Міністерство освіти і науки України
Запорізька державна інженерна академія

ОСНОВИ МІКРОПРОЦЕСОРНОЇ ТЕХНІКИ

Методичні вказівки

до лабораторного практикуму та самостійної роботи
для студентів ЗДІА
спеціальності “Гідроенергетика”

рекомендовано до видання
на засіданні кафедри
“Гідроенергетика”
протокол № 4
від 23 листопада 2007.

Запоріжжя
2008

Основи мікропроцесорної техніки. Методичні вказівки до лабораторного практикуму для студентів ЗДІА спеціальності «Гідроенергетика» (7.090503). Уклад. В. В. Радченко, В. В. Назаренко, В. М. Чван. - ЗДІА, 2008. - 63 с.

Укладачі:
В. В. Радченко - зав. кафедрою
В. В. Назаренко - асистент
В. М. Чван - ст. викладач

Відповідальний за випуск -
зав. кафедрою «Гідроенергетика»
доцент В. В. Радченко.

ЗМІСТ

ВСТУП

Метою розробки методичних вказівок є практичне освоєння теоретичної частини курсу дисципліни «Основи мікропроцесорної техніки».

Цифрове подання і цифрова обробка інформації із застосуванням мікропроцесорної техніки є невід'ємною частиною знань для сучасного фахівця-енергетика. Знання в даній області необхідні для виконання ними робіт по впровадженню (експлуатації) на підприємствах гідроенергетики сучасних мікропроцесорних систем на базі промислових мікропроцесорних контролерів та іншого мікропроцесорного обладнання.

Сучасне впровадження автоматизації технологічних процесів ведеться на основі останніх досягнень мікроелектроніки - мікропроцесорної техніки. Широка номенклатура мікропроцесорів, що випускаються, їх багатофункціональність і швидкість роботи дозволяють успішно вирішувати задачі автоматизації з багаторівневим управлінням в рамках гнучких автоматизованих систем.

Мікропроцесор (МП) - програмно-керуємий пристрій, що здійснює процес обробки цифрової інформації і управління ним, побудований на одній або декількох інтегральних мікросхемах.

1 МЕТА РОБОТИ

В процесі виконання лабораторного практикуму студент повинен освоїти:

- елементи математичного апарату цифрової техніки;
- функції алгебри логіки;
- аналіз і синтез комбінаційних цифрових пристроїв;
- метод спрощення виразів для функцій алгебри логіки за допомогою карт Карно;
- мінімізація заданої логічної функції із застосуванням засобів обчислювальної техніки;
- програмну модель 8-розрядного мікропроцесора (МП);
- команди 8-розрядного МП;
- принципи виконання обчислень МП.

2 ТЕОРЕТИЧНА ЧАСТИНА

2.1 Системи числення.

Система зображення будь-яких чисел за допомогою обмеженого числа символів називається *системою числення.* Використані в системі числення символи називаються цифрами.

Існують різні системи числення, і від їх особливостей залежить наглядність представлення числа за допомогою цифр і складність виконання арифметичних операцій. Якщо в системі числення кожній цифрі в будь-якому місці числа відповідає одне і те ж значення - кількісний еквівалент, то така система числення називається непозиційною. Таким чином, для непозиційних систем числення місцеположення цифри в записі числа не має ніякої ролі. Прикладом непозиційної системи числення є римська система, в якій використовуються римські цифри I, V, X, L, C, M. Відповідно значення числа, наприклад, CCXXIV обчислюється таким чином: C=100, X=10, V=5,
I = 1. При цьому вага цифри не залежить від її місцеположення в записі числа, а знак залежить. Якщо цифра з меншою вагою стоїть зліва від цифри з великою вагою, то її знак «-», а якщо цифра з меншою вагою стоїть праворуч від цифри з великою вагою, то її знак «+». Загальним недоліком непозиційних систем числення є труднощі запису в таких системах великих чисел і трудності виконання арифметичних операцій, оскільки для цього використовуються громіздкі правила. Тому в цифровій техніці непозиційні системи практично не знайшли застосування.

У цифровій техніці використовуються позиційні системи числення. Система числення називається позиційною, якщо одна і та ж цифра має різне значення, яке визначається її позицією в послідовності цифр, що зображає число. Це значення міняється в однозначній залежності від позиції цифри по деякому закону. У позиційних системах число

$$x_m x_{m-1} \ldots x_2 x_1 x_0$$

буде рівне сумі

$$x_m P_m + x_{m-1} P_{m-1} + \ldots + x_2 P_2 + x_1 P_1 + x_0 P_0 = \sum_{i=1}^{m} x_i P_i$$

де $x_m, x_{m-1}, \ldots, x_2, x_1, x_0$ - символи, що позначають цілі числа;

$P_m, P_{m-1}, \ldots P_2, P_1, P_0$ - ваги, тобто кількісні значення кожної одиниці, які визначаються місцем відповідного символу в зображенні числа. При зображенні різних чисел в будь-якій конкретній системі числення символи $x_m, x_{m-1}, \ldots, x_2, x_1, x_0$ можуть міняти свій цифровий вираз, але ваги одиниць, розташованих на одних і тих же позиціях у всіх числах, зберігають одне і те ж

наперед обумовлене значення. Номер позиції, який визначає вагу одиниці, розташованої на цій позиції, називається розрядом.
Серед позиційних систем особливе значення мають системи числення, в яких вагу окремих розрядів є ряд членів геометричної прогресії із знаменником р. В цьому випадку число

$$x_m x_{m-1} \ldots x_1 x_0 , x_{-1} x_{-2} \ldots$$

матиме значення

$$x_m P^m + x_{m-1} P^{m-1} + \ldots + x_1 P^1 + x_0 P^0 + x_{-1} P^{-1} + x_{-2} P^{-2} .$$

У цій послідовності кома відділяє цілу частина числа від дробної, тобто коефіцієнти при позитивних ступенях, включаючи нуль, від коефіцієнтів при негативних ступенях. Кома опускається, якщо немає негативних ступенів.
Число різних символів - цифр, необхідних для запису довільного числа, рівно знаменнику геометричної прогресії *p.* В більшості випадків використовуються числа натурального ряду 0, 1, 2 ..., *(p-1).* Знаменник геометричної прогресії *p,* рівний відношенню ваги будь-якого розряду до ваги сусіднього справа розряду, називається *основою* системи числення. Відповідно кількість цифр, що вживаються в позиційній системі числення, рівна її основі.
У десятичній системі основа $p = 10$ і для запису чисел використовується десять цифр: 0, 1, 2...,9. Кожна цифра числа займає в нім певний розряд, який має вагові коефіцієнти для розрядів ліворуч від коми 10^0, 10^1, 10^2... і вправо від коми 10^{-1}, 10^{-2}, 10^{-3}... Таким чином, запис 547,359 в десятичній системі числення означає наступну кількість:

$$547{,}359 = 5 \cdot 10^2 + 4 \cdot 10^1 + 7 \cdot 10^0 + 3 \cdot 10^{-1} + 5 \cdot 10^{-2} + 9 \cdot 10^{-3}.$$

Позиційні системи числення мають ряд переваг перед непозиційними. Основною перевагою слід вважати зручність виконання таких арифметичних операцій, як додавання, віднімання, множення, ділення, вичислення кореня і ін. Тому в цифровій техніці, як правило, застосовуються позиційні системи числення. Вибір основи системи числення залежить від фізичних елементів, на основі яких будується те або інший пристрій. У цифровій техніці широко використовуються елементи з двома стійкими станами. У цих елементах відмінність між окремими фіксованими станами носить якісний, а не кількісний характер, завдяки чому представлення чисел з їх допомогою може бути реалізоване значно надійніше, ніж за допомогою елементів, в яких число чітко різних станів перевищує два. Зокрема, виконання елементу з десятьма чітко різними станами є складним технічним завданням. Вказана обставина являється однією з головних причин розповсюдження в цифровій техніці позиційних

систем з недесятичною основою, в першу чергу двоїчної, а також восьмиричної і шістнадцятиричної систем числення.

Найбільше розповсюдження в цифровій техніці має двоїчна система числення. У цій системі використовуються тільки дві цифри: 0 і 1. У двоїчній системі будь-яке число може бути представлене послідовністю двоїчних цифр:

$$N_2 = a_m a_{m-1} ... a_1 a_0 , a_{-1} a_{-2}$$

де a_i приймає значення або 0, або 1.

Цей запис відповідає сумі ступенів числа два, узятих з вказаними в ній коефіцієнтами

$$N_2 = a_m 2^m + a_{m-1} 2^{m-1} + ... + a_1 2^1 + a_0 2^0 + a_{-1} 2^{-1} + a_{-2} 2^{-2} ... \ .$$

Вага розрядів, що відлічуються ліворуч від коми, в цілій частині числа рівна відповідно 1, 2, 4, 8, 16 ..., вага ж розрядів правіше за кому в дробовій частині буде ½, ¼, $^1/_8$, $^1/_{16}$... . Наприклад, число 11010,112 відповідає наступній кількості:

$$11010,11_2 = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2}$$

яке, як випливає з приведеного розподілу його по ступенях числа 2, рівно десятичному числу 26,7510. У восьмиричній системі числення використовуються вісім цифр: 0, 1,2 ..., 7. Будь-яке число у восьмиричній системі представляється послідовністю, в якої цифри можуть приймати значення від 0 до 7. Вага розрядів цілої частини 1, 8, 64, 256 ..., у дробовій частині 1/8, 1/64, 1/256... Наприклад, восьмиричне число 756,25

$$756,25_8 = 7 \cdot 8^2 + 5 \cdot 8^1 + 6 \cdot 8^0 + 2 \cdot 8^{-1} + 5 \cdot 8^{-2}$$

рівно десятичному числу 494, 328125_{10}.

У шістнадцятиричній системі числення для зображення чисел використовуються 16 цифр: 0...15. При цьому, щоб одну цифру не зображати двома символами, доводиться вводити спеціальні позначення для цифр більше дев'яти. Для шести символів зазвичай використовуються букви латинського алфавіту: A, B, C, D, E, F, яким в десятичній системі відповідають числа 10, 11, 12, 13, 14, 15. Таким чином, шістнадцятиричне число A7B,C816 відповідає наступній кількості:

$$A7B,C8_{16} = 10 \cdot 16^2 + 7 \cdot 16^1 + 11 \cdot 16^0 + 12 \cdot 16^{-1} + 8 \cdot 16^{-2}$$

рівному десятичному числу $2683{,}78125_{10}$.

2.2 Переклад чисел з однієї системи числення в іншу.

Правила перекладу чисел з двоїчної системи у восьмиричну, шістнадцятиричну і назад достатньо прості, оскільки основа восьми- і шістнадцятиричної систем числення виражаються цілим ступенем два ($8=2^3$, $16=2^4$). Для перекладу чисел з восьмиричної системи в двоїчну достатньо кожну цифру восьмиричного числа представити трьохрозрядним двоїчним числом - тріадою. Наприклад:

$$762{,}35_8 = \underset{7}{111}\ \underset{6}{110}\ \underset{2}{010},\ \underset{3}{011}\ \underset{5}{101}$$

Перевод шістнадцятиричних чисел в двоїчну систему числення здійснюється представленням цифр шістнадцятиричного числа чотирьохрозрядними двоїчними числами - тетрадами. Наприклад:

$$A7B{,}C7_{16} = \underset{A}{1010}\ \underset{7}{0111}\ \underset{B}{1011},\ \underset{C}{1100}\ \underset{7}{0111}$$

При зворотному переводі чисел з двоїчної системи у восьми- і шістнадцятиричну системи необхідно розряди двоїчного числа, відлічуючи від коми вліво і вправо, розбити на групи по три розряди при переводі у восьмиричну систему або в групи по чотири розряди при переводі в шістнадцятиричну систему. Неповні крайні групи доповнюються нулями. Потім кожна двоїчна група представляється цифрою тієї ж системи числення, в яку переводиться число. Наприклад:

$$\underset{1}{001}\ \underset{7}{111},\ \underset{5}{101}\ \underset{2}{010} = 17{,}52_8 \qquad \underset{5}{0101}\ \underset{C}{1100},\ \underset{D}{1011}\ \underset{6}{011} = 5C{,}D6_{16}$$

У загальному випадку, при некратній основі, переклад числа, що містить цілу і дрібову частини:

$$N_p = x_m p^m + x_{m-1} p^{m-1} + \cdots + x_1 p^1 + x_0 p^0 + x_{-1} p^{-1} + \cdots + x_{-l} p^{-l}$$

з системи з основою p_1 в систему з основою p_2 можна виконати по універсальному алгоритму, при якому ціла і дробова частини приводяться до наступного вигляду:

$$N_{p\text{ц}} = ((\ldots((x_m p + x_{m-1}) p + x_{m-2}) p + \ldots) p + x_0),$$
$$N_{p\text{др}} = p^{-1}(x_{-1} + p^{-1}(x_{-2} + \ldots + p^{-1}(x_{-(l-1)} +$$
$$+ x_{-l} p^{-l})\ldots)).$$

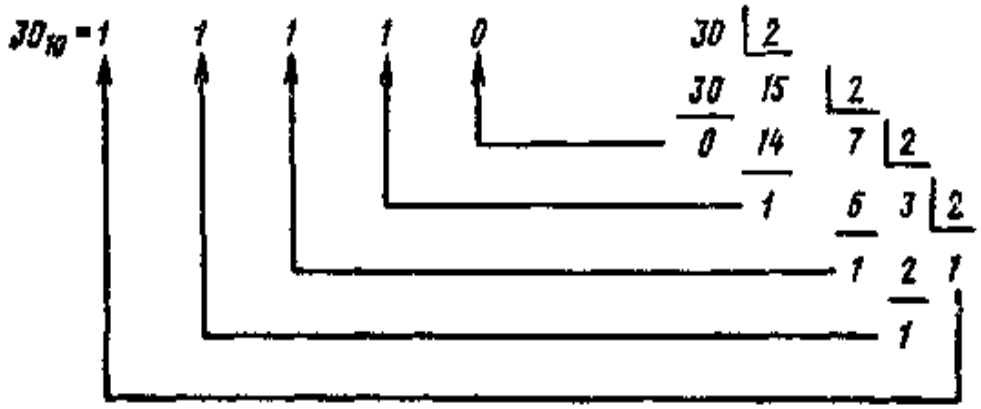
Згідно цьому алгоритму переклад числа складається з обчислювальних процесів двох видів:

- послідовного ділення цілої частини і що утворюються цілих часток на основу нової системи числення
- послідовного множення дробової частини і дробових частин добутків, що виходять, на ту ж нову основу, записану, як і в першому випадку, цифрами початкової системи числення.

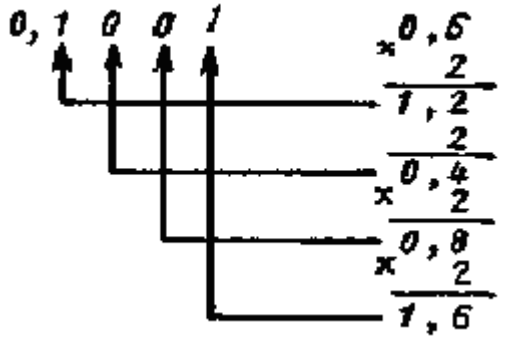
При переводі цілої частини числа залишки, що виходять в результаті процесу послідовного ділення, представляють цифри a_0, a_1... цілої частини числа в новій системі числення, записані цифрами початкової системи числення. Останній залишок є старшою цифрою переведеного числа.

При перекладі дробової частини числа цілі частини, що виходять при кожному множенні, не беруть участь в подальших множеннях. Вони представляють цифри в дробовій частині початкового числа в новій системі числення, зображені цифрами початкової системи. Значення першої цілої частини є першою цифрою після коми переведеного числа.

При перетворенні десятичних чисел в двоїчні - ціла частина повинна послідовно ділитися на 2, а дробова частина - умножатися на 2. Наприклад, при перекладі десятичнго числа 30,6 в двоїчне отримаємо цілу частина шляхом ділення:

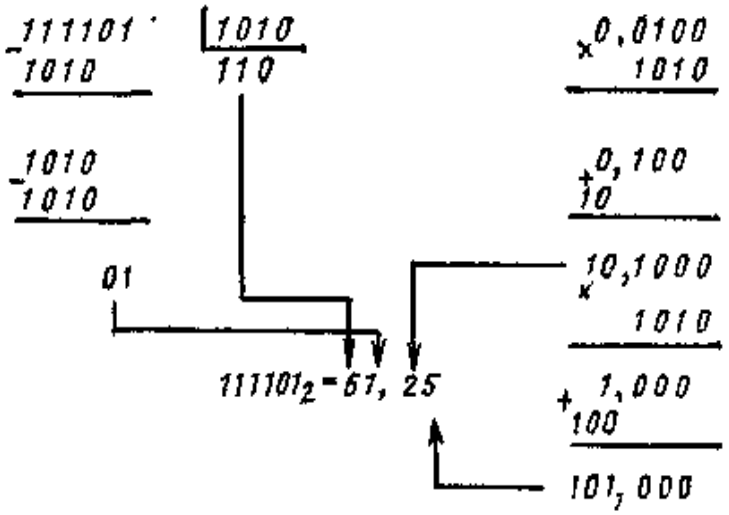


і дробову частину шляхом множення на 2:



В результаті маємо двоїчне число 11110, 1001... Якщо при перекладі дробової частини виходить періодичний дріб, то проводиться округлення, виходячи із заданої точності обчислення. У даному прикладі дріб визначений з точністю до п'ятого знаку після коми.

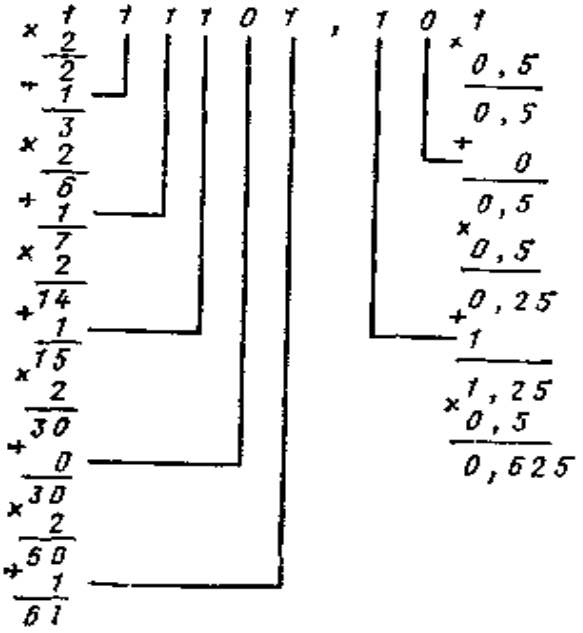
Аналогічно здійснюється переклад чисел з двоїчної системи в десятичну. Наприклад, при перекладі двоїчного числа 111101,01 в десятичне його цілу частину треба ділити на основу десятичної системи, яка в двоїчному записі буде 1010_2, а дріб умножати послідовно на це число:



При переводах чисел по розглянутому алгоритму необхідно виконувати послідовні перетворення, користуючись таблицями складання і множення початкової системи числення. При переході вручну до системи з великою основою такий процес виявляється незручним. Тому в цьому випадку застосовується інший алгоритм. Спочатку для перших чисел знаходиться добуток старшого розряду числа на основу початкової системи числення, до якого додається наступна цифра числа, що переводиться. Потім отримана сума також умножається на основу *p*, до отриманого добутку додається наступна цифра, і т.д. до останнього циклу, в якому здійснюється тільки збільшення

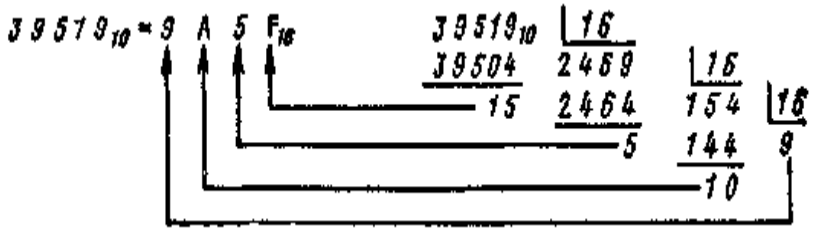
цифри молодшого розряду без подальшого множення. При цьому дії виконуються в новій системі числення.

Аналогічна послідовність дій застосовується і для переводу дробової частини, з тією лише різницею, що для множення використовується величина, зворотна основі і рівна p^{-1}. Наприклад, при перекладі двоїчного числа 111101,101 в десятичну систему:

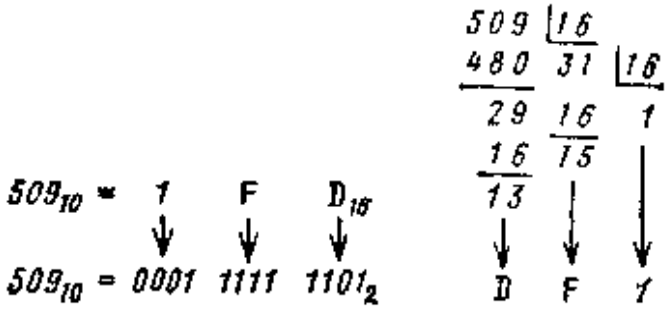


В результаті отримуємо десятичне число 61,625.

Користуючись стандартним алгоритмом, можна переводити числа з десятичної системи у восьми- і шістнадцятиричну системи і назад. При цьому ціла частина числа в десятичній системі послідовно ділиться на 16 або на 8, а дріб умножається відповідно на 16 або на 8. Наприклад, при перетворенні десятичного числа 3951910 в шістнадцятиричну систему отримуємо число $9A5F_{16}$.



Застосування шістнадцятиричної системи числення дозволяє переводити десятичні числа в двоїчні і назад. При цьому спочатку десятичне число переводиться в шістнадцятиричне, а потім шістнадцятиричне число замінюється двоїчним. Наприклад, для перекладу числа 509_{10} в двоїчну систему спочатку воно повинне бути перетворене в шістнадцятиричну систему



Аналогічно здійснюється зворотний переклад двоїчних чисел в десятичні. Як видно з приведеного прикладу, при використанні проміжного переходу до шістнадцятиричної системи числення істотно скорочується кількість операцій ділення, хоч і з'являється додатково операція по перетворенню цифр шістнадцятиричної системи в двоїчний код.

2.3 Двоїчна арифметика

Правила арифметики у всіх позиційних системах аналогічні, і в двоїчній позиційній системі числення виконання арифметичних дій над двоїчними числами задається за допомогою таблиць двоїчного додавання, віднімання і множення. Основною операцією, яка використовується в цифрових пристроях при виконанні різних арифметичних дій, є операція алгебраїчного додавання чисел, тобто додавання, в якому можуть брати участь як позитивні, так і негативні числа. Віднімання легко зводиться до додавання шляхом зміни на зворотний знак від'ємника, а операції множення і ділення також зводяться до алгебраїчного додавання і деяких логічних дій.

Додавання двох чисел в двоїчній системі числення виконується на основі таблиці двоїчного додавання:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10.

Двозначна сума в останньому випадку означає, що при додавання двох двоїчних цифр, рівних 1, в якому-небудь розряді двоїчного числа виникає перенесення в сусідній старший розряд. Це перенесення повинне бути додане до суми цифр, що утворилася в сусідньому розряді.

При додавання двох багаторозрядних двоїчних чисел цифри розрядів суми формуються послідовно, починаючи з молодшого розряду. Цифра молодшого розряду суми утворюється підсумовуванням цифр молодших розрядів доданків. При цьому окрім цифри розряду суми формується цифра перенесення в наступний, більш старший розряд, якщо обидва молодші розряди одиниці. Таким чином, в розрядах, починаючи з другого, можуть

підсумовуватися три цифри: цифри відповідного розряду доданків і перенесення, що поступає в даний розряд з попереднього.
Приклад складання двох багаторозрядних двоїчних чисел:

```
  1 1 0 1 1 0 1 - перший доданок
+
  1 0 0 1 1 1 1 - другий доданок
  0 1 0 0 0 1 0 - порозрядна сума без урахування
+                 перенесень
1   1 1  1      - перенесення
10 1 1 1 1 0 0 - остаточна сума
```

Безпосередньо під двома доданками записаний результат порозрядного складання без урахування перенесення. У тих розрядах, в яких обидва доданків рівні одиниці, порозрядна сума рівна 0. У цих розрядах утворилося перенесення в сусідній старший розряд, який відмічений в наступному рядку. В результаті складання рядка порозрядних сум з рядком перенесень виходить остаточна сума. При складанні порозрядної суми з перенесеннями зручно користуватися наступним правилом: якщо в результаті порозрядного підсумовування утворилася група одиниць, розташованих поряд, і в молодший розряд цієї групи поступає перенесення 1, то він переводить всі одиниці цієї групи в нулі, а найближчий за рядом одиниць 0 - в 1. Це правило можна використовувати при складанні наступних чисел:

```
  1 0 1 0 0 0 0 0 1 - перший доданок
+
  1 1 0 0 1 1 1 1 1 - другий доданок
  0 1 1 0 1 1 1 1 0 - порозрядна сума
+
1               1   - перенесення
1 0 1 1 1 0 0 0 0 0 - остаточна сума
```

Використання цього правила дозволяє прискорити формування остаточної суми.

Правила арифметичних операцій в восьмиричній і шістнадцятиричній системах також основані на порозрядній обробці чисел. При додавання для восьмиричної системи, коли порозрядна сума в молодшому розряді рівна восьми або більше – від неї віднімається вісім, решту заносять в молодший розряд, а в старший розряд переноситься одиниця. Для шістнадцятиричної системи, коли порозрядна сума в молодшому розряді рівна шістнадцяти або більше – від неї віднімається шістнадцять, решту заносять в молодший розряд, а в старший розряд переноситься одиниця. Наприклад:

- додавання в восьмиричній системі

```
  7 2 4 6
+ 1 3 3 6
-------
1 0 6 0 4
```

- додавання в шістнадцятиричній системі

```
  5 A 4 C
+ F 6 2 7
---------
1 5 0 7 3
```

де A=10, C=12, F=15.

Віднімання двох чисел в двоїчній системі виконується на основі таблиці двоїчного віднімання:

0 - 0 = 0
1 - 0 = 1
1 - 1 = 0
10 – 1 = 1.

Якщо при порозрядному відніманні доводиться віднімати з нуля в зменшуваному одиницю від'ємника, то робиться позика в сусідньому старшому розряді, тобто одиниця старшого розряду представляється як дві одиниці даного розряду. Віднімання в цьому випадку виконується відповідно до таблиці. Якщо в сусідньому розряді або в декількох старших розрядах стоять нулі, то позика робиться в найближчому старшому розряді, в якому стоїть одиниця. Ця одиниця представляється у вигляді суми числа, що складається з одиниці у всіх проміжних розрядах, в яких знаходилися нулі, і двох одиниць в даному розряді. Далі проводиться порозрядне віднімання відповідно до таблиці. Звичайно, що необхідності в додатковому заємі у всіх проміжних розрядах з'явитися не може. Наприклад, при відніманні чисел:

```
  1 1 1 0 0  0  1 1 - зменшуване
_ 1 1 0 1 1 (11) 1 1 - зменшуване з урахуванням заєма
  1 0 0 1 0  1  1 0 - від'ємник
  0 1 0 0 1  1  0 1 - різниця
```

У цифровій техніці операція віднімання з використанням заєма практично не застосовується (за винятком окремих пристроїв) і реалізується як алгебраїчне додавання із застосуванням спеціальних кодів для представлення негативних чисел. При цьому операція віднімання зводиться до операції простого арифметичного додавання за допомогою зворотного і додаткового кодів, використаних для представлення негативних чисел.

Зворотний код негативних двоїчних чисел може бути сформований за наступним правилом: цифри всіх розрядів, окрім знакового, замінюються на зворотні (інвертуються) - одиниці замінюються нулями, а нулі одиницями. У

знаковий розряд ставиться одиниця Зворотне перетворення із зворотного коду в прямий проводиться за тим же правилом. При використанні зворотного коду операція віднімання реалізується як арифметичне додавання позитивного числа, представленого в прямому коді, з негативним числом, представленим в зворотному коді. Наприклад, при відніманні з числа 10110 числа 01101 зменшуване представляється як позитивне число в прямому коді 0 10110, а від'ємник- як негативне число в зворотному коді 1 10010. У представленні чисел знакові розряди виділені напівжирним шрифтом. При виконанні операції арифметичного складання над цими числами отримуємо алгебраїчну суму:

0 1 0 1 1 0 - перший доданок в прямому коді
+
1 1 0 0 1 0 - другий доданок в зворотному коді
1 0 0 1 0 0 0
+ └──→ 1
0 0 1 0 0 1 - сума в прямому коді

Перенесення, що виникає із знакового розряду, при використанні зворотного коду повинен додаватися в молодший розряд суми. У даному прикладі зменшуване по модулю більше за від'ємник, тому сума алгебри позитивна і представлена в прямому коді. При зміні знаків доданків в приведеному прикладі на зворотні:

1 01001 - перший доданок в зворотному коді
+
0 01101 - другий доданок в прямому коді
1 10110 - сума в зворотному коді

результатом складання буде негативне число і воно буде представлено в зворотному коді.

Додатковий код негативних двоїчних чисел може бути сформований за наступним правилом: цифри всіх розрядів, окрім знакового, інвертуються, і в молодший розряд додається одиниця. Додатковий код може бути отриманий і із зворотного шляхом додавання одиниці до молодшого розряду зворотного коду. При цьому в знаковий розряд негативного числа в додатковому коді ставиться одиниця. Зворотне перетворення з додаткового коду в прямий проводиться за тим же правилом.

При використанні додаткового коду для віднімання двоїчних чисел з попереднього прикладу отримаємо:

0 1 0 1 1 0 - перший доданок в прямому коді
+
1 1 0 0 1 1 - другий доданок в додатковому коді
0 0 1 0 0 1 - сума в прямому коді

При складанні складаються цифри знакових розрядів з відкиданням перенесення, що виникає з цього розряду. Алгебраїчна сума, отримана в результаті складання, є позитивним числом і тому представлена в прямому коді. Якщо знаки доданків міняються на зворотні:

1 0 1 0 1 0 - перший доданок в додатковому коді
+
0 0 1 1 0 1 - другий доданок в прямому коді
1 1 0 1 1 1 - сума в додатковому коді

то результат складання є негативне число і воно виявляється представленим в додатковому коді.

При складанні алгебри двоїчних чисел в сумі, що утворилася, можливе переповнення розрядної сітки, яке полягає в тому, що в результаті операції сума містить більше число розрядів, чим число розрядів в пристрої, призначеному для їх зберігання. Для виявлення переповнення розрядної сітки використовується модифікований код. У нім два знакові розряди і в обох розрядах позитивні числа містять нулі, а негативні числа - одиниці. Виконання операцій підсумовування з використанням модифікованого додаткового або модифікованого зворотного коду проводиться за сформульованими вище правилами. Якщо результат підсумовування містить в знакових розрядах комбінації 01 або 10, то це служить ознакою переповнення розрядної сітки. Наприклад, при складанні чисел:

00 11011 - перший доданок в прямому модифікованому коді
+
11 01011 - другий доданок в додатковому модифікованому коді
00 00110 - сума в прямому модифікованому коді

переповнення розрядної сітки не виникає. Перенесення із старшого знакового розряду відкидається. При складанні чисел 00 10110 і 00 11011:

00 1 0 1 1 0
+
00 1 1 0 1 1
01 1 0 0 0 1

у знакових розрядах результату підсумовування виникає комбінація 01, що свідчить про переповнення розрядної сітки і помилковості зафіксованого результату. Виникнення помилки пов'язане з тим, що при підсумовуванні позитивних чисел перенесення із старшого розряду виявилося зафіксованим в

другому із знакових розрядів. Для реєстрації результату підсумовування в даному прикладі потрібно шість розрядів (окрім знакових). При підсумовуванні негативних чисел також можливе переповнення розрядної сітки:

11 010011
+
11 100011
10 110110

В цьому випадку комбінація 10 в знакових розрядах указує на переповнення розрядної сітки.

Множення двоїчних багаторозрядних чисел включає операції - визначення знаку добутку і визначення його абсолютної величини. Знаковий розряд може бути отриманий підсумовуванням цифр знакових розрядів співмножників без формування перенесення:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 (без формування перенесення)

При неспівпаданні цифр виходить 1, що відповідає знаку добутку двох співмножників з різними знаками.

Абсолютна величина значення добутку визначається шляхом множення чисел без урахування їх знаків. Множення багаторозрядних двоїчних чисел проводиться на основі таблиці двоїчного множення:

0 x 0 = 0
0 x 1 = 0
1 x 0 = 0
1 x 1 = 1.

При множенні двох двоїчних чисел множене послідовно умножається на кожну цифру множника, починаючи або з молодшою, або із старшою, і для урахування ваги відповідної цифри множника зрушується або вліво, якщо множення проводиться, починаючи з молодшого розряду множника, або вправо, якщо множення проводиться, починаючи із старшого розряду множника, на таке число розрядів, на яке відповідний розряд множника зміщений щодо молодшого або старшого розряду.

Множення, що виходять в результаті, і зміщені часткові добутки після підсумовування дають повний добуток. Особливість множення двоїчних чисел полягає в тому, що частковий добуток може бути або зміщений на відповідне

число розрядів множеним, якщо відповідна цифра множника рівна 1, або нулем, якщо відповідна цифра множника рівна 0:

```
   10111    - множене
 x  1101    - множник
   10111    - перший частковий добуток
 +  00000   - другий частковий добуток
 + 10111    - третій частковий добуток
 + 10111    - четвертий частковий добуток
100101011   - добуток
```

Той же результат можна отримати при множенні, починаючи із старших розрядів множника:

```
  10111
x  1101
  10111
+ 10111
+   00000
+    10111
100101011
```

У цифрових пристроях процес додавання часткових добутків має послідовний характер: формується один з часткових добутків, до нього з відповідним зрушенням додається наступний частковий добуток, до отриманої суми додається з відповідним зрушенням черговий частковий добуток, і т. д., поки не виявляться підсумованими всі часткові добутки і не буде отримано повний добуток.

Приклад множення чисел цим методом:

```
   10111    - четвертий частковий добуток
  101110    - зміщення на розряд вліво
+  10111    - третій частковий добуток
 1000101    -додавання третього часткового добутку
10001010    - зміщення на розряд вліво
+   00000   - другий частковий добуток
 10001010   - з додавання другого часткового добутку
100010100   - зміщення на розряд вліво
+    10111  - перший частковий добуток
100101011   - додавання першого часткового добутку
```

При такому методі всі часткові добутки додаються з необхідними зміщеннями один щодо одного, завдяки чому утворюється раніше приведений результат множення цих чисел.

При множенні дробових чисел менше одиниці множення зручніше починати з молодшого розряду множника. Так, при перемножуванні дробових чисел 0,10111 і 0,1101 отримаємо:

```
  0,10111:          - перший частковий добуток
  0,01011: 1        - зміщення на розряд вправо
+  00000:           - другий частковий добуток
  0,01011:          - збільшення другого часткового добутку
  0,00101: 11       - зміщення на розряд вправо
+   10111:          - третій частковий добуток
  0,11100: 11       - збільшення третього часткового добутку:
  0,01110: 011      - зміщення на розряд вправо
+   10111:          - четвертий частковий добуток
  1,00101: 011      - збільшення четвертого часткового добутку
  0,10010: 1011     - зміщення на розряд вправо
```

Якщо потрібно зберегти всі розряди в добутку, то в розрядній сітці пристрою повинне бути передбачено число розрядів, рівне сумі числа розрядів множеного і множника. Проте при множенні дробових чисел часто в добутку потрібно мати те ж число розрядів, що і в множеному. У такому наближеному представленні результату не фіксуються цифри розрядів при зміщеннях, що висуваються правіше за вертикальну лінію, показану в приведеному вище прикладі. Таким чином, цифри молодших розрядів виявляться втраченими і буде отриманий наближений результат 0,100101. Далі відкидається останній з розрядів, і якщо цей розряд містить 1, то 1 додається до наступного розряду для округлення результату. Отже, отриманий результат 0,10011.

Якщо множене, або множник, або обидва разом містять і цілу і дробову частини, то коми в множеному і множнику не враховуються, вони умножаються як два цілі числа і від отриманого добутку справа відділяються комою $m+n$ розрядів, де n - число дробових розрядів множеного, а m - число дробових розрядів множника.

Ділення двоїчних багаторозрядних чисел включає дві операції - визначення знаку частки і визначення її абсолютної величини.

Знаковий розряд частки може бути отриманий, як і знаковий розряд добутку, підсумовуванням цифр знакових розрядів ділимого і дільника без формування перенесення. Абсолютна величина частки визначається діленням чисел без урахування їх знаків.

Ділення починається з того, що від ділимого зліва відділяється група розрядів, причому кількість розрядів в цій групі повинна або дорівнювати кількості розрядів в дільнику, або бути на один розряд більше. Якщо відділення такої групи можливе, в старший розряд частки записується 1, інакше в розряд одиниць частки записується нуль. Якщо виявилося, що частка містить цілу частину, то утворюється нова група розрядів шляхом віднімання з виділеної групи дільника і приписування до різниці чергової цифри ділимого. Якщо в результаті вийшло число, що перевищує дільник, то в частку записується 1, інакше наступна цифра буде рівна 0.

Надалі виконується ряд однакових циклів. Якщо остання цифра частки була рівна 1, то нова група утворюється відніманням дільника з попередньої групи і приписуванням чергової цифри ділимого. Якщо остання цифра частки 0, то для утворення нової групи досить приписати до попередньої групи чергову цифру ділимого. Остання цифра цілої частини частки виходить тоді, коли після визначення чергової цифри частки 1 або 0 в ділимому не залишиться більше цифр для того, щоб приписувати їх до різниці між попередньою групою і дільником або до самої попередньої групи. Після цього починається виділення дробових членів частки. Воно відрізняється від обчислення цілих членів тільки тим, що замість чергових цифр ділимого до попередніх груп приписуються нулі. Розглянемо приклади, в яких ділиме більше і менше дільника:

```
 11010111 |10110
-10110    1001, 110
------
  100111
- 10110
------
  100010
- 10110
------
   11000
- 10110
------
Остаток 1000
```

```
 1011001   |11011010
-101100100 0,0110
 11011010
---------
 100010100
- 11011010
---------
  11101000
- 11011010
---------
Остаток 111000
```

У цифрових пристроях при виконанні операції ділення так само, як і при виконанні операції алгебраїчного додавання, використовується додатковий і модифікований коди.

2.4 Елементи алгебри логіки. Синтез логічних схем на елементах комбінаційного типу

Для опису схем комбінаційних цифрових пристроїв використовується математичний апарат булевих функцій - алгебра логіки. Змінні x_1, x_2 ., x_n називаються двоїчними, якщо вони можуть приймати тільки значення: 0 або 1.

Функція від двоїчних змінних $f(x_1, x_2, ..., x_n)$ називається булевою, якщо вона, також як і її аргумент, приймає тільки два значення: 0 або 1. Зв'язки між вхідними і вихідними сигналами в комбінаційних схемах аналітично описуються булевими функціями. Існують різні способи завдання або представлення булевих функцій: словесний опис функцій, табличний спосіб (функція представляється у вигляді таблиць істинності), алгебраїчний спосіб, при невеликій кількості змінних - за допомогою карт Карно.

Від таблиць істинності можна перейти до форми алгебраїчного представлення функцій. У такій формі зручно проводити різні перетворення функцій, наприклад з метою їх мінімізації. Основні булеві функції одної і двох змінних, їх позначення і найменування приведені в таблиці 2.1, графічне зображення на рис.2.1.

Таблиця 2.1. - Основні функції алгебри логіки для однієї і двох змінних

Функції	Аргумент X=0 У=0	Аргумент X=0 У=1	Аргумент X=1 У=0	Аргумент X=1 У=1	Позначення функції	Найменування функції
$F_1(x)$	0	0	0	0	0	Константа «0»
$F_2(x)$	1	1	1	1	1	Константа «1»
$F_3(x)$	0	0	1	1	x	Змінна «x»
$F_4(x)$	1	1	0	0	$\bar{x}$	Інверсія – «НЕ»
$F_5(x.y)$	0	1	1	1	$x \vee y$	Диз'юнкція – «АБО»
$F_6(x.y)$	0	0	0	1	$x \cdot y$	Кон'юнкція –«І»
$F_7(x.y)$	0	1	1	0	$x \oplus y$	Складання по модулю 2
$F_8(x.y)$	1	0	0	0	$x \downarrow y$; $(\overline{x \vee y})$	Стрілка Пірса (АБО-НЕ)
$F_9(x.y)$	1	1	1	0	$x \mid y$; $(\overline{x \cdot y})$	Штрих Шеффера (І-НЕ)
$F_{10}(x.y)$	1	0	0	1	$x \infty y$	Рівнозначність
	АБО		І		НЕ	

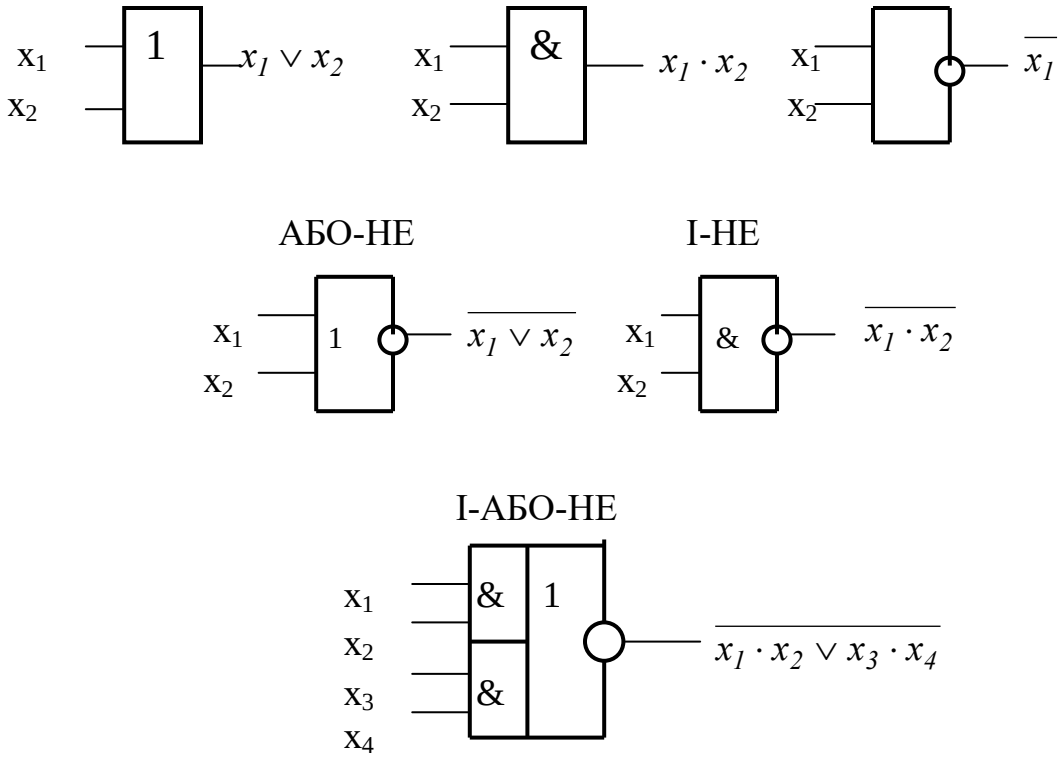


Рис.2.1 - Графічне позначення логічних елементів на схемах

Основні теореми алгебри логіки:

$x \vee 0 = x$ $\quad$ $x \cdot 1 = x$

$x \vee 1 = 1$ $\quad$ $x \cdot x_{...} x = x$

$x \vee x \vee x_{...} \vee x = x$ $\quad$ $\bar{x} \cdot x = 0$

$\bar{x} \vee x = 1$ $\quad$ $\bar{\bar{x}} = x$

Теореми для двох змінних і більше:

$x \vee y = y \vee x$; $\quad x \cdot y = y \cdot x$ (закон переміщення),

$x \vee y \vee z = x \vee (y \vee z) = (x \vee y) \vee z$ (сполучний закон)

$x \cdot y \cdot z = x \cdot (y \cdot z) = (x \cdot y) \cdot z$

$x(y \vee z) = x \cdot y \vee x \cdot z$ (розподільний закон)

$\overline{x \vee y} = \bar{x} \cdot \bar{y}$; $\quad \overline{x \vee y \vee z} = \bar{x} \cdot \bar{y} \cdot \bar{z}$ (теорема де-Моргана)

$\overline{x \cdot y} = \bar{x} \vee \bar{y}$; $\quad \overline{xyz} = \bar{x} \vee \bar{y} \vee \bar{z}$

Якщо система містить функції $F_1(x,y) = x \cdot y$ (кон'юнкція), $F_2(x,y) = x \vee y$ (диз'юнкція), $F_3(x) = \bar{x}$ (заперечення), то вона є функціонально

повною, тобто за допомогою даного набору логічних функцій можна реалізувати функції алгебри логіки будь-якого вигляду.

Проте з цієї системи можна виключити деякі функції без порушення функціональної повноти. Функціонально повною буде також система, що складається з однієї єдиної булевої функції «штрих Шеффера», - $F(x,y)=\overline{x\cdot y}$. У цій системі інверсію, диз'юнкцію, кон'юнкцію отримують, використовуючи закони і теореми алгебри логіки:

$$\bar{x}=\overline{x\cdot x};\qquad x\vee y=\overline{\overline{(x\cdot x)}\,\overline{(y\cdot y)}}=\overline{\bar{x}\cdot\bar{y}}=x\vee y;\qquad x\cdot y=\overline{\overline{x\cdot y}}=x\cdot y.$$

Реалізація логічних елементів за допомогою елементів І-НЕ «штрих Шеффера» показана на рис.2.2.

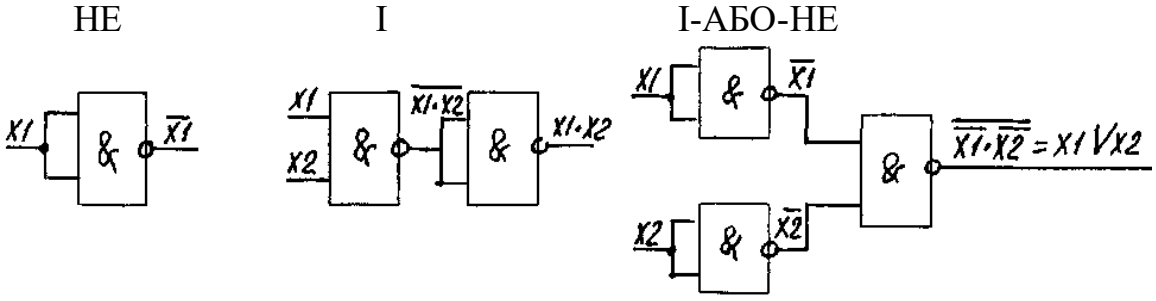


Рис.2.2 - Реалізація різних логічних функцій за допомогою елементів І-НЕ:

Можна показати, що функціонально повною є система, що складається з булевої функції «стрілка Пірса».

Технічний аналог булевої функції - комбінаційна схема, що виконує відповідне цій функції перетворення інформації.

Елементарні логічні операції над двоїчними змінними реалізуються електронними схемами, які називаються електронними логічними елементами або просто логічними елементами (ЛЕ). Число входів ЛЕ відповідає числу аргументів відповідної їм булевої функції. Один і той же закон перетворення інформації можна реалізувати, використовуючи різні типи комбінації ЛЕ і зв'язки між ними.

Для набору ЛЕ можна ввести поняття функціональної повноти, подібно до того, як це було зроблено для випадку системи булевих функцій. Набір ЛЕ володіє функціональною повнотою, якщо за допомогою кінцевого числа цих елементів можна побудувати схему з будь-яким законом функціонування. Будь-яка комбінаційна схема може бути побудована із застосуванням лише трьох видів логічних елементів (АБО, НЕ, І), сукупність яких є функціонально повною системою.

Логічні функції, що є диз'юнкціями окремих членів, кожний з яких, у свою чергу, є деякою функцією, що містить тільки кон'юнкції і інверсії, називаються *логічними функціями диз'юнктивної форми.* Форма представлення диз'юнктивної функції, в якій інверсія застосовується лише безпосередньо до аргументів, але не до складніших функцій від цих аргументів, називається *диз'юнктивною нормальною формою* представлення функцій (ДНФ), наприклад:

$$F(x_1, x_2, x_3) = x_1.\overline{x_2} \vee x_2 \cdot x_3 \vee \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3}\,.$$

Якщо ж кожен член диз'юнктивної нормальної функції від *n* аргументів містить всі ці *n* аргументів, частина з яких входить в нього з інверсією, а частина - без неї, то така форма представлення функції називається *досконалою диз'юнктивною нормальною формою* (ДДНФ);

Наприклад, функція задана в табличній формі:

X_1	X_2	X_3	У
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Тоді ДДНФ буде:

$$У = \overline{x_1} \cdot x_2 \cdot x_3 \vee x_1 \cdot \overline{x_2} \cdot x_3 \vee x_1 \cdot x_2 \cdot \overline{x_3} \vee x_1 \cdot x_2 \cdot x_3\,.$$

Для реалізації отриманої функції необхідно мати чотири 3-входових елементів «І» і один 4-входовий елемент «АБО» та 3 елементи НЕ (див. рис.2.3).

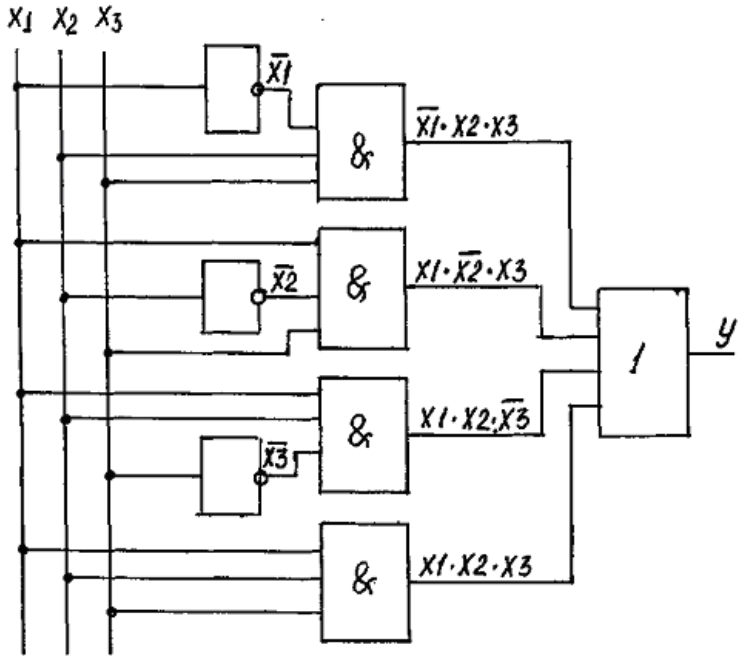


Рис.2.3 - Функціональна схема для виконання заданої функції алгебри логіки

Логічні функцій, такі, що є кон'юнкцією окремих членів, кожний з яких є функція, що містить тільки диз'юнкції і інверсії, називаються *логічними функціями кон'юнктивної форми.* По аналогії з диз'юнктивними формами можливі кон'юнктивні нормальні форми (КНФ) і досконалі кон'юнктивні нормальні форми (ДКНФ).

Можливість запису функцій в диз'юнктивних формах і кон'юнктивних формах визначається виходячи з наступного. Розглянемо довільну логічну функцію від *n* аргументів типу конституенти одиниці, яка повністю визначається завданням набору, що обертає її в одиницю. У цьому наборі деякі аргументи можуть бути рівними «1», а інші рівні «0». Складемо кон'юнкцію від всіх *n* аргументів, причому ті аргументи, які у вказаному наборі рівні «0», візьмемо з інверсією, а аргументи рівні «1» - без інверсії. Якщо всі аргументи відповідатимуть заданому набору, то функція перетворюється в кон'юнкцію *n* одиниць і буде рівна «1». Для решти всіх наборів хоч би один аргумент відрізняється від заданого набору, а значить, і вся кон'юнкція перетворюється в «0». Таким чином, довільна функція від *n* аргументів типу конституенти «1» може бути виражена через кон'юнкцію і інверсію. Наприклад, функція чотирьох аргументів, яка перетворюється в «1» при *x1=0, x2=1, x3=1, x4=0* і в «0» на решті всіх наборів, може бути записана у вигляді:

$$F(x_1,x_2,x_3,x_4)=\overline{x_1}\cdot x_2\cdot x_3\cdot\overline{x_4}.$$

Аналогічно можна показати, що довільну логічну функцію від *n* аргументів типу конституенти нуля можна виразити через диз'юнкцію і інверсію. Наприклад, якщо функція чотирьох аргументів при

$$x1=1,\ x2=0,\ x3=1,\ x4=1$$

пертворюється в «0», а на решті всіх наборів рівна «1», то вона матиме вигляд:

$$F(x_1,x_2,x_3,x_4)=\overline{x_1}\vee x_2\vee\overline{x_3}\vee\overline{x_4}\ .$$

Довільна функція може бути виражена через функції диз'юнкції, кон'юнкції і інверсії як у вигляді ДДНФ чи ДКНФ, так і у вигляді ДНФ чи КНФ.

Функція у вигляді ДДНФ може бути отримана на основі таблиці істинності при використанні наступного правила запису ДДНФ. Необхідно записати стільки членів у вигляді кон'юнкцій всіх аргументів, скільки одиниць містить функція в таблиці. Кожна кон'юнкція повинна відповідати набору аргументів, що обертають функцію в «1», і якщо в цьому наборі значення аргументу рівне «0», то в кон'юнкцію входить інверсія даного аргументу. Наприклад, по таблиці істинності функції:

X_1	0	1	0	1	0	1	0	1
X_2	0	0	1	1	0	0	1	1
X_3	0	1	0	0	1	1	1	1
$F(x_1,x_2,x_3)$	0	1	0	1	1	0	1	0

можна записати функцію в ДДНФ у вигляді:

$$F(x_1x_2x_3x_4)=x_1\cdot\overline{x_2}\cdot x_3\vee x_1\cdot x_2\cdot\bar{x}_3\vee\overline{x_1}\cdot\overline{x_2}\cdot x_3\vee\overline{x_1}\cdot x_2\cdot x_3.$$

Функція у вигляді ДКНФ також може бути отримана безпосередньо з таблиці істинності при використанні наступного правила. Необхідно записати стільки кон'юктівних членів, що є диз'юнкціями всіх аргументів, при скількох наборах функція рівна «0», і якщо в наборі значення аргументу рівне «1», то в диз'юнкцію входить інверсія цього аргументу. Наприклад, по приведеній вище таблиці істинності можна записати функцію в ДКНФ в наступному вигляді:

$$F(x_1x_2x_3x_4)=(x_1\vee x_2\vee x_3)\cdot(x_1\vee\bar{x}_2\vee x_3)\cdot(\overline{x_1}\vee x_2\vee\bar{x}_3)\cdot(\overline{x_1}\vee\bar{x}_2\vee\bar{x}_3).$$

Слід зазначити, що будь-яка функція має єдині ДДНФ і ДКНФ. У ряді випадків така форма запису не є найпростішою для виразу заданої функції в аналітичній формі і можна спростити логічний вираз не порушуючи значення

функції. Методи такого спрощення функції називають методами мінімізації (синтезу). В результаті мінімізації логічні функції можуть бути представлені в ДНФ або в КНФ з мінімальним числом членів і з мінімальним числом аргументів в кожному членові. Для спрощення виразів функцій алгебри логіки розроблені як графічні (за допомогою карт Карно), так і алгебраїчні.

Приклад рішення: фунція представлена слідуючою таблицею істинності -

X_1	0	0	0	0	1	1	1	1
X_2	0	0	1	1	0	0	1	1
X_3	0	1	0	1	0	1	0	1
$F(x_1,x_2,x_3)$	0	0	0	1	0	1	1	1

У ДДНФ функція буде виражена в наступному вигляді:

$$Y = \overline{x_1} \cdot x_2 x_3 \vee x_1 \overline{x_2} \cdot x_3 \vee x_1 x_2 \overline{x_3} \vee x_1 x_2 x_3 .$$

Після мінімізації з використанням основних теорем алгебри логіки отримаємо:

$$Y = \overline{x_1} \cdot x_2 \cdot x_3 \vee x_1 \cdot \overline{x_2} \cdot x_3 \vee x_1 \cdot x_2 \cdot \overline{x_3} \vee x_1 \cdot x_2 \cdot x_3 \vee x_1 \cdot x_2 \cdot x_3 =$$
$$= x_2 \cdot x_3 (\overline{x_1} \vee x_1) \vee x_1 \cdot x_3 (\overline{x_2} \vee x_2) \vee x_1 x_2 (\overline{x_3} \vee x_3) = x_1 x_2 \vee x_1 x_3 \vee x_2 x_3 .$$

Для спощення використана теорема алгебри логіки $\overline{x} \vee x = 1$.

Схема з'єднань ЛЕ із застосуванням 2-, 3-входовых елементів вказана на рис.2.4.

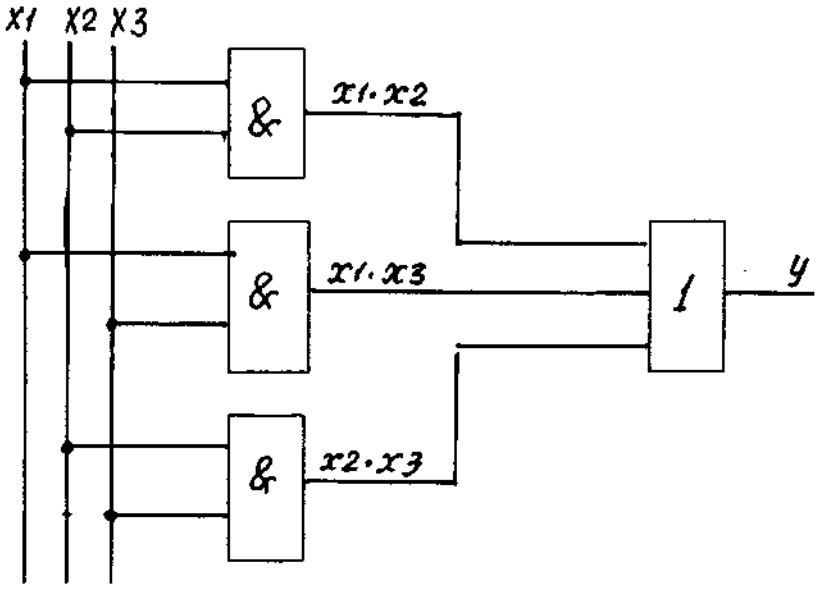


Рис.2.4 - Функціональна схема для заданої функції після мінімізації функції.

2.5 Мінімізація функцій алгебри логіки із застосуванням карти Карно

Як було показано вище, будь-яка функція алгебри логіки може бути записана в ДДНФ або ДКНФ. У ряді випадків така форма запису не є найпростішій для виразу заданої функції в аналітичній формі і можна спростити логічний вираз, не порушуючи значення функції.

В результаті мінімізації логічні функції можуть бути представлені в ДНФ або в КНФ з мінімальним числом членів і з мінімальним числом аргументів в кожному членові. Для спрощення виразів функцій алгебри логіки розроблені як графічні, так і алгебраїчні методи.

При невеликому числі змінних зручний графічний метод спрощення виразів для функцій алгебри логіки за допомогою карт Карно. Карта Карно є певною формою таблиці істинності для двох, трьох і чотирьох аргументів, як показано на рис.2.5, *а* - *в*.

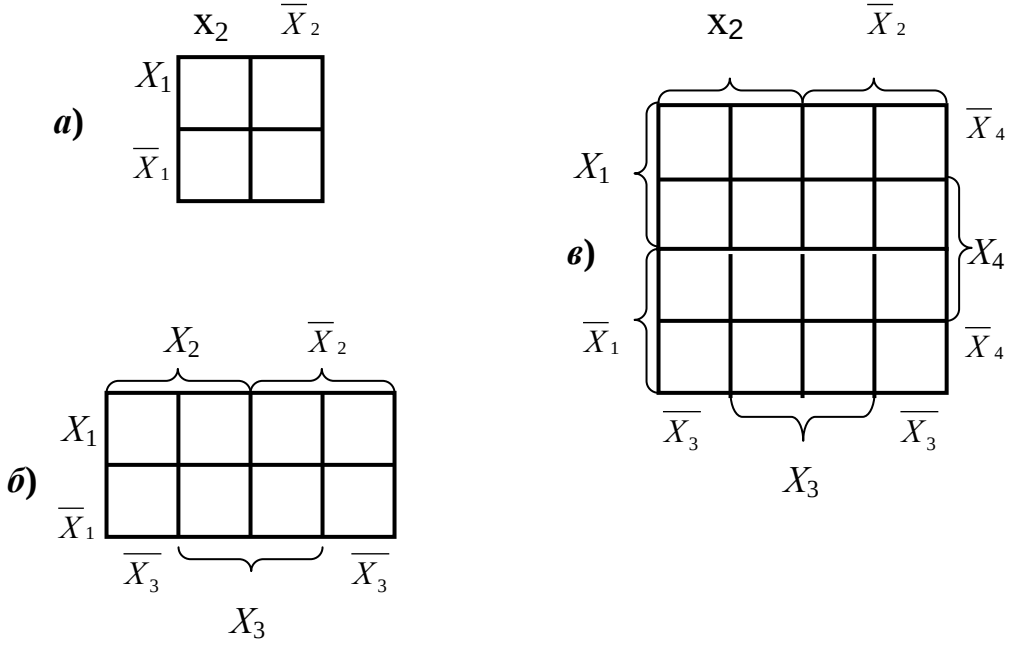


Рис.2.5 Таблиця карти Карно: *а*) для двох аргументів; *б*) для трьох аргументів; *в*) для чотирьох аргументів

Кожна клітка відповідає певному набору аргументів, причому цей набір визначається привласненням значення 1 аргументам, на перетині рядків і стовпців яких розташована клітка. Число кліток карти рівне числу можливих наборів значень аргументів і при числі аргументів *n* рівне 2n. У кожну клітку карти записується значення функції при відповідному цій клітці наборі значень аргументів. Наприклад, якщо функція задана таблицею істинності (див.

таблицю 2.2), то у формі карти Карно ця функція буде представлена так, як показано на мал. 2.6, а.

Таблиця 2.2 - Таблиця істинності функції $F(x_1x_2x_3)$

X_1	0	0	0	0	1	1	1	1
X_2	0	0	1	1	0	0	1	1
X_3	0	1	0	1	0	1	0	1
$F(x_1x_2x_3)$	0	1	0	1	0	0	1	1

При цьому клітки верхнього рядка відповідають наступним наборам (див. рис.2.6.а):

перша клітка: $x_1 = 1$; $x_2=1$; $\bar{x}_3 = 1$
друга клітка: $x_1 = 1$; $x_2=1$; $x_3=1$
третя клітка: $\bar{x}_1 = 1$; $x_2=1$; $x_3=1$
четверта клітка: $\bar{x}_1 = 1$, $\bar{x}_2 = 1$, $x_3=1$.

При записі функції в мінімальній формі по картах Карно використовуються наступні правила. Всі клітки, що містять 1, об'єднуються в замкнуті області. При цьому кожна область повинна бути прямокутником з числом кліток $2k$, де k=0, 1, 2, 3, 4 ... . Області можуть перетинатися, і одні і ті ж клітки можуть входити в різні області. Потім проводиться запис мінімального виразу в диз'юнктивній нормальній формі ДНФ. Кожна область в такому записі представляється членом, число аргументів в якому на k менше загального числа аргументів функції n (рівне n-k). Кожен член ДНФ складається лише з тих аргументів, які для відповідної області мають значення або без інверсій, або тільки з інверсією.

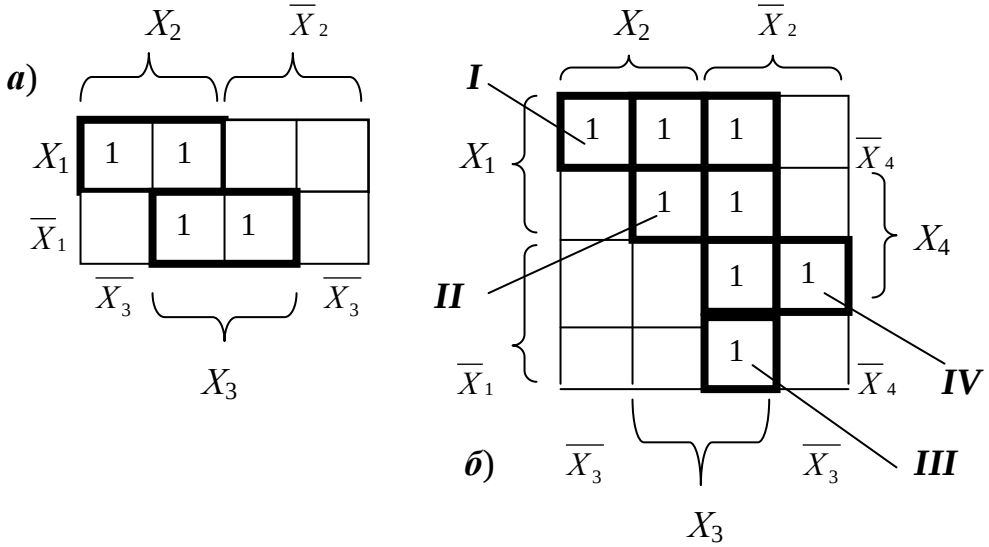


Рис.2.6 - Карта Карно: *а*) для функції $F(x_1,x_2,x_3)$ = для трьох аргументів;
б) для функції $F(x_1,x_2,x_3,x_4)$ - для чотирьох аргументів

Таким чином, при обхваті клітоk карти замкнутими областями слід прагнути, щоб число областей було мінімальним, оскільки при цьому буде мінімальним число членів в ДНФ, а кожна область містила можливо більше число клітоk, оскільки при цьому число аргументів в членах буде мінімальним. Так, для функції трьох аргументів, представленої на мал. 2.6.*а*, клітки, що містять 1, охоплюються двома областями. У кожній області дві клітки, і так як $2k = 2$, то, отже $k = 1$. Тому для цих областей n-k=3-1=2. В результаті в ДНФ буде два члени і в кожному з них по два аргументи. Першій області відповідає імпліканта x_1x_2, а другій області - імпліканта $\bar{x}_1x_3$. Отже, мінімальна ДНФ для цієї функції буде:

$$f(x_1, x_2, x_3) = x_1 x_2 \vee \bar{x}_1 x_3.$$

Прикладом завдання функції чотирьох аргументів за допомогою карти Карно може служити карта, приведена на рис.2.6.*б*, де виділено чотири області. Першу і четверту області мають по дві клітки; для них n-k=3 і відповідні ним члени будуть $x_1x_2\bar{x}_4$ і $\bar{x}_1\bar{x}_2x_4$. Області II і III містять по чотири клітки, і в ДНФ вони будуть виражені членами, що містять по два аргументи x_1x_3 і $\overline{x_2}x_3$. Таким чином, мінімальна форма функції буде

$$F(x_1,x_2,x_3,x_4) = x_1x_2\overline{x_4} \vee x_1x_3 \vee \bar{x}_2x_3 \vee \bar{x}_1\bar{x}_2x_4.$$

При побудові замкнутих областей допускається згортання карти в циліндр як по горизонтальній, так і по вертикальній осі з об'єднанням протилежних граней карти. Представлення функції і мінімізація її за допомогою карт Карно ускладнюється, якщо число аргументів більше чотирьох. Так, для представлення функції п'яти аргументів необхідно використовувати дві карти, кожна з яких є картою чотирьох змінних; одна для x_5=1, а інша для x_5=0. Ці карти розташовуються одна над іншою, і області обхвату клітоk можуть бути тривимірними, тобто в одну область можуть входити клітки двох карт. Для мінімізації функцій з числом аргументів більше 7-8 карти Карно практично не використовуються, і в цих випадках використовуються методи алгебри, такі як метод Квайна, метод невизначених коефіцієнтів і ін.

При побудові логічних схем як базисні функції окрім диз'юнкції, кон'юнкції і заперечення можуть бути використані і інші функції, створюючі повну систему. В цьому випадку мінімальні форми можуть бути отримані переходом від базису «диз'юнкція, кон'юнкція і інверсія» до нового базису шляхом відповідних перетворень. Наприклад, при переході до базису у вигляді функції Шеффера використовуються інверсія і формули Моргана.

2.6 Система команд однокристального мікропроцесора

Успіхи мікроелектроніки привели до широкого розповсюдження однокристальних мікропроцесорів (ОМП), всі компоненти яких реалізовані у вигляді однієї великої інтегральної схеми (ВІС). Розглянемо 8-розрядний ОМП КР580ВМ80.

Основні характеристики ОМП КР580ВМ80: довжина інформаційного слова 8 біт; число основних команд 111 (з модифікаціями 250); час виконання команди 2-9 мкс (при тривалості такту Т = 0,5 мкс); число регістрів загального призначення РЗП - 6; місткість пам'яті, що адресується, 64 Кбайт.

Електричні дані: три джерела напруги +5 В, +12 В, -5 В; сумарне споживання близько 750 мВт; для всіх сигналів стандартні ТТЛ-рівні, окрім двох послідовностей синхросигналів Ф1 і Ф2, які мають одиничний рівень – 12 В. Використовується пластмасовий 40-виводний корпус, кристал виготовляється по *n*-МОП-технології. Структурні особливості: передбачені можливість організації переривань, режим ПДП, асинхронний обмін інформацією.

Мікропроцесорний комплект КР580, до складу якого входить ОМП КР580ВМ80, містить близько 20 мікросхем для побудови МП-систем різного призначення.

Структурна схема КР580ВМ80 приведена на рис.2.7. Для неї характерні всі риси універсального ОМП: наявність арифметично-логічного пристрою АЛП з набором регістрів (Рг1, Рг2, Ак, РгF) для виконання заданої сукупності операцій; пристрої управління у складі регістра команд (РгК); дешифратора команд і шифратора машинного циклу; схем управління і синхронізації, за допомогою яких організовується взаємодія всіх елементів МП, а також взаємодія із зовнішнім середовищем в процесі його функціонування; системи з трьох шин для зв'язку із зовнішнім середовищем, зокрема двонаправленої 8-розрядної шини даних, однонаправленої 16-розрядної шини адреси і двонаправленої 10-розрядної шини управління. У ОМП передбачені засоби для організації переривань, прямого доступу до пам'яті, асинхронного обміну інформацією.

Проводиться обробка даних у вигляді 8-розрядних чисел, адресна інформація - 16-разрядна. У складі МП є 8-розрядний АЛП, що забезпечує апаратне виконання арифметичних (складання, віднімання) і логічних (множення, складання, інверсія, складання по модулю 2, порівняння кодів) операцій над 8-розрядними двоїчними кодами.

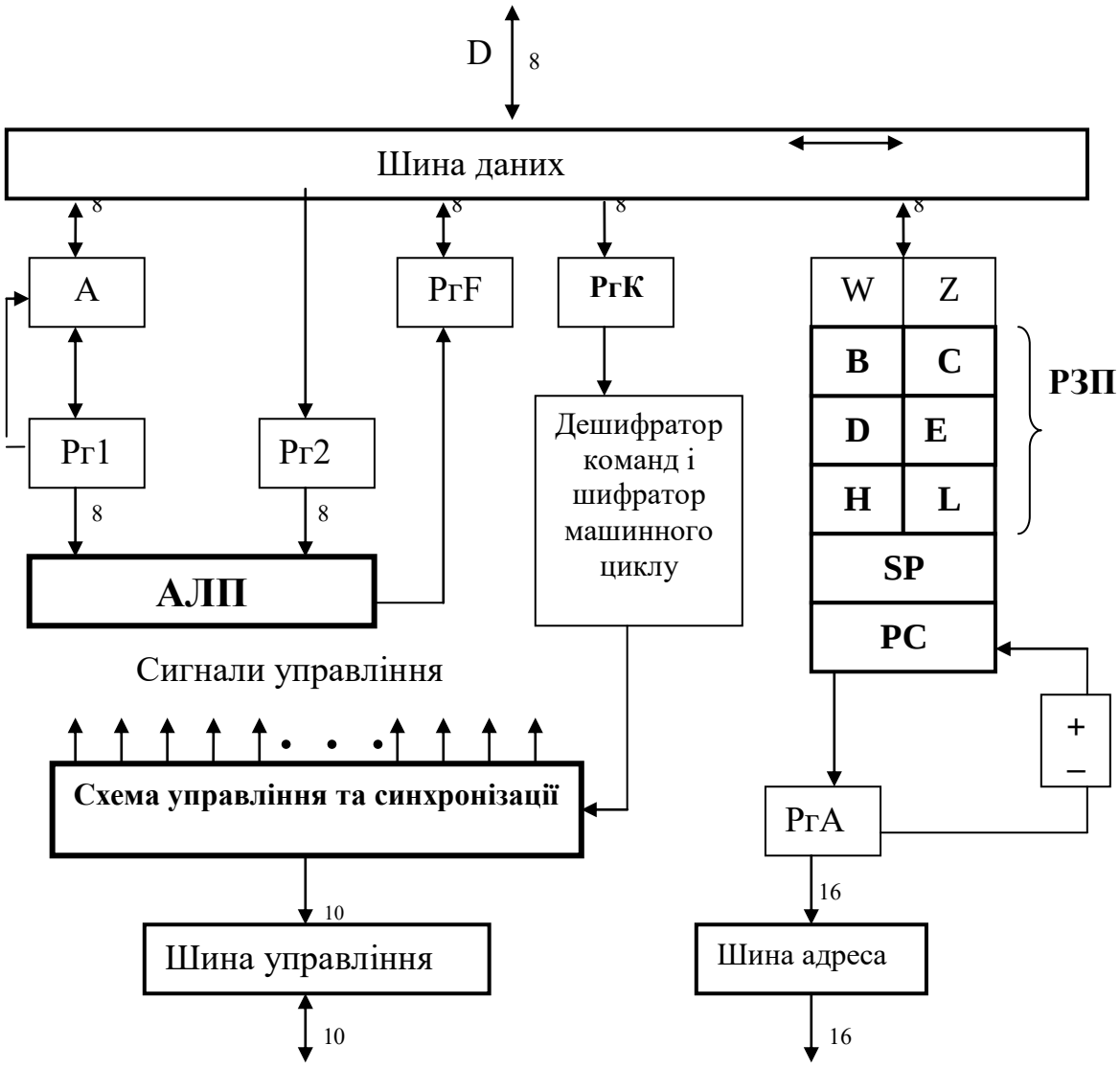


Рис.2.7 - Структурна схема мікропроцесора ОМП КР580ВМ80

Результат виконання операцій АЛП, як правило, поміщається в накопичуючий регістр - акумулятор (А). Вміст цього регістра зазвичай використовується як один з операндів в більшості операцій АЛП.

Обчислення результату операцій АЛП часто приводить до формування певних ознак, серед яких: ознака перенесення старшого розряду результату CY (якщо має місце перенесення, то CY=1); ознака нульового значення результату Z (якщо результат нульовий, то Z = 1); ознака негативного результату S (при негативному результаті S = 1); парного числа одиниць в байті результату P (при парному числі одиниць P=1); допоміжного перенесення між півбайтами результату AC (якщо має місце перенесення, то AC = 1). Остання ознака

використовується схемою десятичної корекції при обробці чисел з двоїчно-десятичним кодуванням інформації. Решта ознак застосовується для організації умовних переходів в програмах, що виконуються МП. Ознаки (прапори) поміщаються у відповідний регістр (РгF) і зберігаються там до моменту, поки не будуть сформовані нові значення ознак.

У складі МП використовується велике число регістрів. Частину з них виконують функції буферних елементів для узгодження тимчасових характеристик при передачі інформації всередині МП і обміні із зовнішнім середовищем (Рг1, Рг2, РгК, РгА, буферний регістр у складі шини даних). Інша частина включена до складу блоку регістрів. У цьому блоці є 8-розрядні регістри загального призначення (РЗП): B, C, D, E, H, L, що виконують функції надоперативної пам'яті МП. Програмне звернення до вказаних регістрів дозволяє скоротити число обмінів інформацією між МП і зовнішнім середовищем і тим самим підвищити його продуктивність. Є можливість звернення до пар РЗП (B і C, D і E, H і L) при обробці 16-розрядних чисел. Регістри W і Z використовуються як буферні при виконанні деяких операцій, програмне звернення до них не передбачене. Деякі з елементів блоку регістрів виконують спеціалізовані функції. Серед них 16-розрядний програмний лічильник (РС), який служить для формування адреси чергового байта команди, що прочитується з пам'яті. Вміст РС може модифікуватися за допомогою схеми ± для отримання адрес всіх байтів команд виконуваної програми. Іншим спеціалізованим елементом є 16-розрядний регістр, званий покажчиком стека SP. З його допомогою в МП-системі організовується стекова пам'ять.

Процес звернення до області ОЗП, відведеної під таку пам'ять, ілюструє рис.2.8. Кожен прямокутник тут позначає одну з 8-розрядних комірок області стекової пам'яті. Передбачається, що при записі інформації заповнення комірок пам’яті відбувається від низу до верху (заштриховані прямокутники). Стрілкою показана верхівка стека - верхній заповнений рівень стекової пам'яті.

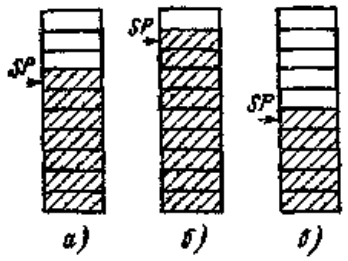


Рис.2.8 - Принцип роботи стека

Адресу саме цієї комірки в кожен момент часу задає SP. Запис в стек задається командою МП і передбачає заповнення двох комірок, розташованих вище за покажчик стека, вмістом два РЗП або РС з переміщенням вгору на дві позиції верхівки стека (мал. .8,*б*). Читання інформації із стекової пам'яті також задається певними командами МП і передбачає передачу в МП, в пару РЗП або

PC, вмісту двох елементів пам'яті. Адреса однієї з них задається SP, інший на одиницю більше, ніж попередній. Звільнення двох елементів стекової пам'яті супроводжується пониженням верхівки стека на дві позиції (рис.8.*в*). Подібний стек працює за принципом: число, записане в стек останнім, прочитується з нього першим (така організація іноді позначається LIFO - last in first out). Стекова пам'ять з організацією LIFO виявляється дуже зручною для реалізації апаратних і програмних переривань, оскільки дозволяє після обробки переривання передавати управління останній програмі, яка оброблялася до надходження запиту на переривання. Початкове завантаження PC і SP, передбачена певними командами МП, задає області пам'яті, що відводяться для зберігання програм і під стек.

Управління роботою МП забезпечується керуючим автоматом, складається з дешифратора команд і шифратора машинного циклу, а також схем управління і синхронізації. Відповідно до кодів команд, що поступають в МП, він забезпечує генерацію послідовностей сигналів, необхідних для управління всіма елементами МП (підключення сигналів управління до відповідних елементів на рис.2.7 не показане).

Для зв'язку із зовнішнім середовищем МП використовує систему з трьох шин. Двонаправлена 8-розрядна шина даних D дозволяє організувати прийом в МП команд і операндів і передачу результатів обробки. Вона через буферний регістр (на рис.2.7 не показаний у вигляді окремого елементу) сполучена з внутрішньою шиною даних, використовуваною для обміну 8-розрядними кодами між окремими елементами МП. Однонаправлена 16-розрядна шина адреси *A* забезпечує передачу з МП кодів для адресації пам'яті і пристроїв вводу-виводу. Обидві шини мають виходи з трьома станами. Організація взаємодії МП із зовнішнім середовищем забезпечується десятьма лініями, які утворюють шину управління. Для цього використовуються також слова стану, що видаються з МП на певних тимчасових інтервалах.

МП є програмно-керуємий пристрій, який здатний виконувати певний набір дій (операцій), що задається відповідним набором команд. При розгляді системи команд МП велику допомогу може надати так звана програмна модель, яка містить всі доступні програмістові компоненти МП без вказівки внутрішніх зв'язків між ними, що складається з 10 регістрів: шість регістрів загального призначення, акумулятор, лічильник команд, покажчик стека, регістр ознак.

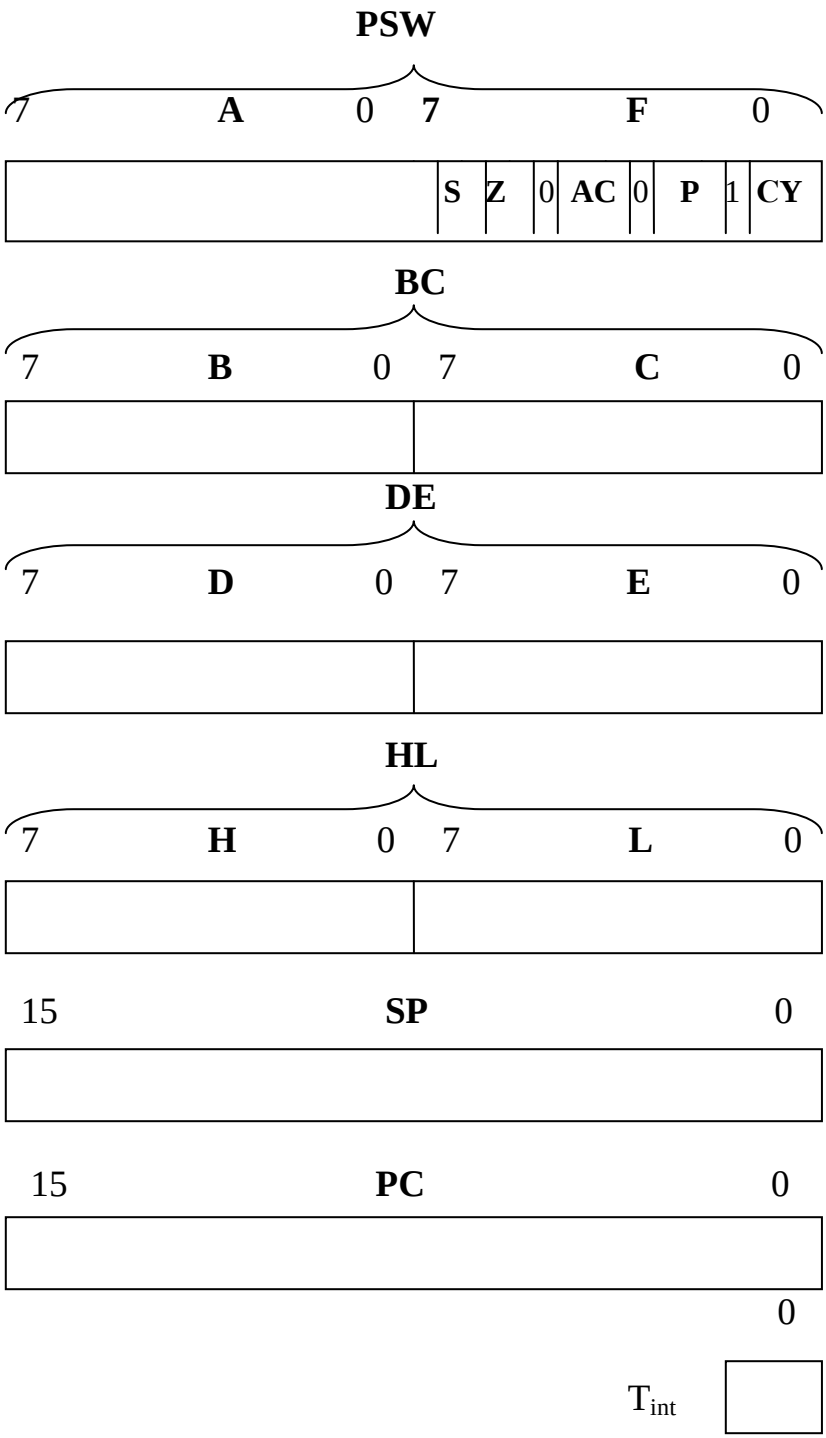


Рис.2.9 - Програмна модель 8-розрядного ОМП КР580ВМ80

Функції цих компонентів забезпечуються відповідними командами МП. На рис.2.9 зображена програмна модель 8-розрядного ОМП КР580ВМ80. На ній

представлений 8-розрядний накопичуючий регістр (акумулятор) A, вміст якого як джерело або (і) приймач інформації бере участь в більшості команд. Поряд з ним знаходиться регістр ознак F, у відповідних його розрядах показані всі п'ять використовуваних ознак. Розміщення регістрів A і F в програмній моделі поряд викликано тим, що їх вміст (PSW - загальне позначення регістрів A і F) записується в стек однією командою. Регістри загального призначення B, C, D, E, H, L можуть бути використані окремо або парами, що визначає їх розташування в програмній моделі. У стек записується однією командою вміст пари РЗП.

Покажчик стека SP і лічильник команд PC завжди оперують 16-розрядними кодами. Трігер T_{int} призначений для зберігання заборони переривань.

Обробка інформації і функціонування МП забезпечуються за допомогою програмного управління. Програма записується в ОЗП у вигляді послідовності команд. Кожна команда визначає вид операції, що виконується в даному циклі роботи, адреси слів, що беруть участь в операції, місце розташування результату операції, адресу розташування наступної команди. Із-за малої розрядності МП дуже важко задати таку обширну інформацію тільки одним словом. Проблема вибору формату команд і кодування полів команд МП мають особливе значення. Гнучкість МП і його ефективність визначаються числом команд і повнотою системи команд, засобами і способами адресації, можливостями організації розгалужених обчислювальних процесів.

Із збільшенням розрядності команди ростуть і можливості МП. Обмежена розрядність команди створює істотні труднощі в розміщенні інформації про хід операції і метод адресації даних. Для подолання цих труднощів в систему команд вводяться операції з подвоєною розрядністю, а також команди із змінною розрядністю.

Окрім поля коду операції і кодів адрес даних команда повинна містити поле ознак з вказівками способів адресації. Способи адресації визначають механізм формування прямої адреси пам'яті по полю адреси і полю ознак адресації. Гнучкість системи команд значною мірою визначається різноманітністю способів адресації. Вибір системи команд є складним завданням при побудові МП. Команди можна класифікувати по функціональному призначенню, по числу адрес, за способом кодування команд, по довжині команди, за способом адресації.

По функціональному призначенню розрізняються команди передачі даних, обробки даних, передачі управління і додаткові команди. Команди передачі даних включають підгрупи команд передачі кодів між регістрами МП, пересилки кодів між МП і ОЗП, передачі кодів між МП і зовнішніми пристроями. Команди обробки даних підрозділяються на арифметичні, логічні і команди зміщення. Команди передачі управління використовуються для організації порядку виконання команд і організації циклічних ділянок в

програмах. Серед них виділяються команди безумовного і умовного переходів Додаткові команди використовуються для завдання останову програми, початкової установки апаратних засобів, реалізації очікування.

По числу адрес розрізняють нуль-адресні, одноадресні, двоадресні і багатоадресні команди.

За способом кодування розрізняються команди з фіксованим полем коду операцій, а також з полем, що розширюється.

По довжині розрізняють команди завдовжки в один, два, три байти.

Механізм адресації значною мірою впливає на ефективність обробки інформації в МП. Для подолання обмежень із-за малої розрядності кодів команд використовуються всілякі способи адресації, які дозволяють визначити повну адресу пам'яті меншим числом біт, обчислювати адреси під час обробки, обчислювати адреси даних щодо позиції команди таким чином, що можна завантажувати програму в будь-яку область пам'яті без змін адреси в програмі. Способи адресації можна розділити на дві групи. До першій групи належать способи, в яких виконавча адреса визначається одним значенням коду в команді. Такими є пряма регістрова, непряма регістрова, безпосередня, автоінкрементна і автодекрементна адресації. До другої групи належать такі способи адресації, в яких використовується вміст адресної частини команди і декількох регістрів для формування виконавчої адреси. Такими є сторінкова, індексна, відносна адресації.

При *прямій адресації* код адреси в команді є виконавчою адресою звернення до пам'яті. При *регістровій адресації* оброблюване слово (операнд) міститься в одному з регістрів МП. При *регістровій непрямій адресації* непряма адреса витягується з внутрішнього регістра МП. *Безпосередня адресація* дозволяє задавати операнд в команді.

Автоінкрементна адресація заснована на обчисленні виконавчої адреси так само, як і при регістровій непрямій адресації, потім здійснюється збільшення вмісту регістра на деяку константу.

При *адресації автодекрементній* спочатку з вмісту регістра віднімається константа, потім отриманий результат використовується як виконавча адреса. Сумісне використання автоінкрементної і автодекрементної адресації забезпечує застосування будь-якого регістра як стек.

При *сторінковій адресації* пам'ять розбивається на ряд сторінок однакової довжини. Адресація сторінок здійснюється за допомогою регістра сторінок, а адресація елементів пам'яті всередині сторінки - адресою в команді. Номери всіх сторінок можуть знаходитися в таблиці сторінок, яка є нульовою сторінкою

Індексна адресація використовується при зверненні до масивів слів і таблиць. Для утворення виконавської адреси до адресної частини команди додається зсув (індекс) з регістра, званого індексним. Вміст індексного регістра можна змінювати; це дозволяє змінювати виконавчу адресу без модифікації адресної частини команди.

При *відносній адресації* виконавча адреса утворюється складанням базової адреси з адресою команди. Як базова адреса використовується вміст програмного лічильника. Така адресація дозволяє будувати вільно переміщувані в пам'яті програми.

Кожна команда МП має певну структуру (формат), в якій можна виділити частину (поле) коду операції (КОП) і поле операнда, що визначає числа (операнди), що беруть участь в операції відповідно до КОП. Спосіб визначення операнда на основі структури команди називається режимом адресації. Використання декількох режимів адресації розширює можливості при складанні програми. Найширше застосовуються наступні способи адресації:

- неявна адресація, коли місце розташування операнда мається на увазі і його адреса окремо ні в якій частині команди не задається;
- пряма адресація, що передбачає запис в полі операнда адреси елементу пам'яті з операндом;
- безпосередня адресація, коли в полі операнда знаходиться сам операнд;
- регістрова адресація, коли в полі операнда указується номер РЗП з операндом;
- непряма адресація, що передбачає запис в окремих розрядах КОП номерів РЗП, в яких знаходиться адреса елементу пам'яті з операндом.

Саме ці способи адресації застосовуються в МП КР580ВМ80, причому в одній і тій же команді може одночасно використовуватися декілька способів. Наприклад, в командах для обробки двох чисел один операнд може бути заданий регістровою адресацією, а інший - безпосередньою. Частіше всього один з операндів мається на увазі таким, що знаходиться в акумуляторі, туди ж поміщається і результат виконання операції.

При обробці інформації в МП кожна команда є двоїчним кодом. Проте при підготовці програм користувачеві зазвичай зручніше застосовувати символічні позначення (мнемокоди) команд (див. табл.2.3). Найчастіше як мнемокоди використовуються скорочення від англійських найменувань відповідних операцій. Наприклад, LDA - load direct accumulator (пряме завантаження акумулятора). Іноді мнемокоди є слова, що визначають суть виконуваних операцій. Наприклад, PUSH - заштовхнути, POP - виштовхнути. У структурі команд в символічному вигляді можуть приводитися відомості про операнди і адреси, по яких розташовані операнди (це можуть бути регістри МП, регістрові пари, елементи пам'яті М, 8- або 16-розрядні числа, 8- або 16-розрядні адреси).

Таблиця 2.3 - Система команд МП КР580ВМ80

Мнемоніка команди	Опис команди	**Код команди** DDDDDDDD 7 6 5 4 3 2 1 0	Довжина команди (байт)	**Число** тактів	Ознаки умов S Z AC P CY
MOV R1,R2	Передача з R2 в R1	01DDDSSS	1	5	- - - - -
MOV M,R	Передача з R в пам'ять	01110SSS	1	7	- - - - -
MOV R,M	Передача з пам'яті в R	01DDD110	1	7	- - - - -
MVI R,d8	Передача байта в R	00DDD110	2	7	- - - - -
MVI M,d8	Передача байта в пам'ять	00110110	2	10	- - - - -
LXI RP,d16	Завантаження парних регістрів BC,DE,HL,SP	00RP0001	3	10	- - - - -
LDAX RP	Завантаження акумулятора по адресу [BC] або [DE]	00RP1010	1	7	- - - - -
STAX RP	Занесення вмісту акумулятора по адресу [BC] або [DE]	00RP0010	1	7	- - - - -
LDA adr	Завантаження акумулятора по адресу, вказаному в команді	00111010	3	13	- - - - -
STA adr	Занесення вмісту акумулятора по адресу, вказаному в команді	00110010	3	13	- - - - -
LHLD adr	Завантаження регістрів L, H з двох сусідніх комірок, починаючи з адреси, вказаної в команді	00101010	3	16	- - - - -
SHLD adr	Занесення вмісту регістрів L, H в дві сусідні комірки, починаючи з адреси, вказаної в команді	00100010	3	16	- - - - -
XCHG	Обмін даними між парами регістрів HL і DE	11101011	1	4	- - - - -
XTHL	Обмін даними між SP і HL	11100011	1	18	- - - - -
SPHL	Занесення вмісту регістра HL в SP	11111001	1	5	- - - - -
PUSH RP	Введення вмісту регістрів BC,DE, HL в стек	11RP0101	1	11	- - - - -
PUSH PSW	Введення PSW в стек	11110101	1	11	+ + + + +
POP RP	Виведення даних з стека в регістри BC,HL,DE	11RP0001	1	10	+ + + + +
POP PSW	Виведення даних з стека в акумулятор и PSW	11110001	1	10	+ + + + +
ADD R	Додавання вмісту R і акумулятора	10000SSS	1	4	+ + + + +

Продовження таблиці 2.3

ADC R	Те ж, але з урахуванням переносу CY	10001SSS	1	4	+ + + + +
ADD M	Додавання вмісту комірки пам'яті і акумулятора	10000110	1	7	+ + + + +
ADC M	Те ж, але с урахуванням переносу CY	10001110	1	7	+ + + + +
ADI d8	Складання байта з вмістом акумулятора	11000110	2	7	+ + + + +
ACI d8	Те ж, але с урахуванням переносу CY	11001110	2	7	+ + + + +
DAD RP	Складання вмісту регістрів BC,DE,HL,SP з вмістом HL	11RP1010	1	10	- - - - -
SUB R	Віднімання вмісту R з акумулятора	10010SSS	1	4	+ + + + +
SBB R	Те ж, але з заємом	10011SSS	1	4	+ + + + +
SUB M	Віднімання вмісту комірки пам'яті з акумулятора	10010110	1	7	+ + + + +
SBB M	Те ж, але з заємом	10011110	1	7	+ + + + +
SUI	Віднімання байта з вмісту акумулятора	11010110	2	7	+ + + + +
SBI d8	Те ж, але з заємом	11011110	2	7	+ + + + +
INR R	Збільшення вмісту R на 1	00DDD100	1	5	+ + + + -
INR M	Збільшення вмісту комірки пам'яті на 1	00110100	1	10	+ + + + -
DCR R	Зменшення вмісту R на 1	00DDD101	1	5	+ + + + -
DCR M	Зменшення вмісту комірки пам'яті на 1	00110101	1	10	+ + + + -
INX RP	Збільшення вмісту BC,HL,DE,SP на 1	00RP0011	1	5	- - - - -
DCX RP	Зменшення вмісту BC,HL,DE,SP на 1	00RP1011	1	5	- - - - -
ANA R	Порозрядне логічне множення вмісту R і акумулятора	10100SSS	1	4	+ + 0 + 0
ANA M	Порозрядне логічне множення вмісту комірки пам'яті і акумулятора	10100110	1	7	+ + 0 + 0
ANI d8	Порозрядне логічне множення вмісту акумулятора і байта	11100110	2	7	+ + 0 + 0
XRA R	Порозрядне виключаюче АБО щодо вмісту R і акумулятора	10101SSS	1	4	+ + 0 + 0

Продовження таблиці 2.3

XRA M	Порозрядне виключаюче АБО щодо вмісту комірки пам'яті і акумулятора	10101110	1	7	+ + 0 + 0
XRI d8	Порозрядне виключаюче АБО щодо вмісту акумулятора і байта	11101110	2	7	+ + 0 + 0
ORA R	Порозрядне логічне додавання вмісту R і акумулятора	10110SSS	1	4	+ + 0 + 0
ORA M	Порозрядне логічне додавання вмісту комірки пам'яті і акумулятора	10110110	1	7	+ + 0 + 0
ORI d8	Порозрядне логічне додавання вмісту акумулятора і байта	11110110	2	7	+ + 0 + 0
CMP R	Порівняння вмісту R і акумулятора	10111SSS	1	4	+ + + + +
CMP M	Порівняння вмісту комірки пам'яті і акумулятора	10111110	1	7	
CPI d8	Порівняння байта з вмістом акумулятора	11111110	2	7	
RLC	Циклічний зсув вмісту акумулятора вліво	00000111	1	4	- - - - +
RRC	Те ж, але вправо	00001111	1	4	- - - - +
RAL	Циклічний зсув вмісту акумулятора вліво через перенос	00010111	1	4	- - - - +
RAR	Те ж, але вправо	00011111	1	4	- - - - +
CMA	Інвертування акумулятора	00101111	1	4	- - - - -
STC	Установка прапора переносу CY в 1	00110111	1	4	- - - - +
CMC	Інвертування прапора переносу	00111111	1	4	+ + + + +
DAA	Двоїчно-десятична корекція вмісту акумулятора	00100111	1	4	- - - - -
JMP	Безумовний перехід	11000011	3	10	- - - - -
JC	Перехід при переносі	11011010	3	10	- - - - -
JNC	Те ж, при відсутності переноса	11010010	3	10	- - - - -
JZ	Те ж, при нулі	11001010	3	10	- - - - -
JNZ	Те ж, за відсутності нуля	11000010	3	10	- - - - -
JP	Те ж, при плюсі	11110010	3	10	- - - - -
JM	Те ж, при мінусі	11111010	3	10	- - - - -

Продовження таблиці 2.3

JPE	Те ж, при парності	11101010	3	10	- - - - -
JPO	Те ж, при непарності	11101001	3	10	- - - - -
PCHL	Занесення в лічильник команд вмісту регістра HL	11101001	1	5	- - - - -
CALL	Виклик підпрограми	11001101	3	17	- - - - -
CC	Те ж, при переносі	11011100	3	11/17	- - - - -
CNC	Те ж, при відсутності переносу	11010100	3	11/17	- - - - -
CZ	Те ж, при нулі	11001100	3	11/17	- - - - -
CNZ	Те ж, при відсутності нуля	11000100	3	11/17	- - - - -
CP	Те ж, при плюсі	11110100		11/17	- - - - -
CM	Те ж, при мінусі	11111100	3	11/17	- - - - -
CPE	Те ж, при парності	11101100	3	11/17	- - - - -
CPO	Те ж, при непарності	11100100	3	11/17	- - - - -
RET	Повернення з підпрограми	11001001	1	10	- - - - -
RC	Те ж, при переносі	11011000	1	5/11	- - - - -
RNC	Те ж, при відсутності переносу	11010000	1	5/11	- - - - -
RZ	Те ж, при нулі	11001000	1	5/11	- - - - -
RNZ	Те ж, при відсутності нуля	11000000	1	5/11	- - - - -
RP	Те ж, при плюсі	11110000	1	5/11	- - - - -
RM	Те ж, при мінусі	11111000	1	5/11	- - - - -
RPE	Те ж, при парності	11101000	1	5/11	- - - - -
RPO	Те ж, при непарності	11100000	1	5/11	- - - - -
RST	Повторний запуск	11NNN111	1	11	- - - - -
IN port	Виведення з порту	11011011	2	10	- - - - -
OUT port	Введення в порт	11010011	2	10	- - - - -
EI	Дозволить переривання	11111011	1	4	- - - - -
DI	Заборонить переривання	11110011	1	4	- - - - -
NOP	Відсутність операції	00000000	1	4	- - - - -
HLT	Останов	01110110	1	7	- - - - -

Надалі необхідно користуватися наступними умовними позначеннями:

- DDD,SSS - 3-розрядні поля у форматі команди, що адресують один з регістрів загального призначення або як місце призначення (D), або як джерело операнда (S). Окрім імені кожен РЗП, а також регістр А мають тризначний двоїчний код:

000 B 010 D 100 H 110 M
001 C 011 E 101 L 111 A;

- RP - 2-розрядне поле у форматі команди, що адресують один з парних регістрів (регістрова пара), а також покажчик стека:
  00 - **BC**; 01 - **DE**; 10 - **HL**; 11- **SP** або **PSW**;
- PSW - слово-стан програми, 1-й байт якого рівний A, 2 - вмісту регістра F;
- NNN - двоїчне представлення номера команди RST;
- + - установка або скидання прапора умови;
- – - відсутність впливу на прапор;
- 5/11 - в знаменнику дробу вказано число тактів при виконанні умови, що розглядається в команді, в чисельнику - при невиконанні;
- d8 - байт безпосереднього операнда;
- d16 - двобайтовий безпосередній операнд;
- adr - безпосередня адреса операнда (адреса елементу пам'яті);
- port - однобайтовий номер порту;
- R, R1, R2 ...- один з 8-розрядних РЗП або акумулятор.

Регістр ознак F призначений для зберігання 5 ознак (прапорів), що утворюються при виконанні деяких операцій. Ці ознаки (біти умов) зберігаються у відповідних розрядах регістра ознак (рис.2.10).

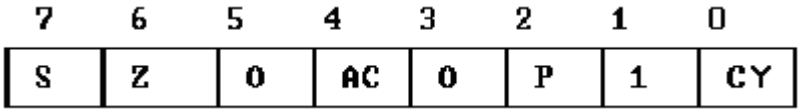


Рис.2.10 - Регістр ознак F

де S (Sign) - ознака знаку результату, що зберігається в акумуляторі. При позитивному знаку S=0, при негативному S=1;

Z (Zero) - якщо вміст акумулятора рівний 0, то ознака Z=1, інакше Z=0.

CY(Carry) - ознака перенесення. C=1,якщо при виконанні команд з'являється одиниця перенесення із старшого розряду (переповнення).

AC (Auxiliary Carry) - додаткова ознака переносу. Встановлюється у одиницю, якщо при виконанні команд виникає одиниця з четвертого розряду числа (з молодшої тетради в старшу)

P (Parity) - ознака парності. P=1, якщо кількість одиниць в розрядах акумулятора буде парним. Нульовий результат також відноситься до парного. Інакше P=0.

Розряди 1,3 і 5 в регістрі ознак не використовуються як ознаки. Повний опис кожної команди МП повинен містити інформацію про те, на які ознаки в регістрі F ця команда впливає.

Двоїчне число, вибране з пам'яті і яке вказує на виконання певної операції, називається командою. Двоїчне число, яке підлягає обробці, називається операндом.

Використані нижче позначення дозволяють разом із записом мнемокодів команд формувати при необхідності їх двоїчні уявлення. Програмно доступні регістри МП позначаються символами R. У структурі команд їм відповідають 3-розрядні коди SSS і DDD, яким привласнені двоїчні коди, що визначають конкретні регістри МП відповідно до табл.2.3. Звернемо увагу, що коду 110 відповідає елемент пам'яті М, адреса якої задається парою регістрів HL (при використанні непрямої адресації).

Будь-яка команда має мнемонічний (символічний) опис, що полегшує написання програми. Наприклад, ADD (скласти), MOV (переслати), XCHG (обмін вмістом регістрових пар D і H) і ін.

У першому стовпці таблиці 2.3 приведені мнемокоди команд і відповідні відомості про операнди. Вони можуть бути задані у вигляді: позначення 8-розрядного регістра R (джерела або приймача) інформації; 8-розрядного елементу пам'яті М, адреса якої визначається регістровою парою або покажчиком стека; 16-розрядного вмісту регістрової пари або покажчика стека; 8-розрядного операнда d8, узятого з другого байта двобайтової команди; 16-розрядного операнда d16, узятого з другого (молодша частина) і третього (старша частина) байтів трьохбайтової команди; 8-розрядної адреси adr пристрою (port) вводу-виводу, взятого з другого байта двобайтових команд IN і OUT; 16-розрядної адреси adr, взятого з другого (молодша частина) і третього (старша частина) байтів трьохбайтової команди.

У другому стовпці табл.2.3 дано скорочене представлення операцій, що виконуються відповідними командами.

Далі в табл.2.3 приводиться варіант формату команди. Зіставлення використаного варіанту формату команди із змістом виконуваної операції дозволяє порівняно просто представити послідовність дій МП при виконанні команд. У наступних стовпцях табл.2.3 вказуються число байтів в команді і число машинних тактів Т, потрібних для виконання команди. Ця інформація використовується для оцінки об'єму програми і часу її виконання

У стовпці « Ознаки умов» відображені відомості про формування в результаті виконання команд всіх п'яти ознак. Символ “-” відповідає збереженню ознаки незмінною; символ “+” - формуванню ознаки на основі аналізу результату (див. опис функціональної схеми МП КР580ВМ80); символи 0 або 1 визначають формування як ознаку відповідних констант. У командах виконання логічного множення (ANA R, ANA M, ANI d8) як ознака допоміжного перенесення АС береться вміст АС результату.

У системі команд зазвичай виділяється п'ять груп команд: пересилка кодів; виконання арифметичних операцій; виконання логічних операцій; передача управління; команди вводу-виводу і спеціальні.

Команди пересилки кодів передбачають передачу 8-розрядного коду з регістра в регістр, з регістра в елемент пам'яті і назад, завантаження вмісту другого байта команди в регістр або елемент пам'яті. У ряді команд

забезпечується передача 16-розрядного коду з двох елементів пам'яті в регістрову пару і назад, зокрема з використанням області пам'яті, відведеної під стек. Передбачено завантаження регістрової пари вмістом другого і третього байтів команди і обмін даними між регістровими парами (HL) і (DE), а також регістровою парою HL і елементами стекової пам'яті. У командах пересилки кодів жоден з ознак не формується.

Розглянемо декілька прикладів представлення команд пересилки кодів. Хай потрібно виконати пересилку коду РЗП L в РЗП B.

Відповідно до першого рядка табл.2.3 цю операцію виконує однобайтова команда. Мнемокод може бути записаний у вигляді MOV B, L, а двоїчний код - 01000101. Аналогічно можна визначити, що операції пересилки 16-розрядного коду, що знаходиться в другому і третьому байтах команди, в регістрову пару BC відповідають мнемокод LXI BC, d16 і двоїчний код першого байта 00000001.

Команди виконання арифметичних операцій забезпечують додавання і віднімання 8-розрядних чисел, одне з яких знаходиться в акумуляторі, з розміщенням результату в акумуляторі. Друге число, що бере участь в цих операціях, може задаватися різними режимами адресації (регістровою, непрямою, безпосередньою). При виконанні деяких команд додавання і віднімання передбачена можливість обліку перенесення CY, що дозволяє організувати обробку багатобайтних чисел окремими частинами. Всі команди додавання і віднімання 8-розрядних чисел формують повний набір ознак результату. Є можливість виконання додавання 16-розрядних чисел з використанням регістрових пар. В цьому випадку результат фіксується в регістровій парі HL і формується тільки признак переносу CY. Ряд команд дозволяє змінити на одиницю у бік збільшення або зменшення вміст регістра, регістрової пари, елементи пам'яті M. В результаті виконання цих команд ознак перенесення не формується, а решта ознак формується тільки в операціях з 8-розрядними числами. У цій групі команд особливе місце займає команда десятичної корекції, що передбачає перетворення вмісту акумулятора. У зв'язку з тим, що це перетворення засноване на виконанні арифметичних операцій над вмістом півбайтів акумулятора, то команда віднесена до даної групи.

Команди виконання логічних операцій передбачають реалізацію найбільш поширених логічних операцій над двома 8-розрядними кодами, один з яких розташований в акумуляторі, інший задається різними режимами адресації з приміщенням результату в акумулятор. Серед цих операцій: логічне множення, логічне складання, складання по модулю 2. Команди порівняння кодів (CMP R, CMP M, CPI d8) виконуються шляхом віднімання з вмісту акумулятора другого операнда з формуванням всіх ознак результату, але без зміни самого вмісту акумулятора. Є команда інвертування вмісту акумулятора без формування ознак результату. У цю ж групу команд віднесені команди інвертування ознаки перенесення і запис як ця ознака одиничного значення, а також команда

порозрядного зрушення вмісту акумулятора вліво і вправо на один розряд (з двома варіантами формування ознаки перенесення CY).

Група команд передачі управління забезпечує можливість зміни порядку виконання команд в програмі. Серед них команда JMP adr передає управління за адресою, що задається другим і третім байтами команди, а PCHL - за адресою, узятою з регістрової пари HL (це команди безумовної передачі управління). Команда виклику підпрограми CALL adr також передає управління за адресою, заданою другим і третім байтами але з одночасним записом в стекову пам'ять поточного значення PC, що дає можливість повернутися до перерваної програми. Це повернення може бути проведене за допомогою команди RET, поновлюючої вміст PC прочитуванням його із стекової пам'яті. Команда RST прочитується із зовнішнього пристрою, який сформував запит на переривання. Вона передає управління програмі обробки переривання, початкова адреса якої задається 3-розрядним кодом nnn. Є команди, які виконують передачу управління, виклик підпрограми або повернення з неї тільки у разі виконання умови (cond), що задається відповідною ознакою. Мнемокоди команд умовного переходу починаються буквою J, умовного виклику підпрограми - буквою C, а умовного повернення з підпрограми - буквою R. Далі в мнемокоді йде мнемонічне позначення умови відповідно до табл.2.3. Проілюструємо це прикладом. Хай потрібно організувати умовний перехід за адресою, що задається другим і третім байтами команди, якщо значення ознаки S = 0. На основі табл.2.3 мнемокод - JP adr. Двоїчний код першого байта команди 11110010.

У останній групі команд є дві команди, що забезпечують введення-виведення інформації через акумулятор. Другий байт цих команд дозволяє адресувати до 256 пристроїв введення і стількох же пристроїв виводу. У цій групі є також декілька спеціальних команд. Команди EI і DI, керують станом трігера Tint, забезпечують програмний дозвіл або заборону режиму переривання Команда HLT дозволяє зупинити виконання програми, а команда NOP не задає виконання операції, вона дозволяє перейти до чергової команди із затримкою на чотири такти T. Данная група команд також не впливає на ознаки. Оскільки більшість представлених в табл.2.3 команд має узагальнену форму, то вони забезпечують значне число модифікацій з використанням комбінацій регістрів, регістрових пар, умов на основі ознак.

У всіх командах розряди КОП розташовуються в першому байті.
Можливі варіанти однобайтових команд приведені на рис.2.11.

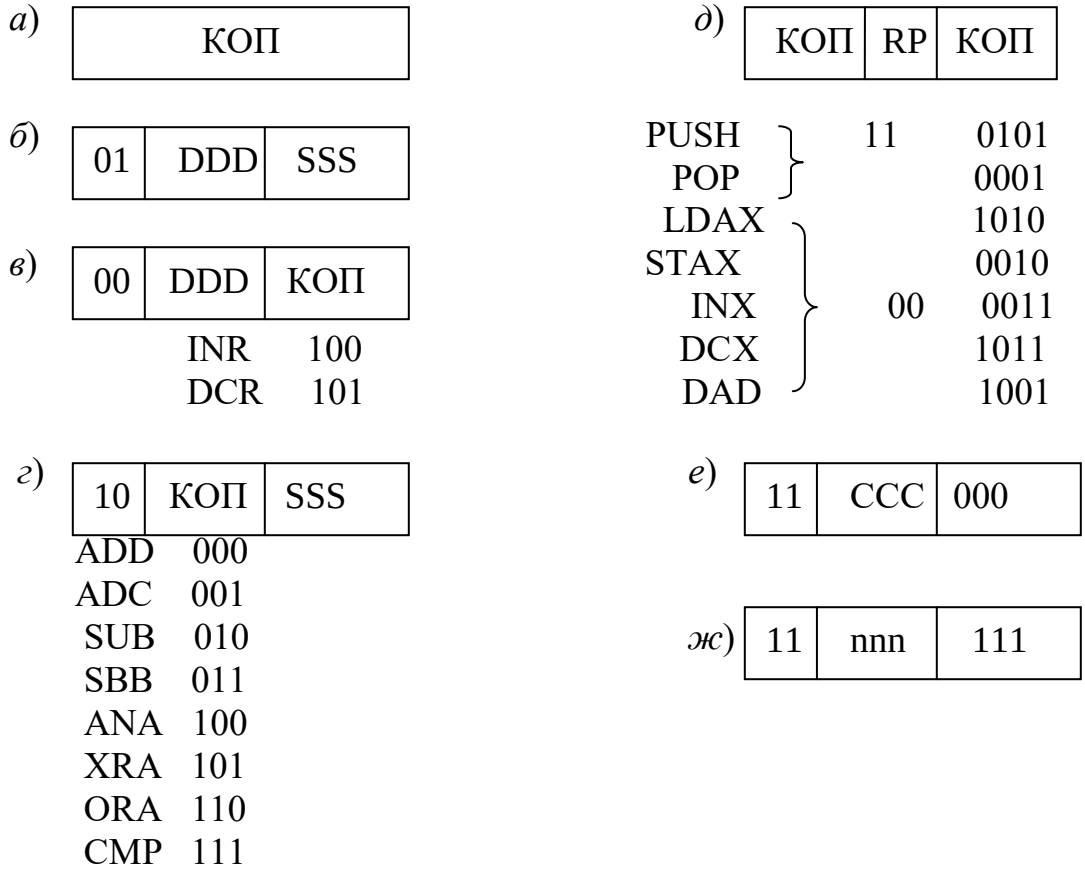


Рис.2.11 - Варіанти однобайтових команд

Варіант.*а* рис.2.11 відповідає випадку, коли весь байт команди займають розряди КОП. Це зазвичай має місце або при неявній адресації операнда (найчастіше мається на увазі, що він знаходиться в акумуляторі), або коли операнд не використовується в команді взагалі. У табл. 2.5 приведені мнемокоди і двоїчні значення КОП всіх команд варіанту *а.*

Таблиця 2.5 - Мнемокоди команд, коли весь байт команди займають розряди КОП

Мнемокоди команд варіанта *а*	Значення КОП 76543210	Мнемокоди команд варіанта *а*	Значення КОП 76543210
SPHL	11111001	RLC	00000111
XCHG	11101011	RRC	00001111
XTHL	11100011	RAL	00010111
DAA	00100111	RAR	00011111
CMA	00101111	PCHL	11101001
CMC	00111111	EI	11111011
STC	00110111	DI	11110011
RET	11001001	HLT	01110110
		NOP	00000000

Варіант *б* рис.2.11 передбачає використання регістрової адресації. Два ліві розряди утворюють КОП із значенням 01, що відповідає командам пересилки інформації з одного РЗП в інший. Самі РЗП, що беруть участь в такій пересилці, визначаються на основі табл.2.6. Виняток становить код джерела або приймача інформації 110, який задає пересилку за участю елементу пам'яті М з адресою, узятою з регістрової пари HL (непряма адресація).

Таблиця 2.6 - Трирозрядні коди SSS або DDD, яким привласнені двоїчні коди, що визначають конкретні регістри МП

SSS або DDD	Позначення регістра
000	B
001	C
010	D
011	E
100	H
101	L
110	M
111	A

Варіанти *в* і *г* рис.2.11 передбачають регістрову адресацію з використанням одного РЗП (джерела або приймача інформації). На рис.2.11 для цих варіантів представлені мнемокоди всіх відповідних ним команд і двоїчні значення КОП.
Варіант *д* рис.2.11 визначає формат однобайтових команд, в яких беруть участь регістрові пари і покажчик стека. Приведені тут значення розрядів КОП разом з даними з табл.2.7 дозволяють отримати двоїчні коди відповідних команд.

Таблиця 2.7 - Дворозрядні коди регістрових пар BC, DE, HL, покажчика стека, регістрів A і F (PSW)

Позначення пари регістрів	RP	Мнемокоди команд
B	00	LDAX, STAX DAD, LXI, INX, DCX, PUSH, POP
D	01	LDAX, STAX DAD, LXI, INX, DCX, PUSH, POP
H	10	DAD, LXI, INX, DCX, PUSH, POP
SP	11	DAD, LXI, INX, DCX
PSW	11	PUSH, POP

Аналогічно варіант *е* рис.2.11 разом з даними табл.2.8 може бути використаний для знаходження двоїчних кодів однобайтових команд для організації переходів в програмах.

Таблиця 2.8 - Трирозрядний код, що позначається у ряді форматів команд символами CCC, що визначає одна з ознак(Z, CY, P, S), використовуваних для організації переходу в програмі

Умова перехода	CCC	Мнемонічне позначення умови
Нерівність нулю (Z=0)	000	NZ
Рівність нулю (Z=1)	001	Z
Відсутність перенесення (CY=0)	010	NC
Наявність перенесення (CY=1)	011	C
Непарність (P=0)	100	PO
Парність (P=1)	101	PE
Позитивний результат (S=0)	110	P
Негативний результат (S=1)	111	M

Трирозрядний код, що позначається символами nnn у варіанті *ж* рис.2.11, використовується для визначення адреси в команді RST.

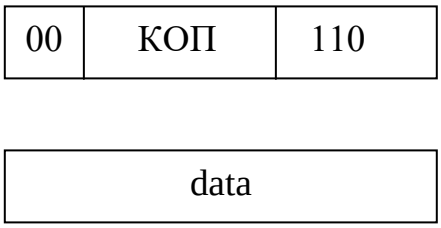
Варіанти двобайтових команд приведені на рис.2.12

Варіант *і* рис.2.12 представляє комбінацію неявної і безпосередньої адресації. Один з операндів неявно передбачається таким, що знаходиться в акумуляторі, інший в другому байті команди (data).

Варіант *к* рис.2.12 представляє комбінацію регістрової (приймач інформації) і безпосередньої (джерело інформації) адресацій.

У варіанті *л* рис.2.12, використовуваному в командах вводу-виводу, при n=0 задається ввод (IN), при n=1 - вивод (OUT). Другий байт визначає адреса пристрою вводу-виводу (adr port).

і)



ADI 000
ACI 001
SUI 010
SBI 011
ANI 100
XRI 101
ORI 110
CPI 111

к)

л)

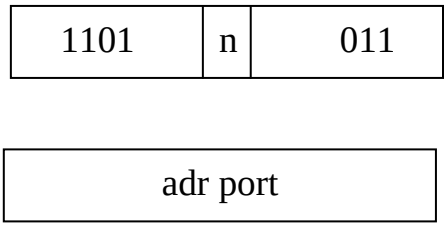
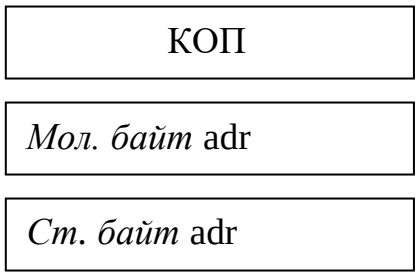


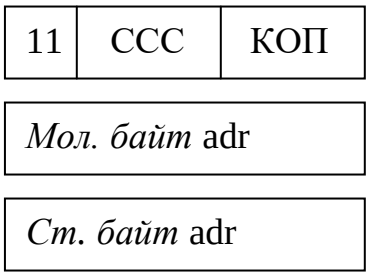
Рис.2.12 - Варіанти двобайтових команд

Варіанти трибайтових команд представлені на рис.2.13.

м)



п)



н)

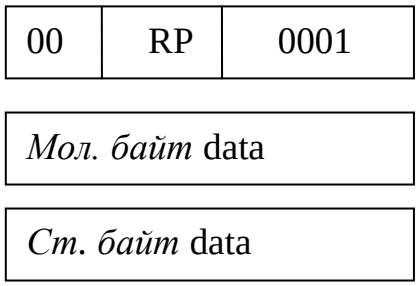


Рис.2.13 - Варіанти трибайтових команд

Пряма адресація відповідає варіанту *м*. Другий байт визначає молодші розряди, а третій - старші розряди. У таблиці 2.9 приведені мнемокоди і двоїчне значення КОП для всіх команд варіанту *м*.

Таблиця 2.9 - Мнемокоди і двоїчне значення КОП для всіх команд варіанту *м*

Мнемокод команд варіант *м*	Значення КОП 76543210
LDA	00111010
STA	00110010
LHLD	00101010
SHLD	00100010
JMP	11000011
CALL	11001101

Безпосередня адресація із завантаженням вмістом другого і третього байтів команди регістрової пари або покажчика стека використовується у варіанті *н*. Вибір приймача інформації відповідно до коду RP проводиться на основі таблиці 2.7 (для мнемокоду LXI).

Варіант *п* використовує пряму адресацію для організації переходів в програмах на основі умов, що задається в таблиці 2.8.

3 МАТЕРІАЛИ, ПРИЛАДИ, ОБЛАДНАННЯ

Лабораторний практикум проводиться із застосуванням персонального комп'ютера.

4 ВКАЗІВКИ ПО ТЕХНІЦІ БЕЗПЕКИ

При виконанні лабораторного практикуму виконувати вимоги по охороні праці згідно інструкції №328 «Вимоги по охороні праці при роботі з комп'ютерною технікою».

5 ПОРЯДОК ПРОВЕДЕННЯ ЛАБОРАТОРНОГО ПРАКТИКУМУ

5.1 Лабораторна робота №1

Перетворити задану послідовність (див. таблицю 5.1):

десятичного, восьмиричного та шістнадцятиричного кодів в двоїчні коди;

двоїчних кодів в десятичний, восьмиричний та шістнадцятиричний коди.

Виконати арифметичні операції над заданими числами в двоїчній, восьмиричній і шістнадцятиричній формі:

Таблиця 5.1 – Варіанти завдання

№ варіанта	Числа в десятичній формі	Числа в двоїчной формі		Числа в восьмиричній формі		Числа в шістнадцятиричній формі	
	D1	D2	B1	B2	Q1	Q2	**H1**	**H2**
1	126;	51	00101101;	01100100	1272;	1757	1AB2;	A23C
2	72;	48	01011001;	10111001	7650;	6531	FA31;	35EF
3	39;	91	10001111;	01010101	4657;	1747	7DF1;	65FF
4	88;	66	00111101;	11001010	1534;	5431	A7CD;	CD4F
5	67;	44	11100011;	00011011	1573;	5551	AC7D;	AD7E
6	29;	54	00110011;	10110011	2345;	7726	6AB1;	A5C7
7	41;	55	01010101;	01010111	1722;	6626	6CFE;	B9EF
8	62;	19	10010011;	01001111	1676;	4343	30AC;	6E3C
9	59;	54	00011111;	01111001	2111;	3111	E71F;	FC2F
10	70;	33	11001100;	00101101	2754;	3347	312C;	4516
11	50;	30	01111011;	01010001	2456;	7105	3892;	2CC7
12	32;	122	10111101;	00101011	1707;	6322	F3C8;	7C5D
13	70;	111	00011111;	10100111	1530;	4776	3AB2;	2BC3

Продовження таблиці 5.1

14	49; 47	00101011; 10011101	3331; 2777	45AB; 36EF
15	98; 51	10001111; 01101010	3732; 2127	3A6E; E7C1
16	81; 55	01110110; 01010010	1705; 5327	9210; 4AA4
17	101; 33	11001101; 00101011	2117; 3731	817C; DF41
18	78; 94	00111110; 01001111	4172; 4037	87AE; CA52
19	79; 70	10001111; 01011111	1712; 2737	7B5A; 54E3
20	122; 64	01110111; 00011101	2771; 5173	BC6E; A58C
21	25; 176	10001101; 00111111	2277; 4327	31AD; F765
22	81; 91	00011101; 01010101	5127; 3273	7FB4; 563A
23	111; 77	10011110; 00110111	1742; 4027	96EC; D6D7
24	142; 43	00110100; 10011011	1427; 3172	8FC4; 59A7
25	128; 44	10101011; 00100111	3237; 4374	1BEF; 4A7E

Контрольні питання для самостійної перевірки:

1. Поясніть суть позиційної системи числення.
2. Представте число у вигляді геометричної прогресії, визначте розряди числа і ваги кожного розряду.
3. Представте визначення основи системи числення.
4. Поясніть правила перекладу чисел з двоїчної системи числення в 8-ричну, в 16-ричну.
5. Поясніть правила перекладу чисел з десятичної системи числення в 2-їчну і виконання зворотного переводу.
6. Представте таблицю двоїчного додавання, множення.
7. Яка мета застосування зворотного і додаткових кодів при виконанні арифметичних операцій.

5.3 Лабораторна робота №2

По заданій таблиці істинності (таблиця 5.2) розробити логічний вираз функції в ДНФ:

- провести мінімізацію логічного виразу, використовуючи теореми алгебри логіки;
- по отриманому виразу розробити схему з'єднання ЛЕ;
- перевірити правильність перетворень шляхом аналізу комбінацій аргументів в отриманні заданої функції із застосуванням комп'ютера для розробленої схеми з'єднань ЛЕ:
  - запустити програму kmap445.exe;

- провести вибір варіанту карти Карно для заданої кількості аргументів шляхом ініціалізації відповідного значка панелі, що відображає сітку карти Карно;
- задати варіанти функцій на панелі програми таким чином:

у таблиці вікна з переліком наборів провести вибір заданих наборів аргументів згідно завданню для всіх необхідних варіантів функцій, відзначивши маніпулятором у відповідних вікнах програми або встановити у відповідні клітки карти Карно маніпулятором значки відповідно до значень варіантів функцій;

- звірити результат обчислень, виконаний згідно програмі kmap445.exe з отриманим раніше;
- розробити логічний вираз функції в КНФ.

Таблиця 5.2 - Варіанти завдання функцій

X_1	X_2	X_3	$У_1$	$У_2$	$У_3$	$У_4$	$У_5$	$У_6$	$У_7$	$У_8$	$У_9$	$У_{10}$
0	**0**	**0**	0	1	0	0	1	1	0	1	1	0
0	**0**	**1**	1	1	0	0	1	1	0	0	0	1
0	**1**	**0**	0	1	0	1	0	1	0	0	1	0
0	**1**	**1**	1	0	0	1	0	1	1	0	0	1
1	**0**	**0**	0	1	0	1	0	0	1	1	1	1
1	**0**	**1**	1	0	1	1	0	0	1	1	1	0
1	**1**	**0**	0	1	1	0	1	0	1	1	1	0
1	**1**	**1**	1	0	1	0	1	1	0	1	0	1

Контрольні питання для самостійної перевірки:

1. Поясніть суть основних функцій алгебри логіки для одного і двох аргументів.
2. Перечисліть системи булевих функцій, що володіють функціональною повнотою.
3. Які способи завдання булевих функцій?
4. У чому полягає завдання мінімізації булевих функцій?
5. Поясніть суть ДНФ і ДДНФ, КНФ і ДКНФ.
6. Вкажіть способи виразу довільної функції для елементів КЦУ.
7. Представте карту Карно для 2, 3, 4 аргументів.
8. Поясніть принципи представлення функцій і спосіб мінімізації із застосуванням карти Карно.

5.4 Лабораторна робота №3

Запустити програму Sim8080. У вікні, що з'явилося, відбивається модель мікропроцесорної системи в складі програмної моделі мікропроцесора, оперативній пам'яті М, стека, регістрів портів вводу-виводу. У лівій верхній частині вікна розміщена зона для програмування (введення команд і операндів);

у правій частині регістр А, регістри РЗП, регістр умов, портів, регістр лічильника команд та покажчика стека; у нижній лівій частині відображені комірки оперативної пам'яті М мікропроцесора, нижній правій частині – комірки пам’яті стека. Дані в регістрах A, B, C, D, E, H, L, покажчику стека, регістрах портів відображаються в двоїчній системі числення - індекс b, шістнадцатиричній - індекс h, десятиричній, - індекс d. Дані в оперативній пам'яті і стеку відбиваються в шістнадцятиричній системі.

Виконати операції, що задаються набором у відповідності з додатком А:

- передача 8-розрядних даних в регістри A, B, C, D, E, H, L;
- передача 16-розрядних даних в регістрові пари BC, DE, HL;
- обмін даними між регістровими парами HL і DE;
- передача даних з регістра B в регістр A;
- вивчення методів адресації;
- послідовне введення вмісту регістрів BC, DE, HL в стек;
- видача даних із стека в регістри BC, DE, HL;
- складання вмісту регістрів B або C або D або E або H або L з вмістом регістра A;
- складання вмісту елементу пам'яті і вмісту акумулятора з видачею результату в акумуляторі;
- складання числа (байта) з вмістом акумулятора з видачею результату в акумуляторі;
- порозрядне логічне множення вмісту регістра (B) і акумулятора, елементу оперативної пам'яті і акумулятора, вмісту акумулятора і байта;
- порозрядне логічне складання вмісту акумулятора і байта;
- реалізація логічної функції «складання по модулю 2»

Перевіряти правильність обчислень в процесі виконання команд по даним, відображеним в вікні програми в відповідності до додатку А. Виконати обчислення із застосуванням команд мікропроцесора.

Контрольні питання для самостійної перевірки:

1.Перечисліть програмно доступні елементи мікропроцесора, їх призначення.
2. Поясніть принцип побудови команди, її характеристики.
3. Поясніть принципи функціонального розподілу команд по групах.
4. Як працює стекова пам'ять?
5. Яка інформація передається через шини МП?
6. Які способи адресації використовуються в командах?
7. Що означає розрядність мікропроцесора?
7. Перечисліть основні елементи структурної схеми МП.

6 ЗМІСТ ЗВІТУ

1 Перетворення послідовностей цифр і арифметичні операції в двоїчній, восьмиричній і шістнадцятиричній системі числення.
2 Матеріали по мінімізації заданої логічної функції згідно варіанту завдання по таблиці 5.2. Синтезована схема з'єднань ЛЕ. Логічний вираз функції в ДНФ і КНФ
3 Результати обчислень, програму обчислень із застосуванням команд мікропроцесора.

7 КОНТРОЛЬНІ ПИТАННЯ ДЛЯ САМОСТІЙНОЇ ПЕРЕВІРКИ І КОНТРОЛЮ ПІДГОТОВКИ СТУДЕНТІВ До РОБОТИ

1. Поясніть суть позиційної системи числення.
2. Представте число у вигляді геометричної прогресії, визначте розряди числа і ваги кожного розряду.
3. Представте визначення основи системи числення.
4. Поясніть правила перекладу чисел з двоїчної системи числення в 8-ричну, в 16-ричну.
5. Поясніть правила перекладу чисел з десятичної системи числення в 2-їчну і виконання зворотного переводу.
6. Представте таблицю двоїчного додавання, множення.
7. Яка мета застосування зворотного і додаткових кодів при виконанні арифметичних операцій.
8. Поясніть суть основних функцій алгебри логіки для одного і двох аргументів.
9. Перечисліть системи булевих функцій, що володіють функціональною повнотою.
10. Які способи завдання булевих функцій?
11. У чому полягає завдання мінімізації булевих функцій?
12. Поясніть суть ДНФ і ДДНФ, КНФ і ДКНФ.
13. Вкажіть способи виразу довільної функції для елементів КЦУ.
14. Представте карту Карно для 2, 3, 4 аргументів.
15. Поясніть принципи представлення функцій і спосіб мінімізації із застосуванням карти Карно.
16. Перечисліть програмно доступні елементи мікропроцесора, їх призначення.
17. Поясніть принцип побудови команди, її характеристики.
18. Поясніть принципи функціонального розподілу команд по групах.
19. Як працює стекова пам'ять?
20. Яка інформація передається через шини МП?
21. Які способи адресації використовуються в командах?
22. Що означає розрядність мікропроцесора?
23. Перечисліть основні елементи структурної схеми МП.

ЛІТЕРАТУРА:

1. Цифровая и вычислительная техника. Учебник. Под ред. Э. В. Евреинова. - М., Радио и связь, 1991. – 464 с.
2. Вычислительные системы, сети и телекоммуникации. Под ред. А. П. Пятибратова. Учебник, - М., Финансы и статистика, 2005. – 560 с.
3. Автоматизированные системы управления. Под ред. В. Н. Четверикова. Лабораторный практикум по техническим средствам. Уч. пособие. - М., Высшая школа, 1986. – 279 с.
4. Микропроцессоры. Под ред. Преснухина. В 3 книгах, - М., Высшая школа, 1986.
5. Гилмор Ч. Введение в микропроцессорную технику. - М., Мир, 1984. – 334 с.
6. Б. М. Каган, В. В. Сташин. Микропроцессоры в цифровых системах. - М., Энергия, 1979.

ДОДАТОК А

ТАБЛИЦІ КОМАНД МІКРОПРОЦЕСОРА

Таблиця А.1 - Передача 8-розрядних даних в регістри A, B, C, D, E, H, L

Мнемоніка	Операнд 1	Операнд 2	Коментар
MVI	A	16	Передача байта (число 16) в регістр A
MVI	B	40	Передача байта (число 40) в регістр B
MVI	C	45	Передача байта (число 45) в регістр C
MVI	D	70	Передача байта (число 70) в регістр D
MVI	E	160	Передача байта (число 160) в регістр E
MVI	H	200	Передача байта (число 200) в регістр H
MVI	L	255	Передача байта (число 255) в регістр L

Таблиця А.2 - Передача 16-розрядних даних в регістри BC, DE, HL

Мнемоніка	Операнд 1	Операнд 2	Коментар
LXI	BC	500	Передача 16-розрядних даних (число 500)в регістр BC
LXI	DE	1600	Передача 16-розрядних даних (число 1600)в регістр DE
LXI	HL	65535	Передача 16-розрядних даних (число 65535)в регістр HL

Таблиця А.3 – Обмін даними між парами регістрів HL і DE

Мнемоніка	Операнд 1	Операнд 2	Коментар
XCHG			Обмін даними між парами регістрів HL і DE. До виконання команди в регістрах: в DE – число 1600, в HL – число 65535, після виконання команди в регістрах: в DE - число 65535, в HL – число 1600. Початкові цифри були введені в попередній операції (таблиця А.2)

Таблиця А.4 – Передача даних із регістра B в регістр A

Мнемоніка	Операнд 1	Операнд 2	Коментар
MOV	A	B	Передача даних одного іх регістрів : B або C або D або E або H або L в регістр A. Позначення відповідного регістра указується в графі „Операнд 2”

Таблиця А.5 – Методи адресації (*безпосередня адресація*)

Мнемоніка	Операнд 1	Операнд 2	Коментар
STA	1		Передача даних з регістра А в пам’ять з адресом комірки -1
SHLD	7		Передача даних з регістрів HL в пам’ять з адресом комірки -7

Таблиця А.6 – Методи адресації *(регістрова непряма адресація)*. Передача даних з регістра А в пам’ять М по адресу, вказаному в регістрах HL або DE або BC

Мнемоніка	Операнд 1	Операнд 2	Коментар
LXI	HL	6	Занесення в регістри HL номера комірки пам’яті (в даному випадку – №6)
MOV	M	A	Передача даних з регістра А в пам’ять М (в даному випадку комірка №6), що вказана в регістрах HL
LXI	DE	3	Занесення в регістри DE номера комірки пам’яті (в даному випадку – №3)
STAX	DE		Передача даних з регістра А в пам’ять М комірка №3, що вказана в регістрах DE
LXI	BC	4	Занесення в регістри BC номера комірки пам’яті (в даному випадку – №4)
STAX	BC		Передача даних з регістра А в пам’ять М (в даному випадку комірка №4), що вказана в регістрах BC

Таблиця А.7 - Методи адресації(*безпосередня адресація*). Передача даних з пам’яті М в регістр А та HL

Мнемоніка	Операнд 1	Операнд 2	Коментар
LDA	8		Завантаження регістра А даними з пам’яті М (в даному випадку комірка №8)
LHLD	1		Завантаження регістрів HL даними з пам’яті М (в даному випадку комірка №1)

Таблиця А.8 - Методи адресації(*непряма адресація*). Передача даних з пам'яті М в регістр А (варіант 1)

Мнемоніка	Операнд 1	Операнд 2	Коментар
LXI	HL	7	Занесення в регістри HL номера комірки пам'яті (в даному випадку №7), з якої завантажуються дані
MOV	A	M	Завантаження регістра А даними з пам'яті М (в даному випадку комірка №7)

Таблиця А.9 - Методи адресації(*непряма адресація*). Передача даних з пам'яті М в регістр А (варіант 2)

Мнемоніка	Операнд 1	Операнд 2	Коментар
LXI	DE	8	Занесення в регістри DE номера комірки пам'яті (в даному випадку №8), з якої завантажуються дані
LDAX	DE		Завантаження регістра А даними з пам'яті М (в даному випадку комірка №8)

Таблиця А.10 - Методи адресації(*непряма адресація*). Передача даних з пам'яті М в регістр А (варіант 3)

Мнемоніка	Операнд 1	Операнд 2	Коментар
LXI	BC	7	Занесення в регістри BC номера комірки пам'яті (в даному випадку №7), з якої завантажуються дані
LDAX	BC		Завантаження регістра А даними з пам'яті М (в даному випадку комірка №7)

Таблиця А.11 –Послідовне введення даних регістрів BC, DE, HL в стек

Мнемоніка	Операнд 1	Операнд 2	Коментар
LXI	BC	160	Завантаження числа 160 в регістри BC
LXI	DE	1600	Завантаження числа 1600 в регістри DE
LXI	HL	65535	Завантаження числа 65535 в регістри HL
PUSH	BC		Занесення даних з регістрів BC в стек
PUSH	DE		Занесення даних з регістрів DE в стек
PUSH	HL		Занесення даних з регістрів HL в стек

Таблиця А.12 –Виведення даних із стека в регістри

Мнемоніка	Операнд 1	Операнд 2	Коментар
POP	BC		Видача даних в регістр BC
POP	DE		Видача даних в регістр DE
POP	HL		Видача даних в регістр HL

Таблиця А.13 – Додавання числа, що знаходиться в регістрі B або C або D або E або H або L до числа, що знаходиться в акумуляторі A.

Мнемоніка	Операнд 1	Операнд 2	Коментар
MVI	A	40	Завантаження регістра A (число 40)
MVI	H	10	Завантаження регістра H (число 10)
ADD	H		Виконання додавання даних регістрів з видачею результату в акумуляторі (в даному випадку число 50) Змінюючи числове значення операндів 1 і 2 аналізувати зміни в регістрі ознак (прапорів).

Таблиця А.14 – Додавання числа, що знаходиться в комірці памяті M до числа, що знаходиться в акумуляторі A.

Мнемоніка	Операнд 1	Операнд 2	Коментар
MVI	H	5	Занесення в регістри H номера комірки пам’яті (в даному випадку №5)
MVI	M	72	Завантаження даних в пам’ять (в даному випадку в комірку №5 числа 72)
MVI	A	88	Завантаження акумулятора (в даному випадку число 88)
ADD	M		Виконання додавання даних комірки №5 і акумулятора з видачею результату в акумуляторі (в даному випадку результат - 160)

Таблиця А.15 – Додавання байта до числа, що знаходиться в акумуляторі

Мнемоніка	Операнд 1	Операнд 2	Коментар
ADI	40		Виконання додавання байта (число 40) і акумулятора (160 – результат попередньї операції)з видачею результату в акумуляторі (в даному випадку результат - 200)

Таблиця А.16 – Порозрядне логічне множення (логічний елемент «І») вмісту регістра В і акумулятора

Мнемоніка	Операнд 1	Операнд 2	Коментар
MVI	A	1	Завантаження регістра А (число 1)
MVI	B	1	Завантаження регістра В (число 1)
ANA	B		Порозрядне логічне множення (А•В) з видачею результату в акумуляторі. Змінити один з операндів на „0” та перевірити правильність результату: 1х1=1; 1х0=0; 0х1=0;0х0=0.

Таблиця А.17 – Порозрядне логічне множення (логічний елемент «І») вмісту комірки памяті і акумулятора

Мнемоніка	Операнд 1	Операнд 2	Коментар
MVI	A	1	Завантаження регістра А (число 1)
MVI	H	3	Завантаження в регістр Н номера комірки памяті (в даному випадку комірка №3)
MOV	M	A	Передача даних з регістра А в пам’ять М (в даному випадку комірка №3),
MVI	A	1	Завантаження регістра А (число 1)
ANA	M		Порозрядне логічне множення (А•М) з видачею результату в акумуляторі. Змінити один з операндів на „0” та перевірити правильність результату: 1х1=1; 1х0=0; 0х1=0;0х0=0.

Таблиця А.18 – Порозрядне логічне множення (функція «І») вмісту байта і акумулятора

Мнемоніка	Операнд 1	Операнд 2	Коментар
MVI	A	1	Завантаження числа в акумулятор (число1)
ANI	1		Порозрядне логічне множення з видачею результату в акумуляторі. Змінити один з операндів на „0” та перевірити правильність результату: 1х1=1; 1х0=0; 0х1=0;0х0=0.

Таблиця А.19 – Порозрядне логічне додавання (функція «АБО») вмісту акумулятора і байта

Мнемоніка	Операнд 1	Операнд 2	Коментар
MVI	A	1	Завантаження числа в акумулятор (число1)
ORI	1		Порозрядне логічне додавання з видачею результату в акумуляторі. Змінити один з операндів на „0” та перевірити правильність результату: 1x1=1; 1x0=0; 0x1=0;0x0=0.

Таблиця А.20 - Порозрядне виключаюче „АБО” щодо вмісту байта і акумулятора

Мнемоніка	Операнд 1	Операнд 2	Коментар
MVI	A	0	Завантаження числа в акумулятор (число 0)
XRI	1		Порозрядне логічне виключаюче АБО з видачею результату в акумуляторі. Змінити один з операндів на 0 чи 1 та перевірити правильність результату: 1x1=0; 1x0=1; 0x1=1;0x0=0.

Таблиця А.21 – Віднімання чисел

Мнемоніка	Операнд 1	Операнд 2	Коментар
MVI	A	50	Завантаження числа в акумулятор (число50)
MVI	H	20	Завантаження числа в регістр H (число20)
SUB	H		Віднімання числа, що знаходиться в регістрі H від числа в акумуляторі з видачею результату в акумуляторі (в даному випадку число30).