Завдання 3. Архітектура й 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 на основі І8080.

Основні характеристики ОМП: довжина інформаційного слова 8 біт; число основних команд 111 (з модифікаціями 250); час виконання команди 2-9 мкс (кратні машинним циклам 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.

Кристал ОМП виготовлено за *n*-МОП-технологією. Структурні особливості: передбачені можливість організації переривань, режим ПДП, асинхронний обмін інформацією.

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

Особливості архітектури ОМП

ОМП є програмно керованим інструментом обробки інформації.

Структурна схема ОМП приведена на рис. 3. 1. 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-розрядна. У складі МП є 8-розрядний АЛП, що забезпечує апаратне виконання арифметичних (складання, віднімання) і логічних (множення, складання, інверсія, складання за модулем 2, порівняння кодів) операцій над 8-розрядними двійковими кодами.

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 десяткової корекції при обробці чисел з двійково-десятковим 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.

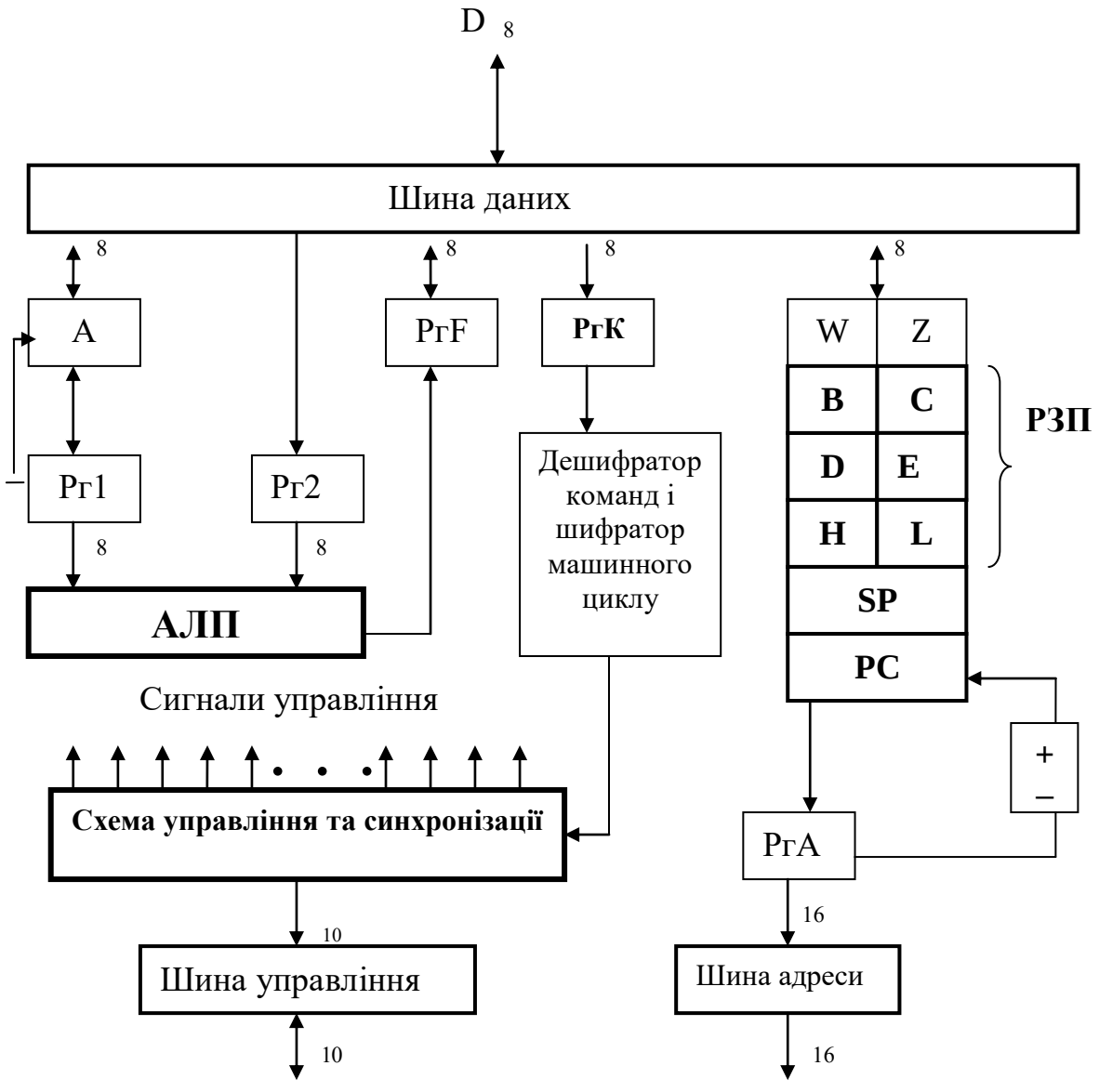

Рис. 3. 1 - Структурна схема ОМП

У складі ОМП використовується значна кількість регістрів. Частина з них виконують функції буферних елементів для узгодження часових характеристик при передачі інформації всередині МП і обміні з 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 суттєво 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), – для формування адреси чергового байта команди, що зчитується 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, покажчик стека SP. З його допомогою в МПС організується стекова пам'ять.

Процес звернення до області ОЗП, відведеної під таку пам'ять, ілюструє рис. 3. 2. Кожен прямокутник тут позначає одну з 8-розрядних комірок області стекової пам'яті. Передбачається, що при запису інформації, заповнення комірок пам’яті відбувається з низу до верху (заштриховані прямокутники). Стрілкою показана верхівка стеку – верхній заповнений рівень стекової пам'яті.

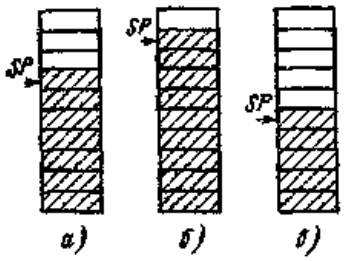

Рис. 3. 2 – Принцип роботи стеку

Адресу саме цієї комірки в кожен момент часу задає SP. Запис в стек задається відповідною 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 (див. рис. 3. 2,*б*).

Зчитування інформації з стекової пам'яті також задається певними командами МП і передбачає передачу в МП, в пару РЗП або РС, 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. 3. 2.*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, передбачене певними командами МП, задає області пам'яті, що відводяться для зберігання програм і під стек.

Управління роботою МП забезпечується керуючим автоматом, що 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. 3. 1 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. 3. 1 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 часових інтервалах.

Характеристика системи команд ОМП

МП є програмно-керований пристрій, здатний виконувати певний набір дій (операцій), що задається відповідним набором команд. Інформація представляється адресами й даними. В процесі обробки МП вони всі є даними. Система команд ОМП фіксована.

При розгляді системи команд МП допомогу надає так звана програмна модель, яка містить всі програмно доступні 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.

Функції цих компонентів забезпечуються відповідними командами ОМП. На рис. 3. 3 зображена програмна модель 8-розрядного ОМП . 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.

Реалізація програм

Обробка інформації і функціонування ОМП забезпечуються за допомогою програмного управління. Програма записується в ОЗП у вигляді послідовності команд.

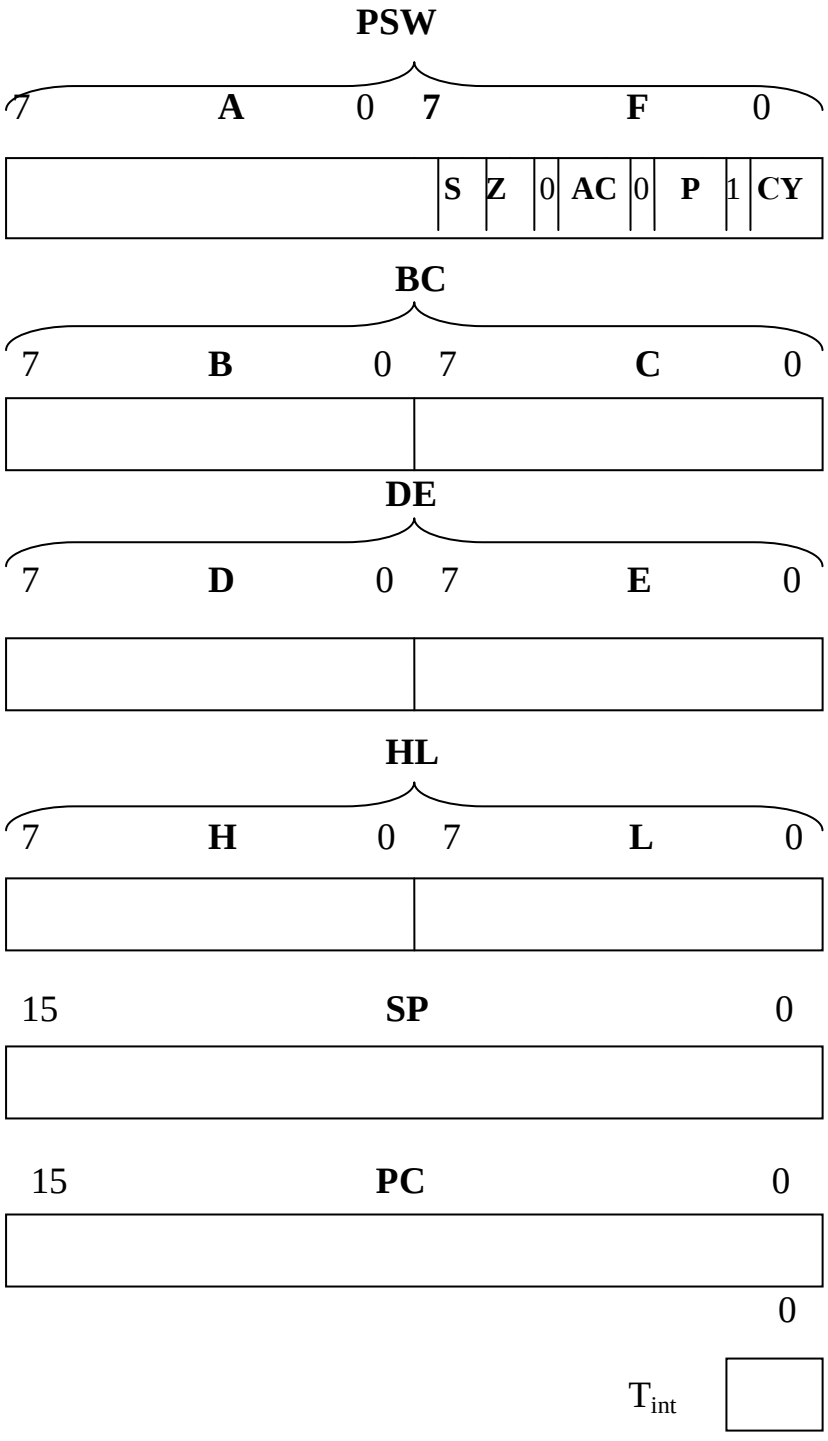

Рис. 3. 3 - Програмна модель 8-розрядного ОМП

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 тощо.

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

Окрім поля коду операції і кодів адрес даних команда має 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 й забезпечується різноманітністю способів адресації. Вибір системи команд є складним завданням при побудові ОМП.

Класифікація команд ОМП

Команди можна класифікувати за функціональним призначенням, числом адрес, за способом кодування команд, довжиною команди, за способом адресації та ін.

За функціональним призначенням 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 останову програми, початкової установки апаратних засобів, реалізації очікування.

За числом 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.

За довжиною є команди завдовжки в один, два, три байти.

Можливості адресації ОМП

Механізми адресації значною мірою впливають на ефективність обробки інформації в ОМП. 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 принципово поділяють на дві групи.

До першої групи належать способи, в яких 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 ОМП. При ***регістровій непрямій адресації*** непряма адреса вибирається з внутрішнього регістра ОМП. ***Безпосередня адресація*** дозволяє задавати операнд в команді.

Автоінкрементна адресація заснована на обчисленні виконуваної 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 виконувана адреса. Сумісне використання автоінкрементної і автодекрементної адресації забезпечує застосування будь-якого регістра в якості стеку.

При ***сторінковій адресації*** пам'ять розбивається на ряд сторінок однакової довжини. Адресація сторінок здійснюється за допомогою регістра сторінок, а адресація елементів пам'яті в межах 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 індексного регістру. Вміст індексного регістра можливо змінювати; це дозволяє змінювати виконувану адресу без модифікації адресної частини команди.

При ***відносній адресації*** виконувана адреса утворюється складанням базової адреси з адресою команди. В якості базової адреси використовується вміст програмного лічильника. Така адресація дозволяє створювати вільно переміщувані в пам'яті програми.

Структури й визначення команд ОМП

Кожна команда ОМП має певну структуру (формат), в якій можна виділити частину (поле) коду операції (КОП) і поле операнду, що визначає числа (операнди), які беруть участь в операції відповідно до КОП. Спосіб

визначення операнду на основі структури команди є режимом адресації. Використання декількох режимів адресації розширює можливості при складанні програми. Найширше застосовуються наступні способи адресації:

- неявна адресація, коли місце розташування операнду 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 ОМП, причому в одній і тій же команді може одночасно використовуватися декілька способів. Наприклад, в командах обробки двох чисел один операнд може бути заданий регістровою адресацією, а інший – безпосередньою. Частіше всього один з операндів є таким, що знаходиться в акумуляторі, туди ж поміщається і результат виконання операції.

При обробці інформації в МП кожна команда є двійковим кодом. Проте при підготовці програм користувачеві зазвичай зручніше застосовувати символічні позначення (мнемокоди) команд, табл. 3. 1. Найчастіше в якості мнемокодів використовуються скорочення від англійських найменувань відповідних операцій.

Наприклад, LDA – load direct accumulator (пряме завантаження акумулятора).

Іноді мнемокоди є словами, що визначають суть виконуваних операцій.

Наприклад, PUSH – заштовхнути, POP – виштовхнути.

У структурі команд в символічному вигляді можуть приводитися відомості про операнди і адреси, за якими розташовані операнди (це можуть бути регістри МП, регістрові пари, елементи пам'яті М, 8- або 16-розрядні числа, 8- або 16-розрядні адреси).

Таблиця 3. 1 - Система команд та кодів ОМП

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
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	00000111	1	4	- - - - +

	акумулятора вліво				
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	- - - - -
JPE	Те ж, при парності	11101010	3	10	- - - - -
JPO	Те ж, при непарності	11101001	3	10	- - - - -
PCHL	Занесення в лічильник ко-манд 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	Зупинка	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 двійковий 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 А, 2 - вмісту регістра F;
- NNN – двійкове представлення номера команди RST;
- + – встановлення 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 регістру ознак, рис. 3. 4.

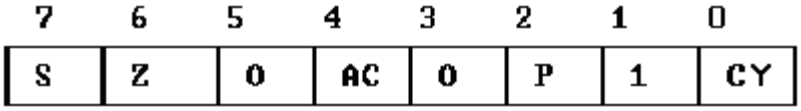

Рис. 3. 4 - Структура регістру ознак F

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 має містити інформацію про те, на які ознаки регістру F ця команда впливає.

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

Використані нижче позначення дозволяють разом із записом мнемокодів команд формувати при необхідності їх двійкові уявлення. Програмно доступні регістри МП позначаються символами R. У структурі команд їм відповідають 3-розрядні коди SSS і DDD, яким привласнені двійкові коди, що визначають конкретні регістри МП відповідно до табл. 3. 1. Слід зважати, що коду 110 відповідає елемент пам'яті M, адреса якої задається парою регістрів HL (при використанні непрямої адресації).

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

У першому стовпці таблиці 3. 1 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 M, 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. 3. 1 надано скорочене представлення операцій, що виконуються відповідними командами.

Далі в табл. 3. 1 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. 3. 1 вказуються число байтів в команді і число машинних тактів T, 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; символи 0 або 1 визначають формування як ознаку відповідних констант. У командах виконання логічного множення (ANA R, ANA M, ANI d8) в якості ознаки допоміжного перенесення AC береться вміст AC результату.

Можливості системи команд ОМП

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. 3. 1 цю операцію виконує однобайтова команда. Мнемокод може бути записаний у вигляді MOV B, L, а двійковий код – 01000101. Аналогічно можна визначити, що операції пересилки 16-розрядного коду, що знаходиться в другому і третьому байтах команди, в регістрову пару BC відповідає мнемокод LXI BC, d16 і двійковий 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 десяткової 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 за модулем 2. Команди порівняння кодів (CMP R, CMP M, CPI d8) виконуються шляхом віднімання з вмісту акумулятора другого операнда та формування 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 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 зчитуванням 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 є мнемонічне позначення умови відповідно до табл. 3. 1. 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. 3. 1 мнемокод – JP adr. Двійковий 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. Дана група команд також не впливає на ознаки. Оскільки більшість представлених в табл. 3. 1 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. 3. 5.

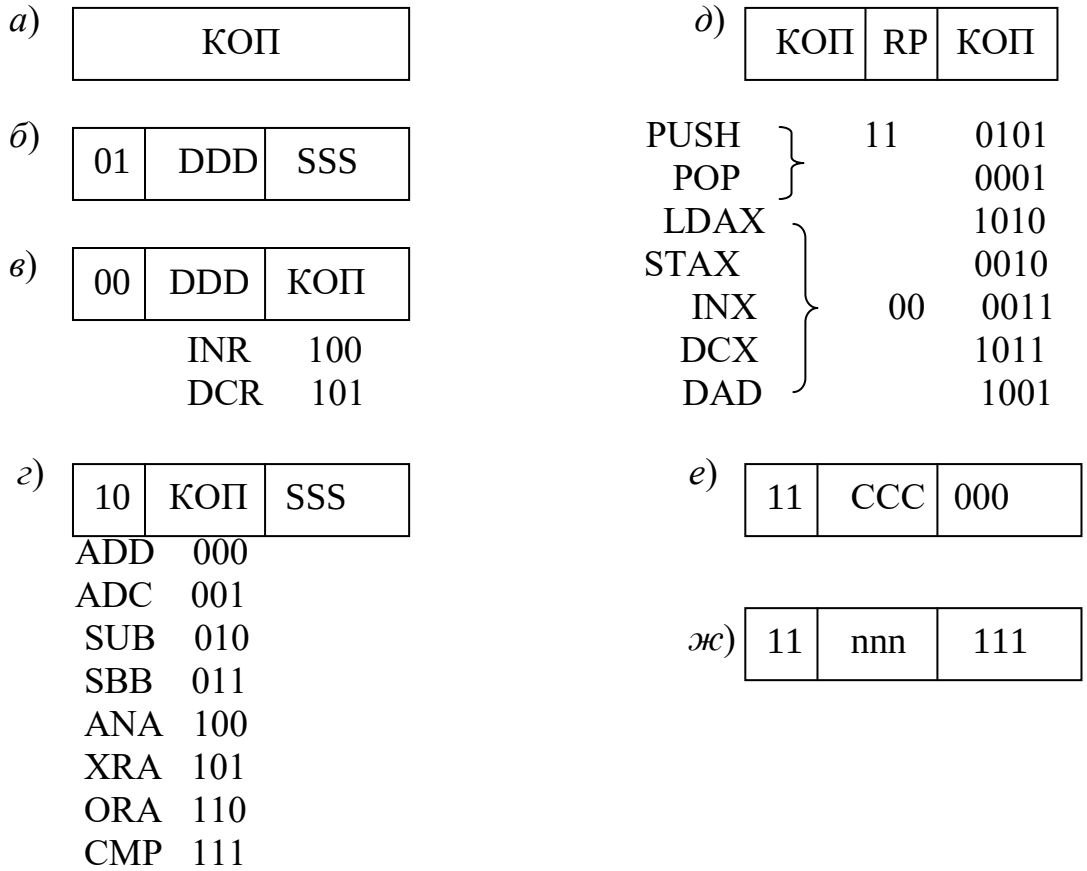

Рис. 3. 5 Варіанти однобайтових команд

Варіант *а* рис. 3. 5 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. 3. 2 наведені мнемокоди і двійкові значення КОП всіх команд варіанту *а.*

Таблиця 3. 2 - 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. 3. 5 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. 3. 3. 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 3. 3 Трирозрядні коди SSS або DDD, яким привласнені двійкові 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. 3. 5 передбачають регістрову адресацію з використанням одного РЗП (джерела або приймача інформації). Для цих варіантів представлені мнемокоди всіх відповідних ним команд і двійкові значення КОП.

Варіант *д* рис. 3. 5 визначає формат однобайтових команд, в яких беруть участь регістрові пари і покажчик стека. Приведені тут значення розрядів КОП разом з даними з табл. 3. 4 дозволяють отримати двійкові коди відповідних команд.

Таблиця 3. 4 - 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. 3. 5 разом з даними табл. 3. 5 може бути використаний для знаходження двійкових кодів однобайтових команд для організації переходів в програмах.

Таблиця 3. 5 - Трирозрядний код, що визначає ознака (Z, CY, P, S), 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. 3. 5, використовується для визначення адреси в команді RST.

Варіанти двобайтових команд приведені на рис. 3. 6

Варіант *і* рис. 3. 6 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. 3. 6 представляє комбінацію регістрової (приймач інформації) і безпосередньої (джерело інформації) адресацій.

У варіанті *л* рис. 3. 6, використовуваному в командах вводу-виводу, при n=0 задається ввід (IN), при n=1 – вивід (OUT). Другий байт визначає адресу пристрою вводу-виводу (adr port).

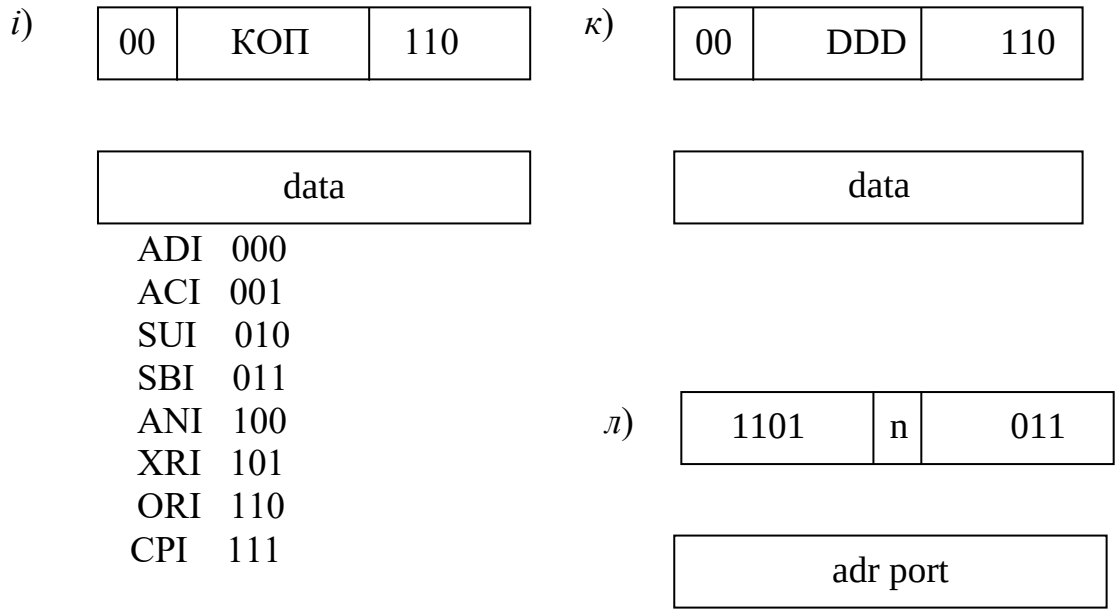

Рис. 3. 6 Варіанти двобайтових команд

Варіанти трибайтових команд представлені на рис. 3. 7.

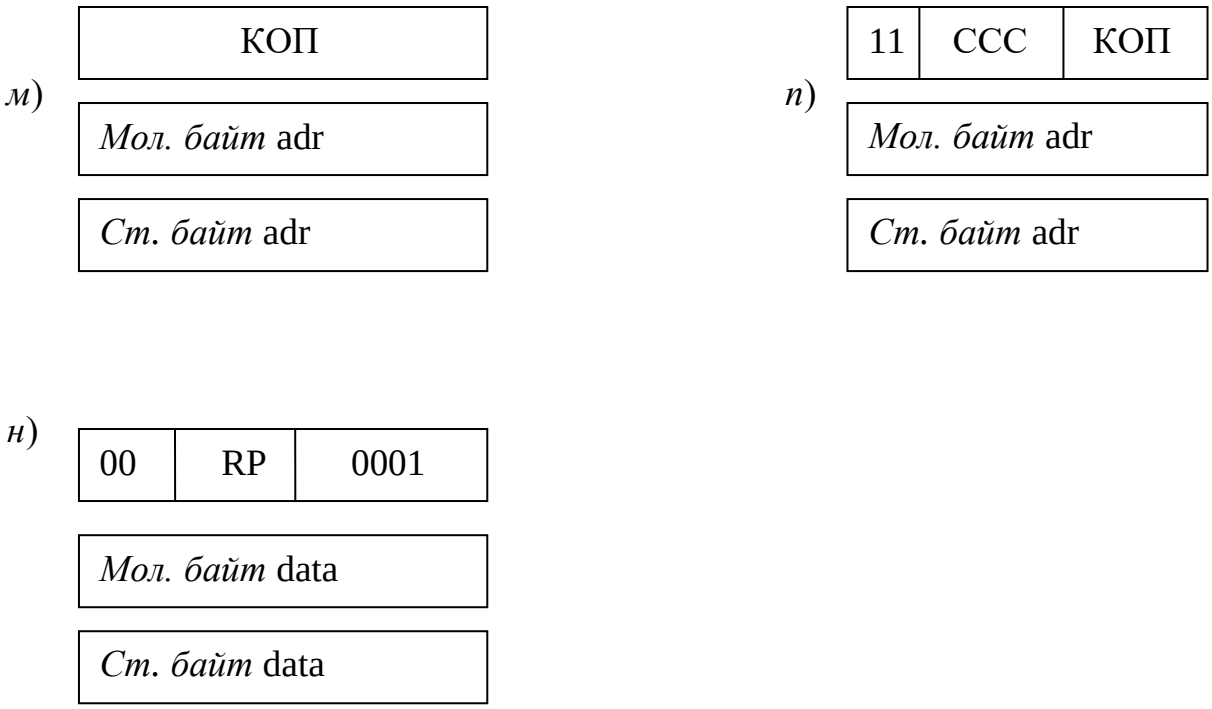

Рис. 3. 7 - Варіанти трибайтових команд

Пряма адресація відповідає варіанту *м*. Другий байт визначає молодші розряди, а третій – старші розряди. У таблиці 3. 6 наведені мнемокоди і двійкове значення КОП для всіх команд варіанту *м*.

Таблиця 3. 6 Мнемокоди і двійкове 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 вмісту 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 3. 4 (для мнемокоду LXI).

Варіант *п* використовує пряму адресацію для організації переходів в програмах на основі умов, що задається в таблиці 3. 5.

ЗАВДАННЯ

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 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 двійковій системі числення – індекс b, шістнадцятковій – індекс h, десятковій, – індекс d. Дані в оперативній пам'яті і стеку відбиваються в шістнадцятковій 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.

Скласти програми перемикання світлофора, світлофора зі стрілкою та керування об’єктом за особистим вибором.

У звіті привести алгоритми, коди програм та результати виконання.

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?
8. Перечисліть основні елементи структурної схеми МП.
9. Архітектура МП, її складові.

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. Гилмор Ч. Введение в микропроцессорную технику. - М., Мир, 1984. – 334 с.

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 за адресою, вказаною в регістрах HL, 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 M в регістр A (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 A даними з пам'яті M (в даному випадку комірка №7)

Таблиця А.9 Методи адресації(*непряма адресація*). Передача даних з пам'яті M в регістр A (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 A даними з пам'яті M (в даному випадку комірка №8)

Таблиця А.10 Методи адресації(*непряма адресація*). Передача даних з пам'яті M в регістр A (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 A даними з пам'яті M (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 до регістрів

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 з регістру B або C або D або E або H або L до числа 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 з комірки памяті M до числа 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 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 (логічне «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 (логічне «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» 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).