ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КЕРУЮЧИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Кафедра транспортного зв'язку

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з дисциплін

«МІКРОПРОЦЕСОРНА ТЕХНІКА», «ОБЧИСЛЮВАЛЬНА ТЕХНІКА ТА МІКРОПРОЦЕСОРИ»

Харків – 2017

Методичні вказівки розглянуто і рекомендовано до друку на засіданні кафедри транспортного зв'язку 13 березня 2017 р., протокол № 10.

У методичних вказівках розглядаються основи створення програм мовою assembler, а також засоби налагодження такого програмного забезпечення. Вивчення цих питань ведеться на прикладі мови програмування мікропроцесора КР580ВМ80А (Intel 8080) та програми-емулятора мікропроцесорної системи на базі однокристального КР580ВМ80.

Рекомендуються для студентів з напрямів підготовки 151 «Автоматизація та комп'ютерно-інтегровані технології», 172 «Телекомунікації та радіотехніка» й 273 «Залізничний транспорт» усіх форм навчання та слухачів ІППК.

Укладачі:

доценти І. В. Ковтун,
Н. А. Корольова

Рецензент

доц. Л. А. Клименко

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт
з дисциплін

«МІКРОПРОЦЕСОРНА ТЕХНІКА»,
«ОБЧИСЛЮВАЛЬНА ТЕХНІКА ТА МІКРОПРОЦЕСОРИ»

Відповідальний за випуск Корольова Н. А.

Редактор Еткало О. О.

Підписано до друку 19.04.17 р.

Формат паперу 60х84 1/16. Папір писальний.

Умовн.-друк.арк. 2,0. Тираж 50. Замовлення №

Видавець та виготовлювач Українська державна академія залізничного транспорту,
61050, Харків-50, майдан Фейєрбаха, 7.
Свідоцтво суб'єкта видавничої справи ДК № 2874 від 12.06.2007 р.

ЗМІСТ

ВСТУП

Лабораторні роботи з дисциплін «Мікропроцесорна техніка», «Обчислювальна техніка та мікропроцесори» призначені для закріплення студентами знань, одержаних на лекційних і практичних заняттях. Метою цих методичних вказівок є отримання первинних навичок з програмування мікропроцесорних систем на рівні мікропроцесора (мовою assembler), а також ознайомлення з апаратними й програмними засобами налагодження таких систем і відповідного програмного забезпечення. Студент, що приступає до вивчення викладеного в методичних вказівках матеріалу, повинен володіти основами інформатики і цифрових електронних схем.

До виконання лабораторних робіт допускаються студенти, що пройшли інструктаж з техніки безпеки й успішно пройшли контрольне опитування. Звіт з лабораторної роботи кожен студент складає окремо. Захист виконаної роботи відбувається під час наступного заняття. Під час перебування у лабораторії студенти повинні суворо дотримуватися вимог техніки безпеки щодо роботи з комп’ютерною технікою. Інструктаж з техніки безпеки проводить викладач на початку циклу лабораторних занять, про що кожен студент і викладач ставлять підпис у лабораторному журналі.

У кінці кожної роботи наведено контрольні запитання та завдання, відповіді на які дають змогу визначити ступінь готовності студентів до виконання лабораторної роботи та рівень отриманих знань.

ЛАБОРАТОРНА РОБОТА 1

Вивчення будови та принципів роботи восьмирозрядного мікропроцесора КР580ВМ80 (Intel 8080)

Мета роботи

Вивчити основні функціональні вузли восьмирозрядного мікропроцесора, послідовність виконання команд програми, навчитися користуватися програмою-емулятором мікропроцесорної системи на базі КР580ВМ80.

Загальні відомості

1.1 Внутрішня будова мікропроцесора

Мікропроцесор складається з таких основних частин (рисунок 1.1):

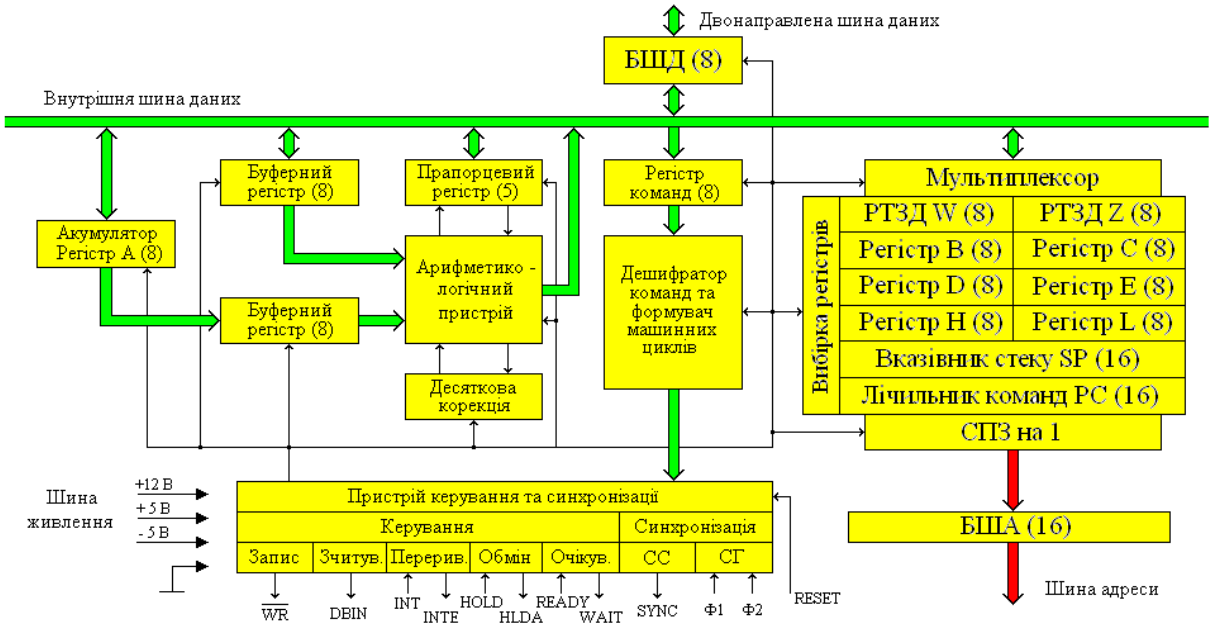

Рисунок 1.1 – Структурна схема мікропроцесора КР580ВМ80

- арифметико-логічний пристрій (АЛП) з регістром А (акумулятор);
- буферні регістри;

- десяткова корекція;
- блок регістрів загального призначення (РЗП) зі схемою вибірки;
- регістр команд з дешифратором команд і формувачем машинних циклів;
- лічильник команд (PC – program counter);
- регістр стекової області (SP – stack pointer – вказівник стека);
- регістр стану (прапорцевий регістр);
- регістри тимчасового зберігання даних (РТЗД) W та Z;
- схема приросту та зменшення (СПЗ);
- пристрій керування та синхронізації.

Буфери шини даних (БШД) та адреси (БША) є додатковими блоками, які обслуговують мікропроцесор, хоча і не входять до його складу.

Число, призначення регістрів, прапорців та команд користувач змінити не може. Він може міняти тільки значення регістрів та використовувати команди в будь-якій, необхідній йому, комбінації.

Регістр – це спеціальний запам’ятовувальний пристрій, який складається з лінійки тригерів і може запам’ятовувати декілька біт інформації одночасно. Більшість регістрів мікропроцесора 8-розрядні, деякі (PC, SP) - 16-розрядні. Усі регістри розбиті на групи і відрізняються різним функціональним призначенням.

Доступними програмісту є такі регістри:

- шість 8-розрядних регістрів, до яких можна звертатися поодинці або парами (регістри B і C, D і E, H і L);
- 8-розрядний акумулятор A;
- 16-розрядні регістри PC (лічильник команд) та SP (вказівник стека).

У деяких випадках можуть бути доступними дані регістра команд (вісім розрядів) та регістр стану (п’ять розрядів). Програмно недоступними є регістри тимчасового зберігання даних W та Z.

Регістри загального призначення. Ці регістри розміром в 1 байт позначаються B, C, D, E, H, L. Вони використовуються для зберігання даних та проміжних результатів обчислень, які виконуються за допомогою арифметико-логічного пристрою. Під час роботи з 16-розрядними числами можна звертатися до пар регістрів (B,C), (D,E), (H,L).

Акумулятор – спеціальний однобайтовий регістр, який позначається А. Під час виконання арифметичних чи логічних операцій він є джерелом одного з операндів та місцем запам'ятовування результату виконання операції. Акумулятор є основною операційною ланкою арифметико-логічного пристрою. Він використовується також як місце зберігання даних та результатів операцій, які виконуються в АЛП.

Регістр команд – однобайтовий регістр, у якому міститься код команди, яка виконується. Цей регістр безпосередньо користувачу недоступний. Це не означає, однак, що не існує команди, яка б могла явно змінити його значення. Після виконання чергової команди в регістр команд автоматично записується код наступної команди із комірки оперативної пам'яті, адреса якої міститься в лічильнику команд (PC).

Лічильник команд (PC) – двобайтовий (16-розрядний) регістр, який часом іще називають програмним лічильником. Цей регістр містить адресу команди, яка повинна виконуватися наступною. Лічильник команд автоматично отримує приріст адреси, який містить у залежності від того, яку за тривалістю команду (одно-, дво- чи трибайтову) мікропроцесор зчитує з пам'яті, вказуючи завжди на перший байт наступної команди. На вміст цього регістра користувач може вплинути тільки за допомогою команд, які змінюють послідовне виконання програми (наприклад команд безумовного переходу), а також за допомогою деяких спеціальних команд.

Вказівник стека (SP) – двобайтовий (16-розрядний) регістр, який містить адресу наступної комірки стека. Стеком називається особливим чином організована область оперативної пам'яті, яку програміст відводить для тимчасового зберігання внутрішніх регістрів мікропроцесора зі спеціальним режимом доступу. Ця область оперативної пам'яті необхідна в тому випадку, коли потрібно припинити виконання поточної послідовності команд та повернутися до неї пізніше, наприклад, для негайного виконання підпрограми або в результаті переривання програми. Дані від мікропроцесора надходять у верхню частину стекової пам'яті, у такому випадку значення вказівника стека зменшується на одиницю для того, щоб завжди вказувати на адресу останньої заповненої комірки стека (дно

стека). Коли дані вибираються зі стека, значення вказівника стека збільшується на одиницю з кожним вибраним байтом. Такі операції зі стеком називаються стековими. З їх допомогою легко організувати багаторівневі (вкладені) переривання та звернення одного рівня підпрограм до другого (вкладені підпрограми).

Регістр стану (прапорцевий регістр) – регістр, який містить п’ять двійкових розрядів, що називаються прапорцями, і які містять спеціальні ознаки результатів деяких операцій. Іноді його називають регістром ознак, або регістром бітів умов. Регістр містить такі прапорці: прапорець нуля (Z – zero), прапорець перенесення (C – carry), прапорець знака (S – sign), прапорець парності (P – parity) та прапорець додаткового перенесення (AC – auxiliary carry). Прапорці завжди встановлюються чи скидаються автоматично після виконання наступної команди, яка впливає на прапорці, у залежності від результату операції. Прапорець вважається встановленим, якщо відповідний розряд регістра набуває значення 1, і скидається, якщо значення розряду 0. Стани прапорців використовують у командах умовного переходу. Результати виконання арифметичних і логічних операцій над вмістом акумулятора, регістрів загального призначення та комірок пам’яті впливають на прапорці таким чином:

Прапорець нуля встановлюється в 1, якщо в результаті виконання якої-небудь команди отримано нульовий результат (усі біти задіяного регістра чи комірки пам’яті встановлено в 0) і скидається в 0 у випадку ненульового результату.

Прапорець перенесення встановлюється в 1, якщо в результаті операцій додавання та зсуву з’являється одиниця перенесення зі старшого розряду байта даних, а також якщо виконується позика зі старшого розряду після виконання операцій віднімання чи порівняння. В іншому випадку прапорець скидається в 0.

Прапорець знака встановлюється в 1, якщо в результаті виконання операцій з’являється одиниця в старшому розряді байта даних (вказує на від’ємний результат) і скидається в 0 у випадку нульового значення старшого розряду (вказує на додатний результат).

Прапорець парності встановлюється в 1, якщо після виконання операцій сума одиниць у байті даних парна (значення суми по модулю 2 дорівнює 0) і скидається в 0, якщо кількість одиниць непарна.

Прапорець додаткового перенесення встановлюється в 1, якщо в результаті виконання команди з’являється одиниця перенесення з третього розряду байта даних у четвертий і скидається в 0, якщо такого перенесення нема. Прапорець додаткового перенесення використовується в багатьох схемах обчислень, однак він особливо необхідний для додавання чисел у двійково-десятковій формі.

1.2 Виконання команди мікропроцесором

Мікропроцесор містить 8-розрядну шину даних і 16-розрядну шину адреси. Шина даних зв'язана з шістьма 8-розрядними регістрами A (акумулятор), B, C, D, E, H, L і 16-розрядними регістрами вказівника стека SP, регістра адреси IP, 8-розрядними регістрами тимчасового зберігання W, Z і регістром прапорців F. Адреса в IP формується:

- коли IP під'єднується до лічильника команд (PC), вміст якого збільшується на 1 після зчитування кожного байта команди;
- коли адреса вводиться послідовно по одному байту і спочатку розміщується в програмно недоступних регістрах тимчасового зберігання (W,Z), а потім передається в регістр адреси PA.

З цієї шини дані можуть передаватися у 8-розрядний регістр коду операції і 16-розрядний програмний лічильник PC. На шину адреси можуть передаватися сигнали адреси з програмного лічильника і регістрів H, L (H-регістр старших розрядів адреси, L-регістр молодших розрядів адреси). Усі дані і команди, що циркулюють по шинах і надходять з пристроїв оперативної пам'яті і введення-виведення, подано у двійковій формі. Але для зручності будемо записувати їх в шістнадцятковій формі. Це відповідає і клавіатурі вводу. Тоді 8-розрядне двійкове число може бути подано у вигляді 2-розрядного шістнадцяткового числа.

Наприклад, двійкове число *11000101* будемо записувати як *C5.* Відповідність між 4-розрядними двійковими і шістнадцятковими числами показана в таблиці 1.1.

Відповідно до таблиці 1.1 шістнадцятковому числу С5 відповідає десяткове число 12*16+5=197. Але навіть у такому вигляді команди запам'ятати дуже важко, тому частіше їх записують скороченими англійськими словами. Такий запис складає мову низького рівня assembler. Переклад з assembler у двійкові коди може виконуватися вручну (за таблицею) або спеціальним транслятором, який є в більшості ЕОМ. Використання мови assembler дає змогу значно зменшити залежність програми від типу мікропроцесора, так як при переході на інший мікропроцесор змінюється лише таблиця відповідності.

Таблиця 1.1 – Відповідність між 4-розрядними двійковими і шістнадцятковими числами

Двійкове число	Десяткове число	Шістнад-цяткове число	Двійкове число	Десяткове число	Шістнад-цяткове число
0000	0	0	1000	8	8
0001	1	1	1001	9	9
0010	2	2	1010	10	A
0011	3	3	1011	11	B
0100	4	4	1100	12	C
0101	5	5	1101	13	D
0110	6	6	1110	14	E
0111	7	7	1111	15	F

Інформація, яка може одночасно передаватися, відповідає одному байту (вісім розрядів) і називається *машинним словом* або *словом*. Можливі такі передавання даних:

- пересилання байта даних від пристрою введення;
- пересилання байта даних до пристрою виведення;
- зчитування байта даних з пам’яті або запис у пам'ять;
- генерування в шину спеціального байта, який називається керівним словом і призначений для встановлення правильного схемного з’єднання.

Робота мікропроцесора базується на принципі мікропрограмного керування. Це означає, що кожна команда реалізується як деяка послідовність мікрокоманд або мікрооперацій, які приводять до необхідного результату. Команда, та фактично її 8-розрядний код зчитується з пам’яті і надходить у регістр команд, де і зберігається до кінця її виконання. Відповідно до результату дешифрування коду команди відбувається формування послідовності мікрокоманд (мікропрограма), процес виконання якої визначає всі наступні операції, необхідні для виконання зчитаної команди. Виконання окремих мікрооперацій синхронізується відповідно до сигналів Ф1 та Ф2 тактового генератора. Найважливішим поняттям усього процесу виконання команд є поняття машинного циклу.

Процес виконання кожної команди можна розбити на ряд основних операцій. Час, необхідний на виконання звернень до пам’яті чи пристроїв введення-виведення, складає *машинний цикл.* Виконання команди займає стільки машинних циклів, скільки необхідно звернень до пам’яті чи пристроїв введення-виведення для її виконання. Машинний цикл у свою чергу складається з машинних тактів (один машинний такт – один період тактових імпульсів синхрогенератора).

Кожна команда в залежності від її виду може займати від одного до п’яти машинних циклів. Мікропроцесор КР580ВМ80 має 10 типів машинних циклів:

- вибірка команди;
- зчитування слова із запам'ятовувального пристрою;
- запис слова в запам'ятовувальний пристрій;
- зчитування слова зі стека;
- запис слова в стек;
- зчитування слова з пристрою введення/виведення;
- запис слова в пристрій введення/виведення;
- підтвердження переривання;
- підтвердження зупинки;
- підтвердження переривання під час зупинки.

Перший цикл завжди є цикл вибірки команди і займає чотири або п’ять тактів. Три наступних цикли завжди виконуються за три такти, а п'ятий – за три або п'ять тактів. З огляду на це завжди можна розрахувати кількість циклів і тактів команди, якщо відомо способи адресації, які вона використовує.

Наприклад: команда MVI M, B2 двобайтова, використовує безпосередню/непряму адресацію. Під час виконання команди дані, які містяться в другому байті програми, завантажуються в комірку пам'яті, адреса якої міститься в регістровій парі HL. Перший цикл – вибірка першого байта з пам'яті (чотири або п’ять тактів), наступний – вибірка другого байта (три такти). Ще один цикл – запис даних у комірку пам'яті (три такти). Отже, команда виконується за три цикли 10 або 11 тактів (насправді 10 тактів).

1.3 Система команд мікропроцесора КР580ВМ80

Система команд мікропроцесора КР580ВМ80 подана 244 кодами операцій (додаток А), які можна підрозділити згідно з декількома ознаками. Найбільш суттєвими ознаками є такі:

- довжина команди або число байтів, які вона займає в пам’яті;
- функціональна ознака або операції, які вона виконує;
- спосіб адресації.

Із 256 можливих кодів команд не використовуються 12 кодових комбінацій, тому число всіх команд: 256-12=244. Усі команди поділяються на три групи відповідно до кількості байтів, які вони займають у пам’яті: однобайтові, двобайтові та трибайтові.

У будь-якій команді перший байт завжди містить код команди, другий і третій байти містять або дані, або адресу комірки пам’яті, яка містить дані.

Мікропроцесор КР580ВМ80 може виконувати 78 базових команд відповідно до функціональної ознаки, а саме: 200 однобайтових, 18 двобайтових та 26 трибайтових команд відповідно до довжини, яку дана команда займає в пам’яті ОЗП.

Для всіх можливих пересилань даних з регістрів у регістри, або з пам’яті в регістри та навпаки, розрізняють регістри – джерела даних, які позначаються символом S (source – джерело), та регістри–приймачі даних, які позначаються символом D (destination – місце призначення). У регістрових парах (B, C), (D, E) та (H, L) старшими є перші регістри пар. Коди регістрів загального призначення, пар регістрів та прапорців строго фіксовані (таблиця 1.2).

Таблиця 1.2 – Коди регістрів та прапорців мікропроцесора КР580ВМ80

Регістр	Код	Пара регістрів	Код	Мнемонічне позначення	Код
A	111	B (B, C)	00	NZ (Z=0)	000
B	000	D (D, E)	01	Z (Z=1)	001
C	001	H (H, L)	10	NC (CY=0)	010
D	010	SP	11	C (CY=1)	011
E	011			PO (P=0)	100
H	100			PE (P=1)	101
L	101			P (S=0)	110
M (пам'ять)	110			M (S=1)	111

Мікропроцесор КР580ВМ80 використовує доволі простий формат команд (рисунок 1.2).

Код операції завжди розміщений у першому байті програми. Кожен біт відмічено х. У першому біті може міститися інформація про місцезнаходження операндів dst (Destination) адресат – приймач, src (Source) адресат - джерело або ціле число n = 0..7.

Другий, а якщо необхідно, і третій байти відводяться під безпосередні дані, адресу порту чи комірки пам'яті.

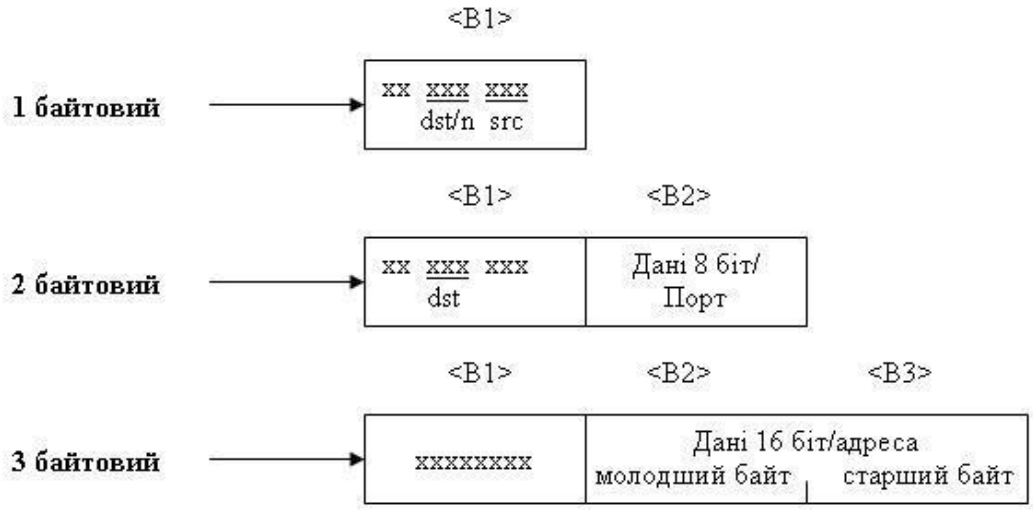

Рисунок 1.2 – Формат команд мікропроцесора КР580ВМ80

Для мікропроцесора КР580ВМ80 буває чотири можливих способи адресації: безпосередня, пряма, регістрова та непряма.

Безпосередня адресація (рисунок 1.3) є найбільш економічним способом зберігання та пошуку інформації, оскільки необхідні дані містить сама команда. Ці дані містяться в другому та третьому байтах трибайтової команди та в другому байті двобайтової команди. У випадку трибайтової команди молодші розряди 16-бітового числа містяться в другому байті, а старші – в третьому байті команди.

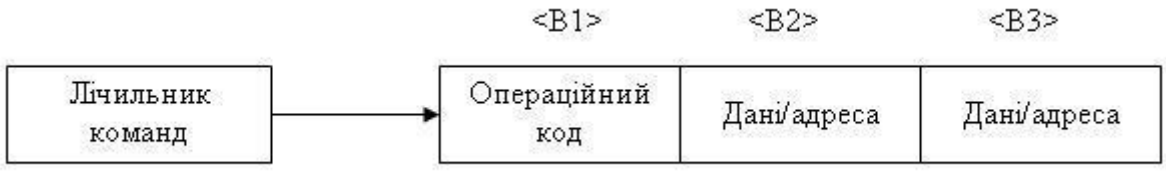

Рисунок 1.3 – Формат команди з безпосередньою адресацією

Менш економічною, але також досить простою є ***пряма адресація*** (рисунок 1.4). В такому випадку в другому і третьому байтах міститься повна 16-бітова адреса комірки пам’яті, у якій є дані. Молодшим байтом адреси є другий байт команди, а старшим – третій байт команди.

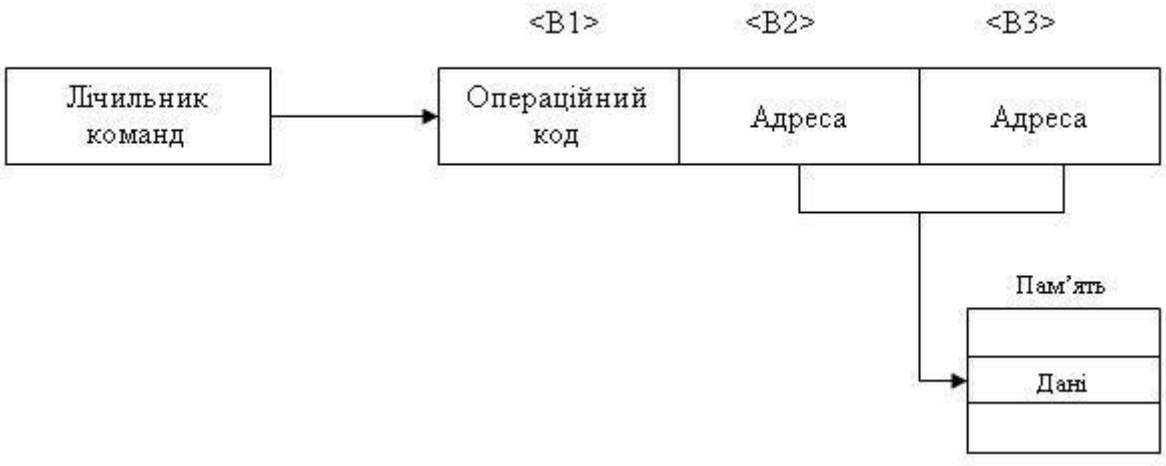

Рисунок 1.4 – Формат команди з прямою адресацією

У випадку ***регістрової адресації*** (рисунок 1.5) код команди вказує на регістр або пару регістрів, у яких містяться дані. Команди, які використовують регістрову адресацію, є однобайтовими. У такому випадку адреси регістрів задаються за допомогою трьох або шести молодших бітів команди.

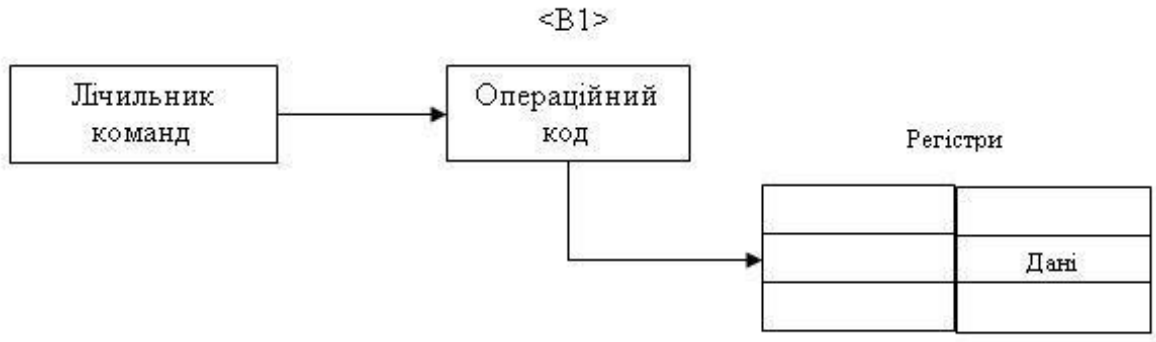

Рисунок 1.5 – Формат команди з регістровою адресацією

Непряма адресація (рисунок 1.6) відрізняється від регістрової тільки тим, що в регістровій парі містяться не дані, а повна 16-розрядна адреса комірки пам'яті, яка містить дані. Старший байт адреси записується в першому регістрі пари, а молодший – у другому. Зазвичай вказівником адреси у випадку непрямої адресації є пара регістрів H, тобто регістри (H, L), але інколи використовуються пари (B, C) і (D, E).

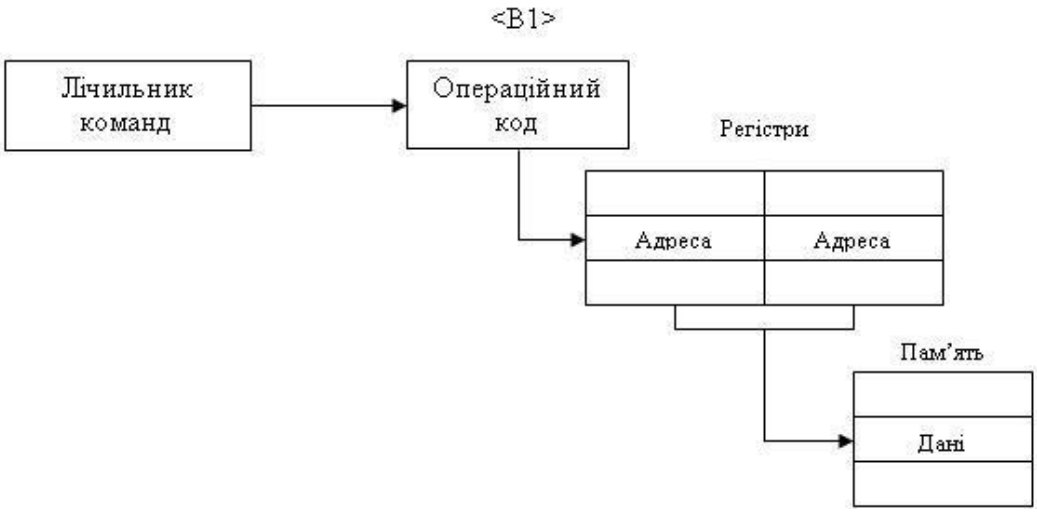

Рисунок 1.6 – Формат команди з непрямою адресацією

Для всіх можливих пересилань даних з регістрів у регістри або з пам’яті в регістри та навпаки розрізняють регістри–джерела даних, які позначаються символом S (source – джерело), та регістри–приймачі даних, які позначаються символом D (destination – місце призначення). У регістрових парах (B, C), (D, E) та (H, L) старшими є перші регістри пар. Коди регістрів загального призначення, пар регістрів та прапорців строго фіксовані.

Усі команди відповідно до функціональної ознаки можуть бути розбиті на п’ять груп:

- група команд пересилання даних;
- арифметичні команди;
- логічні команди;
- команди переходів;
- команди управління і роботи зі стеком.

1.4 Емулятор мікропроцесорної системи на базі КР580ВМ80

Емулятор – це програма, яка відтворює всі процеси, що відбуваються в реальних системах, на екрані монітора. Цей емулятор дає змогу створювати програми мовою assembler, використовуючи систему команд мікропроцесора КР580ВМ80, налагоджувати їх виконання в тактовому, командному та наскрізному режимах, вивчати принципи і порядок виконання команд, отримувати уявлення про організацію зовнішньої та внутрішньої (регістрової) пам’яті та стекової області.

Головне вікно програми має вигляд, поданий на рисунку 1.7.

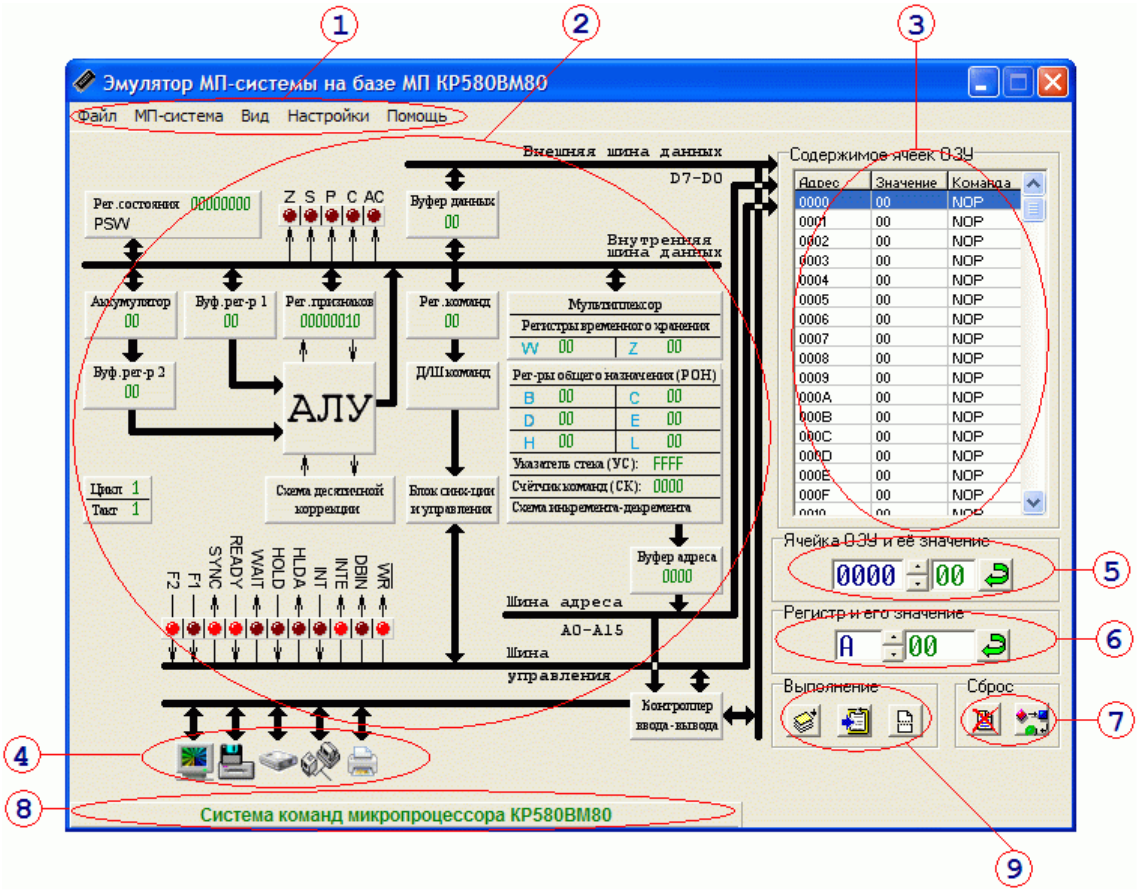

Рисунок 1.7 – Вигляд головного вікна програми емулятора

Зміст головного вікна програми:

1) головне меню програми;

2) структурна схема мікропроцесорної системи;

3) таблиця вмісту ОЗП мікропроцесорної системи;

4) зовнішні периферійні пристрої, які під'єднані до портів МП системи;

5) панель редагування значення вибраної (поточної) комірки ОЗП;

6) панель редагування значення вмісту вибраного регістра загального призначення МП системи;

7) група кнопок «Сброс» для обнуління всіх комірок ОЗП та регістрів загального призначення;

8) панель системи команд мікропроцесора КР580ВМ80;

9) група кнопок «Выполнение» для виконання програми в наскрізному, командному й тактовому режимах.

Структурна схема містить такі елементи:

- регістр слова стану мікропроцесора (PSW) і регістр ознак (прапорців), а також значення у двійковій системі числення та словесній формі;

- буфер даних МП системи, регістр А (акумулятор), буферні регістри, регістр команд, блок регістрів загального призначення, регістри тимчасового зберігання даних, регістри – вказівник стека та лічильник команд, а також їхні значення в шістнадцятковій системі числення;

- блок АЛП та десяткової корекції;

- блок синхронізації та керування;

- дешифратори команд та лічильники машинних мікроциклів;

- індикатори стану і тактування мікропроцесора: F1, F2, SYNC, READY, WAIT, HOLD, HLDA, INT, INTE, DBIN, WR;

- порти системи від 00h до 04h для монітора, дисковода, мережевого адаптера та принтера відповідно;

- шини даних, адреси, керування, внутрішня шина та шина зовнішніх пристроїв.

ОЗП МП системи подано у вигляді блока-таблиці:

- адреса ОЗП – кожна комірка має адресу від 0000h до FFFFh (від 0d до 65535d) – всього 64К комірки;

- значення комірки ОЗП – поточне значення вибраної комірки ОЗП (8 біт). Подано в шістнадцятковій системі числення від 00h до FFh (від 0d до 255d) – всього 256 значень;

- команда мікропроцесора – розшифроване значення комірки ОЗП у вигляді мнемокоду мовою assembler. Деякі комірки можуть містити не команди, а дані, однак їхні значення будуть перекодовані в мнемокод автоматично.

Адреса вибраної комірки автоматично відображається в регістрі–лічильнику команд.

У нижній області ОЗП встановлено виділення коричневим кольором тієї комірки, на яку вказує вказівник стека. Стекова область виділена жовтим кольором.

Зовнішні периферійні пристрої під’єднані до загальної шини контролера введення-виведення. Усього до МП системи під’єднано п’ять віртуальних пристроїв: монітор, який може працювати як у графічному, так і текстовому режимах; накопичувачі на гнучких та твердих магнітних дисках; мережевий адаптер та принтер. Усі пристрої, зокрема монітор, можуть також працювати в режимі реального часу з реальними фізичними периферійними пристроями.

Панелі редагування значення комірок ОЗП або регістрів дають змогу вводити програми та задавати початкові умови, а також редагувати введені або зчитані з файла програми.

Комірки «Сброс» дають змогу обнуляти всі комірки ОЗП або регістри МП системи.

Панель системи команд мікропроцесора КР580ВМ80 подана у вигляді прихованої таблиці 16х16, рядки і стовпчики якої пронумеровані в шістнадцятковій системі числення. Їх послідовна комбінація (рядок – стовпчик) є кодом вибраної команди. Усі команди умовно розбиті на дванадцять груп, об'єднаних за функціональною ознакою, кожна з яких позначена своїм кольором комірки. Панель системи можна активізувати за допомогою клавіші «Space» або наводячи на неї курсор мишки. Панель полегшує програмування емулятора, оскільки вибрані команди можна «перетягувати» в комірки ОЗП за допомогою лівої клавіші мишки, а права клавіша дає змогу отримати повну інформацію про команду.

Група кнопок «Выполнение».

«Виконати такт» - це виконати один такт команди ОЗП, на яку вказує лічильник команд. Якщо команда виконана не повністю, деякі елементи керування стають недоступними для редагування, а ті елементи, які є активними в даному такті, відмічаються червоним кольором;

«Виконати команду» дає змогу виконати програму покомандно і контролювати її виконання та стан регістрів і пам'яті;

«Виконати програму» запускає програму на виконання до тих пір, поки дана кнопка не буде натиснута повторно, або у випадку команди HLT (76h).

Основні принципи роботи з програмою

Початок роботи з програмою полягає в написанні або завантаженні програми мовою assembler в емулятор. Для цього можна скористатися або панеллю системи команд програми, або панеллю редагування значень комірок ОЗП емулятора, або завантажити образ ОЗП з носія.

За необхідності можна заповнити відповідними значеннями регістри загального призначення емулятора.

Для детального вивчення кожного такту конкретної команди можна скористатися кнопкою з тактового виконання команди. Для налаштовування програми використовується кнопка покомандного виконання.

Написану програму мовою assembler можна зберегти у вигляді образу ОЗП та РЗП на будь-який носій, а також за необхідності завантажити з носія в емулятор.

Програма також дає можливість експорту частини ОЗП та/або РЗП емулятора в MS Excel, MS Word та текстовий файл.

2 Порядок виконання експериментів

Запустіть емулятор.

Експеримент 1. *Занесення початкових даних у регістри і пам'ять та виконання простої програми.*

1.1 Задайте в полі 6 емулятора початкові значення регістрів загального призначення: A = 0F; B = F1; C = 12; D = 55; E = 66; H = 00; L = 0E.

1.2 Введіть у полі 5 нижченаведену програму.

Адреса комірки ОЗП	Код команди	Мнемонічний запис команди	Команда мікропроцесора
0000	3C	INR A	A+1 -> A
0001	06	MVI B, d8	дані -> B
0002	1F	RAR	дані 1F
0003	05	DCR B	B-1 -> B
0004	81	ADD C	A+C -> A
0005	82	ADD D	A+D -> A
0006	93	SUB E	A-E -> A
0007	77	MOV M,A	A -> пам'ять (000E)
0008	76	HLT	останов

1.3 Задайте значення регістра PC = 0000. Запустіть програму в полі 9 кнопкою «Выполнить программу». Запишіть значення регістрів загального призначення після виконання програми :

A = ; B = ; C = ; D = ; E = ; H = ; L = ; PC = .

1.4 Запишіть значення комірки пам'яті за адресою 000E: ОЗП(000E) = .

1.5 Порівняйте значення регістрів та комірок пам'яті до та після виконання програми. Зробіть висновок про арифметичні дії над числами, які були занесені в регістри. Порівняйте отриманий практичний результат з розрахунками, зробленими письмово (переведіть початкові дані у двійкову систему числення і виконайте відповідні перетворення).

Експеримент 2. *Вивчення способів адресації мікропроцесора КР580ВМ80.*

2.1 Обнуліть ОЗП та регістри шляхом натискання на кнопку «Сброс ОЗУ» та «Сброс регистров» у полі 7.

2.2 Запустіть виконання програми в потактовому режимі шляхом натискання кнопки «Выполнить такт» у полі 9.

2.3 Результати виконання програми занесіть у таблицю 1.3.

Таблиця 1.3 – Таблиця результатів виконання програми

Адреса комірки ОЗП	Команда	Цикл	Такт	Лічильник команд	Регістр команд	Регістр стану (PSW, тип циклу)	Регістр ознак ZSPCAC	Акумулятор

Безпосередня адресація

Адреса комірки ОЗП	Код команди	Мнемонічний запис команди	Команда мікропроцесора
0000	3E	MVI A, d8	дані наступної комірки ОЗП -> регістр А
0001	AA	XRA D	дані
-	-	-	-

Пряма адресація

Занесіть початкове значення комірки ОЗП (000B)=AA.

Адреса комірки ОЗП	Код команди	Мнемонічний запис команди	Команда мікропроцесора
0000	3A	LDA adr	дані комірки ОЗП -> регістр А
0001	0B	DCX B	молодший байт адреси комірки ОЗП
0002	00	NOP	старший байт адреси комірки ОЗП
-	-	-	-
-	-	-	-
000B	AA	XRA D	дані
-	-	-	-

Регістрова адресація

Занесіть в регістр B значення B=AA.

Адреса комірки ОЗП	Код команди	Мнемонічний запис команди	Команда мікропроцесора
0000	78	MOV A, B	регістр B -> регістр A
-	-	-	-

Непряма адресація

Занесіть у регістри H та L значення H=00, L=0B; у комірку ОЗП (000B)=AA.

Адреса комірки ОЗП	Код команди	Мнемонічний запис команди	Команда мікропроцесора
0000	7E	MOV A, M	дані комірки ОЗП за адресою HL (H-старший байт адреси комірки ОЗП; L-молодший байт адреси комірки ОЗП) -> регістр A
-	-	-	-
-	-	-	-
000B	AA	XRA D	дані
-	-	-	-

Зробіть висновок про обсяг оперативної пам'яті, який займають команди, та час їх виконання для різних способів адресації.

Експеримент 3. *Запис і виконання найпростіших програм*.

3.1 Складіть просту програму для обчислення формули

$$(X - 200b + Y) \rightarrow Z,$$

де X, Y – вміст елементів пам'яті з адресами 0B00h і 0B01h відповідно, де розташовуються перший і другий операнди;

200b - задана константа в десятковій системі числення;

Z - вказує на місце розташування результату, а саме на елемент пам'яті з адресою 0B02h.

Відповідно до формули необхідно завантажити з адреси 0B00h число X, відняти від нього константу 200 і додати число Y, завантажене з комірки пам'яті 0B01h. Результат необхідно зберегти в пам'яті за адресою 0B02h.

Записати за адресами 0B00h і 0B01h числа згідно з варіантом з таблиці 1.4.

Таблиця 1.4 – Варіанти завдань

Адреси комірок	***Значення чисел за варіантами, h***									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0B00h	05	0E	5A	12	B5	B0	25	A0	EE	34
0B01h	06	0F	6A	11	B6	A0	26	99	FF	35

3.2 Занесіть програму в пам'ять мікропроцесорної систем.

Для запису всієї програми (таблиця 1.5) у пам'ять мікропроцесора необхідно перевести отриману програму в машинні коди (додаток А), де всі команди і числа подаються в шістнадцятковій системі числення, при цьому необхідно пам'ятати про розміри кожної з команд, про правила розміщення програм у пам'яті та про машинні коди кожної з команд.

3.3 Здійсніть запуск програми в автоматичному режимі. Візьміть результат виконання програми з пам'яті (елемент пам'яті з адресою 0B02h) і занесіть його у звіт. Порівняйте отриманий практичний результат з розрахунками, зробленими письмово (переведіть початкові дані у двійкову систему числення і виконайте відповідні перетворення).

Таблиця 1.5 – Програма виконання арифметичної програми

Адреса комірки ОЗП	Код команди	Мнемонічний запис команди	Число байтів у команді	Команда мікропроцесора
0800		LDA 0B00h		*Призначення команди LDA:* вміст КП, адреса якої вказана в другому і третьому байтах команди, завантажується в регістр А
0803		SUI C8h		*Призначення команди SUI:* з вмісту А віднімається другий байт команди
0805		LXI H,0B01h		*Призначення команди LXI :* безпосереднє завантаження пари регістрів H, L
0808		ADD M		*Призначення команди ADD:* вміст КП, адреса якої записана в регістрах H, L додається до вмісту регістра А і результат залишається в регістрі А
0809		STA 0B02h		*Призначення команди STA:* вміст А пересилається в КП, адреса якої вказана в другому і третьому байтах команди
080C		HLT		*Призначення команди HLT:* Останов

Експеримент 4. *Дослідження модифікованої програми.*

Замініть у програмі **експерименту 3** команду «SUI d8» на команду згідно з варіантом (таблиця 1.6). Складіть для модифікованої програми таблицю, ідентичну таблиці 1.5, занесіть її у звіт. Для модифікованої програми виконайте дослідження та занесіть отримані дані у звіт.

Примітка - Слід пам'ятати, що початкова команда в таблиці 1.5 з мнемокодом «SUI C8h» займала в пам'яті 2 байти. У зв'язку з цим, якщо модифікована команда містить меншу кількість байтів, то початкову програму необхідно модифікувати (або скорегувати адреси розміщення програми, або скориставшись командою «NOP» - немає операції).

Таблиця 1.6 – Варіанти завдань

Команда, на яку відбувається заміна «SUI d8», за варіантами									
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
ADI 01h	SBI 12h	ACI 33h	INR A	DCR A	ADI 03h	SBI BBh	ACI F3h	INR A	DCR A

Контрольні запитання

1 Назвіть основні складові частини мікропроцесора.

2 Що таке регістр і регістри яких типів входять до складу мікропроцесора?

3 Які регістри є доступними програмісту, а які – ні?

4 Для чого використовуються регістри загального призначення?

5 Призначення регістра А. Дані якої довжини можна завантажувати в акумулятор?

6 Для чого використовується лічильник команд?

7 Що таке стек і яке призначення вказівника стека?

8 Поясніть призначення регістра стану та значення окремих його прапорців.

9 Яким чином мікропроцесор виконує команду?

10 Що таке машинний цикл і чим він відрізняється від машинного такту?

11 Формати команд: однобайтовий, двобайтовий, трибайтовий.

12 Види адресації мікропроцесора КР580ВМ80. Який з видів адресації є найбільш економним з точки зору обсягу оперативної пам'яті, що він займає, з точки зору часу виконання?

13 Назвіть типи команд щодо функціональної ознаки.

14 У чому полягає робота програми-емулятора?

15 Який максимальний обсяг пам'яті може адресувати мікропроцесор КР580ВМ80?

ЛАБОРАТОРНА РОБОТА 2

Команди умовних та безумовних переходів восьмирозрядного мікропроцесора КР580ВМ80 (Intel 8080)

Мета роботи

Вивчити основні типи команд умовних та безумовних переходів, основні способи їх застосування, навчитися створювати програми з їх використанням.

Загальні відомості

Широкі можливості мікропроцесора значною мірою визначаються його здатністю приймати рішення в ході виконання програми відповідно до отриманого результату. Для забезпечення таких можливостей використовуються команди умовних та безумовних переходів, які дають змогу міняти послідовність виконання програм. З їх допомогою в програмах влаштовують цикли та розгалуження, тому вони також ще називаються командами розгалуження.

2.1 Загальні властивості команд переходів

Усі команди поділяються на дві групи: безумовних та умовних переходів. Якщо на виконання команд першої групи не впливають результати попередніх обчислень, то виконання команд другої групи здійснюється саме відповідно до цих результатів.

За винятком команди **PCHL**, яка є однобайтовою, усі інші є трибайтовими. Це означає, що другий та третій байти команди містять повну 16-розрядну адресу деякої комірки пам'яті. У цій комірці пам'яті записана команда програми, яку необхідно виконати наступною у випадку, якщо задовольняється умова переходу. Перехід здійснюється шляхом запису адреси цієї комірки в лічильник команд PC та передачі керування команді, яка в ній записана. Якщо ж умова не задовольняється, то ніякого переходу не відбувається, а мікропроцесор виконує команду, наступну після команди переходу, тобто лічильник команд збільшує своє значення на 3. Оскільки адреса комірки, де міститься код команди, безпосередньо завантажується в **PC**, то вважається, що використовується безпосередня адресація. Єдиною командою переходу, що використовує регістрову адресацію, є однобайтова команда **PCHL**. Необхідно звернути увагу на те, що в другому байті команди міститься молодший байт адреси, а в третьому – старший. Тому, наприклад, команда **JMP** A7, 00 означає перехід за адресою 00A7h.

Слід пам'ятати, що після виконання команди переходу не залишається жодних даних про те, з якої точки програми було здійснено цей перехід, тому всі умовні та безумовні переходи необхідно застосовувати якомога менше, оскільки наявність великої кількості галужень програми значно ускладнює її написання та налагодження.

2.2 Команди безумовного переходу

Під час виконання цих команд ніякі перевірки не здійснюються. У лічильник команд мікропроцесора автоматично завантажується значення другого та третього байту команди, і починається виконання команди за цією адресою. Команди безумовного переходу відрізняються від усіх команд переходів тим, що їх можна розглядати також як команди завантаження 16-розрядного регістра ***PC***.

Команди безумовного переходу використовують у тому випадку, якщо необхідно здійснити перехід на чітко визначену команду програми, а також для зв'язування різних частин програми. Інколи їх використовують під час написання та налагодження програми для переходів з так званих «заготовок» та «заглушок».

За допомогою команд безумовного переходу реалізують оператори типу ***GOTO*** мов високого рівня. Оскільки велика кількість безумовних переходів негативно відображається на надійності та «читабельності» програм, слід обмежувати використання команд такого виду.

У мікропроцесорі КР580ВМ80 передбачено дві команди безумовного переходу:

- 1-ша команда:

1	Безумовний перехід	
2	безпосередня	
3	**JMP B_2, B_3**	
4	**B_3 B_2 → PC**	
5	C3h	303q
6	Команда займає 3 байти пам'яті, виконується за 3 цикли, 10 тактів	
7	Під час виконання команди в молодший байт лічильника команд **PC** записується другий байт команди **B_2**, а в старший байт **PC** – третій байт команди **B_3**. Керування передається команді, яка буде занесена в **PC** (за адресою **B_3 B_2)**	
8	Виконання команди не впливає на регістр стану	

- 2-га команда:

1	Завантаження лічильника команд **PC** (непрямий перехід)	
2	Регістрова	
3	**PCHL**	
4	**HL → PC**	
5	E9h	351q
6	Команда займає 1 байт пам'яті, виконується за 1 цикл, 5 тактів	
7	Під час виконання команди дані, які містяться в регістрі **L**, пересилаються в молодший байт лічильника команд **PC**, а дані, які містяться в регістрі **H**, - у старший байт. Керування передається команді, яка буде занесена в **PC** (за адресою **B_3 B_2)**	
8	Виконання команди не впливає на регістр стану	

2.3 Команди умовних переходів

На виконання всіх команд умовних переходів впливає значення окремих прапорців регістру стану, однак використовуються не всі прапорці. Наприклад, прапорець додаткового перенесення не впливає на жодну із команд переходу. Кожен прапорець може бути встановлений в одне із двох значень, тому всього команд умовного переходу є 8.

Усі команди умовного переходу мають такий формат: якщо умова виконується, то перейти на вказану адресу, якщо ні - виконувати наступну команду програми.

Команди ПЕРЕХІД, ЯКЩО НУЛЬ та ПЕРЕХІД, ЯКЩО НЕ НУЛЬ перевіряють значення прапорця нуля. Якщо його значення дорівнює 1, то це свідчить, що в результаті попередньої дії було набуто нульовий результат. Тому в результаті виконання команди ПЕРЕХІД, ЯКЩО НУЛЬ буде здійснено перехід за адресою, яка міститься в другому та третьому байтах програми, а при нульовому значенні прапорця такий перехід здійснено не буде. Якщо прапорець встановлено в 0, то навпаки, команда ПЕРЕХІД, ЯКЩО НЕ НУЛЬ буде виконуватися з переходом, а команда ПЕРЕХІД, ЯКЩО НУЛЬ – без переходу.

Команду ПЕРЕХІД, ЯКЩО НУЛЬ часом називають командою ПЕРЕХІД, ЯКЩО ДОРІВНЮЄ. Таку назву команда отримала тому, що її часто застосовують одразу ж після команди ПОРІВНЯННЯ. Якщо два числа, що порівнюються, дорівнюють один одному, то прапорець нуля встановлюється в 1. У такому випадку в програмі буде здійснено перехід. Аналогічно, команду ПЕРЕХІД, ЯКЩО НЕ НУЛЬ можна вважати командою ПЕРЕХІД, ЯКЩО НЕ ДОРІВНЮЄ.

1	Перехід, якщо результат нуль	
2	безпосередня	
3	**JZ B_2, B_3**	
4	**B_3 B_2 → PC**	
5	CAh	312q
6	Команда займає 3 байти пам'яті, виконується за 3 цикли, 10 тактів	
7	Якщо прапорець нуля встановлено в 1, то під час виконання команди в молодший байт лічильника команд **PC** записується другий байт команди **B_2**, а в старший байт **PC** – третій байт команди **B_3**. Керування передається команді, яка буде занесена в **PC** (за адресою **B_3 B_2**). Якщо прапорець нуля встановлено в 0, то керування передається наступній команді	
8	Виконання команди не впливає на регістр стану	

1	Перехід, якщо результат не нуль	
2	безпосередня	
3	**JNZ B_2, B_3**	
4	**B_3 B_2 → PC**	
5	C2h	302q
6	Команда займає 3 байти пам'яті, виконується за 3 цикли, 10 тактів	
7	Якщо прапорець нуля встановлено в 0, то під час виконання команди в молодший байт лічильника команд **PC** записується другий байт команди **B_2**, а в старший байт **PC** – третій байт команди **B_3**. Керування передається команді, яка буде занесена в **PC** (за адресою **B_3 B_2**). Якщо прапорець нуля встановлено в 1, то керування передається наступній команді	
8	Виконання команди не впливає на регістр стану	

Команди ПЕРЕХІД, ЯКЩО ПЕРЕНЕСЕННЯ та ПЕРЕХІД, ЯКЩО НЕМА ПЕРЕНЕСЕННЯ перевіряють значення прапорця перенесення. Якщо його значення рівне 1, то це свідчить, що в результаті попередньої дії з’явився біт перенесення із старшого сьомого розряду в наступний розряд (або біт запозичення в сьомий розряд). Тому в результаті виконання команди ПЕРЕХІД, ЯКЩО ПЕРЕНЕСЕННЯ буде здійснено перехід за адресою, яка міститься в другому та третьому байтах програми, а при нульовому значенні прапорця такий перехід здійснено не буде. Якщо прапорець установлено в 0, то навпаки, команда ПЕРЕХІД, ЯКЩО НЕМА ПЕРЕНЕСЕННЯ буде виконуватися з переходом, а команда ПЕРЕХІД, ЯКЩО ПЕРЕНЕСЕННЯ – без переходу.

Команди ПЕРЕХІД, ЯКЩО ПЕРЕНЕСЕННЯ та ПЕРЕХІД, ЯКЩО НЕМА ПЕРЕНЕСЕННЯ використовують, наприклад, для порівняння двох чисел A та B. Якщо $A < B$, то в результаті віднімання $A - B$ з’явиться біт позики в старший розряд, і прапорець перенесення буде встановлено в 1. Якщо прапорець перенесення встановиться в 0, то це означає, що $A \geq B$. Якщо ж одночасно прапорець нуля дорівнює 0 (ненульовий результат віднімання) тоді $A > B$. Таке віднімання можна здійснити за допомогою команди ПОРІВНЯННЯ.

1	Перехід, якщо є перенесення	
2	Безпосередня	
3	**JC B_2, B_3**	
4	**B_3 B_2 → PC**	
5	DAh	332q
6	Команда займає 3 байти пам'яті, виконується за 3 цикли, 10 тактів	
7	Якщо прапорець перенесення встановлено в 1, то під час виконання команди в молодший байт лічильника команд **PC** записується другий байт команди **B_2**, а в старший байт **PC** – третій байт команди **B_3**. Керування передається команді, яка буде занесена в **PC** (за адресою **B_3 B_2).** Якщо прапорець перенесення встановлено в 0, то керування передається наступній команді	
8	Виконання команди не впливає на регістр стану	

1	Перехід, якщо нема перенесення	
2	Безпосередня	
3	**JNC B_2, B_3**	
4	**B_3 B_2 → PC**	
5	D2h	322q
6	Команда займає 3 байти пам'яті, виконується за 3 цикли, 10 тактів	
7	Якщо прапорець перенесення встановлено в 0, то під час виконання команди в молодший байт лічильника команд **PC** записується другий байт команди **B_2**, а в старший байт **PC** – третій байт команди **B_3**. Керування передається команді, яка буде занесена в **PC** (за адресою **B_3 B_2).** Якщо прапорець перенесення встановлено в 1, то керування передається наступній команді	
8	Виконання команди не впливає на регістр стану	

Команди ПЕРЕХІД, ЯКЩО МІНУС та ПЕРЕХІД, ЯКЩО ПЛЮС перевіряють значення прапорця знака. Якщо його значення рівне 1, то це свідчить, що в результаті попередньої дії старший сьомий розряд результату встановлено в 1. У загальному випадку це не означає, що результат – дійсно від'ємне число. Наприклад, результат додавання додатних чисел 01110000b (70h)

та 00010000b (10h) є «від'ємне» число 10000000b (80h). Однак, якщо програма оперує з числами, записаними в оберненому та доповненому кодах, сьомий розряд байта результату вказує на знак числа і може бути використаний програмістом. Тому, якщо прапорець знака встановлено в 1, у результаті виконання команди ПЕРЕХІД, ЯКЩО МІНУС буде здійснено перехід за адресою, яка міститься в другому та третьому байтах програми, а при нульовому значенні прапорця такий перехід здійснено не буде. Якщо прапорець встановлено в 0, то навпаки, команда ПЕРЕХІД, ЯКЩО ПЛЮС буде виконуватися з переходом, а команда ПЕРЕХІД, ЯКЩО МІНУС – без переходу.

Команди ПЕРЕХІД, ЯКЩО МІНУС та ПЕРЕХІД, ЯКЩО ПЛЮС можна використовувати для порівняння двох чисел A та B тільки в тому випадку, якщо ці числа записані в прямому або оберненому кодах, оскільки в цьому випадку в старшому сьомому розряді додатне число повинно містити 0, а від'ємне – 1.

1	Перехід, якщо результат від'ємний	
2	безпосередня	
3	**JM B_2, B_3**	
4	**B_3 B_2 → PC**	
5	F2h	372q
6	Команда займає 3 байти пам'яті, виконується за 3 цикли, 10 тактів	
7	Якщо прапорець знака встановлено в 1, то під час виконання команди в молодший байт лічильника команд **PC** записується другий байт команди **B_2**, а в старший байт **PC** – третій байт команди **B_3**. Керування передається команді, яка буде занесена в **PC** (за адресою **B_3 B_2**). Якщо прапорець знака встановлено в 0, то керування передається наступній команді	
8	Виконання команди не впливає на регістр стану	

1	Перехід, якщо результат додатний	
2	безпосередня	
3	**JP B_2, B_3**	
4	**B_3 B_2 → PC**	
5	FAh	362q
6	Команда займає 3 байти пам'яті, виконується за 3 цикли, 10 тактів	
7	Якщо прапорець знака встановлено в 0, то під час виконання команди в молодший байт лічильника команд **PC** записується другий байт команди **B_2**, а в старший байт **PC** – третій байт команди **B_3**. Керування передається команді, яка буде занесена в **PC** (за адресою **B_3 B_2**). Якщо прапорець знака встановлено в 1, то керування передається наступній команді	
8	Виконання команди не впливає на регістр стану	

Команди ПЕРЕХІД, ЯКЩО ПАРНІСТЬ та ПЕРЕХІД, ЯКЩО НЕ ПАРНІСТЬ перевіряють значення прапорця парності. Якщо його значення рівне 1, то це свідчить, що в результат попередньої дії має парну кількість розрядів, установлених в 1. Тому в результаті виконання команди ПЕРЕХІД, ЯКЩО ПАРНІСТЬ буде здійснено перехід за адресою, яка міститься в другому та третьому байтах програми, а при нульовому значенні прапорця такий перехід здійснено не буде. Якщо прапорець установлено в 0, то навпаки, команда ПЕРЕХІД, ЯКЩО НЕ ПАРНІСТЬ буде виконуватися з переходом, а команда ПЕРЕХІД, ЯКЩО ПАРНІСТЬ – без переходу.

Перевірка на парність використовується для виявлення помилок під час передавання чи приймання даних через зовнішній порт. У такому випадку після передавання даних обов'язково передається так звана контрольна сума, а в спрощеному варіанті – біт парності. Цей біт установлюється таким чином, щоб загальна сума одиниць в усіх байтах повідомлення, включаючи біт парності, була парна. Якщо пристрій-приймач виявить непарну кількість бітів, то це свідчить про помилку передавання. Таким чином перевірку на парність можна використовувати, здійснюючи введення даних з накопичувачів чи мережі.

1	Перехід, якщо результат парний	
2	безпосередня	
3	**JPE B_2, B_3**	
4	**B_3 B_2 → PC**	
5	EAh	352q
6	Команда займає 3 байти пам'яті, виконується за 3 цикли, 10 тактів	
7	Якщо прапорець парності встановлено в 1, то під час виконання команди в молодший байт лічильника команд **PC** записується другий байт команди **B_2**, а в старший байт **PC** – третій байт команди **B_3**. Керування передається команді, яка буде занесена в **PC** (за адресою **B_3 B_2**). Якщо прапорець парності встановлено в 0, то керування передається наступній команді	
8	Виконання команди не впливає на регістр стану	

1	Перехід, якщо результат непарний	
2	безпосередня	
3	**JPO B_2, B_3**	
4	**B_3 B_2 → PC**	
5	E2h	342q
6	Команда займає 3 байти пам'яті, виконується за 3 цикли, 10 тактів	
7	Якщо прапорець парності встановлено в 0, то під час виконання команди в молодший байт лічильника команд **PC** записується другий байт команди **B_2**, а в старший байт **PC** – третій байт команди **B_3**. Керування передається команді, яка буде занесена в **PC** (за адресою **B_3 B_2).** Якщо прапорець парності встановлено в 1, то керування передається наступній команді	
8	Виконання команди не впливає на регістр стану	

2 Порядок виконання експериментів

Запустіть емулятор.

Виконайте наведені нижче приклади програм у потактовому режимі. Зафіксуйте, за скільки циклів і тактів виконується кожна із програм. Зафіксуйте значення прапорців після виконання кожної із команд.

Експеримент 1. *Вивчення команд безумовного переходу.*

1.1 Вивчення команди безумовного переходу ***JMP adr.***

Введіть нижченаведену програму.

Адреса комірки ОЗП	код команди	Мнемонічний запис команди	Команда мікропроцесора
0000		MVI A, d8	дані наступної комірки ОЗП → регістр **A**
0001		LXI B, d16	дані 01h → регістр **A**
0002		INR A	**A** + 1 → регістр **A**
0003		JMP adr	перехід на адресу 000Ah
0004		LDAX B	молодший байт адреси
0005		NOP	старший байт адреси
0006		DCR A	**A** - 1 → регістр **A**
0007		MOV B, A	**A** → регістр **B**
0008		ADD B	**A** + **B** → регістр **A**
0009		ANA B	**A & B** → регістр **A**
000A		JMP adr	перехід на адресу 0002h
000B		STAX B	молодший байт адреси
000C		NOP	старший байт адреси
000D		HLT	зупинка програми
-	-	-	-

Складіть блок-схему алгоритму програми. Покажіть, які із команд програми будуть виконуватися, а які – ні. За яких умов програма закінчить своє виконання і який результат буде отримано?

1.2 Вивчення команди безумовного переходу ***PCHL.***

Введіть нижченаведену програму.

Адреса комірки ОЗП	код команди	Мнемонічний запис команди	Команда мікропроцесора
0000	3E	MVI A, d8	дані наступної комірки ОЗП → регістр **A**
0001	01	LXI B, d16	дані 01h → регістр **A**
0002	26	MVI H, d8	дані наступної комірки ОЗП → регістр **H**
0003	00	NOP	старший байт адреси
0004	2E	MVI L, d8	дані наступної комірки ОЗП → регістр **L**
0005	0A	LDAX B	молодший байт адреси
0006	3C	INR A	**A** + 1 → регістр **A**
0007	E9	PCHL	**HL** → регістр **PC**
0008	24	INR H	**H** + 1 → регістр **H**
0009	2C	INR L	**L** + 1 → регістр **L**
000A	3D	DCR A	**A** - 1 → регістр **A**
000B	87	ADD A	**A** + **A** → регістр **A**
000C	C3	JMP adr	перехід на адресу 0002h
000D	02	STAX B	молодший байт адреси
000E	00	NOP	старший байт адреси
000F	76	HLT	зупинка програми
-	-	-	-

Які з команд програми ніколи не будуть виконані? Чим відрізняється команда ***PCHL*** від команди ***JMP adr***? Зробіть висновки.

Експеримент 2. *Вивчення команд умовного переходу* ***JZ adr*** *та* ***JNC adr.***

Постановка задачі. У регістрах B та C записані числа, які необхідно порівняти. Після виконання програми в регістрі B повинно бути записане більше число. Якщо числа одинакові, то регістри B та C треба обнулити.

Введіть нижченаведену програму.

Адреса комірки ОЗП	код команди	Мнемонічний запис команди	Команда мікропроцесора
0000		MOV A, B	**B** → регістр **A**
0001		SUB C	**A - C** → регістр **A**
0002		JZ adr	ПЕРЕХІД, ЯКЩО НУЛЬ на адресу 000Eh
0003		MVI C, d8	молодший байт адреси
0004		NOP	старший байт адреси
0005		JNC adr	ПЕРЕХІД, ЯКЩО НЕМА ПЕРЕНЕСЕННЯ на адресу 0011h
0006		LXI D, d16	молодший байт адреси
0007		NOP	старший байт адреси
0008		MOV A, B	**B** → регістр **A**
0009		MOV B, C	**C** → регістр **B**
000A		MOV C, A	**A** → регістр **C**
000B		JMP adr	ПЕРЕХІД на адресу 0011h
000C		LXI D, d16	молодший байт адреси
000D		NOP	старший байт адреси
000E		XRA A	обнулення **A**
000F		MOV B, A	**A** → регістр **B**
0010		MOV C, A	**A** → регістр **C**
0011		HLT	зупинка програми
-	-	-	-

Задайте початкові значення B=50h, C=20h. Запустіть програму на виконання в покроковому режимі. Запишіть значення прапорців стану після виконання команди ***SUB C***. Запишіть усі команди, які будуть виконані наступними. Задайте інші значення регістрів: спочатку B=35h, C=67h, а потім B=33h, C=33h. Запустіть програму для кожного із наборів. Прослідкуйте за значеннями прапорців стану та послідовністю виконання команд. Складіть блок-схему алгоритму програми. Зробіть висновки.

Експеримент 3. *Дослідження програми обробки масиву.*

Потрібно знайти суму елементів масиву (числа без знака), розташованого починаючи з адреси 0B02h. Кількість оброблюваних елементів масиву міститься за адресою 0B01h. Результат обробки необхідно помістити в елемент пам'яті з адресою 0B00h.

У рамках виконання цього завдання необхідно:

а) скласти програму зі знаходження суми елементів масиву, враховуючи зміни для кожного варіанта, задані у вигляді таблиці 2.1;

б) підготувати програму до запису в емулятор, перевівши її в машинний код (таблиця 2.2, додаток А);

в) занести створену програму в ОЗП емулятора, досліджувати її (при цьому елементи масиву вибрати на власний розсуд, але їх кількість повинна бути не менше 20), проаналізувати результат, зробити висновок про роботу, окремо розглянути питання функціонування системи і зміну утримуваного регістра стану (прапорців) склавши і заповнивши таблицю 2.3 для командного режиму роботи емулятора, в ній допустимо пропускати елементи (при проходженні циклу), що повторюються.

Таблиця 2.1 – Варіанти завдань

Призначення адреси пам'яті	**Значення адрес елементів пам'яті за варіантами, h**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
*	0B11h	0B21h	0B31h	0B41h	0B51h	0B61h	0B71h	0B81h	0B91h	0BA1h
**	0B12h	0B22h	0B32h	0B42h	0B52h	0B62h	0B72h	0B82h	0B92h	0BA2h
***	0B10h	0B20h	0B30h	0B40h	0B50h	0B60h	0B70h	0B80h	0B90h	0BA0h
* Кількість оброблюваних чисел. ** Початок розташування масиву чисел. *** Місце розташування результату										

Таблиця 2.2 – Програма обробки масиву

Мітка	Етап	Коментар етапів алгоритму	Мнемоніч-ний запис команди	Команда мікропроцесора
1	2	3	4	5
	1	Встановити адресу 0B01h	LXI H, 0B01h	*Призначення команди LXI:* Безпосереднє завантаження пари регістрів H, L
	2	Записати кількість чисел у регістр B	MOV B,M	*Призначення команди MOV:* Вміст КП, адреса якої вказана в регістрах H, L, пересилається в B
	3	Зменшити B на одиницю	DCR B	*Призначення команди DCR:* Вміст регістра B зменшується на одиницю

Продовження таблиці 2.2

1	2	3	4	5
	4	Збільшити адресу H на одиницю	INX H	*Призначення команди INX:* Вміст пари регістрів H, L збільшується на одиницю
	5	Завантажити в A число з адреси M, вважати його max	MOV A,M	*Призначення команди MOV:* Вміст КП з адресою в H, L пересилається в A
Mk1:	6	Збільшити адресу H на одиницю	INX H	*Призначення команди INX:* Вміст пари регістрів H, L збільшується на одиницю
	7	Порівняти A змістом адреси M	CMP M	*Призначення команди CMP:* Вміст КП, адреса якої вказана в H, L віднімається від вмісту A. При цьому вміст A не змінюється. За результатами порівняння встановлюються прапорці: C = 1, якщо A <КП [H, L]
	8	Перевірити, чи максимум не змінився	JNC Mk2	*Призначення команди JNC:* Якщо прапорець C не дорівнює 1, то перехід по мітці Mk2, що являє собою явно задану адресу
	9	Завантажити в A число з адреси M вважати його max	MOV A,M	*Призначення команди MOV:* Вміст КП, адреса якої вказана в регістрах H, L, пересилається в A
Mk2:	10	Зменшити B на одиницю	DCR B	*Призначення команди DCR:* Вміст регістра B зменшується на одиницю
	11	Перевірити закінчення циклу	JNZ Mk1	*Призначення команди JNZ:* Якщо останній результат не дорівнює 0, то перехід по мітці Mk1, являє собою явно задану адресу ділянки програми
	12	Зберегти результат за адресою 0B00h	STA 0B00h	Призначення команди STA: Вміст A пересилається в КП за вказаною адресою
			HLT	*Призначення команди HLT:* Останов

Таблиця 2.3 – Таблиця результатів виконання програми

Адреса комірки ОЗП	Команда	Цикл	Такт	Лічильник команд	Регістр команд	Регістр стану (PSW, тип циклу)	Регістр ознак ZSPCAC	Акумулятор

Контрольні запитання

1 Яке основне призначення команд безумовних та умовних переходів?

2 Яким чином здійснюються безумовні та умовні переходи на вказану адресу?

3 На які прапорці регістра стану впливає результат виконання команд переходів?

4 Яким чином вказується адреса, на яку необхідно перейти?

5 Назвіть основні недоліки команд безумовних та умовних переходів.

6 Вкажіть відмінності між командами **PCHL** та **JMP adr**. У яких випадках застосовуються ці команди?

7 Чим відрізняються команди безумовних та умовних переходів між собою?

8 З якою метою використовують команди безумовних переходів?

9 Які прапорці регістра стану використовуються командами умовних переходів?

10 Перерахуйте всі команди умовних переходів? Чому таких команд є вісім? Чи існує команда переходу, яка враховує прапорець додаткового перенесення?

11 З якою метою використовують команди переходу, які залежать від значення прапорця парності?

12 Яким чином здійснюється порівняння чисел у мікропроцесорі КР580ВМ80? Які прапорці стану використовуються при цьому?

13 У яких випадках використовують команди ПЕРЕХІД, ЯКЩО МІНУС та ПЕРЕХІД, ЯКЩО ПЛЮС?

Список літератури

1 Токхайм Р. Микропроцессоры. Курс и упражнения. – М.: Энергоатомиздат, 1988.

2 Костинюк Л. Д., Паранчук Я. С., Щур І. З. Мікропроцесорні засоби та системи: Навч. посібник. – 2-ге вид., перероб., доп. – Львів: Видавництво Національного університету "Львівська політехніка", 2002.

3 Медведев В. С., Орлов Г. А., Рассадкин Ю. И. и др. Управляющие и вычислительные устройства роботизированных комплексов на базе микро-ЭВМ. – М.: Высш.шк.,1990.

4 Зубчук В. И., Сигорский В. П., Шкуро А. Н. Справочник по цифровой схемотехнике. – К.:Техника, 1990.

Додаток А

Система команд МП КР580ІК80А

Таблиця А.1

Мнемо-код	Дов-жина, байт	Число циклів	Двійко-вий код	Опис операції	Шістнад-цятковий код	Умови				
						C	Z	S	P	AC
1	2	3	4	5	6	7	8	9	10	11
Команди арифметичних операцій										
ADD M	1	2	10000110	Додавання акумулятора до пам’яті (A)←(A)+[M]	86	+	+	+	+	+
ADC M	1	2	10001110	Додавання акумулятора до пам’яті та перенесення (A)←(A)+[M]+(перенесення)	8E	+	+	+	+	+
SUB M	1	2	10010110	Віднімання пам’яті від акумулятора (A)←(A)-[M]	96	+	+	+	+	+
SBB M	1	2	10011110	Віднімання пам’яті від акумулятора з позикою (A)←(A)-[M]-(позика)	9E	+	+	+	+	+
INR M	1	3	00110100	Збільшення на одиницю вмісту комірки, що адре-сується регістрами H, L [M]←[M]+1	34	-	+	+	+	+
DSR M	1	3	00110101	Зменшення на одиницю вмісту комірки пам’яті, що адресується регістрами H, L [M]←[M]-1	35	-	+	+	+	+
ADI	2	3	11000110	Додавання акумулятора до другого байта команди (A)←(A)+<B2>	C6	+	+	+	+	+
ACI	2	2	11001110	Додавання акумулятора до другого байта команди і перенесення (A)←(A)+<B2>+(перенесення)	CE	+	+	+	+	+
SUI	2	2	11010110	Віднімання другого байта команди від акумулятора (A)←(A)-<B2>	D6	+	+	+	+	+
SBI	2	2	11011110	Віднімання другого байта команди від акумулятора з позикою (A)←(A)-<B2>	DE	+	+	+	+	+
INR r	1	1	00DDD100	Інкремент регістра r (r)←(r)+1	··	-	+	+	+	+
DSR r	1	1	00DDD101	Декремент регістра r (r)←(r)-1	··	-	+	+	+	+
ADD r	1	1	10000DDD	Додавання регістра r до акумулятора (A)←(A)+(r)	8.	+	+	+	+	+
ADD r	1	1	10001DDD	Додавання регістра r до акумулятора і перенесення (A)←(A)+(r)+перенесення	8.	+	+	+	+	+

Продовження таблиці А.1

1	2	3	4	5	6	7	8	9	10	11
SUB r	1	1	10010DDD	Віднімання регістра від акумулятора (A)←(A)-(r)	9.	+	+	+	+	+
SBB r	1	1	10011DDD	Віднімання регістра від акумулятора з позикою (A)←(A)-(r)-(позика)	9.	+	+	+	+	+
Команди логічних операцій										
ANA M	1	2	10100110	Кон'юнкція акумулятора з коміркою пам'яті (A)←(A)^[M]	A6	0	+	+	+	0
ORA M	1	1	10110110	Диз'юнкція акумулятора з коміркою пам'яті (A)←(A)∨[M]	B6	0	+	+	+	0
XRA M	1	2	10101110	Додавання по модулю 2 акумулятора до комірки пам'яті (A)←(A)⊕[M]	AE	0	+	+	+	0
CMP M	1	2	10111110	Порівняння акумулятора з коміркою пам'яті (A)=[M]	BE	+	+	+	+	+
ANI	2	2	11100110	Кон'юнкція акумулятора з другим байтом команди (A)←(A)^ <B2>	E6	0	+	+	+	0
ORI	2	2	11110110	Диз'юнкція акумулятора з другим байтом команди (A)←(A)∨<B2>	F6	0	+	+	+	0
XRI	2	2	11101110	Додавання по модулю 2 акумулятора з другим байтом команди (A)←(A) ⊕<B2>	EE	0	+	+	+	0
CPI	2	2	11111110	Порівняння акумулятора з другим байтом команди (A)=<B2>	FE	+	+	+	+	+
ANA r	1	1	10100DDD	Кон'юнкція акумуля-тора з регістром r (A)←(A)^(r)	A.	0	+	+	+	0
ORA r	1	1	10110DDD	Диз'юнкція акуму-лятора з регістром r (A)←(A)∨(r)	B.	0	+	+	+	0
XRA r	1	1	10101DDD	Додавання по модулю 2 акумулятора з регістром r (A)←(A)⊕(r)	A.	0	+	+	+	0
CMP r	1	1	10111DDD	Порівняння акумуля-тора з регістром r (A)=(r)	B.	+	+	+	+	+

Продовження таблиці А.1

1	2	3	4	5	6	7	8	9	10	11
Команди пересилань										
MOV r_1, r_2	1	1	01DDDSSS	Пересилання з регістра r_2 у регістр r_1 $(r_1)\leftarrow(r_2)$	…	-	-	-	-	-
MOV r, M	1	2	01DDD110	Пересилання з комірки пам'яті у регістр r (r)←[M]	…	-	-	-	-	-
MOV M, r	1	2	01110SSS	Пересилання з регістра r у комірку пам'яті [M]←(r)	7.	-	-	-	-	-
MVI r	2	2	00DDD110	Пересилання другого байта команди у регістр r	…	-	-	-	-	-
MVI M	2	3	00110110	Пересилання другого байта команди у комірку пам'яті	36	-	-	-	-	-
Команди читання/запису										
STA	3	4	00110010	Запам'ятовування вмісту акумулятора за адресою, вказаною другим і третім байтами команди [<B3>,<B2>]←(A)	32					
LDA	3	4	00111010	Завантаження акумулятора вмістом комірки, адреса якої вказана другим і третім байтами команди (A)←[<B3>,<B2>]	3A	-	-	-	-	-
LXI B	3	3	00000001	Завантаження у пару регістрів B C третього та другого байтів команди (C)←<B2>; (B)←<B3>	01	-	-	-	-	-
LXI D	3	3	00010001	Завантаження у пару регістрів DE третього та другого байтів команди (E)←<B2>; (D)←<B3>	11	-	-	-	-	-
LXI H	3	3	00100001	Завантаження у пару регістрів HL третього та другого байтів команди (L)←<B2>; (H)←<B3>	21	-	-	-	-	-
LDAX D	1	2	00011010	Завантаження в акумулятор числа з комірки з адресою у парі регістрів DE (A)←[(D)(E)]	1A	-	-	-	-	-
LDAX B	1	2	00001010	Завантаження в акумулятор числа з комірки з адресою, що записана у парі регістрів BC (A)←[(B)(C)]	OA	-	-	-	-	-

Продовження таблиці А.1

1	2	3	4	5	6	7	8	9	10	11
STAX B	1	2	00000010	Завантаження вмісту акумулятора за адресою, вказаною в парі регістрів BC (A)←[(B)(C)]	02	-	-	-	-	-
STAX D	1	2	00010010	Завантаження вмісту акумулятора за адресою, вказаною в парі регістрів DE (A)←[(D)(E)]	12	-	-	-	-	-
SHL D	3	5	00100010	Завантаження вмісту регістрів HL за адресою, вказаною у другому і третьому байтах команди L→[<B3><B2>] H→[<B3>,<B2>+1]	22	-	-	-	-	-
LHL D	3	5	00101010	Завантаження регістрів HL з комірки за адресою, вказаною у другому і третьому байтах команди L←[<B3>,<B2>] H←[<B3>,<B2>+1]	2A	-	-	-	-	-
Команди зсуву										
RAL	1	1	00010111	(C)←A7 – зсув у тригер перенесення A0←(C) – зсув біта перенесення в АО A_{m+1}←A_m	17	+	-	-	-	-
RAR	1	1	00011111	Зсув уліво з перенесенням A7←(C) (C)←A0 A_m←A_{m+1}	1F	+	-	-	-	-
RLC	1	1	00000111	Зсув уліво циклічний A_{m+1}←A_m A0←A7 (C)←A7	07	+	-	-	-	-
RRC	1	1	00001111	Зсув управо циклічний A_m←A_{m+1} A7←A0 (C)←A0	0F	+	-	-	-	-
Команди умовних переходів										
JMP	3	3	11000011	Безумовний перехід за адресою, вказаною у другому та третьому байтах команди (PC)←<B3>,<B2>	C3	-	-	-	-	-
JC	3	3	11011010	Перехід за знаком перенесення, якщо (C)=1, то (PC)←<B3>,<B2>, інакше (PC)←(PC)+3	DA	-	-	-	-	-

Продовження таблиці А.1

1	2	3	4	5	6	7	8	9	10	11
JNC	3	3	11010010	Перехід за відсутністю перенесення, якщо (C)=0, то (PC)←<B3>,<B2>, інакше (PC)←(PC)+3	D2	-	-	-	-	-
JZ	3	3	11001010	Перехід по нулю, якщо (Z)=1, то (PC)←<B3>,<B2>, інакше (PC)←(PC)+3	CA	-	-	-	-	-
JNZ	3	3	11000010	Перехід по ненулю, якщо (Z)=0, то (PC) ← <B3>,<B2>, інакше (PC)←(PC)+3	C2	-	-	-	-	-
JP	3	3	111100S0	Перехід по плюсу, якщо (S)=0, то (PC)←<B3>,<B2>, інакше (PC)←(PC)+3	F2	-	-	-	-	-
JM	3	3	11111010	Перехід по мінусу, якщо (S)=1, то (PC)←<B3>,<B2>, інакше (PC)←(PC)+3	FA	-	-	-	-	-
JPE	3	3	11101010	Перехід по парності, якщо (P)=1, то (PC)←<B3>,<B2>, інакше (PC)←(PC)+3	EA	-	-	-	-	-
JPO	3	3	11100010	Перехід по непарності, якщо (P)=0, то (PC)←<B3>,<B2>, інакше (PC)←(PC)+3	E2	-	-	-	-	-
Команди переходу до підпрограм і повернення з них										
CALL	3	5	11001101	Перехід на підпрограму за адресою, вказаною у другому і третьому байтах команди. Занесення вмісту (PC) у стек за адресою вказівника стека. Зменшення вмісту регістра (SP) на 2 [SP-1][SP-2]←(PC) (SP)←(SP)-2 (PC)←<B3>,<B2>	CD	-	-	-	-	-
CC	3	3/5	11011100	Перехід на підпрограму по перенесенню, якщо (C)=1, то [SP-1][SP-2]←(PC) (SP)←(SP)-2 (PC)←<B3>,<B2>, інакше (PC)=(PC)+3	DC	-	-	-	-	-

Продовження таблиці А.1

1	2	3	4	5	6	7	8	9	10	11
CNC	3	3/5	11010100	Перехід на підпрограму за відсутністю перенесення, якщо (С)=0, то [SP-1][SP-2]←(PC) (SP)←(SP)-2 (PC)←<B3>,<B2>, інакше (PC)=(PC)+3	D4	-	-	-	-	-
CZ	3	3/5	11001100	Перехід на підпрограму по нулю, якщо (Z)=1, то [SP-1][SP-2]←(PC) (SP)←(SP)-2 (PC)←<B3>,<B2>, інакше (PC)=(PC)+3	CC	-	-	-	-	-
CNZ	3	3/5	11000100	Перехід на підпрограму по ненулю, якщо (Z)=1, то [SP-1][SP-2]←(PC) (SP)←(SP)-2 (PC)←<B3>,<B2>, інакше (PC)=(PC)+3	C4	-	-	-	-	-
CP	3	3/5	11110100	Перехід на підпрограму по плюсу. Умовою переходу є (S)=0	F4	-	-	-	-	-
CM	3	3/5	11111100	Перехід на підпрограму по мінусу. Умовою переходу є (S)=1	FC	-	-	-	-	-
CPE	3	3/5	11101100	Перехід на підпрограму по парності. Умовою переходу є (P)=1	EC	-	-	-	-	-
CPO	3	3/5	11100100	Перехід на підпрограму по непарності. Умовою переходу є (P)=0	E4	-	-	-	-	-
RET	1	3	11001001	Безумовне повернення з підпрограми (PC)←[SP][SP+1] (SP)←(SP)+2	C9	-	-	-	-	-
RC	1	1/3	11011000	Повернення з підпрограми по перенесенню, якщо (С)=1,то (PC)←[SP][SP+1] (SP)←(SP)+2, інакше (PC)=(PC)+1	D8	-	-	-	-	-
RNC	1	1/3	11010000	Повернення з підпрограми за відсутності перенесення. Умова повернення (С)=0	D0	-	-	-	-	-
RZ	1	1/3	11001000	Повернення з підпрограми по нулю. Умова повернення (Z)=1	C8	-	-	-	-	-
RNZ	1	1/3	11000000	Повернення з підпрограми по ненулю. Умова повернення (Z)=0	C0	-	-	-	-	-

Продовження таблиці А.1

1	2	3	4	5	6	7	8	9	10	11
RP	1	1/3	11110000	Повернення з підпрограми по плюсу. Умова повернення (S)=0	F0	-	-	-	-	-
RM	1	1/3	11111000	Повернення з підпрограми по мінусу. Умова повернення (S)=1	F8	-	-	-	-	-
RPE	1	1/3	11101000	Повернення з підпрограми по парності. Умова повернення (P)=1	E8	-	-	-	-	-
RPO	1	1/3	11100000	Повернення з підпрограми по непарності. Умова повернення (P)=0	E0	-	-	-	-	-
RCT	1	3	11AAA111	Повторний запуск [SP+1][SP+2]←(PC) (SP)←(SP)-2 (PC)=(00000000 00aaa000) вектор переривання	..	-	-	-	-	-
Команди звернення до стека										
LXI SP	3	3	00110001	Безпосереднє завантаження вказівника стека <B3> - в молодший розряд <B2> - в старший розряд (SP)←<B3><B2>	31	-	-	-	-	-
PUSH PSW	1	3	11110101	Введення до стека вмісту акумулятора і регістра умов [SP-1]← (A) [SP-2]←(F) (SP)←(SP)-2	F5	-	-	-	-	-
PUSH B	1	3	11000101	Введення до стека вмісту регістрів B і C [SP-1]←(B) [SP-2]←(C) (SP)←(SP)-2	C5	-	-	-	-	-
PUSH D	1	3	11010101	Введення до стека вмісту регістрів D і E	D5	-	-	-	-	-
PUSH H	1	3	11100101	Введення до стека вмісту регістрів H і L	E5	-	-	-	-	-
POP PSW	1	3	11110001	Виведення зі стека в акумулятор і регістр умов [SP]→(F) [SP+1]→(A) (SP)←(SP)+2	F1	+	+	+	+	+
POP B	1	3	11000001	Виведення зі стека в регістри B і C [SP]→(C) [SP+1]→(B) (SP)←(SP)+2	C1	-	-	-	-	-

Продовження таблиці А.1

1	2	3	4	5	6	7	8	9	10	11
POP D	1	3	11010001	Виведення зі стека в регістри D і E	D1	-	-	-	-	-
POP H	1	3	11100001	Виведення зі стека в регістри H і L	E1	-	-	-	-	-
XTHL	1	5	11100011	Обмін вмістом регістрів HL з вершиною стека	E3	-	-	-	-	-
SPHL	1	1	11111001	Пересилання вмісту регістрів HL у вказівник стека	F9	-	-	-	-	-
DAD SP	1	3	00111001	Додавання вмісту регістрової пари HL і вказівника стека (H)(L)+(SP)→(H)(L)	39	+	-	-	-	-
INX SP	1	1	00110011	Інкремент вказівника стека (SP)←(SP)+1	33	-	-	-	-	-
DCX SP	1	1	00111011	Декремент вказівника стека (SP)←(SP)-1	3B	-	-	-	-	
Команди введення-виведення										
IN	2	3	11011011	Введення вмісту УВВ, адреса якого розміщена у другому байті команди (A)←(дані, що вводяться)	DB	-	-	-	-	-
OUT	2	3	11010011	Виведення вмісту акумулятора на пристрій виводу, адреса якого розміщена у другому байті команди (A)→(дані, що вводяться)	D3	-	-	-	-	-
EI	1	1	11111011	Дозвіл переривання «1»→ у тригер дозволу переривання	FB	-	-	-	-	-
DI	1	1	11110011	Заборона переривання «0»→ у тригер дозволу переривання	F3	-	-	-	-	-
Інші команди										
CMA	1	1	00101111	Інвертування акумулятора	2F	-	-	-	-	-
STC	1	1	00110111	Встановлення тригера перенесення в «1» (C)←1	37	1	-	-	-	-
CMC	1	1	00111111	Інвертування прапорця перенесення (C)←($\overline{C}$)	3F	+	-	-	-	-
PCHL	1	1	11101001	Завантаження вмісту регістрів HL у лічильник команд (PC)←(H)(L)	E9	-	-	-	-	-
INX B	1	1	00000011	Інкремент пари регістрів BC (BC)←(BC)+1	03	-	-	-	-	-

Продовження таблиці А.1

1	2	3	4	5	6	7	8	9	10	11
INX D	1	1	00010011	Інкремент пари регістрів DE (DE)←(DE)+1	13	-	-	-	-	-
INX H	1	1	00100011	Інкремент пари регістрів HL (HL)←(HL)+1	23	-	-	-	-	-
DAD B	1	3	00001001	Додавання вмісту регістрових пар (HL)←(BC)+(HL)	09	+	-	-	-	-
DAD H	1	3	00101001	Додавання вмісту регістрових пар (HL)←(HL)+(HL)	29	+	-	-	-	-
DAD D	1	3	00011001	Додавання вмісту регістрових пар (HL)←(HL)+(DC)	19	+	-	-	-	-
DCX B	1	1	00001011	Декремент пари регістрів (BC)←(BC)-1	0B	-	-	-	-	-
DCX H	1	1	00101011	Декремент пари регістрів (HL)←(HL)-1	2B	-	-	-	-	-
DCX D	1	1	00011011	Декремент пари регістрів (DE)←(DE)-1	1B	-	-	-	-	-
XCNG	1	1	11101011	Обмін вмістом регістрів (HL) і (DE) (H)↔(D), (L)↔(E)	EB	-	-	-	-	-
DAA	1	1	00100111	Двійково-десяткова корекція вмісту акумулятора	27	+	+	+	+	+
HLT	1	1	01110110	Останов	76	-	-	-	-	-
NOP	1	1	00000000	Операція не виконується	00	-	-	-	-	-