МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
"КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ"

МЕХАНІКО-МАШИНОБУДІВНИЙ ІНСТИТУТ

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт
з дисципліни "Мікропроцесорна техніка"
для студентів напрямку 6.050502 "Інженерна механіка"

Затверджено на засіданні кафедри
"Конструювання верстатів та машин"
Протокол №__ від "___"_____ 20__ р.

Київ – 2011

Методичні вказівки до виконання лабораторних робіт з дисципліни "Мікропроцесорна техніка" для студентів напрямку підготовки 6.050502 "Інженерна механіка" / Уклад. О.В. Самойленко, Ю.В. Єлісеєв. – К.: НТУУ "КПІ", 2011. – 50 с.

Укладачі: О.В. Самойленко
Ю.В. Єлісеєв
Відповідальний
редактор: В.Б. Струтинський
Рецензент: Ю.М. Данильченко

ЗМІСТ

ВСТУП

Основу пристроїв числового програмного керування верстатами і промисловими роботами складають мікропроцесорні системи, які використовують програмний спосіб обробки інформації, мають підвищену надійність, перешкодостійкість. Застосування мікропроцесорної техніки дає можливість реалізувати складні системи автоматичного керування приводами технологічного обладнання.

Курс "Мікропроцесорна техніка" є складовою частиною дисципліни "Електротехніка, електроніка та мікропроцесорна техніка" і базується на використанні основних положень курсів, що вивчались раніше: "Математика", "Фізика", "Інформатика". Він пов'язаний з подальшими курсами: "Системи керування верстатами", "Промислові роботи", "Гідропневмоавтоматика", "Системи керування верстатами та верстатними комплексами".

По даній дисципліні є лабораторні заняття, на яких вивчається система команд мікропроцесора та набуваються навички складання програм на мові асемблера для мікропроцесора КР580ИК80А.

Мета курсу – набуття студентами: знання принципів побудови мікропроцесорів, дій двійкової арифметики, системи команд програмного забезпечення та архітектури мікропроцесора, особливостей апаратних засобів мікропроцесорів та периферійних пристроїв; вміння складати рівняння логічних функцій виконувати різними методами мінімізацію системи логічних рівнянь, перетворювати дані з однієї системи числення в іншу, складати програми з використанням команд асемблера, кодувати керуюче слово паралельного інтерфейсу; практичних навичок по складанню простих програм з використанням команд пересилки даних, по запису і виконанню на мікроЕОМ програм з використанням команд арифметичних і логічних

операцій, по організації роботи мікропроцесора в режимі вводу-виводу інформації.

Предметом курсу є мікропроцесорні засоби, логічні схеми та елементи, що використовуються для створення систем керування виробничими процесами, а також система команд мікропроцесора для складання програм керування.

Для полегшення вивчення принципів програмування мікропроцесорних систем на низькому рівні на мові асемблера, а також набуття базових понять у галузі організації ЕОМ та МПС, використовується мікроЕОМ "Микролаб КР580ИК80 907", що побудована на базі мікропроцесора КР580ИК80.

Дана мікроЕОМ дозволяє: написання програм на мові асемблера, використовуючи систему команд мікропроцесора КР580ИК80, їх відладку та виконання в тактовому, командному та наскрізному режимах; вивчити принципи і порядок виконання команд; набути навичок роботи з зовнішніми пристроями мікропроцесорної системи; отримати уявлення про організацію зовнішньої і внутрішньої (реєстрової) пам'яті і стекової області.

1. ОБЛАДНАННЯ ДЛЯ ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

Мікропроцесорний комплекс КР580 виконаний за n-МДП і ТТЛШ технологіями. Він характеризується архітектурним єдністю, забезпечуваним автономністю та функціональної закінченістю окремих інтегральних мікросхем, уніфікацією їх інтерфейсу, програмовані, їх логічної та електричної сумісністю.

Мікропроцесорний комплекс КР580 відрізняють:

– 8-розрядна організація;

– фіксований набір команд;

– великий вибір периферійних великих інтегральних схем (ВІС) різного призначення;

– відносно високе швидкодія;

– помірне споживання потужності.

Особливості мікропроцесора КР580ИК80 (Intel 8080):

– 8-розрядна шина даних;

– 16-розрядна шина адреси (дозволяє адресувати до 64Кб пам'яті);

– можливість організації зворотного стека;

– система команд складається з 78 базових команд, що містять 111 кодових операцій;

– дозволяє працювати з 256-ма зовнішніми пристроями;

– інтеграція - 4500 транзисторів;

– 40-вивідних корпус;

– тактова частота 3,125МГц.

МікроЕОМ "Микролаб КР580ИК80 907" має в своєму розпорядженні такі засоби введення та виведення інформації (рис. 1.1).

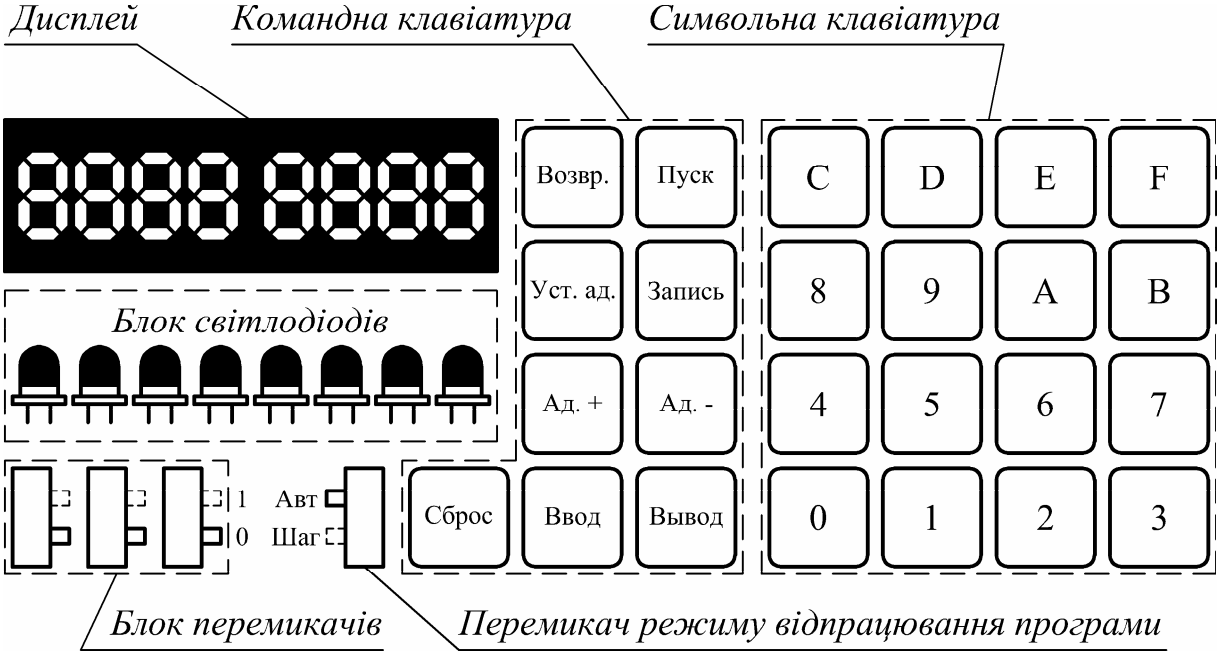


Рис. 1.1. Розташування засобів введення та виведення інформації

Введення інформації та керування виконанням програми реалізується за допомогою символьної та командної клавіатур, а також за допомогою блоку перемикачів. Виведення інформації здійснюється за допомогою дисплею та блоку світлодіодів.

1.1. Засоби введення інформації

Символьна клавіатура призначена для введення чисел в шістнадцятковій системі. Вона містить 16 клавіш, за допомогою яких можна вводити арабські цифри 0…9 та латинські літери A…F.

Командна клавіатура містить 9 клавіш, які за функціональним призначенням поділяються на дві групи.

Група клавіш *керування пам'яттю* складається з чотирьох клавіш, що мають таке призначення:

Уст. ад. — підключення до комірки пам'яті, адресу якої введено з символьної клавіатури;

Ад. + підключення до наступної комірки пам'яті;

Ад. - підключення до попередньої комірки пам'яті;

Запись запис даних, введених з числової клавіатури, до поточної комірки пам'яті (після запису відбувається автоматичний перехід до наступної комірки пам'яті);

Група клавіш *керування виконанням програми* складається з п'яти клавіш, які виконують наступні функції:

Пуск виконання програми з попередньо встановленої початкової адреси;

Сброс обнулення всіх розрядів дисплея (дані, записані за допомогою клавіші "Запись", зберігаються);

Возвр. повернення до виконуваної програми;

Ввод введення інформації із зовнішнього запам'ятовуючого пристрою (магнітофона);

Вывод виведення інформації на зовнішній запам'ятовуючий пристрій (магнітофон).

УВАГА! Клавіші "Возвр.", "Ввод" та "Вывод" при виконання лабораторних робіт не використовуються.

Блок перемикачів (рис. 1.2) містить 3 перемикачі, кожен з яких має 2 фіксованих положення: 0 – "вимкнено" та 1 – "увімкнено".

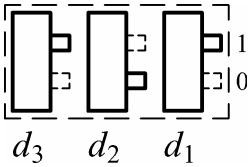


Рис. 1.2. Блок перемикачів

За допомогою блоку перемикачів можуть бути введені 3 розряди $(d_3 \ldots d_1)$ молодшого напівбайту числа. Перемикачі підключені до електронної схеми мікроЕОМ "Микролаб КР580ИК80 907" таким чином, що розряди старшого напівбайту числа $(d_7 \ldots d_4)$, а також розряд d_0 молодшого напівбайту завжди дорівнюватимуть нулю.

Фактично, з блоку перемикачів можуть бути введені лише 8 чисел:
$0000\ 0000_2 = 00_{16}$; $0000\ 0010_2 = 02_{16}$; $0000\ 0100_2 = 04_{16}$; $0000\ 0110_2 = 06_{16}$;
$0000\ 1000_2 = 08_{16}$; $0000\ 1010_2 = 0A_{16}$; $0000\ 1100_2 = 0C_{16}$; $0000\ 1110_2 = 0E_{16}$.

В даному випадку (рис. 1.2) з блоку перемикачів введено число $0000\ 1010_2 = 0A_{16}$.

1.2. Засоби виведення інформації

Дисплей (рис. 1.3) має 8 розрядів, розділених на дві групи по 4 розряди в кожній.

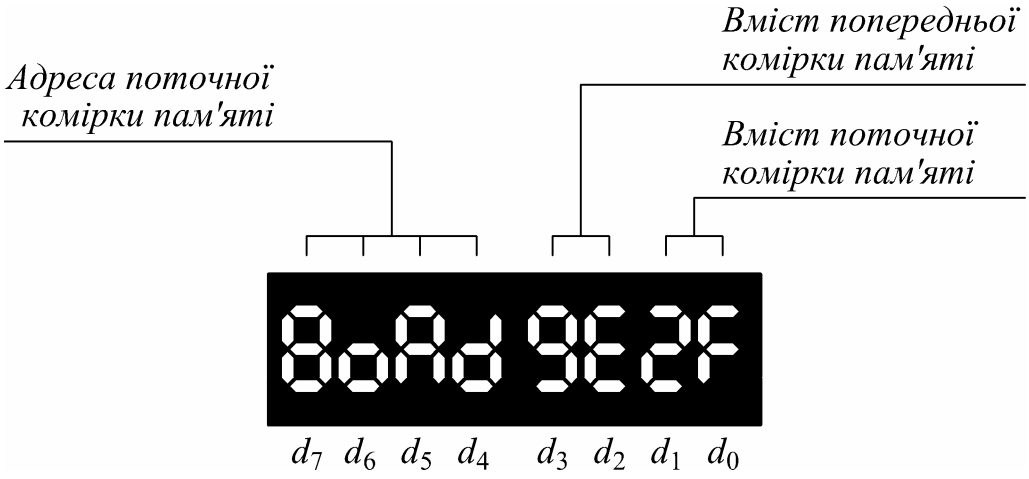


Рис. 1.3. Дисплей

Ліва група розрядів $(d_7 \ldots d_4)$ відображає адресу поточної комірки пам'яті, наприклад $80AD_{16}$.

Молодший байт (d_1, d_0) правої групи розрядів відображає вміст комірки пам'яті (в даному випадку він дорівнює $2F_{16}$), адреса якої відображе-

на в лівій групі розрядів. Старший байт (d_3, d_2) правої групи розрядів відображає вміст попередньої комірки пам'яті (в даному випадку він дорівнює $9E_{16}$).

Блок світлодіодів (рис. 1.4) містить 8 світлодіодів, кожен з яких може набувати 2 стани:  – 0 або "вимкнено" та  – 1 або "увімкнено".

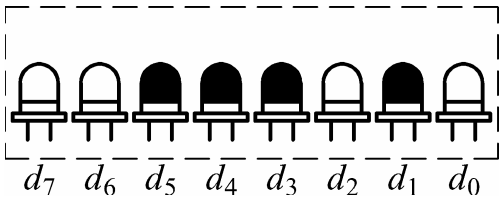


Рис. 1.4. Блок світлодіодів

Таким чином, за допомогою блоку світлодіодів може бути відображено будь-яке шістнадцяткове число від 00_{16} до FF_{16}. Старший розряд d_7 цього числа відображається на крайньому лівому світлодіоді, а молодший d_0 – на крайньому правому. Наприклад, на рис. 1.4 блок світлодіодів відображає число $0011\ 1010_2 = 3A_{16}$.

Слід звернути увагу, що після ввімкнення живлення мікроЕОМ "Микролаб КР580ИК80 907" всі світлодіоди знаходяться в стані "увімкнено".

Перемикач режиму відпрацювання програми (рис. 1.5) може бути умовно віднесений до групи клавіш керування виконанням програми.

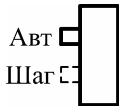


Рис. 1.5. Перемикач режиму відпрацювання програми

Він має два фіксовані положення: "Авт" та "Шаг".

В положенні "Авт" відбувається автоматичне виконання програми з початку до кінця. В положенні "Шаг" виконання програми здійснюється

покроково. Зазвичай даний режим використовують в процесі перевірки або налагодження програми.

1.3. Запис та зчитування інформації, пуск програми

Запис інформації до пам'яті мікроЕОМ "Микролаб КР580ИК80 907" складається з двох етапів:

1) підключення до потрібної комірки пам'яті;

2) введення даних до комірки пам'яті.

Підключення до потрібної комірки пам'яті, до якої необхідно записати дані, здійснюють наступним чином. Спочатку обнулюють розряди дисплея клавішею "Сброс". Далі з символьної клавіатури вводять адресу потрібної комірки, яка складається з чотирьох шістнадцяткових символів.

Введена адреса спочатку відображається на правій частині дисплею (розряди d_3, d_2, d_1, d_0). Потім натискають клавішу "Уст. ад.", після чого введена адреса відображається в лівій частині дисплею (розряди d_7, d_6, d_5, d_4). В молодших розрядах d_1, d_0 правої частини дисплею відображається вміст поточної комірки пам'яті, а в старших розрядах d_3, d_2 – вміст попередньої комірки пам'яті.

Запис даних до потрібної комірки пам'яті здійснюється так. Дані вводяться побайтно, тобто групами по два шістнадцяткових символи. Після введення кожної групи натискають клавішу "Запись".

При цьому введений байт записується до поточної комірки пам'яті. Одночасно здійснюється підключення до наступної комірки пам'яті, адреса якої також відображається в лівій частині дисплею (розряди d_7, d_6, d_5, d_4).

Якщо необхідно *переглянути* вміст будь-якої комірки пам'яті, то здійснюють підключення до неї, як це розглянуто вище. У випадку, коли

потрібно переглянути вміст однієї або декількох сусідніх комірок пам'яті. то користуються клавішами "Ад. +" та "Ад. –".

Наприклад, необхідно підключитися до комірки пам'яті з адресою $82BC_{16}$, після чого записати до пам'яті трьохбайтну команду, яка в шістнадцятковій системі має вигляд "3E 5D 00". При потребі здійснити перегляд записаного. Для цього натискають таку послідовність клавіш:

Клавіша	**Дисплей**	**Коментарі**
"Сброс"	0000 0000	Обнулення розрядів дисплею
"8"	0000 0008	Введення адреси потрібної комірки пам'яті з символьної клавіатури
"2"	0000 0082	
"B"	0000 082B	
"C"	0000 82BC	
"Уст. ад."	82BC BC**	Підключення до потрібної комірки пам'яті
"3"	82BC C**3	Введення першого байту команди
"E"	82BC **3E	
"Запись"	82BD 3E**	Запис першого байту команди до пам'яті
"5"	82BD 3E*5	Введення другого байту команди
"D"	82BD 3E5D	
"Запись"	82BE 5D**	Запис другого байту команди до пам'яті
"0"	82BE AD*0	Введення третього байту команди
"0"	82BE AD00	
"Запись"	82BF 00**	Запис третього байту команди до пам'яті
"Ад. –"	82BE 5D00	Перехід на два кроки назад
"Ад. –"	82BD 3E5D	
"Ад. +"	82BE 5D00	Перехід на один крок вперед

Для *запуску* вже записаної програми необхідно здійснити наступні процедури:

1) підключення до комірки пам'яті, в якій записано початковий байт програми;

2) безпосередньо здійснити запуск програми.

Підключення до потрібної комірки пам'яті не відрізняється від вищеописаного. Після підключення до комірки пам'яті з початковим байтом програми натискають клавішу "Пуск".

Наприклад, необхідно виконати програму, початок якої знаходиться в комірці пам'яті з адресою $80A0_{16}$. Для цього натискають таку послідовність клавіш:

Клавіша	**Дисплей**	**Коментарі**
"Сброс"	0000 0000	Обнулення розрядів дисплею
"8"	0000 0008	Введення адреси комірки пам'яті, в якій знаходиться початок програми, з символьної клавіатури
"0"	0000 0080	
"А "	0000 080A	
"0"	0000 80A0	
"Уст. ад."	80A0 A0**	Підключення до комірки пам'яті, в якій знаходиться початок програми
"Пуск"	80A0 A0**	Запуск виконання програми

Якщо перемикач режиму відпрацювання програми (рис. 1.5) встановлено в позицію "Авт", то після натиснення клавіші "Пуск" програма відпрацьовується повністю. Якщо перемикач режиму відпрацювання програми перебуває в позиції "Шаг", то програма відпрацьовується покроково; для виконання наступної операції програми слід натискати клавішу "Пуск". Слід зауважити, що обнулення розрядів дисплея клавішею "Сброс" не є обов'язковою процедурою і виконується, зазвичай, для зручності.

2. ПРОГРАМНІ РЕСУРСИ

Програма складається на *мові* "Асемблер – 80" відповідно до варіанту завдання і виконується на мікроЕОМ "Микролаб КР580ИК80 907", центральним процесором якої є *мікропроцесор* КР580ИК80А.

Мікропроцесор КР580ИК80А має доступні для програміста ресурси (рис. 2.1):

1. Один 8-розрядний регістр-акумулятор *A* та шість 8-розрядних регістрів загального призначення (*РЗП*) – *B*, *C*, *D*, *E*, *H*, *L*.

2. Один 8-розрядний регістр *F* ознак результатів. Три розряди регістра *F* не використовуються і мають постійні значення – $d_1 = 1$, $d_3 = 0$ та $d_5 = 0$. Інші розряди регістра ознак, які називаються прапорцями, відображають характеристики розміщеного в акумуляторі результату виконаної операції.

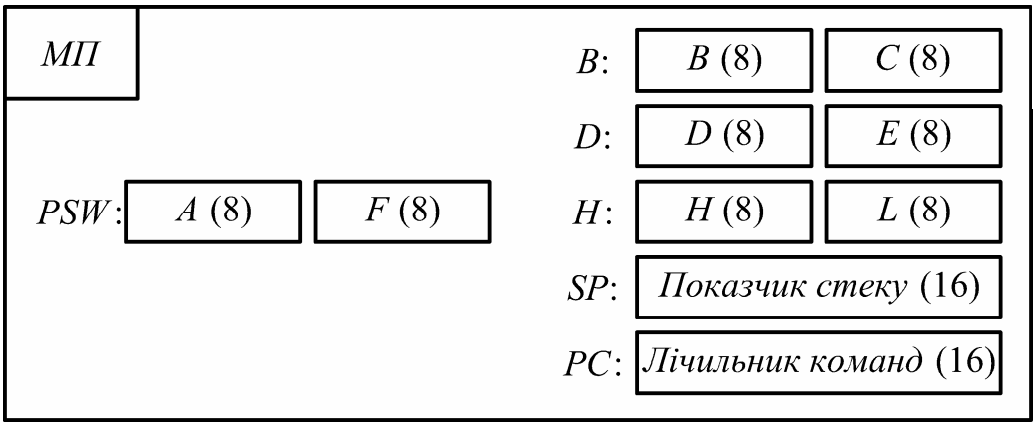


Рис. 2.1. Доступні ресурси мікропроцесора КР580ВМ80А

Регістр ознак має такий формат (рис. 2.2).

d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
S	*Z*	0	*AC*	0	*P*	1	*C*

Рис. 2.2. Розподіл ознак у регістрі F

Прапорці ознак фіксують результати виконаних дій:

S – прапорець знаку, встановлюється в стан логічної "1" при наявності в старшому розряді акумулятора "1", що відповідає від'ємному числу;

Z – прапорець нуля, встановлюється в стан логічної "1", якщо результат обчислення дорівнює нулю;

P – прапорець парності, встановлюється в стан логічної "1" при парній кількості "1" в числі, записаному в акумуляторі;

C – прапорець перенесення, встановлюється в стан логічної "1", якщо результат обчислення виходить за межі восьми розрядів;

AC – прапорець допоміжного перенесення, встановлюється в стан логічної "1", якщо в двійковому числі результату обчислень, який знаходиться в акумуляторі, було перенесення з третього в четвертий розряд. Цей прапорець аналізується при виконанні команди *DAA*.

3. Лічильник команд *PC* – 16-розрядний програмно доступний регістр, у якому зберігається адреса наступної команди.

Основною одиницею інформації, яка оброблюється в мікропроцесорі, є *байт*, тобто машинне слово, що складається з 8 двійкових розрядів (*бітів*).

Ряд команд дає можливість виконувати операції над 16-ти розрядним словами, використовуючи не окремі регістри, а регістрові пари – *BC*, *DE*, *HL* та *AF*, звертання до яких здійснюється за іменами старших регістрів – *B, D, H* та *PSW* (слово стану програми).

Оперативний (ОЗП) та постійний (ПЗП) запам'ятовуючі: пристрої призначені для зберігання програми та даних. В мікроЕОМ "Микролаб КР580ИК80 907" пам'ять розміщена за такими адресами: ПЗП – 0000_{16}…$05FF_{16}$; ОЗП – 8000_{16}…$83FF_{16}$. Таким чином, місткість ПЗП – 1,5 *Кбайт*, ОЗП – 1 *Кбайт*.

3. РЕКОМЕНДАЦІЇ ПО ВИКОНАННЮ ЛАБОРАТОРНИХ РОБІТ

Рекомендується такий загальний порядок дій при виконанні лабораторних робіт з дисципліни "Мікропроцесорна техніка":

І. Отримати **завдання** у викладача. Завдання може бути індивідуальне або бригадне.

Робоча програма дисципліни "Мікропроцесорна техніка" протягом семестру передбачає виконання трьох лабораторних робіт різного рівня складності. Лабораторна робота полягає в написанні та відладці програми на мові програмування "Асемблер – 80". Кожне завдання має спеціальний шифр, який складається з трьох символів: однієї латинської літери і двозначного числа. Літера позначає номер лабораторної роботи ("А" – перша, "В" – друга і "С" – третя), а число – номер варіанту в межах однієї лабораторної роботи.

Перша лабораторна робота

ІІ. Уважно **ознайомитись** із завданням. При необхідності уточнити завдання у викладача.

Завдання на лабораторну роботу полягає в певних маніпуляціях з числовими даними (арифметичні і логічні операції, перевірка умов тощо). Слід звернути увагу, що в умові задачі числові дані можуть бути представлені в різних системах числення: двійковій, вісьмірковій, десятковій та шістнадцятковій. В кінці кожного числа в умові задачі є спеціальний символ, який вказує на систему числення, в якій це число представлене.

Число в *двійковій* системі може мати вигляд $****\ ****_2$ або ****** ******B*, де * – будь-яка двійкова цифра (0 або 1).

Число в *вісімковій* системі може мати вигляд $***_8$, де * – будь-яка арабська цифра від 0 до 7.

Число в *десятковій* системі може не позначатись взагалі або мати вигляд $***_{10}$ чи ***D.

Число в *шістнадцятковій* системі може мати вигляд $**_{16}$ або **H, де * – будь-який шістнадцятковий символ (арабські цифри від 0 до 9 та латинські літери від A до F).

Іноді може зустрічатись числа, представлені в *четвірковій* системі. Вони мають вигляд $****_4$ або ****Q, де * – арабська цифра від 0 до 3.

УВАГА! При виконанні лабораторних робіт всі числа мають бути переведені в шіснадцяткову систему.

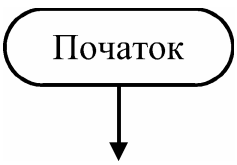
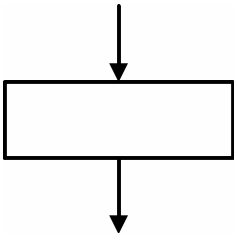
III. Переписати завдання у **символьному вигляді**, використовуючи загальноприйняті математичні символи та умовні позначення.

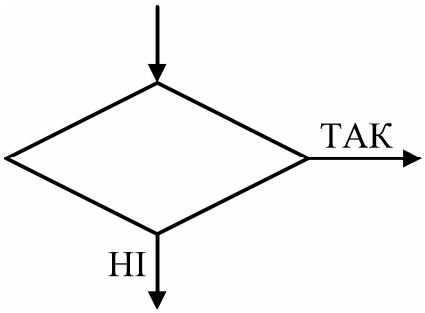
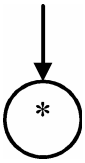
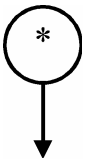
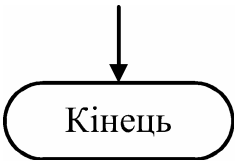
Представлення завдання в символьному вигляді не є обов'язковим і може бути застосованим для наочності та покращення його сприйняття студентом. При переведенні завдання в символьний вид використовують загальновживані математичні символи.

IV. Скласти **блок-схему алгоритму** програми. Уважно її перевірити.

Для складання блок-схеми алгоритму програми використовують спрощений набір умовний позначень, прийнятих у програмуванні (табл. 3.1).

Таблиця 3.1

Зображення блоку	Опис блоку
	Початок (запуск) роботи програми. В переважній більшості випадків даний блок лише символізує початок програми і не підкріплений конкретною командою.
	Блок обробки даних (арифметична, логічна чи спеціальна операції, пересилання даних тощо). Всередині блоку розміщується символьний запис виконуваної операції.

Зображення блоку	Опис блоку
	Блок перевірки умови (стану прапорців). Всередині блоку в символьному вигляді наводиться умова. що перевіряється. Відповідає командам умовних переходів (див. п. 4.6.2). Оскільки при виконанні умови команда здійснює перехід до виконання підпрограми, то при побудові блок-схеми алгоритму рекомендується вихід "ТАК" спрямовувати вбік від основного напрямку розвитку блок-схеми.
	Закінчення гілки блок-схеми алгоритму перед розривом (* – номер розриву).
	Початок гілки блок-схеми алгоритму після розриву (* – номер розриву).
	Стрілка вказує на напрямок переходу керування в блок-схемі алгоритму від одного блоку до іншого. Лінія може бути ламаною.
	Кінець (зупинка) роботи програми. Відповідає спеціальній команді (див. п. 4.5.6).

V. За складеною блок-схемою записати **текст програми**.

Текст програми записується у вигляді таблиці наступного вигляду (табл. 3.2).

Таблиця 3.2

Адреса	Машинний код	Мітка	Мнемоніка	Коментар
...	...	...	...	...

Адреса вказується чотиризначним шістнадцятковим числом в межах, обумовлених запам'ятовуючими пристроями мікроЕОМ "Микролаб КР580ИК80 907" (див. р. 2).

Машинний код, призначений для введення з клавіатури мікроЕОМ "Микролаб КР580ИК80 907", являє собою послідовність двозначних шіснадцяткових чисел (див. п. 1.3).

Мітки використовуються при написанні тих програм, в яких використовуються умовні або безумовний переходи. Фактично мітка – це та адреса, на яку здійснюється перехід. Мітки позначаються M_1, M_2, M_3 тощо.

В колонці "Мнемоніка" записують мнемонічні коди команд разом з операндами (див. р. 4).

В колонці "Коментар" як правило наводять словесний або символьний опис операції (див. р. 4).

VI. Здійснити **ручний підрахунок** задачі.

Ручний підрахунок задачі особливо актуальний при виконанні першої і другої лабораторних робіт.

VII. Ввести програму в пам'ять мікроЕОМ.

VIII. Запустити програму.

Процес введення машинного коду програми до пам'яті мікроЕОМ "Микролаб КР580ИК80 907" та її запуску наведений в п. 1.3.

IX. Порівняти результати роботи програми та ручного підрахунку. При необхідності здійснити відладку програми.

Результати роботи програми в межах першої та другої лабораторних робіт записуються в задані умовою комірки пам'яті. Тому перед запуском програми необхідно обнулити ці комірки пам'яті. Результатом роботи програми по третій лабораторній роботі виводяться на блок світлодіодів і тому є наочними.

Співпадіння результатів ручного підрахунку з результатами роботи програми свідчить про правильність написання останньої.

X. Оформити протокол лабораторної роботи і винести її на захист.

Протокол лабораторної роботи оформлюється студентом індивідуально і повинен містити таку інформацію:

– прізвище та ініціали студента-виконавця;

– шифр групи;

– шифр завдання на лабораторну роботу;

– текст завдання повністю;

– завдання у символьному вигляді (не обов'язково);

– блок-схема алгоритму програми;

– текст програми.

4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Програма – це послідовність команд, за допомогою яких здійснюється обробка інформації. Команди в мікропроцесоорі КР580ИК80А мають одно-, дво- та трибайтні формати. В першому байті або його частині вказують код операції (КОП), а в частинах і байтах, що залишились – *операнди* або адреси пристроїв, де знаходяться операнди. Таким чином створюються нуль, одно-, та двоадресні команди. У деяких командах разом із кодом операції може безпосередньо передаватися операнд, необхідний для виконання цієї команди.

Передача кодів операцій, адрес і безпосередніх даних здійснюється по 8-розрядній шині даних, а звертання до комірок пам'яті здійснюється по 16-розрядній адресній шині (АШ). В таких командах крім КОП (1-й байт) у 2-му та 3-му байтах вказується адреса комірки пам`яті, з якою мікропроцесор буде здійснювати обмін інформацією.

Команди можуть бути одно-, дво-, або трьохбайтними, і, відповідно, можуть займати одну, дві або три сусідні комірки пам'яті (адреса кожної сусідньої буде відрізнятися на одиницю).

Блок керування та синхронізації мікропроцесора розшифровує двійковий код команди, що надійшла до регістра команд, визначає кількість байтів, яка потрібна для повного зчитування команди, автоматично збільшуючи після цього на одиницю вміст програмного лічильника (РС) при кожному звертанні до ОЗП. Після виконання поточної команди вміст програмного лічильника буде збільшено на одиницю, тобто він містить адресу комірки пам'яті, у якій зберігається КОП наступної команди.

Мікропроцесор здійснює обробку інформації згідно команд програми, які представлені в машинних (двійкових) кодах. Коди команд і дані, записані у двійковій системі числення, важко запам'ятовувати і вводити їх

у мікропроцесор. Значно зручніші шістнадцяткові коди, які використовуються в мікроЕОМ "Микролаб КР580ИК80 907". При написанні програм використовують мнемонічне зображення кодів, тобто скорочений запис назви команди, що складається з двох, трьох або чотирьох букв латинського алфавіту.

4.1. Пересилання даних

Для обміну інформації між регістрами мікропроцесора, а також між мікропроцесором і комірками пам'яті запам'ятовуючого пристрою використовують команди пересилання даних.

УВАГА! Команди пересилання *не змінюють* стан прапорців (ознак результатів) регістра *F*.

Команди ***MOV r1, r2*** використовують для пересилання даних між регістрами загального призначення. Цією командою дані, які записані в регістрі *r*2 (джерело інформації) пересилають у регістр *r*1 (приймач інформації). При цьому вміст джерела інформації залишається незмінним.

Мнемоніка: *MOV r*1, *r*2
Символьний запис: (*r*1) ← (*r*2)
Кількість байтів: 1
Формат команди: КОП

Команди ***MOV r, M*** використовують для пересилання в регістр *r* вмісту комірки пам'яті, адреса якої попередньо записана в регістровій парі *H, L*.

Мнемоніка: *MOV r, M*
Символьний запис: (*r*) ← ((*H*)(*L*))
Кількість байтів: 1
Формат команди: КОП

Команди ***MOV M, r*** використовують для пересилання вмісту регістру *r* до комірки пам'яті *M*, адреса якої попередньо записана в регістровій парі *H, L*.

Мнемоніка: *MOV M, r*

Символьний запис: ((*H*)(*L*)) ← (*r*)

Кількість байтів: 1

Формат команди: КОП

Команди ***MVI r, ⟨байт⟩*** використовують для безпосереднього пересилання даних, які записані в 2-му байті команди, до регістру *r*.

Мнемоніка: *MVI r, ⟨байт⟩*

Символьний запис: (*r*) ← ⟨*байт*⟩

Кількість байтів: 2

Формат команди: КОП, ⟨*байт*⟩

Команди ***MVI M, ⟨байт⟩*** використовують для безпосереднього пересилання даних, які записані в 2-му байті команди, в комірку пам'яті *M*, адреса якої записана в регістровій парі *H*, *L*.

Мнемоніка: *MVI M, ⟨байт⟩*

Символьний запис: ((*H*)(*L*)) ← ⟨*байт*⟩

Кількість байтів: 2

Формат команди: КОП, ⟨*байт*⟩

Команду ***LXI rp, ⟨2 байти⟩*** використовують для завантаження регістрової пари *rp* адресою комірки пам'яті, вказаної у 2-му і 3-му байтах команди, причому 3-й байт команди пересилається в старший регістр *rh*, а 2-й байт команди – в молодший регістр *rl* вказаної регістрової пари *rp*. Тут *rp* – одна з регістрових пар *B* (*B*, *C*), *D* (*D*, *E*), *H* (*H*, *L*) або регістр *SP* – показник стеку.

Мнемокод: *LXI rp, ⟨2 байти⟩*

Символьний запис: (*rh*) ← ⟨*байт* 3⟩ та (*rl*) ← ⟨*байт* 2⟩

Кількість байтів: 3

Формат команди: КОП, ⟨*байт* 3⟩, ⟨*байт* 2⟩

Команду ***LDA ⟨адреса⟩*** використовують для прямого завантаження акумулятора *A* вмістом комірки пам'яті, адреса якої вказується в 2-му і 3-му байтах команди.

Мнемокод: *LDA ⟨адреса⟩*

Символьний запис: (*A*) ← (⟨*байт* 3⟩, ⟨*байт* 2⟩)

Кількість байтів: 3

Формат команди: КОП, ⟨*байт* 3⟩, ⟨*байт* 2⟩

Команду ***STA ⟨адреса⟩*** використовують для прямого запису вмісту акумулятора *A* в комірку пам'яті, адреса якої вказується в 2-му і 3-му байтах команди.

Мнемокод: *STA* ⟨*адреса*⟩

Символьний запис: (⟨*байт* 3⟩, ⟨*байт* 2⟩) ← (*A*)

Кількість байтів: 3

Формат команди: КОП, ⟨*байт* 3⟩, ⟨*байт* 2⟩

Команду ***LDAX rp*** використовують для непрямого завантаження акумулятора *A* вмістом комірки пам'яті, адреса якої вказана в регістровій парі *rp*, причому можуть бути вказані регістрові пари *B* або *D*.

Мнемокод: *LDAX* (*rp*)

Символьний запис: (*A*) ← ((*rp*))

Кількість байтів: 1

Формат команди: КОП

Команду ***STAX rp*** використовують для непрямого завантаження вмістом акумулятора *A* комірки пам'яті, адреса якої вказана в регістровій парі *rp*, причому можуть бути вказані регістрові пари *B* або *D*.

Мнемокод: *STAX rp*

Символьний запис: ((*rp*)) ← (*A*)

Кількість байтів: 1

Формат команди: КОП

Регістрова пара *H*, *L* виконує функції регістра пам'яті в командах непрямої адресації. Тому в мікропроцесорі є декілька команд, які призначені для роботи з регістровою парою *H*, *L*.

Команду ***LHLD ⟨2 байти⟩*** використовують для прямого завантаження регістрової пари *H, L*. При виконанні цієї програми вміст комірки пам'яті, адреса якої записана в 2-му та 3-му байтах команди, завантажується в регістр *L*, а в регістр *H* завантажується вміст наступної комірки пам'яті, тобто тієї, адреса якої більша на одиницю.

Мнемокод: *LHLD* ⟨2 *байти*⟩

Символьний запис: (*L*) ← (⟨*байт* 3⟩ ⟨*байт* 2⟩) та (*H*) ← (⟨*байт* 3⟩ ⟨*байт* 2⟩ + 1)

Кількість байтів: 3

Формат команди: КОП, ⟨*байт* 3⟩, ⟨*байт* 2⟩

Команду ***SHLD ⟨2 байти⟩*** використовують для запам'ятовування вмісту регістрів *H* та *L* в двох комірках пам'яті. Адресу першої задано в команді, а адреса другої буде автоматично збільшена на одиницю.

Мнемокод: *SHLD* ⟨2 *байти*⟩

Символьний запис: (⟨*байт* 3⟩ ⟨*байт* 2⟩) ← (*L*) та (⟨*байт* 3⟩ ⟨*байт* 2⟩ + 1) ← (*H*)

Кількість байтів: 3

Формат команди: КОП, ⟨*байт* 3⟩, ⟨*байт* 2⟩

Команду ***XCHG*** використовують тоді, коли регістри *H* та *L* необхідно тимчасово використати для інших цілей, а їх вміст потрібно зберегти для подальшого використання в програмі. Для тимчасового збереження вмісту регістрів *H* та *L* використовуються регістри *D* та *E*.

Мнемокод: *XCHG*

Символьний запис: (*H*) ↔ (*D*) та (*L*) ↔ (*E*)

Кількість байтів: 1

Формат команди: КОП

Приклад. Потрібно число з комірки пам'яті, адреса якої $01C3_{16}$, переслати до регістра-акумулятора *A*, копію зберегти в регістрі *B*, інвертувати вміст акумулятора *A*, результат записати в комірку пам'яті, адреса якої 8050_{16}, закінчити програму.

Мнемоніка	**Коментар**
LDA $01C3_{16}$	Записати в акумулятор вміст комірки пам'яті, адреса якої $01C3_{16}$
MOV B, A	Переслати вміст акумулятора в регістр *B*
CMA	Інвертувати вміст регістру акумулятора
STA 8050_{16}	Записати результат (вміст акумулятора) в комірку пам'яті, адреса якої 8050_{16}
HLT	Зупинка мікропроцесора

Для занесення програми до пам'яті МікроЕОМ "Микролаб КР580ИК80 907" необхідно представити мнемонічні коди команд у машинних кодах і розмістити в комірках пам'яті. Залежно від формату команд (одно-, дво- або трьохбайтні) їх розміщення потребує відповідно одну, дві або три комірки пам'яті. Запишемо програму за такою формою:

Адреса	Машинний код	Мнемоніка	Коментар
8000	3A	*LDA* 01C3	Запис в регістр *A* (акумулятор) вмісту комірки пам'яті $01C3_{16}$
8001	C3		
8002	01		
8003	47	*MOV B, A*	Копіювання в регістр *B* вмісту регістру *A*
8004	2F	*CMA*	Інвертування вмісту регістру *A*
8005	32	*STA* 8050	Запис результату (вміст регістру *A*) в комірку пам'яті 8050_{16}
8006	50		
8007	80		
8008	76	*HLT*	Зупинка мікропроцесора

Розміщення в пам'яті двохбайтного операнда (адреси) здійснюють, починаючи з молодшого байта.

Наприклад, команду *LDA* $01C3_{16}$ розмістити в пам'яті починаючи з адреси 8000_{16}: в $8000_{16} \rightarrow$ 3A (код команди), в $8001_{16} \rightarrow$ C3 (молодший байт адреси), а в $8002_{16} \rightarrow$ 01 (старший байт адреси).

Програму на папері можна записати, вказуючи тільки початкову адресу кожної команди з урахуванням її формату. Залежно від формату (1…3 байти) команда може займати в пам'яті від однієї до трьох послідовно розташованих комірок пам'яті. Таку форму запису ілюструє наступна програма, в якій використані команди непрямої регістрової адресації та інвертування даних.

Адреса	Машинний код	Мнемоніка	Коментар
8010	21 50 80	*LXI H,* 8050	Записати в регістрову пару *H* число 8050_{16} – адресу комірки пам'яті
8013	7E	*MOV A, M*	Переслати в регістр *A* вміст комірки пам'яті, адреса якої вказана в регістровій парі *H*
8014	2F	*CMA*	Інвертувати вміст регістру *A*
8015	23	*INX H*	Збільшити вміст регістрової пари *H* на одиницю
8016	77	*MOV M,* A	Переслати вміст регістру *A* в пам'ять за адресою, зазначеною в регістровій парі *H*
8017	76	*HLT*	Зупинка мікропроцесора

4.2. Виконання арифметичних і логічних операцій

Команди цієї групи виконують арифметичні або логічні перетворення даних в арифметико-логічному пристрої (АЛП) МП.

Алгоритм виконання арифметичних і логічних команд має передумову, що перший операнд (зменшуване або 1-й доданок) обов'язково знаходиться в акумуляторі. Другий операнд може перебувати в одному з РЗП, комірці пам'яті або вказується безпосередньо в команді. Виконання операції завершується пересиланням результату в акумулятор, замість операнду, який знаходився там на початку виконання команди.

Отже, команди *арифметичних* та *логічних* операцій виконуються над вмістом:

– акумулятора і регістра.

– акумулятора й комірки пам'яті, адреса якої попередньо зазначена в регістровій парі *H*, *L*.

– акумулятора і даних, які безпосередньо вказані в другому байті команди.

УВАГА! Розглянуті команди арифметичних операцій можуть виробляти всі ознаки результату: *S, Z, AC, P, C.*

4.2.1. Арифметичні операції

МікроЕОМ "Микролаб КР580ИК80 907" може виконувати такі арифметичні операції:

– додавання двійкових чисел;

– віднімання двійкових чисел;

– інвертування двійкових чисел;

– збільшення на одиницю вмісту регістра (регістрової пари);

– зменшення на одиницю вмісту регістра (регістрової пари).

До арифметичних операцій слід також віднести операції ділення на 2 (див. п. 4.4.2) та множення на деяке ціле число (див п. 3.4.3).

Команду ***ADD r*** застосовують для додавання до вмісту акумулятора числа, яке знаходиться в регістрі *r*.

Мнемоніка: *ADD r*
Символьний запис: (*A*) ← (*A*) + (*r*)
Кількість байтів: 1
Формат команди: КОП
Ознаки результату: змінюються *S, Z, P, C* та *AC*

Команду ***ADD M*** застосовують для додавання до вмісту акумулятора числа, що знаходиться в комірці пам'яті, адреса якої вказана в регістровій парі *H*, *L*.

Мнемоніка: *ADD M*
Символьний запис: (*A*) ← (*A*) + ((*H*)(*L*))
Кількість байтів: 1
Формат команди: КОП
Ознаки результату: змінюються *S, Z, P, C* та *AC*

Команду ***ADI ⟨байт⟩*** використовують для додавання до вмісту акумулятора даних, які безпосередньо записані в другому байті команди.

Мнемоніка: *ADI ⟨байт⟩*
Символьний запис: (*A*) ← (*A*) + ⟨*байт*⟩
Кількість байтів: 2
Формат команди: КОП, ⟨*байт*⟩
Ознаки результату: змінюються *S, Z, P, C* та *AC*

Цей набір команд використовується для *додавання молодших (перших) байтів* двійкових чисел.

Наступна аналогічна група команд застосовується для *додавання старших байтів* (починаючи з другого) двійкових чисел, додаючи до отриманої суми значення розряду *C* регістра ознак.

Команду ***ADC r*** використовують для додавання до вмісту акумулятора числа, яке знаходиться в регістрі *r*, та біту прапорця *C* переносу.

Мнемоніка: *ADC r*

Символьний запис: (*A*) ← (*A*) + (*r*) + *C*

Кількість байтів: 1

Формат команди: КОП

Ознаки результату: змінюються *S*, *Z*, *P*, *C* та *AC*

Команду ***ADC M*** застосовують для додавання до вмісту акумулятора числа, що знаходиться в комірці пам'яті, адреса якої вказана в регістровій парі *H*, *L*, та біту прапорця *C* переносу.

Мнемоніка: *ADC M*

Символьний запис: (*A*) ← (*A*) + ((*H*)(*L*)) + *C*

Кількість байтів: 1

Формат команди: КОП

Ознаки результату: змінюються *S*, *Z*, *P*, *C* та *AC*

Команду ***ACI ⟨байт⟩*** використовують для додавання до вмісту акумулятора даних, які безпосередньо записані в другому байті команди, та біту прапорця *C* переносу.

Мнемоніка: *ACI ⟨байт⟩*

Символьний запис: (*A*) ← (*A*) + ⟨*байт*⟩ + *C*

Кількість байтів: 2

Формат команди: КОП, ⟨*байт*⟩

Ознаки результату: змінюються *S*, *Z*, *P*, *C* та *AC*

Для *віднімання молодших байтів* (старших байтів з позикою) даних використовують наступні команди.

Команду ***SUB r*** застосовують для віднімання від вмісту акумулятора числа, яке знаходиться в регістрі *r*.

Мнемоніка: *SUB r*

Символьний запис: (*A*) ← (*A*) – (*r*)

Кількість байтів: 1

Формат команди: КОП

Ознаки результату: змінюються *S*, *Z*, *P*, *C* та *AC*

Команду ***SUB M*** застосовують для віднімання від вмісту акумулятора числа, що знаходиться в комірці пам'яті, адреса якої вказана в регістровій парі *H*, *L*.

Мнемоніка: *SUB M*

Символьний запис: (*A*) ← (*A*) – ((*H*)(*L*))

Кількість байтів: 1

Формат команди: КОП

Ознаки результату: змінюються *S*, *Z*, *P*, *C* та *AC*

Команду ***SUI ⟨байт⟩*** використовують для віднімання від вмісту акумулятора даних, які безпосередньо записані в другому байті команди.

Мнемоніка: *SUI ⟨байт⟩*

Символьний запис: (*A*) ← (*A*) – ⟨*байт*⟩

Кількість байтів: 2

Формат команди: КОП, ⟨*байт*⟩

Ознаки результату: змінюються *S*, *Z*, *P*, *C* та *AC*

Команду ***SBB r*** використовують для віднімання від вмісту акумулятора числа, яке знаходиться в регістрі *r*, та біту прапорця *C* переносу.

Мнемоніка: *SBB r*

Символьний запис: (*A*) ← (*A*) – (*r*) – *C*

Кількість байтів: 1

Формат команди: КОП

Ознаки результату: змінюються *S*, *Z*, *P*, *C* та *AC*

Команду ***SBB M*** застосовують для віднімання від вмісту акумулятора числа, що знаходиться в комірці пам'яті, адреса якої вказана в регістровій парі *H*, *L*, та біту прапорця *C* переносу.

Мнемоніка: *SBB M*

Символьний запис: (*A*) ← (*A*) – ((*H*)(*L*)) – *C*

Кількість байтів: 1

Формат команди: КОП

Ознаки результату: змінюються *S*, *Z*, *P*, *C* та *AC*

Команду ***SBI ⟨байт⟩*** використовують для віднімання від вмісту акумулятора даних, які безпосередньо записані в другому байті команди, та біту прапорця *C* переносу.

Мнемоніка: *SBI ⟨байт⟩*

Символьний запис: (*A*) ← (*A*) – ⟨*байт*⟩ – *C*

Кількість байтів: 2

Формат команди: КОП, ⟨*байт*⟩

Ознаки результату: змінюються *S*, *Z*, *P*, *C* та *AC*

Для збільшення та зменшення на одиницю вмісту регістру, комірки пам'яті або регістрової пари використовують наступні команди.

Команду ***INR r*** використовують для збільшення на одиницю вмісту регістру *r*.

Мнемоніка: *INR r*
Символьний запис: (*A*) ← (*A*) + 1
Кількість байтів: 1
Формат команди: КОП
Ознаки результату: змінюються *S*, *Z*, *P* та *AC*

Команду ***INR M*** використовують для збільшення на одиницю числа, що знаходиться в комірці пам'яті, адреса якої вказана в регістровій парі *H*, *L*.

Мнемоніка: *INR M*
Символьний запис: ((*H*)(*L*)) ← ((*H*)(*L*)) + 1
Кількість байтів: 1
Формат команди: КОП
Ознаки результату: змінюються *S*, *Z*, *P* та *AC*

Команду ***DCR r*** використовують для зменшення на одиницю вмісту регістру *r*.

Мнемоніка: *DCR r*
Символьний запис: (*A*) ← (*A*) – 1
Кількість байтів: 1
Формат команди: КОП
Ознаки результату: змінюються *S*, *Z*, *P* та *AC*

Команду ***DCR M*** використовують для зменшення на одиницю числа, що знаходиться в комірці пам'яті, адреса якої вказана в регістровій парі *H*, *L*.

Мнемоніка: *DCR M*
Символьний запис: ((*H*)(*L*)) ← ((*H*)(*L*)) – 1
Кількість байтів: 1
Формат команди: КОП
Ознаки результату: змінюються *S*, *Z*, *P* та *AC*

Команду ***INX rp*** використовують для збільшення на одиницю вмісту регістрової пари *rp*.

Мнемоніка: *INX rp*

Символьний запис: $(rh)(rl) \leftarrow (rh)(rl) + 1$

Кількість байтів: 1

Формат команди: КОП

Ознаки результату: не змінюються

Команду ***DCX rp*** використовують для зменшення на одиницю вмісту регістрової пари *rp*.

Мнемоніка: *DCX rp*

Символьний запис: $(rh)(rl) \leftarrow (rh)(rl) - 1$

Кількість байтів: 1

Формат команди: КОП

Ознаки результату: не змінюються

4.2.2. Логічні операції

МікроЕОМ "Микролаб КР580ИК80 907" може виконувати такі логічні операції (табл. 4.1):

– логічне множення ("І");

– логічне додавання ("АБО");

– сума по модулю "2" ("ВИКЛ. АБО").

Таблиця 4.1

Логічне множення $Z = X \wedge Y$		
X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

Логічне додавання $Z = X \vee Y$		
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

Сума по модулю "2" $Z = X \oplus Y$		
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

Команду ***ANA r*** застосовують для виконання логічної операції "І" (логічного множення) над вмістом акумулятора та регістру *r*.

Мнемоніка: *ANA r*

Символьний запис: (*A*) ← (*A*) ∧ (*r*)

Кількість байтів: 1

Формат команди: КОП

Ознаки результату: змінюються *S*, *Z*, *P*, *C* та *AC*

Команду ***ANA M*** застосовують для виконання логічної операції "І" (логічного множення) над вмістом акумулятора та комірки пам'яті, адреса якої вказана в регістровій парі *H*, *L*.

Мнемоніка: *ANA M*

Символьний запис: (*A*) ← (*A*) ∧ ((*H*) (*L*))

Кількість байтів: 1

Формат команди: КОП

Ознаки результату: змінюються *S*, *Z*, *P*, *C* та *AC*

Команду ***ANI ⟨байт⟩*** застосовують для виконання логічної операції "І" (логічного множення) над вмістом акумулятора та другого байту команди.

Мнемоніка: *ANI ⟨байт⟩*

Символьний запис: (*A*) ← (*A*) ∧ ⟨*байт*⟩

Кількість байтів: 2

Формат команди: КОП, ⟨*байт*⟩

Ознаки результату: змінюються *S*, *Z*, *P*, *C* та *AC*

Команду ***ORA r*** застосовують для виконання логічної операції "АБО" (логічного додавання) над вмістом акумулятора та регістру *r*.

Мнемоніка: *ORA r*

Символьний запис: (*A*) ← (*A*) ∨ (*r*)

Кількість байтів: 1

Формат команди: КОП

Ознаки результату: змінюються *S*, *Z*, *P*, *C* та *AC*

Команду ***ORA M*** застосовують для виконання логічної операції "АБО" (логічного додавання) над вмістом акумулятора та комірки пам'яті, адреса якої вказана в регістровій парі *H*, *L*.

Мнемоніка: *ORA M*

Символьний запис: $(A) \leftarrow (A) \vee ((H)\,(L))$

Кількість байтів: 1

Формат команди: КОП

Ознаки результату: змінюються *S*, *Z*, *P*, *C* та *AC*

Команду ***ORI ⟨байт⟩*** застосовують для виконання логічної операції "АБО" (логічного додавання) над вмістом акумулятора та другого байту команди.

Мнемоніка: *ORI ⟨байт⟩*

Символьний запис: $(A) \leftarrow (A) \vee \langle байт \rangle$

Кількість байтів: 2

Формат команди: КОП, *⟨байт⟩*

Ознаки результату: змінюються *S*, *Z*, *P*, *C* та *AC*

Команду ***XRA r*** застосовують для виконання логічної операції "ВИКЛ. АБО" (нерівнозначності) над вмістом акумулятора та регістру *r*.

Мнемоніка: *XRA r*

Символьний запис: $(A) \leftarrow (A) \oplus (r)$

Кількість байтів: 1

Формат команди: КОП

Ознаки результату: змінюються *S*, *Z*, *P*, *C* та *AC*

Команду ***XRA M*** застосовують для виконання логічної операції "ВИКЛ. АБО" (нерівнозначності) над вмістом акумулятора та комірки пам'яті, адреса якої вказана в регістровій парі *H*, *L*.

Мнемоніка: *XRA M*

Символьний запис: $(A) \leftarrow (A) \oplus ((H)(L))$

Кількість байтів: 1

Формат команди: КОП

Ознаки результату: змінюються *S*, *Z*, *P*, *C* та *AC*

Команду ***XRI ⟨байт⟩*** застосовують для виконання логічної операції "ВИКЛ. АБО" (нерівнозначності) над вмістом акумулятора та другого байту команди.

Мнемоніка: *XRI ⟨байт⟩*

Символьний запис: $(A) \leftarrow (A) \oplus \langle байт \rangle$

Кількість байтів: 2

Формат команди: КОП, ⟨*байт*⟩

Ознаки результату: змінюються *S*, *Z*, *P*, *C* та *AC*

3.3. Виконання спеціальних операцій

При виконанні лабораторних робіт на мікроЕОМ "Микролаб КР580ИК80 907" найчастіше використовують такі спеціальні операції:

– порівняння;

– зсув;

– інвертування вмісту акумулятора;

– запис "1" до розряду *C* регістру ознак (спеціальна);

– інвертування розряду *C* регістру ознак (спеціальна).

4.3.1. Порівняння

Як і більшість попередніх операцій, порівняння вмісту акумулятора може здійснюватись з вмістом регістру, комірки пам'яті або числа, яке задане в другому байті команди.

В процесі порівняння відбувається віднімання порівнюваного числа від вмісту акумулятора. Але результат даного віднімання в акумулятор не записується, тобто вміст акумулятора після виконання даної операції зберігається. Змін зазнають лише прапорці *Z* та *C* регістру ознак.

Команду ***CMP r*** використовують для порівняння вмісту акумулятора і вмісту регістру *r*. При цьому $Z = 1$ при $(A) = (r)$ та $C = 1$ при $(A) < (r)$.

Мнемоніка: *CMP r*

Символьний запис: (*A*) – (*r*)

Кількість байтів: 1

Формат команди: КОП

Ознаки результату: змінюються *Z* та *C*

Команду ***CMP M*** використовують для порівняння вмісту акумулятора і вмісту регістру комірки пам'яті, адреса якої вказана в регістровій парі *H*, *L*. При цьому $Z = 1$ при $(A) = ((H)(L))$ та $C = 1$ при $(A) < ((H)(L))$.

Мнемоніка: *CMP M*

Символьний запис: (*A*) – ((*H*)(*L*))

Кількість байтів: 1

Формат команди: КОП

Ознаки результату: змінюються *Z* та *C*

Команду ***CPI*** ⟨*байт*⟩ використовують для порівняння вмісту акумулятора і числа, заданого в другому байті команди. При цьому $Z = 1$ при (*A*) = ⟨*байт*⟩ та $C = 1$ при (*A*) < ⟨*байт*⟩.

Мнемоніка: *CMP M*

Символьний запис: (*A*) – ⟨*байт*⟩

Кількість байтів: 2

Формат команди: КОП, ⟨*байт*⟩

Ознаки результату: змінюються *Z* та *C*

4.3.2. Зсув

При складанні програм найчастіше виникає потреба у зсуві через перенос. Вміст акумулятора зсувається ліворуч або праворуч на одну позицію через прапорець *C* переносу. З цього випливає, що ці команди зсуву змінюють тільки значення прапорця *C* переносу.

При застосуванні команди ***RAL*** відбувається зсув вмісту акумулятора (в двійковому еквіваленті) на одну позицію ліворуч. Молодший біт вмісту акумулятора встановлюється рівним прапорцю переносу, а біт прапорця переносу стає рівним величині старшого біта вмісту акумулятора.

Мнемоніка: *RAL*

Символьний запис: $A_{n+1} \leftarrow A_n$, $C \leftarrow A_7$ та $A_0 \leftarrow C$

Кількість байтів: 1

Формат команди: КОП

Ознаки результату: змінюються *C*

При застосуванні команди ***RAR*** відбувається зсув вмісту акумулятора (в двійковому еквіваленті) на одну позицію праворуч. Старший біт вмісту акумулятора встановлюється рівним прапорцю переносу, а біт прапорця переносу стає рівним величині молодшому біта вмісту акумулятора.

Мнемоніка: *RAR*

Символьний запис: $A_n \leftarrow A_{n+1}$, $C \leftarrow A_0$ та $A_7 \leftarrow C$

Кількість байтів: 1

Формат команди: КОП

Ознаки результату: змінюються *C*

4.3.3. Інвертування акумулятора

Інвертувати, тобто отримати зворотний код двійкового еквіваленту числа, записаного в акумулятор, можна за допомогою команди ***CMA***. При цьому в двійковому еквіваленті вмісту акумулятора відбувається заміна кожної "1" на "0" і навпаки.

Мнемоніка: *CMA*

Символьний запис: $(A) \leftarrow (\overline{A})$

Кількість байтів: 1

Формат команди: КОП

Ознаки результату: не змінюються

4.3.4. Маніпуляції з прапорцем переносу

Іноді при виконанні команд потрібно або чітко визначити значення прапорця *C* переносу (встановити рівним 0 або 1), або змінити будь-яке (часто невідоме) його значення на протилежне. При цьому слід зауважити, що спеціальної команди, яка б обнулювала значення прапорця *C* переносу, не існує. Тому при програмуванні користуються тільки двома наступними командами.

Команду ***STC*** застосовують для запису в розряд *C* регістру ознак логічної "1".

Мнемоніка: *STC*

Символьний запис: $C \leftarrow 1$

Кількість байтів: 1

Формат команди: КОП

Ознаки результату: змінюється *C*

4.4. Введення та виведення інформації

Дія, під час якої відбувається обмін інформацією між ядром мікроЕОМ "Микролаб КР580ИК80 907" (центральний процесор та основна пам'ять), і периферійними пристроями, називається введенням-виведенням (ВВ).

Підключення пристрою ВВ до шин мікроЕОМ здійснюється через інтерфейсні схеми (адаптери), центральна частина яких являє собою групу регістрів, значення яких доступно процесору під час операцій введення і виведення даних.

Структура програмованого паралельного адаптера – ППА (його роль виконує мікросхема КР580ВВ55) показана на рис. 4.1. Це є універсальна програмована велика інтегральна схема, яка забезпечує обмін інформацією в паралельному форматі.

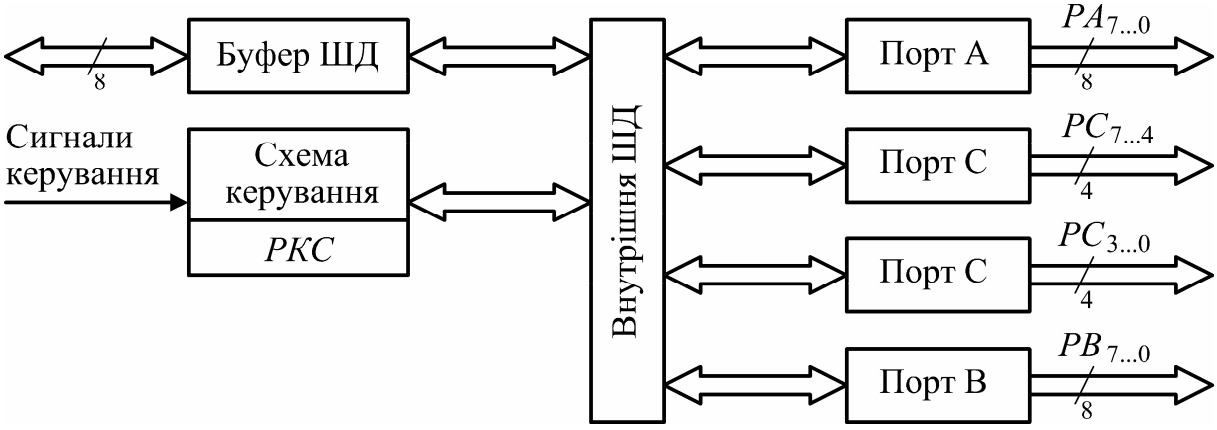


Рис. 4.1. Структура програмованого паралельного адаптеру

Підключення зовнішніх пристроїв здійснюється через три двонапрямлені 8-ми розрядні порти *A*, *B* і *C*. Порти *A* і *B* не розділені, а лінії порту *C* розділені на дві 4-и розрядні групи, які можна використовувати як два незалежні порти. Залежно від режиму роботи адаптера порти використовуються по-різному

4.4.1. Програмування інтерфейсу

Для завдання режиму роботи адаптера необхідно визначити керуюче слово і записати в регістр керуючого слова. Запис керуючого слова в РКС настроює ППА на один із режимів роботи:

– режим 0 – просте введення-виведення;

– режим 1 – введення-виведення з стробуванням (квітуванням);

– режим 2 – двонаправлений канал.

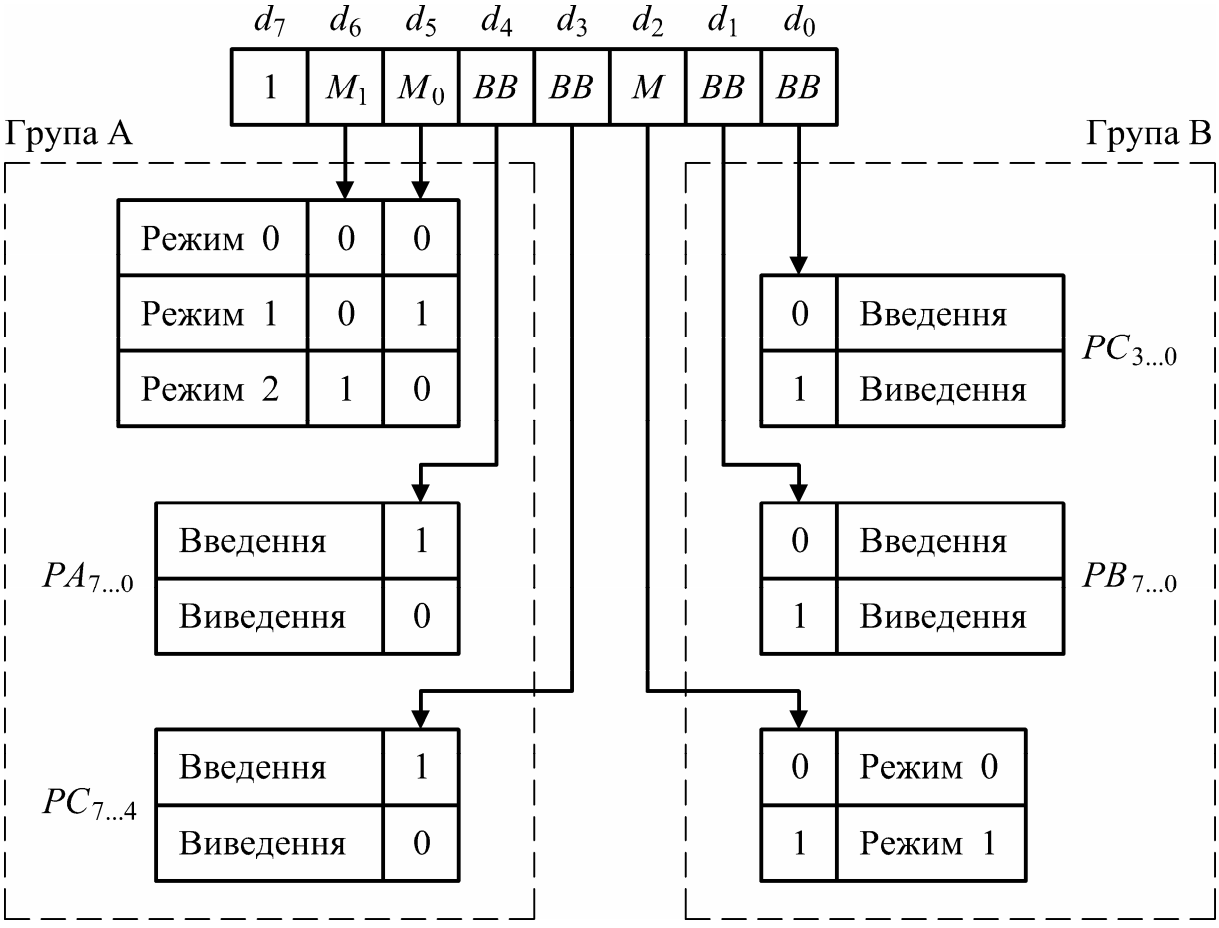


Рис. 4.2. Формат керуючого слова

У режимі 0 забезпечується просте введення-виведення інформації через кожний з трьох каналів. Сигнали керування (квітування) про встановлення зв'язку з периферійним пристроєм не потрібні. Цей режим використ-

товують при організації ВВ із повільно діючими периферійними пристроями.

Режим 1 призначений для однонаправлених передач даних, ініційованих перериваннями. Через порти *A* і *B* відбувається обмін даними, а лінії порту *C* використовуються для прийому й видачі сигналів керування.

Режим 2 забезпечує двонаправлений обмін інформацією між мікропроцесором і периферійними пристроями через порт *A*. Порт *C* використовується для передачі керуючої інформації.

Формат керуючого слова показано на рис. 4.2 і визначається він за умовою $d_7 = 1$ для заданого режиму роботи.

Приклад завдання керуючого слова. Режим 0: групу А налаштувати на передачу (виведення), порт В теж на передачу, а частину порту С ($PC_{3...0}$) на прийом (введення) інформації. Для визначення керуючого слова скористаємося рис. 4.1. Порти виділені в 2 групи: група А містить у собі порт А і половину порту С ($PC_{7...4}$), а в групу В входять порт В та друга половина порту С ($PC_{3...0}$). Для даного прикладу визначають значення розрядів керуючого слова: $d_7 = 1$, $d_6 = 0$, $d_5 = 0$, $d_4 = 0$, $d_3=0$, $d_2 = 0$, $d_1 = 0$, $d_0 = 0$. Таким чином, код слова визначення режиму $1000\ 0001_2=81_{16}$. Цей код необхідно переслати в регістр керуючого слова, що можна здійснити наступною програмою (RUS – символічна адреса регістра керуючого слова).

Адреса	**Машинний код**	**Мнемоніка**	**Коментар**
8000	3E 81	*MVI* A, 81	Записати код слова визначення режиму для ППА в акумулятор
8002	D3 FB	*OUT* RUS	Переслати код режиму ППА до регістра керуючого слова (RUS) із акумулятора

МікроЕОМ "Микролаб КР580ИК80 907" внутрішні пристрої мікросхеми КР580ВВ55 мають такі адреси:

порт А → F8;

порт В → F9;

порт С → FA;

RUS → FB.

Після настройки ППА виведення байта даних на зовнішній пристрій або запис байта даних із зовнішнього пристрою в акумулятор можуть бути здійснені в будь-якому місці прикладної програми.

4.4.2. Виведення інформації на блок світлодіодів

Для виведення інформації на периферійні пристрої (в тому числі й на блок світлодіодів) використовують спеціальну команду ***OUT*** ⟨*порт*⟩, яку необхідно налаштувати для роботи з блоком світлодіодів.

Мнемоніка: *OUT* ⟨*порт*⟩
Символьний запис: ⟨*дані*⟩ ← (*A*)
Кількість байтів: 1
Формат команди: КОП
Ознаки результату: не змінюються

Для виведення інформації на блок світлодіодів мнемонічний запис команди буде наступний: ***OUT*** **СД** (СД – скорочено від "світлодіод"). Оскільки блок світлодіодів підімкнено до порту B, то в якості адреси порту вказується число $F9_{16}$.

Приклад виведення інформації на блок світлодіодів. За завданням необхідно при виконанні деякої умови засвітити чотири середніх світлодіоди. З п. 1.2 вдомо, що блок світлодіодів може відображати число в двійковій системі. Таким чином, на блок світлодіодів необхідно вивести число $00111100_2 = 3C_{16}$. Для цього треба скористатися такими командами (див. фрагмент програми):

Адреса	Маш. код	Мнемокод	Коментар
8020	3E 3C	*MVI* A, 3C	Записати число $00111100_2 = 3C_{16}$ в акумулятор
8022	D3 F9	*OUT* СД	Вивести вміст акумулятора на блок світлодіодів

4.4.3. Зчитування інформації з блоку перемикачів

Для зчитування інформації з периферійних пристроїв (в тому числі й з блоку перемикачів) використовують спеціальну команду ***IN*** ⟨*порт*⟩, яку необхідно налаштувати на роботу з блоком перемикачів.

Мнемоніка: *IN* ⟨*порт*⟩
Символьний запис: (*A*) ← ⟨*дані*⟩

Кількість байтів: 1

Формат команди: КОП

Ознаки результату: не змінюються

Для зчитування інформації з блоку перемикачів мнемонічний запис команди буде наступний: ***IN* БП** (БП – скорочено від "блок перемикачів"). Оскільки блок перемикачів підімкнено до порту С ($PC_{3...0}$), то в якості адреси порту вказується число FA_{16}.

Приклад зчитування інформації з блоку перемикачів. За завданням необхідно інвертувати число, введене з перемикачів. Для цього треба скористатися такими командами (див. фрагмент програми):

Адреса	Маш. код	Мнемокод	Коментар
8050	DB FA	*IN* БП	Зчитування числа з блоку перемикачів
8052	2F	*CMA*	Інвертувати вміст акумулятора

4.5. Типові процедури

При складанні програм в межах виконання лабораторних робіт часто використовуються деякі типові процедури, які являють собою стійку послідовність команд. До таких процедур відносяться:

– отримання додаткового коду числа;

– ділення вмісту акумулятора на 2;

– множення вмісту акумулятора на деяке ціле число.

3.4.1. Отримання додаткового коду числа

Додатковий код числа отримують шляхом арифметичного додавання одиниці до його зворотного коду. Отже, при потребі отримати додатковий код числа, спочатку необхідно отримати його зворотний код.

Для цього може бути використані два набори послідовно виконаних команд:

1) команди ***CMA* → *ADI* 01** (вміст акумулятора інвертується, після чого до нього додається число 01_{16});

2) команди ***CMA → INR A*** (вміст акумулятора інвертується, після чого він збільшується на одиницю).

При програмуванні перевагу слід надавати другому набору команд, оскільки він займає 1 + 1 = 2 байти, тоді як другий набір команд займає 1 + 2 = 3 байти.

3.5.2. Ділення на 2

Теоретично для цілочисельного ділення вмісту акумулятора на 2 достатньо лише команди ***RAR*** зсуву праворуч.

Однак, це справедливо лише в тому випадку, коли в значення прапорця *C* переносу завідомо дорівнює 0. Але на проміжних етапах виконання програми значення прапорця *C* переносу може бути рівним 1, тоді результат операції буде некоректним.

Тому перед здійсненням операції зсуву необхідно обнулити значення прапорця *C* переносу.

З п. 3.3.4 відомо, що спеціальної команди для обнулення прапорця *C* переносу не існує. Тому користуються послідовністю з команд ***STC → CMC***.

Вся процедура ділення на 2 складається з такої послідовності команд: ***STC → CMC → RAR***. Перша команда встановлює значення прапорця *C* переносу рівним 1, друга – інвертує це значення (тобто, в розряді *C* отримують 0), а третя здійснює зсув праворуч розрядів акумулятора.

4.5.3. Множення на ціле число

В арсеналі команд мікроЕОМ "Микролаб КР580ИК80 907" відсутні програми алгебраїчного множення. Тому при програмуванні їх заміняють певною кількістю операцій арифметичного додавання.

Слід пам'ятати, що при виконанні операцій додавання початкове значення акумулятора втрачається, тому його необхідно попередньо зберегти, наприклад в регістрі *B*.

Наприклад, при необхідності помножити вміст акумулятора на 3 виконують таку послідовність команд (2 варіанти):

1) ***MOV B, A → ADD A → ADD B***;

2) ***MOV B, A → ADD A → ADD B***;

В даному випадку обидві послідовності є рівнозначними.

При множенні вмісту акумулятора на 5 менше місця в пам'яті займатиме така послідовність команд: ***MOV B, A → ADD A → ADD A → ADD B***.

При множенні вмісту акумулятора на число 2, 4, 8 ... 2^n (n – ціле число) потреба в збереженні початкового вмісту акумулятора в регістрі *B* відпадає. Так, множення на 4 виконується такою послідовністю команд: ***ADD A → ADD A***.

Іноді в умові задачі стоїть завдання помножити вміст акумулятора на деяке дробне число, наприклад, на 2,5. В цьому випадку множення на 2,5 замінюють послідовними процедурами множення на 5 та ділення на 2.

4.5.5. Маскування

В багатьох випадках при складанні програм є необхідність перевірити або змінити стан одного чи декількох розрядів двійкового числа. Така процедура здійснюється з використанням відповідних констант для фільтрування "сторонніх" розрядів числа і виділення або зміни тільки тих розрядів, значення яких нас цікавлять.

Таку константу звичайно називають маскою, а сам процес – операцією маскування. Операція маскування зазвичай здійснюється за допомогою команд безпосередніх логічних операцій:

1) логічного множення над числом в акумуляторі і маскою (числом, яке вказано у 2-му байті команди ***ANI ⟨байт⟩***). Значення розрядів результату дорівнюють "0", якщо у відповідному розряді маски буде записаний "0", і зберігають значення розряду числа в акумуляторі, якщо відповідний розряд маски дорівнює "1".

2) логічного додавання над числом в акумуляторі і маскою (числом, яке вказано у 2-му байті команди ***ORI ⟨байт⟩***). Значення розряду результату буде дорівнювати "1", якщо в відповідному розряді маски записана "1", і не змінюється, якщо в цьому розряді маски записаний "0".

3) логічного додавання "по модулю 2" над числом в акумуляторі і маскою (числом, яке вказано у 2-му байті команди ***XRI ⟨байт⟩***). Дія команди інвертує значення розряду числа, якщо у відповідному розряді маски буде записана "1", і не змінює значення розряду числа, якщо в цьому розряді маски записаний "0".

Виконання логічних операцій з метою маскування можливо також із умістом регістрів загального призначення і акумулятора.

4.5.6. Зупинка програми

Для зупинки програми використовується спеціальна команда ***HLT***.

Мнемоніка: *HLT*
Символьний запис: *Кінець*
Кількість байтів: 1
Формат команди: КОП
Ознаки результату: не змінюються

Наявність конкретизованого завершення програми є обов'язковою, оскільки, не зустрівши команди зупинки, програма може працювати некоректно.

4.6. Організація переходів

Алгоритми програм для мікроЕОМ "Микролаб КР580ИК80 907" можуть мати не тільки лінійний характер, але і мати розгалуження, наприклад, для переходу до підпрограм або повернення до деякої точки основної програми.

Команди керування програмою передають керування з однієї зони пам'яті в іншу. Змінюючи стан лічильника команд *PC*, вони змінюють звичайну послідовність виконання операцій. Ці команди поділяються на команди безумовних (вони виконуються завжди) і умовних (виконуються за певних умов) переходів. Реалізація переходів в програмах, які розгалужуються, а деякі фрагменти циклічно повторюються, здійснюється на основі аналізу ознак результатів виконання логічних і арифметичних команд. Команди переходів мають трибайтний формат: 1-й байт містить КОП, 2-й – молодший байт адреси комірки пам'яті, а 3-й – старший байт адреси.

Отже, переходи, які зустрічаються при програмуванні, бувають двох типів:

– безумовні;

– умовні.

4.6.1. Безумовний перехід

Безумовний перехід призначений для передачі керування команді, записаній в комірку пам'яті, адреса якої зазначена у другому й третьому байтах команди. Для виконання безумовного переходу використовується команда ***JMP ⟨адреса⟩***.

Мнемоніка: *JMP* ⟨*адреса*⟩.

Символьний запис: (*PC*) ← ⟨*байт* 3⟩⟨*байт* 2⟩

Кількість байтів: 3

Формат команди: КОП, ⟨*байт* 3⟩, ⟨*байт* 2⟩

Ознаки результату: не змінюються

4.6.2. Умовні переходи

Команди умовних переходів виконуються за таким алгоритмом. Перевіряються прапорці регістра ознак результатів. При виконанні умови, що перевіряється, здійснюється перехід за адресою комірки пам'яті, яка вказана у другому і третьому байтах команди. Якщо умова не підтверджується, то виконується наступна команда основної програми. Таким чином, чотири пари команд умовних переходів забезпечують передачу керування за значеннями ознак результатів.

По одній з команд пари відбувається перехід при значенні ознаки "1" і не відбувається при значенні "0", а по другій команді пари – навпаки.

Команди для організації умовних переходів мають спільний формат ***J** ⟨адреса⟩***, де ** – позначення умови, що перевіряється (табл. 3.1).

Мнемоніка: *J** ⟨адреса⟩*.

Символьний запис: (*PC*) ← ⟨*байт* 3⟩⟨*байт* 2⟩

Кількість байтів: 3

Формат команди: КОП, ⟨*байт* 3⟩, ⟨*байт* 2⟩

Ознаки результату: не змінюються

Таблиця 3.1

Прапорець	**Умова**		**Стан прапорця**	**Мнемоніка команди**
	Опис	**Познач.**		
Z	результат не дорівнює нулю	*NZ*	*Z* = 0	*JNZ ⟨адреса⟩*
	результат дорівнює нулю	*Z*	*Z* = 1	*JZ ⟨адреса⟩*
C	перенос присутній	*NC*	*C* = 0	*JNC ⟨адреса⟩*
	перенос відсутній	*C*	*C* = 1	*JC ⟨адреса⟩*
P	результат містить непарну к-сть "1"	*PO*	*P* = 0	*JPO ⟨адреса⟩*

Прапорець	**Умова**		**Стан прапорця**	**Мнемоніка команди**
	Опис	**Познач.**		
	результат містить парну к-сть "1"	*PE*	*P* = 1	*JPE* ⟨*адреса*⟩
S	результат від'ємний	*P*	*S* = 0	*JP* ⟨*адреса*⟩
	результат додатний	*M*	*S* = 1	*JM* ⟨*адреса*⟩

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. *Погорелый С.Д., Слободянюк Т.Ф.* Программное обеспечение микропроцессорных систем: Справочник. – К.: Техніка, 1989.

2. *Балашов Е.П., Григорьев В.А., Петров Г.А.* Микро- и миниЭВМ. – Л.: Энергоатомиздат, 1984.

3. *Гилмор Ч.* Введение в микропроцессорную технику. – М.: Мир, 1984.

4. *Зубчук В.И.* и др. Справочник по цифровой схемотехнике. – К.: Техника, 1990.

5. *Самофалов К.Г., Викторов О.В.* Микропроцессоры. – К.: Техніка, 1989.

6. *Токхайм Р.* Микропроцессоры. – Л.: Энергоатомиздат, 1986.