків регенерації і формування сигналу $\overline{RAS}$.

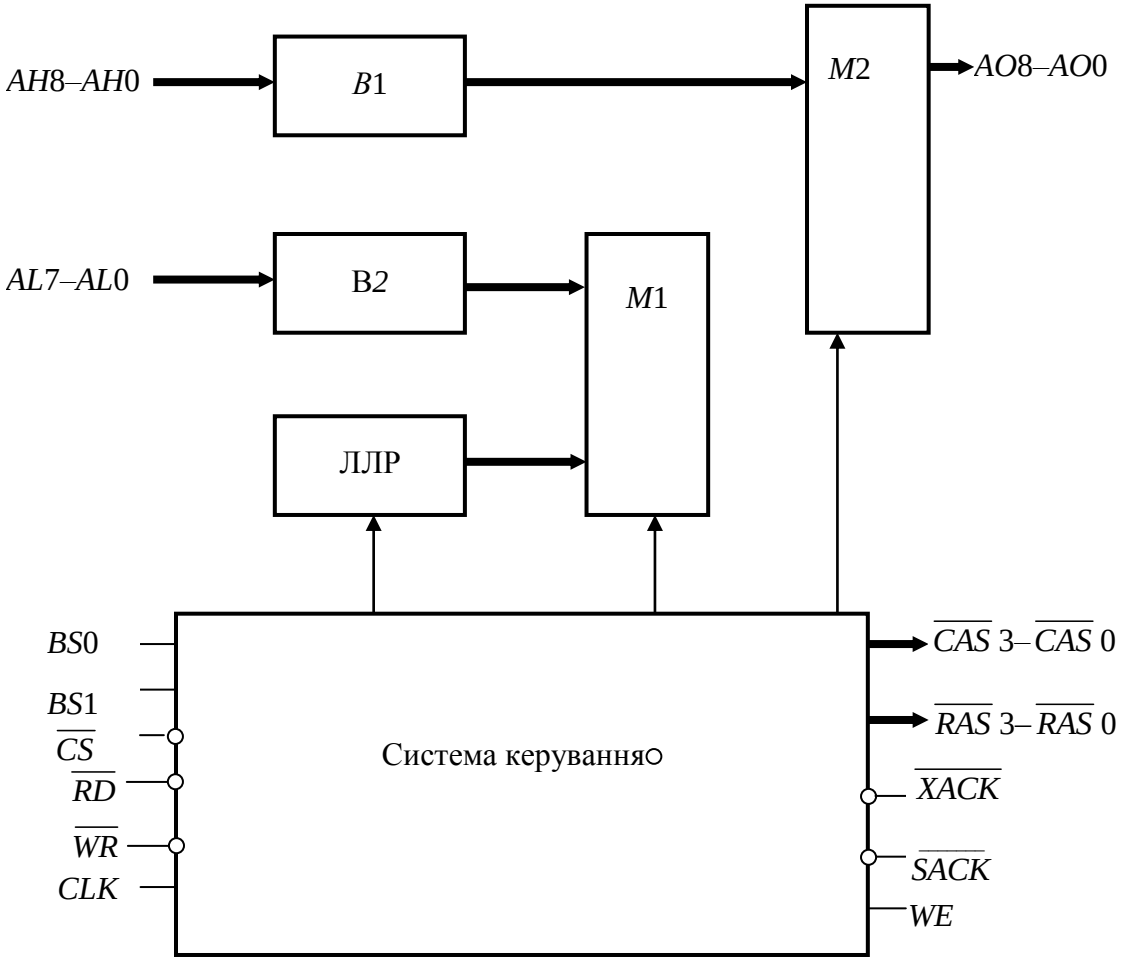


Рис. 5.27. Структурна схема контролера динамічної пам’яті *i*8207

Вибір однієї з чотирьох ВІС пам’яті здійснюється сигналами *BS*1–*BS*0 відповідно до табл. 5.13.

Таблиця 5.13. **Вибір ВІС динамічної пам’яті**

*BS*1	*BS*0	Блок пам’яті
0	0	Блок 0
0	1	Блок 1
1	0	Блок 2
1	1	Блок 3

Контролер формує сигнал $\overline{XACK}$ наприкінці циклу читання/запису, який указує на закінчення циклу взаємодії з ЦП, та сигнал $\overline{SACK}$ на початку циклу звернення до пам’яті.

Використання цих сигналів показано на прикладі приєднання контролера *i*8207 із системною шиною 16-розрядного процесора (рис. 5.28).

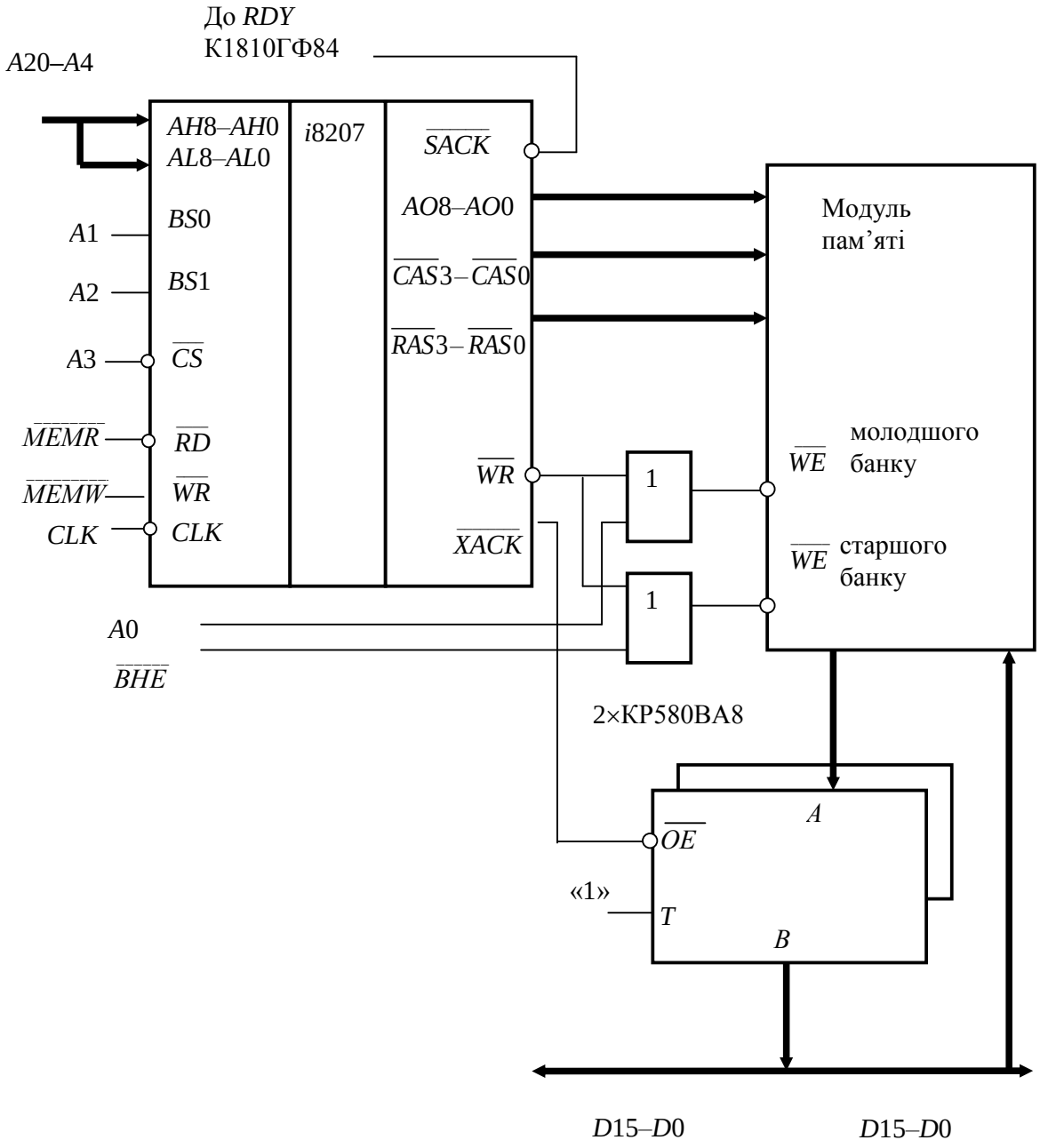


Рис. 5.28. З'єднання контролера динамічної пам'яті із системною шиною

Схему модуля динамічної пам'яті 16-розрядного процесора показано на рис. 5.29.

На виході $\overline{SACK}$ формується сигнал логічного нуля у тому разі, якщо контролер виконує такти регенерації інформації в динамічному ОЗП. Сигнал $\overline{XACK}$ використовується як сигнал стробування для керування шинними формувачами модуля пам'яті. У цьому прикладі використано два шинних формувачі К580ВА86 (*i*8286), оскільки шина даних 16-розрядна.

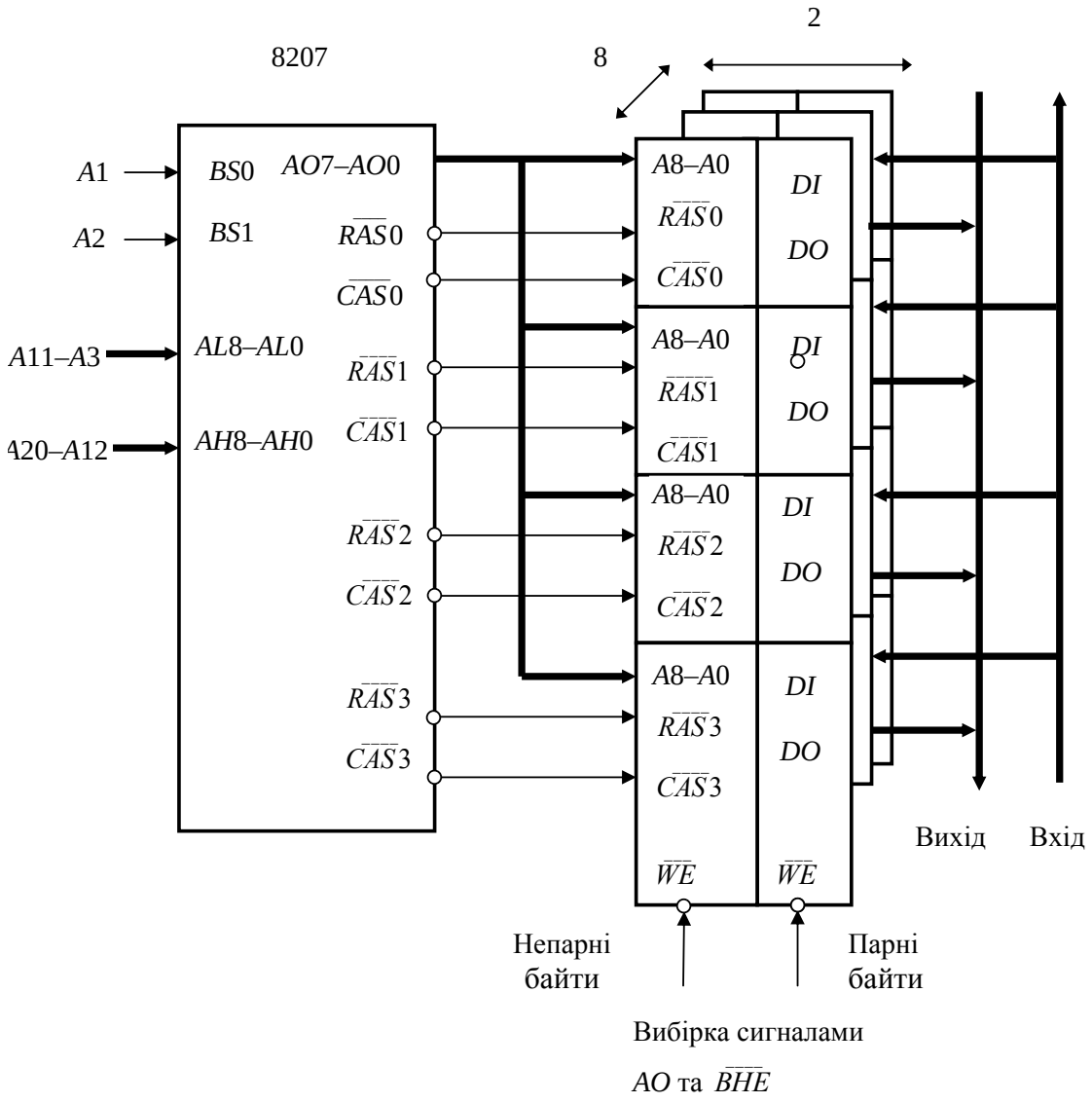


Рис. 5.29. Система динамічної пам’яті

Контролер динамічної пам’яті фактично виконує роль інтерфейсу модуля пам’яті і шини процесора. Динамічну пам’ять організовано у вигляді двох банків, кожний з яких складається із чотирьох блоків ємністю по 512 кбайт. Загальна ємність пам’яті дорівнює 2 Мбайт. Шина адреси процесора з’єднується з контролером. Лінії *A*1 і *A*0 приєднують до входів вибирання банку *BS*0, *BS*1, біти *A*3–*A*11 є адресою рядка, а біти *A*12–*A*20 – адресою колонки динамічної пам’яті.

Контрольні запитання

1. Охарактеризуйте способи адресування ОЗП динамічного типу.
2. Що таке режим регенерації пам’яті?
3. Які функції виконує контролер динамічної пам’яті?
4. Поясніть призначення сигналів $\overline{RAS}$ і $\overline{CAS}$.

5.8. Принципи організації кеш-пам'яті

Збільшення ємності пам'яті МПС зумовлює зниження швидкодії операцій обміну інформацією між процесором та модулем пам'яті. Навіть за час звернення до пам'яті, що дорівнює 70 нс, неможливо отримати потрібну інформацію за один цикл шини. Це призводить до потреби виконання тактів очікування у процесі роботи процесора для того, щоб час звернення до пам'яті був узгоджений із часом виконання команди у процесорі. Підвищення швидкодії обміну інформацією можливе через реалізацію додаткової пам'яті порівняно невеликої ємності, звернення до якої відбувається на тактовій частоті процесора. Така пам'ять отримала назву *кеш-пам'яті* або *буферної пам'яті.* Кеш-пам'ять реалізується на основі ВІС ОЗП статичного типу. Інформаційна ємність та принцип організації кеш-пам'яті є одними з основних чинників, що визначають продуктивність роботи МПС.

Кеш-пам'ять використовують не тільки для обміну даними між МП і ОЗП, але й для обміну між ОЗП і зовнішніми накопичувачами. В основу роботи кеш-пам'яті покладено принципи часової і просторової локальностей програм.

Принцип часової локальності полягає в тому, що під час зчитування будь-яких даних із пам'яті існує висока ймовірність звернення програми протягом деякого невеликого проміжку часу знову до них.

Принцип просторової локальності ґрунтується на високій імовірності того, що програма через деякий невеликий проміжок часу звернеться до комірки пам'яті, наступної за тією, до якої вона зверталася перед цим.

Згідно з принципом часової локальності інформацію у кеш-пам'яті доцільно зберігати протягом деякого часу, а принцип просторової локальності вказує на доцільність розміщення у кеш-пам'яті вмісту декількох сусідніх комірок, тобто певного блоку інформації. Лінійні ділянки програм (без переходів) у більшості випадків не перевищують 3–5 команд, тому недоцільно використовувати блоки інформації, ємність яких перевищує ємність пам'яті, потрібну для зберігання 3–5 команд. Як правило, інформація з основної пам'яті завантажується у кеш-пам'ять блоками по 2–4 слова і зберігається там деякий час.

Під час звернення процесора до пам'яті спочатку перевіряють наявність у кеш-пам'яті даних, які запитують, і якщо їх немає, здійснюють завантаження у кеш-пам'ять потрібної інформації. Правильна організація роботи кеш-пам'яті забезпечує підвищення швидкодії системи, оскільки у більшості випадків відбувається звернення процесора до кеш-пам'яті, а не до більш повільної основної оперативної пам'яті.

Залежно від способу відображення інформації з основної пам'яті на кеш-пам'ять розрізняють такі типи кеш-пам'яті:

– кеш-пам'ять з прямим відображенням;

– повністю асоціативна кеш-пам’ять;
– множинна асоціативна кеш-пам’ять.

Кеш-пам’ять з прямим відображенням – найпростіший тип кеш-пам’яті (рис. 5.30). Кеш-пам’ять містить дві частини – кеш-пам’ять даних та кеш-пам’ять ознак. Припустімо, що ємність ОЗП МПС становить 4 Гбайт. Ця ємність розбивається на 64 К рівних частин по 64 кбайт. Блок даних ємністю 4 байт пересилається з кожної із цих частин оперативної пам’яті в один 32-розрядний рядок кеш-пам’яті даних. Ємність кеш-пам’яті даних складає 64 кбайт, тому кількість рядків дорівнює 16 К. Отже, під кожну з 64 К частин ОЗП у кеш-пам’яті відводиться один рядок: 32-розрядна адреса чотирибайтового блоку в ОЗП поділяється на дві частини. Молодші 16 розрядів адреси *A*15–*A*0 називають *індексом*, старші 16 розрядів *A*31–*A*16 – *ознакою*. Ознака пересилається у кеш-пам’ять ознак, яка містить 16 К рядків і має загальну ємність 32 кбайт.

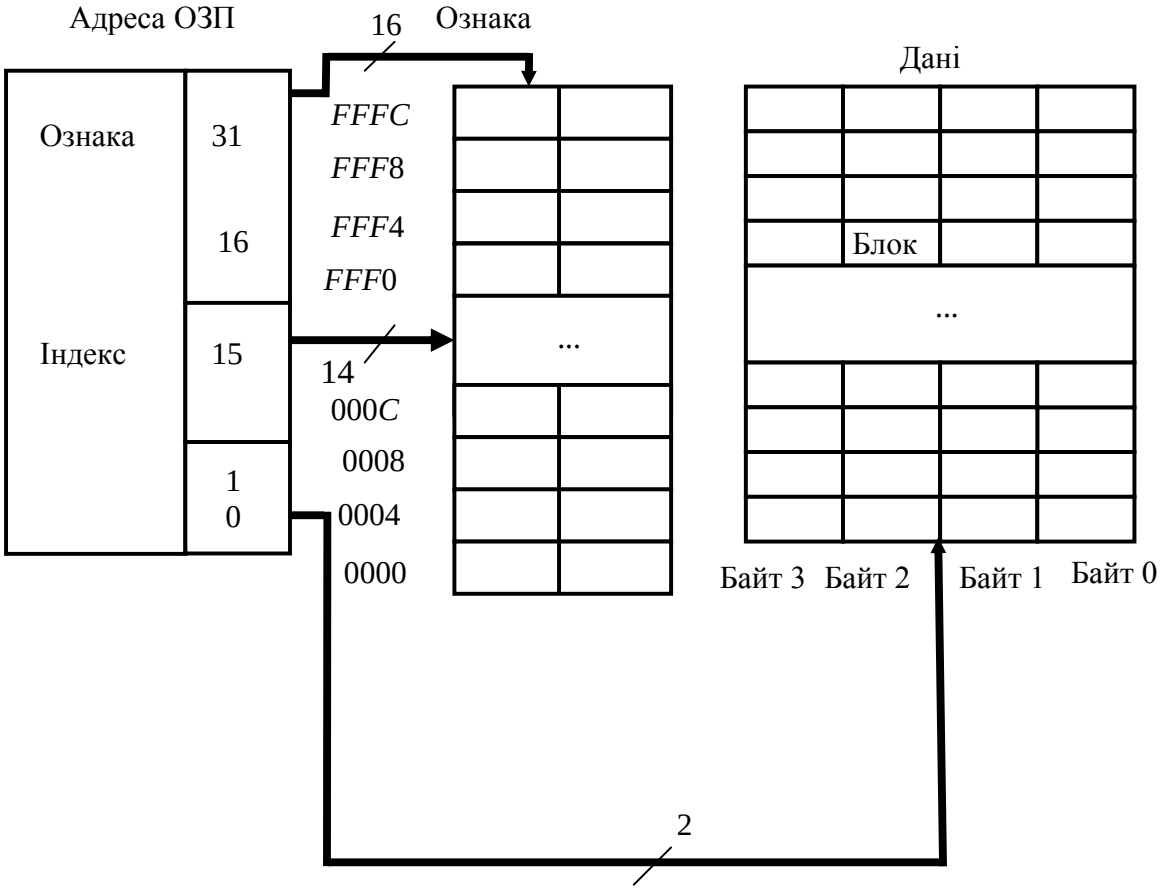


Рис. 5.30. Структурна схема кеш-пам’яті з прямим відображенням

Для визначення адреси одного рядка кеш-пам’яті ознак треба 14 адресних розрядів *A*15–*A*2. Для визначення адреси одного рядка кеш-пам’яті даних потрібно 14 адресних розрядів *A*15–*A*2, а для визначення одного байта у рядку – два розряди *A*1, *A*0.

У разі потреби зчитування даних із пам’яті процесор звертається спочатку до кеш-пам’яті та перевіряє, чи містить вона відповідні дані. Цей процес відбувається порівнянням ознаки, записаної у кеш-пам’яті ознак, з 16 старшими розрядами адреси, яку процесор виставляє на 32-розрядну шину адреси. Збіг цих величин означає, що у кеш-пам’яті зберігаються потрібні дані. У цьому разі ці дані зчитуються з кеш-пам’яті. Якщо величини не збігаються, то виконується копіювання відповідних даних із оперативної пам’яті у кеш-пам’ять.

Перевагою цього типу кеш-пам’яті є порівняно висока швидкодія, що пояснюється потребою лише в одному порівнянні ознаки зі старшими розрядами адреси ОЗП.

Недоліком кеш-пам’яті з прямим відображенням є виникнення конфліктів у разі, якщо старші 16 розрядів адреси, виставленої процесором, збігаються з ознакою, записаною у кеш-пам’яті ознак, а індекси потрібного блока та рядка у кеш-пам’яті даних не збігаються. Це означає, що рядок у кеш-пам’яті даних, відведений для цієї частини ОЗП, яка визначається старшими 16 адресними розрядами, уже зайнятий. Тому у цьому випадку вміст рядка кеш-пам’яті даних пересилається назад у ОЗП, а у рядок – необхідний чотирибайтовий блок. У результаті збільшується кількість пересилань між кеш-пам’яттю і ОЗП та час обміну інформацією.

Структурну схему *повністю асоціативної кеш-пам’яті* показано на рис. 5.31.

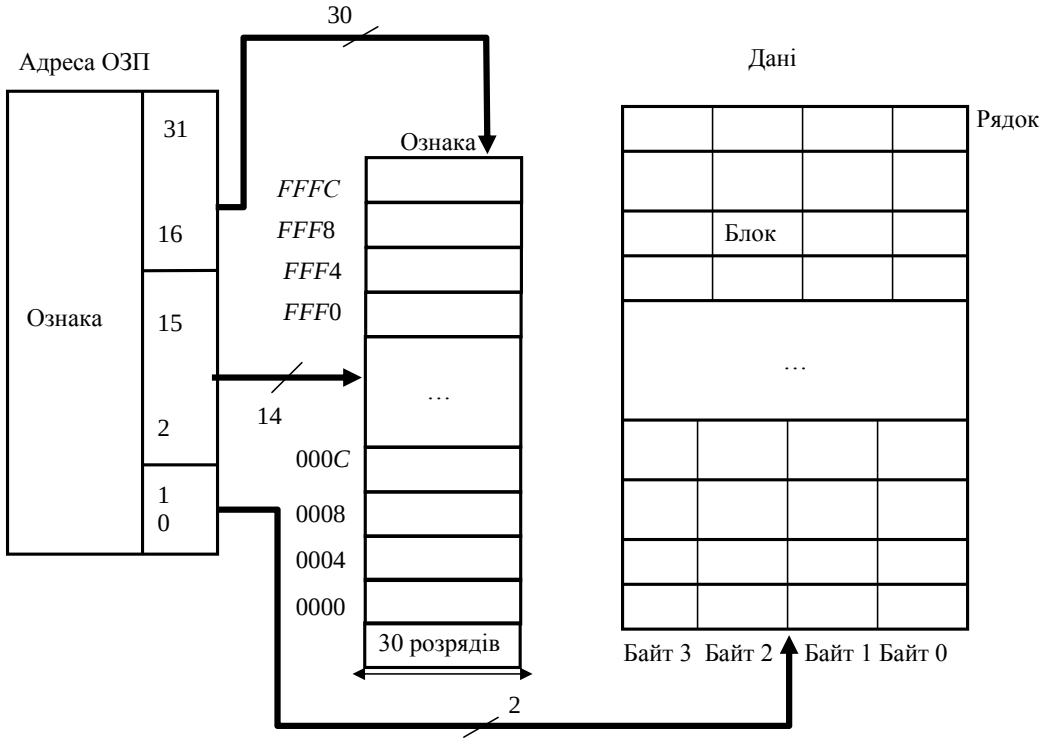


Рис. 5.31. Структурна схема повністю асоціативної кеш-пам’яті

У цій схемі усунено недолік кеш-пам’яті з прямим відображенням, оскільки будь-який блок ОЗП може відображатися у будь-якому рядку кеш-пам’яті. У кеш-пам’ять ознак записується 30-розрядна ознака, тобто старші 30 розрядів *A*31–*A*2 адреси чотирибайтового блоку ОЗП. У рядок кеш-пам’яті даних записується чотирибайтовий блок.

Коли кеш-пам’ять не заповнена, блок записується у будь-який вільний рядок. Коли кеш-пам’ять заповнена, блок з ОЗП записується в той рядок кеш-пам’яті даних, до якого було найменше звернень. Недоліки кеш-пам’яті з прямим відображенням усуваються за рахунок додаткового обладнання, призначеного для визначення блока до якого було найменше звернень. При цьому також збільшується час оброблення запитів через необхідність порівняння 30-розрядної адреси та ознаки, записаної у кеш-пам’яті ознак. Максимальна кількість таких порівнянь становить 16 К.

Множинна асоціативна кеш-пам’ять поєднує переваги обох попередніх типів. Рядки цієї кеш-пам’яті об’єднуються в групи по два, чотири і більше (згідно із цим розрізняють дво-, чотиривхідну тощо множинну асоціативну кеш-пам’ять). Структурну схему двовхідної множинної асоціативної кеш-пам’яті показано на рис. 5.32.

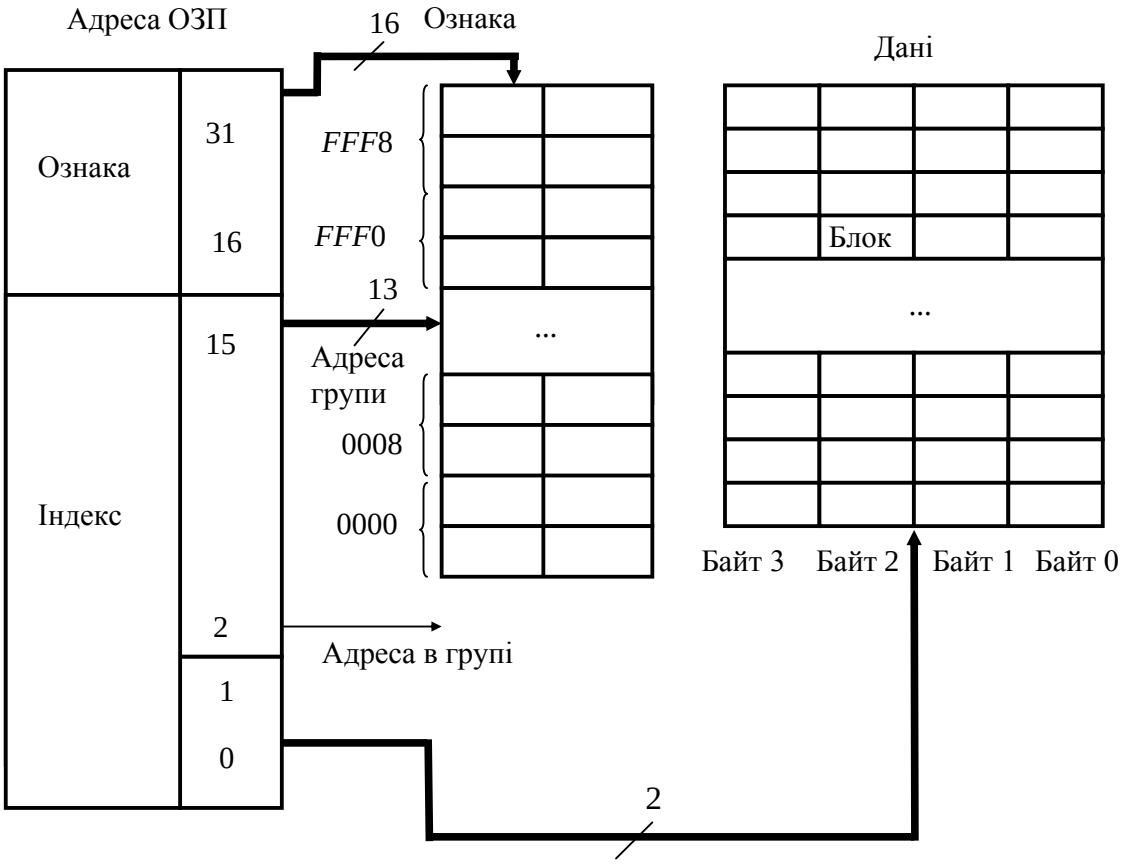


Рис. 5.32. Структурна схема двовхідної множинної асоціативної кеш-пам’яті

Кеш-пам’ять даних складається з 8 К груп, кожна з яких містить два рядки. Індекс, тобто 16 молодших розрядів адреси ОЗП, адресує байт у кеш-пам’яті даних (13 розрядів адресують одну з 8 К груп, один розряд – рядок у групі і два розряди – байт у рядку). Ознака, тобто 16 старших адресних розрядів, записується в рядок пам’яті ознак. Таким чином, для блоків з одним і тим самим індексом відводиться два рядки буфера. Отже, якщо один з рядків групи зайнято деяким блоком, то наступний блок з таким самим індексом буде розміщено у вільний рядок. Усередині групи кеш-пам’ять повністю асоціативна. Кількість порівнянь адрес ОЗП з ознаками дорівнює двом.

Зростання ємності кеш-пам’яті (тобто кількості рядків у групі) збільшує ефективність її роботи, проте зростає кількість порівнянь адреси і, отже, час оброблення запиту комірки ОЗП. Ефективність роботи кеш-пам’яті характеризується коефіцієнтом вдалих звернень. Кеш-пам’ять з прямим відображенням інформаційною ємністю $N \times n$ має такий самий коефіцієнт вдалих звернень, що і двовхідна множинна асоціативна кеш-пам’ять ємністю $(N \times n)/2$.

Усі розглянуті типи кеш-пам’яті мають властивість зберігати окремі копії інформації, яка міститься в основній пам’яті. Під час запису в кеш-пам’ять може порушуватися цілісність збережуваних даних, тобто вміст кеш-пам’яті перестане відповідати вмісту ОЗП. Існує декілька способів відновлення інформації в ОЗП, основними з яких є спосіб наскрізного запису та спосіб зворотного запису.

Спосіб наскрізного запису полягає в тому, що інформація записується як у кеш-пам’ять, так і в ОЗП.

Спосіб зворотного запису передбачає запис інформації в ОЗП лише у тому разі, якщо вона змінюється в кеш-пам’яті. Кожному рядку кеш-пам’яті зіставляється спеціальний біт – біт запису, що вказує на зміну вмісту рядка. Заміщуючи рядок кеш-пам’яті новим блоком інформації з ОЗП, перевіряють стан біта запису, і якщо біт установлено, то виконують перезапис блока з кеш-пам’яті в ОЗП. Тільки після цього в кеш-пам’яті розміщується новий блок з ОЗП. Цей спосіб більш ефективний, оскільки дозволяє зменшити кількість звернень до ОЗП.

Правильне розміщення даних у ОЗП дозволяє раціонально організовувати роботу програмного забезпечення та підвищувати швидкодію роботи МПС, оскільки, по-перше, пов’язані між собою дані доцільно розміщувати у найближчих комірках ОЗП. У цьому разі під час завантаження блока даних у кеш-пам’ять існує висока ймовірність того, що після оброблення першого слова процесор обиратиме друге слово з кеш-пам’яті, а не з ОЗП, що дозволить ефективніше використовувати кеш-пам’ять. По-друге, під час запису треба вирівнювати дані в ОЗП по межі рядка

кеш-пам’яті. Припустімо, що програма обробляє трибайтове слово, а довжина рядка кеш-пам’яті дорівнює 4 байт. Якщо розмістити слова в ОЗП підряд (рис. 5.33), то частини одного трибайтового слова, наприклад, слів *D*2 та *D*3, будуть розміщені у сусідніх рядках кеш-пам’яті.

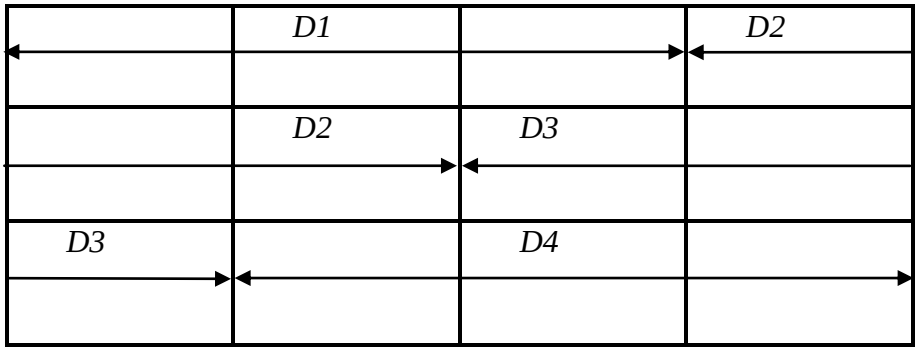


Рис. 5.33. Розміщення трибайтових слів без вирівнювання по межі блока

Без вирівнювання даних досить часто трапляються невдалі звернення до кеш-пам’яті. На рис. 5.34 показано запис трибайтових слів *D*1–*D*4, вирівняних по межі чотирибайтового рядка кеш-пам’яті.

Це дозволяє значно зменшити кількість невдалих звернень до кеш-пам’яті.

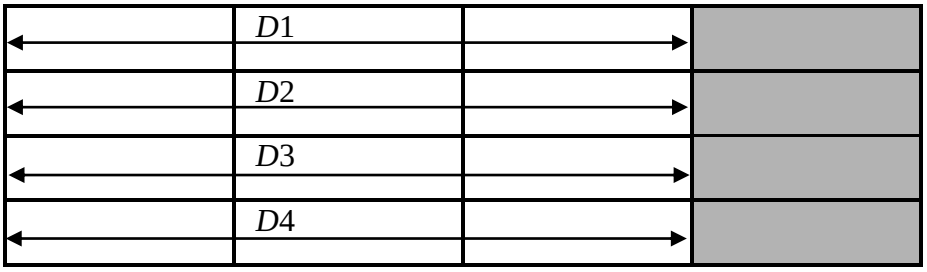


Рис. 5.34. Розміщення трибайтових слів з вирівнюванням
по межі чотирибайтового рядка кеш-пам’яті

Контрольні запитання

1. Наведіть визначення та призначення кеш-пам’яті.
2. У чому полягає принцип часової локальності?
3. У чому полягає принцип просторової локальності?
4. Поясніть принцип дії кеш-пам’яті з прямим відображенням.
5. Поясніть принцип дії повністю асоціативної кеш-пам’яті.
6. Поясніть принцип дії множинної асоціативної кеш-пам’яті.
7. Порівняйте два типи кеш-пам’яті з прямим відображенням та повністю асоціативну.
8. Порівняйте повністю асоціативну кеш-пам’ять з множинною асоціативною кеш-пам’яттю.
9. Як відновлюється інформація в ОЗП за способом наскрізного запису?
10. Як відновлюється інформація в ОЗП за способом зворотного запису?

5.9. Принципи організації стекової пам'яті

Стековою пам'яттю або *стеком* називають пам'ять, у якій реалізовано принцип: останній увійшов – перший вийшов (*LIFO – Last Input First Output*), тобто дані, записані останніми зчитуються першими. У МПС стекова пам'ять використовується для викликів підпрограм, у тому числі і вкладених, та оброблення переривань.

За способом реалізації розрізняють *апаратний* і *апаратно-програмний* стеки.

Апаратний стек являє собою сукупність регістрів, зв'язок між якими організовано таким чином, що під час записування і зчитування даних уміст стека автоматично зсувається. Принцип роботи апаратного восьмирівневого стека ілюструє рис. 5.35.

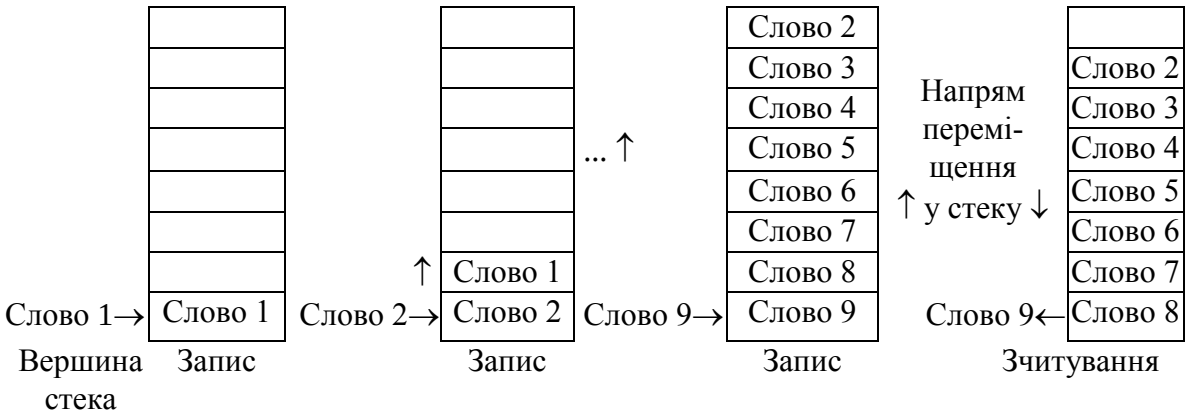


Рис. 5.35. Принцип роботи апаратного стека

Під час записування слова 1 у стек воно розміщується у першій вільній комірці пам'яті (у першому регістрі) – вершині стека. Наступне слово зсуває попереднє на одну комірку вгору, займає його місце і т. д. Запис слова 9 призводить до переповнення стека і втрати слова 1. Зчитування слів зі стека здійснюється у зворотному порядку, тобто спочатку зчитується слово 9, записане останнім. Зчитування відбувається у зворотному порядку, наприклад, зчитування слова 6 не можливе, поки не будуть зчитані слова 7, 8 та 9.

Інформаційна ємність апаратного стека визначається як $N \times n$, де кількість n-розрядних слів N дорівнює кількості регістрів, яка може складати декілька десятків. Апаратні стеки, застосовувані у *PIC*-процесорах, мають 2, 8 або 16 регістрів ($N = 2, 8, 16$), у яких розміщуються 12-, 14- або 16-розрядні слова ($n = 12, 14, 16$). Основною перевагою апаратного стека є висока швидкодія, а недоліком – обмежена інформаційна ємність.

Апаратно-програмний стек реалізується через використання частини ОЗП статичного типу та спеціального регістра *SP* (*Stack Pointer* – вказівник стека), який містить адресу останньої зайнятої комірки стека.

Принцип роботи апаратно-програмного стека для мікропроцесорів 80*x*86 показано на рис. 5.36. У апаратно-програмному стеку фізичного зсуву даних під час запису і зчитування не відбувається. Зсув даних буває за зміною значення *SP*. На початку програми в регістр *SP* заносять адресу вершини стека. Після кожної операції запису (зчитування) вміст регістра *SP* змінюється. Для МП 80*x*86 одночасно можна записувати у стек або зчитувати з нього двобайтові слова, тому *SP* змінюється на два.

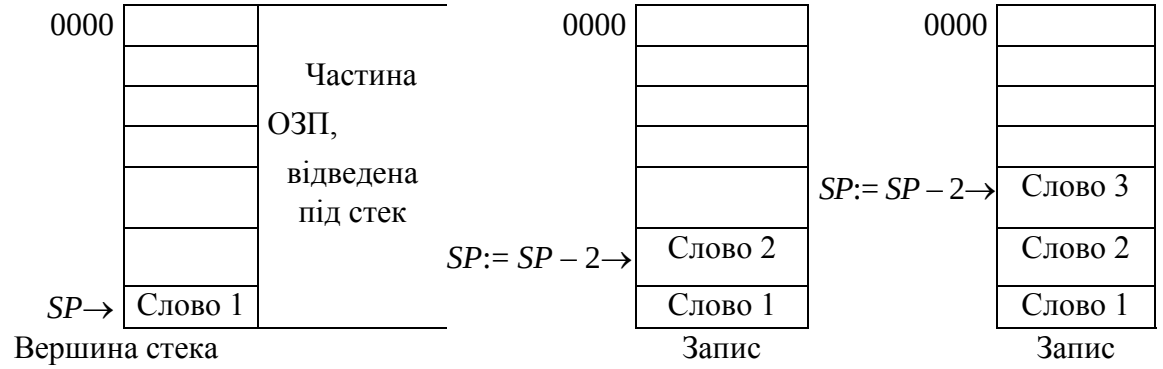


Рис. 5.36. Принцип роботи апаратно-програмного стека

Під час запису у стек значення *SP* зменшується на два (стек «росте» в область малих адрес), а під час зчитування зі стека – збільшується на два. Отже, вказівник стека *SP* завжди містить адресу комірки, до якої відбулося останнє звернення. У деяких командах, наприклад у командах викликів підпрограм *CALL*, переривань *INT*, повернень з підпрограм *RET*, звернення до стека здійснюється автоматично. Під час виклику підпрограми (рис. 5.37) у стеку запам'ятовується адреса команди, наступної після виклику команди *ADD*, тобто вміст програмного лічильника *PC* запам'ятовується у верхній незайнятій комірці стека, а вказівник стека зменшується на два.

Після повернення з підпрограми за командою *RET* уміст верхньої комірки стека перезаписується у програмний лічильник *PC*, вказівник стека *SP* збільшується на два. Після цього починає виконуватися команда *ADD*. Крім команд *CALL*, *INT* і *RET*, для роботи зі стеком використовують також команди *PUSH* і *POP*, призначені для тимчасового запам'ятовування у стек умісту регістрів і відновлення, тобто пересилання інформації зі стека у регістри. У МП *Intel*, починаючи з *i*286, існують команди *PUSHA* і *POPA* (*PUSH All* і *POP All*), призначені для тимчасового запам'ятовування у стеку і відновлення вмісту всіх регістрів МП. До апаратно-програмного стека можна звернутися також як до ОЗП з довільною вибіркою. У МП *i*80*x*86 для цього використовують непряму адресацію за допомогою регістра *BP*. Отже, у стек можна записати значення параметрів підпрограм перед їх викликом.

Використання стекової пам’яті дозволяє підвищити швидкодію МПС, зменшуючи час однієї з найповільніших операцій – звернення до зовнішньої пам’яті.

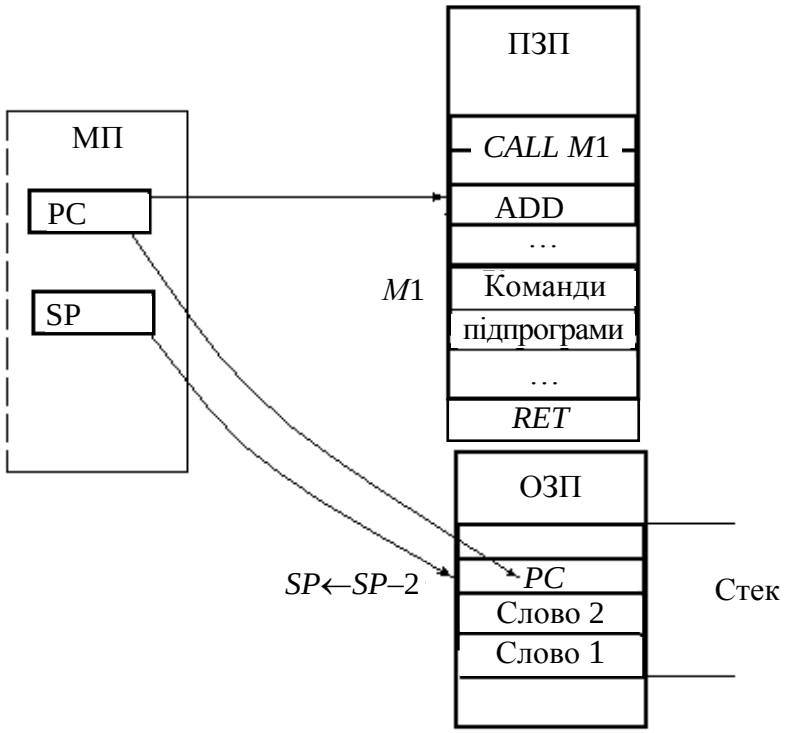


Рис. 5.37. Робота стека під час виклику підпрограм

Контрольні запитання

1. Дайте визначення стекової пам’яті.
2. Яке призначення стекової пам’яті?
3. Поясніть принцип дії апаратного стека.
4. Поясніть принцип дії апаратно-програмного стека.
5. Дайте порівняльну характеристику апаратного та аппаратно-програмного стеків.
6. Поясніть призначення регістра *SP*.
7. Які операції виконує процесор за командами *CALL*, *RET*, *PUSH*, *POP*, *PUSHA*, *POPA*?

Розділ 6

ІНТЕРФЕЙС ПРИСТРОЇВ ВВЕДЕННЯ-ВИВЕДЕННЯ

6.1. Функції інтерфейсу введення-виведення

Одним з найважливіших завдань проектування МПС є організація взаємодії із зовнішніми пристроями – джерелами і приймачами даних. Прикладами ПВВ, що є як джерелами, так і приймачами інформації, є нагромаджувачі на гнучких і твердих магнітних дисках. До пристроїв уведення належать перемикачі, клавіатура, аналого-цифрові перетворювачі (АЦП), датчики двійкової інформації, а до пристроїв виведення – індикатори, світлодіоди, дисплеї, друкувальні пристрої, цифро-аналогові перетворювачі (ЦАП), транзисторні ключі, реле, комутатори. ПВВ відрізняються: розрядністю даних, швидкодією, протоколами, тобто визначеним порядком обміну, керувальними сигналами. Дані у ПВВ змінюються у довільний або чітко визначений момент часу. З'єднання ПВВ із системною шиною МПС здійснюється за допомогою ІВВ, який узгоджує ПВВ із системною шиною МПС. Зазвичай інтерфейс складається з одного або декількох портів уведення-виведення та схем керування ними.

Проектуючи ІВВ, необхідно забезпечити:

– зберігання інформації, яка надходить від ПВВ;
– доступ до інформації з боку МП;
– керування обміном;
– перетворення форматів даних.

Зберігання інформації та доступ до неї з боку МП. Уведення та виведення інформації виконується за допомогою портів уведення-виведення, які являють собою 8- або 16-розрядні регістри зі схемами вибірки та керування читанням/записом. Як порти можуть бути використані буферні регістри, наприклад, *i*8282, *i*8285, КР580ИР82, КР589ИР12, КР580ВВ55. Використання регістра КР580ИР82 для з'єднання з пристроєм уведення та пристроєм виведення показано на рис. 6.1.

Якщо регістр використовується як порт уведення (рис. 6.1, *а*), то дані від пристрою введення надходять у регістр по лініях *DI*7–*DI*0 і записуються за

стробом *STB*. Вихідні дані *DO*7–*DO*0 порту надходять у МПС по шині даних. мікропроцесор формує також сигнал керування читанням і вибіркою порту, який надходить на вхід $\overline{OE}$. Якщо регістр використовується як порт виведення (рис. 6.1, *б*), то дані від МП надходять по шині даних на входи *DI*7–*DI*0 порту і супроводжуються сигналами керування записом і вибірки ВІС. Вихідні дані *DO*7–*DO*0 порту надходять у пристрій виведення.

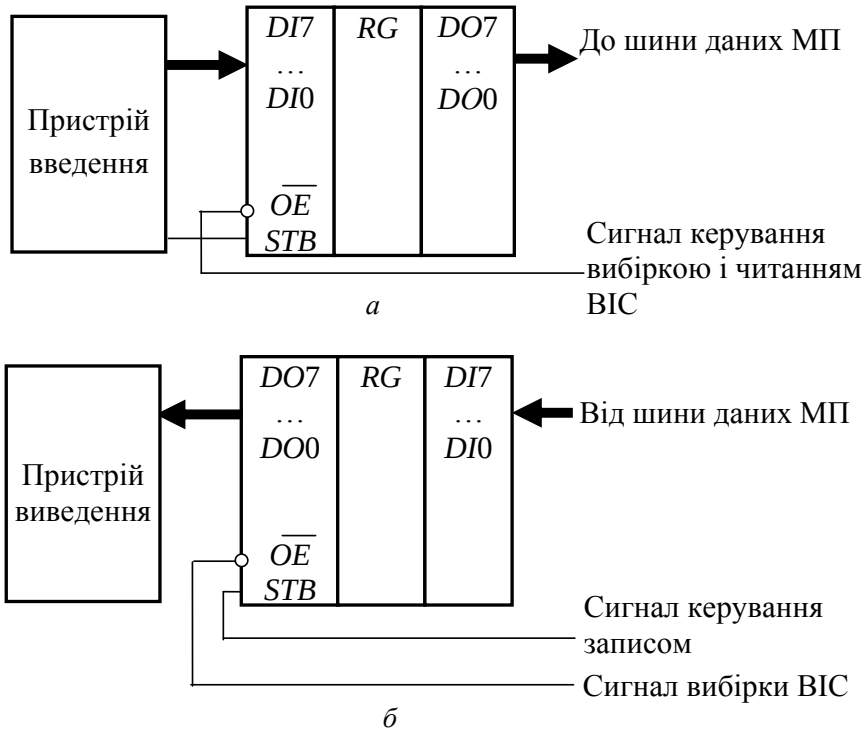


Рис. 6.1. Використання регістра КР580ИР82 для з'єднання:
а – із пристроєм уведення; *б* – із пристроєм виведення

Уведення або виведення даних можна здійснювати двома способами:

1) з використанням окремого адресного простору ПВВ;
2) з використанням спільного з пам'яттю адресного простору, тобто з відображенням на пам'ять.

У першому випадку введення і виведення даних виконується за командами *IN* та *OUT* (див. табл. 3.11).

Приклад 6.1. Виконати виведення даних на 16-розрядний порт з адресою 1000*H*.

Адреса порту займає два байти, тому для адресації порту треба використати непряму регістрову адресацію. Для цього треба адресу 1000*H* занести у регістр *DX*, а потім виконати команду виведення:

```
MOV DX, 1000H    ; Занести у DX число 1000H
OUT DX, AX       ; Вивести вміст AX на 16-розрядний порт
                 ; виведення з адресою, яка знаходиться у DX,
                 ; тобто AX → P16(DX)
```

Під час виконання команди *OUT DX, AX* на лініях *A*15–*A*0 шини адреси з'являється адреса порту:

*A*15	*A*14	*A*13	*A*12	*A*11	*A*10	*A*9	*A*8	*A*7	*A*6	*A*5	*A*4	*A*3	*A*2	*A*1	*A*0	
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	= 1000*H*,

установлюється низький рівень сигналу запису введення-виведення $\overline{IOW}$, і вміст акумулятора *AX* передається на 16-розрядну шину даних. Для фіксації даних (умісту *AX*) треба використати два 8-розрядні порти, один з яких з'єднаний з молодшою половиною шини даних, а другий – зі старшою. Сигнали керування записом і вибірки подаються паралельно на 2 порти.

Приклад 6.2. Виконати введення даних з 8-розрядного порту з адресою 32*H*.

Уведення даних здійснюється за командою введення:

```
IN AL, 32H          ; AL ← P8(32H)
```

Дії МП у цьому випадку аналогічні наведеним у прикл. 6.1. Відмінність полягає в тому, що активним стає сигнал $\overline{IOR}$ читання введення-виведення, і передача інформації здійснюється від порту до МП по молодшій половині шини даних *D*7–*D*0.

За другим способом адреси портів розташовуються у спільному з пам'яттю адресному просторі, і звернення до портів не відрізняється від звернення до комірки пам'яті.

Сигнали вибірки ВІС (див. рис. 6.1) конкретних портів формуються за допомогою дешифраторів. Адреса 16-розрядного порту P_{16} має бути парною, щоб звернення до неї відбулося за один цикл шини. Адреси 8-розрядних портів уведення-виведення P_8 можуть бути будь-якими (парними, непарними), але за парної адреси 8-розрядні порти потрібно з'єднати з молодшою половиною шини даних *D*7–*D*0, а за непарної – зі старшою половиною *D*15–*D*8.

Приклад 6.3. Розробити функціональну схему дешифратора на ВІС К155ИД7 для адресації восьми 8-розрядних і восьми 16-розрядних портів, причому адреси 8-розрядних портів брати непарними, а адреси 16-розрядних – парними.

Функціональну схему дешифратора показано на рис. 6.2.

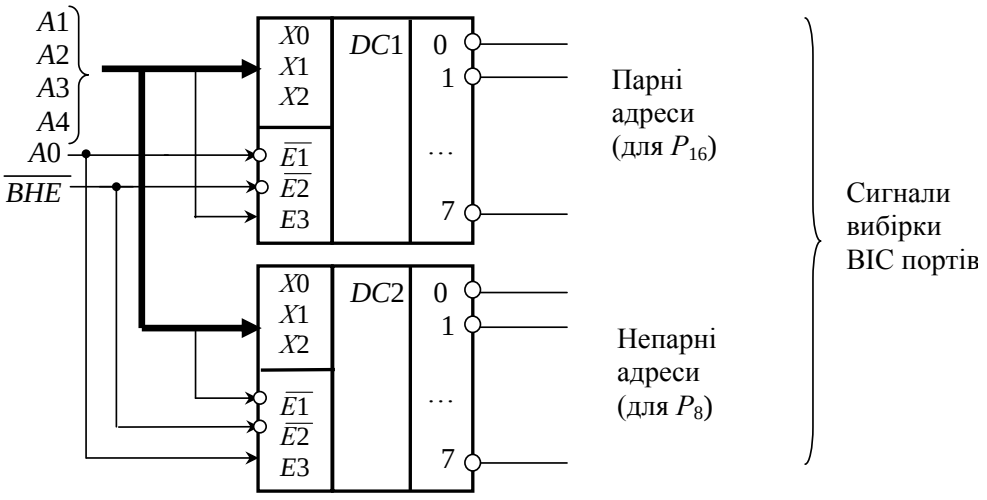


Рис. 6.2. Функціональна схема дешифратора портів

Схема містить дві ВІС дешифраторів *DC*1 і *DC*2. Із виходів дешифраторів сигнали надходять на входи $\overline{OE}$ відповідних портів. Отже, схема (див. рис. 6.2) дозволяє адресувати 16 портів. Усі вихідні сигнали 0–7 ВІС мають *H*-рівні, якщо не забезпечене надходження сигналів *L*-рівня на інверсні входи дозволу *E*1 і *E*2 та сигналу *H*-рівня на вхід *E*3. Інакше сигнал на тому виході *DC*, двійковий код номера якого визначається кодом на інформаційних входах *DC X*2, *X*1, *X*0, є активним, тобто має *L*-рівень. Сигнали на інших виходах мають *H*-рівень. Цей принцип роботи *DC* дозволяє в будь-який момент роботи МП звернутися лише до одного з портів. Визначимо тепер адреси портів.

Низькі рівні на виході *DC*1 з'являються при значенні сигналів на адресних лініях $A4 = 1$, $A0 = 0$ і сигналу вибірки старшого банку $\overline{BHE} = 0$. Сигнал на виході 0 *DC*1 буде активним (*L*-рівень) при адресі

*A*15	*A*14	*A*13	*A*12	*A*11	*A*10	*A*9	*A*8	*A*7	*A*6	*A*5	*A*4	*A*3	*A*2	*A*1	*A*0	
x	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	1	0	0	0	0	= 0010*H*,

на виході 1 – при адресі

x	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	1	0	0	1	0	= 0012*H*,

на виході 2 – при адресі

x	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	1	0	1	0	0	= 0014*H*,

……

на виході 7 – при адресі

x	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	1	1	1	1	0	= 001*EH*.

Низькі рівні на виході *DC*2 з'являються при $A4 = 0$, $A0 = 1$, $\overline{BHE} = 0$. Сигнал на виході 0 *DC*2 буде активним (*L*-рівень) при адресі

*A*15	*A*14	*A*13	*A*12	*A*11	*A*10	*A*9	*A*8	*A*7	*A*6	*A*5	*A*4	*A*3	*A*2	*A*1	*A*0	
x	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	0	0	0	0	1	= 0001*H*,

на виході 1 – при адресі

x	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	0	0	0	1	1	= 0003*H*,

……

на виході 7 – при адресі

x	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	0	1	1	1	1	= 000*FH*.

Зазначимо, що 8-розрядні порти з непарними адресами мають бути з'єднаними зі старшою половиною шини даних *D*15–*D*8.

Керування обміном. Існують три способи керування обміном:

1) програмний обмін;
2) обмін за перериванням;
3) обмін у режимі ПДП.

Програмний обмін ініціюється МП і здійснюється під його керуванням. Розрізняють *простий програмний обмін* і *програмний обмін за стробом готовності*. При простому програмному обміні вважається, що ПВВ у будь-який момент готовий до обміну за командами *IN* або *OUT*. У разі обміну за стробом готовності ПВВ сповіщає про свою готовність до обміну стробом. Наприклад, видача 8-розрядних даних супроводжується дев'ятим бітом – стробом. За такого обміну схема інтерфейсу містить тригер або порт керування для зберігання інформації про готовність зовнішнього пристрою до обміну. Процесор опитує відповідний розряд порту керування для визначення стану готовності зовнішнього пристрою.

Приклад 6.4. Розробити функціональну схему введення і виведення 8-розрядних даних за стробом готовності. Адреса порту введення – 02*H*, порту керування – 03*H*, порту виведення – 04*H*.

Функціональну схему обміну за стробом готовності показано на рис. 6.3. Схема містить: пристрій введення, з'єднаний з портом уведення; пристрій виведення, з'єднаний з портом виведення; порт керування для зберігання сигналів готовності пристроїв уведення і виведення. Пристрій введення має вісім інформаційних вихідних ліній і одну вихідну лінію стробу супроводження даних. Поява цього стробу сигналізує про те, що дані на інформаційних линіях дійсні (коректні). Пристрій виведення має вісім інформаційних вхідних ліній і одну вихідну лінію стробу підтвердження прийому даних. Поява цього стробу сигналізує про те, що дані прийняті пристроєм, і МП може передавати нову порцію даних. Порт керування зберігає інформацію про строби від двох пристроїв.

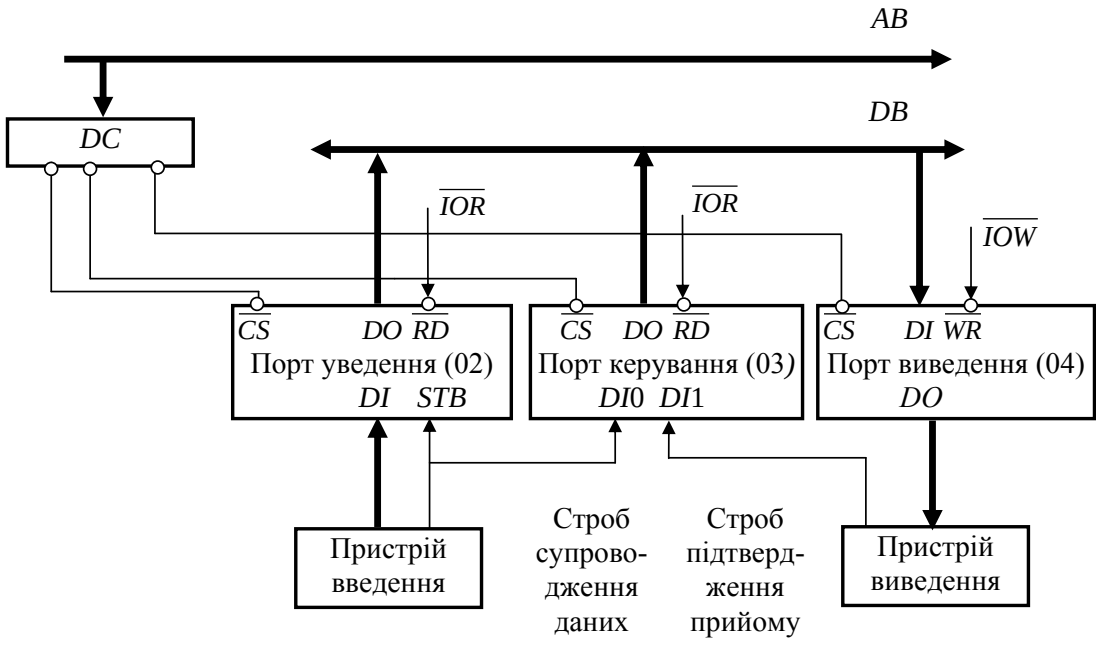


Рис. 6.3. Схема введення-виведення даних за стробом готовності

Програма введення за стробом готовності має такий вигляд:

```
M1:  IN AL, 03          ; AL ← порт керування (адреса 03)
     AND AL, 00000001B  ; Маскування всіх розрядів, крім D0
```

```
     JZ M1              ; Якщо D0 = 0 (порт не готовий), то на M1,
     IN AL, 02          ; інакше – введення інформації з порту
                        ; введення (адреса 02)
```

Програма виведення за стробом готовності має такий вигляд:

```
M2:  IN AL, 03          ; AL ← порт керування (адреса 03)
     AND AL, 00000010B  ; Маскування всіх розрядів, крім D1
     JZ M2              ; Якщо D1 = 0 (порт не готовий), то на M2,
                        ; інакше – виведення інформації на порт
     OUT 04, AL         ; виведення (адреса 04)
```

Якщо ПВВ має вбудований апаратний засіб для визначення готовності до обміну, про стан пристрою свідчить прапорець готовності *READY* або прапорець готовності/зайнятості *READY*/*BUSY*. Інформація про готовність пристроїв належить до *статусної інформації* і входить до складу слова стану пристрою. Іноді стан готовності і зайнятості ідентифікується окремими прапорцями: *READY* і *BUSY*. Прапорець *READY* замінює біт порту керування (див. рис. 6.3).

Рис. 6.4. Алгоритм програмного введення

Схему алгоритму програмного обміну даними за значенням прапорця *READY* показано на рис. 6.4.

Якщо ПВВ не готовий до обміну, то МП знаходиться у режимі програмного очікування готовності зовнішнього пристрою, виконуючи команди блоків 1 і 2. Після виявлення стану готовності МП передає дані за командами блока 3, а потім працює з продовженням основної програми. На читання статусної інформації та її аналіз МП витрачає декілька циклів роботи, що призводить до непродуктивних втрат його часу. Недоліками програмного обміну за стробом готовності є те, що цей спосіб обміну інформацією не дозволяє зовнішнім пристроям ініціювати обмін. Перевага програмного обміну полягає у простоті реалізації, а також у тому, що немає потреби у додаткових апаратних засобах.

Програмний обмін використовується для обміну з ПВВ, продуктивність яких менша від продуктивності МП.

Обмін за перериванням ініціюється ПВВ і здійснюється під керуванням МП. У цьому разі сигнал готовності ПВВ до обміну використовується як запит переривання і надходить до програмовного контролера

переривань (ПКП) (рис. 6.5). Уведення або виведення здійснюється за підпрограмою обробки запиту переривання.

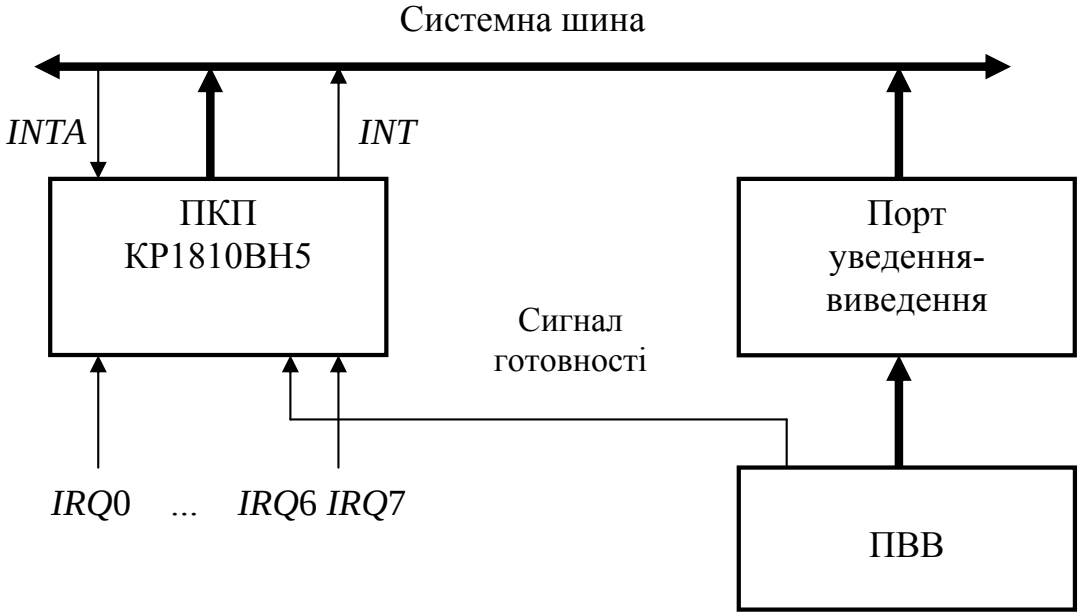


Рис. 6.5. Схема обміну за перериванням

Пристрій введення-виведення формує сигнал готовності *IRQ*, коли він готовий до обміну. ПКП (рис. 6.5) здатний сприйняти 8 сигналів *IRQ*7–*IRQ*0. На рис. 6.5 сигнал готовності ПВВ надходить на вхід *IRQ*6. Сигнал готовності ПВВ являє собою вихідний сигнал тригера, який фіксує стан готовності *READY*. На виході ПКП асинхронно з діями МП формується сигнал *INT*. Заздалегідь не відомо, у який момент і які периферійні пристрої ініціюють переривання. Реагуючи на сигнал *INT*, МП перериває виконання програми, ідентифікує пристрій, переходить до підпрограми обслуговування переривань роботи цього пристрою, а після її завершення відновлює виконання перерваної програми. За командою *INT* уміст програмного лічильника та прапорців автоматично запам’ятовується у стеку. Вміст акумулятора та РЗП необхідно занести у стек за допомогою команд *PUSH* у підпрограмі обробки переривання.

У кожному МП реалізовано особливу структуру системи переривань. Однак загальна послідовність обміну за перериванням містить такі дії:

– ПВВ генерує сигнал готовності, який викликає появу сигналу переривання, що подається на вхід *INT* МП;
– МП завершує виконання поточної команди і, якщо переривання дозволені (не замасковані), формує сигнал *INTA* підтвердження переривання;
– МП здійснює запам’ятовування вмісту акумулятора, програмного лічильника, РЗП у стеку;
– МП ідентифікує пристрій, що викликав переривання, і виконує відповідну підпрограму обслуговування переривання;

– за допомогою команд *POP* відновлюються значення вмісту акумулятора та РЗП зі стеку;
– за командою повернення з переривання *RET*, що є останньою командою підпрограми обслуговування переривання, відновлюються значення програмного лічильника та прапорців, і продовжується виконання перерваної програми.

Обмін за перериванням продуктивніший від програмного обміну, оскільки не потребує часу для опитування стану готовності ПВВ до обміну.

Обмін у режимі ПДП ініціюється ПВВ і здійснюється під керуванням контролера ПДП без участі МП. За необхідності обміну між ПВВ і пам'яттю немає потреби у пересиланні даних через МП. Дані за допомогою контролера ПДП пересилаються безпосередньо з ПВВ у пам'ять або навпаки. ПДП при виконанні операцій введення-виведення дозволяє значно збільшити швидкість передачі даних і підвищити ефективність використання засобів МП. Схему обміну в режимі ПДП показано на рис. 6.6. Контролер прямого доступу приймає запит від ПВВ, формує сигнал запиту захоплення шин МП *HOLD* і, отримавши від МП дозвіл *HLDA*, формує адреси пам'яті та керувальні сигнали $\overline{MEMR}$, $\overline{IOW}$ у разі читання пам'яті або $\overline{MEMW}$, $\overline{IOR}$ у разі запису в пам'ять.

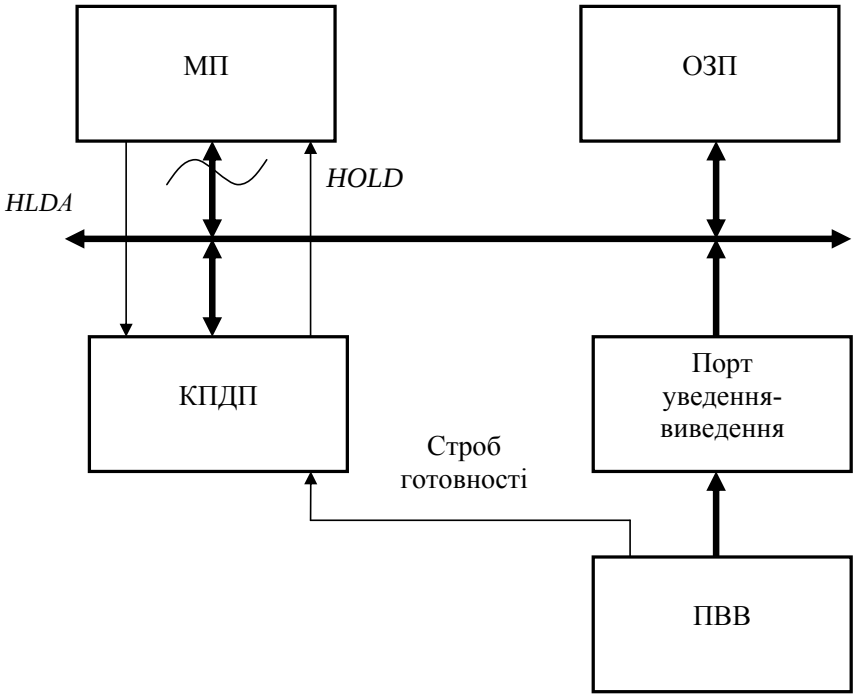


Рис. 6.6. Схема обміну у режимі ПДП

Інформацію про область пам'яті, що використовується при обміні у вигляді початкової адреси і довжини масиву, завантажують у контролер ПДП під час програмування. Продуктивність обміну в режимі ПДП є найвищою.

Перетворення форматів даних. Якщо розрядність даних, якими оперує МП, менша від розрядності даних, якими оперує ПВВ, то для узгодження

розрядності збільшується кількість портів уведення-виведення. Якщо розрядність даних, якими оперує МП, більша від розрядності даних, якими оперує ПВВ, то для узгодження розрядності виконується пакування даних, отриманих із декількох джерел, у одне слово потрібної розрядності або доповнення нулями. Для перетворення послідовного коду на паралельний і навпаки використовують контролер послідовного обміну.

Контрольні запитання

1. Як забезпечується зберігання інформації, що надходить від ПВВ?
2. Назвіть способи адресування портів уведення-виведення.
3. Наведіть порівняльну характеристику видів обміну.
4. Які існують типи програмного обміну?
5. Наведіть структурну схему обміну за стробом готовності.
6. Наведіть структурну схему обміну за перериванням.
7. Наведіть структурну схему обміну в режимі ПДП.
8. Як відбувається запам'ятовування вмісту акумулятора, РЗП, програмного лічильника та прапорців при обміні за перериванням?
9. У яких випадках доцільне застосування ПДП?

6.2. Програмовний паралельний інтерфейс

Програмовний паралельний інтерфейс (ППІ) КР580ВВ55 призначений для введення-виведення паралельної інформації у 8-байтовому форматі, що дозволяє реалізувати більшість відомих протоколів обміну по паралельних каналах. ВІС ППІ може використовуватися для з'єднання МП зі стандартним периферійним устаткуванням (дисплеєм, телетайпом, накопичувачем тощо).

Структурну схему ВІС показано на рис. 6.7, а її графічне позначення – на рис. 6.8.

До складу ВІС входять:

– двонапрямлений 8-розрядний буфер даних *Bufer of Data* (*BD*), що з'єднує лінії даних ВІС із системною шиною даних;
– блок керування читанням/записом *Read*/*Write Control Unit* (*RWCU*), що забезпечує керування зовнішнім і внутрішнім передаванням даних і керувальних слів;
– три 8-розрядні порти введення-виведення (*Port A, Port B, Port C*) для обміну інформацією, причому порт С поділений на два 4-розрядні: *C′* (*PC*7–*PC*4) і *C′′*(*PC*3–*PC*0). Порти *A* і *C′* об'єднані у групу *A*, порти *B* і *C′′* – у групу *B*.

Схема ВІС містить також блоки керування групою *A Control Unit A* (*CUA*) та групою *B* (*CUB*), що формують сигнали керування для відповідних груп.

Блок *RWCU* (*Register of Control Word Unit*) містить регістр керувального слова, який зберігає керувальні слова, що надходять від МП.

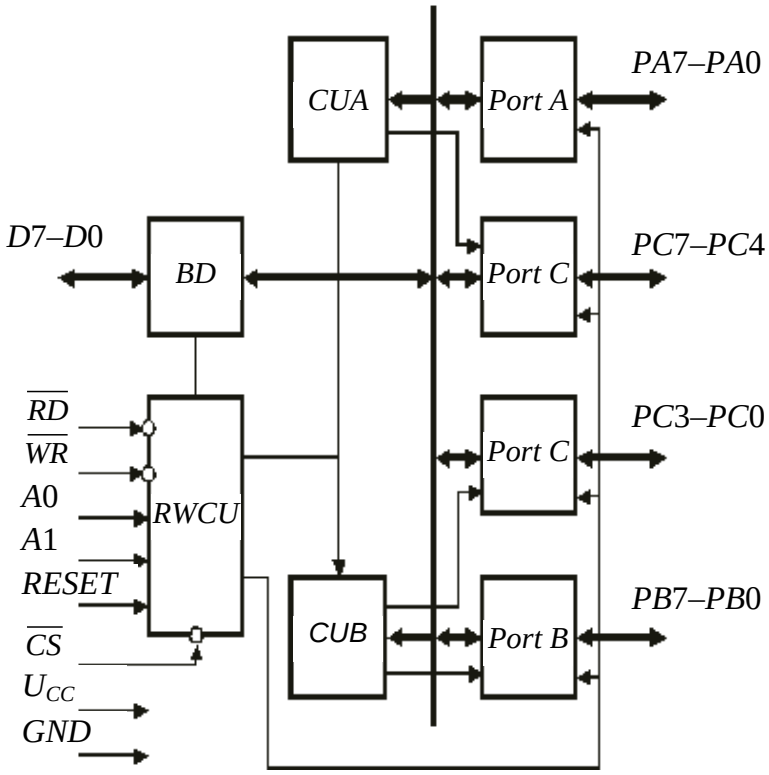


Рис. 6.7. Структурна схема ВІС КР580ВВ55

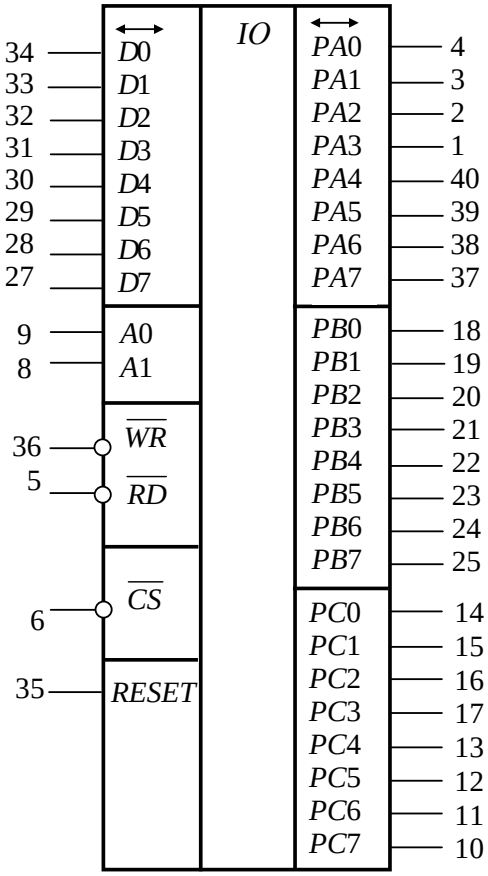


Рис. 6.8. Графічне позначення ВІС КР580ВВ55

Адресні розряди *A*1, *A*0 дозволяють обирати один з портів або регістр керувального слова *RCW* (табл. 6.1).

Таблиця 6.1. **Вибір портів ВІС КР580ВВ55**

*A*1	*A*0	Порт
0	0	*A*
0	1	*B*
1	0	*C*
1	1	*RCW*

Сигнал керування третім станом шини даних $\overline{CS}$, сигнал читання $\overline{RD}$, сигнал запису $\overline{WR}$ та сигнал скидання *RESET* подаються на блок *RWCU* і разом із адресними сигналами *A*0, *A*1 задають вид операції, що виконується (табл. 6.2).

Таблиця 6.2. **Визначення виду операцій залежно від сигналів керування та адресних розрядів *A*1, *A*0**

Операція	$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	*A*1	*A*0
Запис керувального слова із МП	0	1	0	1	1
Запис даних у порт *A*	0	1	0	0	0
Запис даних у порт *B*	0	1	0	0	1
Запис даних у порт *C*	0	1	0	1	0
Зчитування даних із порту *A*	0	0	1	0	0
Зчитування даних із порту *B*	0	0	1	0	1
Зчитування даних із порту *C*	0	0	1	1	0
Від'єднання ВІС від *D*7–*D*0	1	*x*	*x*	*x*	*x*

Примітка. *x* – будь-яке значення (0 або 1).

Призначення виводів ВІС наведено у табл. 6.3.

Таблиця 6.3. **Призначення виводів ВІС КР580ВВ55**

Позначення виводу	Номер виводу	Призначення виводів
1	2	3
*D*7–*D*0	27, 28, 29, 30, 31, 32, 33, 34	Вхід/вихід даних
$\overline{RD}$	5	Читання: *L*-рівень сигналу дозволяє зчитування інформації з регістра, що адресується розрядами *A*0, *A*1 на лінії *D*7–*D*0
$\overline{WR}$	36	Запис: *L*-рівень сигналу дозволяє запис інформації із шини *D*7–*D*0 у порт паралельного інтерфейсу, що адресується розрядами *A*0, *A*1
*A*0, *A*1	9, 8	Входи для адресування внутрішніх регістрів ППІ
RESET	35	Скидання: *H*-рівень сигналу скидає регістр керувального слова і встановлює усі порти в режим уведення

Продовження табл. 6.3

1	2	3
$\overline{CS}$	6	Вхід вибірки мікросхеми: *L*-рівень сигналу з'єднує шину даних *D*7-*D*0 ВІС із системною шиною
*PA*7–*PA*0	37, 38, 39, 40, 1, 2, 3, 4	Вхід/вихід порту *A*
*PB*7–*PB*0	15, 24, 23, 22, 21, 20, 19, 18	Вхід/вихід порту *B*
*PC*7–*PC*0	10, 11, 12, 13, 17, 16, 15, 14	Вхід/вихід порту *C*
U_{CC}	26	Вивід напруги живлення +5 В
GND	7	Спільний вивід 0 В

Функціональну схему з'єднання ВІС із системною шиною МП показано на рис. 6.9.

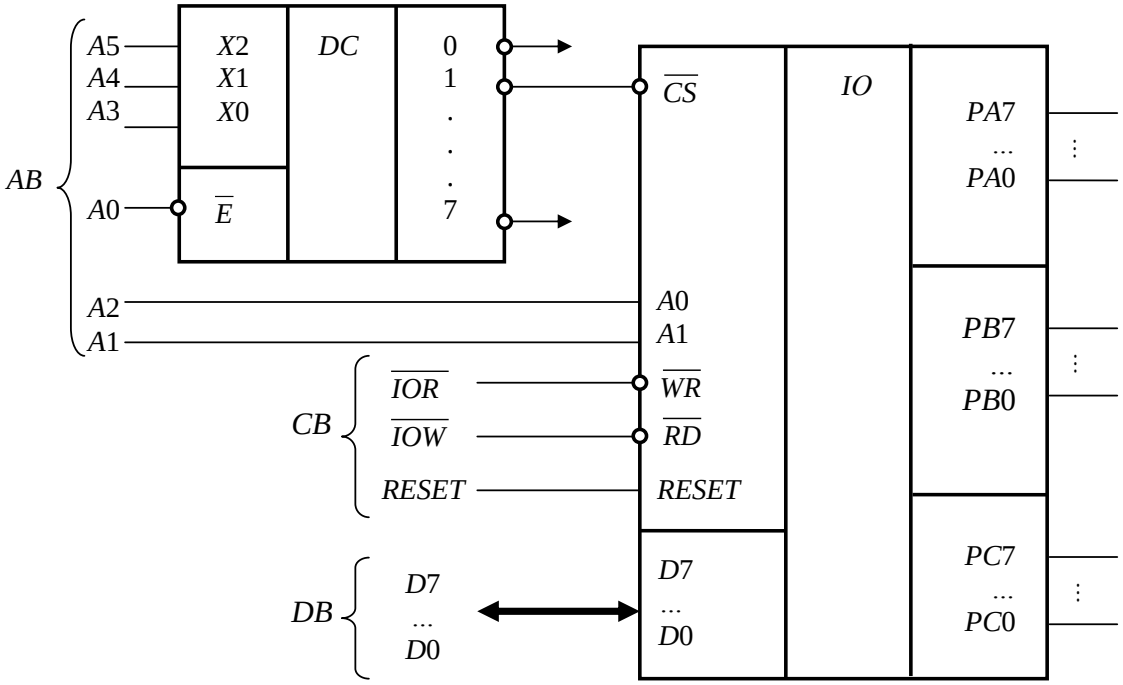


Рис. 6.9. Функціональна схема з'єднання ВІС КР580ВВ55 із системною шиною

Відповідно до схеми (рис. 6.9) і табл. 6.2 визначаються адреси портів і регістра керувального слова *RCW* (табл. 6.4).

Таблиця 6.4. **Адреси портів і регістра *RCW***

Порт	*A*7	*A*6	*A*5	*A*4	*A*3	*A*2	*A*1	*A*0	Адреса
A	0	0	0	0	1	0	0	0	08*H*
B	0	0	0	0	1	0	1	0	0*AH*
C	0	0	0	0	1	1	0	0	0*CH*
RCW	0	0	0	0	1	1	1	0	0*EH*

Програмування ВІС полягає у завантаженні керувального слова режиму при $A1 = 1$, $A0 = 1$. Формат керувального слова режиму показано

на рис. 6.10. Керувальне слово визначає один з трьох режимів портів паралельного інтерфейсу: режим 0 – основний режим введення-виведення, режим 1 – режим введення-виведення за стробом готовності, режим 2 – режим двонапрямленої передачі інформації.

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
1	*M*1	*M*0	*IOA*	*IOC′*	*M*	*IOB*	*IOC′′*

Рис. 6.10. Формат керувального слова режиму

На рис. 6.10 позначено:

– біти *M*1, *M*0 задають режим групи *A*. При *M*1*M*0 = 00 – режим 0, 01 – режим 1, 1*x* – режим 2;
– біт *IOA* задає режим уведення або виведення порту *A* (1 – введення інформації, 0 – виведення);
– біт *IOC′* задає режим уведення або виведення порту *C′* (1 – введення інформації, 0 – виведення);
– біт *M* задає режим групи *B* (0 – режим 0, 1 – режим 1);
– біт *IOB* задає режим уведення або виведення порту *B* (1 – введення інформації, 0 – виведення);
– біт *IOC′* задає режим уведення або виведення порту *C′′* (1 – введення інформації, 0 – виведення).

Керувальне слово може встановлювати різні режими роботи для кожного з портів. Порт *A* може працювати в будь-якому з трьох режимів, порт *B* – у режимах 0 та 1. Порт *C* можна використовувати для передачі даних тільки в режимі 0, в інших режимах його застосовують для передачі керувальних сигналів, що супроводжують процес обміну по портах *A* і *B*.

Окремі розряди порту *C* можна встановлювати або скидати програмно за допомогою керувального слова встановлення/скидання, формат якого показано на рис. 6.11.

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
0	*x*	*x*	*x*	*N*2	*N*1	*N*0	*S/R*

Рис. 6.11. Формат керувального слова встановлення/скидання порту *C*

На рис. 6.11 позначено:

– біти *N*2, *N*1, *N*0 задають номер розряду, який треба встановити або скинути. Якщо значення цих бітів дорівнює 000 – обирається розряд *PC*0, 001 – *PC*1, 010 – *PC*2, 011 – *PC*3, 100 – *PC*4, 101 – *PC*5, 110 – *PC*6, 111 – *PC*7;
– біт *S/R* задає режим установлення або скидання розряду порту *C*, який обрано значеннями *N*2, *N*1, *N*0. При *S/R* = 1 відбувається встановлення розряду, при 0 – скидання.

Приклад 6.5. Сформувати імпульс тривалістю *n* (у мікросекундах). Адреси ВІС КР580ВВ55 задано в табл. 6.4.

Для того щоб сформувати імпульс заданої тривалості, треба встановити розряд *PC*4 у стан логічної одиниці, потім виконати підпрограму затримки на *n* (у мікросекундах) та скинути розряд *PC*4.

Визначимо керувальні слова для встановлення і скидання розряду *PC*4. Керувальне слово встановлення розряду *PC*4 згідно з рис. 6.11 показано на рис. 6.12.

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
0	0	0	0	1	0	0	1

Рис. 6.12. Формат керувального слова встановлення *PC*4

У 16-ковій системі числення воно дорівнює 09*H*.

Керувальне слово скидання розряду *PC*4 показано на рис. 6.13:

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
0	0	0	0	1	0	0	0

Рис. 6.13. Формат керувального слова скидання *PC*4

У 16-ковій системі числення воно дорівнює 08*H*.

Програма формування імпульсу тривалістю *n* (у мікросекундах) на виході *PC*4 порту *C*:

```
        MOV AL, 09  ; Формування керувального слова
                    ; встановлення розряду PC4
        OUT 0EH, AL ; Запис умісту AL до регістра
                    ; керувального слова
        CALL DELAY  ; Затримка часу на n (у мікросекундах)
        MOV AL, 08  ; Формування керувального слова
                    ; скидання розряду PC4
        OUT 0EH, AL ; Виведення до регістра керувального
                    ; слова
                    ; Підпрограма затримки на n
                    ; (у мікросекундах)
DELAY : MOV CX, 134 ; CX←134 (4 такти)
D:      LOOP D      ; CX←CX – 1, якщо не 0, то перехід
                    ; на D (при виконанні переходу
                    ; на мітку D команда виконується за 16
                    ; тактів, якщо CX = 0 – за 4 такти)
        RET         ; Повернення з підпрограми (8 тактів)
```

Визначимо тривалість затримки у цьому прикладі. У коментарі до команд підпрограми затримки у дужках запишемо час виконання команд у тактах (див. табл. 3.11). Загальна кількість тактів, потрібна для виконання підпрограми *DELAY*, дорівнює

$$4 + 134 \cdot 16 + 4 + 8 = 2\ 160.$$

При тактовій частоті 5 МГц тривалість одного такту дорівнює 200 нс. Тоді значення *n* визначається так:

$$n = 0{,}2 \cdot 2\ 159 = 431{,}8 \text{ мкс}.$$

Для кожної групи *A* або *B* у ВІС є тригер дозволу переривання *INTE*, установлення/скидання якого здійснюється керувальним словом установлення/скидання визначеного розряду порту *C*. Якщо тригер дозволу переривання відповідної групи встановлений (*INTE* = 1), то паралельний інтерфейс може сформувати запит переривання у разі готовності ПВВ до обміну.

Розглянемо режими роботи портів ППІ.

Режим 0 застосовується при синхронному обміні або програмній організації асинхронного обміну. У цьому режимі ВІС являє собою пристрій, що складається із чотирьох портів (два 8-розрядні і два 4-розрядні), які можуть незалежно налагоджуватися на введення або виведення інформації. Виведення інформації здійснюється за командою *OUT* із фіксацією виведеної інформації у регістрах портів, а введення – за командою *IN* без запам'ятовування інформації.

Приклад 6.6. Установити порт *A* у режим уведення 0, порт *B* – у режим виведення 0, порт *C′* – у режим уведення 0, порт *C′′* – у режим виведення 0, а потім увести або вивести інформацію через порти *A* та *B* відповідно.

Керувальне слово режиму у цьому разі показано на рис. 6.14.

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0	
1	*M*1	*M*0	*IOA*	*IOC′*	*M*	*IOB*	*IOC′′*	
1	0	0	1	1	0	0	0	= 98*H*

Рис. 6.14. Формат керувального слова режиму

Програма має такий вигляд:

```
MOV AL, 98H      ; Формування керувального слова режиму 98H у AL
OUT OEH, AL      ; Запис до регістра RCW
       …
IN AL,08H        ; Уведення з порту A
       …
OUT 0AH, AL      ; Виведення на порт B
```

Зазначимо, що в цьому прикладі адреси портів визначаються за схемою рис. 6.9.

Режим 1 забезпечує однонапрямлений обмін інформацією МП з ПВВ за стробом готовності. Інформація передається по портах *A* і *B*, а лінії порту *C* керують передачею. Роботу порту в режимі 1 супроводжують три керувальні сигнали. Якщо один з портів запрограмовано на режим 1, то інші 13 інтерфейсних ліній можна використовувати у режимі 0. Якщо обидва порти запрограмовано на режим 1, то дві інтерфейсні лінії порту *C*, що залишилися, можуть бути налагоджені на введення або виведення. Призначення розрядів порту *C* при введенні даних із портів *A* і *B* у режимі 1 показано на рис. 6.15.

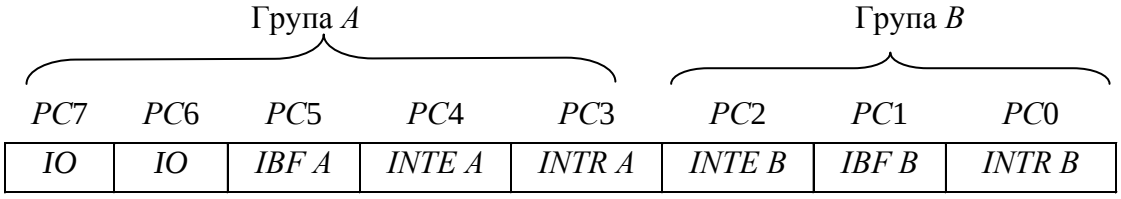


Рис. 6.15. Призначення розрядів порту *C* при введенні даних із портів *A* і *B* у режимі 1: *IBF (Input Buffer Full)* – вихідний сигнал ВІС, що повідомляє про наповненість вхідного буфера порту даними; *INTR (INTeRrupt)* – вихідний сигнал, що повідомляє про завершення приймання інформації; *INTE (INTerrupt Enable)* – сигнал дозволу переривання (вхід стробу приймання)

Приклад схеми з'єднання пристрою введення з портом *A*, пристрою виведення – з портом *B* у режимі 1 показано на рис. 6.16.

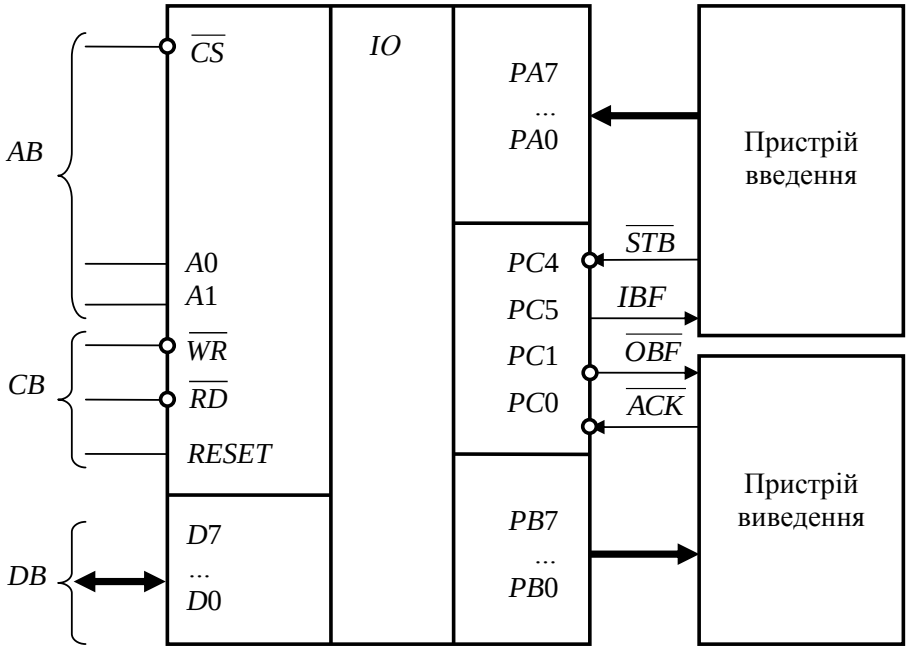


Рис. 6.16. Схема з'єднання пристрою введення з портом *A*, пристрою виведення – з портом *B* у режимі 1

Уведення даних у режимі 1 здійснюється по каналу *A*, а керувальні сигнали передаються по лініях *PC*4 і *PC*5. Пристрій введення видає строб приймання $\overline{STB}$, який указує на готовність до введення інформації. Цей строб надходить на вхід дозволу переривання від каналу *A* (*PC*4). Вихідний сигнал *IBF* з виводу *PC*5 використовується для підтвердження прийому. Він формується за спадом $\overline{STB}$ і повідомляє пристрій введення про закінчення приймання даних.

Крім показаних на рис. 6.16 сигналів, ППІ формує також сигнал запиту переривання *INTR*, який інформує МП про завершення приймання інформації. У разі обміну за перериванням цей сигнал використовується

як запит переривання, а при програмному обміні може ігноруватися. Високий рівень цього сигналу встановлюється, якщо $\overline{STB} = 1$, $IBF = 1$. Нульовий рівень сигналу *INTR* установлюється з надходженням сигналу $\overline{IOR}$.

Призначення розрядів порту *C* у режимі виведення 1 показано на рис. 6.17. Сигнал $\overline{OBF}$ (*Output Buffer Full*) – вихідний сигнал, що повідомляє про заповненість вихідного буфера порту даними.

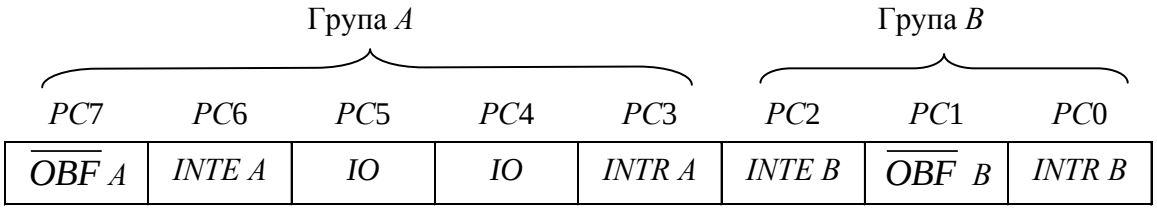


Рис. 6.17. Призначення розрядів порту *C* при виведенні даних на порти *A* і *B* у режимі 1

Інші сигнали мають такий самий сенс, що і на рис. 6.15.

Приклад виведення даних на порт *B* у режимі 1 показано на рис. 6.16. Для виведення даних у цьому режимі використовуються такі керувальні сигнали: $\overline{OBF}$ – вихідний сигнал, що формується за фронтом $\overline{WR}$ і повідомляє ПВВ про готовність до виведення; $\overline{ACK}$ – вхідний сигнал, що підтверджує приймання інформації з ВІС інтерфейсу; *INTR* – вихідний сигнал ВІС, що повідомляє МП про завершення виведення. Сигнал *INTR* установлюється в одиницю при $\overline{OBF} = 1$, $\overline{ACK} = 1$ і скидається у нуль сигналом $\overline{IOW}$ у разі запису даних у паралельний інтерфейс.

Розряди *PC*6, *PC*7 при введенні (див. рис. 6.15) та *PC*5, *PC*4 при виведенні (див. рис. 6.17) не беруть участі у керуванні обміном і можуть бути запрограмовані на просте введення або виведення (*IO*). Введення здійснюється читанням порту *C*, а виведення – записом керувальних слів установлення/скидання окремих розрядів (див. рис. 6.11).

Обмін за стробом готовності може здійснюватися за перериванням або за програмою. Під час обміну за перериванням сигнал *INTR* надходить до системи переривання та ініціює обмін. Під час обміну за програмою готовність портів *A* або *B* визначається опитуванням *INTR A* або *B* відповідно.

Приклад 6.7. Написати програму встановлення порту *A* в режим уведення 1 та ввести дані за стробом готовності. Адреси порту *A* визначаються за рис. 6.9.

Керувальне слово режиму у цьому випадку показано на рис. 6.18.

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0	
1	*M*1	*M*0	*IOA*	*IOC′*	*M*	*IOB*	*IOC′′*	
1	0	1	1	0	0	0		0 = 0B0*H*

Рис. 6.18. Керувальне слово режиму

Програма має такий вигляд:

```
    MOV AL, 0B0H     ; Формування керувального слова режиму в AL
    OUT 0EH, AL      ; Запис до регістра RCW ВІС КР580ВВ55
    MOV AL, 09       ; Формування керувального слова
                     ; встановлення розряду PC4 (INTE A)
                     ; дозвіл переривань
    OUT 0EH, AL      ; Запис вмісту AL до регістра керувального слова
            …
M1: IN AL, 0CH       ; AL ← вміст порту C
    AND AL, 00001000B ; Маскування всіх розрядів, крім PC3 (INTR A)
    JZ M1            ; Якщо дані не готові, то на M1,
    IN AL, 08H       ; інакше – введення інформації з порту A
```

Режим 2 забезпечує двонапрямлену передачу інформації з порту *A* до зовнішнього пристрою і навпаки. Процес обміну супроводжують 5 керувальних сигналів, що подаються по лініях *PC*7–*PC*3 (див. рис. 6.17); 11 інтерфейсних ліній, що залишилися, можуть налагоджуватися на режим 0 або 1. Призначення розрядів порту *C* у режимі 2 показано на рис. 6.19, а схему з'єднання з ППІ – на рис. 6.20.

*PC*7	*PC*6	*PC*5	*PC*4	*PC*3	*PC*2	*PC*1	*PC*0
$\overline{OBF}$ *A*	*INTE*1	*IBF A*	*INTE*2	*INTR A*	*x*	*x*	*x*

Залежать від режиму порту *B*

Рис. 6.19. Призначення розрядів порту *C* у режимі 2

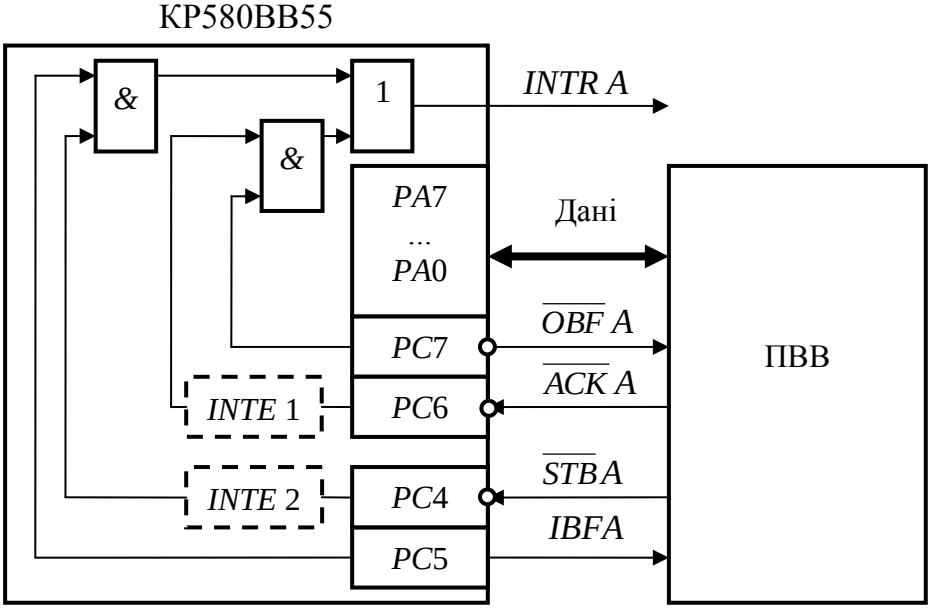


Рис. 6.20. Схема приєднання ПВВ до ВІС КР580ВВ55 у режимі 2

Призначення керувальних сигналів у режимі 2 аналогічне призначенню сигналів у режимі 1. Керування формуванням внутрішнього

сигналу *INTE* для операції введення здійснюється по лінії *PC*4, для операції виведення – по лінії *PC*6.

Вивід ВІС *INTR A* використовують як запит переривань як при введенні, так і при виведенні інформації. Розподіл сигналів по інтерфейсних лініях, керувальне слово режиму 2 і часові діаграми роботи ілюструє рис. 6.21.

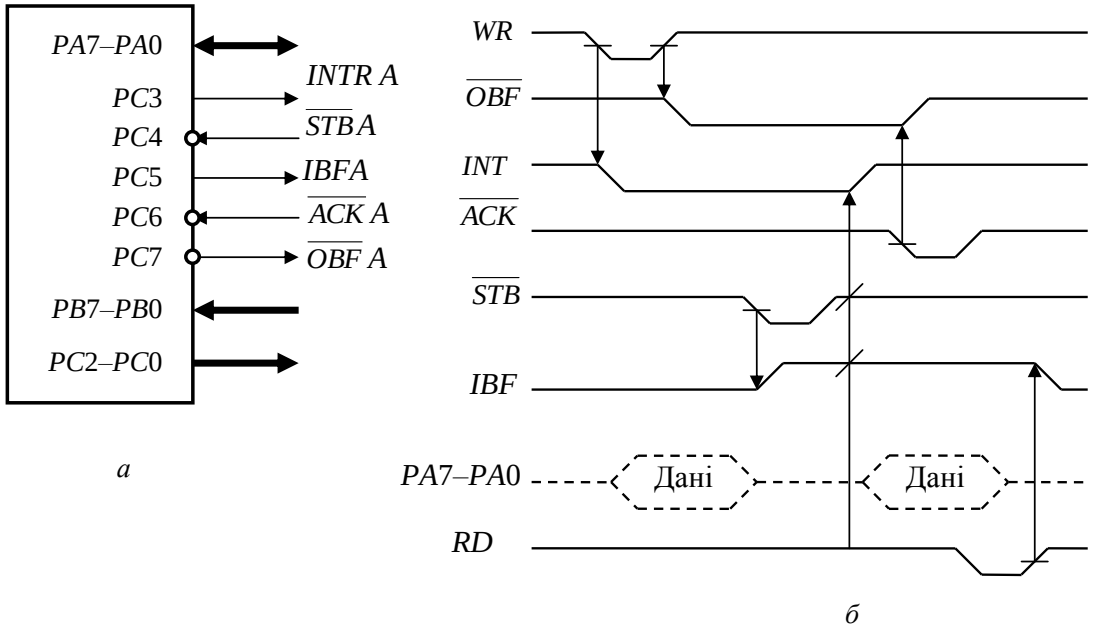


Рис. 6.21. Робота ВІС у режимі 2: *а* – розподіл сигналів по інтерфейсних лініях; *б* – часові діаграми роботи

Приклад 6.8. Написати програму встановлення порту *A* у режим 2, а потім здійснити введення та виведення інформації через порт *A* в цьому режимі.

Керувальне слово режиму в цьому випадку буде дорівнювати 0*C*0*H* (рис. 6.22).

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
1	*M*1	*M*0	*IOA*	*IOC´*	*M*	*IOB*	*IOC´´*
1	1	0	0	0	0	0	0

Рис. 6.22. Керувальне слово режиму

Програма двонапрямленого введення-виведення за стробом готовності має спочатку виявити готовність порту до введення або виведення за одиничним станом сигналу *INTR A* (лінія *PC*3), а потім установити, які саме дані готові – для введення (одиничний стан лінії *PC*4) чи виведення (одиничний стан лінії *PC*6). Після цього можна здійснювати обмін даними. Програма має такий вигляд:

```
    MOV AL, 0C0H       ; Формування керувального слова режиму в AL
    OUT 0EH, AL        ; Запис до регістра RCW ВІС КР580ВВ55
            …
M1: IN AL, 0CH         ; AL ← вміст порту C
    MOV BL, AL         ; Зберігання вмісту AL у регістрі BL
    AND AL, 00001000B  ; Маскування всіх розрядів, крім PC3
                       ; (INTR A)
```

```
    JZ M1              ; Якщо дані не готові, то на M1
    MOV AL, BL         ; AL ← вміст порту C
    AND AL, 00010000B ; Маскування всіх розрядів, крім PC4
                       ; (INTR A)
    JZ M2              ; Якщо дані не готові для введення,
                       ; то перехід на виведення,
    IN AL, 08H         ; інакше – введення інформації
                       ; з порту A
    JMP M3
M2: OUT 08H, AL        ; Виведення інформації на порт A
M3:                    ; Продовження програми
```

Контрольні запитання

1. Укажіть призначення ВІС ППІ КР580ВВ55.
2. Опишіть режими роботи ППІ.
3. Назвіть можливі комбінації ввімкнення портів ВІС КР580ВВ55.
4. Які порти паралельного інтерфейсу можуть працювати у всіх можливих режимах?
5. Запишіть керувальне слово для роботи паралельного інтерфейсу в режимі 0 при налагодженні портів *A* і *B* на виведення, порту *C* – на введення.
6. Поясніть принцип установлення/скидання розрядів порту *C*.

6.3. Програмовний інтерфейс клавіатури та індикації

Програмовний інтерфейс клавіатури та індикації КР580ВВ79 призначений для реалізації обміну інформацією між МП і матрицею клавіш (датчиків) та індикації. На відміну від ВІС паралельного інтерфейсу КР580ВВ55 (див. підрозд. 6.2), який може використовуватись для будь-якого пристрою, що здійснює введення-виведення даних у паралельному форматі, програмовний інтерфейс клавіатури та індикації спеціалізований і призначений для обміну інформацією лише з деякими типами клавіатури та індикаторів. Структурну схему ВІС показано на рис. 6.23.

Схема містить: двонапрямлений 8-розрядний буфер даних *BD*, що з'єднує лінії даних ВІС із системною шиною даних; блок *RWCU*, що забезпечує керування зовнішнім і внутрішнім передаванням даних та керувальних слів; блок керування; блок інтерфейсу індикації; блок інтерфейсу клавіатури.

Блок керування містить схему керування та синхронізації і лічильник сканування *CT*. Схема керування та сигналізації формує сигнали, які керують усіма блоками ВІС, сигнали внутрішньої синхронізації та сигнал $\overline{BD}$ для гасіння індикатора під час зміни символів. До складу схеми входить подільник частоти з програмовним коефіцієнтом ділення для генерації внутрішніх імпульсів синхронізації частотою до 100 кГц.

Лічильник сканування формує коди на лініях *S*3–*S*0 для опитування матриці клавіатури та керування індикацією. При цьому залежно від керувальних слів можна налагоджувати схеми видачі стану лічильника сканування або на безпосереднє виведення вмісту чотирьох молодших розрядів лічильника, або на виведення вмісту двох молодших розрядів через дешифратор із чотирма виходами.

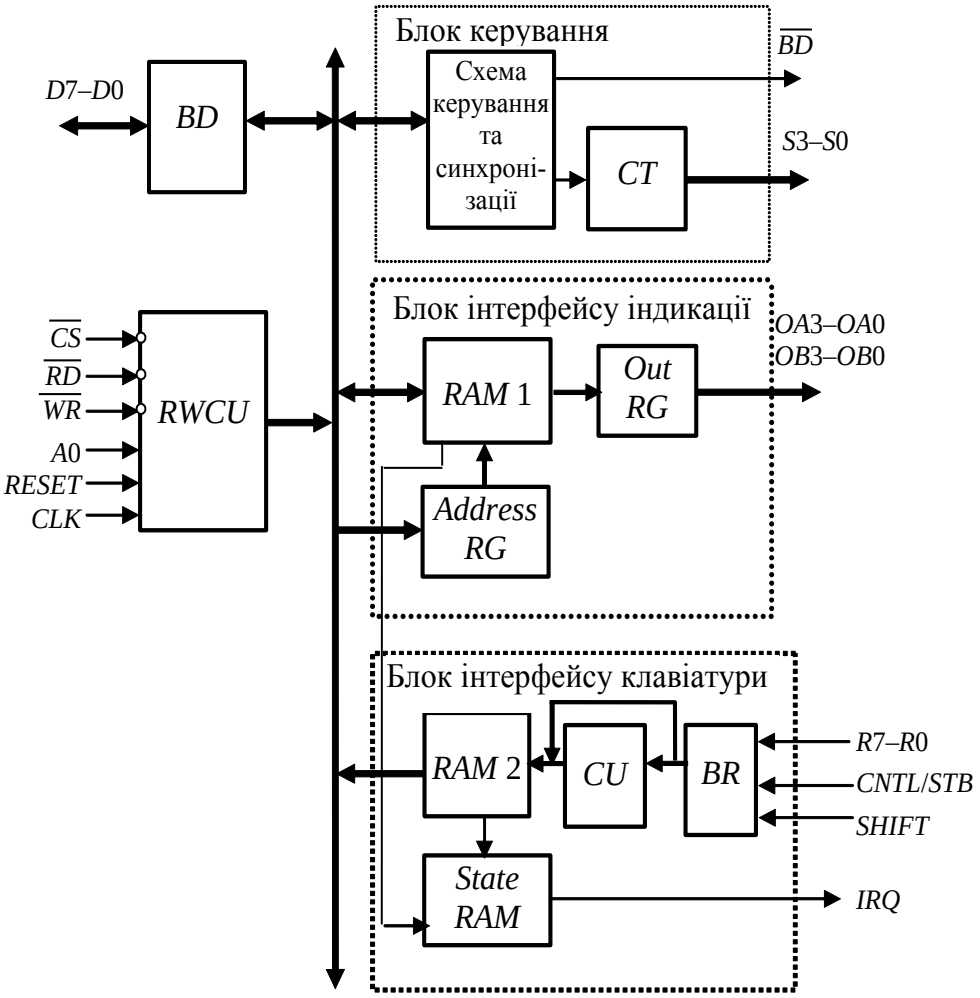


Рис. 6.23. Структурна схема програмовного інтерфейсу клавіатури індикації

Блок інтерфейсу індикації містить ОЗП індикації *RAM* 1 інформа-ційною ємністю 16×8, адресний регістр *Address RG* та вихідний регістр *Out RG*. ОЗП складається з двох незалежних частин по 16 4-розрядних слів кожний та зберігає коди символів для індикації на 8 або 16 знакомісць. Тип індикації задається відповідним керувальним словом. Дані з ОЗП передаються через вихідні регістри на виходи *OA*3–*OA*0 (старша частина 8-розрядного слова) та *OB*3–*OB*0 (молодша частина).

До блока інтерфейсу клавіатури входить ОЗП *RAM* 2 клавіатури/датчиків, буфер повернення *BR*, схема усунення брязкоту контактів *CU*, схема аналізу стану ОЗП *State RAM*.

Позначення виводів дано в табл. 6.5.

Таблиця 6.5. **Призначення виводів ВІС КР580ВВ79**

Позначення виводу	Номер виводу	Призначення виводів
*D*0–*D*7	12, 13, 14, 15, 16, 17, 18, 19	Вхід/вихід даних
CLK	3	Вхід синхросигналів. Частота не має перевищувати 3,2 МГц
$\overline{RD}$	10	Читання: *L*-рівень сигналу дозволяє зчитування інформації з регістра, що адресується розрядом *A*0 і вказаний у попередньому керувальному слові
$\overline{WR}$	11	Запис: *L*-рівень сигналу дозволяє запис інформації із шини *D*7–*D*0 у порт ВІС, що адресується розрядом *A*0 і вказаний у попередньому керувальному слові
*A*0	21	При одиниці у ВІС передається керувальне слово або з нього зчитується слово стану. При нулі – передаються дані
RESET	9	Скидання: *H*-рівень сигналу скидає ВІС у початковий стан
$\overline{CS}$	22	Вхід вибірки мікросхеми: *L*-рівень сигналу з'єднує шину даних *D*7–*D*0 ВІС із системною шиною
*S*0–*S*3	32, 33, 34, 35	Лінії сканування як клавіатури (матриці датчиків), так і позицій дисплею можуть працювати в режимі лічильника або інверсного дешифратора
*R*0–*R*7	38, 39, 1, 2, 5, 6, 7, 8	Лінії повернення. З'єднуються з лініями сканування через клавіатуру (матрицю датчиків). Натискання клавіші призводить до появи нуля на одній з ліній повернення
SHIFT	36	Зсув. Стан цієї лінії запам'ятовується в коді клавіші і може бути використаний як ознака переключення клавіатури
CNTR/STB	37	У режимі клавіатури використовується так само, як і лінія *SHIFT*. У режимі введення за стробом використовується як вхід стробу (уведення здійснюється за переднім фронтом *STB*)
*OA*0–*OA*3 *OB*0–*OB*3	27, 26, 25, 24 31, 30, 29, 28	Виходи регістрів даних дисплею. Можуть бути використані як один 8-розрядний або два 4-розрядні виходи
$\overline{BD}$	23	Гасіння дисплея при переключеннях цифр або при виданні керувального слова очищення дисплея
U_{CC}	40	Вивід напруги живлення +5 В
GND	20	Спільний вивід 0 В

Блок інтерфейсу клавіатури забезпечує введення інформації через лінії повернення (*R*7–*R*0) з клавіатури. Збереження введеної інформації здійснюється у ОЗП *RAM* 2, який являє собою стек ємністю 8×8 біт. Вхідні лінії *R*7-*R*0 мають високий внутрішній опір, що дає можливість безпосереднього при'єднання до них матриці клавіатури або датчиків. Для забезпечення режиму введення даних із датчиків за стробом готовності передбачено лінію *CNTL/STB*. Виходи буфера *BR* з'єднані з входами схеми усунення брязкоту контактів, яка виявляє заборонені ситуації при натисканні клавіш і не допускає повторного введення коду клавіш, що може статися внаслідок брязкоту контактів. Схема аналізу стану ОЗП формує статусну інформацію про роботу ОЗП та сигнал запиту переривання *IRQ*.

Функціональну схему прикладу з'єднання ВІС із системною шиною МПС показано на рис. 6.24.

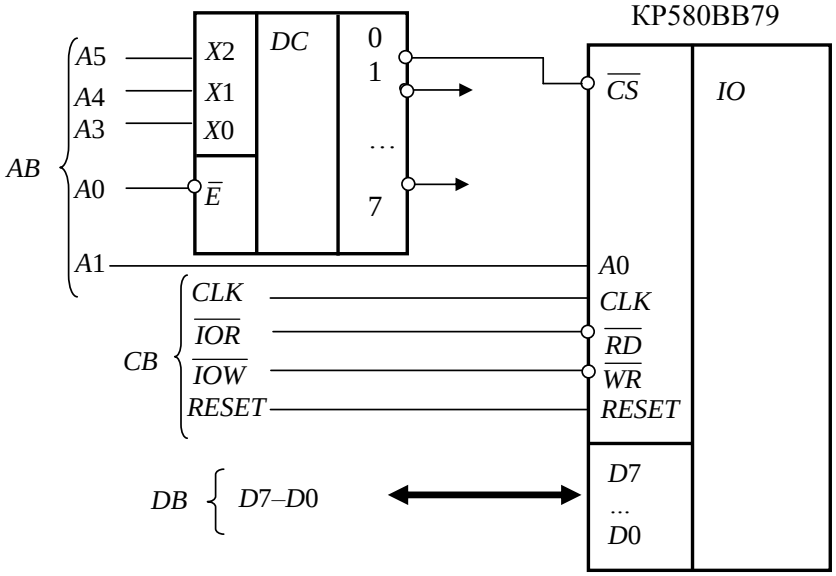


Рис. 6.24. З'єднання ВІС КР580ВВ79 із системною шиною МПС

Програмування мікросхеми КР580ВВ79 здійснюється завантаженням керувального слова ініціалізації клавіатури та дисплея у відповідний регістр керувальних слів, розташований у блоці керування. Під час записування керувальних слів на вхід *A*0 потрібно подавати сигнал логічної одиниці. Формат керувального слова ініціалізації клавіатури та дисплея показано на рис. 6.25.

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
0	0	0	*D*	*D*	*K*	*K*	*S*

Рис. 6.25. Формат керувального слова ініціалізації клавіатури та дисплея

Розряди *D*3 і *D*4 керувального слова (див. рис. 6.25) визначають режим роботи дисплея (табл. 6.6).

Таблиця 6.6. **Режими роботи дисплея**

*D*4	*D*3	Режим роботи
0	0	Дисплей на 8 символів із уведенням з лівого боку
0	1	Дисплей на 16 символів із уведенням з лівого боку
1	0	Дисплей на 8 символів із уведенням з правого боку
1	1	Дисплей на 16 символів із уведенням з правого боку

Розряди *D*1 та *D*2 визначають режим роботи клавіатури (табл. 6.7).

Таблиця 6.7. **Режими роботи клавіатури**

*D*2	*D*1	Режим роботи
0	0	Клавіатура в режимі одиночного натискання клавіш
0	1	Клавіатура в режимі *N*-клавішного натискання
1	0	Сканування матриці датчиків
1	1	Режим стробованого введення

Розряд *S* визначає режим сканування: при *S* = 0 – сканування в режимі 4-розрядного двійкового лічильника; при *S* = 1 – сканування в режимі інверсного дешифратора по лініях *S*3–*S*0. Якщо сканування здійснюється в режимі дешифратора, то дисплей містить не більше 4 символів, а клавіатура – не більше 8×4 клавіш.

У керувальному слові ініціалізації опорної частоти (рис. 6.26) розряди *D*4–*D*0 визначають коефіцієнт *PPPPP* ділення частоти зовнішнього синхросигналу *CLK* для одержання внутрішнього опорного сигналу із частотою не більше 100 кГц. Після скидання ВІС сигналом *RESET* установлюється максимальний коефіцієнт *PPPPP* = 11111.

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
0	0	1	*P*	*P*	*P*	*P*	*P*

Рис. 6.26. Формат керувального слова ініціалізації опорної частоти

Приклад 6.9. Запрограмувати ВІС контролера клавіатури та індикації для роботи з клавіатурою 8×8 клавіш у режимі *N*-клавішного натискання та з дисплеєм з 8 символів у режимі введення з лівого боку. Відповідно до схеми підключення (див. рис. 6.24) ВІС КР580ВВ79 має адреси 00*H* для даних і 02*H* для запису керувальних слів і читання статусної інформації. Частота імпульсів на вході *CLK* – 2 МГц.

Визначимо керувальні слова. За умовою прикладу та рис. 6.25 керувальне слово ініціалізації клавіатури та дисплея має такий вигляд (рис. 6.27):

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
0	0	0	0	0	0	1	0

Рис. 6.27. Керувальне слово ініціалізації клавіатури та дисплея

і дорівнює 02Н. Керувальне слово ініціалізації опорної частоти (див. рис. 6.25) зображено на рис. 6.28.

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
0	0	1	1	0	1	0	0

Рис. 6.28. Керувальне слово ініціалізації опорної частоти

Клерувальне слово дорівнює 34*H*. Значення *PPPPP* = 10100_2 визначає двійковий код коефіцієнта ділення імпульсів із частотою 2 МГц для забезпечення внутрішньої частоти 100 кГц (2000/100 = 20 = 10100_2).

Програма ініціалізації ВІС має такий вигляд:

```
MOV AL, 02    ; Формування першого керувального слова режиму в AL
OUT 02, AL    ; Виведення в інтерфейс
MOV AL, 34H   ; Формування другого керувального слова режиму в AL
OUT 02, AL    ; Виведення в інтерфейс
```

Після такої послідовності команд інтерфейс клавіатури та індикації готовий до роботи у запрограмованому режимі.

З’єднання ВІС інтерфейсу з клавіатурою та індикацією зображено на рис. 6.29.

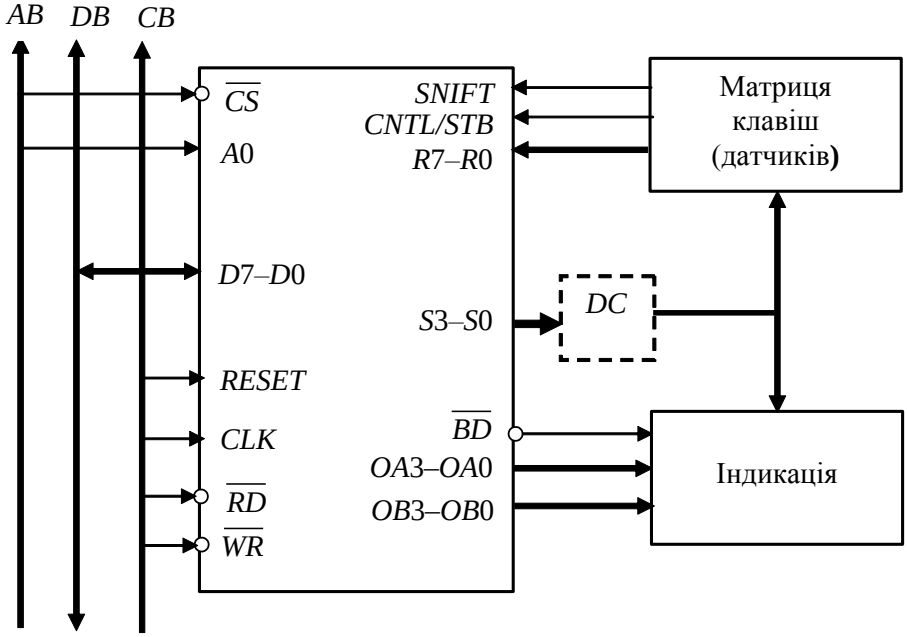


Рис. 6.29. Структурна схема з’єднання ВІС із клавіатурою та індикацією

Пунктиром показано приєднання зовнішнього дешифратора *DC* до ВІС. У випадку, якщо клавіатура містить менше ніж 4×8 клавіш, а кількість символів дисплея менше 4, то сигнали на виводах *S*3–*S*0 можна безпосередньо використовувати для керування клавіатурою та індикацією, оскільки на виводах *S*3–*S*0 формуються сигнали дешифратора із чотирма виходами. Приєднання зовнішнього дешифратора (до 16 виходів) дозволяє керувати клавіатурою 16×8 клавіш та 16 символами дисплея.

Функціонування блока інтерфейсу індикації. Після запису керувального слова ініціалізації клавіатури та дисплея (див. рис. 6.25) блок інтерфейсу індикації встановлюється в один із чотирьох режимів, які визначаються розрядами *D*3 та *D*4. В усіх режимах для висвітлення символу на індикації треба завантажити *керувальне слово запису в ОЗП індикації* (рис. 6.30) з адресою *A*0 = 1, а потім завантажити дані при адресі *A*0 = 1. При індикації дані з ОЗП передаються на 8 ліній *OA*3–*OA*0, *OB*3–*OB*0. Дані можуть бути подані семисегментним кодом при безпосередньому з'єднанні індикаторів з лініями *OA*3–*OA*0, *OB*3–*OB*0 або двома 4-розрядними кодами при підключенні зовнішніх шифраторів.

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
0	1	1	*I*	*A*	*A*	*A*	*A*

Рис. 6.30. Формат керувального слова запису в ОЗП індикації

Розряди *D*3–*D*0 керувального слова (див. рис. 6.30) містять адресу *AAAA* позиції дисплея, яка має бути прочитана. Розряд *D*4 містить ознаку автоінкрементної адресації *I*. Якщо *I* = 1, то адреса буде інкрементуватися після кожної операції читання.

Для зчитування даних із ОЗП індикації необхідно завантажити *керувальне слово зчитування з ОЗП індикації* (рис. 6.31) при *A*0 = 1, а потім зчитати інформацію з ОЗП при *A*0 = 0.

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
1	0	0	*I*	*A*	*A*	*A*	*A*

Рис. 6.31. Формат керувального слова зчитування з ОЗП індикації

Призначення розрядів керувального слова зчитування з ОЗП індикації аналогічне призначенню розрядів керувального слова запису в ОЗП індикації. Поле *AAAA* визначає адресу позиції в ОЗП індикації, яка має бути зчитана.

Якщо дисплей містить 8 символів, то блок керування ВІС КР580ВВ79 сканує дисплей за 5,1 мс при внутрішній частоті 100 кГц, якщо ж 16 символів, то за 10,23 мс. Процес сканування дисплея полягає у видачі у вихідний регістр індикації *OUT RG* (див. рис. 6.23) вмісту кожної комірки ОЗП індикації. Функціонування блока інтерфейсу індикації залежить від способу видачі кодів сканування *S*3–*S*0 (рис. 6.23). У процесі *сканування в режимі інверсного дешифратора* інформація з'являється тільки в перших чотирьох знакомісцях дисплея, у режимі 4-розрядного двійкового лічильника з використанням зовнішнього дешифратора – на 16. Одночасно зі зміною станів лічильника сканування і вмісту вихідного регістра індикації на виводі $\overline{BD}$ з'являється сигнал логічного нуля тривалістю 150 мкс, що використовується для гасіння індикації зі зміною символів.

У режимах виведення інформації на індикацію з уведенням нових символів з лівого боку кожному знакомісцю дисплея відповідає один байт у ОЗП індикації. Комірці ОЗП з нульовою адресою відповідає крайнє ліве знакомісце, а комірці з адресою 7 (або 15) – крайнє праве в індикації на 8 або 16 символів відповідно. У режимах виведення інформації з уведенням нових символів з правого боку код нового символу записується в комірку ОЗП з нульовою адресою, при цьому раніше записана інформація зсувається вліво. У цьому режимі немає прямої відповідності між адресою комірки ОЗП і знакомісцем індикації.

Програмним шляхом можна заборонити видачу однієї або обох тетрад умісту вихідних регістрів. Керувальне слово *заборони запису в ОЗП індикації/гасіння* показано на рис. 6.32.

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
1	0	1	*x*	*IWA*	*IWB*	*BLA*	*BLB*

Рис. 6.32. Формат керувального слова заборони запису в ОЗП індикації/гасіння

Байти ОЗП індикації поділяють на тетради: *A* – старша, *B* – молодша. Розряди *D*3 (*IWA*) та *D*2 (*IWB*) – біти заборони запису інформації у тетради *A* та *B* ОЗП індикації відповідно, розряди *D*1 (*BLA*) та *D*0 (*BLB*) – біти гасіння або бланкування (установлення спеціального коду, наприклад, коду пробілу). Керувальне слово (див. рис. 6.32) дозволяє маскувати одну з тетрад у випадку подвійного 4-позиційного дисплея. У разі заборони запису в одну з тетрад тривалість низького рівня сигналу гасіння не менше 150 мкс, а у разі заборони запису в обидві тетради визначається часом дії керувального слова.

Для *встановлення коду бланкування*, а також *сканування ОЗП індикації та скидання байта стану* використовується керувальне слово, формат якого показано на рис. 6.33.

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
1	1	0	*CD*	*BC*1	*BC*2	*CF*	*CA*

Рис. 6.33. Формат керувального слова встановлення коду бланкування, сканування ОЗП індикації та скидання байта стану

Розряди *D*3 (*BC*1) та *D*2 (*BC*2) дозволяють обрати один з кодів бланкування (табл. 6.8).

***Таблиця 6.8*. Коди бланкування**

*BC*1	*BC*2	Код бланкування
0	*x*	00
1	0	20*H* (код пробілу)
1	1	0*FFH*

З установленням розряду *D*4 (*CD*) здійснюється процедура скидання ОЗП індикації шляхом заповнення кодами бланкування. Установлення розряду *D*1(*CF*) скидає байт стану, сигнал переривання і встановлює покажчик пам'яті матриці клавіатури на нульовий рядок. Дія розряду *D*0 (*CA*) аналогічна одночасній дії розрядів *D*4 і *D*1.

Приклад 6.10. Інтерфейс клавіатури й індикації запрограмовано на режим сканування 8-символьного дисплея з уведенням з лівого боку. До ВІС інтерфейсу приєднано три семисегментні індикатори в позиціях 0, 1, 2. Навести функціональну схему приєднання дисплея та написати програму видачі на дисплей вмісту трьох 8-бітових комірок пам'яті з початковою адресою *DS*:*SI*, у яких записано семисегментні коди. Адреси ВІС контролера клавіатури й індикації такі, як і у прикл. 6.9.

Функціональну схему приєднання дисплея показано на рис 6.34.

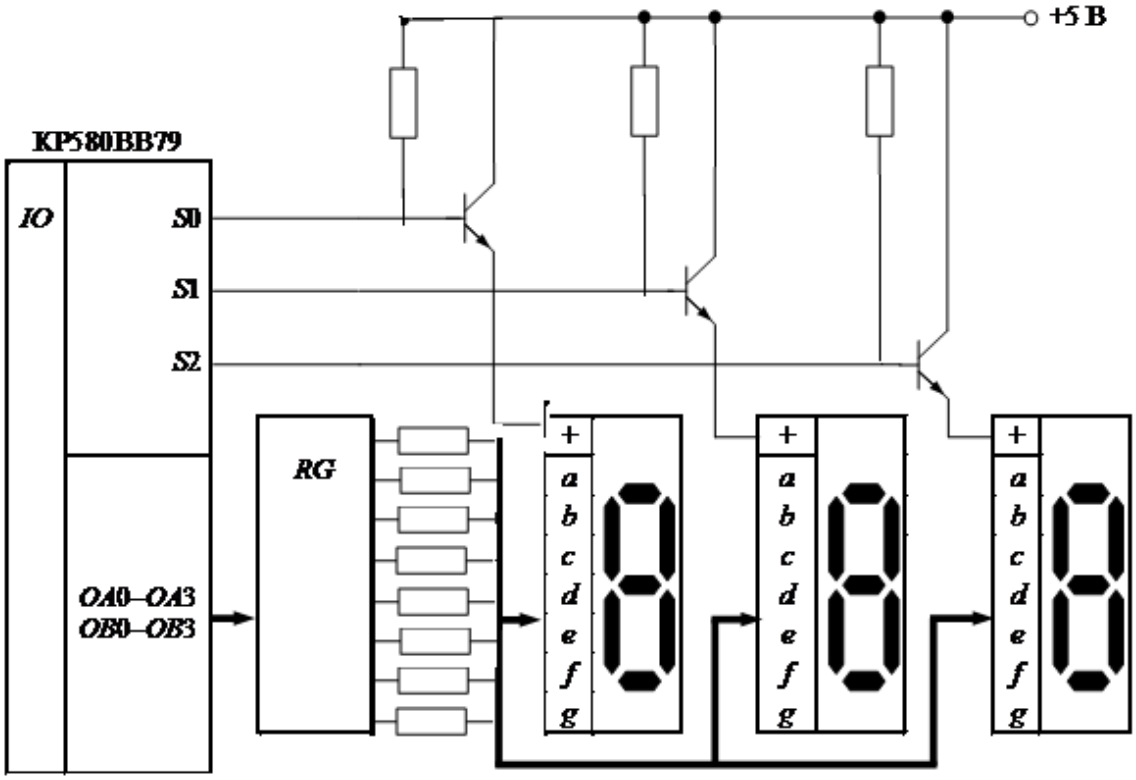


Рис. 6.34. Функціональна схема приєднання дисплея

Дисплей складається з трьох семисегментних індикаторів типу АЛС321Б, схема кожного з яких містить вісім світлодіодів, з'єднаних за схемою зі спільним анодом (рис. 6.35, *а*). Світлодіод висвітлюється при надходженні на входи *a*–*g* сигналу низького рівня (рис. 6.35, *б*), а на спільний анод – сигналу високого рівня.

Використання семисегментних індикаторів цього типу потребує подання інверсних семисегментних кодів. Індикатори гасяться з надходженням коду бланкування 0*FFH*.

Програма виведення символів на індикацію починається з гасіння дисплея записуванням коду 0*FFH* в ОЗП індикації. Керувальне слово встановлення коду бланкування, сканування ОЗП індикації і скидання байта стану (див. рис. 6.33) має містити значення *CD* = 1, *BC*1 = *BC*2 = 1. Тоді значення керувального слова дорівнює 0*DCH*.

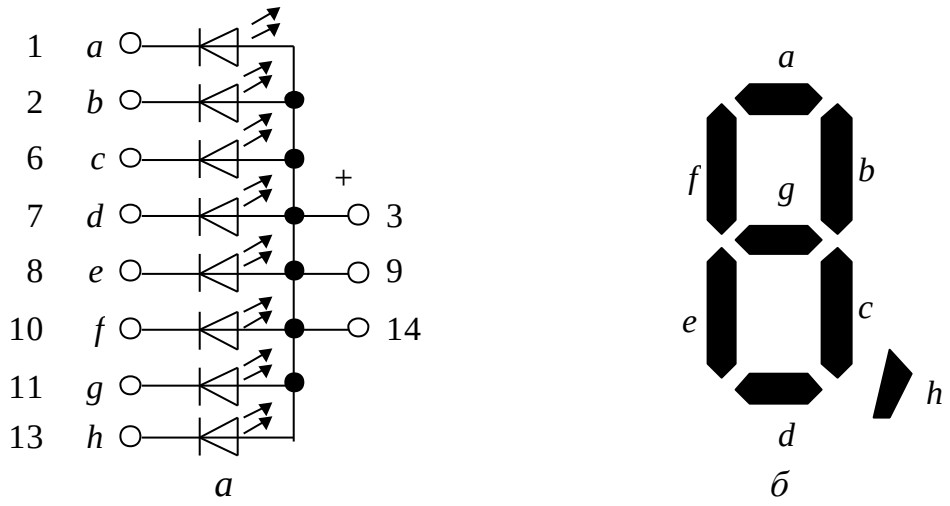


Рис. 6.35. Семисегментний індикатор АЛС321Б:
а – принципова схема; *б* – розміщення світлодіодів

Інформація на дисплей виводиться після завантаження керувального слова запису в ОЗП індикації (див. рис. 6.32), у якому *AAAA* = 0000, *I* = 1. Тоді значення керувального слова дорівнює 70*H*.

Програма має такий вигляд:

```
      MOV     AL, 0DCH   ; Формування керувального слова
                         ; встановлення коду бланкування,
                         ; сканування ОЗП індикації та скидання
                         ; байта стану
      OUT     02, AL     ; Виведення на інтерфейс
      CALL    DELAY1     ; Затримка на час бланкування
      MOV     AL, 70H    ; Формування керувального слова запису
                         ; в ОЗП індикації
      OUT     02, AL     ; Виведення на інтерфейс
      MOV     CX, 3      ; Завантаження лічильника байтів
M0:   MOV     AL, [SI]   ; Зчитування з ОЗП індикації
      OUT     00, AL     ; Виведення даних на інтерфейс
      INC     SI         ; Збільшення адреси на одиницю
      LOOP    M0
      CALL    DELAY2     ; Затримка на час сканування
```

Оскільки нові значення на індикацію можна передавати лише після затримки на час сканування даних в ОЗП дисплея (5,1 мс), то наведена програма містить підпрограму часової затримки *DELAY*2.

Функціонування блока інтерфейсу клавіатури (датчиків). Усі режими роботи блока інтерфейсу клавіатури (датчиків), що визначаються розрядами *D*1 та *D*2 керувального слова ініціалізації клавіатури і дисплея (див. рис. 6.25), можна поділити на три групи:

1) опитування матриці клавіатури;
2) опитування матриці датчиків;
3) уведення даних за стробом.

У режимі опитування матриці клавіатури (при $D1 = D2 = 0$ та при $D1 = 1$, $D2 = 0$) натискання будь-якої клавіші ініціює генерацію високого рівня сигналу переривання на виводі *IRQ*, а код натиснутої клавіші записується в ОЗП клавіатури (датчиків). Під час опитування матриці клавіатури функціонує схема усунення брязкоту контактів. Звернення до ОЗП відбувається за принципом черги: код, записаний в ОЗП першим, зчитується з нього першим. Щоб зчитати код клавіші з ОЗП, треба завантажити в інтерфейс клавіатури й індикації керувальне слово *читання з ОЗП клавіатури (датчиків)* (рис. 6.36) при $A0 = 1$, а потім зчитати дані з ОЗП при $A0 = 0$.

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
0	1	0	*I*	*x*	*A*	*A*	*A*

Рис. 6.36. Формат керувального слова читання з ОЗП клавіатури (датчиків)

Розряд *D*4 містить ознаку автоінкрементної адресації *I*, розряди *D*2–*D*0 – адресу *AAA* байта ОЗП клавіатури (датчиків), що має бути зчитаним. Якщо біт *I* встановлено, то наступні команди читання даних будуть викликати автоматичне інкрементування адреси. Для читання вмісту всього ОЗП необхідно завантажити керувальне слово читання ОЗП клавіатури (датчиків) при $I = 1$, а після цього 8 разів зчитати дані.

Формат даних при читанні з ОЗП клавіатури (датчиків) показано на рис. 6.37. У розрядах *D*5–*D*3 розміщується номер рядка матриці натиснутої клавіші – значення розрядів *S*2–*S*0 лічильника сканування; у розрядах *D*2–*D*0 розміщений номер стовпця матриці натиснутої клавіші – значення розрядів *R*2–*R*0. Розряди *D*7–*D*6 можуть використовуватися при введенні з розширеної клавіатури – в них записується стан додаткових клавіш, з'єднаних із виводами *SHIFT* і *CNTL*.

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
CNTL	*SHIFT*	*S*2	*S*1	*S*0	*R*2	*R*1	*R*0

Рис. 6.37. Формат даних при читанні з ОЗП клавіатури (датчиків)

При зчитуванні даних із ОЗП клавіатури (датчиків) сигнал переривання *IRQ* скидається, але якщо ОЗП клавіатури (датчиків) містить ще не зчитані дані, то на виводі *IRQ* знов генерується сигнал високого рівня.

Режим опитування матриці клавіатури містить:

а) режим одиничного натискання клавіш із забороною введення кодів при натисканні двох або більше клавіш ($D1 = D2 = 0$);

б) *N*-клавішне натискання з дозволом уведення кодів при натисканні *N* клавіш ($D1 = 1$, $D2 = 0$).

У режимі одиничного натискання, якщо натиснуто дві або більше клавіш, в ОЗП клавіатури (датчиків) записується код лише однієї з клавіш – першої натиснутої або тієї, що опитується першою.

У режимі *N*-клавішного натискання в ОЗП заносяться коди всіх натиснутих клавіш по черзі їх опитування при скануванні матриці клавіатури. У цьому режимі можна запрограмувати ВІС інтерфейсу на спеціальний режим помилки сканування клавіатури встановленням одиничного значення розряду *D*4 (*E*) у *керувальному слові скидання переривання/встановлення режиму помилки сканування* (рис. 6.38). Це керувальне слово також установлює *L*-рівень сигналу переривання на лінії *IRQ*.

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
1	1	1	*E*	*x*	*x*	*x*	*x*

Рис. 6.38. Формат керувального слова скидання переривання/встановлення режиму помилки сканування

При одночасному натисканні кількох клавіш установлюється прапорець помилки у байті стану і генерований високий рівень сигналу на виводі *IRQ*. *Формат байта стану* показано на рис. 6.39.

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
DU	*S/E*	*O*	*U*	*F*	*N*2	*N*1	*N*0

Рис. 6.39. Формат байта стану

Одиничне значення розряду *D*7 (*DU*) вказує на недоступний дисплей, тобто на те, що не закінчена операція очищення ОЗП індикації. Одиничне значення розряду *D*6 (*S/E*) – на те, що датчик замкнений (у режимі опитування матриці датчиків), або на помилку багатоклавішного натискання (у режимі опитування клавіатури). Одиничне значення розряду *D*5 (*O*) вказує на помилку переповнення. Цей розряд установлюється тоді, коли робиться спроба запису в заповнену пам’ять клавіатури. Одиничне значення розряду *D*4(*U*) вказує на помилку спустошення і установлюється тоді, коли робиться спроба зчитування даних із порожньої ОЗП клавіатури (датчиків). Одиничне значення розряду *D*3 (*F*) вказує на заповненість ОЗП клавіатури (датчиків). Розряди *D*2–*D*0 (*N*2–*N*0) визначають кількість символів у ОЗП клавіатури (датчиків).

У режимі опитування матриці датчиків зміна стану одного з датчиків ініціює генерацію високого рівня сигналу на виводі *IRQ*. При цьому значення розрядів *R*7–*R*0 безпосередньо записуються в ОЗП клавіатури (датчиків) без передачі керування схемі усунення брязкоту контактів.

У режимі введення за стробом значення розрядів *R*7–*R*0 записуються в ОЗП клавіатури (датчиків), але введення стробується сигналом на виводі *CNTL*/*STB*.

Приклад 6.11. Навести функціональну схему з'єднання ВІС з інтерфейсом клавіатури, яка містить клавіші 10 цифр та клавішу <*ENTER*>. Визначити коди клавіш. Інтерфейс клавіатури й індикації запрограмовано на режим опитування матриці клавіатури із забороною введення кодів при натисканні *N* клавіш.

Функціональну схему з'єднання ВІС із клавіатурою показано на рис. 6.40.

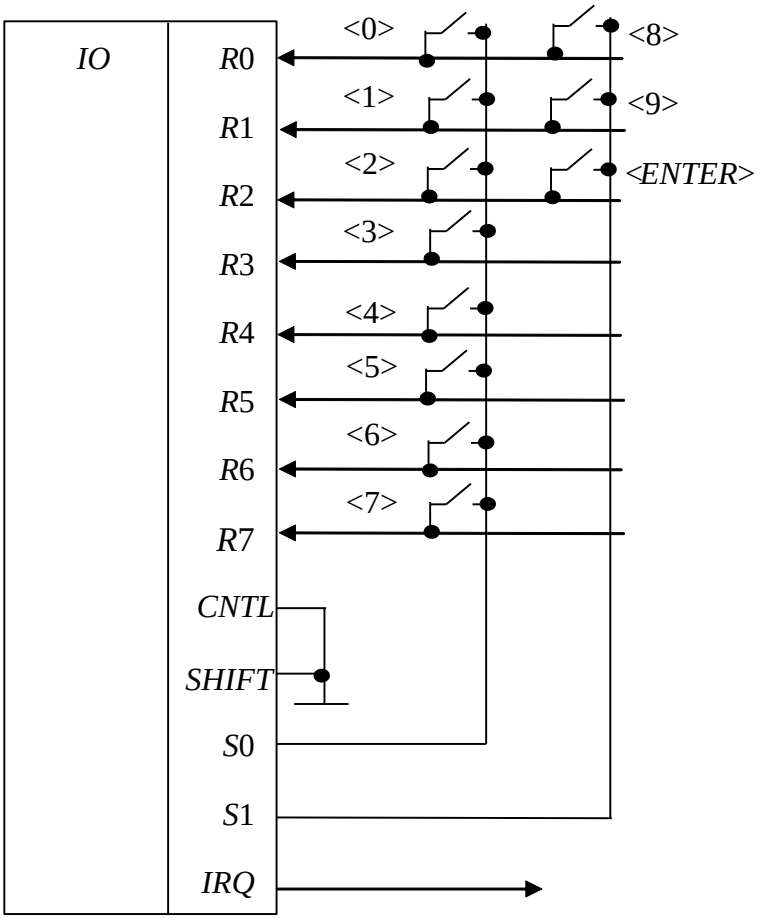


Рис. 6.40. Функціональна схема з'єднання ВІС із клавіатурою

При такому з'єднанні коди клавіш, які записуються в ОЗП клавіатури, визначаються за рис. 6.37. Для цифр від 0 до 9 коди збігаються із цифрами, позначеними в лапках (див. рис. 6.40). Код клавіші *ENTER* визначається як 00 001 010 і дорівнює 0*AH*.

Контрольні запитання

1. Укажіть призначення ВІС програмовного інтерфейсу клавіатури та індикації.
2. Яку максимальну кількість клавіш можна з'єднати з ВІС програмовного інтерфейсу клавіатури та індикації КР580ВВ79?
3. Поясніть особливості режиму опитування матриці клавіатури.
4. Поясніть особливості режиму опитування матриці датчиків.
5. Поясніть особливості режиму введення за стробом.

6.4. Програмовний таймер

Програмовний таймер (ПТ) КР1810ВИ54 призначений для організації роботи МП систем і формування сигналів з різними часовими та частотними характеристиками. Структурну схему ВІС показано на рис. 6.41, а умовне позначення – на рис. 6.42.

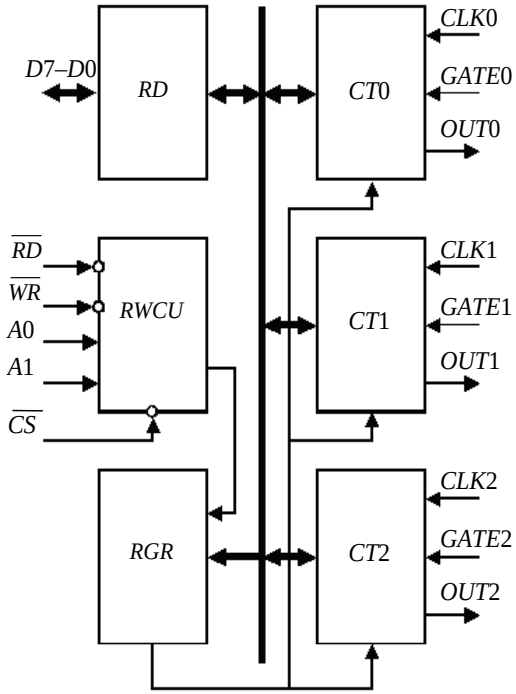


Рис. 6.41. Структурна схема ВІС КР580ВИ54: *CLK*0–*CLK*2 – входи тактових імпульсів; *GATE*0–*GATE*02 – входи дозволу лічення; *OUT*0–*OUT*2 – виходи лічильника

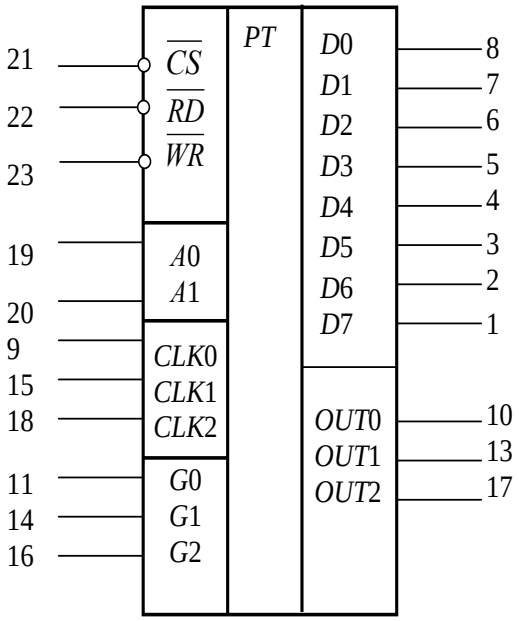


Рис. 6.42. Умовне позначення ВІС КР580ВИ54

Схема таймера містить: блок керування читанням/записом *RWCU* з регістром керувального слова *RCW*, тристабільний буфер даних *BD*, три канали на базі 16-розрядних від'ємних лічильників *CT*0–*CT*2. Кожний канал містить лічильник, вхідні та вихідні буферні регістри. Лічильники можуть працювати у двійковому або двійково-десятковому коді. Максимальна частота лічильника становить 2 МГц для КР580ВИ53 та 5МГц для КР1810ВИ54.

Призначення виводів ПТ КР580ВИ54 наведено в табл. 6.9.

Розряди *A*1, *A*0 (див. рис. 6.41) обирають звернення до лічильників *CT*0–*CT*2 або до регістра керувального слова *RCW* (табл. 6.10).

Таблиця 6.9. **Призначення виводів ВІС таймера КР580ВИ54**

Позначення	Номер виводу	Призначення
*D*7–*D*0	1, 2, 3, 4, 5, 6, 7, 8	Шина даних
$\overline{CS}$	21	Вибірка кристала; при $\overline{CS}$ = 0 дозволена робота ВІС
$\overline{RD}$	22	Читання: сигнал $\overline{RD}$ = 0 налагоджує вхідний буфер на виведення, при якому ПТ видає інформацію у МП
$\overline{WR}$	23	Запис: сигнал $\overline{WR}$ = 0 налагоджує вхідний буфер на введення, при якому ПТ приймає інформацію від МП
*A*0, *A*1	19, 20	Адресні входи, за якими відбувається адресація до одного з трьох каналів таймера
*CLK*2–*CLK*0	9, 15, 18	Вхід тактових сигналів для керування лічильником/таймером. Зріз сигналу на вході *CLK* призводить до зменшення вмісту лічильника таймера на одиницю
*GATE*2–*GATE*0	11, 14, 16	Входи дозволу лічби, дія яких залежить від режиму роботи канал
*OUT*2–*OUT*0	10, 13, 17	Виходи лічильника/таймера

Таблиця 6.10. **Адресація *CT*0–*CT*2, *RCW***

*A*1	*A*0	Звернення
0	0	*CT*0
0	1	*CT*1
1	0	*CT*2
1	1	*RCW*

Сигнали керування роботою ВІС $\overline{WR}$, $\overline{RD}$, $\overline{CS}$ подаються на блок *RWCU* і разом з адресними розрядами *A*0, *A*1 задають вид виконуваної операції згідно з табл. 6.11.

Таблица 6.11. **Вид операції ПТ залежно від сигналів керування та адресних розрядів**

Операція	Сигнали керування				
	$\overline{WR}$	$\overline{RD}$	$\overline{CS}$	*A*0	*A*1
Запис керувального слова в *RCW*	0	1	0	1	1
Завантаження *CT*0	0	1	0	0	0
Завантаження *CT*1	0	1	0	0	1
Завантаження *CT*2	0	1	0	1	0
Читання *CT*0	1	0	0	0	0
Читання *CT*1	1	0	0	0	1
Читання *CT*2	1	0	0	1	0
Від’єднання ПТ від шини	1	1	0	*x*	*x*

Примітка. x – будь-яке значення (0 або 1).

Узагальнену схему приєднання ПТ до шин МП показано на рис. 6.43. Як приклад, на адресній лінії *A*1, *A*0 можна приєднати до ліній *A*2, *A*1 шини адрес, на вхід $\overline{CS}$ подати сигнал з виходу дешифратора, як на рис. 6.9, 6.24.

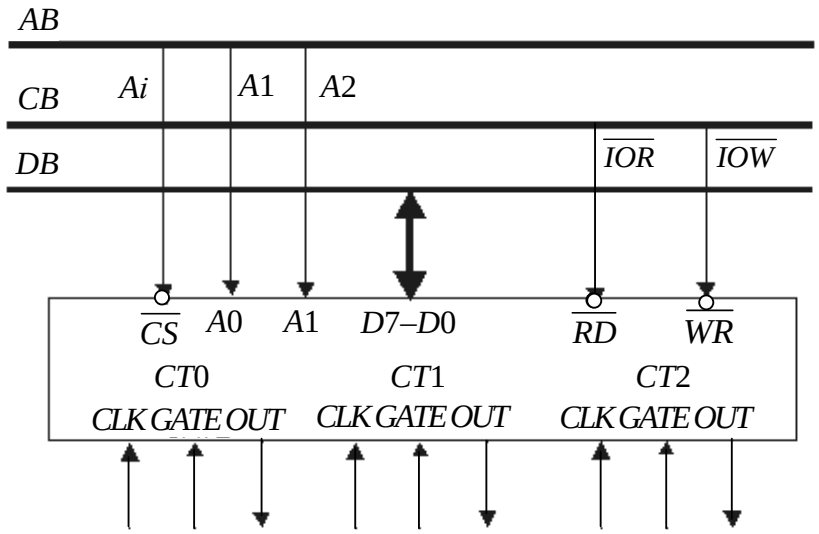


Рис. 6.43. Приєднання ПТ до шин МП

Установлення режиму роботи кожного каналу ПТ здійснюється програмно записуванням керувального слова (рис. 6.44) і початкового вмісту лічильника.

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
*CNT*1	*CNT*0	*RW*1	*RW*0	*M*2	*M*1	*M*0	*K*

Рис. 6.44. Формат керувального слова ПТ

За значеннями розрядів *D*7 (*CNT*1) та *D*6 (*CNT*0) обирають лічильник (табл. 6.12).

Таблиця 6.12. **Вибір лічильника**

*D*7	*D*6	Лічильник
0	0	Лічильник (*CT*0)
0	1	Лічильник (*CT*1)
1	0	Лічильник (*CT*2)
1	1	Заборонена комбінація для КР580ВИ53 та команда читання слова стану для КР1810ВИ54

За значеннями розрядів *D*5 (*RW*1) та *D*4 (*RW*0) обирають спосіб читання/запису (табл. 6.13).

Таблиця 6.13. **Спосіб читання/запису**

*D*5	*D*4	Спосіб читання/запису
0	0	Читання вмісту лічильника
0	1	Запис тільки молодшого байта
1	0	Запис тільки старшого байта
1	1	Запис молодшого, а потім старшого байтів

За значеннями розрядів *D*3–*D*1 (*M*2–*M*0) обирають один із 6 режимів роботи лічильника (табл. 6.14).

Таблиця 6.14. **Режими роботи ПТ**

*M*2	*M*1	*M*0	Режим
0	0	0	Режим 0
0	0	1	Режим 1
x	1	0	Режим 2
x	1	1	Режим 3
1	0	0	Режим 4
1	0	1	Режим 5

Розряд *D*0 (*K*) визначає спосіб кодування:

– *D*0 = 0 – двійковий лічильник;

– *D*0 = 1 – двійково-десятковий лічильник.

Приклад 6.12. Запрограмувати лічильник *CT*0 у режим 1. Адреса лічильника *CT*0– 10*H*, адреса регістра керувального слова – 16*H*.

Визначимо керувальне слово: 0011 0010 = 32*H*.

Програма буде мати такий вигляд:

```
MOV AL, 32H               ; Формування керувального слова
OUT 16H, AL               ; Виведення в RCW
MOV AL, "молодший байт"   ; Завантаження молодшого байта коду
OUT 10H, AL               ; попереднього встановлення
MOV AL, "старший байт"    ; Завантаження старшого байта коду
OUT 10H, AL               ; попереднього встановлення
```

Порядок програмування каналів таймера надзвичайно гнучкий. Можна записати керувальні слова режимів в усі канали, а потім у довільному порядку завантажувати коди попереднього встановлення, а можна запрограмувати окремо кожний канал (як у прикладі).

У процесі роботи ПТ уміст будь-якого з лічильників можна прочитати двома способами:

– призупинити роботу лічильника надходженням сигналу *GATE* = 0 або блокуванням тактових імпульсів, а потім прочитати вміст лічильника, починаючи з молодшого байта за допомогою двох команд уведення. Перша команда введення прочитає молодший байт, друга – старший;

– записати у ПТ керувальне слово, що містить нулі в розрядах *D*4, *D*5 (нулі в цих розрядах указують на виконання операції фіксації вмісту лічильника у вихідному регістрі каналу в момент записування керувального слова). Потім прочитати вміст лічильника за допомогою команд уведення.

Приклад 6.13. Запрограмувати лічильник 0 в режимі 1. Прочитати вміст лічильника *CT*0 і записати його у регістр *BX*.

Візьмемо адреси таймера такі, як у прикл. 6.12.

Визначимо керувальне слово для фіксації вмісту лічильника: $0000\ 0010_2 = 02H$.

Програма буде мати такий вигляд:

```
MOV AL, 02H         ; Формування керувального слова
OUT 16H, AL         ; Виведення в RCW
IN AL, 10H          ; Читання молодшого байта
MOV BL, AL          ; Пересилання молодшого байта у BL
IN AL, 10H          ; Читання старшого байта
MOV BH, AL          ; Пересилання старшого байта у BH
```

Отже, після виконання програми у *BX* буде вміст лічильника на момент його читання, а лічильник продовжуватиме лічбу.

Крім того, у ВІС К1810ВИ54 можна прочитати слова стану лічильника. Для цього необхідно записати керувальне слово (рис. 6.45).

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
1	1	*COUNT*	*STAT*	*CT*2	*CT*1	*CT*0	0

Рис. 6.45. Вигляд керувального слова ВІС К1810ВИ54:
*CT*0, *CT*1, *CT*2 – вибір лічильника

Лічильник вибирається при записі одиниці у відповідний двійковий розряд. Значення *STAT* = 0 вказує на те, що буде прочитано слово стану каналу, зазначеного в розряді *D*3–*D*1. Значення *COUNT* = 0 вказує на те, що буде запам’ятовано вміст лічильників, зазначених у *D*3–*D*1, у вихідних регістрах каналів.

Слово стану каналу, показано на рис. 6.46.

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
OUT	*FN*	*RW*1	*RW*0	*M*2	*M*1	*M*0	*K*

Рис. 6.46. Вигляд слова стану каналу: *OUT* – стан виходу *OUT* (0,1);
FN – прапорець перевантаження (*FN* = 1, якщо було перезавантаження коду попереднього встановлення); розряди *RW*1, *RW*0, *M*2, *M*1, *M*0 та *K* дублюють розряди керувального слова (див. рис. 6.38)

Під час запису керувального слова в лічильник завантажується спочатку молодший, а потім старший байт коду попереднього встановлення. Надалі робота таймера залежить від обраного режиму роботи.

Режими роботи таймера: 0 – програмовна затримка; 1 – програмовний мультивібратор; 2 – програмовний генератор тактових імпульсів; 3 – генератор прямокутних сигналів; 4 – програмнокерований строб; 5 – апаратнокерований строб. Вплив сигналу *GATE* на відповідний лічильник залежить від режиму роботи.

*Режим 0 – програмовна затримка***.** У цьому режимі (рис. 6.47) на виході вибраного каналу таймера формується сигнал *H*-рівня із програмнокерованою затримкою. Затримка відраховується від заднього фронту першого імпульсу *CLK* після запису молодшого байта коду перестановки константи. Після запису керувального слова на виході *OUT*

вибраного каналу таймера встановлюється сигнал *L*-рівня. Такий же стан зберігається при записі молодшого байта константи. Якщо під час лічби *GATE* = 0, лічення припиняється, а з появою *GATE* = 1 – відновлюється з перерваного значення. Після лічення на виході *OUT* встановлюється сигнал *H*-рівня. Завантаження у лічильник нового значення молодшого байта в процесі лічби зупиняє лічення, а завантаження старшого байта починає новий цикл лічби.

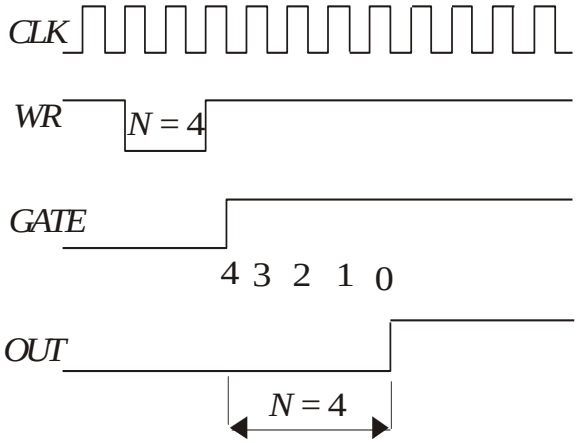


Рис. 6.47. Режим 0 – програмовна затримка

Режим 1 – програмовний мультивібратор. На виході лічильника формується імпульс *L*-рівня з програмнокерованою тривалістю, причому точкою початку відліку є задній фронт першого імпульсу *CLK* після появи сигналу *GATE* (рис. 6.48).

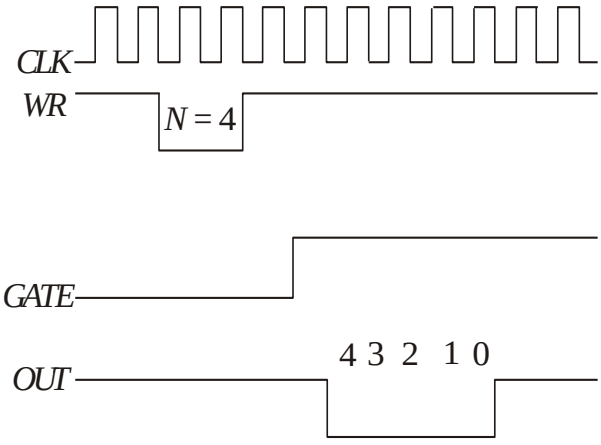


Рис. 6.48. Режим 1 – програмовний мультивібратор

При значенні сигналу *GATE* = 1 на виході *OUT* формується імпульс *L*-рівня тривалістю *N* періодів тактових імпульсів *CLK*. Завантаження у процесі лічби нового значення *N* не змінює поточного режиму лічби.

Мультивібратор автоматично перезапускається по кожному передньому фронту сигналу *GATE*.

Режим 2 – програмовний генератор тактових імпульсів. У цьому режимі (рис. 6.49) обраний канал здійснює розподіл частоти імпульсів *CLK* на програмнокерований коефіцієнт *N*, тобто ПТ генерує періодичний сигнал із частотою, у *N* разів меншою від частоти тактових імпульсів *CLK*.

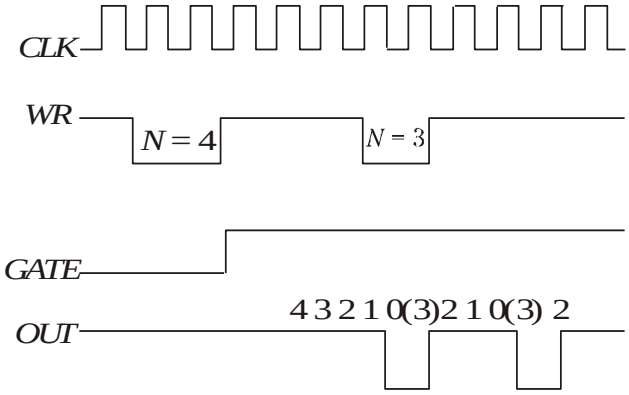


Рис. 6.49. Режим 2 – програмовний генератор тактових імпульсів

Вихідний сигнал *L*-рівня встановлюється на останньому такті періоду. Завантаження лічильника новим значенням *N* у процесі лічби призводить до зміни розміру періоду. Сигнал *GATE* можна використовувати для зовнішньої синхронізації ПТ, оскільки значення *GATE* = 0 забороняє лічбу, установлюючи значення сигналу *OUT* = 1, а значення *GATE* = 1 починає лічбу спочатку.

Режим 3 – генератор прямокутних імпульсів. Обраний канал формує прямокутні імпульси з програмнокерованим періодом. Дія сигналу *GATE* аналогічна режиму 0. При парному значенні *N* на виході лічильника генерується сигнал *H*-рівня протягом першої половини періоду і сигнал *L*-рівня протягом другої половини. При непарному *N* тривалість сигналу *H*-рівня на один такт більша, ніж для сигналу *L*-рівня. У режимі 3 число *N* = 3 не можна завантажувати у лічильник. Часові діаграми для цього режиму показано на рис. 6.50.

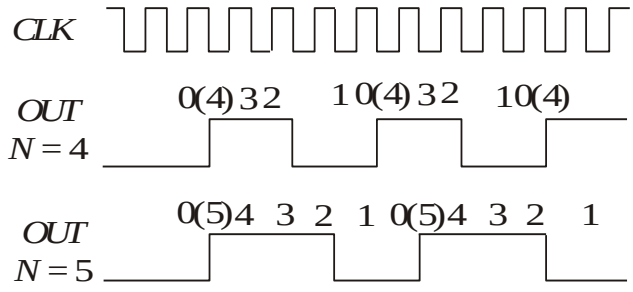


Рис. 6.50. Режим 3 – генератор прямокутних імпульсів

Режим 4 – програмнокерований строб. У цьому режимі на виході лічильника формується строб *L*-рівня тривалістю T_{CLK} з програмнокерованою затримкою щодо моменту запису молодшого байта команди. Переза-

вантаження молодшого байта у процесі лічби не впливає на поточне лічення, а завантаження старшого байта починає новий цикл лічби.

Режим 5 – апаратнокерований строб. Цей режим аналогічний режиму 4. Його відмінність полягає в тому, що початком відліку програмнокерованої затримки є передній фронт сигналу *GATE*. Запуск лічильника здійснюється переднім фронтом сигналу *GATE*. Завантаження у лічильник нового значення *N* у процесі лічби не впливає на тривалість поточного циклу, але такий цикл буде відповідати новому значенню *N*.

Діаграми роботи таймера, які ілюструють дію сигналу *GATE*, показано на рис. 6.51, *а* для таймера КР580ВИ53 і на рис. 6.51, *б* для таймера КР1810ВИ54. Для таймера КР580ВИ53 з появою *L*-рівня сигналу *GATE* лічення припиняється, а з появою *H*-рівня – відновлюється з перерваного значення. Для таймера КР1810ВИ54 з появою *L*-рівня сигналу *GATE* лічення також припиняється, а з появою *H*-рівня – починається із значення коду попереднього встановлення.

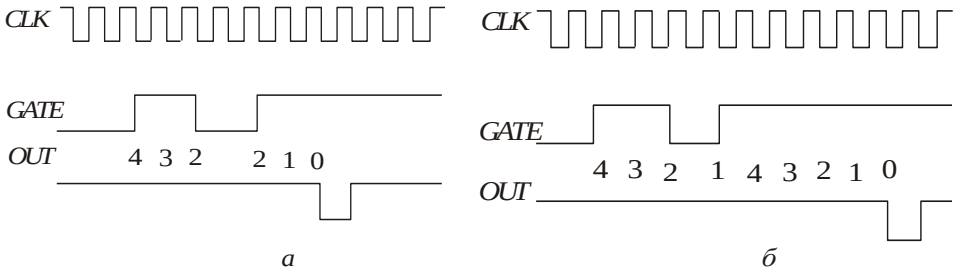


Рис. 6.51. Діаграми роботи таймера: *а* – для таймера КР580ВИ53; *б* – для таймера КР1810ВИ54

Приклад 6.14. Запрограмувати лічильник *CT*0 у режим генератора прямокутних імпульсів для отримання частоти $f_{вих}$ =1 кГц.

Візьмемо адреси таймера такі, як у прикл. 6.12. Для отримання послідовності імпульсів 1 кГц підключимо до виводу *G*0 сигнал *H*-рівня, на вивід *CLK*0 – тактові імпульси з частотою 5 МГц.

Знаходимо значення коефіцієнта ділення:

$$N = \frac{f_{CLK}}{f_{вих}} = \frac{5\,000}{1} = 5\,000.$$

Визначимо керувальне слово згідно з рис. 6.50 для програмування лічильника *CT*0 у режим 3 з двійково-десятковим засобом кодування:

$$00110111_2 = 37H.$$

Тоді програма матиме такий вигляд:

```
MOV     AL,37H      ; Програмування
OUT     16H,AL      ; таймера
MOV     AL,00       ; Запис молодшого
OUT     10H,AL      ; байта 00 попереднього встановлення
MOV     AL,50H      ; Запис старшого байта
OUT     10H,AL      ; попереднього встановлення
```

Після виконання програми на виводі *OUT*0 прямокутні імпульси із частотою 1 кГц будуть доти, доки не буде перепрограмовано таймер або вимкнено джерело живлення таймера.

Контрольні запитання

1. Які функції ПТ у МПС?
2. З яких блоків складається таймер? Що входить до складу одного лічильника?
3. Опишіть режими роботи таймера.
4. Чим відрізняється дія сигналу *GATE* в режимах 0 та 1?
5. Які нові функції має ВІС КР1810ВИ54 порівняно з ВІС КР580ВИ53?
6. Назвіть можливі комбінації роботи каналів таймера.
7. Назвіть приклади використання таймера МПС.
8. Запишіть керувальні слова для роботи таймерів у режимах 0, 2, 5.
9. Яким способом можна прочитати вміст внутрішніх регістрів таймера?
10. У чому полягає відмінність між режимами 4 і 5?
11. У якому режимі таймер працює як подільник частоти?
12. Яке максимальне число можна завантажити у внутрішні регістри таймера?
13. Для чого потрібні лінії *A*0, *A*1 таймера?

6.5. Архітектура і функціональні можливості контролера прямого доступу до пам’яті

Контролер прямого доступу до пам’яті КР580ВТ57 призначений для організації швидкісного обміну даними між пам’яттю і зовнішніми пристроями, який ініціюється зовнішнім пристроєм. (підрозд. 6.1, рис. 6.6).

Структурну схему контролера зображено на рис. 6.52. Вона містить:

– двонапрямлений двостабільний буфер даних (*BD),* призначений для обміну інформацією між МП і КПДП;
– схему керування читанням/записом (*RWCU*), що адресує внутрішні регістри КПДП і керує обміном по шині *D*7–*D*0;
– блок керування (*CU*), який містить регістри режиму і стану КПДП та задає режими роботи КПДП;
– блок керування пріоритетами *(PCU),* який забезпечує порядок обслуговування запитів зовнішніх пристроїв;
– чотири канали прямого доступу (*CH*0–*CH*3), кожний з яких містить регістр адреси комірки пам’яті, з якої починається обмін, лічильник циклів обміну, два старші розряди якого відведені для задання операцій обміну, та схему формування запитів/підтвердженнь.

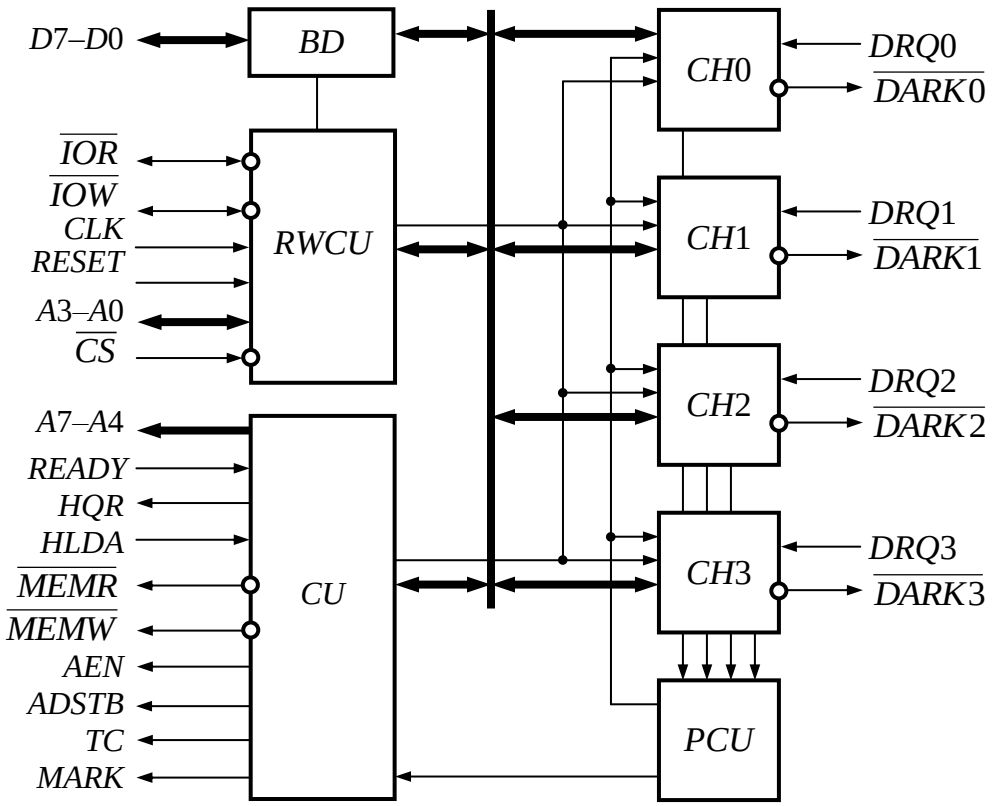


Рис. 6.52. Структурна схема КР580ВТ57

Призначення виводів ВІС наведено в табл. 6.15.

Таблиця 6.15. **Призначення виводів КПДП КР580ВТ57**

Позначення	Номер виводу	Призначення виводу
1	2	3
*D*7–*D*0	21, 22, 23, 36, 27, 28, 29, 30	Входи/виходи даних для обміну з МП
$\overline{IOR}$	1	Читання введення-виведення – двонапрямлений тристабільний вхід/вихід. Вхідний сигнал *L*-рівня дозволяє зчитування інформації з КПДП, вихідний сигнал низького рівня дозволяє зчитування з ПВВ
$\overline{IOW}$	2	Запис уведення-виведення – двонапрямлений тристабільний вхід/вихід. Вхідний сигнал низького рівня дозволяє програмування КПДП, вихідний сигнал низького рівня дозволяє запис у ПВВ
CLK	12	Вхід тактових імпульсів
RESET	13	Вхідний сигнал скидання
*A*3-*A*0	35,34,33,32	Двонапрямлені тристабільні адресні виводи
$\overline{CS}$	11	Вибірка ВІС
*A*7-*A*4	40,39,38,37	Тристабільні адресні виводи
READY	6	Вхідний сигнал готовності. *H*-рівень указує на готовність КПДП до обміну

Продовження табл. 6.15

1	2	3
HOLD	10	Вихідний сигнал запиту захоплення. *H*-рівень указує на запит захоплення КПДП системної шини
HLDA	7	Вхідний сигнал підтвердження захоплення. *H*-рівень указує на дозвіл доступу до системної шини
$\overline{MEMR}$	3	Вихідний сигнал читання з пам'яті. Тристабільний вихід. *L*-рівень дозволяє читання комірки пам'яті, що адресується КПДП
$\overline{MEMW}$	4	Вихідний сигнал запису у пам'ять. Тристабільний вихід. *L*-рівень дозволяє запис у комірку пам'яті, що адресується КПДП
AEN	9	Дозвіл адреси. *H*-рівень блокує шини адреси/даних
ASTB	8	Строб адреси. *H*-рівень указує на знаходження на шині *D*7–*D*0 старшого байта адреси пам'яті
TC	36	Кінець лічби. *H*-рівень указує на виконання останнього циклу передачі блоку даних
MARK	5	Маркер. *H*-рівень указує, що до кінця передачі блоку необхідно виконати кількість циклів обміну, яка кратна 128
*DRQ*3–*DRQ*0	16, 17, 18, 19	Запити прямого доступу до пам'яті каналів *CH*3–*CH*0. *H*-рівень вказує на запит від ПВВ
$\overline{DACK3}$ – $\overline{DACK0}$	15, 14, 24, 25	Підтвердження запитів прямого доступу до пам'яті каналів *CH*3–*CH*0. *L*-рівень вказує на дозвіл обміну
U_{CC}	31	Напруга живлення +5 В
GND	20	Спільний вивід

Кожний з чотирьох каналів ПДП забезпечує передачу блоку даних ємністю до 16 кбайт з довільною початковою адресою в діапазоні 0–64 кбайт. Пріоритети каналів можуть бути фіксованими (канал 0 має найвищий пріоритет, канал 3 – найнижчий) або змінюватися циклічно. В останньому випадку каналу, у якому відбулося обслуговування запиту, присвоюється нижчий пріоритет, а каналу з наступним номером – вищий.

Схему підключення КПДП до системної шини показано на рис. 6.53.

Молодший байт адреси пам'яті видається по лініях *A*3–*A*0 та *A*7–*A*4, які безпосередньо підключені до шини адреси *AB*. Старший байт адреси передається через шину *D*7–*D*0, тому в схему підключено буферний регістр К589ИР12, який фіксує значення старшого байта по сигналу *ADSTB* при *AEN* = 0. На вивід $\overline{CS}$ ВІС надходить сигнал з виходу дешифратора адрес уведення-виведення. Інші виводи КПДП приєднуються до однойменних ліній шин МП системи.

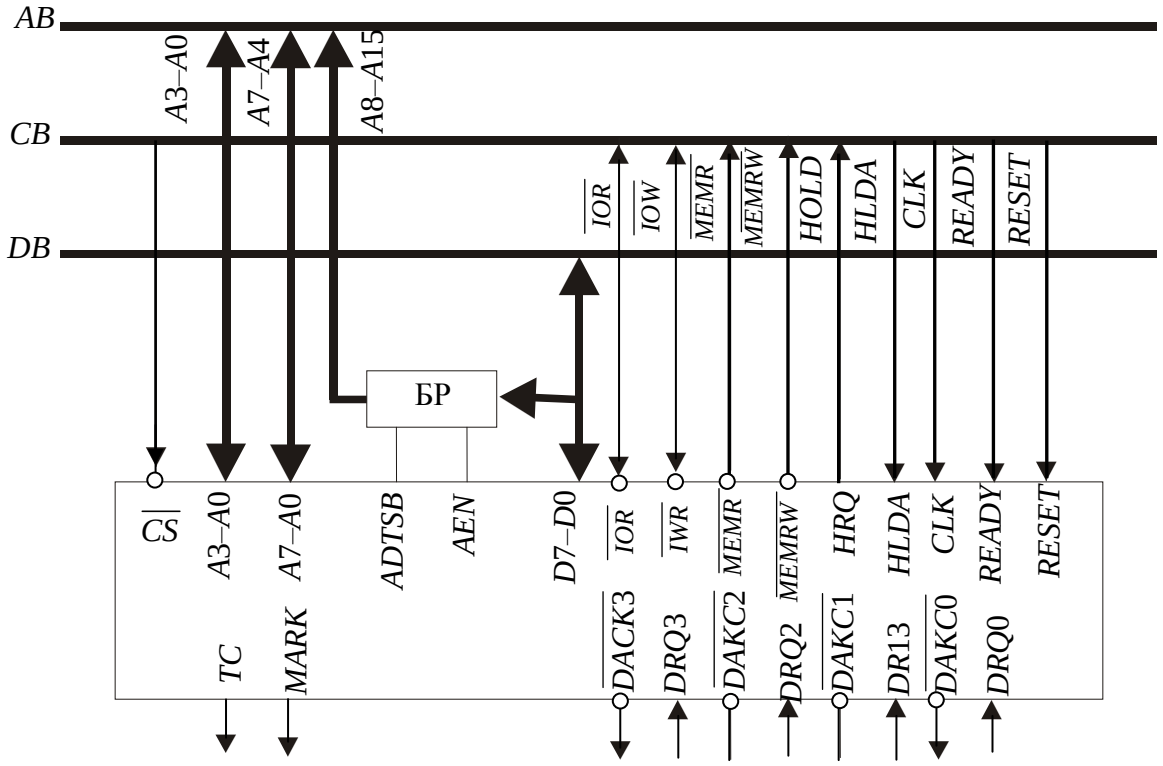


Рис. 6.53. Схема з'єднання КПДП із системною шиною з використанням буферного регістра (БР)

Програмуванням КПДП задають режим роботи каналів, напрям обміну інформацією між пам'яттю та ПВВ, початкову адресу та довжину масиву пам'яті.

Значення адресних розрядів *A*3–*A*0 і сигналу $\overline{CS}$ при записі та читанні регістрів ВІС наведено в табл. 6.16. Запис інформації в 16-розрядні регістри здійснюється двома командами, спочатку записується молодший байт, а потім – старший.

Таблиця 6.16. **Адресація внутрішніх регістрів КПДП**

Регістр	$\overline{CS}$	*A*3	*A*2	*A*1	*A*0
*RQ*0	0	0	0	0	0
*CT*0	0	0	0	0	1
*RQ*1	0	0	0	1	0
*CT*1	0	0	0	1	1
*RQ*2	0	0	1	0	0
*CT*2	0	0	1	0	1
*RQ*3	0	0	1	1	0
*CT*3	0	0	1	1	1
Регістр режиму (запис)	0	1	0	0	0
Регістр стану (читання)	0	1	0	0	1
Відключення КПДП від шини даних	1	*x*	*x*	*x*	*x*

Режим роботи каналів задається керувальним словом, формат якого показано на рис. 6.54. Розряди *D*3–*D*0 (*EN*3–*EN*0) дозволяють (при одиничному значенні) або забороняють (при нулі) обмін по відповідному каналу.

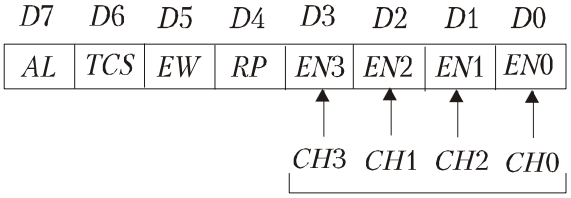


Рис. 6.54. Формат керувального слова: *AL* – автозавантаження; *TCS* – відключення каналу; *EW* – розширений запис; *RP* – циклічний пріоритет

Розряд *D*4 (*RP*) установлює порядок обслуговування запитів від каналів. При *D*4 = 0 задається фіксований пріоритет каналів, при *D*4 = 1 встановлюється режим циклічного пріоритету. Циклічний зсув пріоритетів відбувається після кожного циклу прямого доступу.

Установлення розряду *D*5 (*EW*) в одиницю задає режим розширеного запису, за якого тривалість сигналів $\overline{IOW}$ і $\overline{MEMW}$, що генеруються КПДП, збільшується за рахунок зсуву переднього фронту. Це дозволяє ПВВ, що формує сигнал *READY* по фронту сигналу запису, зменшити або збільшити швидкість обміну.

При *D*6 (*TCS*) = 1 поява сигналу *TC* в одному з каналів скидає відповідний розряд *D*3–*D*0, у результаті чого канал від'єднується, і його подальша робота можлива після перезавантаження регістра режиму. Якщо *D*6 (*TCS*) = 0, то поява сигналу *TC* не впливає на розряд дозволу роботи каналу і закінчувати передачу має ПВВ припиненням вироблення сигналу *DRQ*.

Одиничне значення розряду *D*7 (*AL*) задає режим автозавантаження, у якому працює тільки другий канал, використовуючи вміст своїх внутрішніх регістрів і внутрішніх регістрів третього каналу. Після передачі даних відповідно до параметрів регістрів другого каналу з появою сигналу *TC* уміст регістрів третього каналу автоматично завантажується в регістри другого каналу. При цьому у регістрі стану каналів (рис. 6.55) установлюється в логічну одиницю розряд *D*4 (*UF*) – прапорець відновлення. Потім передача даних продовжується відповідно до нових параметрів регістрів другого каналу, а в кінці першого циклу прямого доступу з новими параметрами прапорець відновлення скидається. Режим автозавантаження дозволяє організовувати повторні пересилання блоків даних із однаковими параметрами або з'єднувати декілька блоків з різними параметрами.

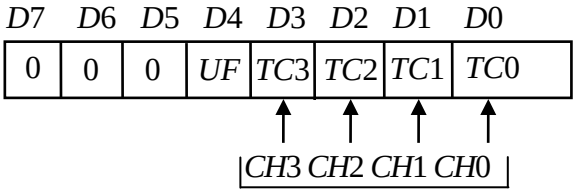


Рис. 6.55. Слово стану каналів: *UF* – прапорець відновлення

Розряди *D*3–*D*0 слова стану встановлюються одночасно з появою сигналу «кінець лічби» *TC* відповідного каналу і скидаються сигналом *RESET* при читанні вмісту регістра станів. Прапорець відновлення *UF* може бути скинутий, якщо записати логічний нуль у розряді *D*7 регістра режиму (див. рис. 6.54).

Початкова адреса ОЗП задається записом двох байтів у регістри каналів *RG*0–*RG*3.

Довжина масиву пам’яті та напрям обміну інформацією між пам’яттю і ПВВ задається записом двох байтів у лічильники *CT*0–*CT*3 циклів. Два старші розряди лічильника циклів визначають напрям обміну в такий спосіб: запис у пам’ять – 01, зчитування з пам’яті – 10, контроль – 00. Комбінація 11 заборонена. Інші розряди лічильника визначають кількість байтів, що будуть переслані.

Якщо два старші розряди лічильника циклів каналів установлюють режим контролю *VERIFY*, то передача даних не відбувається, оскільки не генеруються сигнали керування записом і читанням; усі інші функції прямого доступу зберігаються. Цей режим може використовуватися ПВВ для контролю прийнятих даних.

Роботу КПДП пояснює діаграма станів (рис. 6.56) і часові діаграми основних сигналів (рис. 6.57). Після запису слова режиму в регістр керувального слова КПДП переходить у холостий стан *SI*, який триває доти, доки на один із входів КПДП не надійде запит *DRQ* від зовнішнього пристрою на ПДП. Переходячи у стан *S*0, він виробляє сигнал *HRQ* і очікує надходження від МП сигналу *HLDA*. Після надходження сигналу підтвердження *HLDA* починається цикл обміну. У стані *S*1 формується сигнал *AEN* для блокування інших пристроїв системи від шин даних і керування, видається код молодших розрядів на виходи *A*7–*A*0, а код старших розрядів – на виходи *D*7–*D*0. Видача старших розрядів адреси супроводжується стробом *ADSTB* для запису їх у зовнішній буферний регістр. У стані *S*2 формуються сигнали $\overline{MEMR}$, $\overline{IOR}$ або $\overline{MEMW}$, $\overline{IOW}$, які визначають напрям обміну та сигнал $\overline{DACK}$, який вказує на початок обміну. У стані *S*3 здійснюється передача даних у ОЗП або ПВВ.

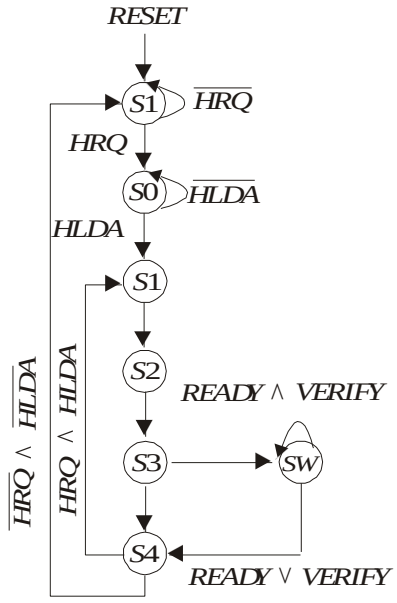


Рис. 6.56. Діаграма станів роботи КПДП

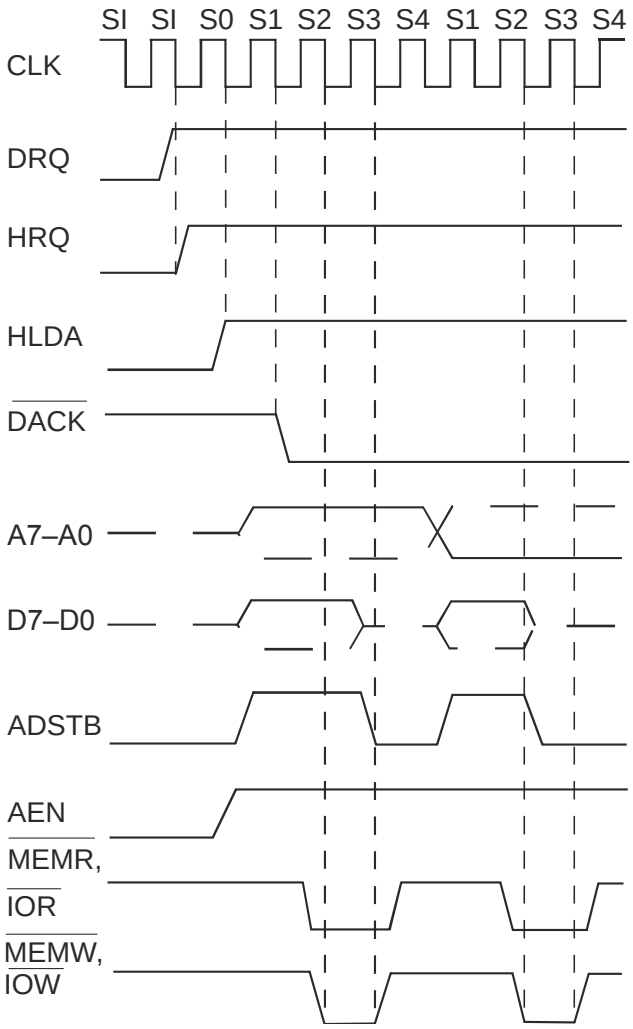


Рис. 6.57. Часові діаграми основних сигналів

Стан *S*4 завершує цикл прямого доступу. У цьому стані при передачі останнього байта блоку видається сигнал *TC*, а по закінченні – сигнал *MARK*. За необхідності узгодження швидкодії ОЗП і ПВВ за допомогою сигналу *READY* між станами *S*3 і *S*4 вводиться відповідна кількість станів очікування *SW*. У режимі контролю перехід у стан *SW* не дозволяється.

Приклад 6.15. Запрограмувати канал 1 КПДП на режим із фіксованими пріоритетами, з від'єднанням каналу після передачі, без автозавантаження і розширенного запису для пересилання по ньому контролера 10 байт із ОЗП з початковою адресою 0000:0700*H* у пристрій введення. Вхід $\overline{CS}$ КПДП з'єднати з адресною лінією *A*4.

Визначимо адреси КПДП. Згідно з табл. 6.15 адреса регістра каналу 1 буде 02*H*, лічильника – 03*H*, регістра режиму – 08*H*.

Керувальне слово режиму визначимо відповідно до рис. 6.54.

Програма матиме такий вигляд:

```
MOV    AL, 42H   ; Формування керувального слова режиму
OUT    08H, AL   ; запис його в регістр керувального слова
MOV    AL, 00    ; Молодший байт адреси ОЗП
OUT    02H, AL   ; переслати в регістр каналу 1
MOV    AL, 07H   ; Старший байт адреси ОЗП
OUT    02H, AL   ; переслати в регістр каналу 1
MOV    AL, 0AH   ; Молодший байт числа циклів
OUT    03H, AL   ; переслати в лічильник 1
MOV    AL, 80H   ; Задати напрям передачі-читання
OUT    03H, AL   ; пам'яті
```

Стан КПДП можна контролювати читанням умісту *RGA*, *CT* і 8-розрядного регістра стану, спільного для всіх каналів, за допомогою команди *IN*. Для читання вмісту 16-розрядного регістра використовуються дві команди *IN* з однією і тією самою адресною частиною, причому спочатку відбувається зчитування молодшого байта.

Контрольні запитання

1. Які функції виконує КПДП у МПС?
2. Опишіть режими роботи КПДП.
3. У яких випадках доцільне застосування ПДП?
4. Укажіть призначення регістра станів.
5. Опишіть принцип призначення пріоритетів каналів.

6.6. Програмовний послідовний інтерфейс

Програмовний послідовний інтерфейс КР580ВВ51 (*i*8251) являє собою універсальний синхронно-асинхронний приймач-передавач (УСАПП), призначений для організації обміну між МП і зовнішніми пристроями в послідовному форматі. Він приймає дані з 8-розрядної шини даних МП і передає їх у послідовному форматі периферійному пристрою або одержує послідовні дані від периферійного пристрою і перетворює їх у паралельну форму для передачі МП. Обмін може бути як *напівдуплексним* (однонапрямленим), так і

дуплексним (двонапрямленим). Послідовний інтерфейс може здійснювати обмін даними в асинхронному режимі зі швидкістю передачі до 9,6 кбіт/с або в синхронному – зі швидкістю до 56 кбіт/с залежно від запрограмованого режиму. Довжина переданих даних – від 5 до 8 біт. При передачі в МП символів завдовжки менше 8 біт невикористані біти заповнюються нулями. Формат символу містить також службові біти і необов'язковий біт контролю парності.

Структурна схема УСАПП (рис. 6.58) містить:

– буфер передавача *TBF* зі схемою керування передавачем *TCU*, який призначено для приймання даних від МП і видачі їх у послідовному форматі на вихід *TxD*;
– буфер приймача *RBF* зі схемою керування приймачем *RCU*, що виконує приймання послідовних даних із входу *RxD* і передачу їх у МП у паралельному форматі;
– буфер даних *BD*, який являє собою паралельний 8-розрядний двонапрямлений буфер шини даних із тристабільними каскадами, який використовується для обміну даними та керувальними словами між МП і УСАПП;
– блок керування читанням/записом *RWCU*, що приймає керувальні сигнали від МП і генерує внутрішні сигнали керування;
– блок керування модемом *MCU*, який обробляє керувальні сигнали, призначені для зовнішнього пристрою.

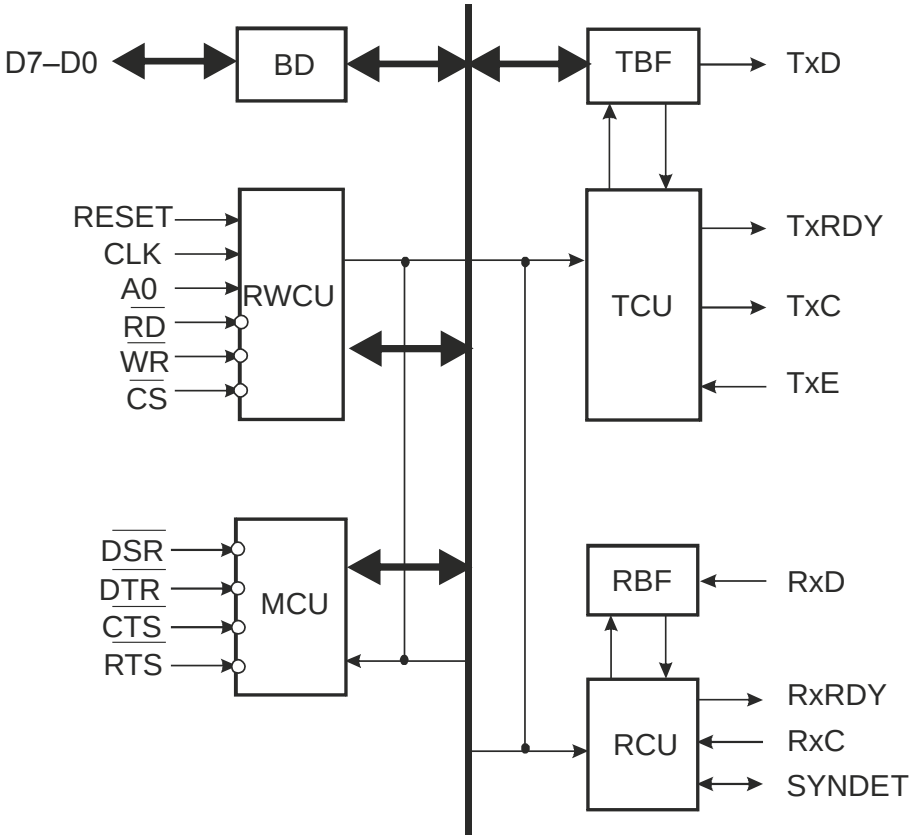


Рис. 6.58. Структурна схема УСАПП

Призначення виводів ВІС УСАПП наведено в табл. 6.17.

Таблиця 6.17. **Призначення виводів УСАПП**

Позначення виводу	Номер виводу	Призначення
*D*7–*D*0	8, 7, 6, 5, 2, 1, 28, 27	Канал даних
RESET	21	Установлення 0 (вихідний стан)
CLK	20	Синхронізація
*A*0	12	Адресація: *L*-рівень визначає можливість запису або читання даних в(із) УСАПП, *H*-рівень визначає можливість запису керувальних слів або читання слова стану в (із) УСАПП
$\overline{RD}$	13	Читання: дозвіл виведення даних або слова стану з УСАПП на шину даних
$\overline{WR}$	10	Запис: дозвіл уведення інформації із шини даних
$\overline{CS}$	11	Вибірка кристала: з'єднання УСАПП із шинами даних МП
$\overline{DSR}$	22	Готовність передавача термінала
$\overline{DTR}$	24	Запит передавача термінала
$\overline{CTS}$	17	Готовність приймача термінала
$\overline{RTS}$	23	Запит приймача термінала
SYNDET	16	Вид синхронізації: для синхронного режиму вихідний сигнал високого рівня – ознака внутрішньої синхронізації, для синхронного режиму із зовнішньою синхронізацією сигнал є вхідним, в асинхронному режимі сигнал є вихідним
RxC	25	Синхронізація приймача
RxRDY	14	Готовність приймача
RxD	3	Вхід приймача
TxC	9	Синхронізація передавача
TxE	18	Кінець передачі. Сигнал *H*-рівня є ознакою закінчення посилки даних
TxRDY	15	Готовність передавача
TxD	19	Вихід передавача
U_{CC}	26	Напруга живлення +5 В
GND	4	Спільний вивід 0 В

З'єднання УСАПП із шинами МП показано на рис. 6.59.

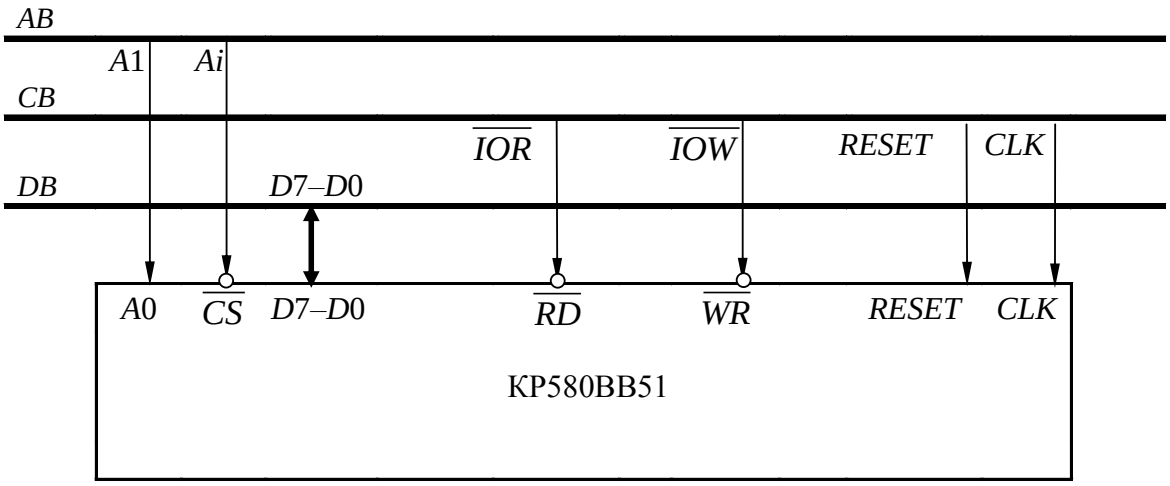


Рис. 6.59. З'єднання УСАПП із шинами МП

Сигнал *Ai,* поданий на вивід ВІС *A*0, визначає дві адреси УСАПП. При адресі *Ai* = 0 будуть передаватися дані, при *Ai* = 1 записуватися команди або читатися слова стану. Інші виводи приєднуються до однойменних ліній шин МП системи.

Програмування УСАПП відбувається завантаженням керувальних слів. Розрізняють керувальні слова двох типів: керувальне слово ініціалізації та операційне керувальне слово.

Значення сигналів адреси *A*0, керування читанням $\overline{RD}$, записом $\overline{WR}$ і вибіркою $\overline{CS}$ при записі та читанні регістрів ВІС наведено в табл. 6.18.

Таблиця 6.18. **Визначення операцій сигналами керування від МП**

Операції	Сигнали керування			
	*A*0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$
Читання даних з УСАПП на шину *D*7–*D*0	0	0	1	0
Запис даних із шини *D*7–*D*0 в УСАПП	0	1	0	0
Зчитування слова стану із УСАПП на шину *D*7–*D*0	1	0	1	0
Запис керувального слова із шини *D*7–*D*0 в УСАПП	1	1	0	0
Високоімпедансний стан виводів *D*7–*D*0	*x*	1	1	0
	x	*x*	*x*	1

Керувальне слово ініціалізації задає синхронний або асинхронний режим роботи, формат даних, швидкість приймання або передавання, контроль правильності даних. Це слово заноситься відразу після встановлення УСАПП у вихідний стан програмно або за сигналом *RESET*, а замінюється лише зі зміною режиму. Формат керувального слова різний в асинхронному чи синхронному режимах.

В асинхронному режимі роботи дані, що передаються, містять нульовий старт-біт, біти даних, біт контролю і стоп-біти. Кількість бітів даних і стоп-бітів, а також наявність або відсутність біта контролю задаються записом в УСАПП керувального слова режиму (рис. 6.60, *а).* Розряди *D*0

і *D*1 визначають коефіцієнт ділення сигналів синхронізації *CLK*. Розряди *D*3 і *D*2 визначають кількість бітів даних. Режим контролю задається розрядами *D*5 і *D*4. При *D*4 = 0 контроль парності заборонений; значення розряду *D*5 встановлює вид контролю – парності або непарності. Розряди *D*7 і *D*6 визначають кількість переданих стоп-бітів.

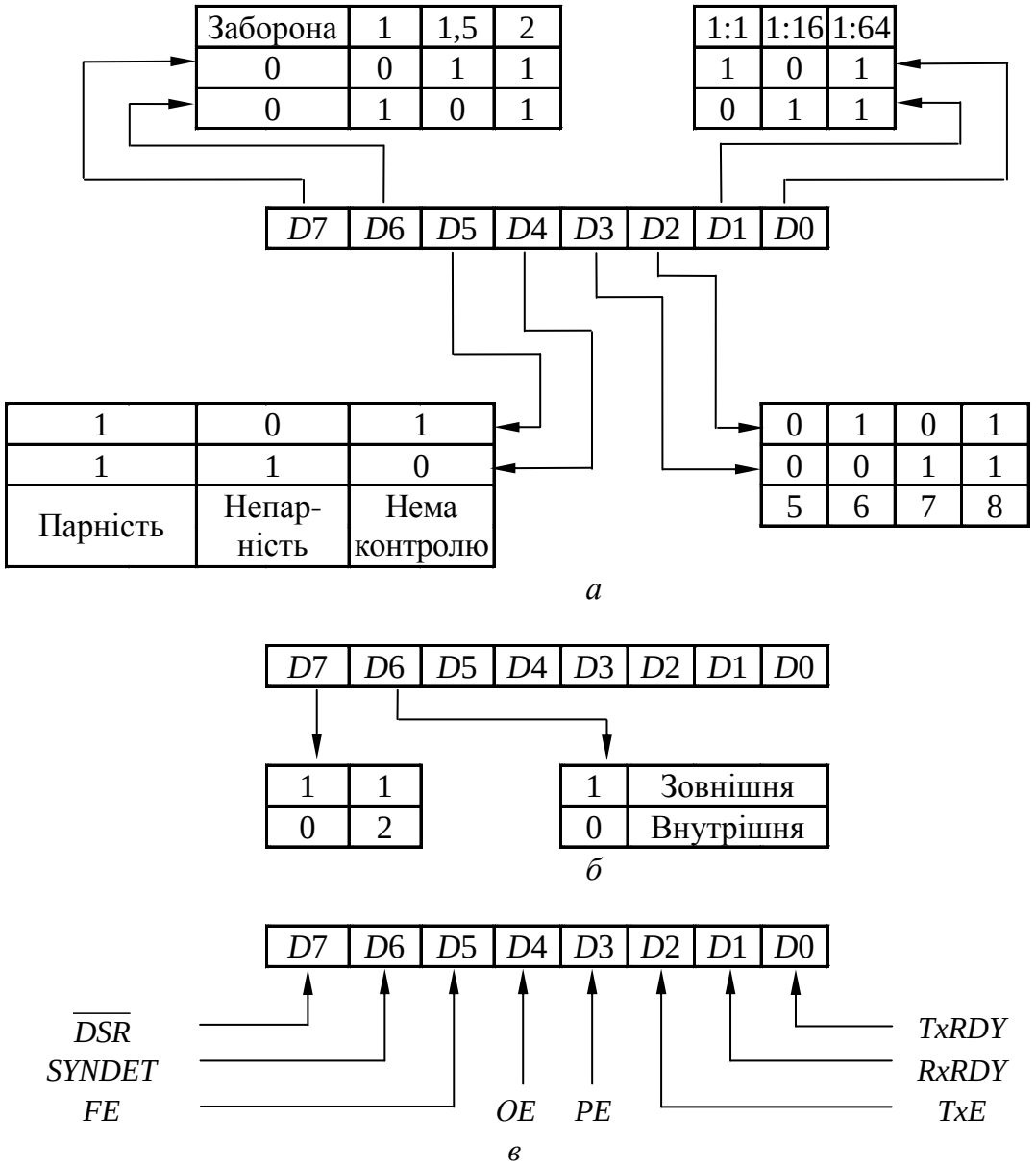


Рис. 6.60. Формат керувального слова ініціалізації: *а* – для асинхронного обміну; *б* – для синхронного обміну; *в* – формат слова стану

Синхронний обмін передбачає передачу даних у вигляді масивів слів. Для синхронізації запуску під час приймання даних використовуються один або два символи синхронізації (спеціальні кодові комбінації, наприклад, 10010100). Формат керувального слова ініціалізації режиму

для синхронного обміну показано на рис. 6.60, *б*. Розряди *D*1 і *D*0 мають нульове значення. Розряд *D*6 установлює тип синхронізації (зовнішню або внутрішню). Розряд *D*7 визначає використання одного (*D*7 = 1) або двох (*D*7 = 0) символів синхронізації. Призначення розрядів *D*3, *D*2 і *D*5, *D*4 аналогічне призначенню цих розрядів при асинхронному обміні.

Контроль стану УСАПП у процесі обміну даними МП здійснюється за допомогою команди читання слова стану. На рис. 6.60, *в* показано формат слова стану УСАПП. Розряд *D*3 (*PE*) установлюється тоді, коли виникають помилки парності; розряд *D*4 (*OE*) – коли виникають помилки переповнення, якщо МП не прочитав символ; розряд *D*5 (*FE*) – коли виникає помилка, яка полягає в тому, що для асинхронного режиму не виявлений стоп-біт. Інші розряди слова стану мають такий самий сенс, як і однойменні виводи МП, наведені в табл. 6.17.

Керування роботою УСАПП після ініціалізації здійснюється записуванням *операційних керувальних слів*, які можуть багаторазово задаватися у процесі обміну, керуючи різними його етапами. Призначення окремих розрядів операційного керувального слова УСАПП наведено в табл. 6.19.

Таблиця 6.19. **Призначення розрядів операційного керувального слова УСАПП**

Розряд	Позначення	Призначення
*D*0	*TxE*	Дозвіл передачі: при нульовому значенні передача інформації неможлива, при одиничному – можлива
*D*1	*DTR*	Запит про готовність передавача до передачі: при одиничному значенні – запис нуля на виводі $\overline{DTR}$
*D*2	*RxE*	Дозвіл на прийом: при нульовому значенні приймання інформації неможливе, при одиничному – можливе
*D*3	*SBRK*	Кінець передачі: при нульовому значенні – нормальна робота каналу передачі, при одиничному значенні – установлення високого рівня на виводі *TxD*
*D*4	*ER*	Виявлення помилок: при одиничному значенні – установлення розрядів помилок у вихідний стан
*D*5	*RTS*	Запит про готовність приймача термінала до прийому: при *D*5 = 1 – запис нуля на виводі $\overline{RTS}$
*D*6	*IR*	Програмне скидання схеми у вихідний стан: при одиничному значенні – установлення УСАПП у вихідний стан і готовність до приймання інструкції режиму
*D*7	*EH*	Режим пошуку імпульсів синхронізації: при одиничному значенні – установлення режиму пошуку символів синхронізації

Під час асинхронного обміну операційне керувальне слово завантажується відразу після керувального слова ініціалізації, а під час синхронного перед ним розташовуються один або два символи синхронізації.

Часові діаграми сигналів керування УСАПП під час запису керувального слова режиму, символів синхронізації команди показано на рис. 6.61, *а*, а часові діаграми сигналів при читанні слова стану – на рис. 6.61, *б*.

Читання слова стану (рис. 6.61, *а*) здійснюється при поданні на вхід *A*0 *H*-рівня, а на $\overline{RD}$ – *L*-рівня; читання даних здійснюється при поданні на вхід *A*0 *L*-рівня. Інформацію, яка міститься у слові стану (див. рис. 6.60, *в*), можна використовувати для організації обміну між МП і УСАПП, наприклад, визначати, чи не було помилки при передаванні інформації, який стан готовності приймача (передавача) до обміну

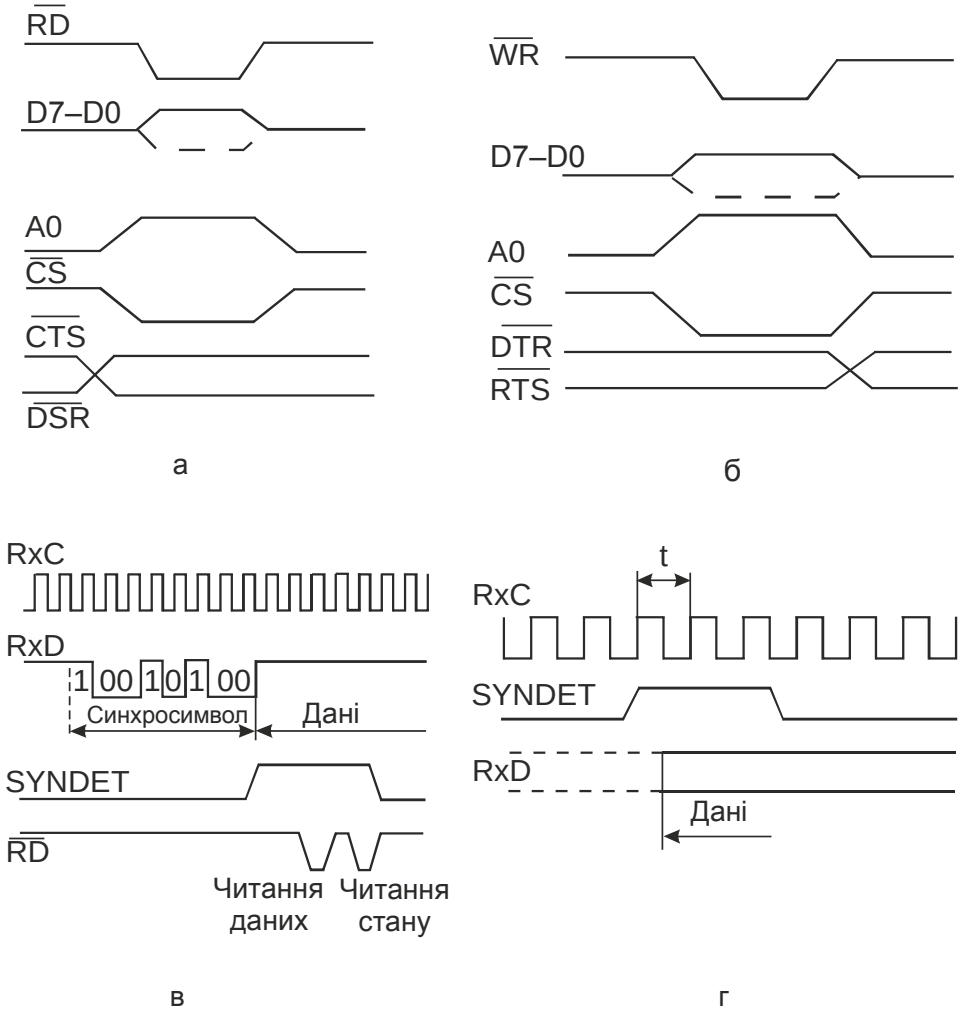


Рис. 6.61. Часові діаграми сигналів керування УСАПП: *а* – запис керувального слова режиму, символів синхронізації та операційного слова; *б* – читання слова стану; *в* – прийом вхідних даних із внутрішньою синхронізацією; *г* – синхронний прийом із зовнішньою синхронізацією

Запис керувальних слів та символів синхронізації здійснюється через шину даних *DB* при поданні на вхід *A*0 *H*-рівня, а на вхід $\overline{WR}$ – *L*-рівня (рис. 6.61, *б*). Після початкового встановлення УСАПП приймає інформацію на *DB* як керувальне слово ініціалізації і розміщує його у відповідному регістрі. Блок *RWCU* дешифрує це слово і, якщо запрограмований асинхронний режим, то наступне слово сприймається як операційне керувальне слово,

а якщо синхронний – інформація на *DB* сприймається як символ синхронізації.

Після запису керувального слова режиму і операційного слова УСАПП готовий до виконання обміну даними в одному з п’яти режимів:

1) синхронна передача;
2) синхронне прийомання із внутрішньою синхронізацією;
3) синхронне прийомання із зовнішньою синхронізацією;
4) асинхронна передача;
5) асинхронне прийомання.

При синхронній передачі даних на виході *TxD* із частотою сигналу синхронізації формується послідовність, що починається із символів синхронізації, запрограмованих керувальним словом режиму (див. рис. 6.60, *б*). Потім передаються коди символів, що надходять з МП, кожний з яких може закінчуватися бітом контролю. Якщо МП не завантажив черговий символ до моменту передачі, то УСАПП вставляє в передану послідовність символи синхронізації, а на виході *TxD* генерується сигнал високого рівня, який вказує на порожню передачу.

У режимі синхронного приймання із внутрішньою синхронізацією (рис. 6.61, *в*) УСАПП починає роботу з пошуку у вхідній послідовності символів синхронізації. УСАПП порівнює записані в нього при програмуванні символи синхронізації з прийнятими символами. Після виявлення символів синхронізації на виводі *SYNDET* установлюється сигнал високого рівня і починається прийом вхідних даних (рис. 6.61, *в*). Сигнал на виводі *SYNDET* автоматично скидається при читанні слова стану УСАПП.

Під час синхронного приймання із зовнішньою синхронізацією (рис. 6.61, *г*) на вивід *SYNDET* подається сигнал від зовнішнього пристрою, що дозволяє прийом даних на вході *RxD* зі швидкістю сигналів синхронізації, які надходять на вхід *RxD*. Можлива організація прийому даних у МП за перериванням, якщо сигнали на виводі *SYNDET* використовуються як запит переривання.

У режимі асинхронного передавання послідовні дані формуються на виході *TxD* за заднім фронтом сигналу синхронізації *TxC* з періодом, що задається керувальним словом режиму і дорівнює 1, 16 або 64 періодам сигналу синхронізації. Якщо після передачі символу наступного символу немає, то на виході *TxD* установлюється напруга високого рівня доти, доки нові дані не надійдуть від МП. У програмі, яка реалізує алгоритм асинхронної передачі, запис чергового байта в УСАПП здійснюється за командою виведення *OUT*, якщо у слові стану розряд *D*0 = 1. Сигнал на виході *TxDY* використовується як сигнал запиту переривання.

Асинхронне приймання даних починається з пошуку старт-біта, який встановлює на вході *RxD* напруга *L*-рівня. Наявність цього біта вдруге перевіряється внутрішнім строб-імпульсом. Якщо старт-біт підтвердже-

ний, то запускається внутрішній лічильник бітів, який визначає початок і кінець бітів даних, біт контролю і стоп-біт. Прийом стоп-біта вказує на закінчення приймання байта інформації та супроводжується встановленням сигналу високого рівня на виході *RxRDY*. У програмі асинхронного приймання-передавання чергового байта даних у МП може здійснюватися за командою введення *IN*, якщо у слові стану розряд $D1 = 1$, що відповідає *H*-рівню сигналу на виході *RxRDY*, або за перериванням, якщо сигнал на виході *RxRDY* використовується як сигнал запиту переривання.

Приклад 6.16. Запрограмувати УСАПП на асинхронний режим з таким форматом даних: старт-біт, 8 біт даних, біт контролю парності та 1,5 стоп-біта. Коефіцієнт ділення частоти синхросигналів 1/64. Вхід $\overline{CS}$ УСАПП з'єднати з адресною лінією $A4$, вхід $A0$ з лінією $A1$. Здійснити передавання даних із регістра *BL* по послідовному порту.

Визначимо адреси УСАПП. Запис керувального слова буде відбуватися при $A4 = 0$, $A1 = 1$ тобто при адресі 02, передавання даних – при $A1 = 0$, тобто при адресі 00.

Керувальне слово ініціалізації згідно з рис. 6.60, *а* дорівнює 10111111.

Операційне керувальне слово визначимо за табл. 6.18 як 00010001. Одиничне значення біта $D0$ дозволяє передачу, біт $D4$ скидає значення розрядів помилок у слові стану у вихідне положення. Нульове значення біта $D3$ визначає нормальну роботу каналу передачі. Інші біти не впливають на цей режим.

Програма матиме такий вигляд:

```
    MOV AL, 10111111B ; Формування в AL керувального слова
                      ; ініціалізації
    OUT 02, AL        ; і пересилання його в УСАПП
    MOV AL, 00010001B ; Формування в AL операційного
                      ; керувального слова
    OUT O2, AL        ; і пересилання його в УСАПП
M1: IN AL, 02         ; Читання слова стану
    AND AL, 01        ; Виділення розряду D0
    JZ M1             ; Якщо D0 = 0 (УСАПП не готовий до передачі),
                      ; то перехід на M1
    MOV AL, BL
    OUT 00, AL        ; Пересилання даних
```

Ознакою того, що дані можна передавати по послідовному порту, тобто УСАПП готовий до обміну, у цій програмі прийнята умова $D0 = 1$ у слові стану. Іншим шляхом є використання сигналу на виході *TxDY* як запиту переривання.

Контрольні запитання

1. Які функції виконує порт послідовної передачі даних у *VGC*?
2. Опишіть режими роботи ВІС.
3. Поясніть роботу ВІС послідовного інтерфейсу в асинхронному режимі.
4. Поясніть роботу ВІС послідовного інтерфейсу у синхронному режимі.
5. Як здійснюється обмін даними при синхронному режимі з внутрішньою і зовнішньою синхронізацією?

6.7. Програмовний контролер переривань

Програмовний контролер переривань КР580ВН59А являє собою пристрій, що реалізує в МПС обробку запитів переривань від зовнішніх пристроїв, як-то датчиків аварійних ситуацій або ПВВ, що реалізують протокол обміну за перериванням (див. підрозд. 6.1, рис. 6.5). ВІС ПКП виконує такі функції:

- запам'ятовує запити переривання, які задаються переднім фронтом або потенціалом;
- маскує, тобто забороняє виконання обраних запитів;
- формує вектор переривання та виконує дії по переходу на підпрограму обробки запиту;
- формує сигнал переривання для МП;
- виконує пріоритетну обробку запитів переривання.

ВІС КР580ВН59А залежно від того, як її запрограмували, може виробляти або код команди 8-розрядного МП *i*8080 *CALL ADRV*, де *ADRV* – адреса підпрограми обробки, або видавати на шину даних номер переривання *n* для команди *INT n* 16-розрядного МП *i*8086.

Одна ВІС ПКП обробляє вісім запитів на переривання, але за каскадного вмикання кількох ВІС кількість запитів переривання може бути збільшена до 64.

Спрощену структурну схему ПКП показано на рис. 6.62. До складу ПКП входять:

- двонапрямлений 8-розрядний буфер даних (*BD*), призначений для з'єднання ПКП із системною інформаційною шиною;
- блок керування читанням/записом (*RWCU*), що приймає керувальні сигнали від МП і задає режим функціонування ПКП;
- схема каскадного буфера-компаратора (*CMP*), яка використовується при вмиканні в систему декількох ПКП;
- схема керування (*CU*), що формує сигнали переривання і(або) трибайтову команду *CALL* або вектор переривання *n*;
- регістр запитів переривань (*RGI*), що використовується для зберігання всіх запитів переривань;
- схема обробки за пріоритетами (*PRB*), яка ідентифікує пріоритети запитів і вибирає запит із найвищим пріоритетом;
- регістр обслуговуваних переривань (*ISR*), що зберігає рівні запитів переривань, які знаходяться на обслуговуванні ПКП;
- регістр маскування переривань (*RGM*), що забезпечує заборону однієї або декількох ліній запитів переривання.

Призначення вхідних, вихідних і керувальних сигналів ПКП наведено при описі виводів мікросхеми в табл. 6.20

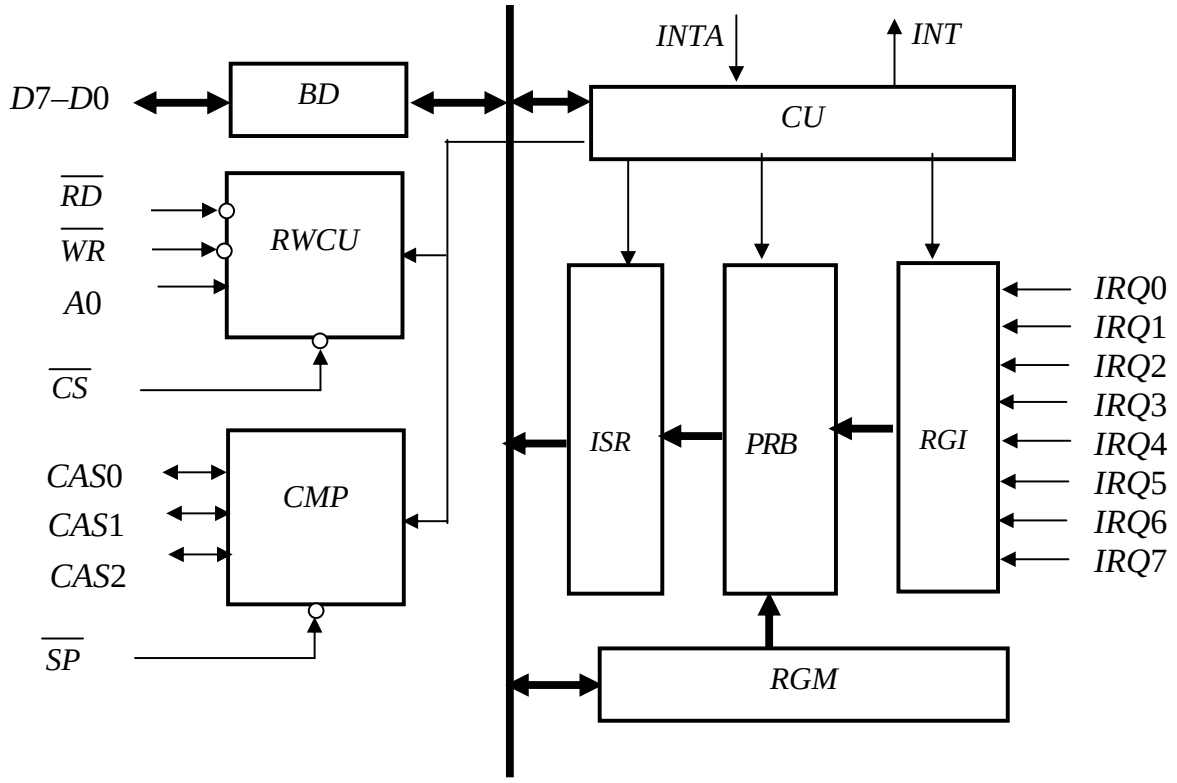


Рис. 6.62. Структурна схема ПКП

Таблиця 6.20. **Опис виводів ПКП**

Позначення виводу	Номер контакту	Призначення виводу
*D*7–*D*0	4, 5, 6, 7, 8, 9, 10, 11	Вхід/вихід даних
$\overline{RD}$	3	Вхід стробу читання
$\overline{WR}$	2	Вхід стробу запису
*A*0	27	Вхід нульового розряду адреси, використовуваний при завантаженні команд і зчитуванні стану ПКП
$\overline{CS}$	1	Вхід вибору мікросхеми
*CAS*0–*CAS*2	12, 13, 15	Входи/виходи каскадування
$\overline{SP}$	16	Ознака підпорядкування: напруга *H*-рівня вказує, що ПКП є старшим (головним) контролером; напруга *L*-рівня визначає ПКП підпорядкованим (веденим) контролером
INTA	26	Підтвердження переривання – вхідна напруга *H*-рівня вказує про видачу ПКП команди *CALL* на шину
INT	17	Переривання: напруга *H*-рівня вказує про запит на обслуговування
*IRQ*0–*IRQ*7	18, 19, 20, 21, 22, 23, 24, 25	Входи запитів переривань (передній фронт)
U_{CC}	28	Напруга живлення +5 В
GND	14	Спільний вивід 0 В

З’єднання ВІС КР580ВН59 зі стандартною системною шиною показано на рис. 6.63.

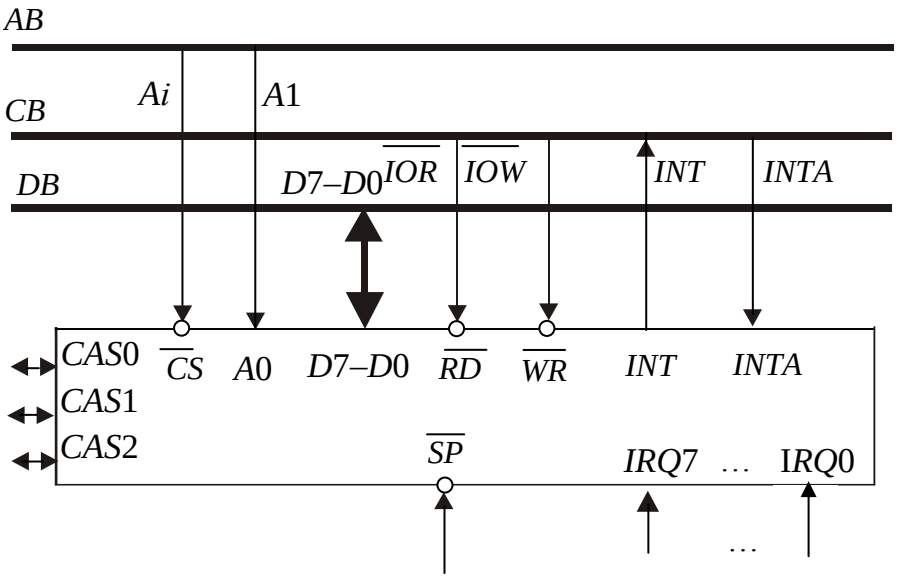


Рис. 6.63. З’єднання ВІС КР580ВН59 зі стандартною системною шиною

Ця схема працює так. Запити переривань від зовнішніх пристроїв надходять на входи *IRQ*0–*IRQ*7 і запам’ятовуються в регістрі *RGI*. Далі сигнали надходять на схему обробки за пріоритетами *PRB*, що дозволяє або не дозволяє подальшому проходженню запиту переривання залежно від його пріоритету та заборони маскуванням. Будь-який запит переривання можна заборонити, записуючи маску в регістр *RGM*. Далі дозволені запити надходять у регістр *ISR* і встановлюють відповідні розряди. Схема керування (*CU*) на основі стану регістру *ISR* формує сигнал переривання *INT* для МП, який приймає сигнал *INT* і, якщо переривання дозволені, підтверджує прийом видачею сигналу *INTA*. Після одержання сигналу *INTA* ПКП видає на шину *D*7–*D*0 код команди *CALL* або вектор переривання *n*. У першому випадку МП видає ще два сигнали *INTA*, що дозволяють ПКП передати на шину даних 16-розрядну адресу підпрограми обслуговування переривання, причому молодший байт адреси передається за першим сигналом *INTA*, а старший – за другим. У другому випадку МП видає ще один сигнал *INTA*, по якому МП зчитує значення вектора переривання *n*.

Програмовний контролер може працювати і в режимі опитування запитів переривання. У цьому разі МП зчитує код запиту з найвищим рівнем пріоритету за сигналом $\overline{RD}$. Прийом запитів, маскування й аналіз пріоритету виконуються так само, як і при обслуговуванні переривань за запитом.

Для збільшення кількості рівнів переривання ПКП можуть бути об’єднані в систему, що складається з одного ведучого і декількох ведених ПКП (рис. 6.64).

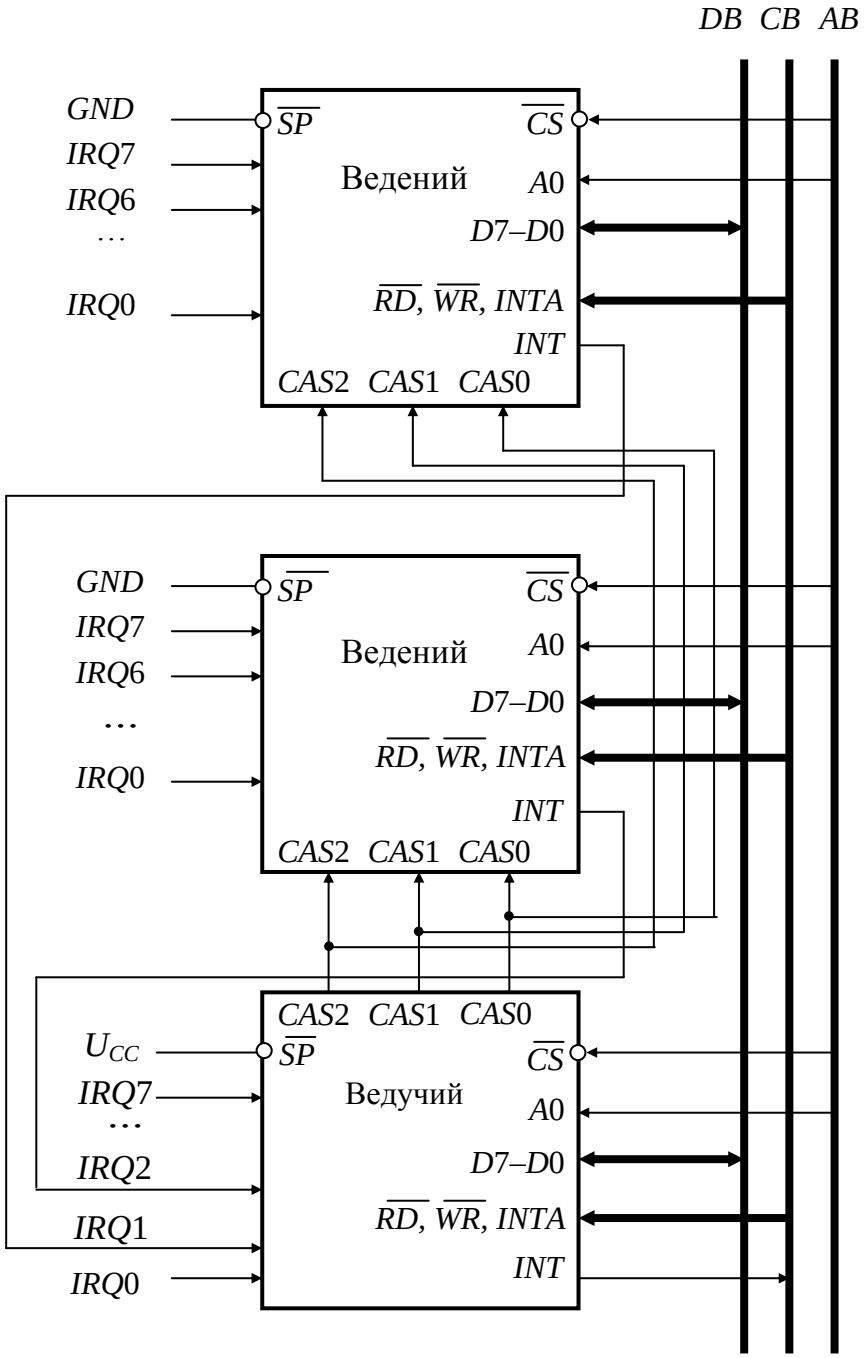


Рис. 6.64. Каскадне з’єднання ПКП

Під час обслуговування запиту, що надійшов на вхід веденого ПКП, ведучий ПКП за сигналом *INTA* видає на шину даних код команди *CALL*, а на шини *CAS*0–*CAS*2 – код номера веденого ПКП. З надходженням другого і третього сигналів *INTA* адреса підпрограми обслуговування даних видає обраний за кодом на шинах *CAS*0–*CAS*2 ведений ПКП. У випадку використання ПКП у МПС з *i*8086 ведений ПКП видає за другим сигналом *INTA* значення номера переривання *n*.

Програмування ПКП полягає в записі в нього у визначеному порядку від 2 до 4 керувальних слів ініціалізації (*ICW*). Далі в будь-якому порядку можна записувати керувальні операційні слова (*OCW*) залежно від необхідних функцій ПКП.

Керувальне слово *ICW*1 (рис. 6.65) скидає регістри *RGI, RGM* і присвоює нижчий пріоритет входу *IRQ*7. Одиничне значення розряду *AD*(*D*0) вказує на те, що треба використати додатне керувальне слово ініціалізації *ICW*4.

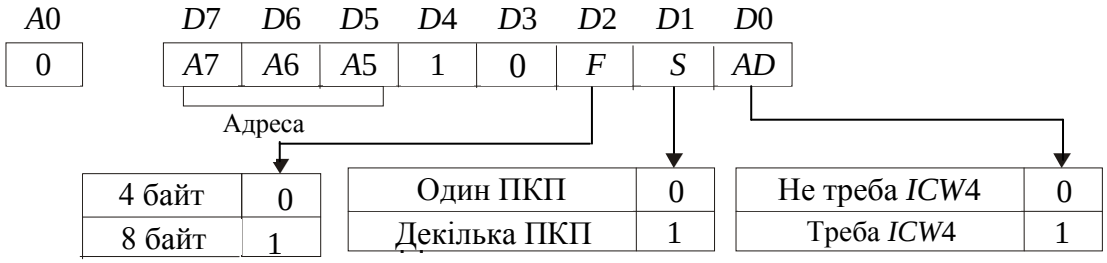


Рис. 6.65. Формат керувального слова ініціалізації *ICW*1

Розряд *S* цього слова визначає наявність одного або декількох ПКП у системі. Інші розряди можуть приймати будь-яке значення в МПС з *i*8086. У МПС із *i*8080 розряд *F* (формат) визначає адресний інтервал 4 або 8 байт між початковими адресами підпрограм обслуговування переривань. Розряди *A*7–*A*5 керувального слова *ICW*1 використовуються для формування молодшого байта адрес підпрограм обслуговування переривань відповідно до табл. 6.21.

Таблиця 6.21. **Молодший байт адреси команди *CALL***

Вхід запиту	Адресний інтервал, 4 байт								Адресний інтервал, 8 байт							
	*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0	*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
*IRQ*7	*A*7	*A*6	*A*5	1	1	1	0	0	*A*7	*A*6	1	1	1	0	0	0
*IRQ*6	*A*7	*A*6	*A*5	1	1	0	0	0	*A*7	*A*6	1	1	0	0	0	0
*IRQ*5	*A*7	*A*6	*A*5	1	0	1	0	0	*A*7	*A*6	1	0	1	0	0	0
*IRQ*4	*A*7	*A*6	*A*5	1	0	0	0	0	*A*7	*A*6	1	0	0	0	0	0
*IRQ*3	*A*7	*A*6	*A*5	0	1	1	0	0	*A*7	*A*6	0	1	1	0	0	0
*IRQ*2	*A*7	*A*6	*A*5	0	1	0	0	0	*A*7	*A*6	0	1	0	0	0	0
*IRQ*1	*A*7	*A*6	*A*5	0	0	1	0	0	*A*7	*A*6	0	0	1	0	0	0
*IRQ*0	*A*7	*A*6	*A*5	0	0	0	0	0	*A*7	*A*6	0	0	0	0	0	0

Керувальне слово *ICW*2 має різний вигляд для МП *i*8080 та *i*8086. У першому випадку (рис. 6.66, *а*) воно являє собою старший байт адреси підпрограми обслуговування переривань, що видається ПКП на шину даних як третій байт команди *CALL*.

У другому випадку (рис. 6.66, *б*) розряди *D*7–*D*3 визначають старші біти номера переривання для кожного входу *IRQ*. Молодші три біти визначаються номером входу *IRQ*, на який надійшов запит переривання.

Рис. 6.66. Формати керувальних слів ініціалізації *ICW*2:
а – для системи з МП *i*8080; *б* – для систем з МП *i*8086

У МПС, що складається з декількох ПКП, для кожного з них після двох перших керувальних слів ініціалізації вводиться слово *ICW*3, що задає режим взаємодії контролерів. У керувальному слові *ICW*3 для ведучого ПКП (рис. 6.67, *а*) наявність логічної одиниці в одному з розрядів указує на приєднання до відповідного входу запиту переривання виходу *INT* веденого ПКП. У керувальному слові *ICW*3 для веденого ПКП (рис. 6.67, *б*) задається код його номера у системі.

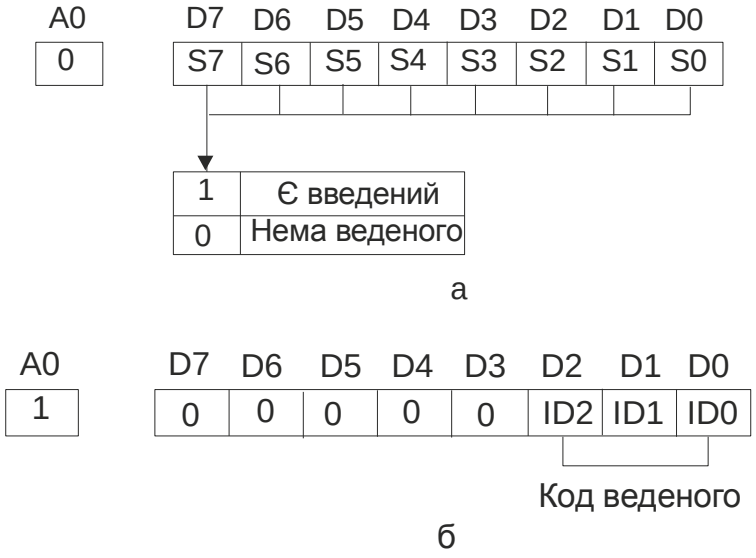


Рис. 6.67. Формати керувальних слів ініціалізації *ICW*3:
а – для ведучого ПКП; *б* – для веденого ПКП

Формат керувального слова ініціалізації *ICW4* зображено на рис. 6.68.

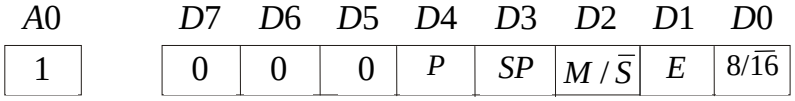


Рис. 6.68. Формат керувального слова ініціалізації *ICW*4

На рис. 6.65–6.68 ліворуч наведено значення входу *A*0, при якому треба завантажувати керувальні слова.

Нульове значення розряду *P*(*D*4), визначає простий пріоритетний режим із фіксованими пріоритетами, одиничне – з циклічними пріоритетами. Розряд *D*3 керує станом лінії *SP*. Розряд $M/\overline{S}$ (*Master/Slave*) (*D*2) дорівнює одиниці для ведучого ПКП і нулю для веденого. Розряд *D*1 задає автоматичне 0 або спеціальне закінчення переривання, а розряд *D*0 дорівнює одиниці в системі з МП *i*8086 і нулю в системі з МП *i*8080.

Після запису керувальних слів ініціалізації ПКП підготовлений до прийому запитів переривання і може працювати в *режимі з фіксованими пріоритетами запитів.* У цьому режимі запити упорядковані за пріоритетами: вхід *IRQ*0 має вищий пріоритет, а *IRQ*7 – нижчий. Цей режим ще називають *режимом повного вкладення підпрограм обробки переривань*. Якщо запити переривань надійдуть у ПКП одночасно, то буде обслуговуватися запит, який має більший пріоритет. Якщо під час обробки одного запита надійшов другий з більш високим пріоритетом, то МП перериває обробку поточного запиту і викликає підпрограму обробки нового запиту. Після його обробки МП продовжує перервану підпрограму обробки. Запити з нижчими пріоритетами не переривають підпрограм обробки запитів з вищими.

Для задання інших режимів функціонування ПКП необхідно використовувати операційні слова *OCW*, що завантажуються після слів ініціалізації у будь-який поточний момент часу.

Операційне слово *OCW*1 (рис. 6.69, *а*) здійснює встановлення або скидання розрядів регістра *RGM*. Установлення деякого розряду регістра маскування приводить до заборони переривання по відповідному входу.

Операційне слово *OCW*2 (рис. 6.69, *б*) здійснює скидання розряду регістра *ISR* і циклічний зсув пріоритетів запитів. Можливі варіанти слова *OCW*2 і його функції наведено в табл. 6.22.

***Таблиця 6.22*. Варіанти слова *OCW*2**

Розряд команди								Режим пріоритетів
*D*7 (*R*)	*D*6 (*SEOI*)	*D*5 (*EOI*)	*D*4	*D*3	*D*2	*D*1	*D*0	
0	0	1	0	0	*x*	*x*	*x*	Фіксований пріоритет: *IRQ*0 – вищий, *IRQ*7 – нижчий
1	0	1	0	0	*x*	*x*	*x*	Циклічний зсув пріоритетів: присвоєння обслуговуваному запиту нижчого пріоритету
0	1	1	0	0	*L*2	*L*1	*L*0	Фіксований пріоритет: *L*2–*L*0 – номер розряду, що скидається
1	1	1	0	0	*L*2	*L*1	*L*0	Циклічний зсув пріоритетів: *L*2–*L*0 – номер розряду, що скидається у регістрі *ISR* (присвоєння йому нижчого пріоритету)
1	1	0	0	0	*L*2	*L*1	*L*0	Циклічний зсув пріоритетів без завершення переривання: *L*2–*L*0 – номер входу *IRQ* з нижчим пріоритетом

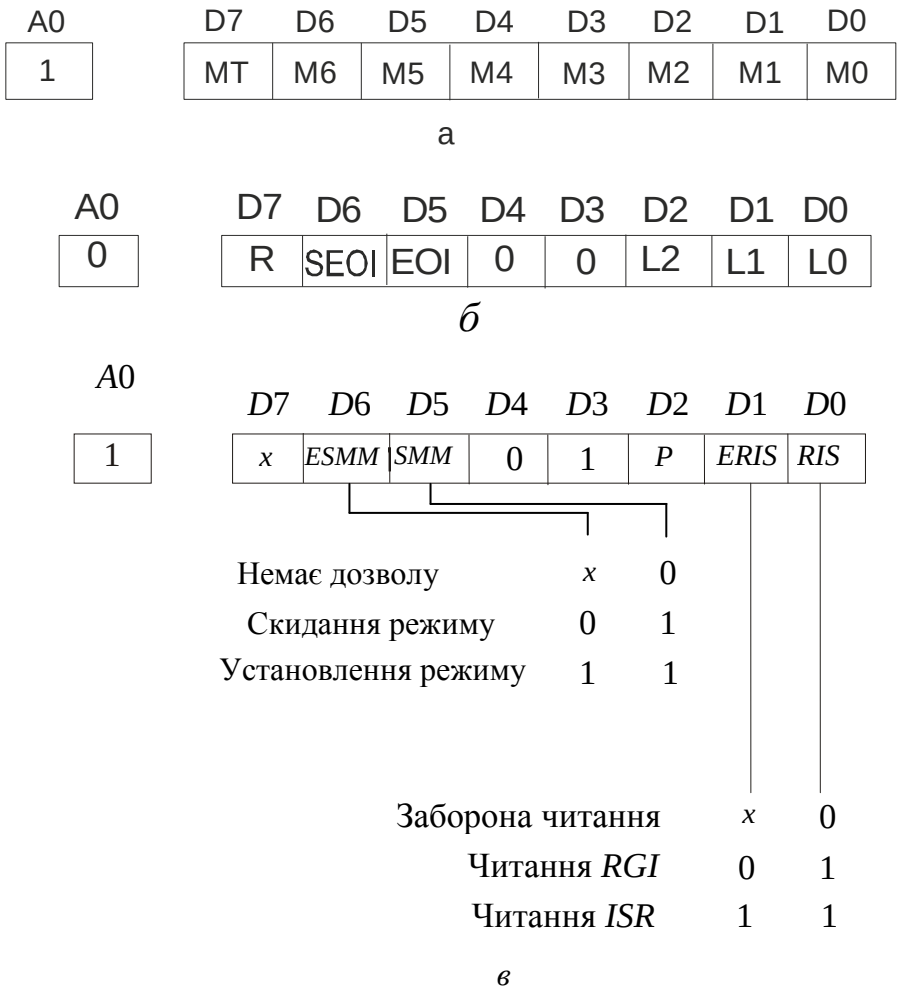


Рис. 6.69. Формати операційних керувальних слів: *а – OCW*1; *б – OCW*2*; в – OCW*3

Якщо обслуговування запиту переривання необхідно закінчити скиданням розряду регістра *ISR* з вищим пріоритетом, то використовується слово *OCW*2 із значеннями *EOI* = 1 і *SEOI* = 1. При *EOI* = 1 і *SEOI* = 1 призначений для скидання рівень переривання вказується в команді розрядами *D*2(*L*2)–*D*0(*L*0). Циклічний зсув пріоритетів задається в команді *OCW*2 розрядом *D*7(*R*).

У циклічному режимі використовується коловий порядок призначення пріоритетів. Останньому обслуговуваному запиту присвоюється нижчий пріоритет, наступному – вищий. Пріоритети інших запитів циклічно зміщуються за шкалою пріоритетів. При *R* = 1 і *SEOI* = 0 команда *OCW*2 присвоює нижчий пріоритет запиту з вищим пріоритетом, а при *R* = 1 і *SEOI* = 1 нижчий пріоритет присвоюється запиту, номер якого вказується розрядами *D*2(*L*2)–*D*0(*L*0). Слово *OCW*2 зазвичай записується у ПКП наприкінці підпрограми обслуговування переривання перед командою повернення з підпрограми *RET*.

Операційне слово *OCW*3 (див. рис. 6.69, *в*) дозволяє задати режим спеціального маскування (розряди *D*6, *D*5), режим опитування (розряд *D*2) і виконати зчитування стану ПКП (розряди *D*1, *D*0).

Режим спеціального маскування дає змогу на деякій ділянці програми вибірково керувати запитами з різними пріоритетами і дозволяти переривання виконуваної програми навіть від входів з меншими пріоритетами. Режим спеціального маскування задається словом *OCW*3 при *ESMM* = 1 і *SMM* = 1 та відміняє цей режим словом з *ESMM* = 1 і *SMM* = 0.

Режим опитування встановлюється за словом *OCW*3 при *P* = 1. У цьому режимі ПКП приймає запити і формує слово стану опитування, що містить номер запиту з найвищим пріоритетом (розряди *D*2–*D*0 слова стану). Запит обслуговується на вимогу програми, яка здійснює за допомогою команди введення *IN* (*A* = 0, *RD* = 0) читання слова стану, програмне декодування його і перехід до відповідної підпрограми обслуговування переривання.

Уміст регістрів *RGI* і *ISR* зчитується після завантаження у ПКП слова *OCW*3 з відповідними значеннями *ERIS* і *RIS* (див. рис. 6.69, *в*) з подальшим виконанням команди введення *IN* (*A*0 = 1 і $\overline{RD}$ = 0). Вміст регістра *RGM* зчитується без попереднього завантаження за командою *IN* (*A*0 = 1, $\overline{RD}$ = 0).

Приклад 6.17. Запрограмувати контролери переривань з каскадним з'єднанням (рис. 6.70) у системі з МП 80*x*86 (в *PC IBM AT*) на режим з фіксованими пріоритетами, неавтоматичним закінченням переривань. Запрограмувати номер переривань для *IRQ*0 ведучого контролера – 08, для веденого – 70*H*. Адреси ведучого контролера 20*H* та 21*H*, веденого – 0*A*0*H*, 0*A*1*H*. Визначити їх номери всіх переривань та пріоритети.

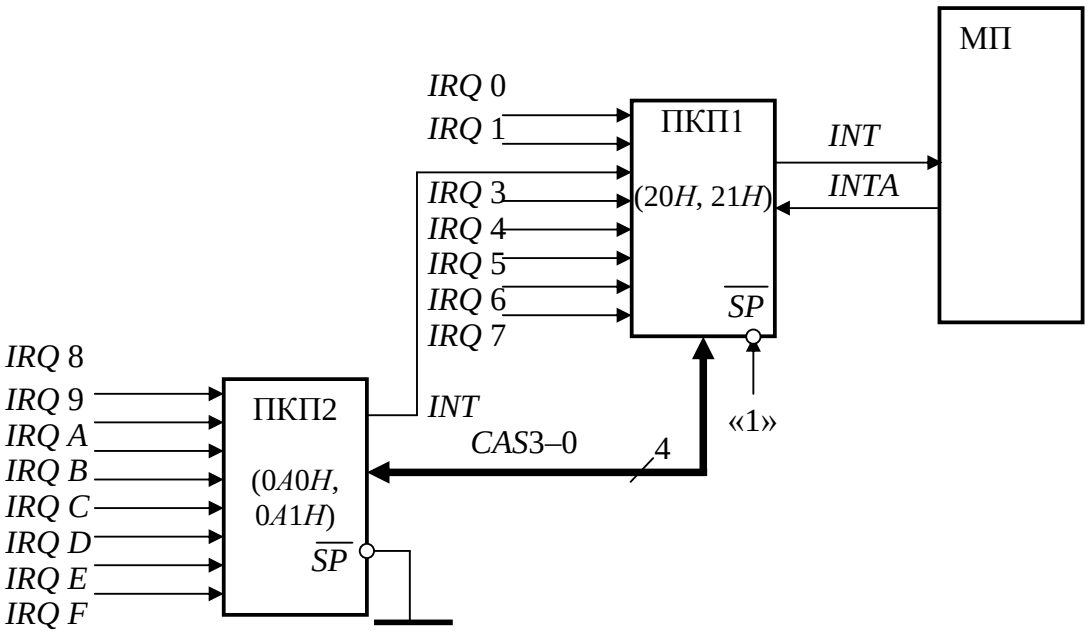


Рис. 6.70. Приклад каскадного з'єднання двох ПКП

Керувальне слово ініціалізації *ICW1* згідно з рис. 6.65 дорівнює 11*H* для обох ПКП.

Запис керувального слова відбувається при *A*0 = 0, тобто при адресі 20*H* для першого ПКП і 70*H* для другого.

Керувальне слово ініціалізації *ICW*2 згідно з рис. 6.66 має вигляд 0 0 0 0 1 *x x x* для ПКП1 і 01110 *x x x*, тобто 08 та 70*H* відповідно.

Керувальне слово ініціалізації *ICW*3, яке необхідне при каскадному з'єднанні згідно з рис. 6.67, *а* для ведучого контролера дорівнює 04. Єдина одиниця в цьому слові визначається підключенням веденого контролера до входу *IRQ*2. Керувальне слово ініціалізації *ICW*3 для веденого контролера визначається номером 2, до якого підключений вихід *INT* ПКП2 (див. рис. 6.70).

Керувальне слово ініціалізації *ICW*4 згідно з рис. 6.68 дорівнює 1*DH* для ПКП1 і 09*H* для ПКП2.

Програма ініціалізації ПКП1 і ПКП2 має такий вигляд:

```
MOV AL, 11H   ; Формування в AL керувального слова ініціалізації ICW1
OUT 20H, AL   ; Пересилання його в ПКП1
OUT 0A0H, AL  ; і в ПКП2
MOV AL, 08H   ; Формування в AL керувального слова ініціалізації ICW2
OUT 21H, AL   ; Пересилання його в ПКП1
MOV AL, 70H   ; Формування в AL керувального слова ініціалізації
              ; ICW2
OUT 0A1H, AL  ; Пересилання його в ПКП2
MOV AL, 04H   ; Формування в AL керувального слова ініціалізації
              ; ICW3
OUT 21H, AL   ; Пересилання його в ПКП1
MOV AL, 02H   ; Формування в AL керувального слова ініціалізації
              ; ICW3
OUT 0A1H, AL  ; Пересилання його в ПКП2
MOV AL, 0DH   ; Формування в AL керувального слова ініціалізації
              ; ICW4
OUT 21H, AL   ; Пересилання його в ПКП1
MOV AL, 09H   ; Формування в AL керувального слова ініціалізації
              ; ICW4
OUT 0A1H, AL  ; Пересилання його в ПКП2
```

Номери переривань та їх пріоритетну схему наведено в табл. 6.23.

Таблиця 6.23. **Пріоритети переривань**

Схема пріоритетів	Номер входу переривання	Команда переривання
Найвищий	*IRQ* 0	*INT* 8
↑	*IRQ* 1	*INT* 9
	IRQ 2	*INT* 0*AH*
	IRQ 8	*INT* 70*H*
	IRQ 9	*INT* 71*H*
	IRQ 0*AH*	*INT* 72*H*
	IRQ 0*BH*	*INT* 73*H*
	IRQ 0*CH*	*INT* 74*H*
	IRQ 0*DH*	*INT* 75*H*
	IRQ 0*EH*	*INT* 76*H*
	IRQ 0	*INT* 77*H*
	IRQ 3	*INT* 0*BH*
	IRQ 4	*INT* 0*CH*
	IRQ 5	*INT* 0*DH*
	IRQ 6	*INT* 0*EH*
Найнижчий	*IRQ* 7	*INT* 0*FH*

Зазначимо, що в схемі з двома ПКП (див. рис. 6.70) максимальна кількість запитів дорівнює 15. Це пов'язано з тим, що один із входів ведучого контролера з'єднаний з виходом веденого.

Приклад 6.18. Заборонити обробку переривання *IRQ*2. Прийняти адресу ПКП1 як у прикл. 6.17.

Для того щоб заборонити переривання, потрібно записати одиницю у відповідний розряд регістра маски, як на рис. 6.69, *а*.

Програма має такий вигляд:

```
IN AL, 21H    ; Прочитати вміст регістра маски
OR AL, 04     ; Установити розряд D2 в одиницю
MOV AL, 70H   ; Переслати в ПКП1
```

Приклад 6.19. Дозволити обробку переривання *IRQ*2. Прийняти адресу ПКП1 як у прикл. 6.17.

Для того щоб дозволити переривання, потрібно записати нуль у відповідний розряд регістра маски, як на рис. 6.69, *а*.

Програма має такий вигляд:

```
IN AL, 21H          ; Прочитати вміст регістра маски
AND AL, 11111011B   ; Установити розряд D2 в одиницю
MOV AL, 70H         ; Пересилати в ПКП1
```

Приклад 6.20. Скласти підпрограму введення даних із порту з адресою 300*H* у комірку пам'яті з адресою *DS*:*SI* по перериванню.

Підпрограма обробки переривання має такий вигляд:

```
; Зберігання регістрів МП у стеку
PUSH AX         ; Переслати AX у стек
PUSH BX         ; Переслати BX у стек
       …

MOV DX, 300H    ; Записати в DX адресу порту
IN AL, DX       ; Увести дані
MOV [SI], AL    ; Переслати в комірку пам'яті
       …
; Закінчення переривання
MOV AL, 20H     ; запис в AL керувального слова OCW2
OUT 20H,AL
; Повернення вмісту регістрів МП зі стеку (у зворотному порядку)
       …
POP BX
POP AX
STI             ; Дозвіл переривань
IRET            ; Повернення з підпрограми обробки переривання
```

Контрольні запитання

1. Які функції виконує ПКП у МПС керування?
2. Яку кількість запитів переривань може обслужити одна ВІС ПКП?
3. Якої максимальної кількості запитів переривань можна досягти в системі переривань на базі ВІС КР580ВН59?
4. Для чого призначений режим спеціального маскування?

5. Які пріоритетні схеми обробки запитів переривань реалізуються у ВІС КР580ВН59?
6. Як змінюються пріоритети у схемі циклічної обробки пріоритетів?
7. Коли доцільно використовувати режим фіксованих пріоритетів?
8. Коли доцільно використовувати режим циклічних пріоритетів?
9. Який із входів *IRQ* має вищий пріоритет?
10. Як можна зменшити пріоритет входів *IRQ*?
11. Як можна вимкнути одну або декілька ліній *IRQ*?
12. Коли можуть записуватися команди ініціалізації та керування?
13. Для чого необхідні сигнали *INT* і *INTA*?

6.8. Приклад розробки мікропроцесорної системи

Нехай треба розробити принципову схему та програмне забезпечення МПС керування широтно-імпульсного стабілізатора напруги із частотою f = 1 кГц. МПС керування забезпечує обробку сигналу зворотного зв'язку U із виходу АЦП за законом пропорційного регулятора:

$$\gamma = 0{,}5 + (U - 20H)/256,$$

де γ – коефіцієнт заповнення імпульсів широтно-імпульсного стабілізатора; $20H$ – код опорного сигналу.

Структурну схему МПС керування показано на рис. 6.71.

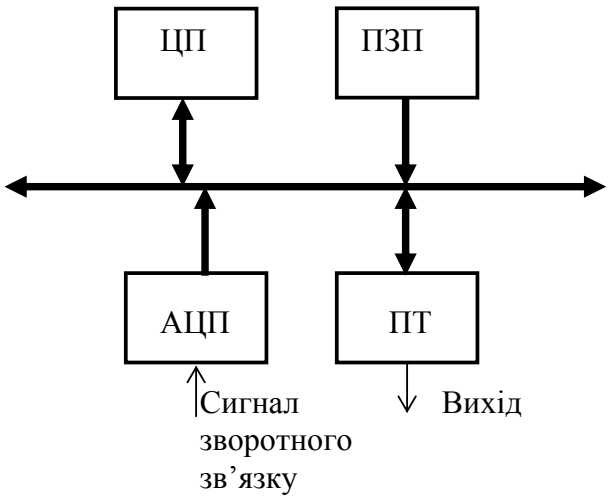


Рис. 6.71. Структурна схема МПС керування

Сигнал зворотного зв'язку надходить на АЦП, де він перетворюється на двійковий код. Цей код по системній шині надходить у ЦП, де за програмою, записаною у ПЗП, обробляється за законом пропорційного регулятора. Результатом обчислення є коефіцієнт заповнення, який завантажується в регістр таймера як константа. На виході таймера одержуємо логічні рівні імпульсів керування широтно-імпульсного стабілізатора.

Функціональну схему МПС керування показано на рис. 6.72.

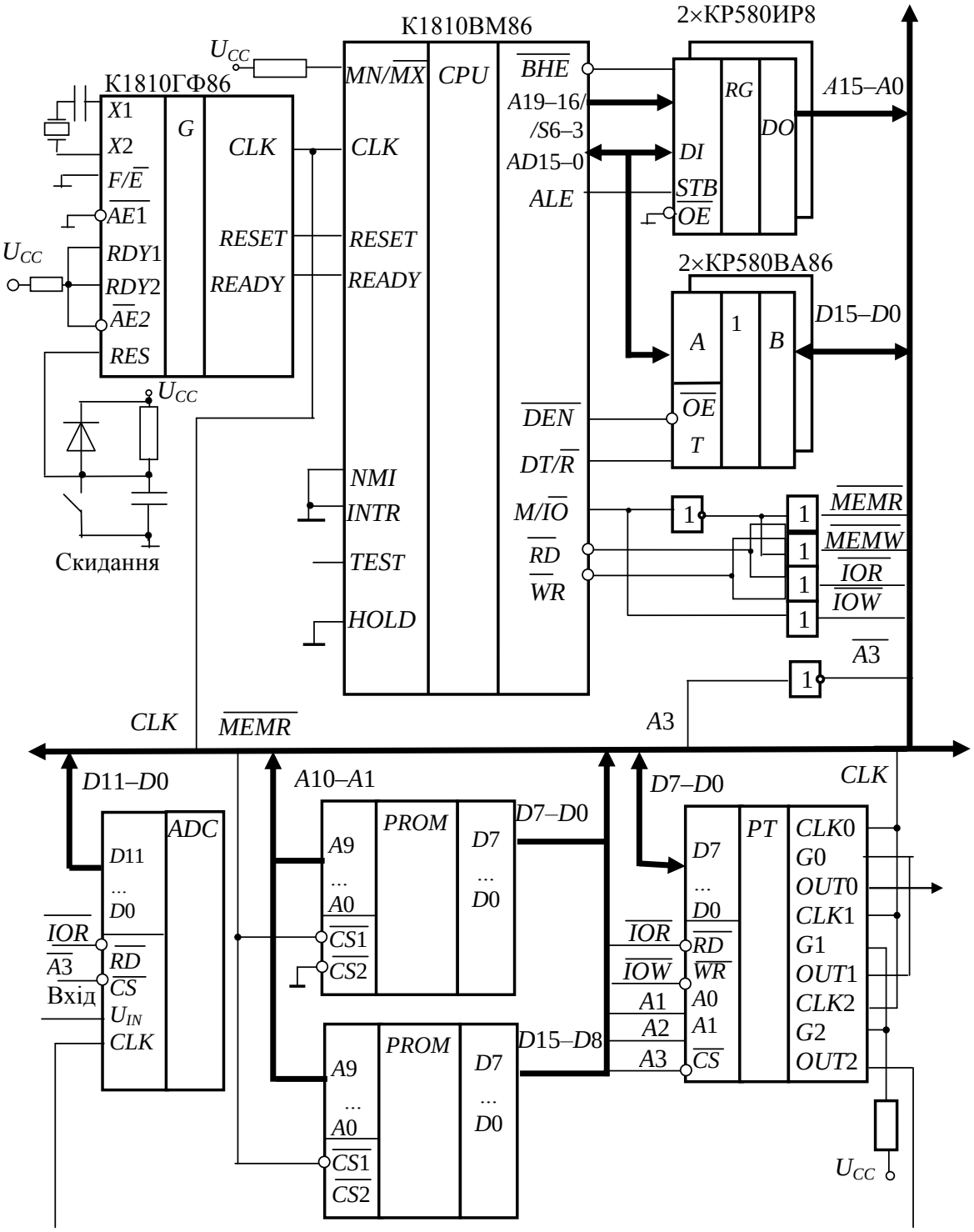


Рис. 6.72. Функціональна схема МПС

Під час розробки функціональної схеми ЦП виникає потреба у демультиплексуванні шини адреси/даних, буферизації шин адреси (*AB*) і шин даних (*BD*), а також у формуванні системних керувальних сигналів пам’яті та зовнішніх пристроїв. Демультиплексування здійснюється за допомогою двох ВІС К1810ИР82, що виконують функції фіксатора

адреси і буфера шини *AB*. Буферізація шин даних виконується за допомогою двонапрямлених шинних формувачів К1810ВА86, які підсилюють сигнали шини даних і формування керувальних сигналів виконується за допомогою комбінаційних логічних елементів. На виході цих елементів формуються сигнали $\overline{MEMR}$, $\overline{MEMW}$, $\overline{IOR}$, $\overline{IOW}$.

Оскільки в розроблюваній МПС не потрібні режими ПДП, переривань і обміну за сигналом готовності, то порівняно зі схемою модуля ЦП (див. рис. 3.34) схема на рис. 6.72 не містить сигналів переривань, готовності, запиту ПДП і дозволу шин *BUSEN*. Тому на входи ВІС генератора, ЦП, регістрів-фіксаторів і буферних регістрів подано постійні логічні рівні нуля або одиниці. Немає також кінцевих каскадів схеми формування керувальних сигналів (у цій схемі немає потреби переводити керувальні сигнали у третій стан).

Модуль ПЗП виконаний на базі двох ВІС КР556РТ5 ємністю 512×8 біт кожна. Пам'ять організовано у вигляді двох банків пам'яті – молодшого і старшого. Молодший банк вмикається до молодшої половини шини даних *D*7–*D*0 і містить тільки комірки пам'яті з парними адресами; старший банк – до старшої половини шини даних *D*15–*D*8 і містить тільки комірки пам'яті з непарними адресами. Зчитування з ПЗП відбувається при виконанні циклу читання пам'яті. При цьому формується сигнал $\overline{MEMR} = 0$, який і переводить виходи ВІС ПЗП в активний стан. Із ПЗП завжди зчитується слово. Початкову адресу ПЗП визначимо при нульових значеннях *A*9–*A*1, *A*0, а кінцеву – при одиничних.

Таким чином, початкова адреса ПЗП – 00000*H*, кінцева адреса ПЗП – 003*FFH*.

Функціональна схема містить також АЦП К572ПВ1, який являє собою 12-розрядний перетворювач напруги на двійковий код низької швидкодії. Оскільки АЦП має внутрішній регістр із входом керування третім станом, зовнішній порт уведення не потрібний. Вихід АЦП з'єднаний з лініями *D*11–*D*0. Для МП АЦП являє собою 16-розрядний порт. Адреса 16-розрядного порту має бути парною. Як видно з рис. 6.72, АЦП вибирається при *A*3 = 1. Таким чином, адреса АЦП може бути будь-якою при *A*3 = 1, *A*0 = 0. Наприклад, оберемо адресу АЦП, що дорівнює 08*H*.

Програмовний таймер К1810ВИ54 у схемі (див. рис. 6.72) призначений для генерації імпульсів керування широтно-імпульсним стабілізатором. Таймер містить три незалежні канали, кожний з яких може бути запрограмований на роботу в одному із шести режимів для двійкового та двійково-десяткового обчислення. Використовують такі режими роботи каналів:

- канал 0 – програмовний мультивібратор (режим 1);
- канал 1 – імпульсний генератор частоти для запуску каналу 0 (режим 2);
- канал 2 – імпульсний генератор для задавання частоти роботи АЦП (режим 2).

Як видно із рис. 6.72, таймер обирається при адресі з $A3 = 0$. Лінії $A1$ і $A2$ обирають один з трьох каналів таймера або регістр керувального слова. Отже, адресами таймера будуть:

– адреса каналу 0 – 00*H*;
– адреса каналу 1 – 02*H*;
– адреса каналу 2 – 04*H*;
– адреса *RWCU* – 06*H*.

Розрахуємо константи завантаження таймера так. Перетворимо коефіцієнт заповнення широтно-імпульсного стабілізатора *gamma* на константу перерахунку, що завантажується в таймер. Зауважимо, що вихідна частота стабілізатора дорівнює 1 кГц, а частота тактових імпульсів $f_{CLK} = 5$ МГц.

Канал 0. Коефіцієнт заповнення імпульсів широтно-імпульсного стабілізатора

$$gamma = 0{,}5 + \frac{1}{256}(I - 20H).$$

Визначимо період роботи широтно-імпульсного стабілізатора як

$$T = \frac{1}{f} = \frac{1}{10^3} = 1 \text{ мс}.$$

Тоді тривалість імпульсу

$$\tau_i = \frac{T}{2} + \frac{T}{256}(U - 20H) = 0{,}5 + \frac{1}{256}(U - 20H) \text{ мс}.$$

Тривалість лічильних імпульсів каналів (*CLK*) при частоті роботи процесора 5 МГц

$$T_{CLK} = 1/(5 \cdot 10^6) = 200 \text{ нс}.$$

Код завантаження каналу 0 таймера визначиться як відношення τ_i до T_{CLK}, тобто

$$N_0 = \frac{0{,}5 \cdot 10^{-3}}{200 \cdot 10^{-9}} + \frac{10^{-3}}{256 \cdot 200 \cdot 10^{-9}}(U - 20H) \approx 2500 + 20(U - 20H).$$

Канал 1. Коефіцієнт ділення каналу 1 визначається як відношення тривалості періоду широтно-імпульсного стабілізатора до тривалості періоду T_{CLK}:

$$N_1 = \frac{T}{T_{CLK}} = \frac{1000000}{200} = 5000.$$

Канал 2. Коефіцієнт ділення каналу 2 визначається як відношення тривалості періоду тактових імпульсів АЦП (170 кГц) до тривалості періоду T_{CLK}:

$$N_2 = \frac{T_{\text{АЦП}}}{T_{CLK}} = 29.$$

Алгоритм керування широтно-імпульсним стабілізатором показано на рис. 6.73. Спочатку в обидва канали таймера завантажуються керувальні слова. Керувальне слово для каналу 0 дорівнює 00110010*B* і визначає режим 1 лічильника 0, завантаження спочатку молодшого, а потім старшого байта, лічбу – двійкову. Керувальне слово для каналів 1 і 2 визначає режим 1, завантаження спочатку молодшого, а потім старшого байтів, лічбу – двійкову. Керувальні слова дорівнюють 01110100*B* і 10110100*B* відповідно.

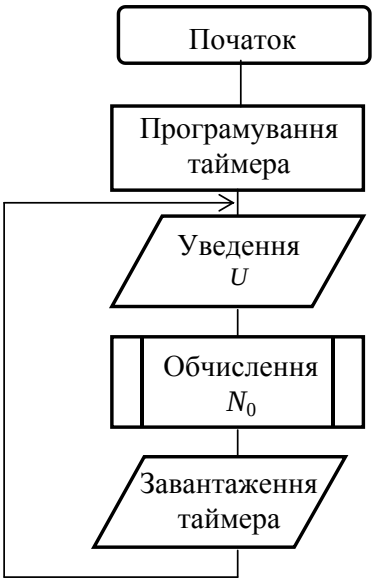


Рис. 6.73. Алгоритм керування широтно-імпульсним стабілізатором

Після завантаження керувальних слів програма завантажує початкові значення у канали 1 та 2 таймера. На цьому ініціалізація закінчується і починається нескінченний цикл уведення коду зворотної напруги з АЦП, перерахування її у код завантаження і виведення його у канал 0.

Зазначимо, що після системного скидання регістри *CS* і *IP* МП набудуть значень 0*FFFFH* і 0000*H* відповідно. Це призведе до того, що в адресі першої комірки ПЗП, до якої після скидання звернеться МП у першому машинному циклі ВИБІРКА КОМАНДИ, чотири молодші розряди будуть нульовими, а всі інші – одиничними. Тоді ця адреса буде дорівнювати 3*F*0*H*.

Програма має такий вигляд:

```
        ORG 3F0H
        JMP START
        ORG 100H
START:  MOV AL, 00110010B          ; Формування в AL керувального
                                   ; слова режиму 1 каналу 0
        OUT 06, AL                 ; Виведення керувального слова
                                   ; в RCW
        MOV AL, 01110100B          ; Формування в AL керувального
                                   ; слова в RCW
```

```
        OUT 06, AL                  ; Виведення керувального слова
                                    ; режиму 2 каналу 1
        MOV AL, 10110100B           ; Формування в AL керувального
                                    ; слова режиму 2 каналу 2
        OUT 06, AL                  ; Виведення керувального слова
                                    ; режиму RCW
        MOV AX, 5000                ; Задавання частоти
                                    ; перетворення ƒ = 1кГц
        OUT 02H, AL                 ; Запис молодшого байта коду
                                    ; попереднього встановлення
                                    ; в канал 1
        MOV AL, AH                  ; Запис старшого байта коду
                                    ; попереднього встановлення
                                    ; в канал 1
        OUT 02H, AL                 ; Задавання частоти АЦП
                                    ; ƒ = 170 кГц
        MOV AL, 29                  ; Запис коду попереднього
                                    ; встановлення в канал 2
        OUT 04H, AL
        MOV AL, 00
        OUT 04H, AL
L:      IN AX,08H                   ; Уведення сигналу
                                    ; зворотного зв'язку U
        AND AX, 0000 1111 1111 1111B ; Виділення значущих цифр U
        SUB AX, 20H                 ; Віднімання U – 20H,
                                    ; результат – у AX
        MOV BX, 20
        MUL BX                      ; Множення на 20, результат –
                                    ; в (DX, AX)
        ADD AX, 2500                ; Додавання
                                    ; 2 500 + 20(U – 20H),
                                    ; результат – в AX
        OUT 00H, AL                 ; Запис молодшого байта коду
                                    ; попереднього встановлення
                                    ; в канал 0
        MOV AL, AH
        OUT 00H, AL                 ; Запис старшого байта коду
                                    ; попереднього встановлення
                                    ; в канал 0
        MOV CX, 100                 ; Затримка на час, більший
                                    ; ніж час перетворення АЦП
D:      LOOP D                      ; t = 16·100·200 нс
        JMP L                       ; Перейти за адресою
                                    ; з міткою L (цикл)

        END
```

6.9. Спеціалізовані співпроцесори

Арифметичний помножувач. Побудова МПС, що працюють у реальному масштабі часу, вимагає підвищення швидкості передачі інформації, що може бути досягнено за рахунок: зменшення затримок у шині та вдосконалення протоколу обміну; збільшення продуктивності обчислювальних засобів; оптимізації алгоритмів обробки. Ці завдання можуть бути вирішені в разі використання спеціалізованих ВІС арифметичної обробки інформації, що є більш ефективними при роботі МПС у реальному масштабі часу, ніж використання арифметичних співпроцесорів. До таких ВІС належать: помножувач-акумулятор К1518ВЖ1; схема реалізації швидкого перетворення Фур'є К1815ВФ3; цифровий процесор обробки сигналів з аналоговим уведенням-виведенням інформації КМ1813ВЕ1.

Розглянемо архітектуру і функціональні можливості ВІС К1518ВЖ1, яка являє собою 16-розрядний помножувач-акумулятор, що працює в двійковій системі числення. ВІС має схемотехніку емітерно-пов'язаної логіки і виконує операцію множення з накопиченням за час до 155 нс. У вхідних і вихідних вузлах використані транслятори рівнів напруг, що забезпечують сумісність логічних рівнів сигналів ВІС із сигналами ТТЛ-схем.

Структурна схема (рис. 6.74) містить: матричний помножувач, два вхідні регістри, суматор множення, регістр множення, схему керування і тристабільні вихідні каскади.

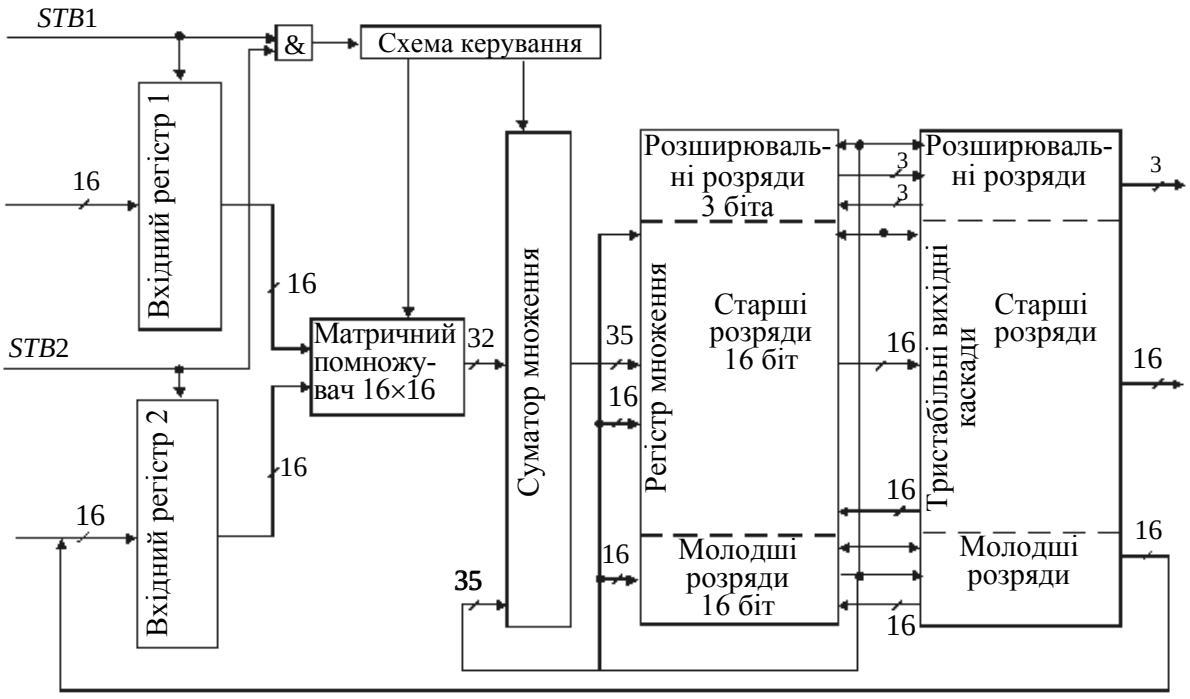


Рис. 6.74. Структурна схема ВІС К1518ВЖ1

Помножувач-акумулятор забезпечує виконання операції множення двох 16-розрядних чисел (у тому числі з накопиченням добутків або відніманням попереднього добутку), попереднє завантаження вихідного регістра добутку (акумулятора), зберігання результату множення. Арифметичні операції виконуються із цілими двійковими числами без знаку і з двійковими числами в додатковому коді. Знаковий розряд у додатковому коді розміщується ліворуч від старшого значимого розряду. Нульове значення знакового розряду означає додатне число, одиничне значення – від'ємне число. Відокремлення цілої та дробової частини здійснюється за допомогою коми, яка знаходиться між знаковим розрядом і старшим значущим розрядом. Вихідний формат, на відміну від вхідного, має додаткове (розширене) знакове поле (3 розряди) для розміщення цілої частини при накопиченні. Введення інформації відбувається за фронтом імпульсів синхронізації. Акумулятор завантажується перед виконанням множення з накопиченням і множення з відніманням. Зберігання результату множення може виконуватися після виконання будь-якої операції

Недоліком ВІС є велика потужність споживання – до 5,5 Вт, проте висока швидкодія, значний набір функцій при відносно простому керуванні обумовлює можливість її широкого застосування для вирішення задач цифрової фільтрації, спектрального аналізу і керування пристроями перетворювальної техніки, які містять досить велику кількість арифметичних операцій множення з накопиченням.

Приєднання помножувача найбільш просто здійснюється за допомогою шинних формувачів, наприклад ВІС КР580ВА86 і КР580ВА87, які не потребують попереднього програмування, але на відміну від буферних регістрів, мають можливість двонапрямленої передачі даних. Мінімально можливий час перемножування двох 8-розрядних двійкових чисел з одержанням 16-розрядного результату досягається при заземленні 8 молодших розрядів регістрів співмножників і зчитуванні результату з 16 старших розрядів регістра добутку за допомогою однієї команди виведення. Завантаження обох співмножників у регістри відбувається одночасно (старший байт 16-розрядного слова завантажується в регістр першого співмножника, а молодший – у регістр другого), що вдвічі зменшує час виведення інформації з керувальної машини у помножувач. Таке приєднання арифметичного помножувача дозволяє використовувати його в усіх можливих режимах.

Арифметичний співпроцесор. Арифметичний співпроцесор призначений для підвищення продуктивності ЦП при виконанні арифметичних операцій із багаторозрядними цілими і дійсними числами. Принцип роботи співпроцесора розглянемо на прикладі ВІС К1810ВМ87. Цей співпроцесор може бути використаний тільки разом із ЦП К1810ВМ86/88, тому що в ньому немає механізму вибірки команд. Арифметичний співпроцесор оперує даними семи

різних форматів (рис. 6.75): цілі двійкові числа (три формати), цілі двійково-десяткові числа (один формат), дійсні числа з плавучою комою (три формати).

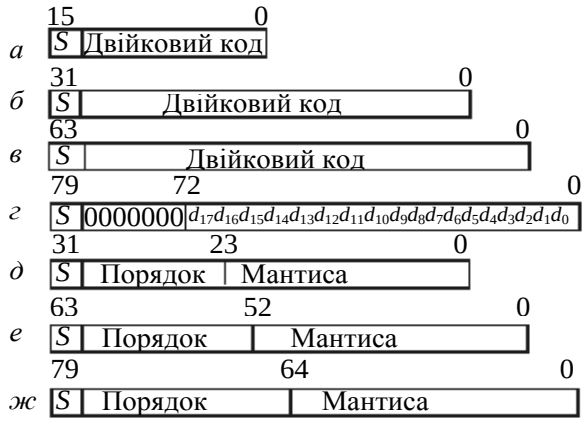


Рис. 6.75. Структура форматів арифметичного співпроцесора К1810 ВМ87: *а* – цілі 16-розрядні двійкові числа; *б* – цілі 32-розрядні двійкові числа; *в* – цілі 64-розрядні двійкові числа; *г* – двійково-десяткове число; *д* – 32-розрядне число з плавучою комою; *е* – 64-розрядне число з плавучою комою; *ж* – 80-розрядне число з плавучою комою

Суміщення ЦП К1810ВМ86 із співпроцесором К1810ВМ87 виконується з'єднанням відповідних виводів без використання додаткових інтегральних схем.

Структурна схема співпроцесора (рис. 6.76) містить:

1) операційний пристрій, що виконує задані операції;
2) шинний інтерфейс, що одержує і декодує команди, зчитує операнди з пам'яті, перетворює їх на формат уведення-виведення, а також записує результати в пам'ять із зворотним перетворенням на необхідний формат.

Обидва пристрої можуть працювати паралельно, що забезпечує суміщення у часі процесів передачі та перетворення даних.

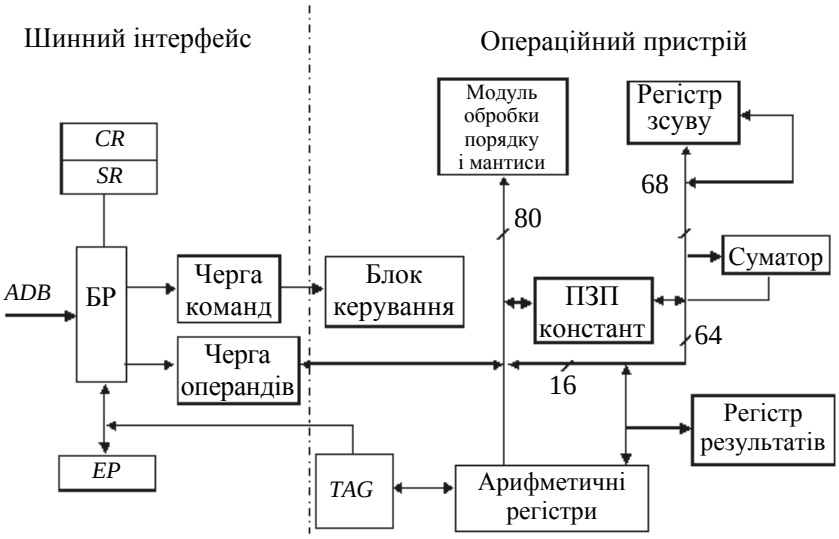


Рис. 6.76. Структурна схема арифметичного співпроцесора

Операційний пристрій містить: групу арифметичних регістрів, модулі обробки порядку і мантиси, ПЗП констант, регістр *TAG* тегів (ознак), блок керування та суматор.

Пристрій шинного інтерфейсу містить групу допоміжних регістрів *CR*, *SR*, *EP*, буферний регістр БР, черги команд і операндів.

Спеціалізований процесор уведення-виведення К1810ВМ89. Спеціалізований процесор уведення-виведення (СПВВ) К1810ВМ89 використовується разом із ЦП К1810ВМ86/ВМ88 (*i*8086/*i*8088) і призначений для підвищення продуктивності систем на базі МПК К1810 завдяки звільненню ЦП від керування введення-виведення і здійсненню високошвидкісних пересилок у режимі ПДП. До основних функцій СПВВ ВМ89 належать: ініціалізація та керування контролерами зовнішніх пристроїв, забезпечення гнучких і універсальних пересилок із ПДП. Процесор може працювати паралельно із ЦП одночасно по двох каналах уведення-виведення, кожний з яких забезпечує швидкість передачі інформації до 1,25 Мбайт/с при стандартній частоті 5 МГц. Організація зв'язку СПВВ із ЦП через пам'ять підвищує гнучкість взаємодії та полегшує створення модульного програмного забезпечення, що підвищує надійність розроблюваних систем.

Процесор уведення-виведення має два ідентичні канали введення-виведення, кожний з яких містить п'ять 20-бітових, чотири 16-бітових і один 4-бітовий регістри. Взаємодія каналів при паралельній роботі здійснюється під керуванням вбудованої логіки пріоритетів. Процесор має 16-бітову шину даних для зв'язку з ОЗП і портами введення-виведення. Шина адреси містить 20 ліній, що дозволяє безпосередньо адресуватися до пам'яті ємністю до 1 Мбайт. Для економії кількості виводів ВІС молодші 16 адресних ліній мультиплексовані у часі з лініями даних і становлять єдину шину адреси/даних. Чотири старші адресні лінії аналогічно мультиплексовані із лініями стану СПВВ. Щоб сигнали цих ліній можна було використовувати в МПС, їх обов'язково демультиплексують або за допомогою зовнішніх схем, що використовуються у ЦП (у місцевій конфігурації), або за допомогою незалежних схем (у віддаленій конфігурації).

Система команд СПВВ містить 53 мнемокоди, причому можливості і набір команд оптимізовані спеціально для гнучкої, ефективної та швидкої обробки даних при введенні-виведенні. СПВВ дозволяє з'єднувати 16- і 8-розрядні шини і периферійні пристрої. Використовуючи процесор у віддаленому режимі, користувач програмно може визначити різні функції шини СПВВ, яку легко з'єднують зі стандартною шиною *MULTIBUS*.

Внутрішня структура СПВВ підпорядкована основному призначенню ВІС – пересилати дані без безпосереднього втручання ЦП, який зв'язується

зі СВПП тільки для ініціалізації і видачі завдання на обробку. В обох випадках ЦП попередньо готує необхідне повідомлення в пам’яті і потім за допомогою сигналу запиту готовності каналу активізує СВПП на виконання дій, визначених у повідомленні. З цього моменту СПВВ працює незалежно від ЦП. У процесі виконання завдання або по його завершенні СПВВ може зв’язатися із ЦП за допомогою сигналу запиту переривання.

Контрольні запитання

1. Укажіть функцію арифметичного помножувача у МПС.
2. З якими типами даних оперує арифметичний помножувач?
3. Як відбувається приєднання арифметичного співпроцесора до системної шини?
4. Укажіть призначення та область використання арифметичного співпроцесора.
5. З якими форматами даних працює арифметичний співпроцесор?
6. Назвіть особливості системи команд ВІС К1810ВМ89.

Розділ 7

ОДНОКРИСТАЛЬНІ МІКРОКОНТРОЛЕРИ З *CISC*-АРХІТЕКТУРОЮ

Однокристальний мікроконтролер являє (ОМК) собою пристрій, виконаний конструктивно в одному корпусі ВІС, що містить усі компоненти МПС: процесор, пам'ять даних, пам'ять програм, програмовні інтерфейси. Для ОМК характерні:

- система команд, орієнтована на виконання завдань керування і регулювання;
- алгоритми, що реалізуються на ОМК, мають багато розгалужень залежно від зовнішніх сигналів;
- дані, якими оперують ОМК, не повинні мати велику розрядність;
- схемна реалізація систем керування на базі ОМК нескладна і має невисоку вартість;
- універсальність і можливість розширення функцій керування значно нижчі, ніж у системах із однокристальними МП.

Однокристальні мікроконтролери являють собою зручний інструмент для створення сучасних вбудованих пристроїв керування різноманітним обладнанням, як-то автомобільною електронікою, побутовою технікою, мобільними телефонами тощо.

7.1. Архітектура і функціональні можливості однокристальних мікроконтролерів

Структуру ОМК та функціонування основних блоків розглянемо на прикладі ВІС К1816ВЕ51 (*i*80*x*51) *MCS*51 (рис. 7.1). Графічне позначення мікросхеми показано на рис. 7.2.

Структурна схема ОМК містить:

- блок 8-розрядного ЦП;
- пам'ять програм ПЗП ємністю 4 кбайт;
- пам'ять даних ОЗП ємністю 128 байт;
- чотири 8-розрядні програмовні порти введення-виведення *P*0–*P*3;
- послідовний порт;
- два 16-розрядні програмовні таймери/лічильники *T*/*C*0, *T*/*C*1;
- систему переривань з п'ятьма векторами і двома рівнями пріоритетів;
- блок керування (БК).

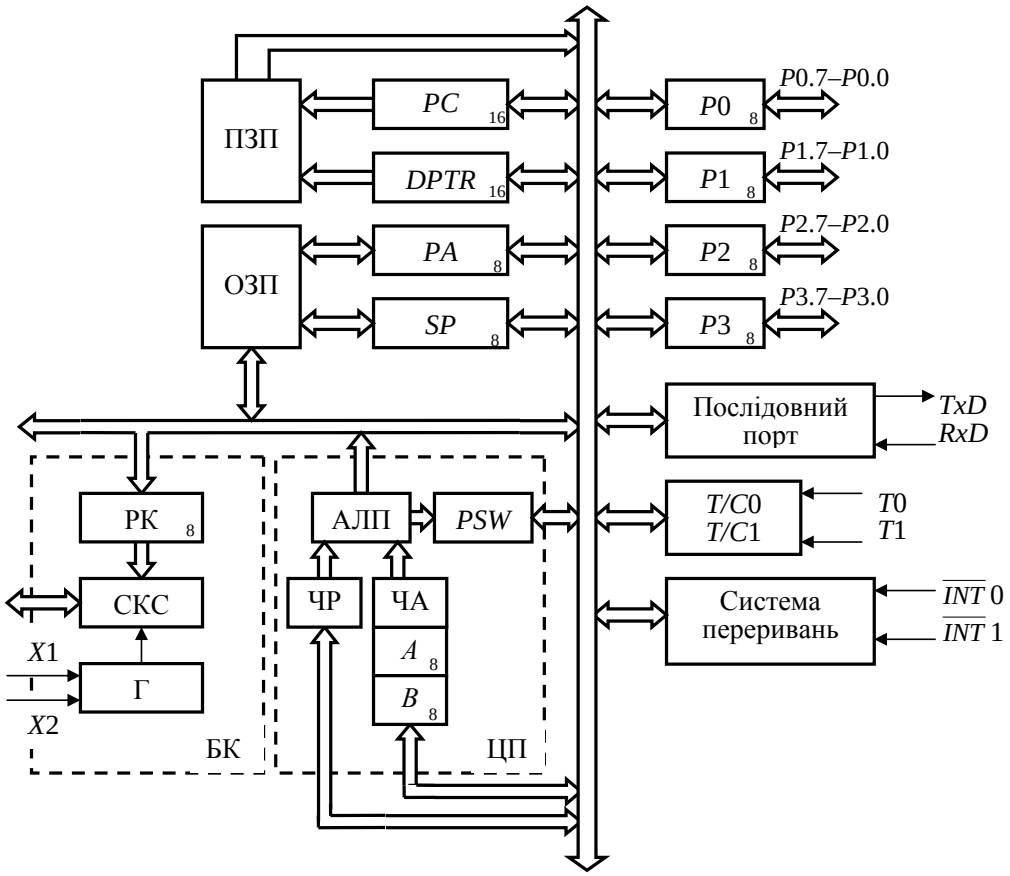
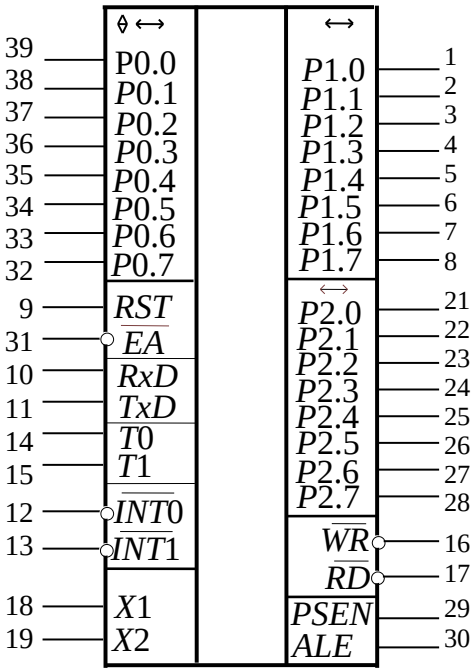


Рис. 7.1. Структурна схема ОМК К1816ВЕ51: ЧР – часовий регістр; ЧА – часовий акумулятор; СКС – схеми керування та сигналізації; Г – генератор



КР1816ВЕ51 (20 = *GND*, 40 = +5 *V*)

Рис. 7.2. Графічне позначення ОМК К1816ВЕ51

Блок ЦП містить 8-розрядний АЛП, два акумулятори *A* і *B*, регістр слова стану процесора *PSW (Processor State Word*) та програмно недоступні буферні регістри ЧА і ЧР, що виконують функції розподілу вхідних та вихідних даних АЛП. ЦП виконує операції додавання, віднімання, множення, ділення, логічні операції І, АБО, НЕ, ВИКЛЮЧНЕ АБО, операції зсуву і скидання. Він оперує з такими типами змінних: булевими (1 біт), цифровими (4 біт), байтовими (8 біт) і адресними (16 біт). Характерною особливістю ОМК є великий набір операцій з бітами: окремі біти змінних можуть бути встановлені, скинуті, інвертовані, перевірені, передані. Це дозволяє легко реалізовувати алгоритми, що містять операції над булевими змінними типу «так-ні» *(«true-false»)*.

Акумулятор *A* є джерелом одного з операндів і місцем розміщення результату виконання багатьох команд. Ряд команд, наприклад, передача інформації в/із ОЗП, команди *TEST, INC, DEC* можуть виконуватися без участі акумулятора. Акумулятор *B* використовується як акумулятор лише в командах множення і ділення, а в інших випадках – як РЗП.

Регістр слова стану процесора *PSW* зберігає інформацію про стан АЛП у процесі виконання програми і має формат, наведений у табл. 7.1.

Таблиця 7.1. **Формат слова стану *PSW***

Біт	Позначення	Призначення	Доступ до біта
7	*C*	Прапорець перенесення	А або П
6	*AC*	Прапорець додаткового перенесення	А або П
5	*F*0	Прапорець користувача	П
4	*RS*1	Вказівник банку робочих регістрів: 00 – банк 0; 10 – банк 2; 01 – банк 1; 11 – банк 3	П
3	*RS*0		
2	*OV*	Прапорець переповнення	П
1	–	Резервний	П
0	*P*	Біт парності	А або П

Примітка. А – біт установлюється апаратно, П – програмно.

Призначення прапорців *C, AC* аналогічне призначенню прапорців *CF, AF* у МП *i*8086 (див. підрозд. 3.2, табл. 3.10). Прапорець *OV* встановлюється у командах додавання і віднімання, якщо результат перевищує ємність 7 біт і старший біт не може бути інтерпретований як знаковий; у командах ділення *OV* скидається, а при діленні на нуль – встановлюється. У командах множення *OV* набуває значення логічної одиниці, якщо результат перевищує 0*FFH*. Прапорець *P* є доповненням умісту акумулятора *A* до парності, тобто 9-розрядне слово, яке складається з 8 біт акумулятора *A* і біта *P*, має завжди парну кількість одиниць.

Постійний запам'ятовувальний пристрій або резидентна пам'ять програм (РПП) має інформаційну ємність 4 кбайт і виконаний у вигляді ПЗП, програмовного маскою. Інші ОМК, наприклад К1816ВЕ751, мають ПЗП *EPROM*. ПЗП має 16-розрядну адресну шину, що дозволяє розширити пам'ять до 64 кбайт через приєднання зовнішніх ВІС ПЗП. Адреса визначається вмістом

лічильника команд *PC* (*Program Counter*) або вмістом регістра-вказівника даних *DPTR* (*Data Pointer Register*). Регістр *DPTR* використовується при непрямих переходах у програмі або при адресації таблиць, або як один 16-розрядний регістр, або як два незалежних 8-розрядні регістри *DPH* і *DPL*.

Розподіл адресного простору ПЗП показано на рис. 7.3. Молодші адреси ПЗП відводяться під обробку переривань і початок роботи ОМК після скидання.

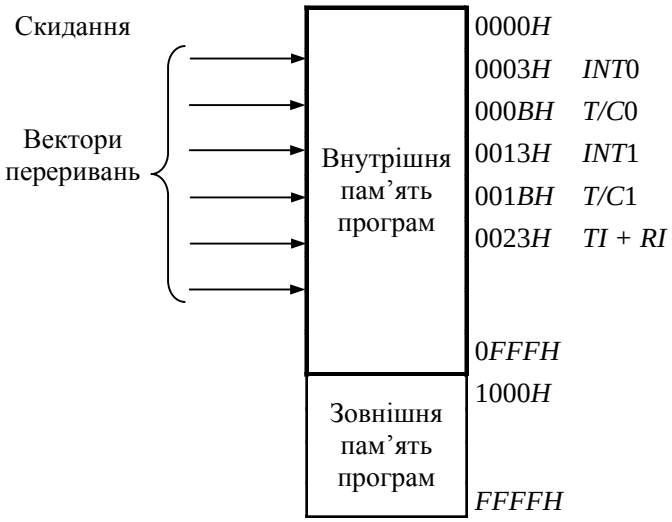


Рис. 7.3. Розподіл адресного простору РПП

Оперативний запам'ятовувальний пристрій або резидентна пам'ять даних (РПД) (рис. 7.4) складається з двох областей.

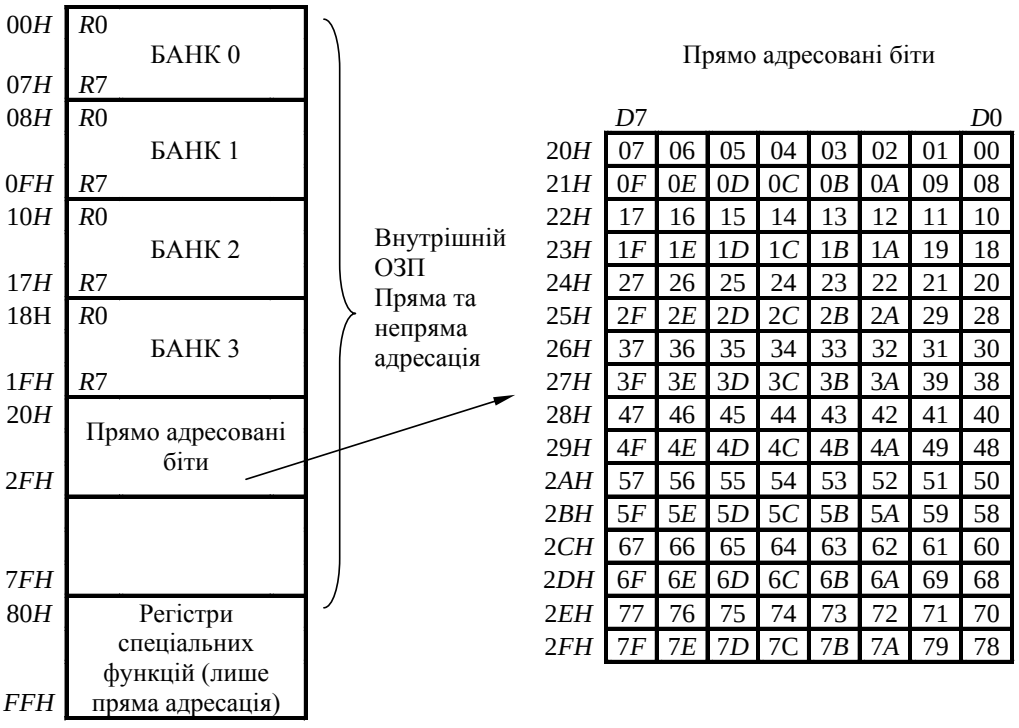


Рис. 7.4. Резидентна пам'ять даних

Перша область – ОЗП даних із інформаційною ємністю 128×8 біт з адресами 0–7*FH*. Друга область – регістри спеціальних функцій *SFR* (*Special Function Registers*) з адресами 80*H–FFH*. Перерахунок регістрів спеціальних функцій наведено в табл. 7.2.

Таблиця 7.2. **Регістри спеціальних функцій**

Позначення	Найменування, адреси бітів	Адреса	Значення після скидання
1	2	3	4
*ACC**	Акумулятор *A* *E*7 *E*6 *E*5 *E*4 *E*3 *E*2 *E*1 *E*0	0*E*0*H*	00
*B**	Акумулятор *B* *F*7 *F*6 *F*5 *F*4 *F*3 *F*2 *F*1 *F*0	0*F*0*H*	00
*PSW**	Слово стану програми *D*7 *D*6 *D*5 *D*4 *D*3 *D*2 *D*1 *D*0 *CY* | *AC* | *F*0 | *RS*1 | *RS*0 | *OV* | – | *P*	0*D*0*H*	00
SP	Регістр-вказівник стека	81*H*	07
DPTR	Регістр-вказівник даних: *DPH* – старший байт *DPL* – молодший байт	 83*H* 82*H*	 00 00
*P*0*	87 86 85 84 83 82 81 80	80*H*	0*FFH*
*P*1*	Порт 1 97 96 95 94 93 92 91 90	90*H*	0*FFH*
*P*2*	Порт 2 *A*7 *A*6 *A*5 *A*4 *A*3 *A*2 *A*1 *A*0 *A*15 | *A*14 | *A*13 | *A*12 | *A*11 | *A*10 | *A*9 | *A*8	0*A*0*H*	0*FFH*
*P*3*	Порт 3 *B*7 *B*6 *B*5 *B*4 *B*3 *B*2 *B*1 *B*0 $\overline{RD}$ | $\overline{WR}$ | *T*1 | *T*0 | $\overline{INT1}$ | $\overline{INT0}$ | *RxD* | *TxD*	0*B*0*H*	0*FFH*
*IP**	Регістр пріоритетів *BF* *BE* *BD* *BC* *BB* *BA* *B*9 *B*8 – | – | *PT*2 | *PS* | *PT*1 | *PX*1 | *PT*0 | *PX*0	0*B*8*H*	*xx*000000*B*
*IE**	Регістр маски переривань *AF* *AE* *AD* *AC* *AB* *AA* *A*9 *A*8 *EA* | – | *ET*2 | *ES* | *ET*1 | *EX*1 | *ET*0 | *EX*0	0*A*8*H*	0*x*000000*B*

Продовження табл. 7.2

1	2	3	4
TMOD	Регістр режиму таймера/лічильника *G*1 | $C/\overline{T1}$ | *M*1.1 | *M*0.1 | *G*0 | $C/\overline{T0}$ | *M*1.0 | *M*0.0	89*H*	00
*TCON**	Регістр керування/статусу таймерів *BF* *BE* *BD* *BC* *BB* *BA* *B*9 *B*8 *TF*1 | *TR*1 | *TF*0 | *TR*0 | *IE*1 | *IT*1 | *IE*0 | *IT*0	88*H*	00
*TH*0	Таймер 0 (старший байт)	8*CH*	00
*TL*0	Таймер 0 (молодший байт)	8*AH*	00
*TH*1	Таймер 1 (старший байт)	8*DH*	00
*TL*1	Таймер 1 (молодший байт)	8*BH*	00
*SCON**	Регістр керування приймачем-передавачем 9*F* 9*E* 9*D* 9*C* 9*B* 9*A* 99 98 *SM*0 | *SM*1 | *SM*2 | *REN* | *TB*8 | *RB*8 | *TI* | *RI*	98*H*	00
SBUF	Буфер приймачів-передавачів	99*H*	*xx*
PCON	Регістр керування потужністю *SMOD* | – | – | – | *GF*1 | *GF*0 | *PD* | *IDL*	87*H*	0*xxxxxxx*

* Позначені регістри припускають адресацію окремих бітів.

Резидентна пам’ять даних адресується 8-розрядним регістром адреси (*PA*) або вказівником стека (*SP*) (див. рис. 7.1). Регістр адреси є програмно недоступним регістром, у який завантажується адреса комірки ОЗП під час виконання команд. Регістр *SP* призначений для адресації стека, який є частиною РПД. Уміст *SP* інкрементується перед запам’ятову-ванням даних у стеку за командами *PUSH* і *CALL* і декрементується за командами *POP* і *RET*. Такий спосіб адресації елементів стека називають *передінкрементним/постдекрементним*. У процесі ініціалізації ОМК після надходження сигналу *RESET* у *SP* автоматично завантажується код 07*H*. Це означає, що якщо програма не перевизначає стек, то перший байт даних у стеку буде розташований у комірці РПД з адресою 08*H*.

Резидентна пам’ять данних, так само як і РПП, може бути розширена до 64 кбайт приєднанням зовнішніх ВІС.

Структурну схему блока керування показано на рис. 7.5.

Код команди, зчитаної з РПП, запам’ятовується у 8-розрядному РК і надходить на ДШК, який входить до складу СКС. ДШК 24-розрядний код, що надходить на ПЛМ, а після цього – на блок логіки керування.

Блок логіки керування на базі декодованого коду команди, зовнішніх керувальних сигналів *RST* (сигналу загального скидання), $\overline{EA}$ (сигналу блокування роботи з РПП) та сигналів від внутрішнього формувача імпульсів синхронізації виробляє внутрішні сигнали керування.

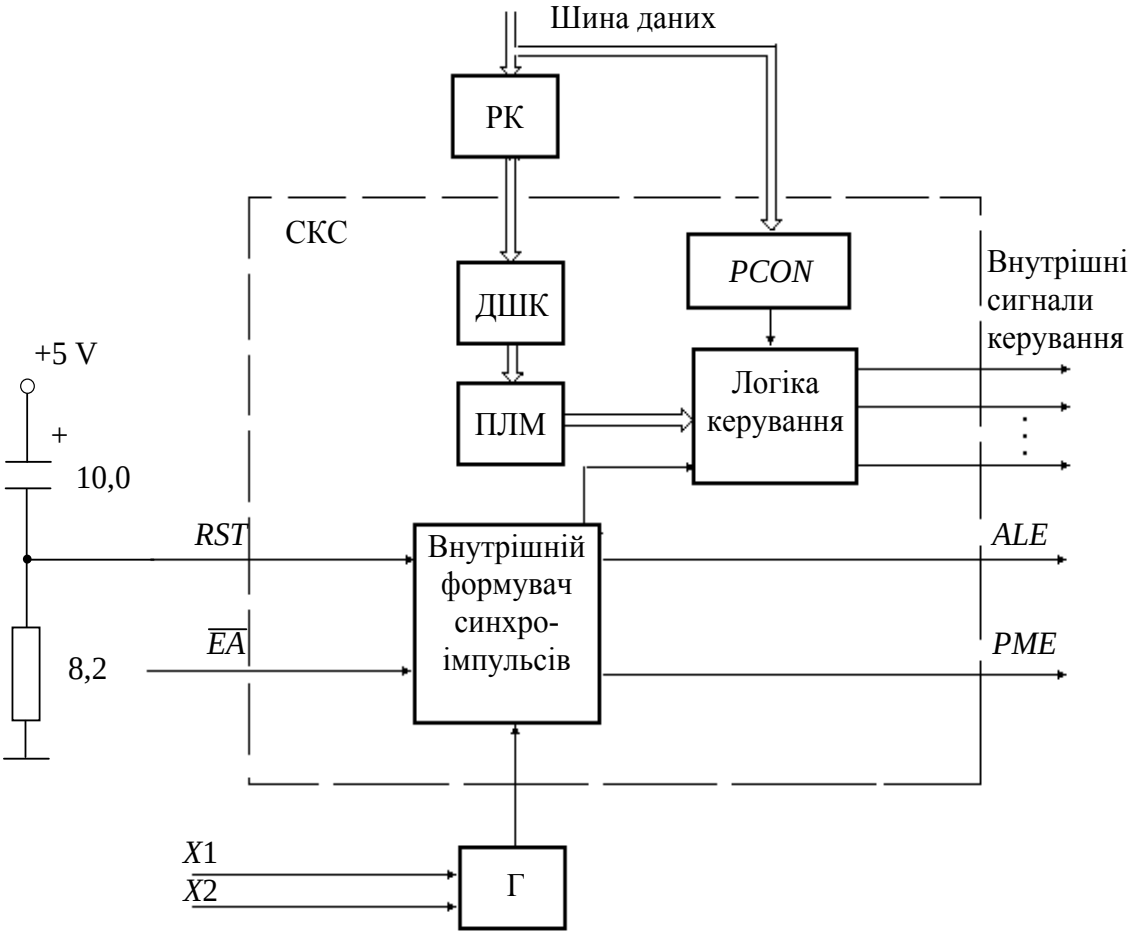


Рис. 7.5. Структурна схема блока керування: Г – генератор тактових імпульсів; РК – регістр команд; ПЛМ – програмовна логічна матриця

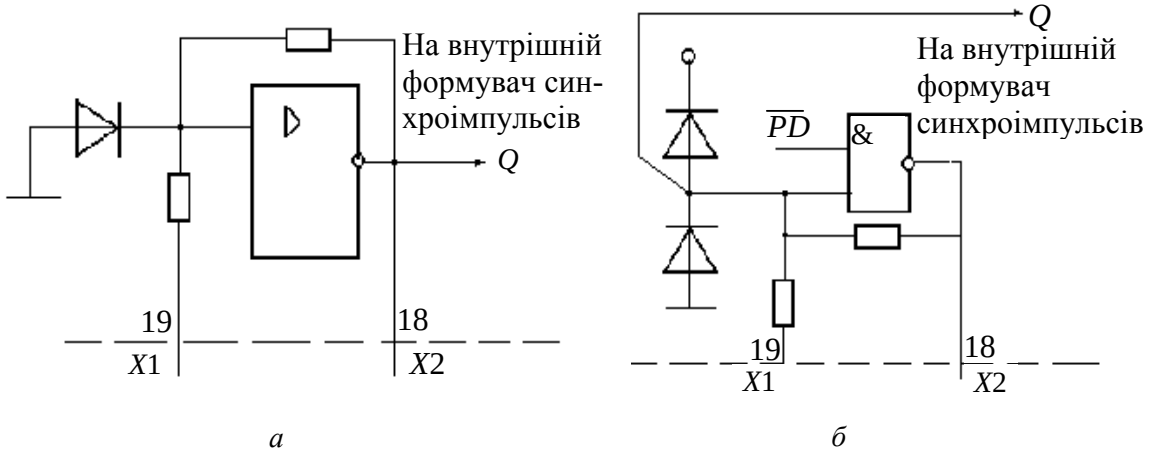
Внутрішній формувач імпульсів синхронізації формує:

1) внутрішні сигнали синхронізації машинних циклів;
2) вихідний сигнал дозволу фіксації адреси *ALE*;
3) сигнал дозволу програмної пам’яті *PME* (формується тільки при роботі із зовнішньою пам’яттю).

Машинний цикл (рис. 7.6) має фіксовану тривалість і містить шість станів – *S*1–*S*6, кожний з яких за тривалістю відповідає одному такту.

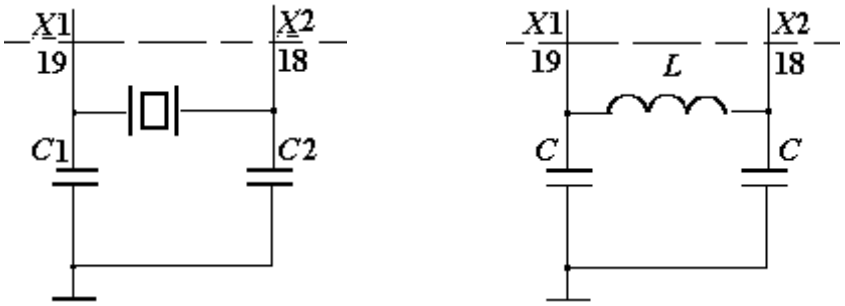
Рис. 7.6. Діаграма формування машинних циклів ОМК

Кожний стан або такт складається з двох фаз – $P1$ і $P2$. Тривалість фази дорівнює періоду сигналу Q, який формується або вбудованим (внутрішнім) тактовим генератором (рис. 7.7) у разі приєднання до виводів 18 ($X2$) та 19 ($X1$) ОМК кварцового резонатора (рис. 7.8, *а*) чи LC-кола (рис. 7.8, *б*), або зовнішнім джерелом тактових сигналів (рис. 7.9).



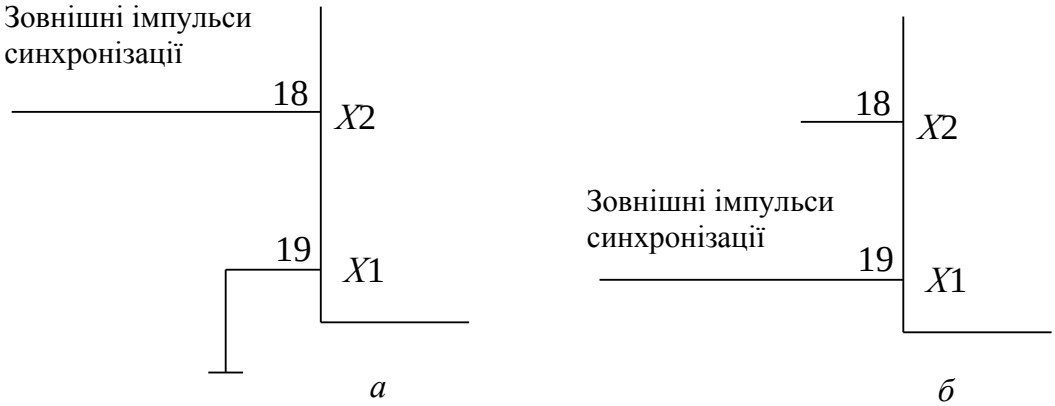
а *б*

Рис. 7.7. Принципові схеми внутрішніх тактових генераторів:
а – *n*-МДН-технологія; *б* – КМДН-технологія



а *б*

Рис. 7.8. Зовнішні кола внутрішнього тактового генератора: *а* – приєднання кварцового резонатора; *б* – приєднання LC-кола; $C1$, $C2$ 30 ± 10 пФ



а *б*

Рис. 7.9. Приєднання зовнішнього джерела тактових сигналів:
а – для *n*-МДН; *б* – для КМДН

Частота імпульсів генератора для схеми на рис. 7.8, а визначається як

$$f = \frac{1}{2\pi\sqrt{LC'}},$$

де $C' = \frac{C + 3C_{pp}}{2}$ ($C_{PP} \approx 10$ пФ – ємність виводу).

Приєднання зовнішнього джерела тактових сигналів до ОМК, виконаних за *n*-МДН- та КМДН-технологіями (див. рис. 7.9), відрізняється тим, що у першому випадку зовнішні імпульси синхронізації надходять на виводи 18 (*X*2) та 19 (*X*1), а у другому випадку зовнішні синхроімпульси надходять на виводи 19 (*X*1) та 20 (спільний), а вивід 18 (*X*2) залишається неприєднаним.

При частоті кварцового резонатора або тактовій частоті зовнішніх імпульсів синхронізації частоті 12 МГц тривалість машинного циклу дорівнює 1 мкс.

Параметри зовнішніх імпульсів синхронізації $f_{\max} = 12$ мГц для частоти показано на рис. 7.10.

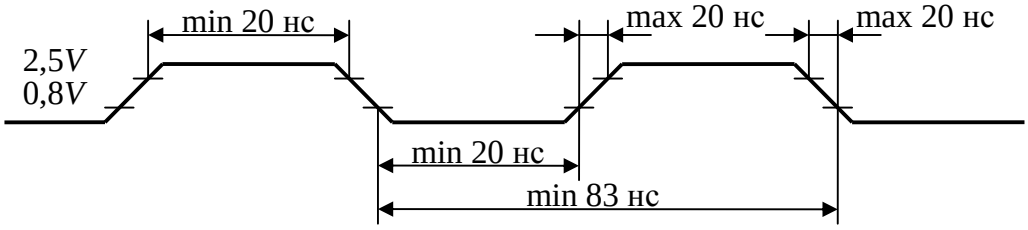


Рис. 7.10. Параметри зовнішніх імпульсів синхронізації

У блок керування входить також регістр керування споживанням *PCON* (*Power CONtrol*).

Порти введення-виведення *P*0–*P*3 (див. рис. 7.1) призначені для забезпечення побайтного обміну інформацією ОМК із зовнішніми пристроями по 32 лініях уведення-виведення. Принципові схеми ліній портів *P*0–*P*3 показано на рис. 7.11. Кожна лінія порту містить керований регістр-фіксатор, два буфери і вихідний транзисторний каскад. Рівні вхідних і вихідних сигналів портів відповідають стандарту ТТЛ-логіки. Будь-яку лінію портів можна використовувати для введення або виведення інформації незалежно від інших ліній.

Для того щоб лінія порту використовувалася для введення, у відповідний *D*-тригер регістра-фіксатора має бути записана логічна одиниця, що закриває МДН-транзистор вихідного каскаду.

Фізичні адреси портів: *P*0–80*H*, при бітовій адресації – 80*H*–87*H*; *P*1–90*H*, при бітовій адресації – 90*H*–97*H*; *P*2–*A*0*H*, при бітовій адресації – *A*0*H*–*A*7*H*; *P*3–*B*0*H*, при бітовій адресації – *B*0*H*–*B*7*H*.

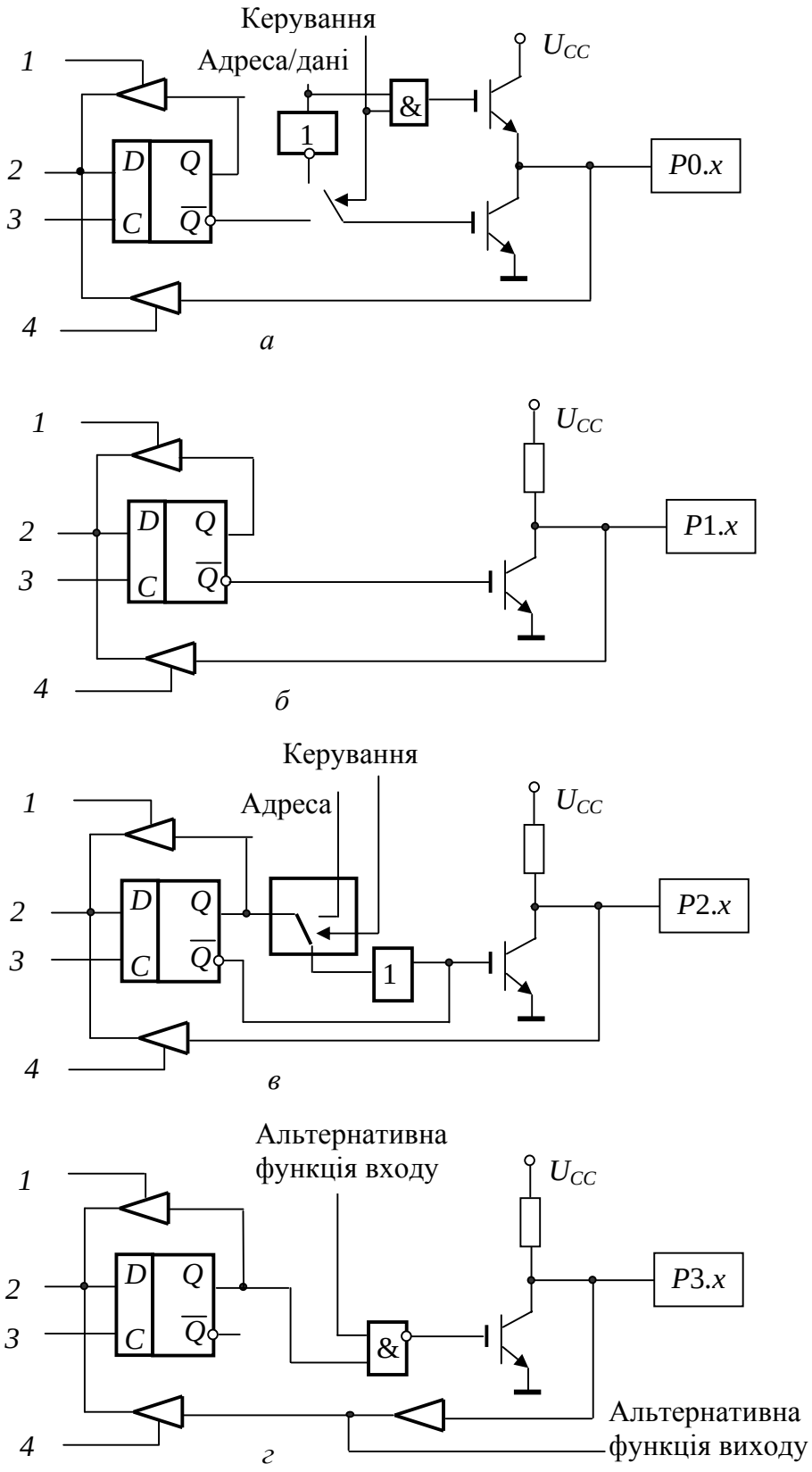


Рис. 7.11. Принципові схеми ліній портів: *а* – *P*0; *б* – *P*1; *в* – *P*3; *г* – *P*3; *1* – читання регістра-фіксатора; *2* – внутрішня шина; *3* – запис у регістр-фіксатор; *4* – читання виводів

Порт *P*0 двонапрямлений – через нього можна в будь-який момент уводити та виводити інформацію. Виводи порта *P*0 мають три стани. Через порт *P*0:

– виводиться молодший байт адреси *A*7–*A*0 при роботі із зовнішнім ПЗП і зовнішнім ОЗП;
– видається з ОМК та приймається у ОМК байт даних при роботі із зовнішньою пам'яттю, при цьому обмін байтом даних і виведення молодшого байта адреси зовнішньої пам'яті мультиплексовано у часі;
– задаються дані при програмуванні внутрішнього ПЗП *EPROM*.

Порти *P*1–*P*3 є квазідвонапрямленими, оскільки в будь-який момент через порти можна тільки виводити інформацію. Для введення інформації необхідно записати в усі розряди регістра-фіксатора логічні одиниці. Після цього можна виконувати введення.

Через порт *P*1 задається молодший байт адреси при програмуванні внутрішнього ПЗП *EPROM* та при читанні внутрішнього ПЗП.

Через порт *P*2:

– виводиться старший байт адреси *A*15–*A*8 при роботі із зовнішнім ПЗП і зовнішнім ОЗП у тих випадках, коли адреса є 16-розрядною;
– задається старший байт *A*15–*A*8 адреси при програмуванні внутрішнього ПЗП *EPROM* та при читанні внутрішнього ПЗП.

Порт *P*3 може використовуватись як для введення-виведення інформації, так і для реалізації альтернативних функцій обміну інформацією (див. рис. 7.11). Альтернативні функції наведено в табл. 7.3. Кожну з восьми ліній порту *P*3 користувач може запрограмувати на виконання альтернативних функцій записом одиниці у відповідні біти регістра-замка (*P*3.0–*P*3.7) порта *P*3.

Таблиця 7.3. **Альтернативні функції порту *P*3**

Біт	Позиція	Альтернативна функція обміну інформацією
RxD	*P*3.0	Вхід приймача послідовного порту в режимах 1–3. Введення-виведення послідовних даних у режимі регістра зсуву
TxD	*P*3.1	Вихід передавача послідовного порту в режимах 1–3. Вихід синхронізації при роботі послідовного порту в режимі 0
$\overline{INT0}$	*P*3.2	Вхід запиту переривання 0. Сприймання сигналів низького рівня або зріз сигналу
$\overline{INT1}$	*P*3.3	Вхід запиту переривання 1. Сприймання сигналів низького рівня або зріз сигналу
*T*0	*P*3.4	Вхід таймера/лічильника 0 або тестовий вхід 0
*T*1	*P*3.5	Вхід таймера/лічильника 1 або тестовий вхід 1
$\overline{WR}$	*P*3.6	Запис. Апаратне формування активного сигналу низького рівня у разі звернення до зовнішньої пам'яті даних
$\overline{RD}$	*P*3.7	Читання. Активний сигнал низького рівня формується апаратно у разі звернення до зовнішньої пам'яті даних

Послідовний порт (див. рис. 7.1) призначений для забезпечення послідовного обміну даними; може використовуватися або як регістр зсуву, або як *універсальний асинхронний приймач-передавач* із фіксованою або змінною швидкістю обміну і з можливістю дуплексного режиму. Послідовний порт може працювати в одному із чотирьох режимів – режим 0, режим 1, режим 2, режим 3). Послідовний порт програмується на один із режимів через запис керувального слова в регістр *SCON* (*Serial port CONtrol*). Призначення бітів регістра *SCON* наведено в табл. 7.4.

Таблиця 7.4. **Призначення бітів регістра *SCON***

Біти	Позначення	Призначення
7	*SM*0	Визначають один із чотирьох режимів роботи послідовного порту: *SM*0 *SM*1 0 0 – режим 0 0 1 – режим 1 1 0 – режим 2 1 1 – режим 3
6	*SM*1	
5	*SM*2	Дозвіл мультипроцесорної роботи: у режимі 0 *SM*2 має бути скинутий у нуль; у режимі 1 при *SM*2 = 1 прапорець *RI* установлюється в одиницю при надходженні стоп-біта, що дорівнює одиниці; у режимах 2 і 3 при *SM*2 = 1 прапорець *RI* дорівнює нулю, якщо дев'ятий біт прийнятих даних дорівнює нулю
4	*REN*	Дозвіл прийняття послідовних даних. Установлюється і скидається програмно відповідно для дозволу та заборони прийняття даних
3	*TB*8	Дев'ятий біт переданих даних у режимах 2 і 3, установлюється і скидається програмно
2	*RB*8	У режимі 0 *RB*8 не використовується; у режимі 1 при *SM*2 = 0 прийнятий стоп-біт; у режимах 2 і 3 – дев'ятий біт прийнятих даних
1	*TI*	Прапорець переривання передавача: у режимі 0 установлюється апаратно при видачі восьмого біта; у інших режимах – установлюється апаратно при формуванні стоп-біта. Скидається програмно в усіх режимах
0	*RI*	Прапорець переривання приймача: при *SM*2 = 0 установлюється апаратно при прийнятті восьмого біта в режимі 0 або через половину інтервала стоп-біта в режимах 1, 2, 3; при *SM*2 = 1 у режимі 1 *RI* не встановлюється, якщо не прийнятий стоп-біт; дорівнює одиниці у режимах 2 та 3; *RI* не встановлюється, якщо дев'ятий прийнятий біт даних дорівнює одиниці. Скидається програмно в усіх режимах

У режимі 0 послідовний порт являє собою 8-розрядний регістр зсуву. Байт інформації передається і приймається через вивід *RxD*, при цьому через вивід *TxD* видаються сигнали синхронізації зсуву. Приймання і видача байта починається з молодшого розряду і закінчується старшим. Швидкість обміну фіксована і дорівнює $f_Q/12$, де f_Q – частота синхронізації ОМК.

На рис. 7.12, *а* зображено функціональну схему послідовного порту в режимі 0, а на рис. 7.12, *б* – відповідні діаграми.

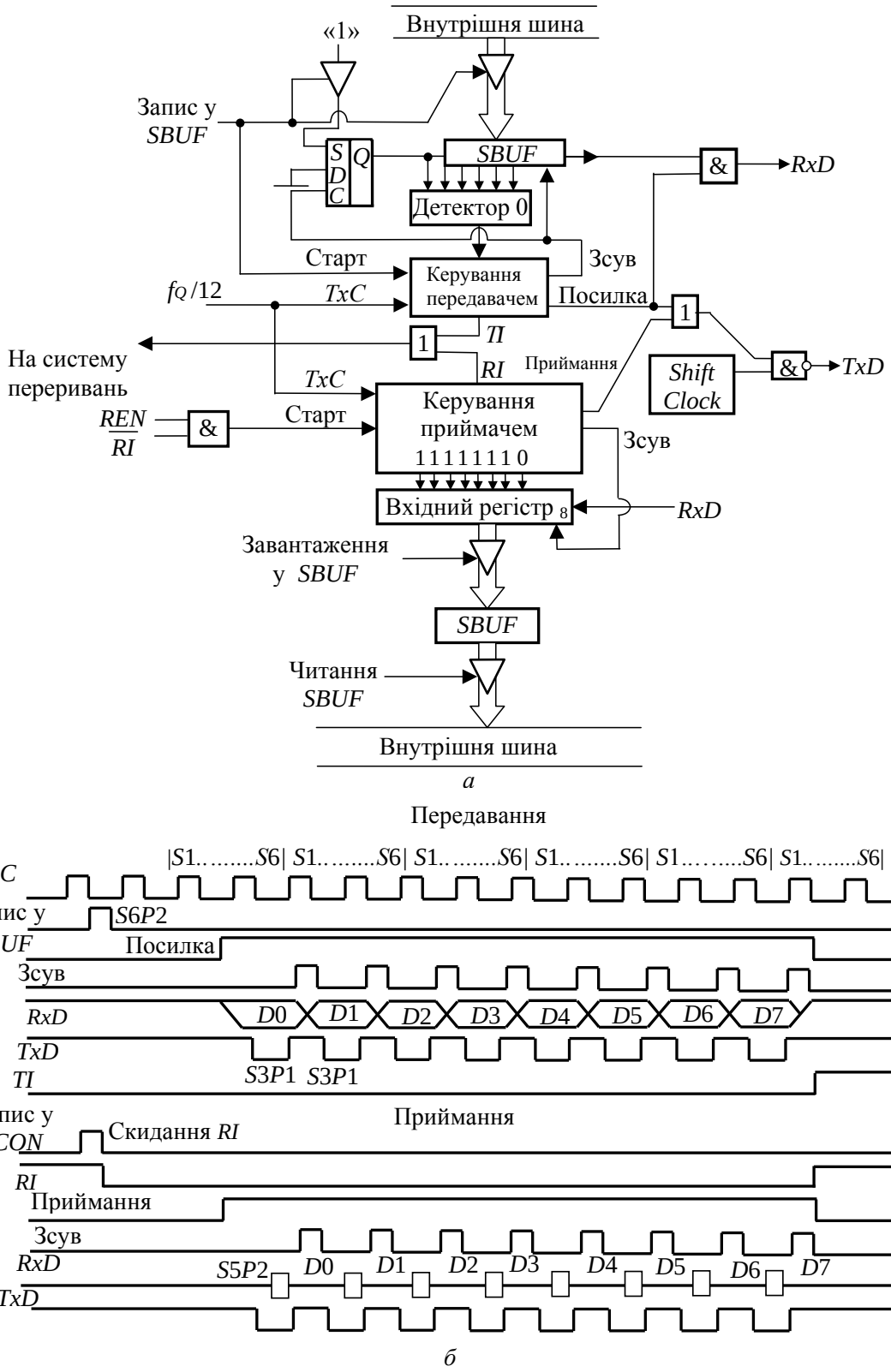


Рис. 7.12. Послідовний порт у режимі 0: *а* – функціональна схема; *б* – діаграми

Передавання починається з будь-якої команди, яка використовує буфер приймача-передавача *SBUF* як регістр призначення, наприклад *MOV SBUF, A.*

У фазі *P*2 стану *S*6 (*S*6*P*2) ПК за сигналом «Запис у *SBUF*» записує байт у регістр зсуву передавача, встановлює *D*-тригер дев'ятого біта в одиницю та ініціює роботу блока керування передавачем. Останній через один машинний цикл формує сигнал дозволу «Посилка», який дозволяє видачу вмісту регістра зсуву передавача на вивід *RxD* та сигналу «Синхронізація зсуву» на вивід *TxD*. За сигналом «Зсув» у момент *S*6*P*2 кожного машинного циклу вміст регістра зсуву передавача зсувається праворуч на одну позицію молодшими бітами вперед і надходить на вивід *RxD*. У старші біти регістра зсуву передавача, що звільняються, записуються нулі.

При отриманні від детектора нуля сигналу «Передавач звільнений» блок керування передавачем знімає сигнал «Посилка» і встановлює прапорець переривання передавача *TI* на початку інтервалу *S*1*P*1 десятого машинного циклу після надходження сигналу «Запис у *SBUF*».

Приймання починається при одночасному виконанні умов *REN* = 1 і *RI* = 0. На початку інтервалу *S*6*P*2 наступного машинного циклу (див. рис. 7.11, *б*) блок керування приймачем формує сигнал дозволу «Приймання», за яким на вихід *TxD* передаються синхросигнали зсуву, і в регістрі зсуву приймача починають формуватися значення бітів даних, які зчитуються з виводу *RxD* в інтервалі *S*5*P*2 кожного машинного циклу. В інтервалі *S*6*P*2 кожного машинного циклу за сигналом «Зсув» здійснюється зсув умісту регістра зсуву приймача ліворуч на одну позицію, і прийнятий біт записується у крайній правий розряд. Після надходження восьмого імпульсу «Зсув» уміст регістра приймача переписується у *SBUF*. В інтервалі *S*1*P*1 десятого машинного циклу блок керування приймачем переписує вміст регістра зсуву в буфер *SBUF*, знімає сигнал «Приймання» та встановлює прапорець переривання приймача *RI* в одиницю.

У режимі 1 послідовний порт являє собою 8-розрядний універсальний асинхронний приймач-передавач зі змінною швидкістю обміну. Через *TxD* передаються, а через *RxD* приймаються 10 біт: нульовий старт-біт, 8 біт інформації та одиничний стоп-біт. Швидкість обміну є змінною. Вона визначається частотою переповнення таймера 1 f_{out1} і бітом *SMOD* регістра *PCON*. На рис. 7.13, показано функціональну схему і діаграми послідовного порту в режимі 1.

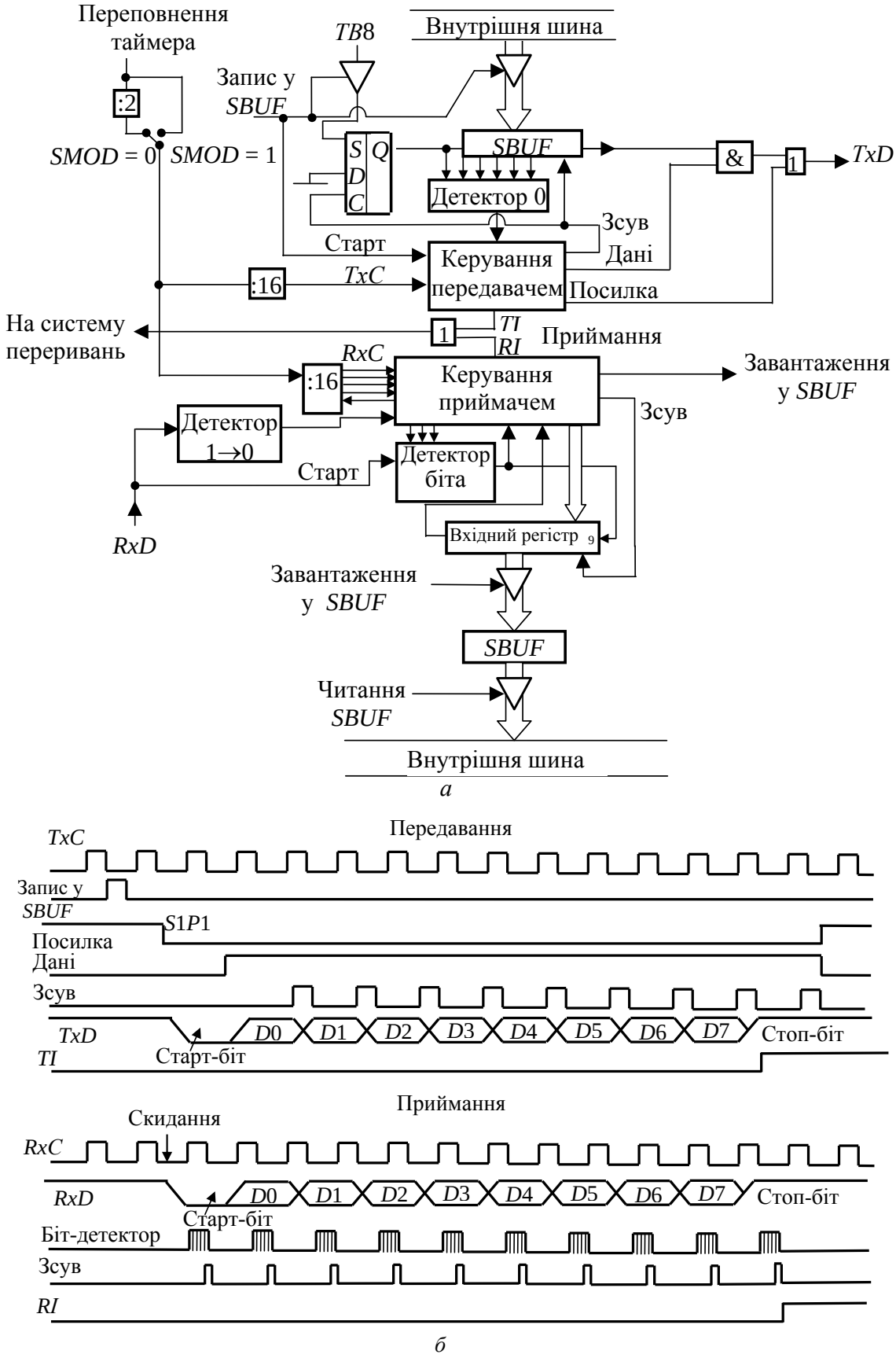


Рис. 7.13. Послідовний порт у режимі 1: *а* – функціональна схема; *б* – діаграми

Передавання починається з будь-якої команди, що використовує *SBUF* як регістр призначення, наприклад:

```
MOV SBUF, #25; переслати в SBUF число 25
```

Пристрій керування ОМК за сигналом «Запис у *SBUF*» завантажує 1 у дев'ятий біт регістра зсуву передавача, ініціює роботу блока керування передавачем і в інтервалі *S*1*P*1 формує сигнал дозволу «Посилка» низького рівня. За цим сигналом на вивід *TxD* спочатку надходить старт-біт, а потім за сигналом дозволу «Дані» та імпульсами «Зсув» уміст регістра передавача зсувається на одну позицію, і на вивід *TxD* по черзі надходять 8 біт даних. За дев'ятим імпульсом «Зсув» формується одиничний стоп-біт, прапорець *TI* встановлюється в нуль, сигнали «Посилка» і «Дані» знімаються, і передавання закінчується.

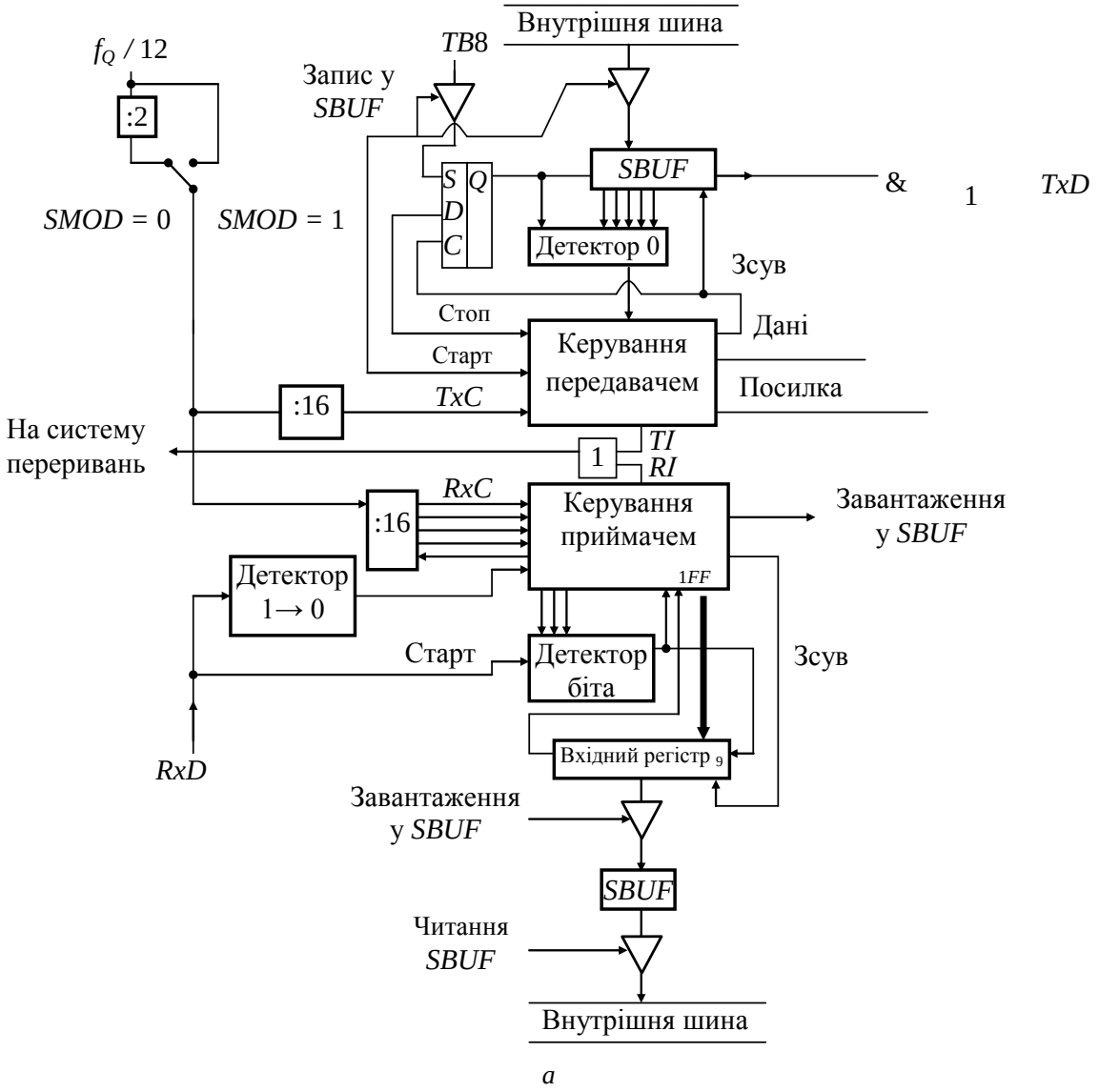
Приймання починається при переході сигналу на вході *RxD* з одиниці в нуль, який виявляється за допомогою детектора спадання. Як тільки перехід з одиниці в нуль виявлений, у регістр зсуву приймача завантажується число 1*FFH*, тобто всі 9 розрядів регістра заповнюються одиницями. При переході сигналу на вході *RxD* з одиниці в нуль також скидається значення внутрішнього лічильника-подільника частоти на 16, який формує сигнал «Синхронізація приймача». Внутрішній лічильник починає відраховувати імпульси синхронізації. Під час сьомого, восьмого та дев'ятого імпульсів здійснюється опитування сигналу на вході *RxD* для підтвердження нульового значення старт-біта. Отримані три значення прийнятого біта надходять на детектор біта, який визначає дійсне значення прийнятого біта за мажоритарним принципом «два з трьох». У тому випадку, якщо дійсне значення старт-біта дорівнює нулю, починається приймання по черзі 8 біт даних. Значення кожного біта даних також перевіряється детектором біта в сьомому, восьмому та дев'ятому імпульсах сигналу «Синхронізація приймача» і лише після цього заноситься в регістр зсуву приймача. Якщо значення старт-біта не дорівнює нулю, то блок керування прийманням знову починає пошук переходів сигналу на вході *RxD* з одиниці в нуль.

Приймання старт-біта та 8 біт даних у кожному машинному циклі супроводжується зсувом умісту регістра приймача на одну позицію за сигналом «Зсув». Після прийняття старт-біта та 8 біт даних приймається стоп-біт, значення якого обов'язково має бути одиничним. Отже, після десятого імпульсу «Зсув» у регістрі приймача знаходяться 8 біт інформації і стоп-біт. Блок керування прийманням формує сигнал «Завантаження буфера», за яким вісім інформаційних бітів надходять у *SBUF*, стоп-біт – у розряд *RB*8 регістра *SCON*. Прапорець переривання приймача *RI* встановлюється

в нуль. Приймання закінчується, і послідовний порт знову починає процес виявлення переходу сигналу на вході *RxD* з одиниці в нуль.

У режимах 2 і *3* послідовний порт являє собою 9-розрядний універсальний синхронний приймач-передавач із фіксованою (для режиму 2) та змінною (для режиму 3) швидкістю обміну. У режимі 2 швидкість обміну дорівнює $f_Q/32$ при *SMOD* = 1 або $f_Q/64$ при *SMOD* = 0. У режимі 3 швидкість обміну визначається таймером 1, як і в режимі 1.

На рис. 7.14, *а* показано функціональну схему, а на рис. 7.14, *б* – діаграми послідовного порту в режимі 2. Функціональна схема послідовного порту в режимі 3 збігається зі схемою на рис. 7.13, *а*, а діаграми – з діаграмами на рис. 7.14, *б*.



а

Рис. 7.14. Послідовний порт у режимі 2: *а* – функціональна схема; *б* – діаграми
(Див. також с. 340.)

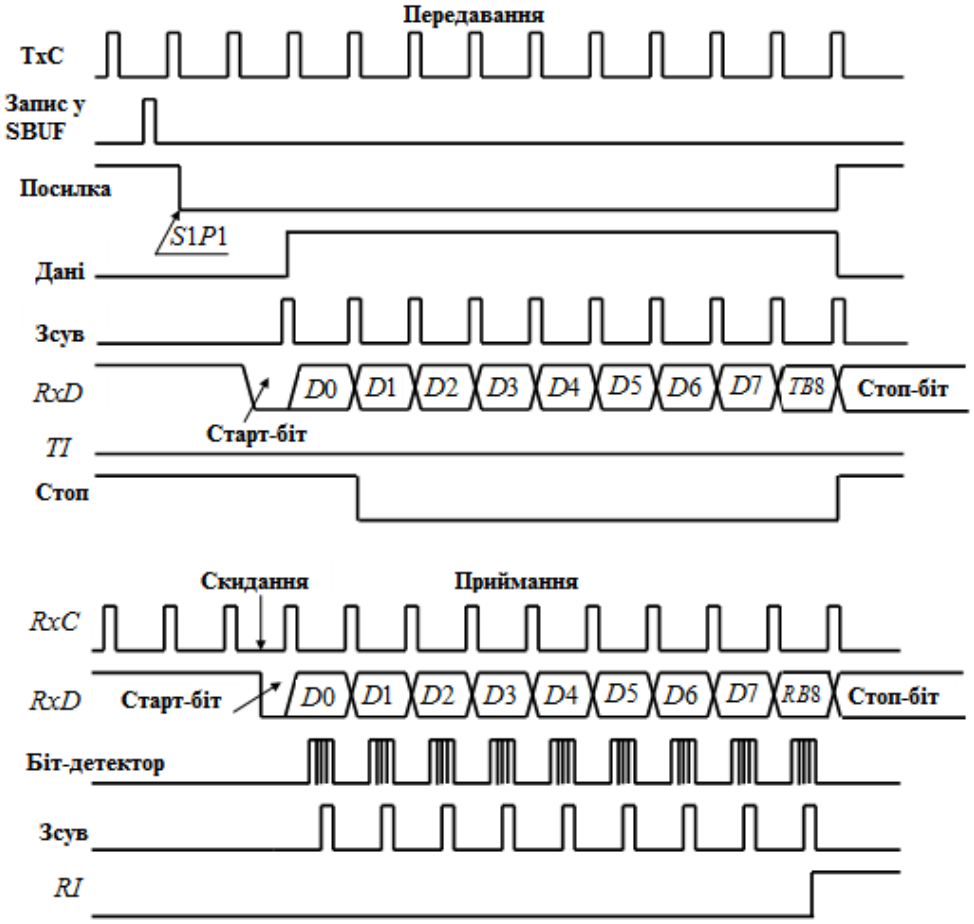


Рис. 7.14. Закінчення

Через вивід *TxD* послідовний порт передає або з виходу *RxD* приймає 11 біт: нульовий старт-біт, 8 біт даних, програмовний дев'ятий біт *TB*8 та одиничний стоп-біт. Режими 2 і 3 відрізняються від режиму 1 лише наявністю дев'ятого програмовного біта. Унаслідок цього змінюються умови закінчення циклу приймання: блок керування прийманням формує сигнал керування «Завантаження буфера», завантажує стоп-біт у розряд *RB*8 регістра *SCON* і встановлює прапорець переривання приймача *RI* в одиницю лише в тому разі, якщо в останньому такті зсуву виконуються дві умови: *RI* = 0 та *SM*2 = 0 або значення дев'ятого прийнятого біта даних дорівнює одиниці.

Значення стоп-біта у режимах 2 і 3 не впливає на *SBUF*, *RB*8 або *RI*.

Швидкість приймання-передавання даних у послідовному порту в різних режимах роботи. У режимі 0 частота передавання залежить лише від резонансної частоти кварцового резонатора $f_0 = f_Q/12$. За один машинний цикл послідовний порт передає 1 біт інформації.

У режимах 1, 2 та 3 швидкість приймання-передавання залежить від значення керувального біта *SMOD* у регістрі *PCON*. У режимі 2 частоту

передавання визначають за формулою $f_2 = (2^{SMOD}/64) f_Q$. Інакше кажучи, при *SMOD* = 0 частота передавання дорівнює $(1/64)f_Q$, а при *SMOD* = 1 – $(1/32)f_Q$.

У режимах 1 і 3 у формуванні частоти передавання, крім біта *SMOD*, бере участь таймер/лічильник 1. При цьому частота передавання залежить від частоти переповнення (*OVT*1) і її визначають таким чином: $f_{1,3} = (2^{SMOD}/32) f_{out}$. Переривання від таймера/лічильника 1 у цьому випадку має бути заблоковане. Сам таймер/лічильник 1 може працювати і як таймер, і як лічильник подій у будь-якому з трьох режимів. Однак найзручніше використовувати режим таймера з автоперевантаженням (старша тетрада *TMOD* = 0010*B*). При цьому частоту передавання визначають так:

$$f_{1,3} = (2^{SMOD}/32)(f_Q/12)/(256 - (TH1)).$$

Блок таймерів/лічильників призначений для підрахунку зовнішніх подій (функція лічильника), реалізації програмнокерованих затримок та виконання функцій задання часу (функція таймера). При виконанні функції таймера вміст *T*/*C* інкрементується в кожному машинному циклі, тобто через кожні 12 періодів резонатора. При виконанні функції лічильника вміст *T*/*C* інкрементується під впливом переходу з одиниці в нуль зовнішнього вхідного сигналу, що надходить на відповідний (*T*0, *T*1) вивід ОМК. Опитування значення зовнішнього вхідного сигналу виконується у фазі *P*2 стану *S*5 кожного машинного циклу. Уміст лічильника збільшується на одиницю, якщо в попередньому циклі надійшов вхідний сигнал високого рівня (1), а в наступному – сигнал низького рівня (0). Нове інкрементоване значення лічильника формується у фазі *P*1 стану *S*3 машинного циклу, що є наступним після того циклу, в якому був зафіксований перехід з одиниці в нуль. Для фіксування переходу необхідні два машинні цикли. Тому максимальна частота підрахунку вхідних імпульсів дорівнює 1/24 частоти резонатора. Для гарантованого зчитування вхідного сигналу він має утримувати своє значення протягом щонайменше одиниці машинного циклу ОМК.

До складу блока таймерів/лічильників входять:

1) два 16-розрядні регістри *T*/*C*0 та *T*/*C*1;
2) 8-розрядний регістр режимів *TMOD*;
3) 8-розрядний регістр керування *TCON*;
4) схема інкремента;
5) схема фіксації сигналів $\overline{INT0}$, $\overline{INT1}$, *T*0, *T*1;
6) схема керування прапорцями;
7) логіка керування *T*/*C*.

*Регістри T/C*0 і *T/C*1 виконують функцію зберігання результатів підрахування. Кожний складається з двох 8-розрядних регістрів – *TH*0, *TL*0 і *TH*1, *TL*1 відповідно (*TH* – старші, *TL* – молодші регістри). Кожний із цих регістрів має свою адресу і може бути використаний як РЗП, якщо відповідний таймер не використовується.

Початковий код лічби заноситься в регістри *T*/*C* програмно. Ознакою закінчення лічби є переповнення регістра *T*/*C*, тобто перехід його вмісту зі стану «всі одиниці» у стан «всі нулі».

Регістр режимів TMOD призначений для приймання та зберігання коду, який визначає:

– один із чотирьох можливих режимів роботи кожного *T*/*C*;
– виконання функцій таймерів або лічильників;
– керування *T*/*C* по зовнішньому виводу.

Призначення бітів регістра *TMOD* наведено в табл. 7.5.

Таблиця 7.5. **Призначення бітів регістра *TMOD***

Біт	Позначення	Призначення
TMOD.3 для *T/C*0 (*TMOD*.7 для *T*/*C*1)	*GATE*0 (*GATE*1)	Дозволяє керувати таймером від зовнішнього виводу ($\overline{INT0}$ – для *T*/*C*0, $\overline{INT1}$ – для *T*/*C*1). *GATE* = 0 – керування заборонено, *GATE* = 1 – керування дозволено
TMOD.2 для *T/C*0 (*TMOD*.6 для *T*/*C*1)	$C/\overline{T}0$ ($C/\overline{T}1$)	Біт вибору функції таймера або лічильника. Якщо біт скинуто, то працює таймер від внутрішнього джерела сигналів синхронізації. Якщо біт встановлено, то працює лічильник від зовнішніх сигналів на вході *T*0 (*T*1)
TMOD.1 для *T/C*0 (*TMOD*.5 для *T*/*C*1)	*M*1.0 (*M*1.1)	Вибір режиму роботи
TMOD.0 для *T/C*0 (*TMOD*.4 для *T*/*C*1)	*M*0.0 (*M*0.1)	

Вибір режиму роботи здійснюється окремо для *T*/*C*0 та *T*/*C*1 згідно зі значеннями бітів *M*1.0 (*M*1.1) та *M*0.0 (*M*0.1) (табл. 7.6).

Таблиця 7.6. **Вибір режиму роботи**

*M*1	*M*0	Режим
0	0	0
0	1	1
1	0	2
1	1	3

Регістр керування TCON призначений для прийняття та зберігання коду керувального слова. Призначення бітів регістра *TCON* наведено в табл. 7.7.

Таблиця 7.7. **Призначення бітів регістра *TCON***

Біт	Позначення	Призначення
7 (5)	*TF*1 (*TF*0)	Прапорці переповнення. Скидаються та встановлюються апаратно і програмно. Доступні для читання
6 (4)	*TR*1 (*TR*0)	Біти вимикання окремо для *T*/*C*0 та *T*/*C*1: *TR* = 0 – вимкнений, *TR* = 1 – увімкнений
3 (1)	*IE*1 (*IE*0)	Прапорці запиту зовнішніх переривань по входу $\overline{INT1}$ ($\overline{INT0}$). Скидаються та встановлюються апаратно і програмно. Доступні для читання
2 (0)	*IT*1 (*IT*0)	Біти, що визначають тип переривання по входу $\overline{INT1}$ ($\overline{INT0}$): *IT* = 0 – переривання по рівню (низькому), *IT* = 1 – переривання по фронту (перехід з одиниці в нуль)

Примітка. Біти 4, 5 належать до *T*/*C*0; біти 6, 7 – до *T*/*C*1. Біти 0, 1 визначають зовнішні переривання по входу $\overline{INT0}$, біти 2, 3 – по входу $\overline{INT1}$.

Прапорці переповнення *TF*0 і *TF*1 встановлюються апаратно при переповненні відповідних *T/C* (перехід умісту регістра *T/C* зі стану «всі одиниці» у стан «всі нулі»). Якщо при цьому переривання від відповідного *T/C* дозволено, то встановлення прапорця *TF* викличе переривання. Прапорці *TF*0 і *TF*1 скидаються апаратно при передаванні керування програмі обробки відповідного переривання. Переривання по *TF*0 та *TF*1 можуть бути викликані (встановлення *TF*) і відмінені (скидання *TF*).

Прапорці *IE*0 та *IE*1 встановлюються апаратно від зовнішніх джерел переривань (відповідно входи ОМК $\overline{INT0}$ та $\overline{INT1}$) або програмно й ініціюють виклик програми обробки відповідного переривання. Скидання цих прапорців виконується апаратно при обслуговуванні переривання лише в тому разі, якщо переривання зумовлене фронтом сигналу. Якщо переривання зумовлене рівнем сигналу на вході $\overline{INT0}$ ($\overline{INT1}$), то скидання прапорця *IE* має виконати програма обслуговування переривання, яка впливає на джерело переривання для зняття ним запиту.

Схема інкремента призначена:

– для збільшення на одиницю в кожному машинному циклі вмісту регістрів *T*/*C*0, *T*/*C*1, для яких установлено режим таймера та дозволено рахування;
– для збільшення на одиницю в циклі вмісту регістрів *T*/*C*0, *T*/*C*1, для яких установлено режим лічильника, дозволено лічбу та на відповідному вході ОМК (*T*0 для *T/C*0 та *T*1 для *T*/*C*1) зафіксовано лічильний імпульс.

Схема фіксації $\overline{INT0}$, $\overline{INT1}$, *T*0, *T*1 являє собою чотири тригери. У фазі *P*2 стану *S*5 кожного машинного циклу в них запам'ятовується інформація, яка надійшла з виводів $\overline{INT0}$, $\overline{INT1}$, *T*0, *T*1.

Схема керування прапорцями встановлює та скидає прапорці переповнення *T*/*C* та прапорці запитів зовнішніх переривань.

Логіка керування синхронізує роботу регістрів *T*/*C*0 та *T*/*C*1 згідно із запрограмованими режимами роботи і синхронізує роботу блока *T*/*C* з роботою ОМК.

Режими роботи *T/C*. Режим роботи кожного *T/C* визначається значеннями бітів *M*0, *M*1 у регістрі *TMOD*. Таймери *T*/*C*0 та *T*/*C*1 мають чотири режими роботи. Режими 0, 1, 2 однакові для обох *T/C*; у цих режимах вони повністю незалежні один від одного. Робота *T*/*C*0 та *T*/*C*1 у режимі 3 різна. При цьому встановлення режиму 3 у *T*/*C*0 впливає на режими роботи *T*/*C*1.

Режим 0 (*M*0 = 0, *M*1 = 0). Таймер у режимі 0 являє собою пристрій на базі 13-розрядного регістра і є 8-розрядним таймером (лічильником) з 5-розрядним переддільником на 32.

Для *T*/*C*0 13-розрядний регістр складається з 8 розрядів регістра *TH*0 та 5 молодших розрядів регістра *TL*0, а для *T*/*C*1 – з 8 розрядів регістра *TH*1 та 5 молодших розрядів регістра *TL*1. Функцію подільника на 32 виконують регістри *TL*0, *TL*1. Вони є програмно доступними, але значущі в них лише 5 молодших розрядів. Функціональну схему *T*/*C*1 у режимі 0 показано на рис. 7.15. Схема *T*/*C*0 є аналогічною.

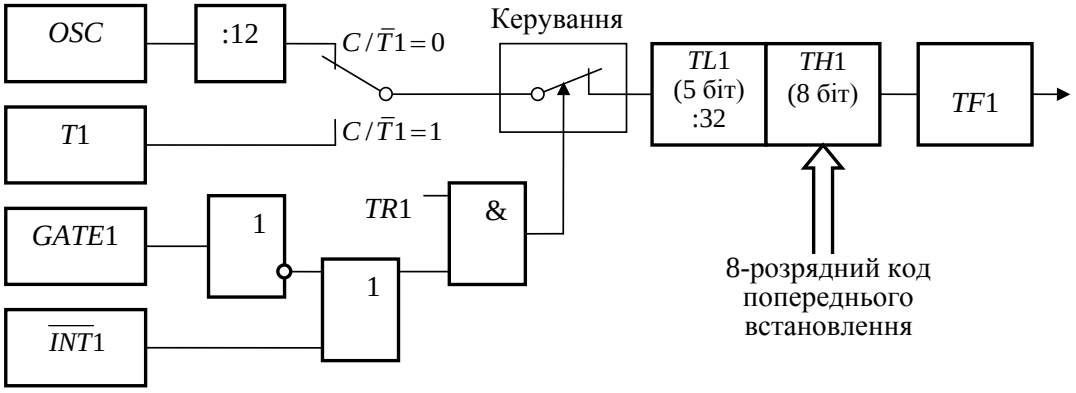


Рис. 7.15. Функціональна схема *T*/*C*1 у режимі 0:
OSC – джерело синхронізації ОМК (внутрішнє або зовнішнє)

На виході *OSC* – сигнал із частотою f_Q. Біт $C/\overline{T}$ регістра *TMOD* визначає виконання функцій таймера $(C/\overline{T}=0)$ або лічильника $(C/\overline{T}=1)$. Лічба починається за командою, яка встановлює біт *TR* регістра *TCON* в одиницю, наприклад, за командою *SETB TR*1. Якщо необхідно керувати лічбою ззовні, то біт *GATE* регістра *TMOD* установлюється в одиницю. Тоді при *TR* = 1 лічбу дозволено, якщо на вході $\overline{INT0}$ (для *T*/*C*0) або $\overline{INT1}$ (для *T*/*C*1) встановлено одиничний стан, і заборонено, якщо встановлено нульовий стан. Установлення біта *TR*0 (для *T*/*C*0) і *TR*1 (для *T*/*C*1) в одиничний стан вимикає відповідний *T/C* незалежно від стану інших бітів.

При переповненні *T/C*, тобто при переході вмісту регістра *T/C* зі стану «всі одиниці» у стан «всі нулі», установлюється прапорець *TF*0 (для *T*/*C*0) і *TF*1 (для *T*/*C*1) у регістрі *TCON*.

Режим 1 (*M*0 = 1, *M*1 = 0). Відмінність від режиму 0 полягає в тому, що встановлення режиму 1 перетворює *T/C* на пристрій із 16-розрядним регістром. Для *T*/*C*0 регістр складається з програмно доступних пар *TL*0, *TH*0, для *T*/*C*1 – з програмно доступних пар *TL*1, *TH*1. Функціональну схему на прикладі *T*/*C*1 показано на рис. 7.16.

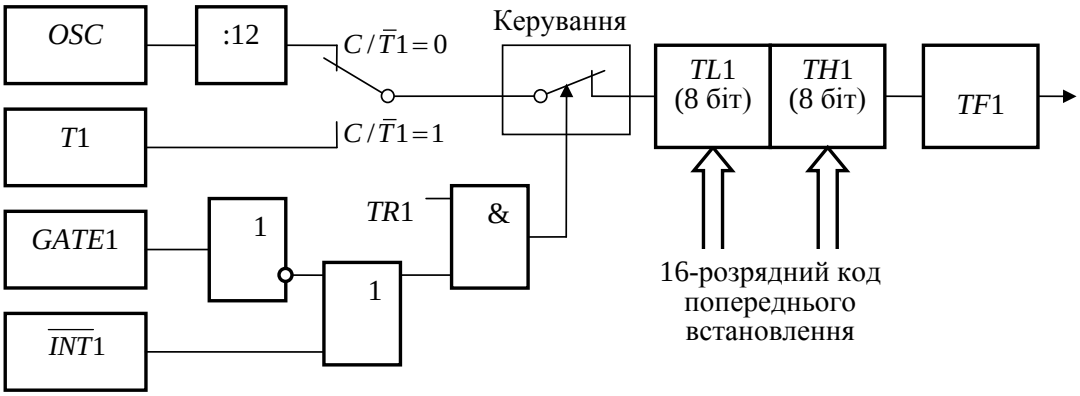


Рис. 7.16. Функціональна схема *T*/*C*1 у режимі 1

Режим 2 (*M*0 = 0, *M*1 = 1). У режимі 2 *T/C* являє собою пристрій на базі 8-розрядного регістра *TL*0 для *T*/*C*0 і *TL*1 для *T*/*C*1. При кожному переповненні *TL*, крім установлення в регістрі *TCON* прапорця *TF*, здійснюється автоматичне перезавантаження вмісту *TH* у *TL*. Регістри *TH*0 і *TH*1 завантажуються програмно. Перезавантаження *TL*0 з *TH*0 та *TL*1 з *TH*1 не впливає на вміст регістрів *TH*0 і *TH*1. Функціональну схему *T*/*C*1 у режимі 2 показано на рис. 7.17.

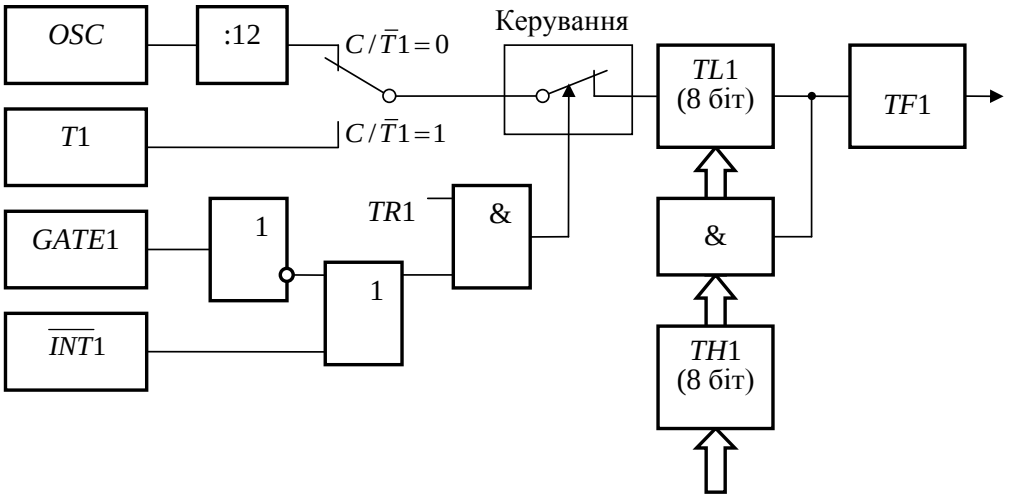


Рис. 7.17. Функціональна схема *T/C*1 у режимі 2

Режим 3. Таймер/лічильник 1 заблокований і зберігає своє значення. Таймер/лічильник 0 у режимі 3 являє собою два незалежні пристрої на базі 8-розрядних регістрів *TL*0 і *TH*0 (рис. 7.18). Пристрій на базі *TL*0

може працювати як у режимі таймера, так і в режимі лічильника, а на базі *TH*0 – тільки в режимі таймера.

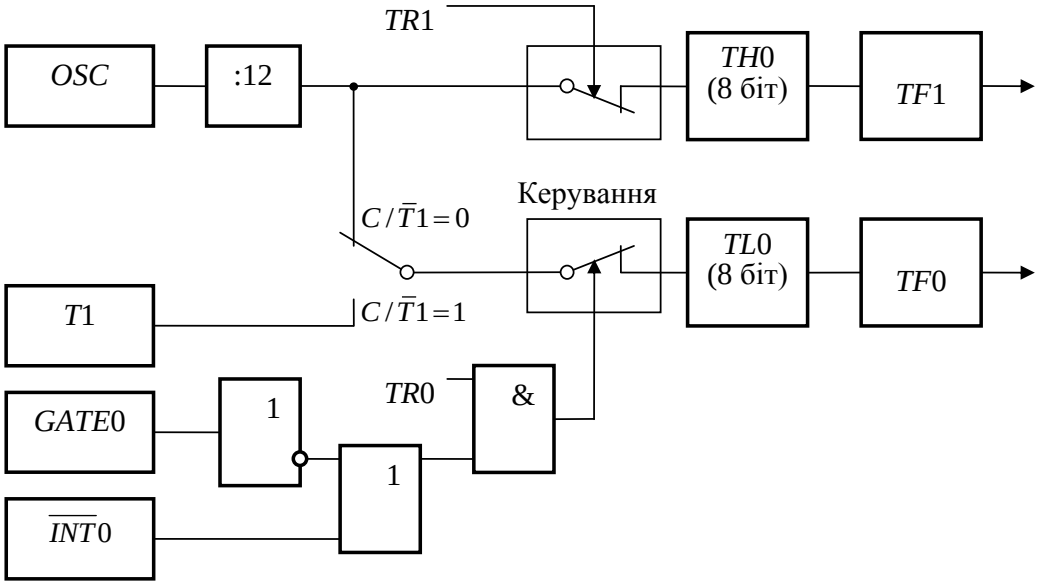


Рис. 7.18. Функціональна схема *T/C*0 у режимі 3

Для забезпечення роботи *T*/*C*0 у режимі 3 використовуються біти *TR*1 і *TF*1, тому вони не можуть використовуватися для керування *T*/*C*1. Це призводить до того, що при встановленні *T*/*C*0 у режим 3, а *T*/*C*1 – у режим 0, 1 або 2, таймер *T*/*C*1 при *GATE1* = 1 завжди ввімкнений. При *GATE*1 = 0 вмикання *T*/*C*1 визначається зовнішніми сигналами аналогічно розглянутому режиму 0. При переповненнях у режимах 0 і 1 *T*/*C*1 обнулюється, а в режимі 2 – перезавантажується без установлення прапорця *TF*1.

Оскільки *T*/*C*1 апаратно пов'язаний з послідовним портом, то під час роботи *T*/*C*0 у режимі 3 можна використовувати *T*/*C*1 у режимі 2 для задання швидкості роботи послідовного порту або для інших завдань, що не потребують переривання.

Система переривань (рис. 7.19) призначена для реагування на зовнішні та внутрішні події. До зовнішніх подій належать появи нульового потенціалу (або зрізу) на виводах $\overline{INT0}$, $\overline{INT1}$, до внутрішніх – переповнення таймерів/лічильників, завершення послідовного обміну. Зовнішні або внутрішні події викликають встановлення відповідних прапорців: *IE*0, *IE*1, *TF*0, *TF*1, *RI* і *TI*, що й спричиняють переривання. Відзначимо, що всі ці прапорці можуть бути програмно встановлені або скинуті, при цьому їх програмне встановлення викликає переривання так само, як і реагування на подію. Отже, переривання можуть програмно викликатися або програмно усуватися. Крім того, переривання на входах $\overline{INT0}$, $\overline{INT1}$ можуть викликатися програмним скиданням бітів *P*3.2 і *P*3.3. Керування системою переривання

здійснюється за допомогою запису керувальних слів у регістри *TCON* (див. табл. 7.7), *IE* і *IP*. Регістр дозволу переривань *IE* призначений для дозволу або заборони переривань від відповідних джерел. Регістр пріоритетів переривань *IP* призначений для встановлення рівня пріоритету переривання для кожного з п'яти джерел переривань.

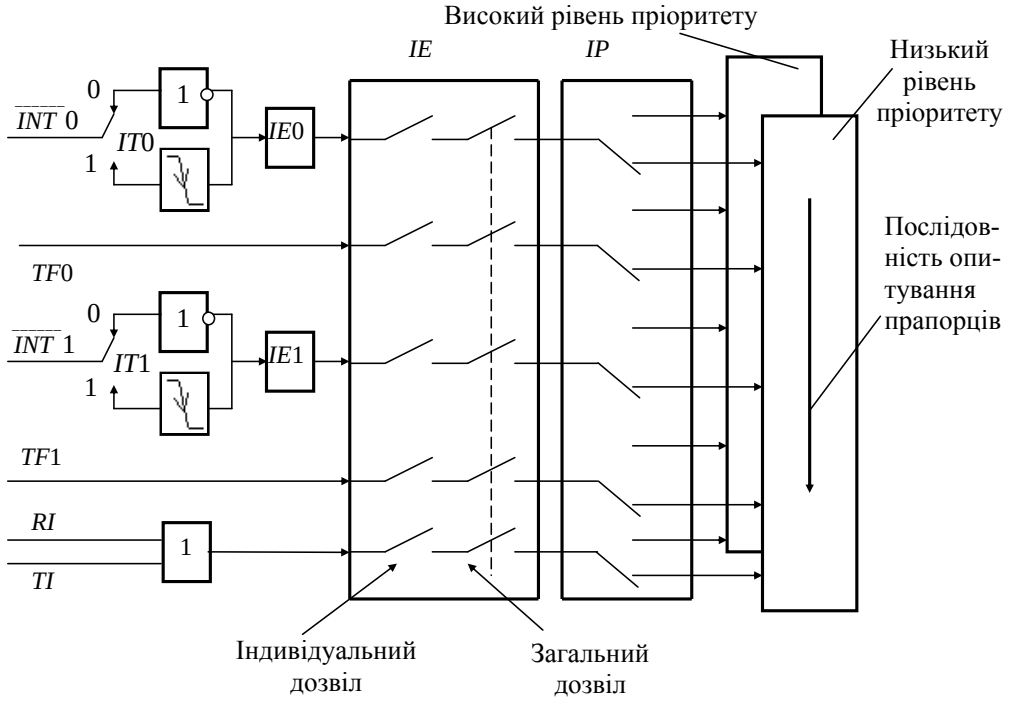


Рис. 7.19. Функціональна схема системи переривань

Призначення бітів регістрів *IE* та *IP* наведено в табл. 7.8 та 7.9 відповідно.

Таблиця 7.8. **Призначення бітів регістра *IE***

Біт	Позначення	Призначення
IE.7	*EA*	Зняття блокування переривань. Скидається програмно для заборони всіх переривань незалежно від стану *IE*4–*IE*0
IE.6 *IE*.5	–	Не використовуються
IE.4	*ES*	Біт дозволу переривання від універсального синхронного приймача-передавача. Установлення/скидання програмою для дозволу/заборони переривань від прапорців *TI* або *RI*
IE.3	*ET*1	Біт дозволу переривання від таймера 1. Установлення/скидання програмою для дозволу/заборони переривань від таймера 1
IE.2	*EX*1	Біт дозволу зовнішнього переривання 1. Установлення/скидання програмою для дозволу/заборони переривань
IE.1	*ET*0	Біт дозволу переривання від таймера 0. Працює аналогічно *IE*.3
IE.0	*EX*0	Біт дозволу зовнішнього переривання 0. Працює аналогічно *IE*.2

Таблиця 7.9. **Призначення бітів регістра *IP***

Біт	Позначення	Призначення
IP.7–*IP*.5	–	Не використовуються
IP.4	*PS*	Біт пріоритету універсального асинхронного приймача-передавача. Установлення/скидання програмою для присвоєння перериванню від універсального асинхронного приймача-передавача вищого/нижчого пріоритету
IP.3	*PT*1	Біт пріоритету таймера 1. Установлення/скидання програмою для присвоєння перериванню від таймера 1 вищого/нижчого пріоритету
IP.2	*PX*1	Біт пріоритету зовнішнього переривання 1. Установлення/скидання програмою для присвоєння вищого/нижчого пріоритету перериванню $\overline{INT1}$
IP.1	*PT*0	Біт пріоритету таймера 0. Працює аналогічно *IP*.3
IH.0	*PX*0	Біт пріоритету зовнішнього переривання 0. Працює аналогічно *IP*.2

Зовнішні переривання сприймаються або за переходом сигналу на входах $\overline{INT0}$ та $\overline{INT1}$ з *H*-рівня у *L*-рівень, або за нульовим рівнем сигналу залежно від стану бітів *IT*0, *IT*1 регістра *TCON* (див. табл. 7.7). При перериванні за нульовим рівнем цей рівень має утримуватися не менше, ніж 12 періодів сигналу тактової частоти *CLK*. При надходженні одного із сигналів $\overline{INT0}$ або $\overline{INT1}$ встановлюється прапорець *IE*0 або *IE*1 у регістрі *TCON*, що викликає відповідне переривання.

Скидання прапорців *IE*0 або *IE*1 здійснюється апаратно лише в тому разі, якщо переривання здійснюється за переходом з одиниці в нуль сигналу.

Якщо переривання зумовлене нульовим рівнем сигналу, то скиданням прапорців *IE*0 або *IE*1 керує відповідна підпрограма обслуговування переривання через вплив на джерело переривання з метою зняття ним запиту.

Переривання від таймерів/лічильників викликаються одиничними значеннями прапорців *TF*0 або *TF*1 у регістрі *TCON*. Прапорці *TF*0 та *TF*1 встановлюються при переповненні відповідних таймерів. Скидання прапорців *TF*0 та *TF*1 виконується автоматично при переході до підпрограм обробки переривань.

Переривання від послідовного порту викликаються встановленням прапорців *TI* або *RI* у регістрі *SCON*. Скидання прапорців *TI* або *RI* здебільшого здійснюється у підпрограмі обробки переривання.

Кожний з описаних типів переривань може бути дозволений або заборонений за допомогою встановлення/скидання відповідного біта в регістрі *IE* (див. табл. 7.8). Скиданням біта *EA* можна одночасно заборонити всі переривання.

До складу системи переривань входять також логіка обробки прапорців переривань і схема формування вектора переривання. Логіка обробки прапорців переривань здійснює пріоритетний вибір запиту переривання,

скидання відповідного прапорця та ініціює апаратну реалізацію команди переходу на підпрограму обслуговування переривання. Кожному з джерел переривань за допомогою встановлення/скидання відповідного біта в регістрі *IP* (див. табл. 7.9) присвоюють один з двох рівнів пріоритету – високий або низький. Програма обробки переривання може перериватися іншим запитом переривання того самого рівня пріоритету. Програма обробки, яка має низький рівень переривань, може бути перервана запитом переривання з високим рівнем. При одночасному надходженні запитів з різними рівнями спочатку обслуговується запит з високим рівнем пріоритету. При одночасному надходженні запитів з однаковими рівнями обробка їх здійснюється в порядку послідовності внутрішнього опитування прапорців (напрям вказано стрілкою на рис. 7.19).

Схема формування вектора переривання формує двобайтові адреси підпрограм обслуговування переривання залежно від джерела переривання (табл. 7.10).

Таблиця 7.10. **Вектори переривання**

Джерело переривання	Вектор переривання
Зовнішнє переривання $\overline{INT0}$	0003*H*
Таймер/лічильник *T*/*C*0	000*BH*
Зовнішнє переривання $\overline{INT1}$	0013*H*
Таймер/лічильник *T*/*C*1	001*BH*
Послідовний порт	0023*H*

Режими енергоспоживання ОМК. В ОМК, виконаних за *n*-МДН-технологією, регістр *PCON* містить лише 1 біт *SMOD*, що керує швидкістю передачі послідовного порту. Тому для них існує лише *режим зниженого енергоспоживання*, який забезпечує живлення внутрішнього ОЗП, якщо значення сигналу на виводі $\overline{RST}$ більше, ніж на виводі U_{CC}. Це реалізується за допомогою двох діодів, з катодів яких здійснюється живлення ОЗП, а їх аноди з'єднані з виводами *RST* та U_{CC} (рис. 7.20).

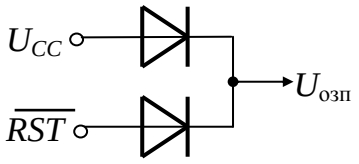


Рис. 7.20. Забезпечення живлення ОЗП у режимі зниженого споживання

В ОМК, виконаних за КМДН-технологією, є два режими зниженого енергоспоживання: *режим холостого ходу* та *режим мікроспоживання*. Джерелом живлення в цих режимах є вивід U_{CC}. Вибір і керування режимами здійснюється за допомогою регістра керування споживанням *PCON*. Адресація окремих бітів у регістрі *PCON* не припустима. Формат регістра *PCON* наведено в табл. 7.11.

Таблиця 7.11*. Позначення бітів регістра *PCON

Біт	Позначення	Призначення	Примітки
7	*SMOD* (*Serial MODe*)	Біт подвоєння швидкості передачі	Керує роботою послідовного порту. При *SMOD* = 1 швидкість передачі подвоюється в режимах 1, 2, 3 послідовного порту
6	–	Резервний	–
5			
4			
3	*GF*1(*General purpose Flag bit*)	Прапорець загального призначення	–
2	*GF2*		
1	*PD* (*Power Down bit*)	Біт вмикання режиму мікроспоживання	Тільки для КМДН-технологій. Якщо *PD* і *IDL* одночасно дорівнюють 1, то перевагу має *PD*, тобто автоматично обирається режим мікроспоживання
0	*IDL* (*IDLe mode bit*)	Біт установлення режиму холостого ходу	

Режими зниженого енергоспоживання ініціюються встановленням бітів *PD* та *IDL*. Вплив значення цих бітів на формування тактових сигналів блоків ОМК показано на рис. 7.21.

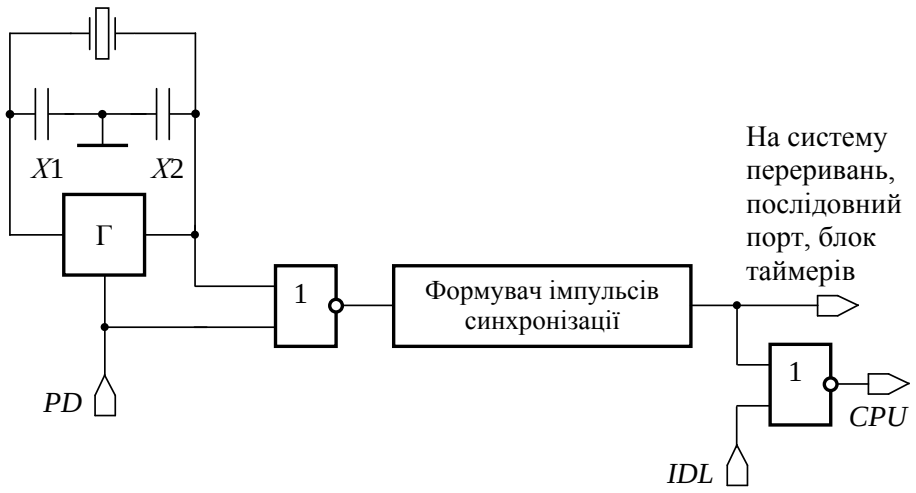


Рис. 7.21. Вплив бітів *PD* і *IDL* на формування сигналів ОМК

Режим холостого ходу задається командою, яка встановлює біт *IDL* в одиницю, наприклад, *MOV PCON*, #01. У цьому режимі блокуються функціональні вузли блока ЦП (*CPU*). Внутрішній генератор сигналів синхронізації продовжує роботу. Усі регістри, вказівник стека, програмний лічильник, *PSW*, акумулятор та внутрішній ОЗП зберігають свої значення. На виводах усіх портів утримується той логічний стан, який був на них у момент переходу в режим холостого ходу. На виводах *ALE* та $\overline{PME}$ формується рівень логічної одиниці.

Існує два способи виходу з режиму холостого ходу – за перериванням або за сигналом апаратного скидання по входу *RST*. Активізація будь-якого дозволеного переривання автоматично веде до встановлення *IDL* у нуль, тобто до припинення режиму холостого ходу. Після виконання команди *RETI* (вихід з підпрограми обслуговування переривання) буде виконано команду, наступну за командою, яка перевела ОМК у режим холостого ходу.

Закінчення режиму холостого ходу відбувається також з появою сигналу апаратного скидання на виводі $\overline{RST}$ тривалістю не менше двох машинних циклів. Активний сигнал на виводі $\overline{RST}$ асинхронно скидає біт *IDL* у логічний нуль. Оскільки тактовий генератор працює, ОМК відразу після скидання *IDL* у логічний нуль починає виконувати програму з команди, наступної після команди, яка викликала режим холостого ходу. Тривалість інтервалу між скиданням біта *IDL* та моментом, коли вмикається внутрішній алгоритм скидання, може становити до двох машинних циклів. Протягом цього інтервалу блокується доступ до внутрішнього ОЗП, але не блокується доступ до портів. Тому небажано використовувати команди звернення до портів безпосередньо після команди встановлення біта *IDL*.

Біти *GF*0 та *GF*1 (див. табл. 7.11) зручно використовувати для визначення того режиму, у якому відбувся виклик програми обробки переривання: нормального режиму або режиму холостого ходу. Наприклад, команда, яка викликає режим холостого ходу, може також установлювати один або декілька прапорців (*GF*0, *GF*1 або інші). Програма обробки переривання, перевіряючи ці прапорці, може визначати режим, у якому це переривання сталося.

Режим мікроспоживання ініціюється встановленням біта *PD* в одиничний стан, наприклад, за командою *MOV PCON*, #02. У цьому режимі генератор вимикається, припиняючи роботу всіх вузлів ОМК, зберігається тільки вміст ОЗП та регістрів спеціальних функцій. На виводах портів утримуються значення, які відповідають умісту їх буферних регістрів. Виходи сигналів *ALE* та $\overline{PME}$ скидаються. Електроживлення відбувається через вивід $\overline{RST}/U_{PD}$. У цьому режимі напруга U_{CC} може бути зменшена до 2 В і має відновитися до номінального значення перед виходом з режиму, здійснити який можна лише за сигналом апаратного скидання на виводі $\overline{RST}$ тривалістю не менше 10 мс (час відновлення роботи генератора імпульсів синхронізації). При одночасному значенні *IDL* = 1 та *PD* = 1 перевагу має біт *PD*.

Контрольні запитання

1. Назвіть основні складові блоки ОМК.
2. Поясніть структуру та принцип функціонування блока керування ОМК.

3. Як здійснюється синхронізація роботи основних блоків ОМК?
4. Назвіть призначення та характеристики РПП.
5. Назвіть призначення та характеристики РПД.
6. Назвіть призначення та режими роботи таймерів.
7. Назвіть призначення та режими роботи портів ОМК.
8. Назвіть призначення та режими роботи послідовного порту ОМК.
9. Назвіть призначення системи переривань і джерела переривань в ОМК.
10. Поясніть принцип пріоритетної обробки запитів переривань.
11. Які існують режими роботи послідовного порту?

7.2. Система команд

Система команд ОМК К1816ВЕ51 містить 111 команд, які можна поділити на п’ять груп:

1) команди пересилання;
2) арифметичні команди;
3) логічні команди;
4) команди операцій з бітами;
5) команди переходів.

Більшість команд (94) мають формат 1–2 байт і виконуються за 1–2 цикли (при тактовій частоті 12 МГц тривалість циклу дорівнює 1 мкс).

В ОМК використовуються такі типи адресації:

– пряма;
– безпосередня;
– непряма.

Систему команд ОМК К1816ВЕ51 наведено в табл. 7.12, де використано такі позначення:

R – один з регістрів *R*0–*R*7;

r – регістр *R*0 або *R*1;

ad – байт з РПД за прямою адресою (у команді вказується або адреса байта – 00–*FF*, або позначення одного з регістрів спеціальних функцій *SFR*, наприклад, *TH*0, *P*1, *TCON*, *SBUF*);

#data – безпосередні 8-розрядні дані (перед числовими значеннями ставлять знак #, наприклад, #34);

*d*16 – безпосередні 16-розрядні дані;

add – байт-приймач з прямою адресацією;

ads – байт-джерело з прямою адресацією;

*ad*16 – 16-розрядна адреса;

*ad*11 – 11-розрядна адреса;

*ad*8 – 8-розрядна адреса-зміщення;

M_i(..) – комірка пам’яті (у дужках вказано адресу; індекс вказує тип пам’яті: РПД – резидентна пам’ять даних, ЗПД – зовнішня пам’ять даних, ПП – пам’ять програм);

bit – біт з прямою адресацією (у команді вказується або адреса біта – 00–*FF*, або розташування його в регістрі спеціальних функцій, наприклад: *ACC*.0 – нульовий біт акумулятора; *P*1.2 – другий біт порту 1 або позначення біта в регістрі спеціальних функцій (наприклад, *EA* – біт загального дозволу переривань).

Таблиця 7.12. **Система команд ОМК К1816ВЕ51**

Мнемокод	Опис	Алгоритм	Байт	Цикл
1	2	3	4	5
КОМАНДИ ПЕРЕСИЛАННЯ				
MOV A, R	Пересилання даних з регістра в акумулятор	A ← R	1	1
MOV A, ad	Пересилання в акумулятор байта з прямою адресацією	$A \leftarrow ad$	2	1
MOV A, @r	Пересилання байта з РПД в акумулятор	$A \leftarrow M_{\text{РПД}}(r)$	1	1
MOV A,#data	Пересилання безпосереднього операнда в *A*	$A \leftarrow$ #*data*	2	1
MOV R, A	Пересилання акумулятора в регістр	$R \leftarrow A$	1	1
MOV R, ad	Пересилання в регістр байта з прямою адресацією	$R \leftarrow ad$	2	2
MOV R, #data	Пересилання безпосереднього операнда в регістр	$R \leftarrow$ #*data*	2	1
MOV ad, A	Пересилання акумулятора за прямою адресою	$ad \leftarrow A$	2	1
MOV ad, R	Пересилання регістра за прямою адресою	$ad \leftarrow R$	2	2
MOV add, ads	Пересилання байта за прямою адресою	$add \leftarrow ads$	3	2
MOV ad, @r	Пересилання байта з РПД за прямою адресою	$ad \leftarrow M_{\text{РПД}}(r)$	2	2
MOV ad, #data	Пересилання безпосереднього операнда за прямою адресою	$ad \leftarrow$ #*data*	3	2
MOV @r, A	Пересилання акумулятора в РПД	$M_{\text{РПД}}(r) \leftarrow A$	1	1
MOV @r, ad	Пересилання в РПД операнда за прямою адресою	$M_{\text{РПД}}(r) \leftarrow ad$	2	2
MOV @r, #data	Пересилання безпосереднього операнда в РПД	$M_{\text{РПД}}(r) \leftarrow$ #*data*	2	1
MOV DPTR, #d16	Завантаження вказівника даних	$DPTR \leftarrow$ #*d*16	3	2
MOVC A, @A+DPTR	Пересилання в акумулятор байта з пам’яті програм	$A \leftarrow M_{\text{ПП}}(A + DPTR)$	1	2
MOVX A, @r	Пересилання байтів із ЗПД в акумулятор	$A \leftarrow M_{\text{ЗПД}}(r)$	1	2
MOVX A, @DPTR	Пересилання байта з розширеної ЗПД в акумулятор	$A \leftarrow M_{\text{ЗПД}}(DPTR)$	1	2

Продовження табл. 7.12

1	2	3	4	5
MOVX @r, A	Пересилання акумулятора в ЗПД	$M_{ЗПД}(r) \leftarrow A$	1	2
MOVX @DPTR, A	Пересилання байта в розширену ЗПД з акумулятора	$M_{ЗПД}(DPTR) \leftarrow A$	1	2
PUSH ad	Завантаження в стек	$SP \leftarrow SP + 1$, $M_{РПД}(SP) \leftarrow ad$	2	2
POP ad	Витягування зі стека	$ad \leftarrow M_{РПД}(SP)$, $SP \leftarrow SP - 1$	2	2
XCH A, R	Обмін акумулятора та регістра	$A \leftarrow\rightarrow R$	1	1
XCH A, ad	Обмін акумулятора та байта з прямою адресацією	$A \leftarrow\rightarrow ad$	1	1
XCH A, @r	Обмін акумулятора та комірки РПД	$A \leftarrow\rightarrow M_{РПД}(r)$	1	1
XCHD A, @r	Обмін молодших тетрад акумулятора та комірки РПД	$A_{0–3} \leftarrow\rightarrow M_{РПД}(r)_{0–3}$	1	1
SWAP A	Обмін тетрад в акумуляторі	$A_{0–3} \leftarrow\rightarrow A_{4–7}$	1	1
АРИФМЕТИЧНІ КОМАНДИ				
ADD A, R	Додавання регістра *R* і акумулятора	$A \leftarrow A + R$	1	1
ADD A, ad	Додавання акумулятора та байта з прямою адресацією	$A \leftarrow A + ad$	2	1
ADD A, @r	Додавання байта з РПД і акумулятора	$A \leftarrow A + M_{РПД}(r)$	1	1
ADD A, #data	Додавання константи з акумулятором	$A \leftarrow A + \#data$	2	1
ADDC A, R	Додавання *R* з акумулятором і прапорця перенесення *C*	$A \leftarrow A + R + C$	1	1
ADDC A, ad	Додавання акумулятора і прямоадресованого байта з прапорцем *C*	$A \leftarrow A + ad + C$	2	1
ADDC A, @r	Додавання байта з РПД з акумулятором і прапорцем *C*	$A \leftarrow A + M_{РПД}(r) + C$	1	1
ADDC A, #data	Додавання константи з акумулятором і прапорцем *C*	$A \leftarrow A + \#data + C$	2	1
DA A	Десяткова корекція *A* при додаванні	Алгоритм десяткової корекції	1	1
SUBB A, R	Віднімання з акумулятора регістра і прапорця *C*	$A \leftarrow A - R - C$	1	1
SUBB A, ad	Віднімання з *A* байта з прямою адресацією і прапорця *C*	$A \leftarrow A - ad - C$	2	1
SUBB A, @r	Віднімання з *A* байта РПД і прапорця *C*	$A \leftarrow A - M_{РПД}(r) - C$	1	1
SUBB A, #data	Віднімання з *A* константи і прапорця *C*	$A \leftarrow A - \#data - C$	2	1
INC A	Інкремент *A*	$A \leftarrow A + 1$	1	1
INC R	Інкремент регістра	$R \leftarrow R + 1$	1	1
INC ad	Інкремент байта з прямою адресацією	$ad \leftarrow ad + 1$	2	1
INC @r	Інкремент байта в РПД	$M_{РПД}(r) \leftarrow M_{РПД}(r) + 1$	1	1

Продовження табл. 7.12

1	2	3	4	5
INC DPTR	Інкремент вказівника даних	$DPTR \leftarrow DPTR + 1$	1	2
DEC A	Декремент *A*	$A \leftarrow A - 1$	1	1
DEC R	Декремент регістра	$R \leftarrow R - 1$	1	1
DEC ad	Декремент байта з прямою адресацією	$ad \leftarrow ad - 1$	2	1
DEC @r	Декремент байта в РПД	$M_{РПД}(r) \leftarrow M_{РПД}(r) - 1$	1	1
MUL AB	Множення *A* на регістр *B*	$(B, A) \leftarrow A \times B$	1	4
DIV AB	Ділення *A* на регістр *B*	$A \leftarrow A/B$, залишок у *B*	1	4
ЛОГІЧНІ КОМАНДИ				
ANL A, R	Логічна операція *I* регістра і *A*	$A \leftarrow A \wedge R$	1	1
ANL A, ad	*I* байта з прямою адресацією і *A*	$A \leftarrow A \wedge ad$	2	1
ANL A, @r	*I* байта РПД і *A*	$A \leftarrow A \wedge M_{РПД}(r)$	1	1
ANL A, #data	*I* константи і *A*	$A \leftarrow A \wedge$ *#data*	2	1
ANL ad, A	*I* байта з прямою адресацією і *A*	$ad \leftarrow ad \wedge A$	2	1
ANL ad, #data	*I* байта з прямою адресацією і константи	$ad \leftarrow ad \wedge$ *#data*	3	2
ORL A, R	АБО регістра і *A*	$A \leftarrow A \vee R$	1	1
ORL A, ad	АБО байта з прямою адресацією і *A*	$A \leftarrow A \vee ad$	2	2
ORL A, @r	АБО байта РПД і *A*	$A \leftarrow A \vee M_{РПД}(r)$	1	1
ORL A, #data	АБО константи і *A*	$A \leftarrow A \vee$ *#data*	2	1
ORL ad, A	АБО байта з прямою адресацією і *A*	$ad \leftarrow ad \vee A$	2	1
ORL ad, #data	АБО байта з прямою адресацією і константи	$ad \leftarrow ad \vee$ *#data*	3	2
XRL A, R	ВИКЛЮЧНЕ АБО регістра і *A*	$A \leftarrow A \oplus R$	1	1
XRL A, ad	ВИКЛЮЧНЕ АБО байта з прямою адресацією і *A*	$A \leftarrow A \oplus ad$	2	1
XRL A, @r	ВИКЛЮЧНЕ АБО байта РПД і *A*	$A \leftarrow A \oplus M_{РПД}(r)$	1	1
XRL A, #data	ВИКЛЮЧНЕ АБО константи і *A*	$A \leftarrow A \oplus$ *#data*	2	1
XRL ad, A	ВИКЛЮЧНЕ АБО байта з прямою адресацією і *A*	$ad \leftarrow ad \oplus A$	2	1
XRL ad, #data	ВИКЛЮЧНЕ АБО байта з прямою адресацією і константи	$ad \leftarrow ad \oplus$ *#data*	3	2
CLR A	Скидання *A*	$A \leftarrow 0$	1	1
CPL A	Інверсія *A*	$A \leftarrow \overline{A}$	1	1
RL A	Циклічний зсув ліворуч	$A_{n+1} \leftarrow A_n$, $n = 0–6$, $A0 \leftarrow A7$	1	1
RLC A	Зсув ліворуч через прапорець *C*	$A_{n+1} \leftarrow A_n$, $n = 0–6$, $A_0 \leftarrow C$, $C \leftarrow A_7$	1	1
RR A	Циклічний зсув праворуч	$A_n \leftarrow A_{n+1}$, $n = 0–6$, $A7 \leftarrow A0$	1	1
RRC A	Зсув праворуч через прапорець *C*	$A_n \leftarrow A_{n+1}$, $n = 0–6$, $A7 \leftarrow C$, $C \leftarrow A0$	1	1

Продовження табл. 7.12

КОМАНДИ ОПЕРАЦІЙ З БІТАМИ				
CLR C	Скидання прапорця *C*	$C \leftarrow 0$	1	1
CPL C	Інверсія прапорця *C*	$C \leftarrow \overline{C}$	1	1
SETB C	Установлення прапорця *C*	$C \leftarrow 1$	1	1
CLR bit	Скидання біта	$Bit \leftarrow 0$	2	1
CPL bit	Інверсія біта	$Bit \leftarrow \overline{bit}$	2	1
SETB bit	Установлення біта	$Bit \leftarrow 1$	2	1
ANL C, bit	Логічне *І* біта і прапорця *C*	$C \leftarrow C \wedge bit$	2	2
ANL C, /bit	Логічне *І* інверсії біта і прапорця *C*	$C \leftarrow C \wedge \overline{bit}$	2	2
ORL C, bit	Логічне АБО біта і прапорця *C*	$C \leftarrow C \vee bit$	2	2
ORL C, /bit	Логічне АБО інверсії біта і прапорця *C*	$C \leftarrow C \vee \overline{bit}$	2	2
MOV C, bit	Пересилання біта у прапорець *C*	$C \leftarrow bit$	2	1
MOV bit, C	Пересилання прапорця *C* у біт	$bit \leftarrow C$	2	2
*LJMP ad*16	Безумовний довгий перехід у повній ємності пам'яті програм	$PC \leftarrow ad16$	3	2
*AJMP ad*11	Абсолютний перехід усередині сторінки 2 кбайт	$PC \leftarrow PC + 2$ $PC_{0-10} \leftarrow ad11$	2	2
*SJMP ad*8	Короткий відносний перехід усередині сторінки ПП 256 байт	$PC \leftarrow PC + 2$ $PC \leftarrow PC + ad8$	2	2
КОМАНДИ ПЕРЕХОДІВ				
JMP @A+DPTR	Непрямий відносний перехід	$PC \leftarrow A + DPTR$	1	2
*JC ad*8	Перехід, якщо $C = 1$	$PC \leftarrow PC + 2$, якщо $C = 1$, то $PC \leftarrow PC + ad8$	2	2
JNC ad8	Перехід, якщо $C = 0$	$PC \leftarrow PC + 2$, якщо $C = 0$, то $PC \leftarrow PC + ad8$	2	2
JZ ad8	Перехід, якщо $A = 0$	$PC \leftarrow PC + 2$, якщо $A = 0$, то $PC \leftarrow PC + ad8$	2	2
*JNZ ad*8	Перехід, якщо $A \neq 0$	$PC \leftarrow PC + 2$, якщо $A \neq 0$, то $PC \leftarrow PC + ad8$	2	2
*JB bit, ad*8	Перехід, якщо біт дорівнює 1	$PC \leftarrow PC + 3$, якщо $bit = 1$, то $PC \leftarrow PC + ad8$	3	2
*JCB bit, ad*8	Перехід, якщо біт дорівнює 1 з наступним скиданням біта	$PC \leftarrow PC + 3$, якщо $bit = 1$, то $Bit \leftarrow 0$, $PC \leftarrow PC + ad8$	3	2
*JNB bit, ad*8	Перехід, якщо біт дорівнює 0	$PC \leftarrow PC + 3$, якщо $bit = 0$, то $PC \leftarrow PC + ad8$	3	2

Продовження табл. 7.12

1	2	3	4	5
*DJNZ R, ad*8	Декремент *R* і перехід, якщо не нуль	$PC \leftarrow PC + 2$, $R \leftarrow R - 1$, якщо $R \neq 0$, то $PC \leftarrow PC + ad8$	2	2
*JNZ ad, ad*8	Декремент байта з прямою адресацією і перехід, якщо не нуль	$PC \leftarrow PC + 2$, $ad \leftarrow ad - 1$, якщо $ad \neq 0$, то $PC \leftarrow PC + ad8$	3	2
*CJNE A, ad, ad*8	Порівняння *A* байта з прямою адресацією і перехід, якщо не дорівнює	$PC \leftarrow PC + 3$, якщо $A \neq ad$, то $PC \leftarrow PC + ad8$; якщо $A < ad$, то $C = 1$, інакше $C = 0$	3	2
*CJNE A, #data, ad*8	Порівняння *A* з константою і перехід, якщо не дорівнює	$PC \leftarrow PC + 3$, якщо $A \neq$ *#data*, то $PC \leftarrow PC + ad8$; якщо $A <$ *#data*, то $C = 1$, інакше $C = 0$	3	2
*CJNE R, #data, ad*8	Порівняння регістра з константою і перехід, якщо не дорівнює	$PC \leftarrow PC + 3$, якщо $R \neq$ *#data*, то $PC \leftarrow PC + ad8$; якщо $R <$ *#data*, то $C = 1$, інакше $C = 0$	3	2
*CJNE @r, #data, ad*8	Порівняння байта в РПД з константою і перехід, якщо не дорівнює	$PC \leftarrow PC + 3$, якщо $M_{РПД}(r) \neq$ *#data*, то $PC \leftarrow PC + ad8$; якщо $M_{РПД}(r) <$ *#data*, то $C = 1$, інакше $C = 0$	3	2
*LCALL ad*16	Довгий виклик підпрограми	$PC \leftarrow PC + 3$, $SP \leftarrow SP + 1$, $M_{РПД}(SP) \leftarrow PC0 - 7$, $SP \leftarrow SP + 1$, $M_{РПД}(SP) \leftarrow PC8 - 15$, $PC \leftarrow ad16$	3	2
*ACALL ad*11	Абсолютний виклик підпрограми всередині сторінки 2 кбайт	$PC \leftarrow PC + 2$, $SP \leftarrow SP + 1$, $M_{РПД}(SP) \leftarrow PC_{0-7}$, $SP \leftarrow SP + 1$, $M_{РПД}(SP) \leftarrow PC_{8-15}$, $PC_{0-10} \leftarrow ad11$	2	2
RET	Повернення з підпрограми*	$PC_{8-15} \leftarrow M_{РПД}(SP)$, $SP \leftarrow SP - 1$, $PC_{0-7} \leftarrow M_{РПД}(SP)$, $SP \leftarrow SP - 1$	1	2

Закінчення табл. 7.12

1	2	3	4	5
RETI	Повернення з підпрограми обробки переривання**	$PC_{8–15} \leftarrow M_{РПД}(SP)$, $SP \leftarrow SP - 1$, $PC_{0–7} \leftarrow M_{РПД}(SP)$, $SP \leftarrow SP - 1$	1	2
NOP	Немає операції	$PC \leftarrow PC + 1$	1	1

* Команда не впливає на прапорці.

** Команда встановлює логіку переривань: дозволяє прийом переривань з рівнем пріоритету, що дорівнює рівню переривання, яке оброблялося перед цим.

Вплив команд на прапорці ілюструє табл. 7.13, в якій позначено: «+» – команда впливає на прапорець, «–» – не впливає; «1» – встановлює в 1; «0» – скидає в нуль.

Таблиця 7.13. **Вплив команд на прапорці**

Операція	Команда	Прапорці		
		OV	*C*	*AC*
Пересилання	*MOV PSW, source*	+	+	+
Додавання	*ADD, ADDC, SUBB,*	+	+	+
Множення	*MUL*	+	0	–
Ділення	*DIV*	+	0	–
Порівняння і перехід	*CJNE*	–	+	–
Десяткова корекція	*DAA*	–	+	+
Зсув	*RRC, RLC*	–	+	–
Керування прапорцями	*SETB C*	–	1	–
	CLR C	–	0	–
	CPL C	–	$\overline{C}$	–
Команди роботи з бітами	*ANL C, bit*	–	+	–
	ANL C, /bit	–	+	–
	ORL C, bit	–	+	–
	ORL C, /bit	–	+	–
	MOV C, bit	–	+	–

Примітка. Прапорець парності *P* установлюється за кожною командою, що модифікує вміст акумулятора згідно із цим умістом. Прапорець нуля *Z* фізично не існує, однак при виконанні команд *JZ, JNZ* відбувається порівняння *A* з нулем.

Контрольні запитання

1. Чим відрізняється прапорець нуля в ОМК К1816ВЕ51 та МП *i*8086?
2. Які є групи команд?
3. Які операції можливі над бітами?
4. Які арифметичні операції можна здійснювати в ОМК К1816ВЕ51?
5. Які логічні операції можна здійснювати в ОМК К1816ВЕ51?
6. Скільки існує байтів з прямою адресацією?
7. Якими форматами даних оперує система команд ОМК К1816ВЕ51?

7.3. Розширення можливостей однокристальних мікроконтролерів

Для розширення можливостей ОМК використовують: розширення пам'яті програм, розширення пам'яті даних, розширення простору введення-виведення.

Розширення пам'яті програм. Розширення пам'яті програм до 64 кбайт виконується приєднанням зовнішніх ВІС ПЗП до ОМК. При цьому, якщо на вивід $\overline{EA}$ подано логічну одиницю, то використовується як внутрішня, так і зовнішня пам'ять програм: при адресах 00–0*FFFH* звернення відбуваються до внутрішньої, при адресах 1000*H*–*FFFFH* – до зовнішньої пам'яті програм. Якщо на вивід $\overline{EA}$ подано логічний нуль, то звернення відбуваються тільки до зовнішньої пам'яті програм.

Функціональну схему з'єднання зовнішньої пам'яті програм показано на рис. 7.22. Через порти *P*0 і *P*2 передаються молодша і старша частини 16-розрядної адреси комірки зовнішньої пам'яті, що супроводжується сигналом *ALE*. Адреса утримується на виводах порту *P*2 протягом усього машинного циклу звернення до пам'яті, а на виводах порту *P*0 – лише протягом одного стану машинного циклу. Тому молодша частина адреси запам'ятовується в регістрі-замку *RG* за сигналом *ALE*. Після видачі молодшого байта адреси порт *P*0 переходить у високоімпедансний стан та очікується надходження даних із ПЗП.

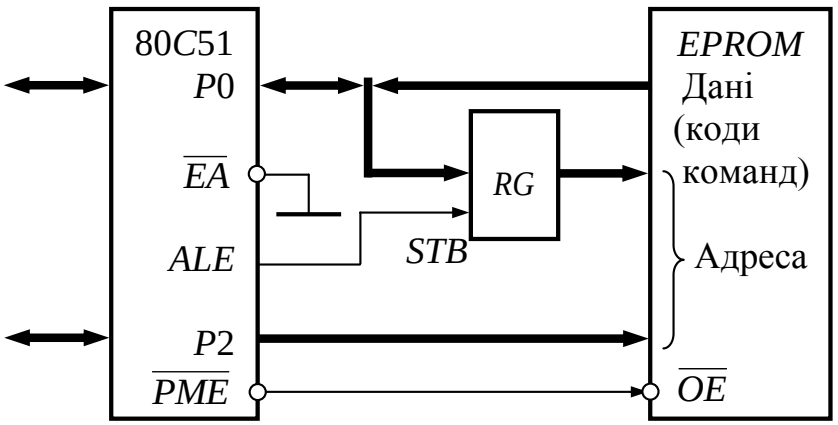


Рис. 7.22. Функціональна схема з'єднання зовнішньої пам'яті програм з ОМК

Порт *P*0 функціонує як мультиплексована шина адреси/даних. Сигнал $\overline{PME}$ формується двічі за машинний цикл. Цей сигнал дозволяє вибірку байта з ПЗП. Таким чином, протягом одного машинного циклу вибирається два байти команди. Якщо команда однобайтова, другий байт ігнорується. Цей байт обирається при переході до наступної команди.

Часові діаграми машинних циклів читання зовнішньої пам’яті програм показано на рис. 7.23.

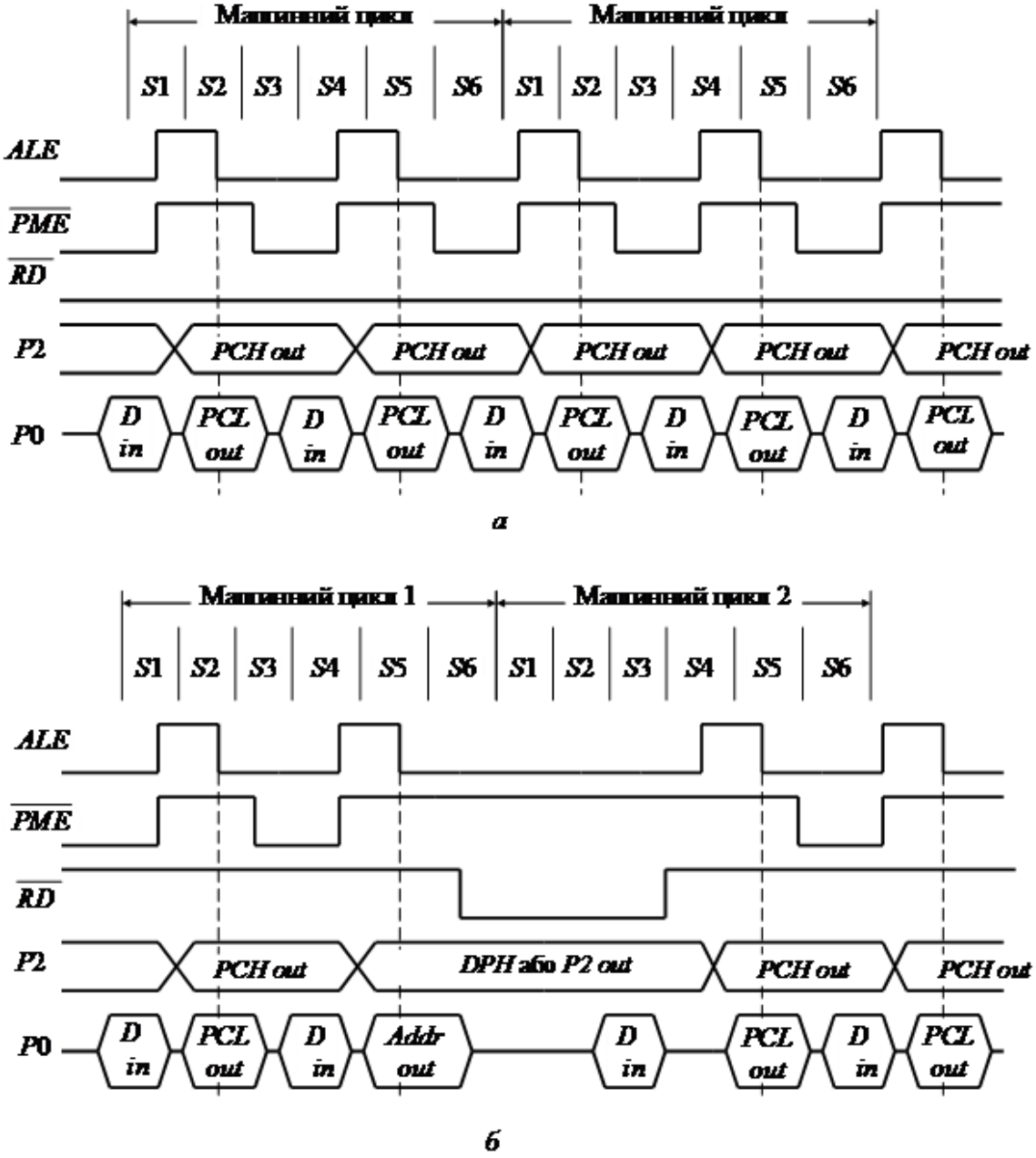


Рис. 7.23. Часові діаграми машинних циклів читання зовнішньої пам’яті програм: *а* – не виконується команда *MOVX*; *б* – виконується команда *MOVX*; *PCL out* – видача молодшого байта лічильника команд *PC*; *PCH out* – видача старшого байта лічильника команд *PC*; *DPH* – видача старшого байта регістра *DPTR*; *P*2 *out* – видача вмісту регістра фіксатора порту *P*2; *Addr out* – видача молодшого байта адреси зовнішньої пам’яті даних з регістрів *R*0, *R*1 або регістра *DPL*; *D in* – уведення байта команди або даних

При виконанні команди *MOVX* сигнал *ALE* формується у другому машинному циклі лише один раз.

Зазначимо, що якщо ОМК працює з внутрішньою пам'яттю програм, то сигнал $\overline{PME}$ не формується, і адреса на порти *P*0 і *P*2 не видається. Проте сигнал *ALE* і в цьому випадку формується двічі за період, якщо не виконується команда *MOVX*. Вивід *ALE* можна використовувати як вихідний сигнал синхронізації.

Розширення пам'яті даних. В ОМК передбачено можливість розширення пам'яті даних до 64 кбайт з'єднанням зовнішніх ВІС пам'яті з ОМК. На рис. 7.24 показано функціональну схему з'єднання зовнішньої пам'яті даних з ОМК.

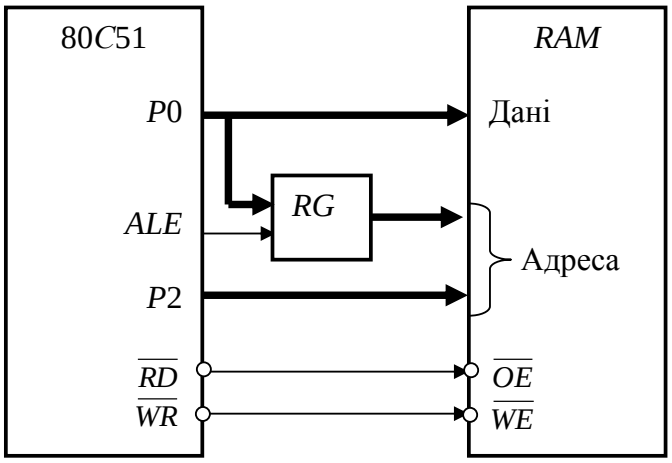


Рис. 7.24. Функціональна схема з'єднання зовнішньої пам'яті даних з ОМК

Часові діаграми циклів читання і запису в зовнішню пам'ять даних ілюструє рис. 7.25. З'єднання пам'яті даних з ОМК аналогічне з'єднанню пам'яті програм.

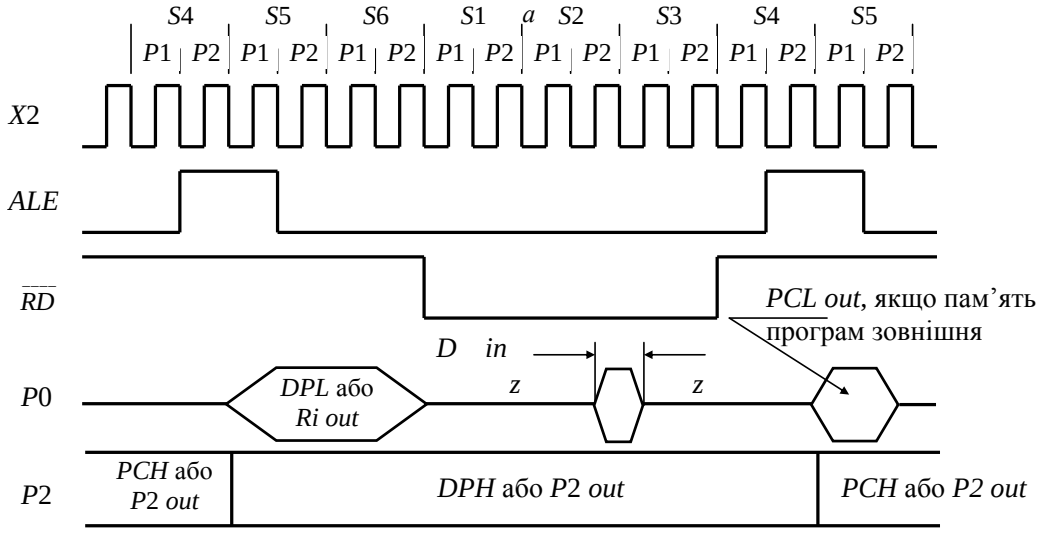


Рис. 7.25. Часові діаграми циклів:
а – читання зовнішньої пам'яті даних; *б* – запис у зовнішню пам'ять даних
(Див. також с. 362.)

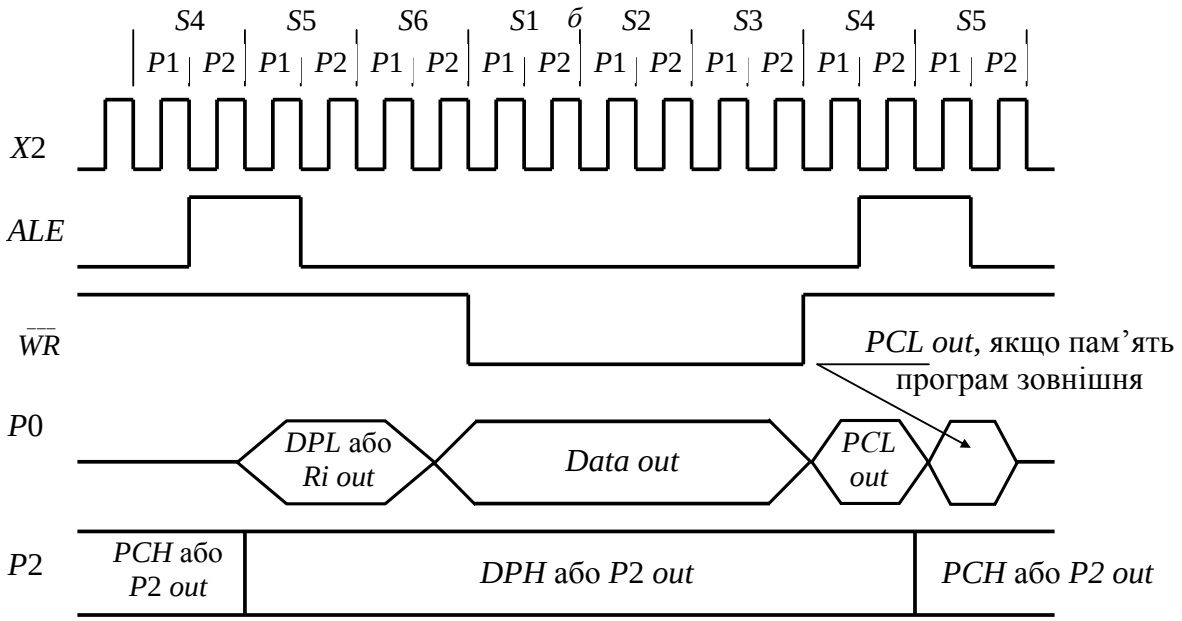


Рис. 7.25. Закінчення

Якщо треба, в ОМК можуть одночасно використовуватися зовнішня пам'ять даних і пам'ять програм (рис. 7.26).

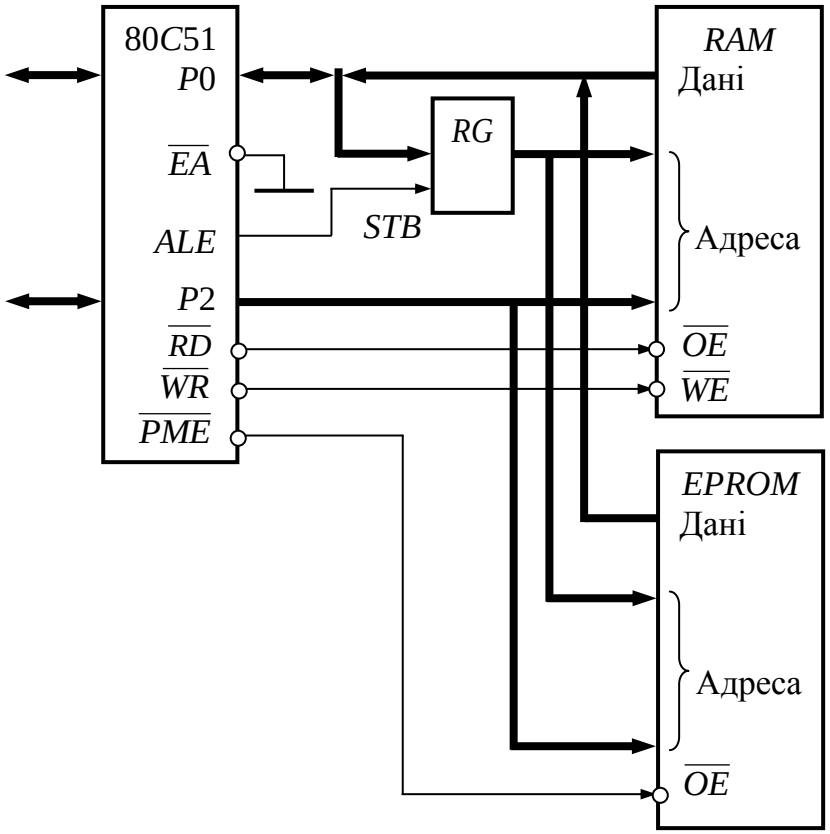


Рис. 7.26. Функціональна схема ОМК із зовнішньою пам'яттю даних і програм

Розширення простору введення-виведення. Одну з можливих схем з'єднання контролера клавіатури і дисплея з ОМК показано на рис. 7.27.

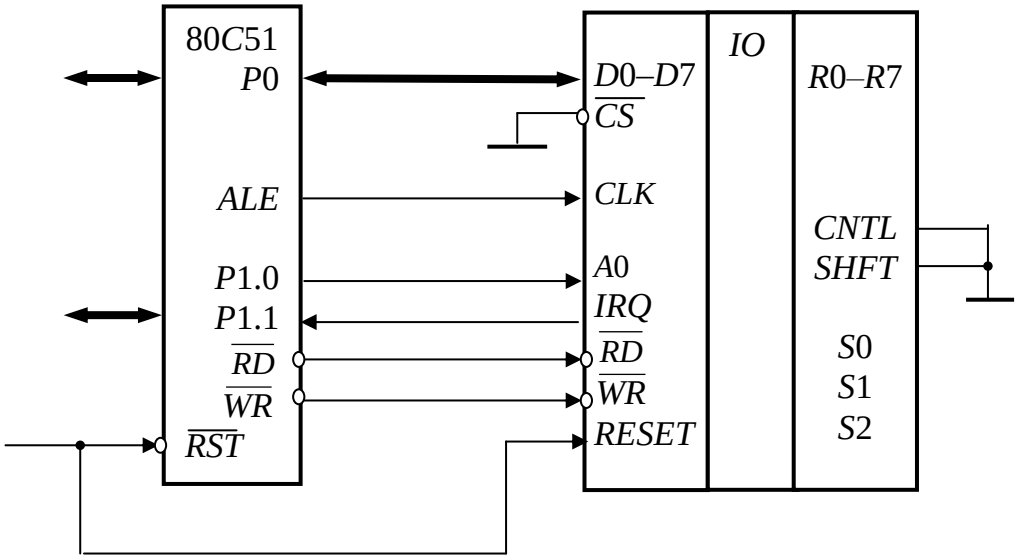


Рис. 7.27. З'єднання контролера клавіатури і дисплея з ОМК

При такому з'єднані адреса контролера входить до адресного простору зовнішньої пам'яті даних. Лінія *P*1.0 порту *P*1 з'єднується з лінією *A*0 контролера і має бути встановлена або скинута перед зверненням ОМК до контролера залежно від типу звернення (керування або передача даних). Вхід вибірки кристала контролера з'єднаний із земляною лінією, тому контролер завжди готовий до обміну інформацією з ОМК. Вихід сигналу запиту переривання *IRQ* контролера з'єднаний з лінією *P*1.1 ОМК. Опитування стану виходу *IRQ* здійснюється для визначення факту натискання клавіші.

Шина даних і лінії читання/запису контролера об'єднуються безпосередньо з відповідними лініями ОМК. На вхід *CLK* подається сигнал частотою приблизно 2 МГц з виходу *ALE*.

Контрольні запитання

1. Як здійснюється розширення пам'яті програм в ОМК?
2. Як здійснюються звернення до внутрішньої та зовнішньої пам'яті програм?
3. За допомогою яких портів здійснюється передача адреси комірки зовнішньої пам'яті програм?
4. Як здійснюється розширення пам'яті даних в ОМК?
5. За допомогою яких портів здійснюється передача адреси комірки зовнішньої пам'яті даних?
6. Скільки додаткових портів введення-виведення можна з'єднати з ОМК?

7.4. Застосування однокристального мікроконтролера 83*C*51*FA* для керування двигуном постійного струму

Двигуни постійного струму широко використовують у промислових і побутових пристроях. У багатьох випадках важливим є прецизійне керування швидкістю обертання двигуна та можливість зміни напряму обертання. Наприклад, двигун постійного струму в побутовій техніці магнітного звукозапису має обертатися з постійною швидкістю. Зміна напряму обертання досягається зміною полярності напруги, що подається або на колекторну обмотку, або на обмотку збудження.

Використання ОМК 83*C*51*FA* дозволяє керувати двигуном постійного струму з реалізацією широтно-імпульсної модуляції напруги, яка подається на обмотку збудження чи на колекторну обмотку. На базі ОМК 83*C*51*FA* може бути запрограмовано до п’яти широтно-імпульсних модуляторів (ШІМ).

Однокристальний мікроконтролер 83*C*51*FA* являє собою 8-розрядний мікроконтролер, що базується на архітектурі *i*8051, але має декілька нових функцій, зокрема *масив програмовних лічильників – Programmable Counter Array* (*PCA*). Масив складається з 16-розрядного таймера *PCA* та 5 окремих модулів. Таймер *PCA* складається з двох 8-розрядних регістрів – *CL* (молодший байт) та *CH* (старший байт), доступних для всіх модулів. Таймер *PCA* можна запрограмувати на використання вхідних даних із чотирьох різних джерел. Максимальна частота лічби – 4 МГц, тобто 1/4 частоти генератора. Окремі виводи порту *P*1 використовуються для взаємодії кожного модуля і таймера із зовнішніми пристроями. Коли виводи порту не використовуються для роботи модулів *PCA*, вони можуть бути використані як лінії звичайного порту введення-виведення.

Модулі *PCA* можна запрограмувати на режим захоплення події (*capture*) або режим порівняння (*compare*), який має такі підрежими: ПТ, режим високошвидкісного виведення, режим ШІМ (*PWM*), режим сторожового таймера (*watchdog*-таймера) (тільки для четвертого модуля). Кожний модуль має 8-розрядний регістр режиму *CCAPMn* (табл. 7.14), 16-розрядний регістр порівняння/захоплення, який складається з двох 8-розрядних регістрів *CCAPnL* та *CCAPnH*, де *n* може набувати значення від 0 до 4.

Установленням відповідних бітів у регістрах режимів *CCAPMn* можна запрограмувати кожний модуль на функціонування в одному з режимів: захоплення події або порівняння.

Таблиця 7.14*. Позначення бітів регістра режиму *CCAPMn

Біт	Позначення	Призначення
7	–	Не використовується
6	*ECOMn*	Біт дозволу функції порівняння; *ECOMn* = 1 для функцій, що потребують порівняння вмісту регістрів порівняння/захоплення із вмістом 16-розрядного таймера, наприклад, функцій ПТ, високошвидкісного виведення, *watchdog*-таймера
5	*CAPPn*	Біт захоплення за переднім фронтом сигналу
4	*CAPNn*	Біт захоплення за заднім фронтом сигналу
3	*MATn*	Біт визначення збігу вмісту регістрів захоплення/порівняння із вмістом 16-розрядного таймера
2	*TOGn*	Біт дозволу виведення за умовою збігу вмісту регістрів захоплення/порівняння із вмістом 16-розрядного таймера
1	*PWMn*	Біт дозволу генерації сигналу ШІМ за збігом молодшого байта вмісту регістра порівняння/захоплення та молодшого байта вмісту таймера *PCA*
0	*ECCCFn*	Біт дозволу генерації переривання за прапорцем порівняння/захоплення *CCFn* регістра *CCON*

Режим захоплення події. Подією називається будь-яка зміна рівня сигналу на входах. ОМК можна запрограмувати на визначення таких подій:

– кожного переходу рівня вхідного сигналу з нуля в одиницю (*CAPPn* = 1);
– кожного переходу рівня вхідного сигналу з одиниці в нуль (*CAPNn* = 1);
– того чи іншого переходу рівня сигналу (*CAPPn* = 1 та *CAPNn* = 1).

Після того, як запрограмована подія відбулася, час настання цієї події, відрахований таймером *PCA*, разом з інформацією про стан входів записується в стек подій *FIFO*. Операція запису часу події в стек називається *захопленням події.* Якщо встановлено біт дозволу *ECCFn*, захоплення події супроводжується генерацією запиту переривання ЦП на обслуговування модуля.

Режим порівняння. Якщо модуль запрограмовано на функціонування в одному з підрежимів порівняння (ПТ, високошвидкісний вивід, *watchdog*-таймер), програма завантажує в регістри захоплення/порівняння величину, яка порівнюється із вмістом 16-розрядного таймера; ОМК генерує переривання в разі збігу цих величин.

У підрежимі ПТ встановлюється прапорець переповнення у разі збігу значень таймера і регістра захоплення/порівняння.

Підрежим високошвидкісного виводу призначений для генерації заданих подій у заданий час. Час задається значенням регістра захоплення/порівняння *CCAPn*.

У підрежимі сторожового таймера генерується переривання, якщо деяка ділянка програми виконується за більший час, ніж заданий у регістрі. Це дозволяє уникнути «зависання» програми.

Підрежим ШІМ – єдиний режим, що використовує тільки 8-розрядний регістр захоплення/порівняння. У старший байт (*CCAPnH*) обраного модуля завантажується значення від 0 до *FFH*. Це значення переноситься в молодший байт того самого модуля і порівнюється з молодшим байтом регістра таймера *PCA*. За умови *CL* < *CCAPnL* на відповідному виводі встановлюється рівень логічного нуля; за умови *CL* > > *CCAPnL* – рівень логічної одиниці.

Крім регістрів *CCAPMn*, для забезпечення роботи таймера *PCA* введено ще два регістри спеціальних функцій – *CCON* та *CMOD*. Регістр *CCON* (табл. 7.15) допускає бітову адресацію. Адреса регістра *CCON* – 0*D*8*H*, значення при скиданні – 00*x*00000*B*.

Таблиця 7.15. **Позначення бітів регістра *CCON***

Біт	Позначення	Призначення
7	*CF*	Прапорець переповнення таймера
6	*CR*	Запуск таймера *PCA*
5	–	Не використовується
4	*CCF*4	Прапорці модулів, що використовуються для визначення модуля, який генерує переривання *PCA*
3	*CCF*3	
2	*CCF*2	
1	*CCF*1	
0	*CCF*0	

Регістр *CMOD* (табл. 7.16) не допускає адресацію окремих бітів. Адреса регістра *CMOD* – 0*D*9*H*, значення при скиданні – 00*xxx*000*B*.

Таблиця 7.16. **Позначення бітів регістра *CMOD***

Біт	Позначення	Призначення
5	–	Не використовується
4		
3		
2	*CPS*1	Біти, що визначають джерело тактування таймера *PCA*: при 00 – внутрішній генератор, F_{osc}/12; при 01 – внутрішній генератор, F_{osc}/4; при 10 – переповнення таймера 0; при 11 – зовнішній сигнал (уведення з P1.2)
1	*CPS*0	
0	*ECF*	Дозвіл переривання за прапорцем *CF*

Якщо один з модулів запрограмований на режим ШІМ, то в старший байт регістра порівняння необхідно завантажити значення від 0 до 255, яке визначає коефіцієнт заповнення ШІМ. Для ОМК 83*C*51*FA* завантаження 0 в *CCAPnH* відповідає коефіцієнту заповнення 1, а 255 (*OFFh*) – коефіцієнту заповнення 0,004. Часові діаграми та значення коефіцієнта заповнення ШІМ і регістра *CCAPnH* наведено в табл. 7.17.

Таблиця 7.17. **Часові діаграми та значення коефіцієнта заповнення ШІМ**

Коефіцієнт заповнення	Значення регістра *CCAPnH*	Вихідний сигнал ШІМ
1	00	
0,9	25	
0,5	128	
0,1	230	
0,004	255	

Запуск таймера *PCA* здійснюється встановленням біта *CR* (див. табл. 7.15) регістра *CCON*, який допускає бітову адресацію. Установлення та скидання цього біта здійснюється відповідними бітовими командами – *CLR bit* на *SETB bit* (див. табл. 7.12).

Приклад 7.1. Запрограмувати модуль 2 для генерації сигналу ШІМ для кола керування двигуном.

Програма має такий вигляд:

```
MOV   CMOD, #06     ; Обирається зовнішній сигнал тактування
MOV   CCAPM2,#42H   ; Установлюється підрежим ШІМ
MOV   CCAP2H,#0     ; Значення нуля відповідає коефіцієнту
                    ; заповнення 100 % (5 V)
SETB  CR            ; Запуск таймера
END
```

Принципову схему модуля керування двигуном постійного струму показано на рис. 7.28.

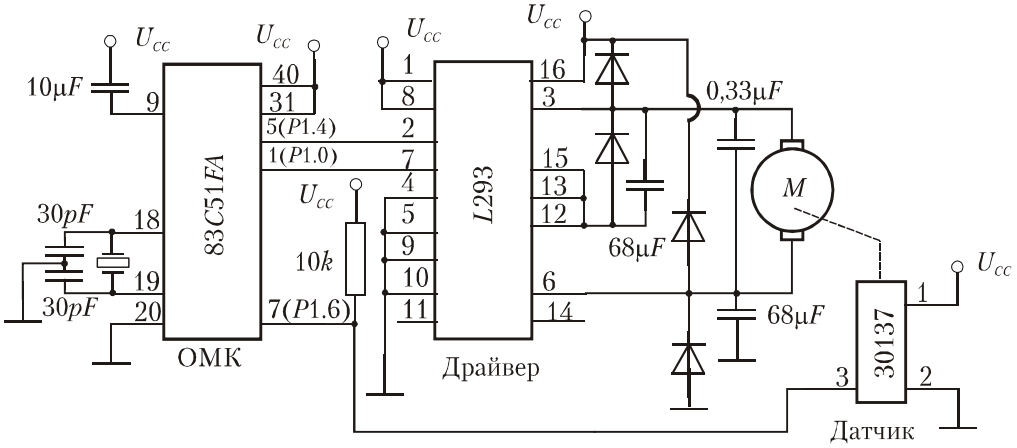


Рис. 7.28. Принципова схема модуля керування двигуном постійного струму

Схема містить ОМК 83*C*51*FA* з колами скидання та синхронізації, спеціалізовану ВІС драйвера *L*293, двигун та імпульсний датчик швидкості 30137.

Імпульси ТТЛ-рівня з виходу цифрового датчика Холла надходять на вхід *P*1.6 ОМК. Залежно від частоти цих імпульсів ОМК збільшує або зменшує коефіцієнт заповнення ЩІМ на виводах *P*1.4, *P*1.5, регулюючи таким чином швидкість обертання двигуна.

Контрольні запитання

1. Наведіть основні характеристики ОМК 83*C*51*FA*.
2. Укажіть особливості режиму захоплення події.
3. Укажіть особливості режиму ШІМ.
4. Укажіть особливості режиму *watchdog*-таймера.
5. Які події можуть бути зафіксовані в режимі захоплення події?
6. Поясніть принцип формування сигналу ШІМ на виводі ОМК.

7.5. Архітектура і функціональні можливості 16-розрядних однокристальних мікроконтролерів серії *MCS*196/296

Загальна характеристика серії *MCS*196/296. 16-розрядні ОМК *MCS* 196/296 виробництва компанії *Intel* використовуються у вмонтованих МПС керування в авіаційній та автомобільній промисловості, верстато- і роботобудівництві, енергетиці та електромеханіці, пристроях побутової техніки. На сьогодні серія 16-розрядних ОМК містить понад 30 типів контролерів із продуктивністю від 1 до 16 млн операцій типу «регістр-регістр»* за секунду. Основні характеристики ВІС наведено в табл. 7.18.

Відповідно до типу інтегрованих на кристал периферійних пристроїв розрізняють такі класи ОМК серії *MCS* 196/296:

– з інтегрованими пристроями високошвидкісного введення-виведення;
– з інтегрованими процесорами подій;
– з інтегрованими засобами керування двигунами;
– з інтегрованими засобами цифрової обробки.

Однокристальні мікроконтролери з інтегрованими пристроями високошвидкісного введення-виведення. 16-розрядні ОМК з інтегрованими пристроями високошвидкісного введення-виведення даних (*HSIO – High-Speed Input/Output*) орієнтовані на вирішення завдань керування об’єктами та процесами в реальному часі. Ці ОМК знаходять широке використання не тільки в комп’ютерній техніці (принтери, плотери і т. ін.), але й в електромеханіці, робототехніці, військовій техніці завдяки наявності інтегрованого АЦП, модуля *HSIO*, процесора периферійних транзакцій та порту послідовного зв’язку. Конкретні ОМК вирізняються тактовою частотою, кількістю послідовних портів, входів АЦП, ємністю інтегрованої на кристал пам’яті, наявністю тих чи інших периферійних блоків, наприклад генератора періодичних сигналів.

* Операції типу «регістр-регістр» є найбільш швидкодіючими, оскільки оперують лише з внутрішніми регістрами МП. Прикладом таких операцій є *MOV R*0, *R*1; *ADD A*, *R*3.

Таблиця 7.18. **Основні технічні характеристики 16-розрядних ОМК**

Тип ВІС	Тактова частота, МГц	Адресний простір, кбайт	Ємність ПЗП, кбайт	Ємність регістрового ОЗП, байт	Ємність флеш-пам'яті програм, байт	Кількість таймерів	Кількість каналів АЦП	Кількість ліній введення-виведення	Кількість послідовних портів	Режим тестування	Тип процесора подій	Характерні особливості
1	2	3	4	5	6	7	8	9	10	11	12	13
Мікроконтролери з вбудованим пристроєм високошвидкісного введення-виведення												
8*x*C196*KB*	12,16	64	8	232	–	2	8	48	1	+	Модуль високошвидкісного введення-виведення (4 входи, 6 виходів)	–
8*x*C198	16	64	8	232	–	2	0 або 4	48	1	+	Те саме	8-розрядна зовнішня шина, менша вартість
8*x*C196*KC*	16,2	64	16	488	–	2	8	48	1	+	„	ШІМ-генератор, сервер периферійних транзакцій
8*x*C196*KD*	16,2	64	32	1000	–	2	8	48	1	+	”	Те саме
Мікроконтролери з вмонтованими процесорами подій												
8*x*C196*KR*	16	64	16	488	256	2	8	56	2	+	10 каналів захоплення/ порівняння, 2 канали тільки порівняння	Блок сегментації пам'яті, порт для міжпроцесорних комунікацій
8*x*C196*KT*	16	64	32	1000	512	2	8	56	2	+	Те саме	Контролер шини з розширеними можливостями, порт для міжпроцесорних комунікацій
8*x*C196*NP*	25	100	4	1000	–	2	–	33	1	+	4 канали процесора подій	Низький рівень живлення 3,3 В, триканальний ШІМ-генератор, сервер периферійних транзакцій, 4 зовнішні переривання

Продовження табл. 7.18

1	2	3	4	5	6	7	8	9	10	11	12	13
8*x*C196*NT*	16	100	32	1000	512	2	4	56	2	+	10 каналів захоплення/ порівняння, 2 канали тільки порівняння	–
8*x*C196*NU*	50	100	–	1000	–	2	–	56	1	+	4 канали процесора подій	32-розрядний акумулятор
Мікроконтролери для керування двигунами												
8*x*C196*MC*	16	64	16	488	–	2	13	53	1	+	4 канали захоплення/порівняння, 4 канали тільки порівняння	Триканальний ШІМ-генератор, процесор подій, процесор транзакцій, триканальний генератор періодичних сигналів
8*x*C196*MD*	16	64	16	488	–	2	14	64	1	+	6 каналів захоплення/порівняння, 6 каналів тільки порівняння	Вмонтований генератор
8*x*C196*MH*	16	64	32	744	–	2	8	52	2	+	2 канали захоплення/порівняння, 4 канали тільки порівняння	–
Мікроконтролери з інтегрованими засобами цифрової обробки												
8*x*0C296*SA*	50	600	32	512	2К	2		64	1		4 канали захоплення/порівняння	Збільшення в 2 або 4 рази тактової частоти, 19 джерел переривань, із них 4 зовнішні, триканальний ШІМ-генератор, набір процесорів подій, вмонтований модуль вибірки зовнішніх пристроїв

Структурна схема ОМК з інтегрованим пристроєм високошвидкісного введення-виведення (рис. 7.29) містить:

– модуль ЦП;
– контролер пам’яті (КП);
– контролер переривань *IC* (*Interrupt Controller*);
– сервер периферійних транзакцій *PTS* (*Peripheral Transaction Server*);
– ПЗП;
– блок АЦП;
– вбудований триканальний ШІМ-генератор *PWM*;
– послідовний порт (*SIO*);
– порти введення-виведення (*P*0–*P*5);
– *watchdog*-таймер (*WDT*).

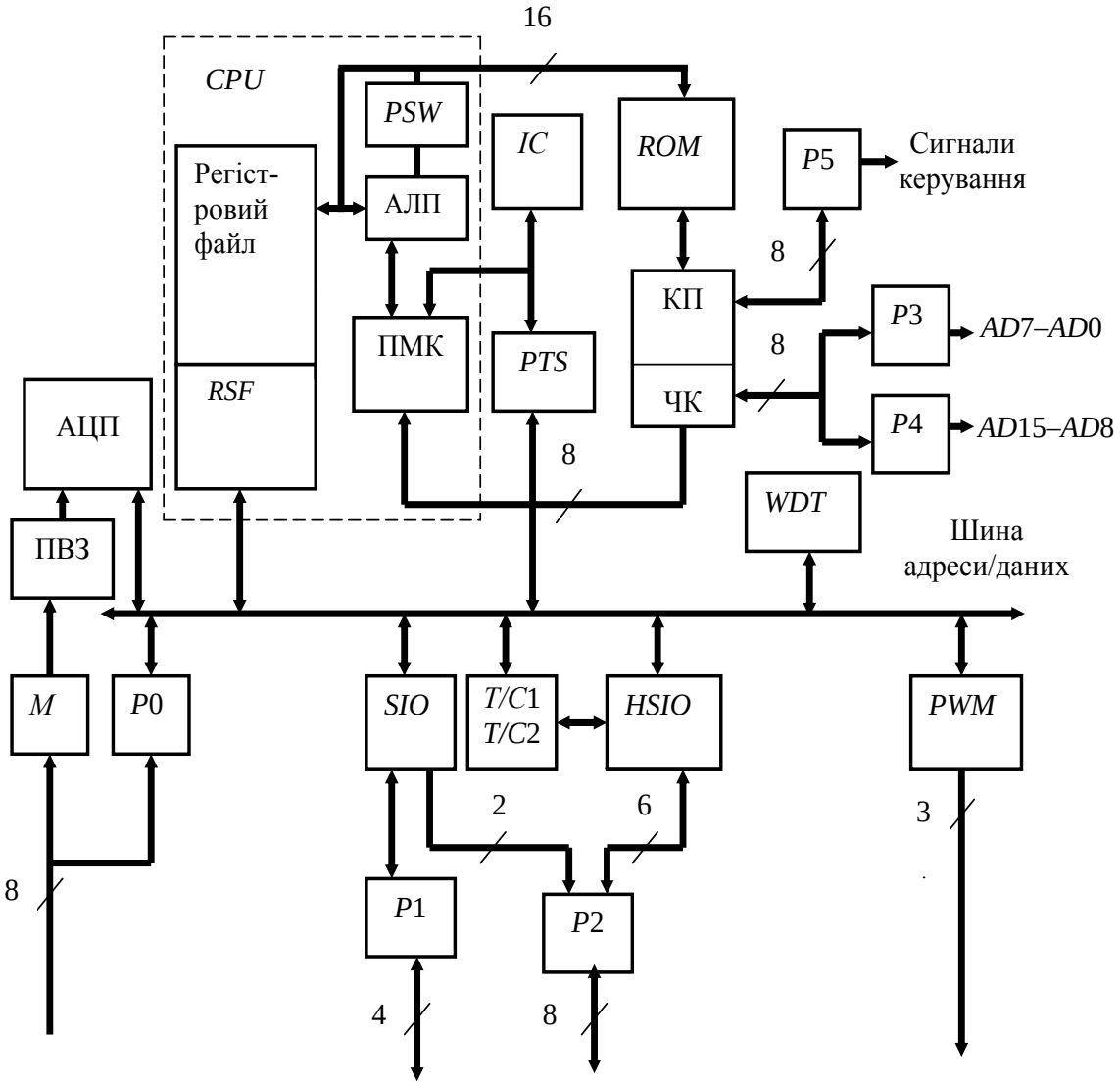


Рис. 7.29. Структурна схема ОМК з інтегрованим пристроєм високошвидкісного введення-виведення: ЧК – черга команд

Модуль ЦП складається з регістрового АЛП, регістра слова стану процесора *PSW*, пристрою мікропрограмного керування і регістрового файла. *Регістровий файл* являє собою масив регістрів - ОЗП статичного типу. Він складається з нижнього та верхнього регістрових файлів. Молодші 24 байт нижнього регістрового файла належать до регістрів спеціальних функцій. У нижньому регістровому файлі розміщений також вказівник стека. Решта комірок регістрового файла, включаючи комірки верхнього регістрового файла, є оперативною пам'яттю загального призначення.

Центральний процесор пов'язаний із контролером пам'яті, контролером переривань і вмонтованими периферійними пристроями за допомогою внутрішньої 16-розрядної шини. Крім того, існує додаткова 8-розрядна шина для безпосередньої передачі байтів команди з контролера пам'яті в регістр команд, який знаходиться в пристрої мікропрограмного керування.

Контролер пам'яті здійснює операції вибірки/запису даних у зовнішню або внутрішню пам'ять даних або команд. До складу контролера пам'яті входить чотирибайтова черга команд. Функціонування контролера пам'яті аналогічне функціонуванню шинного інтерфейсу МП *i*8086 (підрозд. 3.2). Звернення ЦП відбуваються лише до регістрового файла.

Контролер переривань призначений для ефективного керування у реальному часі. Обробка запитів переривань здійснюється з урахуванням установленого рівня пріоритету.

Як запити переривань можуть використовуватися:

– зовнішні сигнали *NMI* або *EXTINT*;
– запити від інтегрованих периферійних пристроїв;
– команди викликів підпрограм обробки переривань.

Сервер периферійних транзакцій виконує функції периферійного процесора введення-виведення і звільняє ЦП від виконання деяких типових операцій введення-виведення. *Транзакцією* називається виконання цілісної операції над даними або пакетами даних, наприклад, введення або виведення пакета даних відповідно до заданого протоколу по послідовному каналу зв'язку, пересилання даних з однієї області пам'яті в іншу, сканування АЦП, тобто зчитування результатів, запис у пам'ять і опитування каналу АЦП. Сервер периферійних транзакцій працює виключно за перериванням (*PTS*-переривання), але обслуговує їх не програмно, а апаратно на мікропрограмному рівні. При цьому підвищується швидкодія обчислень, і деякі *PTS*-переривання можуть оброблятися за час виконання однієї звичайної команди.

Перед тим, як дозволити *PTS*-переривання, необхідно задати: бажаний режим роботи, кількість циклів передачі даних, адреси джерел і приймачів даних. Це здійснюється програмно записуванням необхідної інформації в керувальний блок сервера периферійних транзакцій, що знаходиться в регістровому файлі. По закінченні програми обслуговування

PTS-переривання генерується звичайне переривання для переініціалізації сервера на нове завдання.

Постійний запам'ятовувальний пристрій. Як внутрішня пам'ять програм залежно від типу ВІС використовується програмовний маскою або однократно програмовний ПЗП ємністю від 8 К×8 до 32 К×8 байт.

Блок АЦП призначений для введення аналогових сигналів і складається з аналогового мультиплексора *M* (наприклад, на 8 каналів для ОМК 8*xC*196*KD*), пристрою вибірки/зберігання (ПВЗ) і 8- або 10-розрядного АЦП. За допомогою регістра спеціальних функцій *AD_TIME* задається час вибирання і перетворення, за допомогою регістра *AD_COMMAND* – номер вхідного каналу, для якого необхідно виконати перетворення, та умови запуску процесу перетворення (8 або 10 розрядів АЦП). Якщо використати 10-розрядний АЦП, то результат зчитується з регістра *AD_RESULT* або за опитуванням, або за перериванням (за готовністю АЦП). Час вибірки 10-розрядного АЦП становить 1 мкс, час перетворення – 10–20 мкс у 10-розрядному режимі, 7–20 мкс – у 8-розрядному режимі.

Модуль високошвидкісного введення-виведення призначений для формування періодичних у часі сигналів (наприклад, сигналу ШІМ для прямого цифрового керування інверторами напруги приводів змінного чи постійного струму) та безпосереднього вимірювання часових інтервалів між зовнішніми подіями без додаткових периферійних пристроїв (наприклад, вимірювання частоти, періоду, фазового зсуву). *Подією* вважається будь-яка зміна потенціалу зовнішнього сигналу на входах *HSI*.0–*HSI*.3 ОМК.

До складу модуля входять:

– два таймери (*T/C*1 і *T/C*2) для точного відліку часових інтервалів;
– модуль високошвидкісного введення даних *HSI*;
– модуль високошвидкісного виведення даних *HS*O.

Модуль високошвидкісного введення апаратно контролює зміну потенціалу зовнішнього сигналу на кожному із чотирьох входів *HSI*.0–*HS*I.3 за допомогою вбудованої схеми детектора перепадів. Ці зміни сприймаються ОМК як зовнішні події. Кожний із входів *HSI*.0–*HSI*.3 можна незалежно запрограмувати на визначення таких подій:

– кожний перехід значення вхідного сигналу з логічної одиниці в логічний нуль;
– кожний перехід значення вхідного сигналу з логічного нуля в логічну одиницю;
– кожний перехід значення вхідного сигналу з логічної одиниці в логічний нуль та з логічного нуля в логічну одиницю;
– вісім переходів значення вхідного сигналу з логічної одиниці в логічний нуль.

Якщо запрограмована подія відбулася на одному або декількох входах, то час настання цієї події, відрахований таймером 1, разом з інформацією про стан усіх входів записується у стек подій *FIFO*. Тобто відбувається

захоплення події, яке супроводжується генерацією запиту переривання ЦП на обслуговування модуля *HSI*. Якщо до захоплення події стек подій був порожнім, то запис здійснюється в регістр зберігання (*Holding Register*) інформації про подію, що відбулася. В іншому разі запис здійснюється у стек. Модуль *HSI* може послідовно запам'ятовувати інформацію про максимум вісім подій: сім записів можуть зберігатися у стеку *FIFO* ємністю 7×20 біт та один запис – у регістрі зберігання.

Основне завдання модуля високошвидкісного виведення полягає в генерації заданих подій у заданий час. Існують внутрішні та зовнішні події. Внутрішні події, не пов'язані зі зміною потенціалу на виводах ОМК, використовуються, наприклад, для запуску таких процесів:

– перетворення у вбудованому АЦП у заданий момент часу;
– обслуговування програмнореалізованих таймерів (*Software Timer*), які використовуються для відліку заданих досить великих часових інтервалів. Одночасно може обслуговуватися до чотирьох таких таймерів.

Інформація про подію і час генерації події у вигляді команди записується програмно в асоціативну пам'ять *Content Addressable Memory* (*CAM*-файл). Ємність *CAM*-файла дозволяє одночасно запам'ятовувати до 8 команд, формуючи таким чином деяку програму генерації подій. Кожна з цих команд задає:

– тип події (скидання або встановлення потенціалу на зовнішньому контакті, запуск АЦП, запуск ПТ);
– спосіб реагування ЦП на подію – інформація про те, чи буде генеруватися переривання для обслуговування події;
– номер таймера (1 чи 2), використовуваний для контролю поточного реального часу;
– час генерації події.

У кожному машинному такті (100 нс при тактовій частоті 20 МГц) відбувається порівняння часу генерації події з реальним часом, що відраховує таймер. Якщо вони збігаються, то генерується задана командою подія. Якщо подія відбулася, вона може бути вилучена зі списку подій або перезаписана знову.

Вбудований триканальний ШІМ-генератор. Модуль *HSIO* здатний реалізувати функцію ШІМ, але він має певні обмеження, пов'язані з необхідністю витрат часу звернення ЦП на його обслуговування. Це призводить до обмеження максимального та мінімального коефіцієнтів заповнення ШІМ-сигналів. Тому для генерації ШІМ-сигналів на високих частотах (до 40 кГц) більш зручним є вбудований триканальний ШІМ-генератор, який не має обмежень на коефіцієнт заповнення імпульсів. Частота ШІМ для усіх каналів є однаковою, а коефіцієнт заповнення регулюється від 1/255 до 1 з дискретністю 1/255. Недоліком ШІМ-генератора є обмежені можливості регулювання частоти. Так, для ВІС 8*xC*196*KD* при тактовій частоті 20 МГц частота ШІМ може встановлюватися в одне з двох значень – 39,1 і 19,5 кГц.

Вбудований ШІМ-генератор ефективно використовують для прямого цифрового керування електронними комутаторами, ключами інверторів напруги та струму, а також для формування аналогових сигналів задання у цифро-аналогових системах керування.

Послідовний порт *SIO* являє собою УСАПП, який може настроюватися на один із чотирьох режимів, один з яких є синхронним, а три – асинхронними. Швидкість обміну інформацією по послідовному порту програмно регулюється від кількох сотень бод (біт/с) до кількох десятків тисяч бод. Порт забезпечує роботу в режимі *Master-Slave*, що дає можливість використати мікроконтролер у мультимікропроцесорній системі керування.

Однокристальний мікроконтролер з інтегрованими процесорами подій. 16-розрядний ОМК з інтегрованим процесором подій або *EPA*-мікроконтролер (*Event Processor Array* – масив процесорів подій) відрізняється від ОМК із вбудованим модулем *HSIO* наявністю потужного процесора подій, модифікованим блоком вбудованих таймерів/лічильників, більшою ємністю прямо адресованої зовнішньої пам'яті (від 1 Мбайт до 16 Мбайт), розширеними інтерфейсними можливостями.

Модифікація блока вбудованих таймерів/лічильників полягає в такому:

– обидва таймери є реверсивними і можуть працювати як із внутрішнім, так і з зовнішнім тактуванням;
– є можливість каскадного вмикання, коли таймер *T/C*2 тактується імпульсами переповнення від таймера *T/C*1;
– обидва таймери допускають роботу в квадратурному режимі, коли дві послідовності прямокутних імпульсів, зсунутих на 90 електричних градусів, знімаються з датчика і вводяться безпосередньо в мікроконтролер. Цей режим зручний для роботи з датчиками положення і/або швидкості (наприклад, з датчиками Хола тощо).

Масив процесора подій сумісний знизу з модулем високошвидкісного введення-виведення і призначений для вирішення тих самих завдань, але має деякі вдосконалення. Кожний канал процесора подій працює в одному з двох режимів – режимі захоплення зовнішніх подій або режимі генерації внутрішніх чи зовнішніх подій. Як базовий для кожного з каналів може використовуватись один з таймерів 1 або 2, який тактується за одним з таких способів:

– зовнішнє тактування за допомогою входу зовнішніх тактових імпульсів і входу напряму лічби;
– зовнішнє тактування двома послідовностями імпульсів у квадратурному режимі;
– внутрішнє тактування з можливістю зміни частоти вхідних тактових імпульсів через програмування коефіцієнта ділення;
– тактування сигналом переповнення іншого таймера.

Масив процесорів подій реагує на такі типи зовнішніх подій:

– перехід сигналу на зовнішньому виводі з логічної одиниці в логічний нуль;

– перехід сигналу на зовнішньому виводі з логічного нуля в логічну одиницю;
– перемикання рівня сигналу на зовнішньому виводі.

Кожний канал процесора подій генерує окремий запит на переривання, що значно спрощує та прискорює обслуговування процесора подій з боку ЦП, оскільки не потребує ідентифікації джерела запиту.

Оскільки виконання функцій захоплення/порівняння кожним каналом здійснюється незалежно, часова дискретність визначення події або генерації сигналів збільшується в кілька разів. Так, при частоті 20 МГц часова дискретність *EPA*-мікроконтролера становить 200 нс, що у 4 рази менше, ніж у *HSIO*-мікроконтролера з тією самою тактовою частотою.

Однокристальний мікроконтролер з інтегрованими засобами керування двигунами або *MCF*-мікроконтролер (*Motor Control Family*). Він призначений для прямого цифрового керування перетворювальними пристроями електроприводів. Структурну схему *MCF*-мікроконтролера серії 8*C*196*MH* показано на рис. 7.30.

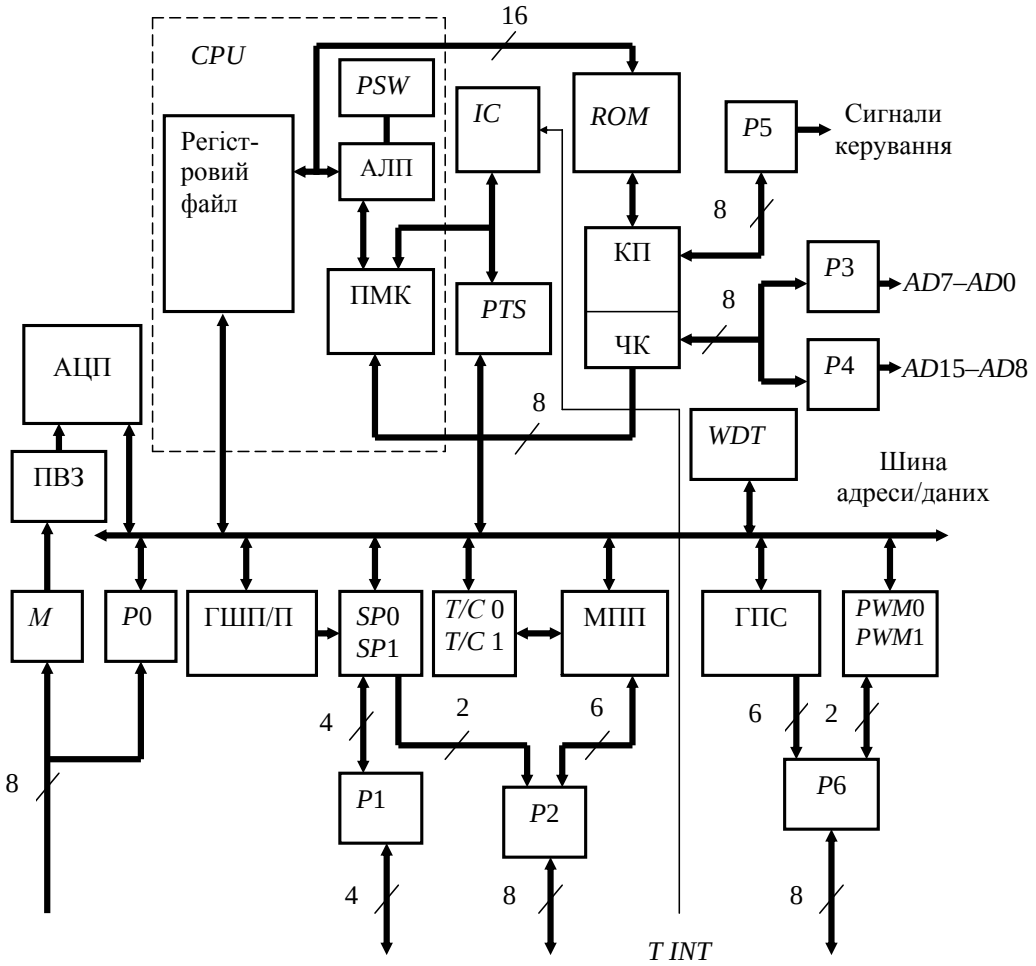


Рис. 7.30. Структурна схема ОМК з інтегрованими засобами керування двигунами: ГШП/П – генератор швидкості прийому/передачі; ГПС – генератор періодичних сигналів; МПП – масив процесора подій (*EPA*)

Крім розглянутих вище блоків ОМК (див. рис. 7.29), схема містить додаткові лінії портів та більш складний генератор періодичних сигналів. Лінії 7 портів ОМК мають таке функціональне призначення:

– порт 0 – функція АЦП;
– порт 1 – чотири лінії послідовного введення-виведення;
– порт 2 – дві лінії послідовного введення-виведення, два канали захоплення/порівняння, чотири канали порівняння процесора подій;
– порти 3 і 4 – суміщена шина адреси/даних;
– порт 5 – вироблення сигналів керування;
– порт 6 – функція генератора періодичних сигналів.

Генератор періодичних сигналів містить:

– блок цифрового опорного генератора (*Timebase Generator*), який задає пилоподібну або трикутну форму опорного цифрового сигналу модулятора;
– блок триканального цифрового компаратора – драйвера формування сигналів фаз (*Phase Driver*), у якому на базі заданого значення коефіцієнта заповнення формується три незалежні ШІМ-сигнали: *WG*1, *WG*2, *WG*3. Цей блок формує також три інверсні сигнали: *WG*1#, *WG*2#, *WG*3# з урахуванням затримки ввімкнення для уникнення режиму наскрізного струму;
– блок керування виходами генератора періодичних сигналів (*Output Control*), який дозволяє програмно ввімкнути чи вимкнути будь-який із сигналів *WGn* або *WG#n*, призначити для виходів генератора активний рівень (*L* або *H*), перевести за менш ніж 2 мкс виходи генератора в пасивний стан при надходженні сигналу аварії, вимкнути всі ШІМ-виходи програмно.

Генератор періодичних сигналів дозволяє в широких межах регулювати частоту і період ШІМ (від 0,25 мкс до 16 мс для односторонньої ШІМ і від 0,125 мкс до 8 мс – для двосторонньої). Час затримки для усунення наскрізного струму становить 0,125–125 мкс.

Однокристальний мікроконтролер з інтегрованими засобами цифрової обробки. ОМК *MCS*296 має традиційну архітектуру *Intel* процесорів обробки подій у реальному масштабі часу, але додатково містить цифровий сигнальний процесор *Digital Signal Processor* (*DSP*) для реалізації функції цифрового регулятора. Таке поєднання різних типів процесорів дозволило отримати високі технічні характеристики конторолерів, а саме:

– підвищення продуктивності ЦП за рахунок зменшення часу вибірки з пам’яті програм і даних;
– використання конвеєрного режиму з одночасним виконанням чотирьох команд для зменшення часу машинного циклу;
– застосування апаратних засобів множення та ділення;

– доповнення системи команд 17 спеціальними командами (наприклад, множення з накопиченням, повторення, автоматизація оновлення даних у таблицях послідовних вибірок вхідних сигналів у процесі їх обробки), оптимізованими для вирішення задач побудови цифрових фільтрів і регуляторів.

Контрольні запитання

1. Дайте визначення події, захоплення події та транзакції.
2. Яку інформацію містить регістровий файл у ОМК з модулем *HSIO*?
3. Укажіть призначення та поясніть принцип дії сервера периферійних транзакцій.
4. Укажіть призначення та поясніть принцип дії модуля високошвидкісного введення-виведення.
5. Опишіть принцип роботи вбудованого АЦП в ОМК з модулем *HSIO*.
6. Порівняйте характеристики послідовного порту ОМК з ВІС КР580ВВ51.
7. Порівняйте характеристики блока таймерів *EPA*- і *HSIO*-мікроконтролерів.
8. Поясніть особливості квадратурного режиму роботи таймера.
9. Порівняйте характеристики генератора періодичних сигналів мікроконтролера з модулем *HSIO*- і *MCF*-мікроконтролерів.

Розділ 8

ОДНОКРИСТАЛЬНІ МІКРОКОНТРОЛЕРИ З *RISC*-АРХІТЕКТУРОЮ

Розглянуті в сьомому розділі *CISC*-мікроконтролери характеризуються досить розвиненою системою, наприклад, мікроконтролери серії *i*80*x*51 мають 111 команд. Однак аналіз програм показав, що 20 % команд використовують в 80 % випадків, а дешифратор команд займає більше 70 % площі кристала. Тому в розробників МП виникла ідея скоротити кількість команд, надати їм єдиний формат і зменшити площу кристала, тобто використати *RISC*-архітектуру *(Reduced Instruction Set Computer)*.
Особливість контролерів, виконаних за *RISC*-архітектурою, у тому, що всі команди виконуються за один – три такти, тоді як у *CISC*-контролерах – за один – три машинні цикли, кожний з яких складається з декількох тактів (наприклад, для *i*80*x*51 – з 12 тактів). Тому *RISC*-контролери мають значно більшу швидкодію. Але повніша система команд *CISC*-контролерів у деяких випадках економить час виконання окремих фрагментів програми і пам'ять програм.

8.1. *PIC*-контролери

Типові представники *RISC*-процесорів – *PIC*-контролери (*Periferial Interface Controller* – контролери периферійних інтерфейсів) виробництва фірми *MicroChip*. *PIC*-контролери застосовують у системах високошвидкісного керування автомобільними й електричними двигунами, приладах побутової електроніки, телефонних приставках з АВН, системах охорони із сповіщенням по телефонній лінії, міні-АТС. Окремі ВІС різняться розрядністю ПЗП: від 12 до 14 біт для серії *PIC*16*Cxx*, 16 біт – для серії *PIC*17*Cxx*. Завдяки скороченій кількості (від 33 до 35) усі команди займають у пам'яті одне слово. Час виконання кожної команди, крім команд розгалуження, становить чотири такти – один цикл (200 нс на частоті 20 МГц). ОЗП виконано за схемою з довільною вибіркою з можливістю безпосередньої адресації в коді команди до будь-якої комірки. Стек реалізовано апаратно з ємністю 2, 8 або 16 комірок. Майже в усіх *PIC*-контролерах є система переривань, джерелом яких може бути

таймер, а також зміна станів сигналів на деяких входах. У *PIC*-контролерах передбачено біт захисту ПЗП, що запобігає нелегальному копіюванню.

Основні характеристики різних типів *PIC*-контролерів наведено в табл. 8.1.

Великі інтегральні схеми *PIC*16*Cxx* мають вбудовані ПЗП ємністю від 0,5 до 4 кілослів і ОЗП ємністю 32–256 байт. Основна частина контролерів має однократно програмовний ПЗП, однак деякі контролери містять ПЗП з ультрафіолетовим стиранням, а *PIC*16*C*84 містить пам'ять програм і пам'ять даних на базі ПЗП з електричним стиранням. Крім того, контролери мають від одного до трьох таймерів, вбудовану систему скидання, *watchdog*-таймер, внутрішній тактовий генератор, який може запускатися як від кварцового резонатора, так і від *RC*-кола в широкому діапазоні частот – 0–25 МГц. Кількість розрядів портів – 12–33. Кожний розряд порту можна запрограмувати на введення або на виведення. Контролер *PIC*16*C*64 додатково має вихід із ШІМ, за допомогою якого можна реалізувати ЦАП з розрядністю до 16 розрядів, а також послідовний двонапрямлений синхронний порт з інтерфейсами *SP*I, I2*C*, *SCI*/*UART*. *PIC*16*C*71 і *PIC*16*C*74 мають внутрішній 8-розрядний АЦП із пристроєм вибирання/зберігання і вхідним аналоговим мультиплексором.

Контролери *PIC*17*Cxx* мають вбудований апаратний 8-розрядний помножувач, два виходи ШІМ з роздільною здатністю 1–10 біт, два виводи з відкритим колектором, чотири таймери/лічильники, 11 джерел переривань, у них передбачено можливість виконання програми із зовнішнього ПЗП.

Контролер *PIC*1400 має програмовний вибір генератора – вбудованого 4 МГц резонатора або зовнішнього кварцового чи керамічного резонатора, сторожовий таймер з окремим вбудованим *RC*-генератором, внутрішньосхемним програмуванням через два виводи, два режими зниженого енергоспоживання (200 мкА при 3 В, якщо генератор вимкнено, а аналогові схеми активні і 5 мкА при 3 В, якщо вимкнено генератор і аналогові схеми).

Процесор *PIC*1400 містить інтегральний АЦП на 8 каналів, діапазон напруги двох з яких може задаватися програмно. Час перетворення АЦП складає 16 мс при тактовій частоті 4 МГц і роздільній здатності 16 біт. Цей контролер містить також 4-розрядний струмовий ЦАП, вбудований датчик температури з роздільною здатністю 0,10 В та вбудований детектор зниженої напруги живлення.

Таблиця 8.1. **Основні технічні характеристики *PIC*-процесорів**

Тип *PIC*	Тактова частота, МГц	Пам'ять програм, біт		Пам'ять даних, байт	*EEPROM* пам'ять даних, байт	Глибина стека	Таймер 0 (8 + 8 біт)	Таймер 1 (16 біт)	Таймер 2 (8 біт)	Кількість ШІМ-виходів	Послідовний порт/ інтерфейс	Кількість каналів АЦП	Кількість запитів переривань	Кількість розрядів введення-виведення	Кількість контактів ВІС
		EPROM	*EEPROM*												
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
16*C*54	16	512×12	–	36	–	2	+	–	–	–	–	–	–	13	18
16*C*55	16	512×12	–	36	–	2	+	–	–	–	–	–	–	21	28
16*C*56	16	1024×12	–	72	–	2	+	–	–	–	–	–	–	13	18
16*C*57	16	2048×12	–	72	–	2	+	–	–	–	–	–	–	21	28
16*C*58	16	2048×12	–	80	–	8	+	–	–	–	–	–	–	13	18
16*C*61	20	–	1024×14	36	–	8	+	–	–	–	–	–	3	13	18
16*C*62	20	2048×14	–	128	–	8	+	+	+	1	*SPI/I2C, SCI/UART*	–	10	21	28
16*C*620	20	–	512×14	80	–	8	+	–	–	–	–	–	4	13	18
16*C*621	20	–	1024×14	80	–	8	+	–	–	–	–	–	4	13	18
16*C*622	20	–	2048×14	128	–	8	+	–	–	–	–	–	4	13	18
16*C*63	20	4096×14	–	192	–	8	+	+	+	1	*SPI/I2C, SCI/UART*	–	+	21	28

Продовження табл. 8.1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
16*C*64	25	2048×14	–	128	–	8	+	+	+	1	*SPI/I2C, SCI/UART*	–	8	21	40
16*C*65	20	4096×14	–	192	–	8	+	+	+	2	*SPI/I2C, SCI/UART*	–	8	21	40
16*C*710	20	–	1512×14	36	–	8	+	–	–	–	–	4	3	13	18
16*C*71	16	–	1024×14	36	–	8	+	–	–	–	–	4	3	13	18
16*C*711	20	–	1024×14	68	–	8	+	–	–	–	–	4	3	13	18
16*C*72	20	2048×14	–	128	–	8	+	+	+	1	*SPI/I2C, SCI/UART*	5	10	21	28
16*C*73	20	4096×14	–	192	–	8	+	+	+	2	*SPI/I2C, SCI/UART*	5	10	21	28
16*C*74	20	4096×14	–	192	–	8	+	+	+	2	*SPI/I2C, SCI/UART*	8	8	21	40
16*C*83	10	512×14	64×8	64	36×8	8	+	–	–	–	–	–	–	13	18
16*C*84	10	1024×14	64×8	64	36×8	8	+	–	–	–	–	–	–	13	18
16*C*84*A*	10	1024×14	64×8	64	68×8	8	+	–	–	–	–	–	–	13	18
1400	20	4096×14	–	192	–	8	+	+	–	–	*SPI/I2C,*	8	–	21	28
17*C*42	25	–	2048×16	232	–	16	+	+	+	2	*SCI/UART*	–	+	32	40
17*C*43	25	–	4096×16	454	–	16	+	+	+	2	*SCI/UART*	–	+	32	40
17*C*44	25	–	8192×16	454	–	16	+	+	+	2	*SCI/UART*	–	+	32	40

Архітектура *PIC*-контролерів. Архітектуру *PIC*-контролерів розглянемо на прикладі ВІС *PIC*16*C*71 (рис. 8.1).

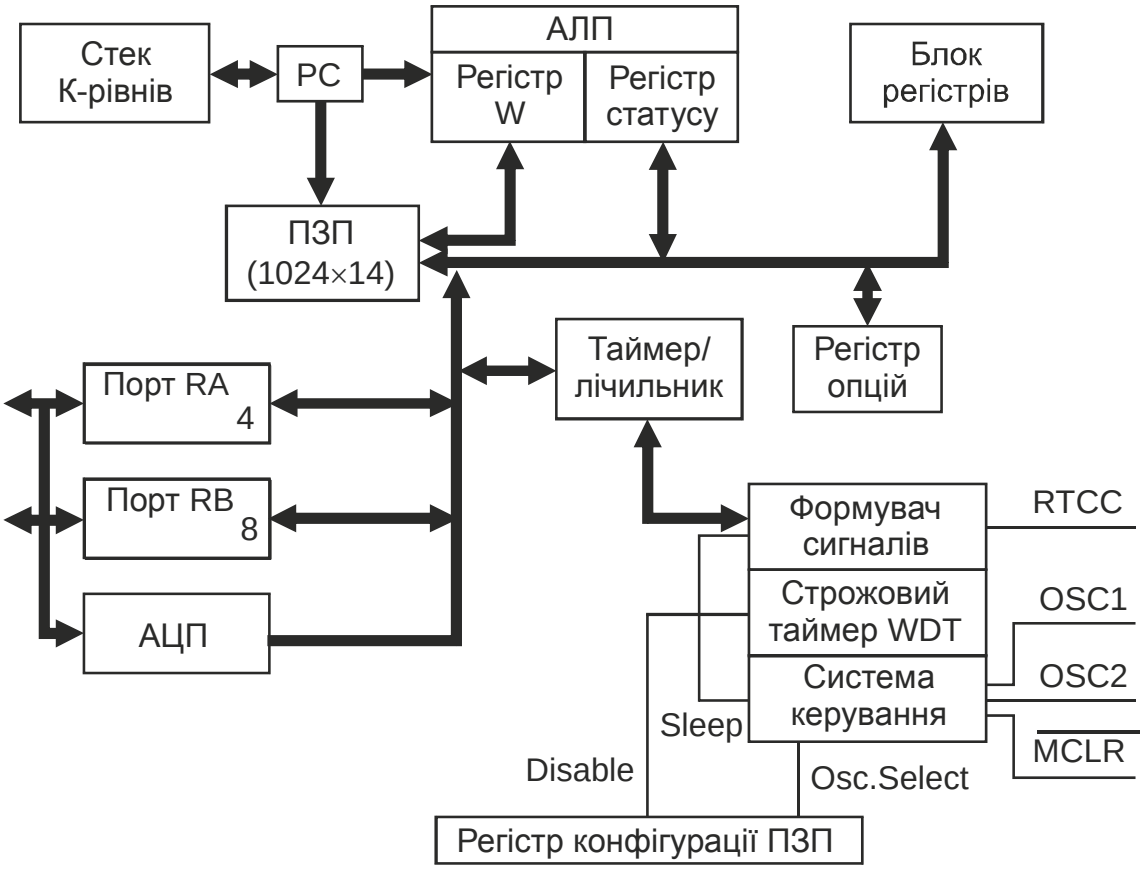


Рис. 8.1. Структурна схема контролера *PIC*16*C*71

Основа архітектури – роздільні шини та області пам’яті для даних і команд. Шина даних і комірка ОЗП – 8-розрядні, а шина команд і програмна пам’ять (ПЗП) – 14-розрядні; 14-розрядна команда вибирається за один цикл. Двоступеневий конвеєр забезпечує одночасне вибирання і виконання команди. Система команд містить 35 команд. Усі команди виконуються за один цикл, за винятком команд переходів, що виконуються за два цикли.

Структурна схема контролера містить:

– восьмирівневий апаратний стек;
– 13-розрядний програмний лічильник *PC*;
– 8-розрядний АЛП;
– ОЗП, який складається з 36 8-розрядних РЗП;
– 15 регістрів спеціальних функцій *SFR* (на рис. 8.1 показано регістри непрямої адресації *W*, статусу, опцій, конфігурації ПЗП);
– 8-розрядний таймер/лічильник з 8-розрядним програмовним переддільником;
– модуль АЦП із чотирма входами;

– 13 ліній введення-виведення (4-розрядний порт *RA*, 8-розрядний порт *RB*, лінія *RTCC*);
– сторожовий таймер *WDT*;
– формувач зовнішнього сигналу *RTCC* або сигналу сторожового таймера;
– систему керування і синхронізації з внутрішнім генератором.

Призначення виводів ВІС наведено у табл. 8.2, а граничні електричні параметри – у табл. 8.3.

Таблиця 8.2. **Призначення виводів *PIC*16*C*71**

Номер виводу	Позначення	Функціональне призначення
1	*RA*2/*AIN*2	Двонапрямлена лінія введення-виведення. Аналоговий вхід каналу 2. Як цифровий вхід має рівні ТТЛ
2	*RA*3/*AIN*3/*Uref*	Двонапрямлена лінія введення-виведення. Аналоговий вхід каналу 3
3	*RA*4/*RTCC*	Вхід через тригер Шмітта. Лінія порту введення-виведення з відкритим стоком або вхід частоти для таймера/лічильника *RTCC*
4	$\overline{MCLR}$ /*Upp*	Сигнал скидання для контролера. Активний низький рівень. Вхід через тригер Шмітта
5	*Uss*	Загальний вивід (земля)
6	*RB*0/*INT*	Двонапрямлена лінія введення-виведення або зовнішній вхід переривання
7	*RB*1	Двонапрямлена лінія введення-виведення
8	*RB*2	Те саме
9	*RB*3	– “ –
10	*RB*4	– “ –
11	*RB*5	– “ –
12	*RB*6	– “ –
13	*RB*7	– “ –
14	*Udd*	Напруга живлення
15	*OSC*2/*CLKOUT*	Вхід увімкнення кварцового резонатора в усіх режимах, крім *RC*-генератора (вихід тактової частоти в режимі *RC*-генератора)
16	*OSC*1/*CLKIN*	Вхід увімкнення кварцового резонатора, *RC*-кола (вхід зовнішньої тактової частоти)
17	*RA*0/*AIN*0	Двонапрямлена лінія введення-виведення Аналоговий вхід каналу 0
18	*RA*1/*AIN*1	Двонапрямлена лінія введення-виведення. Рівні ТТЛ. Аналоговий вхід каналу 1

Таблиця 8.3. **Граничні значення електричних параметрів**

Параметр	Граничне знач.
Інтервал робочих температур	Від –55 до +125 С°
Температура зберігання	Від –65 до +150 С°
Напруга на будь-якому виводі відносно U_{ss} (землі), крім U_{dd} $\overline{MCLR}$	–0,6–U_{dd} + 0,6 В
Напруга U_{dd} відносно U_{ss}	0–7,5 В
Напруга $\overline{MCLR}$ відносно U_{ss}	0–14 В*
Загальна потужність розсіювання	800 мВт
Максимальний струм у виводі U_{ss}	150 мА
Максимальний струм у виводі U_{dd}	100 мА
Максимальний струм у будь-якому виводі	±500 мкА
Максимальний вхідний струм будь-якої лінії порту *RA* або *RB*	25 мА
Максимальний вихідний струм будь-якої лінії порту *RA* або *RB*	20 мА
Максимальний сумарний вхідний струм для всіх ліній порту *RA*	50 мА
Максимальний сумарний вхідний струм для всіх ліній порту *RB*	100 мА
Максимальний сумарний вихідний струм для всіх ліній порту *RA*	80 мА
Максимальний сумарний вихідний струм для всіх ліній порту *RB*	150 мА

* Сигнал на вивід $\overline{MCLR}$ рекомендується подавати через обмежувальний резистор 50–100 Ом.

Як видно з наведених характеристик, *PIC*-контролери за своїми параметрами конкурують з однокристальними мікроЕОМ та ОМК. Деякі модифікації *PIC*-контролерів мають більшу швидкодію ніж ОМК. І *PIC*-контролери, і ОМК застосовують у вбудованих системах керування різного обладнання.

Контрольні запитання

1. Назвіть сфери застосування *PIC*-контролерів.
2. Якими факторами визначається швидкодія *PIC*-контролерів?
3. Назвіть основні блоки структурної схеми *PIC*-контролера.
4. Порівняйте характеристики *PIC*-контролерів серій *PIC*16*Cxx* і *PIC*17*Cxx*.
5. Укажіть типи ПЗП, які використовують у *PIC*-контролерах.
6. Які периферійні пристрої застосовують у *PIC*-контролерах?

8.2. Однокристальні *AVR*-мікроконтролери

Однокристальні *AVR*-мікроконтролери (*AVR*-МК) являють собою 8-розрядні високопродуктивні *RISC*-контролери загального призначення. Їх створила група розробників дослідного центру фірми *Atmel Corp.* (Норвегія), ініціали яких сформували марку *AVR*. Особливість *AVR*-МК – їх широка номенклатура, що дозволяє користувачеві вибрати мікроконтролер із мінімальною апаратною надмірністю і, отже, найменшої вартості. Так, наприклад, у номенклатуру групи *AT*90*S* входять прилади з ПЗП ємністю від 1 до 8 кбайт, з різним набором периферійних пристроїв й у корпусах із кількістю виводів від 8 до 48. Нині в серійному виробництві знаходяться три сім'ї *AVR – Tiny*, *Classic* і *Mega*. Мікроконтролери *Tiny* – найбільш дешеві і прості, *Mega* – найбільш потужні, а *Classic* займають проміжне місце між ними.

Розглянемо архітектуру *AVR*-МК на прикладі мікроконтролерів сім'ї *Classic* – *AT*90*S*8535 (рис. 8.2)

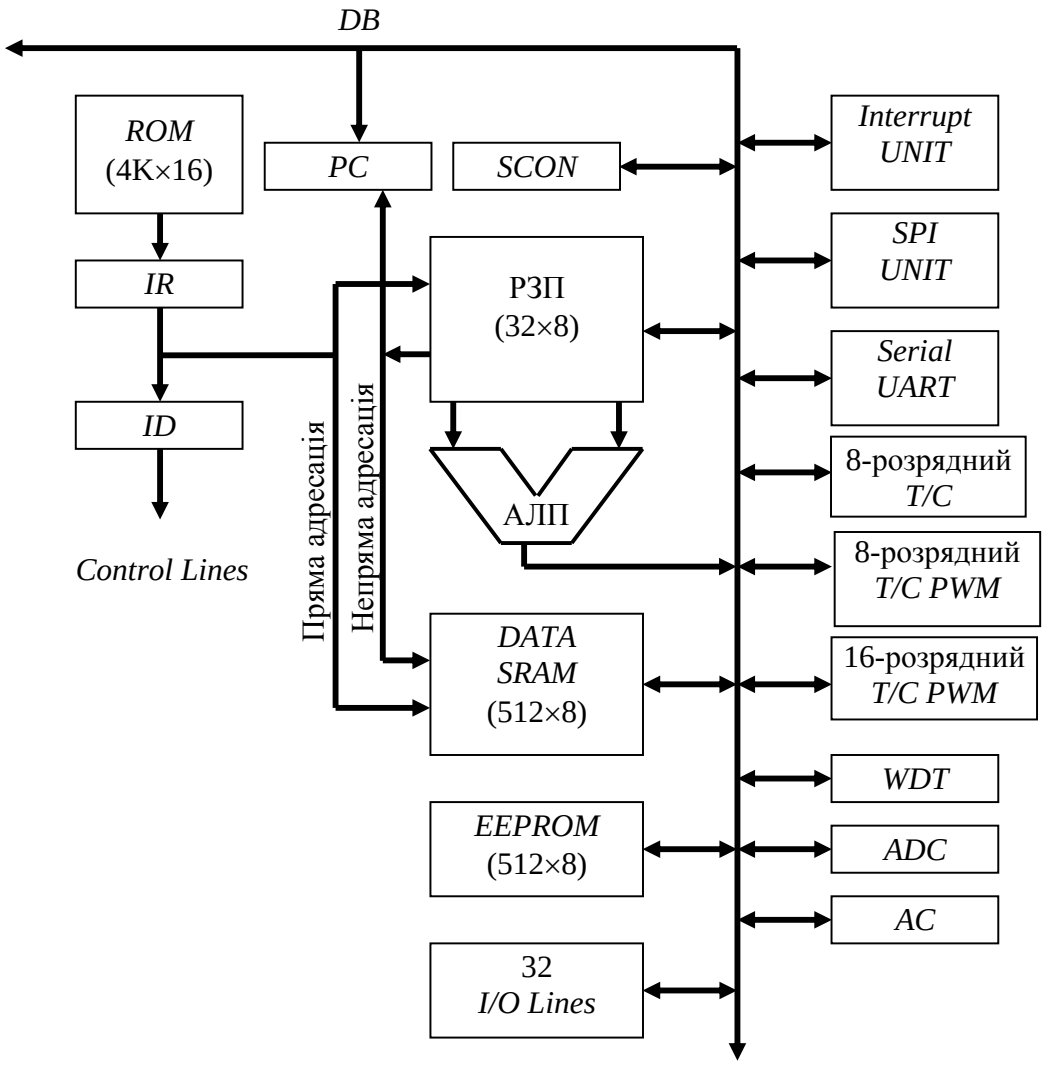


Рис. 8.2. Архітектура мікроконтролера *AT*90*S*8535

Мікроконтролер містить гарвардський процесор, регістровий файл, пам’ять програм, пам’ять даних і різні інтерфейсні схеми (периферію).

Гарвардський процесор. Реалізує повний логічний і фізичний розподіл не тільки адресних просторів, але й інформаційних шин для звертання до пам’яті програм і до пам’яті даних, причому способи адресування і доступу до цих масивів пам’яті також різні.

Така будова вже ближча до структури цифрових сигнальних процесорів і забезпечує істотне підвищення продуктивності. Процесор працює одночасно як із пам’яттю програм, так і з пам’яттю даних; розрядність шини пам’яті програм розширена до 16 біт. В *AVR*-МК використовують технологію конвеєризації, унаслідок чого цикл *вибірка – виконання команди* помітно скорочений. Для порівняння, у мікроконтролерів сім’ї *MCS*51 коротка команда виконується за 12 тактів генератора – один машинний цикл, протягом якого процесор послідовно зчитує код операції і виконує її. У *PIC*-контролерах фірми *MicroChip*, де вже реалізований конвеєр, коротка команда виконується протягом 8 періодів тактової частоти (два машинні цикли). За цей час послідовно дешифрується і зчитується код операції, виконується команда, фіксується результат і одночасно зчитується код наступної операції (однорівневий конвеєр). Тому в загальному потоці команд одна коротка команда реалізується за 4 періоди тактової частоти або за один машинний цикл. В *AVR*-МК теж використовують однорівневий конвеєр під час звертання до пам’яті програм і короткі команди в загальному потоці виконуються, як і в *PIC*-контролерах, за один машинний цикл. Головна ж відмінність у тому, що цей цикл в *AVR*-МК становить лише один період тактової частоти.

Регістровий файл. Займає молодші 32 байт у загальному адресному просторі *SRAM AVR* (рис. 8.3). Шість із 32 регістрів файла можна використовувати як три 16-розрядні покажчики адреси у процесі непрямого адресування даних. Один із цих покажчиків (*Z Pointer*) застосовують також для доступу до даних, записаних у пам’яті програм мікроконтролера. Використання трьох 16-розрядних покажчиків (*X, Y і Z Pointers*) істотно підвищує швидкість пересилання даних під час роботи прикладної програми.

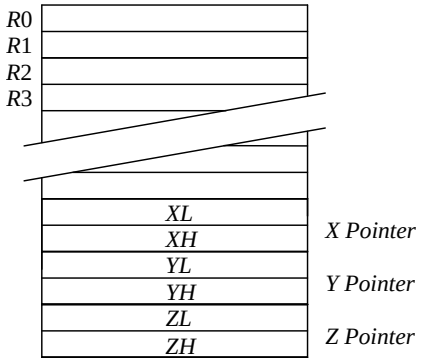


Рис. 8.3. Регістровий файл

Пам'ять програм. Усі *AVR*-МК мають *Flash*-пам'ять програм, яку можна завантажити як за допомогою звичайного програматора, так і за допомогою *SPI*-інтерфейсу, зокрема безпосередньо на цільовій платі. Кількість циклів перезапису – не менше 1000. Деякі версії кристалів сім'ї *Mega* мають можливість самопрограмування, тобто мікроконтролер здатний самостійно, без зовнішнього програматора, змінювати вміст комірок пам'яті програм. Це дає можливість записати в зовнішню енергонезалежну пам'ять декілька робочих версій програми, а потім у міру потреби або за реакцією на зовнішні (внутрішні) логічні умови перевантажувати робочі програми в той самий *AVR*-МК без витягування його з друкарської плати. Для цього весь масив пам'яті програм ділиться на дві нерівні за ємністю ділянки: блок завантажника (програма, що керує перезаписом *Flash*-пам'яті програм) і блок для розміщення робочих програм. Програму-завантажник створює сам розробник, вона має бути запрограмована зовнішнім програматором.

Пам'ять даних. Усі *AVR*-МК мають також блок енергонезалежної пам'яті даних з електричним стиранням *EEPROM.* Цей тип пам'яті використовують для зберігання проміжних даних, різних констант, таблиць перекодувань, каліброваних коефіцієнтів тощо. Дані в *EEPROM* можна завантажити як через *SPI*-інтерфейс, так і за допомогою звичайного програматора. Кількість циклів перезапису – не менше 100 000. Два програмовні біти захисту інформації дозволяють захистити пам'ять програм і енергонезалежну пам'ять даних *EEPROM* від несанкціонованого зчитування.

Внутрішня оперативна пам'ять *SRAM* є у всіх *AVR* сімей *Classic* і *Mega* та в одного нового кристала сім'ї *Tiny* – *ATiny26/L.* До деяких мікроконтролерів можна підключити зовнішню пам'ять даних ємністю до 64 кбайт.

Периферія AVR-МК. До старих периферійних пристроїв належать 8-розрядні порти введення-виведення, послідовний порт, таймери/лічильники, контролер переривань.

Кількість незалежних ліній портів введення-виведення – від 3 до 53. Кожний розряд порту можна запрограмувати на введення або на виведення інформації. Потужні вихідні драйвери забезпечують струмову навантажувальну спроможність 20 мА на лінію порту при максимальному значенні 40 мА, що дозволяє, наприклад, безпосередньо підключати до мікроконтролера світлодіоди і біполярні транзистори. Загальне струмове навантаження на всі лінії одного порту не має перевищувати 80 мА (усі значення наведено для напруги живлення 5 В).

Мікроконтролери *AVR* мають у своєму складі від 1 до 4 таймерів/лічильників загального призначення з розрядністю 8 або 16 біт, що можуть працювати і як таймери від внутрішнього джерела опорної частоти, і як лічильники зовнішніх подій із зовнішнім тактуванням.

Таймер/лічильник *RTC* (є в усіх мікроконтролерах сім'ї *Mega* і в деяких МК *Classic*) реалізує систему реального часу. Таймер має власний переддільник, який можна програмним способом підключити або до основного внутрішнього джерела тактової частоти мікроконтролера, або до додаткового асинхронного джерела опорної частоти (наприклад, зовнішнього генератора). Для цього МК має два зовнішні виводи. Внутрішній осцилятор, навантажений на лічильний вхід таймера/лічильника *RTC*, оптимізовано для роботи із зовнішнім «годинниковим» кварцовим резонатором 32,768 кГц.
Нові периферійні пристрої – блок послідовного периферійного інтерфейсу *(SPI),* сторожовий таймер *(WDT)* і аналоговий компаратор *(AC).*

Блок *SPI* призначено для послідовного введення і виведення даних, його використовують для програмування мікроконтролера після установки на друкарську плату.

Сторожовий таймер *WDT* (*WATCHDOG*) призначено для перезапуску програми тільки якщо з'явиться збій у ході її виконання. Програма, що працює без збоїв, періодично скидає сторожовий таймер, не допускаючи його переповнювання. Сторожовий таймер має свій власний *RC*-генератор, що працює на частоті 1 МГц. На вході *WDT* увімкнено переддільник вхідної частоти з програмовним коефіцієнтом ділення, що дозволяє регулювати часовий інтервал переповнення таймера і скидання мікроконтролера. Таймер *WDT* можна вимкнути програмним способом під час роботи мікроконтролера як в активному режимі, так і в будь-якому з режимів зниженого енергоспоживання. В останньому випадку це значно зменшує споживання струму.

Аналоговий компаратор *AC* порівнює за напругою сигнали, що надходять на входи *P*1.0 і *P*1.1. Результат порівняння подається на вхід *P*3.6, що не має зовнішнього виводу.

Аналого-цифровий перетворювач *ADC* побудовано за класичною схемою послідовних наближень із пристроєм вибірки/зберігання (ПВЗ). Кожний з аналогових входів можна з'єднати із входом ПВЗ через аналоговий мультиплексор. Пристрій вибірки/зберігання має свій власний підсилювач, який гарантує, що вимірюваний аналоговий сигнал буде стабільним протягом усього часу перетворення. Розрядність АЦП складає 10 біт при нормованій похибці +/– 2 розряди. АЦП може працювати у двох режимах – одноразового перетворення сигналу обраного каналу і послідовного циклічного опиту всіх каналів. Час перетворення вибирається програмно за допомогою установки коефіцієнта розподілу частоти спеціального переддільника, що входить до складу блока АЦП, і становить 70–280 мкс для *ATmega*103 та 65–260 мкс для всіх інших мікроконтролерів, що мають у своєму складі АЦП.

Внутрішній тактовий генератор *AVR*-МК можна запускати від декількох джерел опорної частоти (зовнішній генератор, зовнішній

кварцовий резонатор, внутрішнє або зовнішнє *RC*-коло). Оскільки *AVR*-МК цілком статичні, мінімальну припустиму частоту нічим не обмежено, тобто можна легко забезпечити навіть покроковий режим виконання програми. Максимальна робоча частота визначається конкретним типом мікроконтролера.

Мікроконтролери *AVR* можна перевести програмним шляхом в один із шести режимів зниженого енергоспоживання.

1. Режим холостого ходу (*IDLE*), у якому припиняє роботу тільки процесор і фіксується вміст пам'яті даних, а внутрішній генератор синхросигналів, таймери, система переривань і *WDG*-таймер продовжують функціонувати.

2. Режим мікроспоживання (*Power Down*), у якому зберігається вміст регістрового файла, але зупиняється внутрішній генератор синхросигналів. Вихід із *Power Down* можливий або за сигналом загального скидання мікроконтролера, або від зовнішнього джерела переривання. З увімкненим WDG-таймером струм споживання в цьому режимі становить близько 60–80 мкА, а з вимкненим – менше 1 мкА для всіх типів AVR. Вищенаведені значення справедливі для живильної напруги в 5 В.

3. Режим зберігання енергії (*Power Save*), реалізований тільки в тих *AVR*, що мають у своєму складі систему реального часу. Здебільшого режим *Power Save* ідентичний *Power Down*, але він допускає незалежну роботу таймера/лічильника *RTC*. Вихід із режиму *Power Save* можливий за сигналом переривання, викликаним або переповненням таймера/лічильника *RTC,* або спрацьовуванням блока порівняння цього лічильника. Струм споживання в цьому режимі становить 6–10 мкА при напрузі живлення 5 В на частоті 32,768 кГц.

4. Режим заглушення шуму під час роботи АЦП (*ADC Noise Reduction*). У цьому режимі припиняється робота процесора, але дозволено роботу АЦП, двопровідного інтерфейсу I^2C і сторожового таймера.

5. Основний режим очікування (*Standby*). Відрізняється від режиму *Power Down* тим, що робота тактового генератора не припиняється. Це гарантує швидкий вихід мікроконтролера з режиму очікування лише за шість тактів генератора.

6. Додатковий режим очікування (*Extended Standby*). Ідентичний режиму *Power Save*, але робота тактового генератора теж не припиняється. Гарантує швидкий вихід з режиму за шість тактів генератора.

Система команд AVR-МК містить до 133 різних команд. Розрізняють п'ять груп команд: умовного розгалуження, безумовного розгалуження, арифметичні і логічні операції, команди пересилки даних, команди роботи з бітами. В останніх версіях *AVR*-МК сім'ї *Mega* реалізовано

функцію апаратного множення. За розмаїтістю і кількістю реалізованих команд *AVR*-МК більше подібні до *CISC*, ніж до *RISC*-процесорів. Система команд *PIC*-контролерів містить до 75 різних команд, а в *MCS*51 вона становить 111.

Загалом прогресивна RISC-архітектура AVR-МК у сполученні з регістровим файлом і розширеною системою команд дозволяє створювати компактні програми з високою швидкістю виконання.

Контрольні запитання

1. Назвіть склад периферії *AVR*-МК.
2. За якою архітектурою виконано *AVR*-МК? Які особливості цієї архітектури?
3. Назвіть нові блоки структурної схеми *AVR*-МК.
4. За скільки тактів виконується команда в а) *AVR*-МК, б) *PIC*-контролері, в) мікроконтролері *MCS*51?
5. Яке призначення сторожового *WDT*-таймера?
6. Назвіть режими програмування *Flash*-пам'яті програм. Поясніть їх переваги і недоліки.

8.3. Характеристики *AVR*-мікроконтролерів

Розглянемо докладніше характеристики сімей *AVR*:

– *Classic AVR* – основна лінія мікроконтролерів продуктивністю до 16 *MIPS*[*], пам'ять програм *Flash ROM* 2–8 кбайт, пам'ять даних *EEPROM* 64–512 байт, пам'ять даних *SRAM* 128–512 байт;
– *Mega AVR* із продуктивністю 4–6 *MIPS*, пам'ять програм *Flash ROM* 64–128 кбайт, пам'ять даних *EEPROM* 64–4 096 байт, пам'ять даних *SRAM* 1–4 кбайт, вбудований 10-розрядний 8-канальний АЦП, апаратний помножувач 8×8;
– *Tiny AVR* – недорогі мікроконтролери у 8-вивідному виконанні.

Мікроконтролери *Tiny* характеризуються найменшими серед *AVR*-МК ємностями пам'яті програм та обмеженим набором функцій. Однак малогабаритні корпуси, можливість роботи при напрузі живлення 1,8 В (ВІС з індексом *V*) дозволяють використовувати ці мікроконтролери в портативній апаратурі, зокрема, з батарейним живленням.

На іншому полюсі сім'ї *AVR*, за рівнем інтеграції і можливостей, знаходиться група *Mega*. Для мікроконтролерів цієї групи, особливо для недавно розроблених, характерні:

– велика ємність *Flash*-пам'яті програм (нині 8–128 кбайт);
– режим самопрограмування, забезпечений вбудованою програмою-завантажником;

[*] *MIPS – Million Instructions per second* – мільйон операцій за секунду

– вбудований помножувач, що підтримує множення дробових чисел із знаком та без нього;
– розширені набори вбудованої периферії;
– широкий набір спеціальних мікроконтролерних функцій, зокрема до шести режимів енергоощадження і можливість програмного встановлення тактової частоти;
– розширення системи команд до 130–133;
– наявність у нових моделях інтерфейсу граничного сканування (*IEEE* 1149.1/*JTAG*), що підтримує вбудоване налагодження і забезпечує ще один спосіб програмування *Flash*- і *EEPROM*-пам'яті, перемичок і бітів блокування;

Сім'я *Classic* тепер найпоширеніша. Мікроконтролери цього класу мають менші периферію й обчислювальні можливості, ніж контролери сім'ї *Mega*, але більші, ніж *Tiny*. Корпорація *Atmel* не планує подальшого розвитку сім'ї *Classic,* оскільки вважається, що ця сім'я функціонально збалансована і різноманітно подана.

У табл. 8.4.–8.6 наведено характеристики мікроконтролерів сімей *Tiny, Classic і Mega* відповідно.

У таблицях позначено:
– *Flash ROM* – ємність енергонезалежної пам'яті програм (у кілобайтах);
– *EEPROM* – ємність енергонезалежної пам'яті даних (у байтах);
– *SRAM* – ємність статичної пам'яті даних (у байтах);
– *ISP* – можливість програмування мікроконтролера на цільовій платі (*I*) при основній напрузі живлення або можливість самопрограмування без участі зовнішнього програматора (*S*);
– *I/O* – кількість ліній введення-виведення;
– таймери 8/16 – кількість і розрядність таймерів/лічильників;
– *ADC (channels)* – кількість каналів АЦП.

Таблиця 8.4. **Мікроконтролери *AVR* сім'ї *Tiny***

Тип	Напруга живлення, В	Тактова частота, МГц	*I/O*	*Flash*	*EEPROM*	*SRAM*	Інтерфейси	Аналогові входи	Таймери	*ISP*
*AT-tiny*11*L*	2,7–5,5	2	6	1 K	–	–	–	–	1×8	–
*AT-tiny*11	4,0–5,5	6	6	1 K	–	–	–	–	1×8	–
*AT-tiny*12*V*	1,8–5,5	1	6	1 K	64	–	–	–	1×8	*I*
*AT-tiny*12*L*	2,7–5,5	4	6	1 K	64	–	–	–	1×8	*I*
*AT-tiny*15*L*	2,7–5,5	1	6	1 K	64	–	–	*ADC* 4×10	2×8	*I*
*AT-tiny*26*L*	2,7–5,5	8	16<	1 K	128	128	*SPI UART*	*ADC* 11×10	2×8	*I*
*AT-tiny*26	4,0–5,5	16	16	1 K	128	128	*SPI UART*	*ADC* 11×10	2×8	*I*
*AT-tiny*28*V*	1,8–5,5	1	20	2 K	–	–	–	–	1×8	–
*AT-tiny*28*L*	2,7–5,5	4	20	2 K	–	–	–	–	1×8	–

Таблиця 8.5. **Мікроконтролери *AVR* сім'ї *Classic***

Тип	Напруга живлення, В	Тактова частота, МГц	*I/O*	*Flash*	*EEPROM*	*SRAM*	Інтерфейси	Аналогові входи	Таймери	*ISP*
*AT*90*S*12	2,7–6,0 4,0–6,0	4 12	15	1 K	64	–	–	–	1×8	*I*
*AT*90*S*23	2,7–6,0 4,0–6,0	4 10	15	2 K	128	128	*UART*	–	1×8 1×16	*I*
*AT*90*L*2323	2,7–6,0	4	3	2 K	128	128	–	–	1×8	*I*
*AT*90*S*2323	4,0–6,0	10	3	2 K	128	128	–	–	1×8	*I*
*AT*90*LS*2343	2,7–6,0	4	5	2 K	128	128	–	–	1×8	*I*
*AT*90*S*2343	4,0–6,0	10	5	2 K	128	128	–	–	1×8	*I*
*AT*90*LS*4433	2,7–6,0	4	20	4 K	256	128	*UART* *SPI*	*ADC* 6×10	1×8 1×16	*I*
*AT*90*S*4433	4,0–6,0	8	20	4 K	256	128	*UART* *SPI*	*ADC* 6×10	1×8 1×16	*I*
*AT*90*LS*8515	2,7–6,0	4	32	8 K	512	512	*UART* *SPI*	–	2×8 1×16	*I*
*AT*90*S*8515	4,0–6,0	8	32	8 K	512	512	*UART* *SPI*	–	2×8 1×16	*I*
*AT*90*LS*8535	2,7–6,0	4	32	8 K	512	512	*UART* *SPI*	*ADC* 8×10	2×8 1×16	*I*
*AT*90*S*8535	4,0–6,0	8	32	8 K	512	512	*UART* *SPI*	*ADC* 8×10	2×8 1×16	*I*
*AT*90*S*8534	1,8–6,0	4	32	8 K	256	512	*UART* *SPI*	*ADC* 6×10	2×8 1×16	*I*

Таблиця 8.6. **Мікроконтролери *AVR* сім'ї *Mega***

Тип	Напруга живлення, В	Тактова частота, МГц	*I/O*	*Flash*	*EEPROM*	*SRAM*	Інтерфейси	Аналогові входи	Таймери	*ISP*
1	2	3	4	5	6	7	8	9	10	11
*AT-mega*8*L*	2,7–5,5	8	23	8 K	512	1 K	*UART* *SPI*	*ADC* 8×10	2×8 1×16	*S*
*AT-mega*16*L*	2,7–5,5	8	32	16 K	512	1 K	*UART* *SPI*	*ADC* 8×10	2×8 1×16	*S*
*AT-mega*16	4,5–5,5	16	32	16 K	512	1 K	*UART* *SPI*	*ADC* 8×10	2×8 1×16	*S*

Продовження табл. 8.6

1	2	3	4	5	6	7	8	9	10	11
*AT-mega*32*L*	2,7–5,5	8	32	32 K	1 K	2 K	*UART SPI*	*ADC* 8×10	2×8 1×16	*S*
*AT-mega*32	4,5–5,5	16	32	32 K	1 K	2 K	*UART SPI*	*ADC* 8×10	2×8 1×16	*S*
*AT-mega*64*L*	2,7–5,5	8	53	64 K	2 K	4 K	2×*UART SPI*	*ADC* 8×10	2x8 2x16	*S*
*AT-mega*64	4,5–5,5	16	53	64 K	2 K	4 K	2×*UART SPI*	*ADC* 8×10	2×8 2×16	*S*
*AT-mega*103	4,0–5,5	6	48	128 K	4 K	4 K	*UART SPI*	*ADC* 8×10	2×8 2×16	*I*
*AT-mega*128*L*	2,7–5,5	8	53	128 K	4 K	4 K	2×*UART SPI*	*ADC* 8×10	2×8 2×16	*S*
*AT-mega*128	4,5–5,5	16	53	128 K	4 K	4 K	2×*UART SPI*	*ADC* 8×10	2×8 2×16	*S*
*AT-mega*161*L*	2,7–5,5	4	35	16 K	512	1 K	2×*UART SPI*	–	2×8 1×16	*S*
*AT-mega*161	4,0–5,5	8	35	16 K	512	1 K	2×*UART SPI*	–	2×8 1×16	*S*
*AT-mega*162*L*	2,7–5,5	8	35	16 K	512	1 K	2×*UART SPI*	–	2×8 1×16	*S*
*AT-mega*162	4,5–5,5	16	35	16 K	512	1 K	2×*UART SPI*	–	2×8 1×16	*S*
*AT-mega*163*L*	2,7–5,5	4	32	16 K	512	1 K	*UART SPI*	*ADC* 8×10	2×8 1×16	*S*
*AT-mega*163	4,0–5,5	8	32	16 K	512	1 K	*UART SPI*	*ADC* 8×10	2×8 1×16	*S*
*AT-mega*169	2,7–3,6	4	53 4×25 *LCD*	16 K	512	1 K	*UART SPI*	*ADC* 8×10	2×8 1×16	*S*
*AT-mega*8515*L*	2,7–5,5	8	35	8 K	512	512	*UART SPI*	–	1×8 1×16	*S*

Закінчення табл. 8.6

1	2	3	4	5	6	7	8	9	10	11
*AT-mega*8515	4,5–5,5	16	35	8 K	512	512	*UART SPI*	–	1×8 1×16	*S*
*AT-mega*8535*L*	2,7–5,5	8	32	8 K	512	512	*UART SPI*	*ADC* 8×10	1×8 1×16	*S*
*AT-mega*8535	4,5–5,5	16	32	8 K	512	512	*UART SPI*	*ADC* 8×10	1×8 1×16	*S*

Широка номенклатура *AVR*-МК дає користувачеві можливість оптимізувати співвідношення *продуктивність – енергоспоживання – ціна*.

Високу продуктивність забезпечують:

– виконанням команд за один тактовий цикл;
– конвеєром команд, що забезпечують одночасно з виконанням поточної команди вибірку наступної;
– потужною системою команд єдиного 16-розрядного формату;
– вбудованими апаратними пристроями.

Мале енергоспоживання забезпечують:

– *CMOS*-технологією;
– цілком статичною роботою – від покрокового режиму до максимальної тактової частоти.

Малу вартість як на рівні вартості апаратного обладнання, так і на рівні вартості розробки і налагодження прикладних програм, забезпечують:

– *Flash*-пам’яттю програм, яку програмують на цільовій платі;
– можливістю вибору мікроконтролера з достатньою і відповідною кількістю функцій і вбудованої периферії.

Нині співвідношення *продуктивність – енергоспоживання – ціна* для *AVR* – одне з найкращих на світовому ринку 8-розрядних мікроконтролерів.

Контрольні запитання

1. Назвіть сфери застосування мікроконтролерів сімей *Tiny*, *Classic* і *Mega*.
2. Які структурні особливості архітектур сімей *Tiny*, *Classic* і *Mega*?
3. За рахунок чого досягають високої продуктивності *AVR*-МК? Порівняйте її з продуктивністю *CISC*-мікроконтролерів, *PIC*-контролерів.
4. Яка мінімальна частота роботи *AVR*-МК?
5. За рахунок чого досягають невеликого рівня енергоспоживання *AVR*-МК?
6. Які чинники сприяють зменшенню вартості *AVR*-МК?

Розділ 9

СИГНАЛЬНІ МІКРОПРОЦЕСОРИ

Сигнальні мікропроцесори належать до класу спеціалізованих МП (див. підрозд. 2.1). Їх розроблено для розв'язання задач цифрової обробки сигналів, а саме:

- фільтрації сигналу;
- згортки двох сигналів;
- обчислення значень кореляційної функції двох сигналів;
- обчислення автокореляційної функції;
- прямого/зворотного перетворення Фур'є тощо.

Задачі цифрової обробки розв'язують в апаратурі зв'язку і передачі даних, засобах гідро- і радіолокації, медичному устаткуванні і робототехніці, керуванні двигунами, в автомобільній електроніці, телебаченні, вимірювальній техніці тощо.

Відмітна риса задач цифрової обробки сигналів – потоковий характер обробки великих обсягів даних у реальному режимі часу. Робота в реальному часі потребує підвищення швидкодії МП, а обробка великих масивів даних – апаратних засобів інтенсивного обміну із зовнішніми пристроями.

Високої швидкодії сигнальних МП досягають завдяки:

- застосуванню модифікованої *RISC*-архітектури;
- проблемно-орієнтованій системі команд, наприклад включенню до системи команд таких операцій, як множення з нагромадженням *MAC (C:= A × B + C)* із зазначеною в команді кількістю виконань у циклі і з правилом зміни індексів елементів масивів *A* і *B*;
- методам скорочення тривалості командного циклу, як-то конвеєризація команд;
- розміщенню операндів більшості команд у регістрах;
- використанню тіньових регістрів для збереження стану обчислень під час перемикання контексту;
- наявності апаратного множення, що дозволяє виконувати множення двох чисел за один командний такт;
- апаратній підтримці програмних циклів.

Сигнальні процесори різних компаній-виробників утворюють два класи процесорів: простіші та дешевші МП обробки даних у форматі з

фіксованою комою і дорожчі мікропроцесори, що апаратно підтримують операції над даними у форматі з плавучою комою.

9.1. Сигнальні процесори обробки даних у форматі з фіксованою комою

Перший сигнальний процесор *TMS*320*C*10, розроблений фірмою *Texas Instruments* 1982 року, обробляв числа з фіксованою комою.

Структуру типового представника сім'ї з фіксованою комою МП *TMS*320*xC*5*x* наведено на рис. 9.1.

Процесор виконано за гарвардською архітектурою, основаною на розподілі шин доступу до вбудованої пам'яті програм і даних. Це дозволяє зробити вибірку команди і даних в одному машинному циклі і забезпечує виконання більшості команд за один цикл.

Сигнальний процесор *TMS*320*xC*5*x* складається з центрального процесорного пристрою (*CPU*), вбудованої пам'яті програм і даних, багатофункціональних периферійних пристроїв, що здебільшого дозволяють позбутися додаткової зовнішньої апаратури.

Процесор містить шини: *PDATA* – шина даних пам'яті програм; *PADDR* – шина адреси пам'яті програм; *DDATA* – шина даних пам'яті даних; *DADDR* – шина адреси пам'яті даних для незалежного доступу до пам'яті програм і даних.

Центральний процесорний пристрій *CPU*. До його складу входять:

- 32-розрядний АЛП, який виконує більшість команд за один цикл;
- акумулятор *ACC*, розділений на два сегменти по 16 розрядів (*ACCH* і *ACCL*);
- акумуляторний буфер *ACCB;*
- арифметичний пристрій допоміжних регістрів *ARAU*;
- регістровий файл *AR*0–*AR*7 і регістр *INDR*;
- незалежний логічний блок *PLU*;
- апаратний помножувач 16×16;
- регістри зсуву: регістр масштабування і зсуву *SPL*, який виконує зсув ліворуч на 0–16 розрядів і призначений для вирівнювання та перетворення даних, узятих з пам'яті; регістр зсуву *SFL* на виході помножувача; регістр зсуву *SR*;
- стек *STACK*;
- покажчик команд *PC*;
- мультиплексори *MUX*.

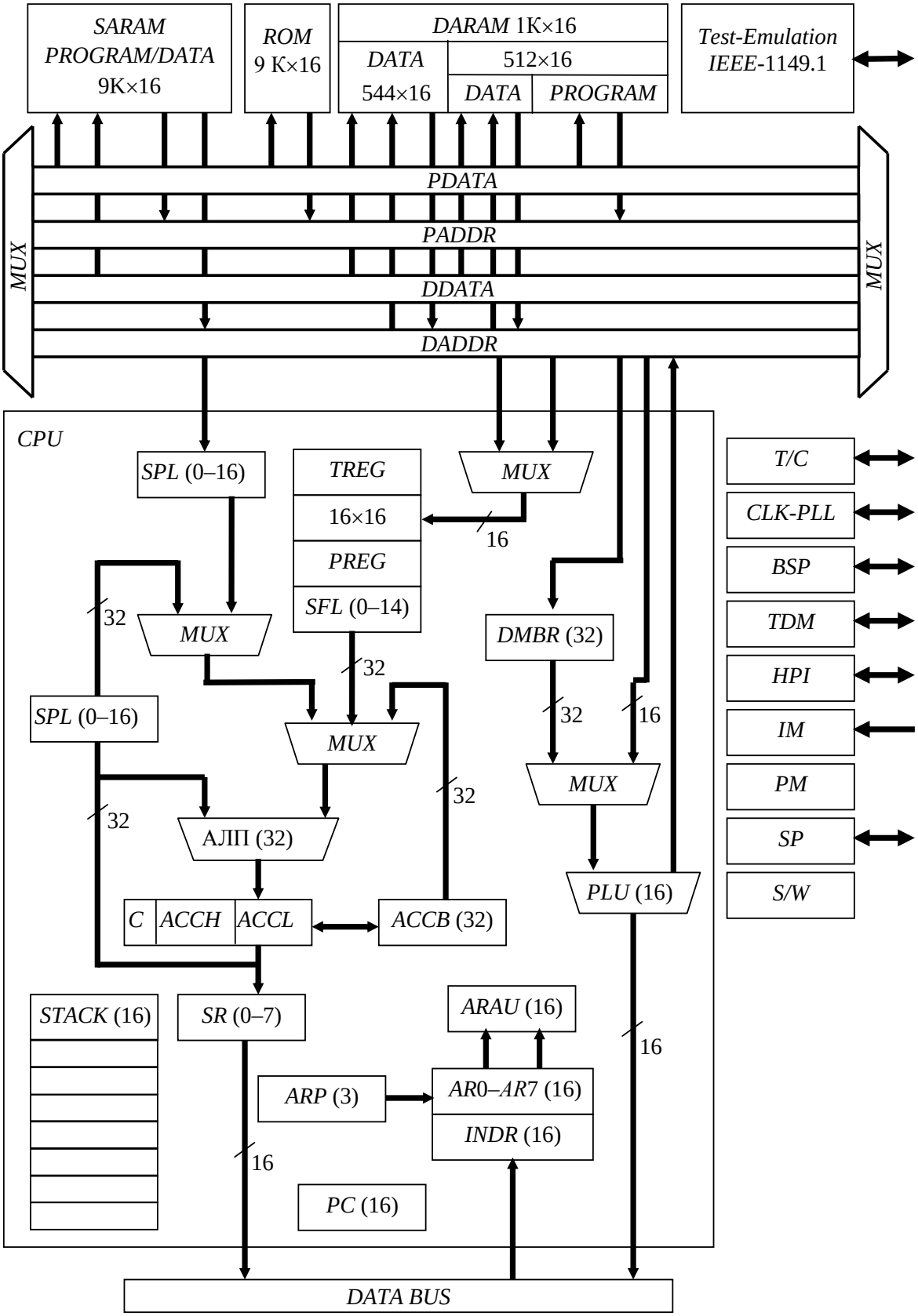


Рис. 9.1. Архітектура цифрових сигнальних процесорів *TMS*320*xC*5*x*

Арифметико-логічний пристрій АЛП. На перший вхід АЛП надходять дані одного з таких пристроїв:

– регістра масштабування і зсуву *SPL*;
– регістра зсуву *SFL* на виході регістра помножувача *PREG*;
– акумуляторного буфера *ACCB*.

На другий вхід АЛП дані завжди надходять з акумулятора *ACC*, а результат виконання операцій надходить також в *ACC*. Регістр зсуву *SR*, з'єднаний з виходом *ACC*, виконує зсув ліворуч на 0–7 розрядів, що відбувається в циклі пересилання даних з АЛП на внутрішню шину даних.

Апаратний помножувач 16×16. Виконує операції над числами зі знаком і без знака. Операнди надходять з пам'яті даних. Один з операндів може бути константою, поданою безпосередньо в команді. Для тимчасового збереження одного з операндів використовують 16-розрядний регістр *TREG*. У 32-розрядний регістр *PREG* завантажується результат множення.

Регістровий файл, що складається з восьми допоміжних регістрів (*AR*0–*AR*7) та індексного регістра (*INDR*) використовують для формування адреси при непрямій адресації. Якщо треба, *AR*0–*AR*7 можна використовувати для тимчасового збереження даних. Для адресації до допоміжних регістрів слугує покажчик допоміжних регістрів (*ARP*). Регістри *AR*0–*AR*7 завантажуються з пам'яті даних, акумулятора *ACC* чи операндом, поданим у команді. Уміст *AR*0–*AR*7 можна зберегти в пам'яті чи використати для обчислень в АЛП.

Арифметичний пристрій ARAU. Разом з регістрами *AR*0–*AR*7 і *INDR* його призначено для генерації адреси. Зазвичай *AR*0–*AR*7 використовують для зберігання адреси, а *INDR* містить зсув. Прості арифметичні операції (додавання, віднімання; інкрементування, декрементування), виконувані в *ARAU* з умістом *AR*0–*AR*7 і *INDR*, дозволяють реалізувати кілька видів непрямої адресації. Операції в *ARAU* виконуються одночасно з адресацією до поточної комірки пам'яті. Блок *ARAU* звільняє АЛП від роботи з обчислення адрес.

Логічний блок PLU виконує операції незалежно від АЛП. Результат операцій у *PLU* не впливає на біти стану АЛП. Перший операнд надходить у *PLU* з пам'яті даних, другий – з пам'яті чи програм регістра маніпуляції бітами (*DBMR*). Спеціальні логічні команди, виконувані тільки *PLU*, дозволяють у 16-розрядному слові встановлювати та очищувати будь-яку кількість біт у довільній комбінації. Результат операцій у *PLU* зберігається в тій самій комірці пам'яті, звідки було обрано перший операнд. Отже, логічні операції можна виконувати безпосередньо зі змістом будь-якої комірки пам'яті даних, зокрема зі змістом перших 16 портів введення-виведення, що можуть адресуватися як пам'ять даних (адреси 50*H*–5*FH*).

Пам'ять. МП *TMS*320*x*C5*x* передбачає роздільну адресацію до пам'яті програм, даних і портів введення-виведення. Ємність кожної області пам'яті 64 кбайт 16-розрядних слів. Вбудована в кристалі пам'ять – *ROM, SARAM, DARAM* знаходиться в загальному адресному просторі пам'яті і може використовуватися як пам'ять програм чи даних.

Пам'ять програм ROM – програмовна маскою пам'ять з можливістю захисту від зовнішнього доступу. У МП *TMS*320*xC*5*x* передбачено два режими роботи: мікропроцесорний і мікрокомп'ютерний. У мікрокомп'ютерному режимі *ROM* доступна, у мікропроцесорному – закрита для доступу. Вибір режиму визначається рівнем напруги на вході *MP/MC* під час скидання. Після старту режим можна змінити програмно. Для зміни режиму роботи в регістрі стану процесорного режиму (*PMST*) передбачено біт керування (*MP/MC*).

Пам'ять даних чи програм/даних SARAM (Single Access RAM) передбачає виконання однієї операції *читання/запис* у повному машинному циклі. *SARAM* складається з незалежних блоків по 2 кбайт чи по 1 кбайт слів. Кожен блок допускає одну операцію *читання/запис* у машинному циклі. Тому в одному машинному циклі *CPU* може двічі звернутися до *SARAM*, але тільки тоді, коли звернення відбувається до різних блоків. Цю особливість *SARAM* треба враховувати під час розподілу пам'яті і написання програм. Ще одна особливість *SARAM* полягає в тому, що процесор чи інший зовнішній пристрій може звертатися безпосередньо до *SARAM* у режимі ПДП. Для ініціалізації доступу до *SARAM* процесор спочатку запитує доступ до зовнішньої пам'яті (установлює сигнал *HOLD*). Після одержання сигналу підтвердження (*HOLDA*) процесор може запросити доступ до *SARAM* (установлює сигнал *BR*). У цьому разі *CPU* зупиняє всі поточні операції і підтверджує можливість доступу до *SARAM* (установлює сигнал *IAQ*). Пам'ять *SARAM* можна використовувати для зберігання тільки даних, програм чи для спільного розміщення програм і даних. Конфігурація *SARAM* змінюється програмно за допомогою двох біт конфігурації — *OVLY* і *RAM*, що знаходяться в регістрі *PMST*. Можливість реконфігурації *SARAM* у процесі виконання програми дозволяє оперативно змінювати розподіл пам'яті МП *TMS*320*xC*5*x*.

Пам'ять даних DARAM (Dual Access RAM) передбачає виконання однієї операції читання й однієї операції запису в повному машинному циклі без конфліктів на внутрішній шині даних. *DARAM* складається з блоків *B*, *B*1, *B*2. Блоки *B*1 (32 слова) й *B*2 (512 слів) використовують тільки як пам'ять даних. Блок *B* (512 слів) можна використовувати як пам'ять даних чи як пам'ять програм. Конфігурація блока *B* змінюється програмно за допомогою біта конфігурації (*CNF*) у регістрі стану (*ST1).* Призначення блока *B* можна змінювати в процесі виконання програми. Передбачено можливість завантаження блока *B* програмним кодом із зовнішньої пам'яті з наступним виконанням.

Периферійні пристрої. Сигнальний процесор має такі периферійні пристрої. *Модуль переривання IM (Interrupt Module)* призначено для обслуговування зовнішніх, внутрішніх і програмних переривань.

До зовнішніх переривань належать два немасковані переривання *RS*, *NM*1 і п'ять маскованих *INT*1–*INT*4. Внутрішні переривання (*RINT*, *XINT*,

TRNT, TXNT) генеруються послідовними портами або таймером (*TINT*). Програмні переривання викликаються командами *TRAP*, *INTR*, *NMI.*

Переривання встановлюють прапорець переривань у регістрі прапорців переривань (*IFR*) і можуть маскуватися в регістрі переривань (*IMR*).

Вектори переривань займають два 16-розрядні слова, потрібні для розміщення команд розгалуження. Після скидання МП вектори переривань розміщуються з нульової адреси пам'яті програм. Під час звернення *CPU* до векторів переривань старші п'ять розрядів адреси формуються покажчиком векторів переривань (*IRTR*) у регістрі *PMST*.

Вбудований механізм захисту багатоциклічних команд забезпечує ввімкнення механізму обслуговування переривань після завершення їх виконання. Дія механізму захисту поширюється також на команди, що стають багатоциклічними внаслідок їх повторення з префіксом RPT, і на команди, що очікують завершення обміну із зовнішньою пам'яттю чи портами введення-виведення.

Блок керування енергоспоживанням PM (Power Management). У МП *TMS*320*xC*5*x* передбачено три енергоощадні режими роботи (зокрема, «сплячий» режим), у яких струм споживання знижують відключенням *CPU* чи периферійних пристроїв. Перехід в енергоощадні режими ініціює активний сигнал *HOLD* (режим захоплення зовнішньої шини) чи команди *IDLE*, *IDLE*2. Вихід з енергоощадних режимів відбувається за сигналами зовнішніх переривань, які треба встановити принаймні протягом п'яти машинних тактів чи за внутрішніми перериваннями. В енергоощадних режимах зберігається стан усіх внутрішніх регістрів, що дозволяє без затримок продовжити роботу після виходу з цих режимів.

У режимі захоплення зовнішньої шини (*HOLD* = 0) продовжують працювати тільки внутрішні ресурси МП *TMS*320*xC*5*x*. Зниження струму споживання відбувається за рахунок переключення зовнішніх шин у високоімпедансний стан.

Інструкція *IDLE*2 викликає зупин *CPU* і периферійних пристроїв («сплячий» режим), що значно знижує струм споживання.

Вбудований генератор CLK-PLL виробляє тактові синхросигнали для роботи *CPU* і периферійних пристроїв. Передбачено можливість підключення зовнішнього кварцового резонатора чи резонатора зовнішнього тактового генератора. Допустимий режим роботи з множенням чи з діленням частоти джерела тактового сигналу.

Таймер/лічильник T/C являє собою 16-розрядний лічильник, що працює на вирахування. Коли досягнено нульове значення, генерується переривання *TINT* і формується імпульс на виході *TOUT*. Тривалість

імпульсу дорівнює періодові сигналу *CLKOUT*1. Таймером керують програмно, його можна зупинити, перезапустити, скинути чи заборонити.

Інтерфейс TE (Test-Emulation) забезпечує можливість тестування мікросхем і підключення емулятора типу *XDS*510. Зв'язок з емулятором відбувається по стандартному послідовному інтерфейсу *IEEE*1149.1 (*JTAG*).

Генератор тактів очікування S/W (Waitstate Generator) призначений для генерації і додавання в цикли обміну тактів очікування для збільшення часу циклів обміну з повільною зовнішньою пам'яттю чи портами введення-виведення. Використання генератора дозволяє обійтися без додаткової зовнішньої апаратури, що формує сигнал готовності *READY*. Генератор керується програмно. Кількість тактів очікування програмується окремо для пам'яті програм, даних, портів введення-виведення та областей адресного простору. Для керування генератором передбачено два регістри керування. Кількість тактів очікування може бути 0, 1, 2, 3, 7.

Послідовний порт SP (Serial Port). Це стандартний послідовний порт, який дозволяє по шести лініях організувати повнодуплексний зв'язок між двома МП *TMS*320*xC*5*x*. Для передачі даних в одному напрямі використовують три лінії, по яких передаються: тактова частота, синхроімпульс, дані синхронно з тактовою частотою. Тактову частоту і синхроімпульс формує МП *TMS*320*xC*5*x*, але, якщо треба, тактову частоту і синхроімпульс можуть формувати і зовнішні пристрої. Можливі два режими передачі даних: пакетний, у якому синхроімпульс формується на початку кожного переданого слова; безупинний, у якому синхроімпульс формується тільки на початку передачі. Вхідні і вихідні регістри зсувів буферизовано. Обмін по стандартному послідовному порту відбувається під керуванням *CPU*. Допускається 8- чи 16-розрядний формат передачі. Максимальна швидкість передачі даних залежить від тактової частоти МП *TMS*320*xC*5*x*. Для циклу 50 нс максимальна швидкість передачі даних становить 5 Мбіт/с.

Послідовний порт із часовим поділом каналів TDM використовують для обміну даними між МП *TMS*320*xC*5*x* у мультипроцесорних системах. *TDM*-порт працює у двох режимах, що переключаються програмно. Перший – режим стандартного послідовного порту, розглянутий вище. Другий – режим часового поділу каналів, у якому для синхронізації передачі даних між процесорами МП *TMS*320*xC*5*x* кожні 128 тактів (*TCLK*) передається синхроімпульс (*TFRM*). По лінії даних (*TDAT)* передаються 16-розрядні дані, по лінії адреси (*TADD*) передається адреса. Керування роботою і контроль за станом *TDM*-порту відбувається за допомогою шести регістрів.

Буферизований послідовний порт BSP (Buffer Serial Port). До його складу входить інтерфейс послідовного порту (*SPI*), що являє собою удосконалену версію стандартного послідовного порту і блока автобуферизації (*ABU*). Блок *ABU* дає можливість виконувати обмін даними безпосередньо з вбудованою пам'яттю МП *TMS*320*xC*5*x* через спеціально виділену шину незалежно від *CPU*. Для буфера обміну даними використовують виділені 2 кбайт вбудованої пам'яті МП *TMS*320*xC*5*x*. Для адресації до пам'яті *ABU* має власний адресний регістр. Ємність і початкова адреса буфера програмуються. Допускається 8-, 10-, 12- чи 16-розрядний формат передачі в пакетному чи безупинному режимі.

8-розрядний паралельний порт – host-інтерфейс HPI (Host Processor Interface). Призначений для обміну даними в мультипроцесорному режимі між *host*-процесором і МП *TMS*320*xC*5*x*. *Host*-інтерфейс *HPI* забезпечує можливість простої інтеграції процесора в мультипроцесорну систему. Обмін даними відбувається через вбудовану буферну пам'ять ємністю 2 кбайт слів по спеціальній внутрішній шині, що дозволяє обмінюватися з пам'яттю без конфліктів з *CPU*. Буферна пам'ять – *SARAM* пам'ять. Для керування *HPI* передбачено регістр керування і контролю (*HPIC*), доступний *host*-процесору і *CPU*. Для адресації до буферної пам'яті з боку *host*-процесора слугує адресний регістр (*HPIA*). *HPI* допускає два режими роботи. Перший – режим (*SAM*), у якому *host*-процесору і *CPU* дозволено доступ до пам'яті, причому *host*-процесор має пріоритет перед *CPU*. Другий – режим (*HOM*), у якому тільки *host*-процесор має доступ до пам'яті. Для передачі через *HPI* одного байта даних потрібно п'ять машинних тактів. При тактовій частоті 40 МГц максимальна швидкість передачі становить 64 Мбіт/с.

Блок початкового завантаження BL (Boot Loader) виконує пересилання програмного коду із зовнішніх джерел у вбудовану пам'ять програм. Ініціалізація програми початкового завантаження відбувається після ввімкнення живлення тільки в мікрокомп'ютерному режимі. Передбачено сім видів завантаження, що різняться способом і форматом передачі даних: через послідовний порт у 8- чи 16-розрядному форматі; через порти введення-виведення у 8- чи 16-розрядному форматі; із зовнішньої пам'яті у 8- чи16-розрядному форматі; «гаряче завантаження». Вид завантаження визначається вмістом молодших 8 розрядів комірки загальної пам'яті з адресою *FFFFH*, до якої МП *TMS*320*xC*5*x* звертається після ввімкнення живлення. Перед пересиланням програмного коду передається заголовок, що містить адресу початку розміщення програмного коду і довжину блоку, що пересилається. Після завершення пересилання в пам'ять програм МП *TMS*320*xC*5*x* стартує з адреси, зазначеної в заголовку.

У МП *TMS*320*xC*5*x* доступ до зовнішньої пам'яті і портів введення-виведення можливий по шині адреси *AT–A*15 і по шині даних *D*0–*D*15 за

допомогою керувальних сигналів *PS, DS, IS* (для вибору відповідно пам'яті програм, даних і портів введення-виведення), строба *STRB*, сигналу *напрямок передачі* в поточному циклі *R/W*, сигналу *читання RD* і сигналу *запис WE*. Максимальна продуктивність забезпечується у процесі обміну з високошвидкісною зовнішньою пам'яттю, що дозволяє працювати без тактів очікування. Можливе підключення повільної і дешевшої пам'яті. У цьому разі в цикли *читання/запис* МП *TMS*320*xC*5*x* треба додавати такти очікування, які генеруються вбудованим генератором тактів очікування, чи формувати зовнішній сигнал *READY* – готовності зовнішньої пам'яті чи портів введення-виведення. Організовуючи обмін із зовнішньою пам'яттю, слід також враховувати, що цикли *читання* мають тривалість одного машинного такту, водночас тривалість циклів *запис* становить два машинні такти або, якщо запис відбувається безпосередньо за читанням, то навіть три такти. У верхніх адресах пам'яті даних може розміщуватися зовнішня глобальна пам'ять даних розміром від 256 до 32 К слів. Адреси від 00*H* до 5F*H* пам'яті даних відведено під внутрішні регістри. Перші 16 портів введення-виведення розміщені в пам'яті даних. Тому звертання до цих портів можливе не тільки за допомогою команд *IN* і *OUT*, але й за допомогою звичайних команд звернення до пам'яті (завдовжки 1 слово), що дозволяють зменшити розмір програмного коду і збільшити швидкість обчислень.

Контрольні запитання

1. Назвіть сфери використання сигнальних процесорів.
2. Наведіть приклади задач цифрової обробки сигналів.
3. Якими факторами визначається швидкодія сигнальних процесорів?
4. Які існують класи сигнальних процесорів?
5. Назвіть основні блоки структурної схеми сигнального процесора обробки даних з фіксованою комою.
6. У чому полягає модифікація гарвардської архітектури в сигнальному процесорі *TMS*320*xC*5*x*?
7. Яке призначення має додатковий арифметичний пристрій *ARAU*?
8. Які функції виконує логічний блок *PLU*?
9. У чому полягає відмінність пам'яті *SARAM* від *DARAM*? Який тип пам'яті більш швидкодійний?
10. Укажіть периферійні пристрої сигнального процесора.
11. Назвіть джерела переривань сигнального процесора.
12. У яких енергоощадних режимах може працювати сигнальний процесор?
13. Яке призначення має генератор тактів очікування *S/W*?
14. Які функції виконує блок початкового завантаження *Boot Loader*?

9.2. Сигнальні процесори обробки даних у форматі з плавучою комою

Використання формату з плавучою комою для сигнальної обробки даних зумовлене рядом задач (інтегральні перетворення, алгоритми компресії, декомпресії, адаптивної фільтрації), які потребують високої точності подання даних у широкому динамічному діапазоні. Робота з даними у форматі з плавучою комою спрощує і прискорює обробку, підвищує надійність програми, оскільки не потребує виконання операцій округлення і нормалізації даних, відстеження ситуацій втрати значущості і переповнення. Але апаратні і вартісні затрати для таких МП значно більші, ніж для процесорів обробки даних у форматі з фіксованою комою.

Першим представником класу процесорів із плавучою комою став МП *TMS*320*C*30, структурну схему якого наведено на рис. 9.2.

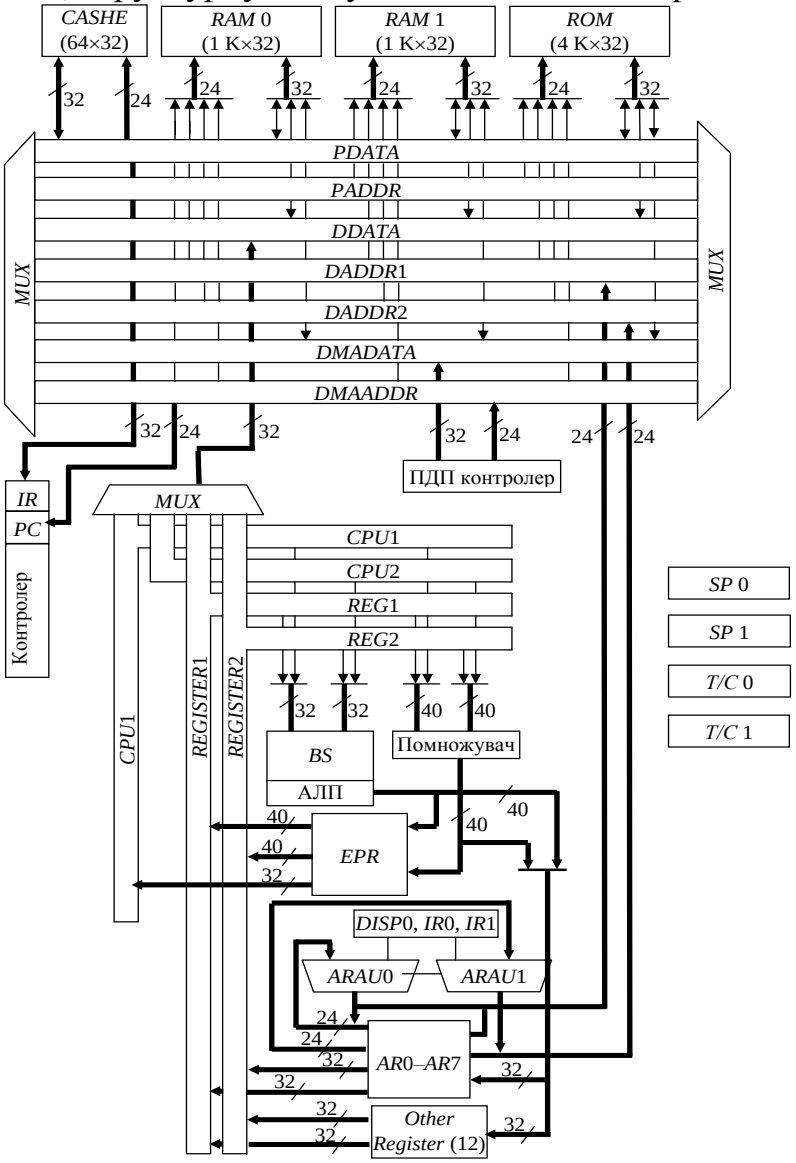


Рис. 9.2. Архітектура цифрових сигнальних процесорів *TMS*320*C*30

Процесор має 32-розрядну шину команд і даних та 24-розрядну шину адреси, містить два блоки ОЗП по 1 К 32-розрядних слів, 32-розрядний блок множення із плавучою комою, кеш-пам'ять команд обсягом 64 32-розрядні слова, 8 регістрів для операцій з підвищеною точністю, два генератори адреси і регістровий файл, реалізує різноманітні методи адресації. 40-розрядний АЛП процесора працює як з цілими числами, так і з числами у форматі з плавучою комою. Вбудований контролер ПДП дозволяє сполучати в часі обчислення та обмін даними з пам'яттю. Наявність у *TMS*320*C*30 мультипроцесорного інтерфейсу, двох зовнішніх інтерфейсних портів, двох послідовних портів, розширеної системи переривань спрощує конструювання систем на його основі.

Усі операції в процесорі виконуються за один такт. Процесор може паралельно виконувати в одному такті операцію множення і арифметико-логічну операцію з числами у форматі з фіксованою чи плавучою комою. Процесор має гнучку систему команд та підтримку мови високого рівня – *C*.

Процесор містить декілька шин для незалежного доступу до вбудованої пам'яті програм і даних:

– *PDATA* – шина даних пам'яті програм;
– *PADDR* – шина адреси пам'яті програм;
– *DDATA* – шина даних пам'яті даних;
– *DADDR*1 – шина адреси пам'яті даних *RAM*0;
– *DADDR*2 – шина адреси пам'яті даних *RAM*1;
– *DMADATA* – шина даних пам'яті в режимі ПДП;
– *DMAADDR* – шина адреси пам'яті в режимі ПДП.

32-розрядні шини даних і 24-розрядні шини адреси забезпечують можливість незалежного вибору операндів з різних джерел.

Центральний процесорний пристрій *CPU*. Містить внутрішні шини даних *CPU*1 і *CPU*2 і регістрові шини *REG*1 і *REG*2. Шини *CPU*1 і *CPU*2 дають можливість процесору передавати два операнди з пам'яті даних і регістрового файла за один машинний цикл. Регістрові шини *REG*1 і *REG*2 можуть за один машинний цикл передавати два слова даних з регістрового файла в помножувач і АЛП. Отже, чотири незалежні внутрішні шини *CPU* забезпечують можливість паралельного виконання команд в АЛП і помножувачі в одному циклі. Помножувач і АЛП, що оперує з цілими числами у форматі з плавучою комою, забезпечують високу продуктивність і точність обчислень.

До складу *CPU* входять також 32-розрядний регістр зсуву *BS* (*Barrel Shifter)*, регістровий файл (28 регістрів), два арифметичні пристрої допоміжних регістрів *ARAU*0 і *ARAU*1, що разом з 8 допоміжними регістрами *AR*0–*AR*7 забезпечують гнучкі режими непрямої адресації. Крім зазначених пристроїв *CPU* містить також конвеєр, механізм контролю за перериваннями, лічильник переривань, лічильник повторень, механізм умовних переходів.

Арифметико-логічний пристрій АЛП виконує за один цикл арифметичні операції з 32-розрядними цілими числами і 40-розрядними числами у форматі з плавучою комою, а також логічні операції з 32-розрядними операндами. 32-розрядний кільцевий регістр зсуву *BS* дозволяє за один цикл виконувати зсув праворуч чи ліворуч на 1–32 розряди.

Помножувач. Виконує множення 24-розрядних цілих чисел і 32-розрядних чисел у форматі з плавучою комою. У першому випадку результат – 32-розрядне число, у другому – 40-розрядне число, подане у форматі з плавучою комою.

Регістровий файл CPU складається з 28 регістрів. Частина регістрів доступна помножувачу і АЛП та може використовуватися як регістри загального призначення. Вісім 40-розрядних регістрів розширеної точності *EPR* підтримують операції з цілими числами і числами у форматі з плавучою комою. Вісім 32-розрядних допоміжних регістрів *AR*0–*AR*7 доступні АЛП і пристроям *ARAU*. Регістри *AR*0–*AR*7 можна використовувати для організації циклічних лічильників. Два 32-розрядні індексні регістри *IR*0, *IR*1 використовуються пристроями *ARAU* для індексації адреси. Крім зазначених регістрів у регістровий файл входять: покажчик системного стека *SP*, регістр стану *ST*, регістр прапорців переривань *IF*, регістр масок переривань *IE*, регістри адреси початку повторень *RS* і адреси кінця повторень *RE* та інші, призначені для керування процесом обчислення.

*Арифметичні пристрої ARAU*0 і *ARAU*1. Ці два пристрої можуть протягом одного циклу генерувати дві адреси на шинах *DADDR*1 і *DADDR*2. Пристрої *ARAU* підтримують адресацію зі зсувом, з індексацією (за допомогою регістрів *IR*0 і *IR*1) і циклічну. Виконання операцій в *ARAU* відбувається паралельно з операціями в АЛП і помножувачі.

CPU підтримує роботу з трьома форматами подання чисел: цілі числа зі знаком і без знака, числа у форматі з плавучою комою. Цілі числа можна подати в короткому форматі (діапазон подання чисел знаходиться в межах $-2^{15} < n < 2^{15}-1$) і форматі одинарної точності (діапазон подання чисел знаходиться в межах $-2^{31} < n < 2^{31}-1$). Числа з плавучою комою можна подати в короткому форматі, форматі одинарної точності і розширеному форматі. Для подання чисел із плавучою комою в розширеному форматі потрібно 40 розрядів, 8 з яких – для значення показника ступеня, а 32 – для значення мантиси. У розширеному форматі подання чисел максимальне додатне число дорівнює $3{,}4028236683 \times 10^{38}$, а мінімальне від'ємне число дорівнює $-3{,}4028236691 \times 10^{38}$. Операції над числами з плавучою комою забезпечують точні обчислення і дозволяють усунути проблеми переповнення і вирівнювання операндів, що виникають у цілочисловій арифметиці. Для перетворення форматів чисел (тобто з цілочислового на формат із плавучою комою і навпаки), округлення і нормалізації передбачено спеціальні команди.

Незалежні шини доступу до вбудованої пам'яті, внутрішні шини *CPU*, а також можливість одночасної генерації двох адрес дозволяють реалізувати гнучкі режими адресації, що забезпечують вибірку даних з пам'яті, з регістрового файла чи вибірку команд. У *CPU* використовують такі типи адресації: регістрова, пряма, непряма (27 видів), безпосередня, *PC*-відносна (*PC – Program Counter).* Зазначені типи адресації використовують у різних режимах адресації – триопера́ндному, паралельному, циклічному і реверсивному.

Команди з триопера́ндним режимом адресації виконують вибірку двох операндів, операцію з обраними операндами і запис результатів операцій у регістр. Усі зазначені дії виконуються в одному циклі.

Команди паралельних операцій виконують паралельне завантаження регістрів, паралельні арифметико-логічні операції та операції завантаження. Усього передбачено 28 таких команд. Приміром, одна з них (*MPYF*3*//ADDF*3) дозволяє перемножити і додати дві пари чисел у форматі з плавучою комою, інша (*MPYI*3*//*SUBI3) – перемножити і відняти дві пари цілих чисел.

Циклічну адресацію застосовують для реалізації багатьох алгоритмів цифрової обробки сигналів (згортки, кореляції та ін.).

Реверсивна адресація знаходить широке застосування для реалізації алгоритмів швидкого перетворення Фур'є.

CPU має механізми, що дозволяють керувати процесом виконання програми. Це лічильник повторень, механізм переходів (стандартних і затриманих) і контролер переривань.

Лічильник повторень використовують для виконання команд із префіксами *RPTB* – повторення блоку команд і *RPTS* – повторення однієї команди, що дозволяє спростити організацію циклів. Такі цикли застосовують для реалізації багатьох алгоритмів цифрової обробки сигналів, у яких існують повторні групи команд, що займають велику частину часу реалізації алгоритму. Використання повторень дозволяє скоротити час реалізації таких алгоритмів.

Контролер переривань обробляє зовнішні, внутрішні та програмні переривання. До зовнішніх переривань належать переривання скидання *RESET* та переривання, що надходять на виводи ВІС *INT*0*–INT*3. Внутрішні переривання формуються послідовними портами *XINT*0, *RINT*0, *XINT*1, *RINT*, таймерами *TINT*0 і *TINT*1 і контролером ПДП *DINT*. Програмні переривання викликаються командою *TRAP*. Вектори переривань займають адреси з 00*H* до 3F*H*. Переривання можуть обслуговуватися як *CPU*, так і контролером ПДП. Контролер ПДП використовує переривання для синхронізації операцій пересилання даних. Переривання встановлюють прапорець переривання в регістрі прапорців переривань *IF* і можуть маскуватися в регістрі *IE* маскування переривань.

*CPU TMS*320*C*31 передбачає два режими роботи зі зменшеним споживанням струму, що задаються командами *IDLE*2 і *LOPOWER*. Зниження

струму споживання в першому режимі досягається зупином *CPU*, а в другому – зменшенням у 16 разів тактової частоти МП. Вихід з цих режимів відбувається за сигналами зовнішніх переривань. Типове значення струму споживання *TMS*320*C*31 з тактовою частотою 50 МГц становить приблизно 250 мА. У режимі *IDLE2* струм споживання знижується до 50 мА.

Пам'ять. Загальний обсяг пам'яті *TMS*320*C*31 становить 16 М×32 слів. У ньому розміщуються області пам'яті програм, даних і пристроїв введення-виведення. Команди, дані і таблиці можуть зберігатися в будь-якому місці загальної пам'яті.

Вбудована пам'ять складається з 4 кбайт *ROM* (тільки для *TMS*320*C*30), 2 кбайт *RAM* (2 блоки по 1 кбайт) і 64 слів кеш-пам'яті програм. У *TMS*320*C*31 і *TMS*320*C*32 замість *ROM* вбудовано блок початкового завантаження (*Boot Loader*). *TMS*320*C*32 має всього 512 слів *RAM*-пам'яті (2 блоки по 256 слів). *ROM*- і *RAM*-пам'ять підтримують подвійне звернення за один цикл. Незалежні шини даних *PDATA, DDATA, DMADATA* дозволяють в одному циклі зробити вибірку команд, читання/запис даних і операцію прямого доступу до пам'яті ПДП.

МП *TMS*320*C*3*x* може працювати у двох режимах: мікропроцесорному і мікрокомп'ютерному. Режим роботи визначається станом входу *MC/MP*. У мікрокомп'ютерному режимі *ROM* розміщується з адреси 000*H* до *FFFH*. За адресами з 00*H* до 3F*H* розміщено вектори переривань. У мікропроцесорному режимі вбудована *ROM* маскується, й у зазначених адресах розміщується зовнішня пам'ять.

Кеш-пам'ять програм призначена для зберігання часто повторюваних наборів команд, що дозволяє зменшити кількість звернень до зовнішньої пам'яті і використовувати повільну і дешевшу зовнішню пам'ять. Зовнішні шини *TMS*320*C*3*x* звільняються в такий спосіб для операцій ПДП чи введення-виведення даних. Вибірка команд із вбудованої пам'яті (*ROM* чи *RAM*) не модифікує кеш-пам'ять. Читання/запис даних у кеш-пам'яті не виконується. Передбачено чотири режими роботи кеш-пам'яті, що встановлюються програмно.

Зовнішня пам'ять. Обмін із зовнішньою пам'яттю і пристроями введення-виведення відбувається через основну і додаткову шину (див. рис. 9.2). Основна шина містить шину даних *D*31–*D*0, шину адреси *A*23–*A*0 і керувальні сигнали: строб звертання до пам'яті *STRB*, сигнал читання/запис *R/W*, сигнал готовності *RDY*, сигнал захоплення основної шини *HOLD* і сигнал підтвердження захоплення *HOLDA*.

Додаткова шина включає шину даних *XD*31–*XD*0, шину адреси *XA*12–*XA*0 і керувальні сигнали: строб звертання до пам'яті *MSTRB*, строб звертання до пристроїв введення-виведення *IOSTRB*, сигнал читання/запис *XR/W* і сигнал готовності *XRDY*.

Основна і додаткова шини мають відповідні регістри керування.

Для того щоб забезпечити максимальну швидкодію процесора, тобто без тактів очікування, будь-який пристрій, що підключається до МП, має мати час вибірки не більш 25 нс (для тактової частоти *TMS*320*C*3*x* – 33 Мгц). Використання повільної і дешевшої пам'яті істотно збільшує гнучкість і знижує вартість системи, але в циклі звертання до пам'яті чи до пристроїв введення-виведення треба додавати такти очікування. *TMS*320*C*3*x* дозволяє формувати такти очікування як на основній, так і на додатковій шинах. При цьому обидві шини мають незалежну логіку керування готовності пам'яті чи пристроїв введення-виведення. Такти очікування може формувати внутрішній генератор тактів очікування чи зовнішній сигнал готовності. Внутрішній генератор тактів очікування може формувати від 0 до 7 тактів (керується програмно), причому кількість тактів однакова для всіх зовнішніх циклів незалежно від використовуваної адреси. Крім цього, для формування внутрішнього сигналу готовності в *TMS*320*C*3*x* передбачено можливість об'єднання сигналу від внутрішнього генератора тактів очікування і зовнішнього сигналу *RDY* за схемою логічного І або логічного АБО, що дозволяє гнучкіше використовувати зовнішні ресурси. Вибір режиму формування внутрішнього сигналу готовності відбувається програмно.

У *TMS*320*C*3*x* передбачено механізм програмного переключення банків пам'яті, що істотно полегшує розробку систем, у яких використовують велику ємність пам'яті. На час переключення банків сигнал *STRB* стає неактивним на один такт. При такому способі усувається потреба додавання тактів очікування, формованих сигналом *RDY* під час перетинання границь адрес банків пам'яті. Ємність банку пам'яті встановлюється програмно. Механізм переключення банків пам'яті передбачено тільки для основної шини.

Обмін по зовнішніх шинах накладає певні обмеження щодо швидкодії. Так, тривалість циклу *читання* зовнішньої пам'яті становить один такт без тактів очікування, тривалість циклу *запис* – два чи навіть три такти, якщо запис відбувається безпосередньо за читанням. Тривалість циклів *читання/запис* пристроїв введення-виведення (активний сигнал *IOSTRB*) на додатковій шині становить два такти (якщо немає тактів очікування).

Контролер ПДП. Виконує пересилання даних усередині адресного простору пам'яті і пристроїв введення-виведення без участі *CPU*, що дає можливість вести обмін з повільними пристроями пам'яті і пристроями введення-виведення (кодеки, послідовні порти, АЦП та ін.), не зменшуючи продуктивності *CPU.* Пересилання даних під керуванням контролера ПДП складаються з операцій *читання* і *запис*. Пересилання даних завершуються тільки тоді, коли виконано як операцію *читання*, так і операцію *запис.* Пересилання даних може бути синхронізовано до джерела, до приймача чи до джерела і приймача. Це означає, що окремі операції (*читання* або *запис* чи обидві разом) не виконуються доти, доки не виникне відповідне переривання.

Контролер ПДП і швидкісні зовнішні шини (основна і додаткова) – могутній засіб для побудови високошвидкісних систем обробки даних. Роботою контролера ПДП керують чотири регістри: загального керування, адреси джерела, адреси приймача і регістр лічильника пересилань. 24-розрядні регістри адреси джерела і приймача можуть інкрементуватися чи декрементуватися незалежно один від одного. 24-розрядний регістр лічильника пересилань, керований 24-розрядним лічильником, використовують для задання розміру блоку, що пересилається. Крім зазначених регістрів, у керуванні контролером ПДП використовується регістр *IE* маскування переривань.

Таймер T/C. Це 32-розрядний таймер/лічильник подій із зовнішнім чи внутрішнім тактуванням. Таймер має зовнішній вихід, що використовують як вихід сигналу таймера або як вхід для тактових імпульсів. Таймер можна використовувати для генерації сигналів, переданих зовнішнім пристроям (наприклад, запуск АЦП), чи для підрахунку зовнішніх імпульсів з наступною генерацією переривань у *CPU* після досягнення заданих показань лічильника. Для керування роботою таймера передбачено регістри керування. 32-розрядний регістр лічильника містить поточне значення лічильника. Зміст лічильника може інкрементуватися за переднім чи заднім фронтом тактового сигналу. 32-розрядний регістр загального керування таймера слугує для задання режимів роботи таймеру і керування ним.

Послідовні порти SP (Serial Port) дають можливість організувати повнодуплексний зв’язок між двома МП *TMS*320*C*3*x* по шести лініях. Послідовні порти цілком незалежні й ідентичні за керуванням. Для передачі даних в одному напрямі використовують три лінії, по яких передають: тактову частоту, дані, синхроімпульси. Допускається 8-, 16-, 24- і 32-розрядний формат передачі, причому передача даних може відбуватися як у фіксованому, так і в змінному форматі. Можливі два режими передачі даних: безупинний і пакетний. Тактова частота і синхроімпульси можуть бути як внутрішніми, так і зовнішніми. Вхідні і вихідні зсувні регістри буферизовано. Обмін даними через послідовний порт відбувається під керуванням *CPU*.

Блок початкового завантаження BL (Boot Loader) виконує пересилання програмного коду із зовнішньої недорогої пам’яті (*EPROM)* чи через послідовний порт із зазначеною адресою. Завантаження із зовнішньої пам’яті може відбуватися у 8-, 16- чи 32-розрядному форматі. Завантаження через послідовний порт виконується в 32-розрядному форматі в пакетному режимі. Під час завантаження через послідовний порт синхроімпульси і тактову частоту формує зовнішній пристрій. Програма початкового завантаження ініціалізується за сигналами переривань *INT*3–*INT*0 тільки в мікрокомп’ютерному режимі. Після старту перевіряють стан прапорців переривань у регістрі *IF* і визначають вид завантаження. Кожному прапорцю переривань (*INT*3, *INT*2, *INT*1, *INT*0) відповідає певний вид завантаження.

Блокові програмного коду, що пересилають, має передувати заголовок, що містить крім коду формату передачі (для завантаження із зовнішньої пам'яті) розмір блоку, що пересилається, і початкову адресу розміщення програмного коду. Після завершення пересилання програмного коду починається виконання програми з початкової адреси розміщення блоку.

Інші ВІС – представники сім'ї *TMS*320*C*3*x* різняться переважно кількістю послідовних портів і каналів прямого доступу до пам'яті ПДП.

Основні сфери застосування МП сім'ї *TMS*320*C*3*xx* – цифрове аудіо, 3-*D* графіка, відеоконференцзв'язок, промислові роботи, копіювально-розмножувальна техніка, телекомунікаційні системи.

Контрольні запитання

1. Назвіть основні блоки структурної схеми сигнального процесора обробки даних з плавучою комою.
2. Які шини використовує сигнальний процесор *TMS*320*C*3*x* для незалежного доступу до вбудованої пам'яті програм і даних?
3. Які формати подання чисел у сигнальному процесорі *TMS*320*C*3*x*? Яке найбільше та найменше число, якими оперує МП?
4. Які переваги обчислення у форматі з плавучою комою перед обчисленням чисел з фіксованою комою?
5. Яке призначення арифметичних пристроїв *ARAU*0 та *ARAU*1?
6. Які функції виконує контролер ПДП?
7. Укажіть периферійні пристрої сигнального процесора *TMS*320*C*3*x*.
8. Назвіть джерела переривань сигнального процесора *TMS*320*C*3*x*.
9. У яких енергоощадних режимах може працювати сигнальний процесор *TMS*320*C*3*x*?
10. Які функції виконує блок початкового завантаження *Boot Loader*?
11. Дайте характеристику режимів адресації сигнального процесора.
12. У яких галузях техніки використовують сигнальні процесори з плавучою комою?

9.3. Технічні характеристики сигнальних процесорів

Сигнальні процесори фірми *Texas Instruments*. Наступними за розглянутими вище сім'ями сигнальних процесорів фірми *Texas Instruments* стали процесори обробки даних з плавучою комою *TMS*320*C*4*x*. Вони сумісні за системою команд із МП *TMS*320*C*3*x*, однак мають більшу продуктивність і кращі комунікаційні можливості.

До сім'ї *TMS*320*C*4*x* входять процесори *TMS*320*C*40, *TMS*320*C*44, *TMS*320*LC*40.

Процесор *TMS*320*C*40 має продуктивність 30 *MIPS*/60 *MFLOPS* і максимальну пропускну здатність підсистеми введення-виведення 384 Мбайт/с.

Він містить на кристалі 6 високошвидкісних (20 Мбайт/с) комунікаційних портів і 6 каналів ПДП, 2 К слів пам'яті, 128 слів програмної кеш-пам'яті та блок початкового завантаження. Дві зовнішні шини забезпечують 4 Г слів загального адресного простору.

Процесор *TMS*320*C*44 – дешевший варіант попереднього представника сім'ї, що має 4 комунікаційні порти й загальний адресний простір 32 М слів. Однак значення показників продуктивності і пропускної здатності процесора ті самі, що й у попереднього представника сім'ї.

Процесор *TMS*320*LC*40 – аналог *TMS*320*C*40, що вирізняється низьким енергоспоживанням, підвищеною продуктивністю (40 *MIPS*/80 *MFLOPS*) і більшою пропускною здатністю (488 Мбайт/с).

Подальшим розвитком сім'ї цифрових процесорів обробки сигналів компанії *Texas Instruments* є процесор принципово нової архітектури – *TMS*320*C*80, випущений наприкінці 1994 року. Процесор орієнтовано на застосування, зв'язані з високопродуктивною цифровою обробкою сигналу в широких галузях науки і техніки. Друга назва процесора – *MVP (Multimedia Video Processor*) – характеризує його високу ефективність у вирішенні завдань обробки зображень, 2- і 3-вимірної графіки, компресії і декомпресії відео- і аудіоданих, обробки зв'язкової інформації та ін.

Процесор *TMS*320*C*80 поєднує в одній мікросхемі п'ять повнофункціональних процесорів, чотири з яких – поліпшені цифрові процесори обробки сигналів *ADSP* (*Advanced Digital Signal Processor*), архітектуру яких орієнтовано на реалізацію алгоритмів цифрової обробки сигналів. Кожний *ADSP* дозволяє виконати за один командний такт кілька *RISC*-подібних операцій. П'ятий процесор, головний *MP* (*Master Processor*), являє собою 32-розрядний *RISC*-процесор з високопродуктивним обчислювачем із плавучою комою. Додатково до розглянутих вище процесорів у цьому МП розміщені:

- контролер обміну *TC* (*Transfer Controller*) – інтелектуальний контролер ПДП, що підтримує інтерфейс із *DRAM* і *SRAM*;
- відеоконтролер *VC* (*Video Controller*);
- система контролю і налагодження – порт *JTAG* (*IEE* 1149.1);
- 50 кбайт *SRAM*.

Продуктивність *TMS*320*C*80 досягає 2 млрд *RISC*-подібних команд на секунду. Пропускна здатність шини досягає 2,4 Гбайт/с – у потоці даних і 1,8 Гбайт/с – у потоці команд.

Процесор *TMS*320*C*80 має такі технічні характеристики:

- тактова частота 40 чи 50 МГц;
- 50 кбайт вбудованої *SRAM*;
- 64-розрядний контролер обміну з динамічним конфігуруванням шини на обмін з 64-, 32-, 16- і 8-розрядними словами;
- режим ПДП до 64-розрядної пам'яті *SRAM* та *DRAM*;

– 4 зовнішні переривання;
– вбудовані засоби внутрісхемної емуляції;
– напруга живлення 3,3 В;
– близько 4 млн транзисторів на кристалі;
– 0,5/0,6 КМДН-технологія;
– 305-контактний корпус *PGA*.

Нова сім'я *DSP*-процесорів компанії *Texas Instruments TMS*320*C*62*x* буде містити в собі процесори як з фіксованою, так і з плавучою комою. Перший представник цієї сім'ї – *TMS*320*C*6201, оперує з даними у форматі з фіксованою комою.

Побудований відповідно до розробленої компанією *Texas Instruments* архітектури *VelociTI* процесор *TMS*320*C*62*xx* – перший із сигнальних *VLIW*-процесорів, що використовує для підвищення продуктивності паралелізм рівня команд. Традиційна *VLIW*-архітектура передбачає наявність декількох функціональних пристроїв, що працюють паралельно і виконують за один такт кілька команд. Архітектура *VelociTI* дозволяє забезпечити кращу ефективність за рахунок ослаблення обмежень на порядок і спосіб виконання команд. Ядро *TMS*320*C*6201 – *VelociTI VLIW*-процесор з 8 функціональними модулями, включаючи 2 помножувачі і 6 АЛП. Модулі взаємодіють через два регістрові файли, що містять по 16 32-розрядних регістрів. ЦП може виконувати до 8 команд за один такт.

У процесорі використовують упаковку команд, що скорочує розміри коду і час вибірки команд. 256-розрядна шина пам'яті програм дозволяє вибирати за один такт вісім 32-розрядних команд. Усі команди містять умови їх виконання, що дозволяє скоротити витрати продуктивності процесора на виконання переходів і збільшити ступінь паралелізму обробки.

Процесор може оперувати з 8-, 16-, 32-розрядними даними. Для додатків, що потребують високої точності обчислень, передбачено можливість обчислень з 40-розрядними операндами. Для результатів усіх основних арифметичних операцій виконується округлення і нормалізація. У процесорі реалізовано операції над бітовими полями, такі як *виділити* (*extract*), *установити* (*set*), *очистити* (*clear*), *підрахунок бітів* (*bit counting*). Процес обробки *VLIW* починається з вибірки з пам'яті команд 256-розрядного пакета. Команди зв'язуються для спільного виконання в пакет (до 8 команд) за значенням молодшого біта команди (*LSB*).

Процесор реагує на 14 переривань, що відповідають сигналові *Reset*, немаскованому перериванню (*NMI*) і перериванням з номерами 4–15.

Внутрішньокристальну пам'ять розділено на пам'ять даних і пам'ять програм (по 64 кбайт). Процесор *TMS*320*C*6201 має два 32-розрядні порти для звернення до пам'яті даних і один 256-розрядний порт для звернення до пам'яті програм для вибірки команд, а також по 64 кбайт пам'яті даних і програм. У процесорі використовують розшарування пам'яті даних

(4 16-розрядні банки) для підвищення швидкості вибірки за рахунок одночасного звертання до різних банків пам'яті.

Додатково процесори сім'ї *TMS*320*C*62*xx* можуть містити на кристалі інтерфейс зовнішньої пам'яті (*EMIF*), контролер ПДП, інтерфейс *host*-порту (*HPI*), засоби енергоощадження, розширені буферизовані послідовні порти, 32-розрядні таймери.

Серед нових сфер застосування МП сім'ї *TMS*320*C*62*xx* компанія *Texas Instruments* називає:

– універсальний безпровідний зв'язок;
– медичну діагностику;
– телефонію;
– персональні засоби інформаційного забезпечення і захисту.

Водночас можливе використання МП *TMS*320*C*62*x* у сучасних прикладних системах для збільшення їх продуктивності. До таких систем належать:

– базові станції мобільного зв'язку;
– модемні пули і сервери;
– кабельні модеми;
– багатоканальні телефонні платформи, включаючи центральні офісні комутатори і системи мовної передачі повідомлень;
– мультимедійні системи.

Сигнальні процесори фірми *Motorola*. Сигнальні МП компанії *Motorola* підрозділяють на сім'ї 16- та 24-розрядних МП з фіксованою комою *DSP* (*Digital Signal Processor* – цифровий сигнальний процесор) – *DSP*560*xx*, -561*xx*, -563*xx*, -566*xx*, -568*xx* і МП з плавучою комою *DSP*960*xx*.

*16-розрядні МП з фіксованою комою DSP*56156 і *DSP*56166 орієнтовані насамперед на використання в системах обробки мови і телекомунікації.

Процесори сім'ї мають такі технічні характеристики:

– продуктивність до 30 *MIPS* на частоті 60 МГц;
– однотактовий паралельний помножувач (16×16 розрядів) з нагромадженням (*MAC*);
– два 40-розрядні акумулятори з байтом розширення;
– гнучку систему адресації;
– апаратну реалізацію вкладених циклів, включаючи нескінченні цикли;
– три 16-бітові внутрішні шини даних і три 16-бітові внутрішні шини адреси;
– програмовний час доступу до зовнішньої шини;
– інтерфейс із відображенням у пам'яті периферійних пристроїв;
– внутрішній блок емуляції і налагодження (*OnCE*);
– низьке споживання енергії і засоби енергоощадження;
– 2 К слів внутрішньокристальної пам'яті.

*24-розрядні МП з фіксованою комою DSP*56000/*DSP*56001 орієнтовані на максимізацію пропускної здатності в додатках *DSP* з інтенсивним обміном даними. МП працюють на частотах до 33 МГц і забезпечують продуктивність близько 16 *MIPS*, що дозволяє виконувати швидке перетворення Фур'є з 1 024 відліками за 3,23 мс. Відмінність між цими процесорами полягає в типі їх внутрішньої пам'яті. Щоб мінімізувати вартість прикладних систем, МП *DSP*56000 орієнтовано на роботу під керуванням програми, збереженої в ППЗП (*ROM*) ємністю 3,75 К слів. Є також варіант процесора *DSP*56000, що має захист від несанкціонованого доступу до програми, збереженої у внутрішній пам'яті.

Наступна лінія МП *DSP*56300 має процесорне ядро нового типу (*NDE – New DSP Engine*), у якому завдяки конвеєризації забезпечено виконання команди за один такт, що підвищує швидкодію вдвічі порівняно з ядром *DSP*560*xx*.

*32-розрядний МП з плавучою комою сім'ї DSP*96002 містить 1 024 слова пам'яті, рівномірно розділеної між пам'яттю даних *X* і *Y*, 1 024 слова програмної пам'яті, два ППЗП даних, двоканальні контролери ПДП, підсистему початкового завантаження програми, а також вбудовані засоби налагодження та емуляції.

Процесори сім'ї мають такі технічні характеристики:

- пристрій множення з нагромадженням розрядністю 32×32;
- спеціалізований набір команд;
- апаратну підтримку виконання програмних циклів і швидкого повернення з переривань;
- розширену до 1 К слів кеш-пам'ять команд;
- п'ять 32-розрядних адресних шин – внутрішні однонапрямлені шини адреси *X* та *Y*, програмну адресну шину і дві зовнішні адресні шини;
- сім 32-розрядних шин даних – внутрішні двонапрямлені шини даних *X* та *Y*, внутрішню двонапрямлену глобальну шину даних, внутрішню двонапрямлену шину даних ПДП, внутрішню двонапрямлену програмну шину даних і дві зовнішні шини даних;
- внутрішньокристальну пам'ять МП, що включає 1 024 слова програмної пам'яті (*RAM*), дві незалежні пам'яті даних по 512 слів кожна (*RAM*), два незалежні ПЗП ємністю 1 024 слова і ПЗП початкового завантаження ємністю 64 слова;
- зовнішню пам'ять процесора, яка може становити по 2×2^{32} 32-розрядних слів для команд і даних;
- продуктивність МП, що на тактовій частоті 40 МГц становить близько 200 *MIPS*.

Сигнальні процесори фірми *Analog Devices*. Це процесори двох сімей *ADSP*21*xx* і *ADSP*210*xx*. Технічні характеристики процесорів наведено в табл. 9.1.

Таблиця 9.1. **Цифрові сигнальні процесори фірми *Analog Devices***

Тип пристрою	Тактова частота, МГц	Час циклу, нс	Продуктивність, *MIPS*	Внутрішня пам'ять	Внутрішня пам'ять програм		Зовнішня пам'ять програм/даних	*I/O* лінії	Послідовний порт	Таймери	ПДП
					RAM	*ROM*					
1	2	3	4	5	6	7	8	9	10	11	12
ADSP-2101	20	50	20	1 K×16	2 K×24	–	16 K×16; 16 K×24	2	2	1	0
ADSP-2103	10,24	97,6	10,24	1 K×16	2 K×24	–	16 K×16; 16 K×24	2	2	1	0
ADSP-2104	10,24	50	20	512×16	1 K×24	–	16 K×16; 16 K×24	2	1	1	0
ADSP-2104*L*	13,824	72,3	13,824	512×16	1 K×24	–	16 K×16; 16 K×24	2	1	1	0
ADSP-2105	10,24	100	20	256×16	512×24	–	16 K×16; 16 K×24	2	1	1	0
ADSP-2109	10,24	50	20	512×16	1 K×24	4 K×24	16 K×16; 16 K×24	2	1	1	0
ADSP-2109*L*	13,824	72,3	13,824	512×16	1 K×24	4 K×24	16 K×16; 16 K×24	2	1	1	0
ADSP-2111	20	50	20	1 K×16	2 K×24	–	16 K×16; 16 K×24	5	2	1	0
ADSP-2115	20	50	20	512×16	1 K×24	–	16 K×16; 16 K×24	2	2	1	0
ADSP-2161	16,67	60	16,67	1 K×16	–	8 K×24	16 K×16; 16 K×24	2	2	1	0
ADSP-2162	10,24	100	10,24	1 K×16	–	8 K×24	16 K×16; 16 K×24	2	2	1	0
ADSP-2163	16,67	60	16,67	512×16	–	4 K×24	16 K×16; 16 K×24	2	2	1	0
ADSP-2164	10,24	100	10,24	512×16	–	4 K×24	16 K×16; 16 K×24	2	2	1	0
ADSP-2165	20	50	20	4 K×16	1K×24	12 K×24	16 K×16; 16 K×24	*N/A*	2	1	0
ADSP-2166	16,67	60	16,67	4 K×16	1 K×24	12 K×24	16 K×16; 16 K×24	*N/A*	2	1	0
ADSP-2171	16,67	30	33	2 K×16	2 K×24	–	16 K×16; 16 K×24	5	2	1	0
ADSP-2172	16,67	30	33	2 K×16	2 K×24	8 K×24	16 K×16; 16 K×24	5	2	1	0
ADSP-2173	10,24	50	20	2 K×16	2 K×24	–	16 K×16; 16 K×24	5	2	1	0

Продовження табл. 9.1

1	2	3	4	5	6	7	8	9	10	11	12
ADSP-2181	16,67	30	33	16 К×16	16 К×24	–	16 К×16; 16 К×24	13	2	1	1
ADSP-2183	16,67	34,7	33	16 К×16	16 К×24	–	16 К×16; 16 К×24	13	2	1	1
ADSP-21*msp*50	20	50	20	1 К×16	2 К×24	–	16 К×16; 16 К×24	*N/A*	2	1	*N/A*
ADSP-21*msp*58	13	38	26	2 К×16	2 К×24	–	16 К×16; 16 К×24	3	2	1	0
ADSP-21*msp*59	13	38	26	2 К×16	2 К×24	4 К×24	16 К×16; 16 К×24	3	2	1	0
ADSP-21*csp*01	25	20	50	4 К×16	4 К×24	–	16 М слів	12	2	1	5
ADSP-21020	33,3	30	33,3	–	–	–	4 Г слів	*N/A*	2	1	*N/A*
ADSP-21060	40	25	40	4 Мбіт	–	–	4 Г слів	4	2 + 6	1	10
ADSP-21061	50	20	50	1 Мбіт	–	–	4 Г слів	4	2	1	6
ADSP-21061*L*	44	22,5	44	1 Мбіт	–	–	4 Г слів	4	2	1	6
ADSP-21062	40	25	40	2 Мбіт	–	–	4 Г слів	4	2 + 6	1	10
ADSP-21065*L*	60	16.67	60	544 кбіт	–	–	64 М слів	12	8	2	10

Сім'я *ADSP*21*xx* – набір однокристальних 16-розрядних МП із загальною базовою архітектурою, оптимізованою для виконання алгоритмів цифрової обробки сигналів та інших додатків, що потребують високошвидкісних обчислень з фіксованою комою. Сім'я на сьогодні нараховує 14 представників, що відрізняються один від одного переважно розміщеними на кристалі периферійними пристроями, такими як кеш-пам'ять, таймери, порти і т. ін.

Друга сім'я МП *ADSP*210*xx* поєднує 32-розрядні МП, орієнтовані на сигнальні алгоритми, що потребують виконання обчислень із плавучою комою. Сім'ю представляють МП *ADSP*21010, *ADSP*21020, *ADSP*21060, *ADSP*21062.

Порівняння архітектур сигнальних процесорів за швидкодією. Основні показники продуктивності процесора *ADSP*-21061 з тактовою частотою 40 МГц (час циклу 25 нс) під час виконання тестових програм такі:

- швидке перетворення Фур'є з 1 024 відліками – 460 мкс (18 221 цикл);
- секція фільтра з кінцевою імпульсною характеристикою (*FIR*) – 25 нс (1 цикл);
- секція фільтра з нескінченною імпульсною характеристикою (*IIR*) 100 нс (4 цикли);
- ділення (*y*/*x*) – 150 нс (6 циклів);
- обчислення $1/\sqrt{x}$ – 225 нс (9 циклів);
- швидкість передачі через канали ПДП – 240 Мбайт/с.

На рис. 9.3. наведено порівняння архітектур сигнальних процесорів за кількістю тактів, що потребують виконання типових алгоритмів цифрової обробки – швидкого перетворення Фур'є, фільтрів *FIR* та *IIR*.

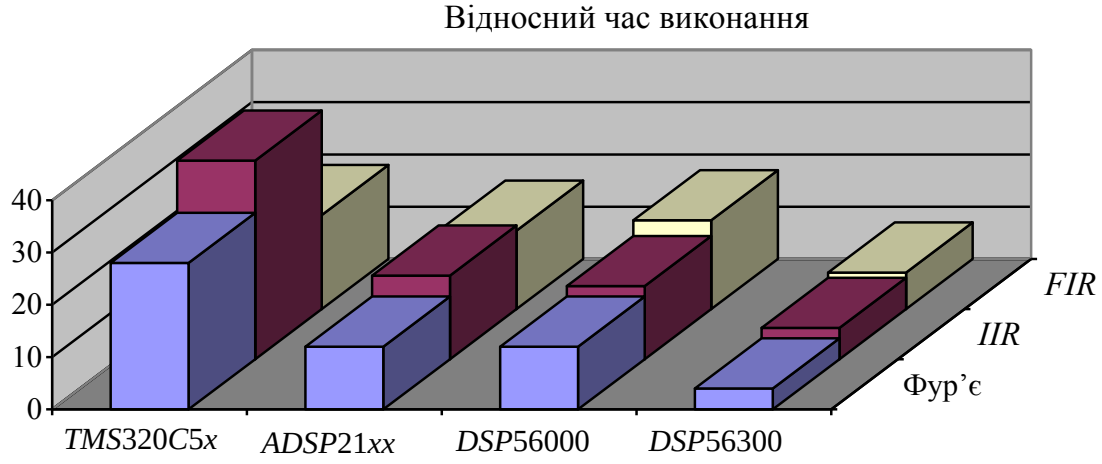


Рис. 9.3. Порівняння архітектур сигнальних процесорів

Як видно з діаграми (рис. 9.3), архітектура процесора *DSP*563*xx* фірми *Motorola* найефективніша і забезпечує швидше виконання завдань за рівних значень тактових частот МП.

Контрольні запитання

1. У якому процесорі поєднано п’ять процесорів? Як вони взаємодіють?
2. Назвіть основні сфери застосування процесорів *Texas Instruments.*
3. Який із процесорів має архітектуру *VelociTI*? Чим вона відрізняється від архітектури *VLIW*?
4. Які типи сигнальних процесорів виробляє фірма *Motorola*? Як застосовують кожну з груп?
5. Які типи сигнальних процесорів виробляє фірма *Analog Devices*?
6. Порівняйте за швидкодією сигнальні процесори фірм *Texas Instruments*, *Motorola* та *Analog Devices* для різних задач цифрової обробки сигналів.

Розділ 10

НЕЙРОННІ ОБЧИСЛЮВАЧІ

10.1. Основні поняття та задачі нейронних обчислювачів

У попередніх розділах розглянуто приклади розв'язання задач, добре формалізованих, тобто для них розроблено математичні моделі і можна застосувати алгоритми, що базуються на правилах типу «якщо А, то Б». Однак є задачі, які важко формалізувати, тобто знайти чіткий алгоритм розв'язання. До них належать такі:

Розпізнавання зображень. Наприклад, розпізнавання рукописних і друкованих символів під час оптичного введення в ЕОМ, розпізнавання типів клітин крові, розпізнавання мови. При цьому розпізнаваний об'єкт – це масив даних, який треба віднести до одного із заздалегідь відомих класів.

Кластеризація даних (пошук закономірностей). Вхідні дані треба віднести до якоїсь групи (кластера) за властивою їм «близькістю», причому число кластерів заздалегідь невідомо. Критерієм «близькості» можуть бути відстань між векторами даних, значення коефіцієнта кореляції тощо.

Апроксимація функцій. Потрібно знайти функцію, що апроксимує невідому (наприклад, набір експериментальних даних). Ця задача актуальна для моделювання складних систем і створення систем керування складними динамічними об'єктами, для робастного керування.

Прогнозування. Потрібно за попереднім поводженням функції прогнозувати її поводження в майбутньому. Ця задача актуальна для керування системами з прогнозуванням, для систем прийняття рішень.

Оптимізація. Мета цих задач знайти оптимальне значення цільової функції, що задовольняє ряд обмежень.

Між тим, людина добре розв'язує задачі, які важко формалізувати, – розпізнає зображення, класифікує дані, прогнозує тощо. Тому ідея створити штучний розум стала такою привабливою. Але для цього треба було провести численні дослідження принципів функціонування мозку людини з погляду обробки інформації.

Мозок людини – найскладніша з відомих систем переробки інформації. У ньому міститься близько 100 млрд нервових клітин, або нейронів, кожна з яких має в середньому 10 000 зв'язків.

Нейрони являють собою особливий вид клітин, основне призначення яких полягає в оперативному керуванні організмом. Схематичне зображення нейрона наведено на рис. 10.1.

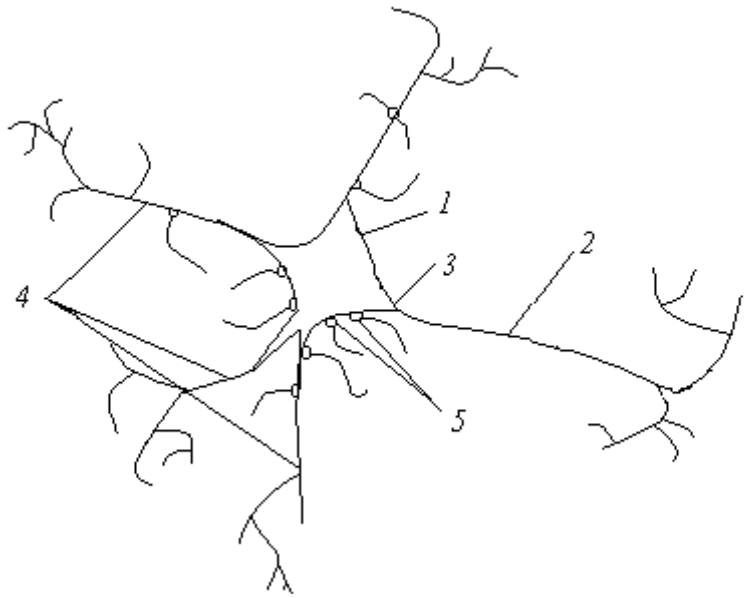


Рис. 10.1. Схематичне зображення нейрона

Нейрон має тіло (сому) *1*, дерево входів (дендритів) *4* і виходів (аксонів) *2*. Дендрити сильно розгалужуються, пронизуючи порівняно великий простір навколо нейрона. Початковий сегмент аксона – аксонний горбок *3*, що прилягає до тіла клітини, потовщений. У міру віддалення від клітини він поступово звужується і на ньому з'являється мієлінова оболонка, що має високий електричний опір. На сомі і на дендритах розміщуються закінчення аксонів, що йдуть від інших нервових клітин. Кожне таке закінчення *5* має вид потовщення, називаного синаптичною бляшкою або синапсом. Вхідні сигнали дендритного дерева (постсинаптичні потенціали) зважуються і підсумовуються на шляху до аксонного горбка, де генерується вихідний імпульс. Його наявність (або інтенсивність) – це функція зваженої суми вхідних сигналів. Вихідний сигнал проходить по гілках аксона і досягає синапсів, що з'єднують аксони з дендритними деревами інших нейронів. Через синапси сигнал трансформується в новий вхідний сигнал для суміжних нейронів. Цей вхідний сигнал може бути додатним і від'ємним (збудливим або гальмівним) залежно від виду синапсів. Значення вхідного сигналу, що генерує синапс, може відрізнятися від значення сигналу, що надходить у синапс. Ці відмінності визначають ефективність, або вагу, синапса. Синаптична вага може змінюватися в процесі функціонування синапса.

Учені різних спеціальностей робили спроби створити математичну модель нейрона. Так, біологи намагалися одержати аналітичне уявлення про нейрон, що враховувало б усі його відомі функціональні характеристики. Але основне завдання – передача інформації нервовим імпульсом – втрачалося серед багатьох параметрів, що стосуються фізики провідності імпульсів. Тому спробували замінити фізичний опис нейрона логічним. При цьому нервову клітину розглядали просто як елемент, що передає інформацію. 1943 року вчені-математики Мак-Каллох і Пітс подали нейрон як простий перемикальний елемент, що може знаходитися в одному з двох стійких станів «увімкнено» або «вимкнено». Нейрон спрацьовує, якщо алгебрична сума входів у певний момент більша за поріг. Нейрон у такому поданні можна використовувати як елемент ЕОМ, що дозволяє побудувати мережу з нейронів із відповідними порогами і зв'язками, що реалізовувала б довільну булеву функцію або таблицю істинності. Ці дослідження зумовили численні винаходи схем обробки інформації, пристроїв розпізнавання і сенсорних аналізаторів.

Нині найчастіше використовують модель нейрона, наведену на рис. 10.2.

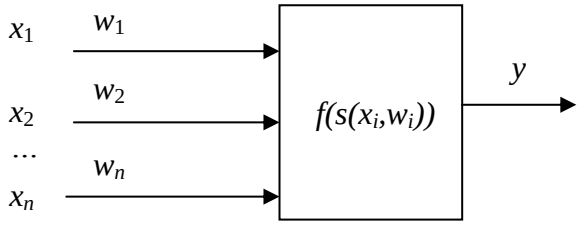


Рис. 10.2. Модель нейрона

Нейрон має n однонапрямлених входів (синапсів), з'єднаних із виходами інших нейронів, а також вихід y (аксон), по якому сигнал (збудження або гальмування) надходить на синапси наступних нейронів. Синапси характеризуються значенням синаптичного зв'язку або ваги w_i, що за фізичним змістом еквівалентно електричній провідності. Кожний нейрон характеризується своїм *поточним станом* s за аналогією з нервовими клітинами головного мозку, що можуть бути збуджені або загальмовані.

Поточний стан нейрона залежить від значення його входів, ваг та інколи ще й від попереднього стану. Найчастіше стан нейрона визначають або як зважену суму його входів:

$$s = \sum_{i=1}^{n} x_i w_i, \tag{10.1}$$

або як відстань між вектором входів і вектором ваг входів:

$$s = \sum_{i=1}^{n-x} \left| w_i - x_i \right|. \tag{10.2}$$

Вихід *y* нейрона – це функція його стану:

$$y = f(s). \tag{10.3}$$

Функцію $f(s)$ називають *функцією активації.*

Найпоширеніші функції активації – східчаста порогова, лінійна порогова, сигмоїдна, лінійна, та гауссіана (табл. 10.1).

Таблиця 10.1. **Функції активації нейрона**

Назва	Визначення
Східчаста порогова	$f(s)=\begin{cases}0, s<a\\1, s\ge a\end{cases}$
Лінійна порогова	$f(s)=\begin{cases}0, & s<a_1\\ ks+b, & a_1\le s<a_2\\1, & s\ge a_2\end{cases}$
Сигмоїдна	$f(s)=(1+e^{-k(s-a)})^{-1}$
Лінійна	$f(s) = ks + b$
Гауссіана	$f(s)=e^{-k(s-a)^2}$

Нейронну мережу створюють об’єднанням виходів одних нейронів зі входами інших, причому нейрони створюють шари, також з’єднані між собою. *Нейронна мережа* – це мережа з кінцевою кількістю шарів із однотипних елементів і різними типами зв’язків між шарами нейронів. При цьому кількість нейронів у шарах має бути достатньою для забезпечення заданої якості розв’язання задачі, а кількість шарів нейронів – якомога меншою, щоб зменшити час розв’язання.

Приклад найпростішої одношарової нейронної мережі, яку називають також простий персептрон, наведено на рис. 10.3. На *n* входів надходять сигнали, які проходять по синапсах на три нейрони, що утворюють єдиний шар з вихідними сигналами $y_j = f[\sum_{i=1}^{n} x_i w_{ij}],\ j = 1...3.$

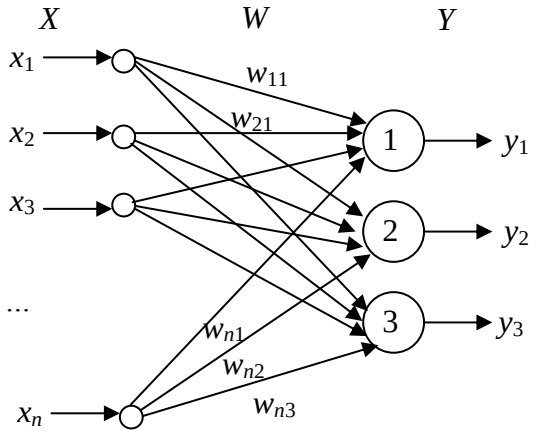


Рис. 10.3. Одношаровий персептрон

На рис. 10.4. подано двошаровий персептрон, отриманий з одношарового додаванням другого шару, що складається з двох нейронів. При цьому нелінійність активаційної функції має велике значення: якби її не було, результат функціонування будь-якої p-шарової нейронної мережі із ваговими матрицями $W^{(i)}$, i = 1, 2, ... p для кожного шару i зводився б до перемножування вхідного вектора сигналів X на матрицю $W^{(S)} = W^{(1)}*W^{(2)}*... *W^{(p)}$, тобто фактично така p-шарова нейронна мережа була б еквівалентна одношаровій з ваговою матрицею єдиного шару $W^{(S)}$: $Y = XW^{(S)}$.

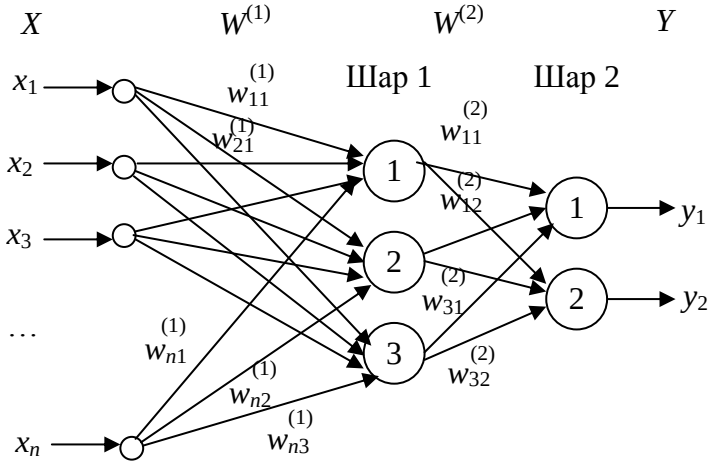


Рис. 10.4. Двошаровий персептрон

Нейронні мережі можна також класифікувати як ациклічні або циклічні. Приклади, наведені на рис. 10.3 та 10.4, стосуються ациклічних нейронних мереж. На рис. 10.5 наведено приклад циклічної нейронної мережі.

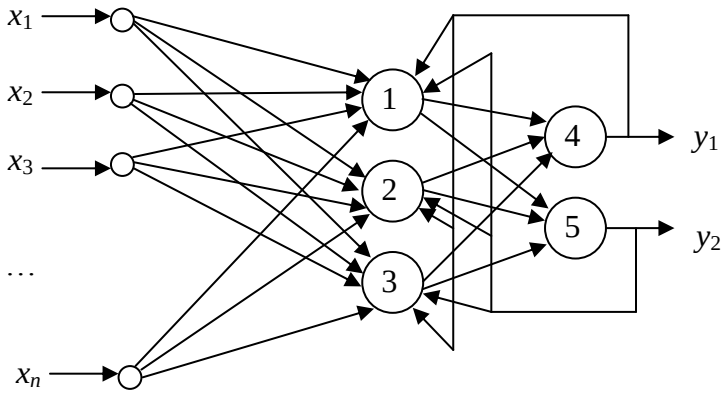


Рис. 10.5. Циклічна нейронна мережа

Якщо розглянуті схеми (рис. 10.3–10.5) доповнити положенням про тактування мережі (задати час спрацювання нейронів), то отримаємо апарат для задання різноманітних алгоритмів обробки даних за допомогою нейронних мереж, які можна використати для розв’язання як формалізованих задач, так і задач, які важко формалізувати. В останньому випадку застосування нейронних мереж основане не на

виконанні запропонованого алгоритму, а на запам'ятовуванні мережею поданих їй прикладів на етапі створення мережі й отримання результатів, погоджених із цими прикладами, на етапі розв'язання задачі.

Сигнали у нейронній мережі поділяють на *бінарні* (*цифрові*) й *аналогові.* Бінарні оперують двійковими сигналами, і вихід кожного нейрона може набувати тільки двох значень: 0 або 1.

За можливістю адаптації можна виділити: нейронні мережі, які *конструюють* та *навчають*. Для мережі, яку конструюють, задають кількість і тип нейронів, граф міжнейронних зв'язків, вагу входів. У мережі, яку навчають, графи міжнейронних зв'язків та ваги входів змінюються під час виконання алгоритму навчання.

За алгоритмом навчання мережі поділяють на мережі, за якими *спостерігають, не спостерігають* і *мішані* (*гібридні*). Перші під час навчання порівнюють заздалегідь відомий результат з отриманим. Другі навчаються, не знаючи правильних значень результату. Вони групують вхідні дані так, щоб вони формували той самий вихід мережі. Такий підхід використовують для розв'язання, наприклад, задачі кластеризації. У разі мішаного алгоритму навчання частину ваг визначають під час спостереження, а частину – без спостереження.

Контрольні запитання

1. Для розв'язання якого класу задач використовують нейропроцесори? Назвіть приклади таких задач.
2. У чому полягає загальна ідея використання нейронних мережевих обчислень?
3. Наведіть визначення понять *нейрон* і *нейронна мережа*.
4. Дайте визначення та наведіть приклади функцій стану нейрона.
5. Дайте визначення та наведіть приклади функцій активізації нейрона.
6. Наведіть приклади одношарових та двошарових нейронних мереж.
7. Чим відрізняються циклічні нейронні мережі від ациклічних?
8. Які є типи нейронних мереж?

10.2. Основи побудови алгоритмів навчання нейронних мереж

Один з найважливіших етапів розробки нейронного обчислювача – навчання мережі. Від якості навчання залежить спроможність мережі вирішувати поставлені перед нею завдання. На етапі навчання крім параметра якості добору вагових коефіцієнтів важливу роль відіграє час навчання. Зазвичай, ці два параметри зв'язані зворотною залежністю і їх доводиться вибирати на основі компромісу.

Навчання починається з вибору вихідної мережі (евристично обраний граф) із заданою кількістю входів і виходів. Наприклад, у літературі рекомендують тришарову мережу з кількістю нейронів внутрішнього шару, що дорівнює півсумі кількості входів і виходів мережі. Кожен нейрон внутрішнього шару зв'язаний з виходами всіх вхідних нейронів мережі. Кожен вихідний нейрон зв'язаний з виходами всіх нейронів внутрішнього шару. Далі пробують обрати ваги входів нейронів мережі так, щоб вирішувалося завдання. Якщо це не вдається, то змінюють граф мережі.

Найпоширеніший алгоритм визначення ваг для навчання зі спостереженням персептронних мереж, для якого доведено збігання процесу, – *алгоритм зворотного поширення помилки.* На сьогодні його реалізують 80 % нейрочипів, орієнтованих на задачі цифрової обробки сигналів. Крім усього іншого, він став певним еталоном для вимірювання продуктивності нейронних обчислювачів, так само, як швидке перетворення Фур'є на 1 024 відліків для сигнальних процесорів (див. пірозд. 8.3).

Під час навчання сигнал помилки поширюється по мережі у зворотному напрямі. Виконується корекція ваг входів нейронів, що запобігає повторній появі цієї помилки.

Алгоритм навчання *одношарового персептрона* (див. рис. 10.2 або рис. 10.3) такий. Є набір прикладів $\langle X_1, D_1 \rangle$, $\langle X_2, D_2 \rangle$,...,$\langle X_m, D_m \rangle$, де $X_j = \{x_{j1}, x_{j2}, ..., x_{jn}\}$ – вхідні значення j прикладу, D_j – вихідне значення цього прикладу. Вважають, що персептрон правильно навчений, якщо для всіх j $\max|D_j - Y_j| \le \delta$, де δ – задане значення помилки.

1. Присвоїти вагам і порогу нейрона випадкові малі значення.

2. Подати на входи нейрона черговий приклад $\langle X_j \rangle$ (починаючи з першого прикладу) і визначити значення виходу нейрона Y_j, $j = 1, ..., m$.

3. Змінити ваги відповідно до виразу $w_i(t+1) = w_i(t) + a(D_j - Y_j)x_i$, i = 1, 2, …, n; a – коефіцієнт, $0 < a < 1$.

4. Виконувати пп. 2, 3 доти, доки помилка в усіх прикладах не буде перевищувати заданого значення δ.

У *багатошарових мережах* алгоритм залишається тим самим, за винятком кроку 2, який складається з етапів корекції шарів, причому корекція починається з вихідного шару.

Крім багатошарових нейронних мереж зворотного поширення, основний недолік яких – неможливість гарантувати найкраще навчання за конкретний часовий інтервал, є досить велика кількість інших варіантів побудови нейронних мереж зі своїми перевагами та недоліками.

Проектування нейросистем – це складний і трудомісткий процес, у якому вибір конкретного алгоритму – тільки один із декількох кроків процесу

проектування. Він найчастіше включає: дослідження предметної області, структурно-функціональне проектування, топологічне проектування тощо.

Контрольні запитання

1. У чому полягає процес навчання нейронної мережі?
2. Назвіть основні кроки алгоритму зворотного поширення помилки для одношарових персептронів.
3. У чому полягає відмінність алгоритму зворотного поширення помилки для одно- і багатошарових персептронів?

10.3. Апаратна реалізація нейронних обчислювачів

Розрізняють такі архітектури обчислювальних систем (рис. 10.6):

– з одиничними потоками команд і даних *SISD (Single Instruction – Single Data)*, рис. 10.6, *а*;
– з одиничним потоком команд і множинним потоком даних *SIMD (Single Instruction – Multy Data)*, рис. 10.6, *б*;
– із множинним потоком команд і одиничним потоком даних *MISD (Multy Instruction – Single Data)*, рис. 10.6, *в*;
– із множинними потоками команд і даних *MIMD (Multy Instruction – Multy Data)*, рис. 10.6, *г*;

Нейронний обчислювач – це система з *MIMD*-архітектурою, яка працює за алгоритмом нейронної мережі.

Є три основні напрями побудови нейронних обчислювачів:

– на базі каскадного з'єднання універсальних *RISC* або *CISC* МП фірм *Intel, AMD, Sparc, Alpha, Power PC, MIPS*;
– на базі програмовних логічних матриць *PLM* або процесорів із паралельною обробкою даних на апаратному рівні, наприклад, сигнальних процесорів фірм *TMS, ADSP, Motorola*;
– на спеціалізованій елементній базі – однобітових процесорах, нейрочипах.

Системи першого напряму називають *нейроемуляторами.* Їх апаратна реалізація базується на використанні універсальних *RISC*- або *CISC*-мікропроцесорів, докладно розглянутих у попередніх розділах. Нейроемулятори реалізують типові *нейрооперації* (обчислюють зважену суму, виконують нелінійні перетворення) на програмному рівні.

Системи другого напряму у вигляді плат розширення стандартних обчислювальних систем називають *нейроприскорювачами*, а системи третього напряму у вигляді функціонально закінчених обчислювальних пристроїв – *нейрокомп'ютерами.*

Нейроприскорювачі поділяють на два класи – віртуальні, які вставляють у слот розширення стандартного комп'ютера, і зовнішні, які з'єднують з керувальною *host*-ЕОМ.

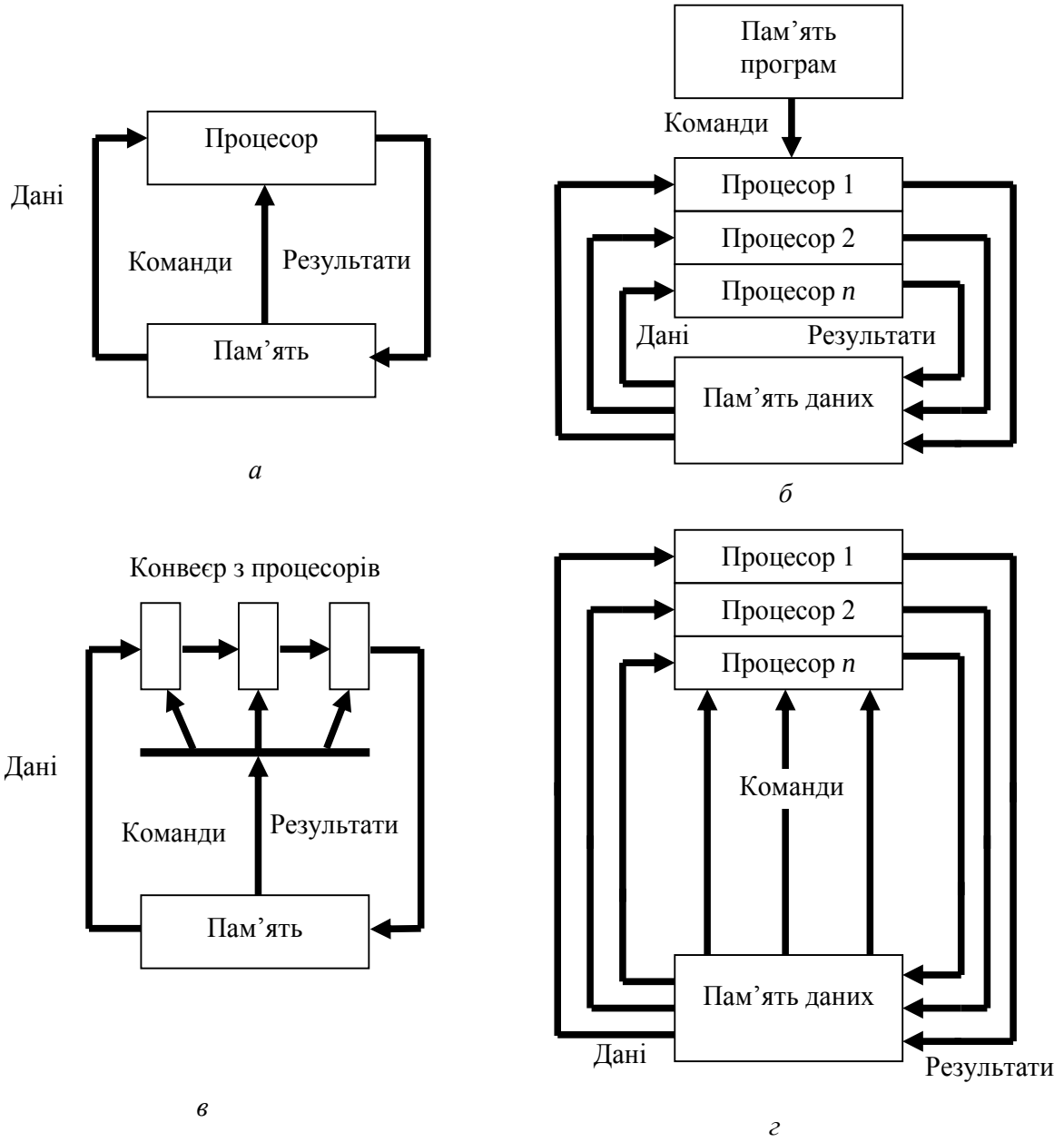


Рис. 10.6. Архітектури обчислювальних систем:
а – SISD; *б – SIMD*; *в – MISD*; *г – MIMD*

У нейроприскорювачах на базі програмовних логічних матриць *PLM* алгоритми нейромережі реалізовано апаратно. Сучасні *PLM* мають значний обсяг ресурсів (наприклад, *PLM* фірми *Xilinx* має на кристалі до 4 млн системних вентилів) і можуть реалізовувати апаратну структуру нейромережі такого типу, як на рис. 10.4, 10.5. Характеристики *PLM* для реалізації нейрообчислювачів подано в табл. 10.2.

Обчислювальні системи на базі *PLM* характеризуються досить високими частотами роботи, але зміна алгоритмів роботи потребує перепрограмування ВІС. Нейроприскорювачі на базі *PLM* тепер використо-

вують як гнучкі нейрообчислювальні системи для науково-дослідних цілей і дрібносерійного виробництва.

Таблиця 10.2*. Особливості реалізації нейрообчислювачів на базі *PLM

Тип *PLM*	Виробник	Кількість нейронів
*XC*4005*E/XL*	*XILINX*	6
*XC*4013*XLA*	XILINX	18
*XC*4020*XLA*	*XILINX*	24
*XC*4044*XLA*	*XILINX*	50
*XC*4062*XLA*	*XILINX*	72
*XC*4085*XL*	*XILINX*	97
*XC*40250*XV*	*XILINX*	200
*EPF*10*K*20	*ALTERA*	4
*EPF*10*K*50*E*	*ALTERA*	11
*EPF*10*K*100*E*	*ALTERA*	19
*EPF*10*K*250*E*	*ALTERA*	50
*M*4*LV*-96/48	*AMD*	3
*M*4*LV*-192/96	*AMD*	6
*M*5*LV*-256	*AMD*	8
*M*5*LV*-512	*AMD*	16

Для побудови продуктивніших нейрообчислювачів зазвичай застосовують сигнальні процесори (див. розд. 8). Сигнальні МП, розроблені для задач цифрової обробки сигналів, як виявилося, спроможні ефективно інтерпретувати алгоритми нейромереж. Вони орієнтовані на обробку масивів (векторів) даних, виконують операцію множення з нагромадженням. Деякі з них можуть одночасно виконувати дві команди. Ці особливості дозволяють легко реалізувати множення з нагромадженням векторів ваг і векторів входів нейронів мережі. Зазвичай під час створення нейрообчислювачів використовують гібридну структуру, коли блок матричних обчислень реалізується на базі каскадного з'єднання сигнальних процесорів, а логіка керування – на основі *PLM*.

Схему зовнішнього нейроприскорювача, що з'єднується з керувальною *host*-ЕОМ і виконана на базі каскадного з'єднання сигнальних процесорів, наведено на рис. 10.7.

Керувальну *host-EOM* реалізовано на основі звичайної обчислювальної системи з *CISC*- або *RISC*-процесорами.

Модуль матричних сигнальних процесорів об'єднує їх між собою відповідно до структури нейромережі. Схема містить також робочу пам'ять, пам'ять програм, модуль введення-виведення сигналів (що включає АЦП, ЦАП і *TTL*-лінії), а також модуль керування, що може бути реалізований на основі спеціалізованого керувального сигнального процесора, на основі *PLM* або мати розподілену структуру, коли функції загального керування розподілено між матричними сигнальними процесорами. Для побудови нейроприскорю-

вача цього типу найперспективніше використати сигнальні процесори із плавучою комою *ADSP*2106*x*, *TMS*320*C*4*x*, 8*x*, *DSP*96002 та ін. Вибір того або того сигнального процесора – багатокритеріальна задача, однак слід зазначити перевагу процесорів *Analog Devices* для додатків, що потребують виконання великих обсягів математичних обчислень, таких, як цифрова фільтрація сигналу, обчислення кореляційних функцій. Для завдань, що потребують інтенсивного обміну із зовнішніми пристроями, доцільно використовувати процесори *Texas Instruments*, що мають високошвидкісні інтерфейси. А компанія *Motorola* – лідер за обсягом виробництва дешевих і досить продуктивних 16- і 24-розрядніих сигнальних МП з фіксованою комою.

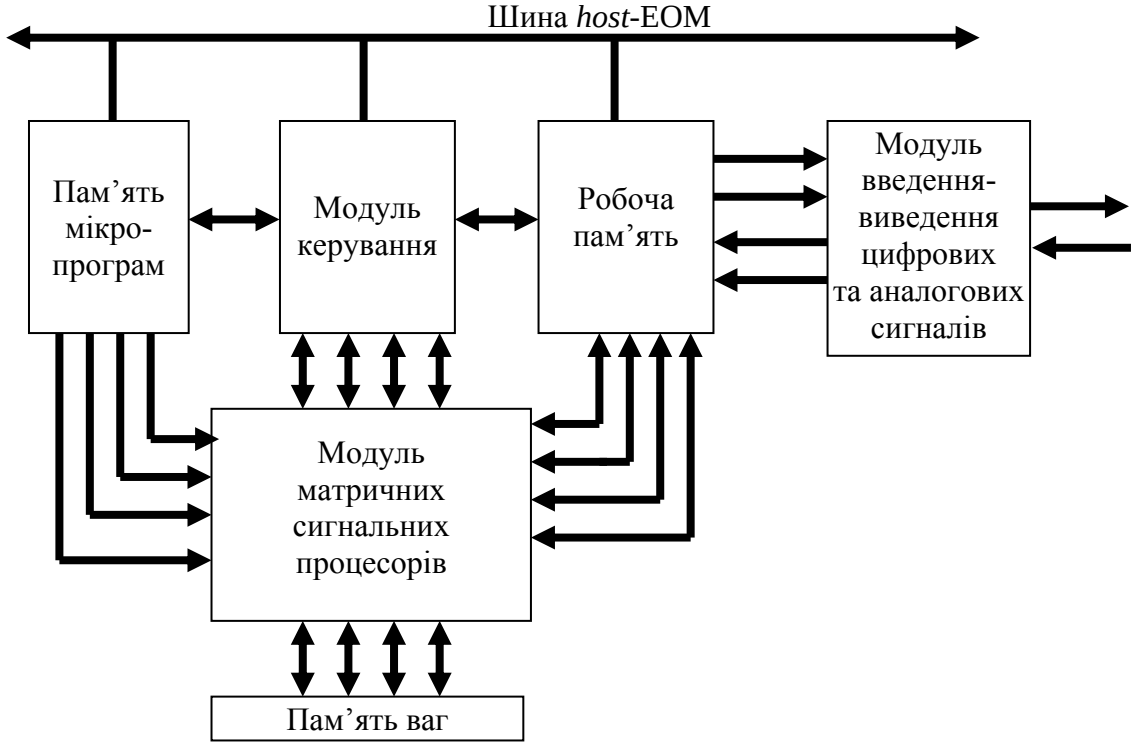


Рис. 10.7. Узагальнена функціональна схема віртуального нейроприскорювача

Елементну базу нейрообчислювачів третього напряму – саме нейрокомп'ютерів – являють собою *нейрочипи*. Нейрочипи бувають цифрові, аналогові й гібридні. Вони також можуть включати схеми настроювання ваг під час навчання, а можуть не мати таких схем і передбачати зовнішнє завантаження ваг. Характеристики деяких поширених нейрочипів наведено в табл. 10.3.

У табл. 10.3 оцінку продуктивності нейрообчислювачів замість традиційних показників *MIPS* і *MFLIPS* проводять за іншим показником – *CPS* (*Connections Per Second*) – кількістю з'єднань за секунду. З'єднання – це множення значень входу на вагу і додавання із нагромадженням.

Таблиця 10.3. **Характеристики нейрочипів**

Тип, фірма	Максимальна кількість			Розрядність		Продуктивність			
	синапсів	нейронів	шарів	входів	ваг	*CPS*	*CPSPW*	*CPPS*	*CUPS*
Цифрові									
*MD*1220 *Micro Device*	64	8	8	1	16	9 М	1 М	142 М	–
*NLX*420 *Adaptive Logic*	1 М	16	16	16	16	10 М	20 К	640 М	–
Lneuro 1,0 *Philips*	1 536	16	192	16	16	26 М	26 К	1,6 Г	32 М
Аналогові									
80170*NW Intel*	2 банки 64×80	64	2	–	–	2 Г	30 М	–	–
Гібридні									
CLNN-32 *Bellcore*	406	32	1	–	5	2 Г	5 М	–	–
CLNN-64 *Bellcore*	1 024	64	1	–	5	2 Г	2 М	–	–
ANNA AT&T	4 096	256	1	–	6	2,1 Г	5,1 К	–	–

Оскільки цей показник не враховує розрядності входів і ваг, іноді наводять такі показники:

– $CPSPW = CPS/N_w$, де N_w – кількість синапсів у нейроні;

– *CPPS* – кількість з'єднань примітивів за секунду, $CPPS = = CPS \times B_w \times B_s$, де B_w, B_s – розрядність ваг і синапсів.

Показник, що оцінює швидкість навчання, – *CUPS* (*Connections Update Per Second*) – кількість змінених значень ваг за секунду.

Цифрові нейрочипи. Одним з перших нейрочипів була BIC MD1220 фірми *Micro Device*. Цей кристал інтерпретує 8 нейронів і 8 зв'язків із 16-розрядними вагами, що зберігаються у внутрішньокристальній пам'яті, і 1-розрядними входами, які мають послідовні помножувачі. Тривалість такту – 7,2 мкс, що забезпечує продуктивність 9 *MCPS*. Із цих нейрочипів їх каскадним з'єднанням можна побудувати нейрокомп'ютери.

Фірма *Adaptive Logic* випускає нейрочип *NLX*420 із шістнадцятьма процесорними елементами (ПЕ), кожний із яких має 32-розрядний суматор. Ваги і входи завантажують як 16-розрядні слова, але їх можна використати як 16 1-розрядних слів, або як 4 4-розрядні, або як 2 8-розрядні, або як 1 16-розрядне слово. Ваги зберігаються поза чипом. Вхід загальний для всіх ПЕ, що дозволяє виконувати паралельно близько 16 операцій множення. Функції активації задає користувач. Кристали можна з'єднувати каскадно.

Кристал *Lneuro* 1,0 фірми *Philips* містить 16 ПЕ, кожний з яких може функціонувати як шістнадцять 1-розрядних, вісім 2-розрядних, чотири 4-розрядні, два 8-розрядні або один 16-розрядний ПЕ. Чип має 1 кбайт пам'яті ваг, що дозволяє використовувати 1 024 8-розрядних або 512 16-розрядних вагових коефіцієнтів. Функція активації реалізується поза чипом, що дозволяє під час каскадування інтерпретувати великі мережі.

Аналогові нейрочипи використовують аналогові схеми – суматори з аналоговими входами, ваги яких теж задають аналоговим способом. Ці чипи зазвичай менші і простіші за цифрові. З другого боку, забезпечення відповідної точності потребує ретельного проектування і виготовлення. Кристал фірми *Intel* 80170*NW* містить 64 нейрони і 2 банки 64×80 ваг. Можливі декілька мережних конфігурацій. Чип має 64 аналогові входи (0–3 В) і 16 внутрішніх зсувів. На кристалі можна реалізувати двошарову мережу з 64 входами, 64 внутрішніми і 64 вихідними нейронами. Інші конфігурації містять тришарові мережі або одношарову з 128 входами. Точність ВІС становить 5–6 розрядів для ваг і виходів.

Гібридні нейрочипи використовують комбінацію аналогового і цифрового підходів. Наприклад, входи можуть бути аналоговими, ваги можуть завантажуватися як цифрові і виходи можуть бути цифровими.

Чипи *CLNN*-32, *CLNN*-64 фірми *Bellcore* містять 32 нейрони. Входи, виходи і внутрішня обробка сигналів – аналогові, а 5-розрядні ваги – цифрові.

Чип *ANNA* фірми *AT&T* переважно цифровий, але всередині використовує конденсаторні заряди для зберігання ваг. Чип містить 4 096 ваг. Кількість нейронів варіюється від 16 до 256 із кількістю входів нейрона 256 або 16 відповідно. Ваги мають точність 6 розрядів.

Є нейрочипи, у яких використовують подання даних частотою або шириною імпульсів.

Контрольні запитання

1. Назвіть типи і наведіть приклади обчислювальних систем.
2. До якого типу обчислювальних систем належать нейронні обчислювачі?
3. Якими показниками оцінюють продуктивність нейромереж?
4. Як порівняти продуктивність традиційних фоннеймановських комп'ютерів і нейромереж?
5. Які основні напрями побудови нейронних обчислювачів?
6. Дайте визначення нейроемулятора, нейроприскорювача, нейрокомп'ютера. Порівняйте ці пристрої за апаратною реалізацією і швидкодією.
7. Назвіть основні типи нейрочипів.
8. Зазначте особливості сигнальних процесорів, що підвищують ефективність інтепретації нейромережевих алгоритмів.

СПИСОК ЛІТЕРАТУРИ

До розділу 1

1. *Акушский И. Я.*, *Юдицкий Д. И.* Машинная арифметика в остаточных классах. – М.: Сов. радио, 1968. – 440 с.
2. *Вычислительные* машины, системы и сети: Учебник / А. П. Пятибратов, С. Н. Беляев, Г. М. Козырева и др.; Под ред. проф. А. П. Пятибратова. – М.: Финансы и статистика, 1991. – 400 с.
3. *Зайцев В. Ф.* Кодирование информации в ЕС ЭВМ. – 2-е изд., перераб. и доп. – М.: Энергоатомиздат, 1990. – 144 с.
4. *Каган Б. М.* Электронные вычислительные машины и системы: Учеб. пособие для вузов. – 3-е изд., перераб. и доп. – М.: Энергоатомиздат, 1991. – 592 с.

До розділу 2

1. *Корнеев В. В.*, *Киселев А. В.* Современные микропроцессоры. – М.: НОЛИДЖ, 1998. – 240 с.
2. *Левенталь Л.* Введение в микропроцессоры. Программное обеспечение, аппаратные средства, программирование. – М.: Энергоатомиздат, 1983. – 464 с.
3. *Лысиков Б. Г.* Арифметические и логические основы автоматов: Учеб. для вузов. – 2-е изд., перераб. и доп. – М.: Высш. шк., 1980. – 336 с.
4. *Лю Ю-Чжен*, *Гибсон Г.* Микропроцессоры семейства 8086/8088. Архитектура, программирование и проектирование микрокомпьютерных систем: Пер. с англ. – М.: Радио и связь, 1987. – 512 с.
5. *Майоров В. Г.*, *Гаврилов А. И.* Практический курс программирования микропроцессорных систем. – М.: Машиностроение, 1989. – 204 с.
6. *Самофалов К. Г.*, *Корнейчук В. И.*, *Тарасенко В. П.* Электронные цифровые вычислительные машины. – К.: Вища шк., 1976. – 480 с.

До розділу 3

1. *Абель П.* Язык Асемблера для IBM PC и программирование: Пер. с англ. Ю. В. Сальникова. – М.: Высш. шк., 1992. – 447 с.
2. *Гольденберг А. М.*, *Малев В. А.*, *Малько Г. Б.* Цифровые устройства и микропроцессорные системы. Задачи и упражнения: Учеб. пособие для вузов. – М.: Радио и связь, 1993. – 256 с.
3. *Горбунов В. Д.*, *Панфилов Д. И.*, *Преснухин Д. Л.* Микропроцессоры. Основы построения микро-ЭВМ. – М.: Высш. шк., 1984. – 144 с.
4. *Микропроцессорный* комплект К1810. Справочная книга / Ю. М. Казаринов. – М.: Высш. шк., 1990. – 269 с.

До розділу 4

1. *Гук М.* Процессоры *Intel*: от 8086 до *Pentium II*. – СПб.: Питер, 1997. – 224 с.
2. *Гук М.* Процессоры *Pentium II*, *Pentium Pro* и просто *Pentium*. – СПб.: Питер Ком, 1999. – 288 с.
3. *Корнеев В. В.*, *Киселев А. В.* Современные микропроцессоры. – М.: НОЛИДЖ, 1998. – 240 с.
4. *Морс С. П.*, *Альберт Д. Д.* Архитектура микропроцессора 80286. – М.: Радио и связь. – 1990. – 304 с.

До розділу 5

1. *Лебедев О. Н.* Микросхемы памяти и их применение. – М.: Радио и связь, 1990. – 234 с.
2. *Микро-* и мини-ЭВМ / Е. П. Балашов, В. Л. Григорьев, Г. А. Петров и др.: Учеб. пособие для вузов. – Л.: Энергоатомиздат, 1984. – 376 с.

До розділу 6

1. *Алексенко А. Г.*, *Галицын А. А.*, *Иванников А. Д.* Проектирование радиоэлектронной аппаратуры на микропроцессорах. – М.: Радио и связь, 1984. – 272 с.
2. *Вершинин О. Е.* Применение микропроцессоров для автоматизации технологических процессов. – Л.: Энергоатомиздат, 1986. – 208 с.
3. *Микропроцессоры* / К. Г. Самофалов, О. В. Викторов, А. К. Кузняк и др.. – К.: Техніка, 1986. – 278 с.
4. *Справочник* по микропроцессорным устройствам / А. А. Молчанов, В. И. Корнейчук, В. П. Тарасенко и др. – К.: Техніка, 1987. – 228 с.

До розділу 7

1. *Гребнев В. В.* Однокристальные микро-ЭВМ семейства *AT*89 фирмы *Atmel*. – СПб.: ЭФО, 1998. – 76 с.
2. *Козаченко В. Ф.* Микроконтроллеры: Руководство по применению 18-разрядных микроконтроллеров *Intel MCD*-196/296 во встроенных системах управления. – М.: Изд-во «ЭКОМ», 1997. – 688 с.
3. *Липовецкий Г. П.*, *Литвинский Г. В.*, *Оксинь О. Н.* Однокристальные микро-ЭВМ. Семейство МК48, семейство МК51. Техническое описание и руководство по применению. – М.: МП «Бином», 1992. – 339 с.
4. *Сташин В. В.*, *Урусов А. В.*, *Мологонцева О. Ф.* Проектирование цифровых устройств на однокристальных микроконтроллерах. – М.: Энергоатомиздат, 1990. – 221 с.
5. *Atmel Corporation*. 8051 *Flash Microcontroller*. *Data Book*. *December* 1997.

До розділу 8

1. *Евстигнеев А. В.* Микроконтроллеры *AVR* семейства *Classic* фирмы *Atmel*. – М.: ДОДЭКА-XXI, 2002. – 288 с.
2. *Микроконтроллеры*: Сборник. – М.: ДОДЭКА-XXI, 1998. – Вып. 1. – 384 с.
3. http://www.atmel.com/atmel/products/prod23.htm
4. http://www.atmel.ru/Articles
5. http://www.gaw.ru/ table.doc
6. http://www.efo.ru/cgi-bin/go.pl?118
7. http://www.atmel.com/atmel/products/prod29.htm

До розділу 9

1. *DSP56800FM/D: DSP56800 Family Manual.*
2. *DSP56F801-7UM/D: DSP56F80x User's Manual.*
3. *DSP56824UM/D: DSP56824 User's Manual*.
4. *DSP56800WP1/D J.P. Gergen, P.Hoang, E.A. Cchemaly: Novel Digital Processing Architecture with Microcontroller Features, White Paper, Motorola*.
5. *DSP5680x Architecture Captures Best of DSP and MCU Worlds, Internal Paper, Motorola*.

До розділу 10

1. *Власов А. И.* Аппаратная реализация нейровычислительных управляющих систем // Приборы и системы управления. – 1999. – № 2. – С. 61–65.
2. *Галушкин А. И.* Некоторые исторические аспекты развития элементной базы вычислительных систем с массовым параллелизмом (80-е и 90-е годы) // Нейрокомпьютер. – 2000. – № 1. – С. 68–82.
3. *Корнеев В. В., Киселев А. В.* Современные микропроцессоры. – М.: НОЛИДЖ, 2000. – 320 с.
4. http://neurnews.iu4.bmstu.ru/
5. http://www.module.ru/.
6. http://www.analog.com/.