МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

АПАРАТНІ ЗАСОБИ МІКРОПРОЦЕСОРНИХ СИСТЕМ

МЕТОДИЧНІ ВКАЗІВКИ

до виконання робіт комп’ютерного практикуму для студентів напрямів підготовки «Автоматизація та комп'ютерно-інтегровані технології» і «Інженерна механіка»

Рекомендовано Вченою радою інженерно-хімічного факультету

Київ
НТУУ “КПІ”
2012

Апаратні засоби мікропроцесорних систем : Метод. вказівки до викон. робіт комп. практ. для студ. напр. „Автоматизація та комп'ютерно-інтегровані технології” і „Інженерна механіка” / Уклад.: М. В. Коржик, О. А. Козачок. – К. : НТУУ «КПІ», 2012. – 36 с.

Гриф надано Вченою радою ІХФ
(Протокол № 3 від 27 лютого 2012 р.)

Навчальне видання

АПАРАТНІ ЗАСОБИ МІКРОПРОЦЕСОРНИХ СИСТЕМ

Методичні вказівки до виконання робіт комп’ютерного практикуму для студентів напрямів підготовки «Автоматизація та комп'ютерно-інтегровані технології» і «Інженерна механіка»

Укладачі: Коржик Михайло Володимирович, канд. техн. наук
Козачок Олександр Анатолійович

Відповідальний редактор А. І. Жученко, докт. техн. наук, проф.

Рецензент С. Ю. Олійник, канд. техн. наук, доц.

За редакцією укладачів

Зміст

Вступ

Цей практикум призначено для вивчення елементів апаратного і програмного забезпечення систем, побудованих на базі мікропроцесора Intel 8080/8085.

На рис. 1, показана спрощена схема мікропроцесорної системи. Мікролаб – лабораторний комп'ютер [4], призначений для вивчення апаратної частини і програмного забезпечення – має ту ж структуру. Він створений на базі мікропроцесора (МП) КР580ИК80 (аналога Intel 8080/8085).

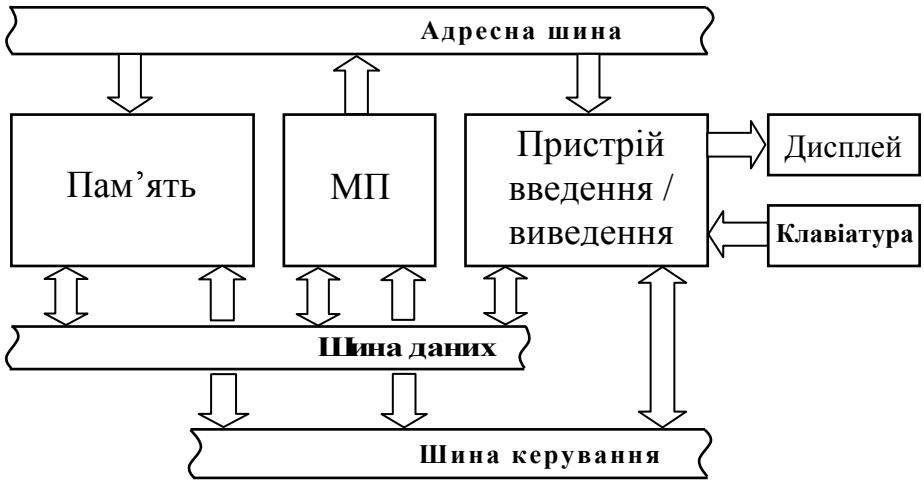

Рис. 1

Постійний пристрій запам'ятовування (ППЗ) побудовано на двох інтегральних схемах (ІС) КР556РТ5 сумарною місткістю 1 кілобайт. Місткість ППЗ може бути розширена на 0.5 кілобайта встановленням додаткового кристала КР556РТ5 у спеціальний адаптер, який є на платі приладу. ППЗ містить закодовану в машинних кодах КР580ІК80 програму монітора, який керує всіма елементами системи.

Пристрій запам'ятовування з довільною вибіркою (ОПЗ) побудовано на восьми ІС К565РУ2 по 1 кілобіту кожна. Сумарна місткість ОПЗ – 1 кілобайт. ОПЗ використовується для зберігання програми користувача і

для розміщення робочої області монітора.

Введення/виведення інформації від зовнішніх пристроїв здійснюється через програмований периферійний паралельний інтерфейс (ППІ) КР580ВВ55. У приладі передбачена імітація зовнішніх датчиків і приймачів інформації. Як датчик паралельного коду введені три ключі, які формують ТТЛ рівні. Приймачами паралельної інформації є вісім світлодіодних індикаторів. Як приймач послідовної інформації використовується динамік.

Для зв'язку з оператором прилад має клавіатуру і внутрішній дисплей. Інформація, що вводиться та виводиться представлена в шістнадцятковій системі числення.

Клавіатура зв'язана з системою через ППІ і застосовується для ручного введення інформації в ОПЗ і для керування операціями монітора. Призначення кожної клавіші наведене в табл. 1.

Таблиця 1

Клавіша	**Код**	**Функція клавіші**
СБРОС		Скидання системи і повернення до монітора в будь-який момент часу
0 – 9 ; A – F	00h – 09h ; 0Ah – 0Fh	Введення даних в шістнадцятковому коді
ПУСК	10h	Виконання програми з адреси, що відображається на дисплеї
ВОЗВР	11h	Повернення до виконання програми, початої по команді ПУСК
УСТ.АД	12h	Встановлення адреси і зчитування даних з пам'яті
АД –	13h	Зменшення адреси на 1 і зчитування даних з пам'яті
АД +	14h	Приріст адреси на 1 і зчитування даних з пам'яті
ЗП	15h	Запис даних у пам'ять і приріст адреси на 1
ВЫВОД	16h	Виведення даних з пам’яті на зовнішній пристрій
ВВОД	17h	Введення даних з зовнішнього пристрою в пам’ять

Вихідним пристроєм, що слугує для спостережень за внутрішнім станом системи, є дисплей, що складається з восьми 8-сегментних індикаторів. На них відображаються вхідні дані, адреси пам'яті, дані пам'яті і вміст регістрів МП відповідно до операцій клавіатури. Індикація інформації динамічна. Дані на індикатори передаються з восьми старших комірок ОПЗ за допомогою прямого доступу до пам'яті без участі МП.

Засобом керування всіма операціями системи є програма монітора. Всі функції монітора задаються за допомогою простих операцій клавіатури. Використовуючи набір команд МП (див. табл. Д.1 і Д.3), користувач може записати свою програму і дані в ОПЗ Мікролаба і виконати її, провівши настройку за допомогою крокового режиму і режиму переривання. В кроковому режимі програма переривається після виконання кожної команди, після чого можна спостерігати стан МП на дисплеї. Перехід у кроковий режим здійснюється за допомогою відповідного перемикача на платі приладу.

Для переривання програми в будь-якій точці (а не після кожної команди) монітор має регістр переривання (адреса молодшого байта 83F0h, старшого – 83F1h) та лічильник циклів переривання (адреса 83F2h). Перед початком виконання програми користувача адреса та номер циклу переривання повинні бути занесені у відповідні регістри робочої області монітора

Мікролаб забезпечує наступні режими і функції:

- автоматичний режим виконання програм;
- кроковий режим;
- скидання і ініціалізацію системи монітором при ввімкненні живлення і при надходженні сигналу СБРОС від відповідної клавіші на платі приладу;
- ручне введення інформації в ОПЗ з клавіатури;

- автоматичне виведення з ОПЗ на дисплей;
- контроль і корекцію записаної інформації;
- виконання програми, починаючи з будь-якої точки;
- переривання програми в будь-якій заданій точці;
- індикацію на дисплеї вмісту акумулятора і ознак стану МП при зупинці програми під час виконання її в кроковому режимі або в режимі переривання;
- корекцію вмісту акумулятора, ознакового регістру, регістрів загального призначення МП на будь-якому кроці виконання програми;
- повторний запуск за таблицею переходів по перериваннях, які подаються на прилад із зовнішніх пристроїв.

Таблиця 2.

Адреси	**Ємність пам'яті**	**Тип пам'яті**	**Призначення**
FFFFh – 8400h	31 кілобайт	Не використовується	
83FFh – 83C7h	57 байт	ОПЗ	Робоча область монітора
83C6h – 8000h	967 байт	ОПЗ	Область користувача
7FFFh – 0600h	30.5 кілобайт	Не використовується	
05FFh – 0400h	512 байт	ППЗ	Область користувача
03FFh – 0300h	256 байт	ППЗ	Додаткова область монітора
02FFh – 0000h	768 байт	ППЗ	Область монітора

Карта розподілу адрес пам'яті для кожного пристрою Мікролаба наведена в табл. 2.

Порядок виконання робіт

Роботи виконуються шляхом програмування на асемблері навчальних задач, наведених у завданнях до кожної роботи.

1. Записати програму у мнемонічних командах згідно з правилами мови асемблера.

2. Після перевірки програми викладачем перевести її в машинні коди у шістнадцятковому форматі.

3. Завантажити програму в ОЗП Мікролаба.

4. Зробити перевірку та (якщо треба) настройку програми.

5. Дослідити виконання програми (якщо треба в кроковому режимі).

6. Занести результати в протокол виконання роботи.

Оформрення результатів робіт

В протоколі виконання роботи для кожного завдання мають бути представлені:

1. Завдання на програмування.

2. Алгоритм розв’язання задачі.

3. Програма в асемблерних та машинних кодах.

4. Результати виконання програми.

Контрольна робота

СИСТЕМИ ЧИСЛЕННЯ ТА ДВІЙКОВА АРИФМЕТИКА

Мета роботи: закріпити практичні навички у переведенні чисел з однієї системи числення в іншу та виконанні елементарних арифметичних й логічних операцій у двійковій системі числення.

Порядок виконання роботи

1. Системи числення.

1.1. Перетворити десяткове число *A* в двійкову систему числення розрахунковим методом. Варіанти завдань наведені в табл. 3.

1.2. Отримане двійкове число перетворити у вісімкову та шістнадцяткову системи числення методом кодування.

1.3. Отримане двійкове число перетворити у десяткову систему числення табличним методом. Таблиця деяких еквівалентів наведена в табл. Д.4 (див. додаток).

Таблиця 3

Вар	*A*	Вар	*A*	Вар	*A*
1	91 .387	21	102 .921	41	120 .323
2	103 .025	22	126 .129	42	98 .225
3	130 .013	23	83 .888	43	79 .114
4	86 .125	24	106 .881	44	113 .417
5	115 .175	25	125 .188	45	76 .314
6	94 .195	26	101 .660	46	78 .449
7	116 .205	27	92 .661	47	121 .529
8	95 .210	28	81 .166	48	90 .150
9	129 .171	29	124 .666	49	112 .099
10	84 .001	30	117 .315	50	99 .155
11	108 .200	31	71 .513	51	80 .717
12	96 .371	32	104 .135	52	111 .819
13	74 .425	33	118 .172	53	93 .989
14	114 .777	34	97 .127	54	122 .333
15	128 .111	35	105 .272	55	73 .359
16	107 .020	36	119 .327	56	110 .002
17	82 .076	37	88 .790	57	89 .739
18	127 .106	38	109 .801	58	100 .921
19	77 .100	39	75 .079	59	123 .821
20	85 .999	40	72 .970	60	87 .730

2. Арифметичні операції.

2.1. Прийняти за число *A* цілу частину двійкового числа, отриманого в п. 1.

2.2. Обчислити суму числа *A* та числа *B*, наданого викладачем (*A*+*B*).

2.3. Обчислити різницю чисел *A* та *B* (*A*–*B*) у прямому, зворотному та доповненому кодах. Результати подати в природній формі зі знаком.

3. Логічні операції.

3.1. Виконати порозрядну логічну операцію кон’юнкції над двійковими числами *A* та *B*, отриманими в п. 2 (*A* and *B*).

3.2. Виконати порозрядну логічну операцію диз’юнкції над числами *A* та B (*A* or *B*).

3.3. Виконати порозрядну логічну операцію різноіменності над числами *A* та *B* (*A* xor *B*).

Робота № 1

ЛОГІЧНІ ТА АРИФМЕТИЧНІ ОПЕРАЦІЙ МОВИ АСЕМБЛЕРА

Мета роботи: дослідити структуру мікропроцесора та набути практичних навичок у виконанні елементарних логічних та арифметичних операцій за допомогою мови асемблера.

Теоретичні відомості

МП Intel 8085 може адресувати 64 кілобайта пам’яті. Крім того в своєму складі МП має регістри – особливі службові комірки пам’яті. Спрощена структурна схема МП зображена на рис. 2.

На рисунку: АЛП – арифметико-логічний пристрій; РТЗ – регістр тимчасового зберігання даних; F – ознаковий регістр та A – акумулятор, вміст яких разом складає *PSW* – слово стана процесора; B, C, D, E, H та L – регістри загального призначення (РЗП); Inc/Dec – схема інкрементора/декрементора, яка дозволяє при обробці адреси, команд та даних виконувати збільшення та зменшення на 1 безпосередньо в блоці РПЗ; SP – покажчик стека; PC – лічильник команд; РК – регістр команд. У цьому МП використовується 8-розрядна шина даних та 16-розрядна адресна шина.

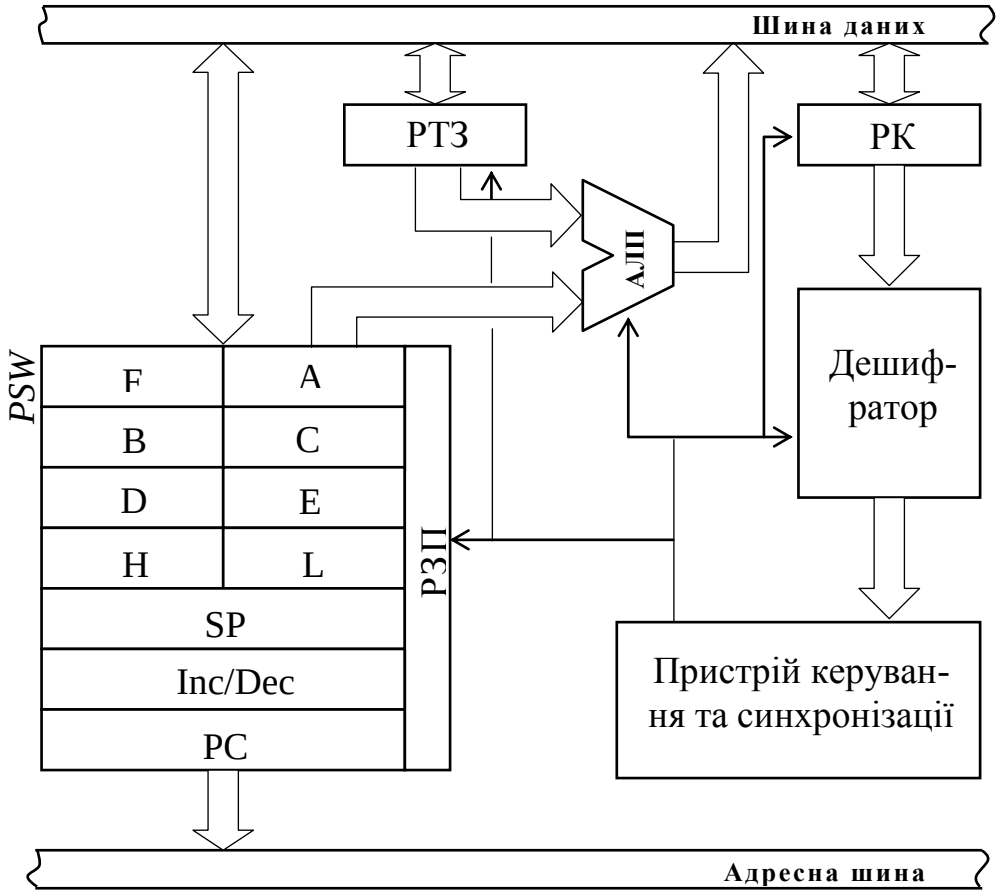

Рис. 2

Арифметичні та логічні операції в цьому МП завжди виконуються над операндами, один з яких розташовано в акумуляторі, а інший знаходиться в одному з РЗП, або зчитується безпосередньо з шини даних.

В Мікролабі (за допомогою монітора) є можливість спостереження за станом МП. При виконанні програми в кроковому режимі на перших чотирьох індикаторах виводиться останній стан РС (адреса наступної команди програми користувача). На наступні два індикатора виводиться вміст акумулятора, та на останні два індикатора – вміст ознакового регістру. Крім того, монітор запам’ятовує останній стан МП в області запам’ятовування регістрів ОПЗ, яка наведена в табл. 4.

Таблиця 4

Адреса	Регістр
83EBh	Акумулятор A
83EAh	Ознаковий регістр F
83E9h	B-регістр
83E8h	C-регістр
83E7h	D-регістр
83E6h	E-регістр
83E5h	H-регістр
83E4h	L-регістр
83E3h	Покажчик стека SP (старший байт)
83E2h	Покажчик стека SP (молодший байт)
83E1h	Лічильник команд PC (старший байт)
83E0h	Лічильник команд PC (молодший байт)

Користувач має змогу продивитися вміст будь-якої комірки пам’яті, вказуючи її адресу, а отже і спостерігати стан МП.

Завдання на програмування

Створити програму лінійної структури, яка виконує арифметичні операцій додавання та віднімання і порозрядні логічні операції кон’юнкції, диз’юнкції та різнойменності над 8-розрядними двійковими числами *A* та *B*, отриманими в попередній роботі. Крім того запрограмувати операцію перетворення від'ємного операнда у зворотний код, здійснити операцію віднімання у зворотному коді та порівняти результат из отриманим вище. Результати всіх операцій мають бути занесені в незадіяні регістри МП або у вільні комірки ОПЗ для подальшої перевірки.

Контрольні запитання

1. Де в МП можуть відбуватися арифметичні операції?

2. Який обсяг пам’яті може адресувати МП з 16-розрядною шиною адреси?

3. Які арифметичні команди мови асемблера ви знаєте?

4. Які логічні команди мови асемблера ви знаєте?

5. Які команди пересилання мови асемблера ви знаєте?

Робота № 2

АРИФМЕТИЧНІ ОПЕРАЦІЙ З БАГАТОБАЙТНИМИ ОПЕРАНДАМИ

Мета роботи: набути практичних навичок у виконанні арифметичних операцій з багатобайтними операндами за допомогою мови асемблера.

Теоретичні відомості

МП Intel 8085 працює з 8-розрядною шиною даних і має 8-розрядні регістри загального призначення та АЛП. Така структура МП дозволяє йому одночасно оперувати даними розміром в 1 байт, що відповідає діапазону зображення беззначних чисел 0 .. 255, або чисел зі знаком –128 .. 127 в десятковій системі числення. Для розширення діапазону зображення чисел використовують багатобайтні дані, операції з якими у 8-розрядному процесорі мають деяку специфіку.

Якщо дані мають розмір 2 байти, операцію додавання можна організувати за допомогою команди DAD, розмістивши операнди у регістрових парах МП.

Виконання операцій додавання та віднімання над даними будь-якого розміру можна організувати за допомогою команд ADC та SBB, як циклічний побайтовий процес з врахуванням значення біта переносу ознакового регістру F. При цьому треба забезпечити нульове початкове значення вказаного біта. Обнулити біт переносу можна за допомогою команд STC та CMC (див. додаток).

Завдання на програмування

1. Створити програму лінійної структури, яка виконує арифметичні операцій додавання та віднімання з врахуванням біта переносу над даними, що перевищують FFh.

2. Створити програму лінійної структури, яка виконує операцію додавання над даними (див. вище), розміщеними у регістрових парах. Порівняти результати роботи програм.

Контрольні запитання

1. Який діапазон зображення чисел забезпечує 16-розрядна регістрова пара?

2. Наведіть алгоритм віднімання операндів із врахуванням позики.

3. Які регістрові пари МП Intel 8085 можуть бути задіяні для організації операції додавання?

4. Як встановити індикатор переносу у відповідному біті ознакового регістру?

5. Як впливає результат дії команди DAD на вміст біта переносу?

Робота № 3

ВИКОРИСТАННЯ ППІ ДЛЯ ВИВЕДЕННЯ ДАНИХ

Мета роботи: дослідити структуру програмованого периферійного паралельного інтерфейсу та набути практичних навичок в його програмуванні для виведення даних.

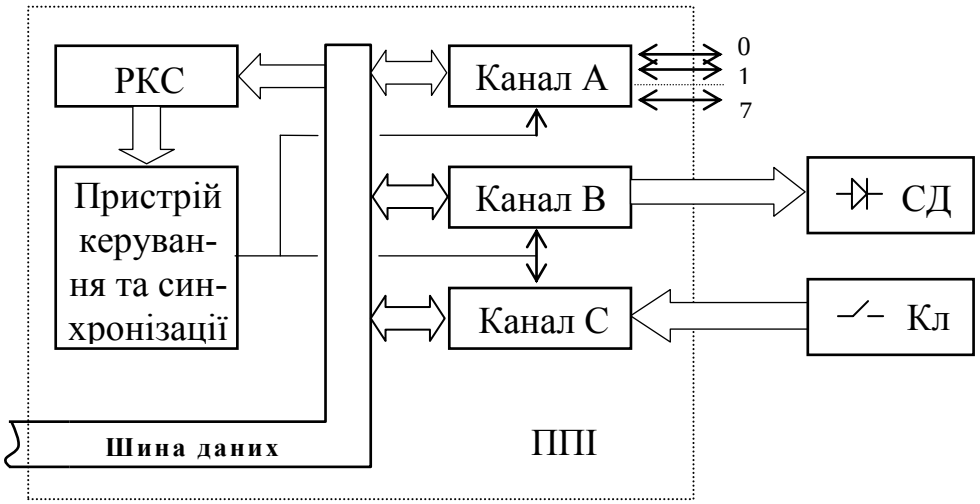

Рис. 3

Теоретичні відомості

Для зв’язку з зовнішніми пристроями МП Intel 8085 може адресувати 256 портів введення/виведення, тому адреса порту має розмір – 1 байт. В Мікролабі для цього використовується ППІ КР580ВВ55, спрощена структурна схема якого зображена на рис. 3.

ППІ має три канали – А, В та С, що створюють три 8-розрядних порти з різними характеристиками. Канали А та В мають групове керування, тобто всі вісім розрядів каналу одразу переводяться в режим введення або виведення. Канал С розділено на два 4-розрядних підканали та має порозрядне керування. Канал А та старший підканал С складають групу А, канал В та молодший підканал С утворюють групу В. Функціональне призначення каналів визначається кодом керуючого слова, яке завантажується в РКС – регістр керуючого слова. Крім того, Мікролаб обладнано СД – приймачем паралельної інформації на світлодіодах та Кл – датчиками паралельного коду у вигляді ключів ТТЛ рівнів.

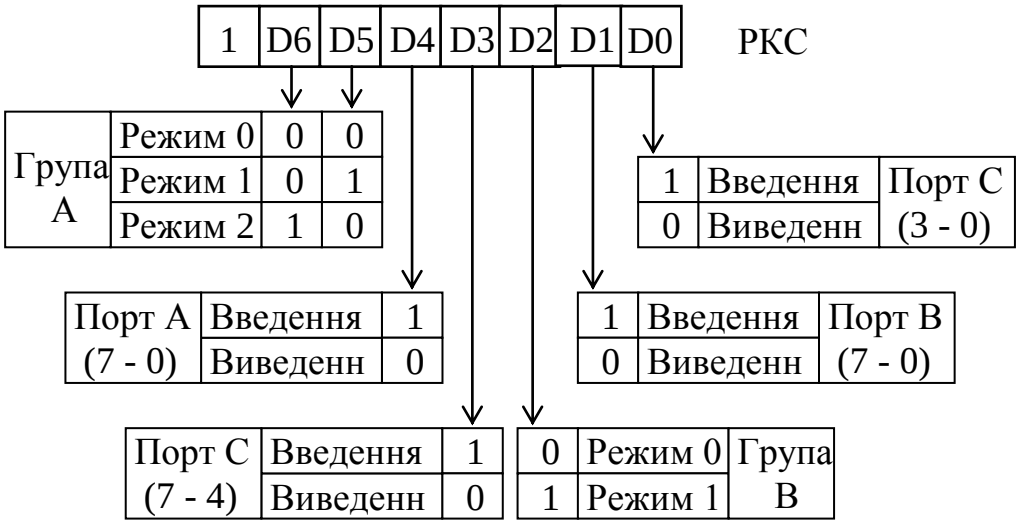

Рис. 4

Селекцію одного з трьох каналів (А, В чи С) або РКС здійснюють за послідовними адресами, які розташовані в наведеному порядку. В Мікролабі базова адреса ППІ (канала А) – F8h. Призначення окремих бітів керуючого слова наведено на рис. 4.

Робота каналів ППІ можлива в трьох режимах. У режимі 0 відбувається асинхронний обмін даними з зовнішніми пристроями через канали А, В та обидва підканали С. У режимі 1 передача даних по каналам А та В відбувається під керуванням сигналів, що формуються в каналі С, котрий використовується не як порт, а як буферний регістр для керування введенням/виведенням. У режимі 2 може працювати тільки канал А. Його лінії стають дуплексними, а керування зв'язком відбувається за допомогою каналу С (як в режимі 1).

Завдання на програмування

Створити програму лінійної структури, яка програмує ППІ для асинхронного виведення інформації в порт В та вивести на СД 8-розрядне двійкове число *А*, отримане в контрольній роботі.

Контрольні запитання

1. Що таке паралельний порт?
2. Яка команда мови асемблера застосовується для виведення даних?
3. Як запрограмувати ППІ?
4. Що таке регістр керуючого слова?
5. Як формується керуюче слово для виведення інформації в порт С?

Робота № 4

ОПЕРАЦІЇ УМОВНОГО ТА БЕЗУМОВНОГО ПЕРЕХОДУ В МОВІ АСЕМБЛЕРА

Мета роботи: дослідити структуру ознакового регістру МП та набути практичних навичок у використанні підпрограм та операцій умовного та безумовного переходу мови асемблера.

Теоретичні відомості

Команди програм лінійної структури виконуються МП послідовно, згідно з вмістом програмного лічильника PC. Якщо виникає потреба в зміні послідовного ходу виконання (організація циклів, розгалудження програми і т.і.), використовують команди переходу.

Безумовний перехід здійснюють простою зміною вмісту PC (наприклад, командою JMP). Велику програму можна спростити шляхом використання підпрограм для дій, які повторюються. Виклик підпрограми відбувається командою CALL. Під час її виконання МП завантажує в PC адресу підпрограми, а попередню – адресу повернення переписує у стек. Стек – це область ОПЗ, яка починається з адреси, що заздалегідь занесена в покажчик стека SP. МП повернется до адреси повернення після виконання команди RET – останньої команди підпрограми.

Умовний перехід здійснюється після перевірки умови переходу, аналізуючи стан одного з бітів ознакового регістру F. Команди умовного переходу наведені в табл. Д.3. Розподіл ознак регістру F по бітам зображено на рис. 5.

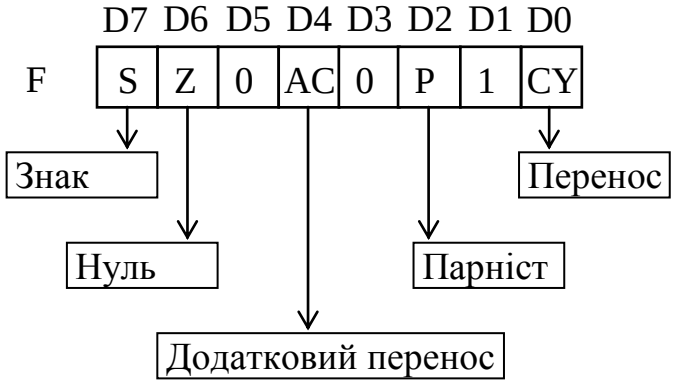

Рис. 5

Завдання на програмування

1. Створити програму почергового виведення на СД 8-розрядних двійкових чисел *A* та *B*, отриманих у контрольній роботі. Затримку між виведенням чисел *A* та *B* організувати окремою підпрограмою як

циклічний процес декрементування вмісту регістрової пари. Для збільшення тривалості затримки створити підпрограму багаторазового виклику вже існуючої підпрограми затримки.

2. Розрахувати тактову частоту Мікролаба (кількість машинних циклів для кожної команди наведено в табл. Д.3).

При спостереженні виконання програми в кроковому режимі слід знати, що монітор Мікролаба завантажує в SP число 83C7h.

Контрольні запитання

1. Які команди мови асемблера для безумовного переходу вам відомі?

2. Які команди умовного виклику підпрограми ви знаєте?

3. Що таке стек та покажчик стека?

4. Які команди роботи із стеком ви знаєте?

5. Які команди умовного повернення з підпрограми вам відомі?

Робота № 5

ВИКОРИСТАННЯ ППІ ДЛЯ ВВЕДЕННЯ ДАНИХ

Мета роботи: набути практичних навичок у програмуванні програмованого периферійного паралельного інтерфейса для введення даних та ознайомитися з операцією маскування.

Теоретичні відомості

Для імітації зовнішніх передавачів інформації Мікролаб обладнано датчиками паралельного коду Кл (див. рис. 3). Для візуалізації вихідної інформаціїї можна використовувати приймач паралельної інформації СД. Формування керуючого слова для програмування ППІ відбувається згідно з рис. 4.

Для виділення та аналізування окремих бітів слова широко застосовується операція маскування. Вона полягає в проведенні порозрядної логічної операції кон’юнкції над маскуючим словом і словом, біти якого підлягають аналізу. При цьому маскуюче слово формується таким чином: в значущі для аналізу розряди записується 1, в інші – 0.

Завдання на програмування

1. Створити програму, яка програмує ППІ для асинхронного введення інформації в порт С та визначити відповідність ключів Кл бітам порту.

2. Створити програми, що виконують операцію кон’юнкції, диз’юнкції та різноіменності над трьома вхідними сигналами. Як вхідний та вихідний пристрої, використовувати Кл та СД.

Контрольні запитання

1. Що таке асинхронний режим обміну даними?

2. Яка команда мови асемблера застосовується для введення даних ззовні?

3. Як формується керуюче слово для введення інформації з порту А?

4. Для чого використовується операція маскування?

5. Як формується маскуюче слово для аналізу стану першого біта акумулятора?

Робота № 6

ПРОГРАМНІ ПЕРЕРИВАННЯ

Мета роботи: дослідити механізм переривань у МП та навчитися зчитувати дані з клавіатури Мікролаба.

Теоретичні відомості

Інколи мікропроцесорна система повинна перервати нормальний хід виконання програми та відреагувати на непередбаченні події. Для цього МП має вхід, що приймає запит на переривання. При наявності сигналу переривання МП, без зміни вмісту PC, зчитує з шини даних команду, яку формує перериваючий пристрій. Найчастіше це команда RST, що визиває одну з восьми 8-байтних підпрограм, розташованих у перших 64 байтах пам'яті.

В Мікролабі – це область ППЗ, в якій записані команди безумовного переходу на адреси ОПЗ, де користувач має розмістити посилання на завантажені ним підпрограми обробки переривань. Адреси переходів наведені в табл. 5.

Таблиця 5

№ переривання	Адреса ППЗ	Команда переходу
RST 0	00 00 h	Використовується монітором
RST 1	00 08 h	
RST 2	00 10 h	JMP 83 D1 h
RST 3	00 18 h	JMP 83 D4 h
RST 4	00 20 h	JMP 83 D7 h
RST 5	00 28 h	JMP 83 DA h
RST 6	00 30 h	JMP 83 DD h
RST 7	00 38 h	Використовується монітором

Для повернення МП до перерваної програми наприкінці підпрограми обробки переривання має стояти команда повернення. При використанні в програмі користувача, команди RST аналогічні командам CALL для адрес ППЗ з табл. 5.

Клавіатура Мікролаба є набором ключів, організованих, як матриця 8×3. Кожен рядок з восьми ключів опитується окремо. Зчитані дані перетворюються в код, відповідний натиснутій клавіші, за допомогою підпрограми монітора KEYIN (адреса 0216h). Після повернення з цієї підпрограми в акумуляторі буде міститися код натиснутої клавіші (див.

табл. 1). Користувач може застосовувати підпрограму KEYIN для сканування клавіатури.

Завдання на програмування

Створити програму, в якій натискання вказаної викладачем клавіші Мікролаба генерує програмне переривання. Підпрограма обробки переривання повинна викликати звуковий сигнал (підпрограма BEEP, що міститься в додатковій області монітора за адресою 0350h).

Контрольні запитання

1. Що таке переривання?
2. Які команди мови асемблера керують перериваннями?
3. Що таке підпрограма обробки переривання?
4. Як в Мікролабі використовується переривання RST0?
5. Як в Мікролабі організована карта переходів за перериваннями?

Робота № 7

ДОСЛІДЖЕННЯ ПРЯМОГО ДОСТУПУ ДО ПАМ'ЯТІ

Мета роботи: дослідити роботу 8-сегментних індикаторів Мікролаба в режимі прямого доступу до пам'яті та набути практичних навичок їх застосування.

Теоретичні відомості

У Мікролабі для індикації інформації використовуються вісім 8-сегментних цифрових індикаторів. Кожному індикатору поставлена відповідна комірка мап'яті, де зберегається 8-сегментний код, що керує свіченням сегментів індикатора. Інформація з цих комірок посилається на індикатори за допомогою спеціальної схеми, яка забезпечує динамічний режим індикації. Таким чином, здійснюється прямий (тобто без участі МП) доступ до пам'яті.

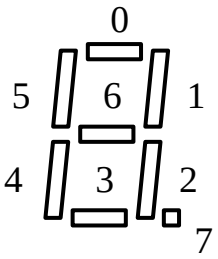

Рис. 7

Дані на індикатори передаються з восьми старших комірок ОПЗ з адресами 83F8h – 83FFh (порядок розташування індикаторів: зліва направо). 8-сегментний код формується згідно з рис. 7. На рисунку сегменти індикатора позначені номерами відповідних бітів коду. Якщо біт дорівнює 1, то відповідний сегмент буде світитися, і – навпаки. Таким чином, можна одержати на індикаторах будь-які символи з множини можливих.

Монітор Мікролаба містить підпрограму SEGCG (адреса 01C0h), що перетворює шістнадцяткові коди у 8-сегментні і розташовує їх у восьми старших комірках ОПЗ. Початкові дані для цієї підпрограми повинні знаходитися в комірках ОПЗ з адресами 83F4 – 83F7. Кожен байт з цих комірок відповідає парі індикаторів (тобто вміст комірки 83F4 після перетворення підпрограмою висвітиться на двох лівих індикаторах і т.д.).

Завдання на програмування

1. Створити програму, яка виводить на 8-сегментні індикатори Мікролаба рядок з восьми символів, наданих викладачем.

2. Створити програму, в якій натискання обраної клавіші Мікролаба викликає приріст на 1 числа, що висвітлюється на одній з пар 8-сегментних індикаторів дисплея. Процедура інкрементування числа має бути організована як підпрограма обробки переривання.

Контрольні запитання

1. Що таке прямий доступ до пам’яті?

2. Як в Мікролабі організовано прямий доступ до пам’яті?

3. Як працює 8-сегментний індикатор?

4. Як формується 8-сегментний код керування індикатором?

5. Які символи можна вивести на дисплей Мікролаба?

Список рекомендованої літератури

1. Вершинин О.Е. Применение микропроцессоров для автоматизации технологических процессов / О.Е. Вершинин. – Л. : Энергоатомиздат, 1986. – 208 с.

2. Токхайм Р. Микропроцессоры : Курс и упражнения / Р. Токхайм. – М. : Энергоатомиздат, 1988. – 336 с.

3. Хвощ С.Т. Микропроцессоры и микроЭВМ в системах автоматического управления : Справочник / С.Т. Хвощ [и др.]. – Л. : Машиностроение, 1987. – 640 с.

4. Микропроцессорная лаборатория «Микролаб КР580ИК80». – М. : Внешторгиздат, 1986. – 128 с.

5. Погорелый С.Д. Программное обеспечение микропроцессорных систем : Справочник / С.Д. Погорелый, Т.Ф. Слободянюк. – К. : Техніка, 1989. – 301 с.

6. Зубчук В.И. Справочник по цифровой схемотехнике / В.И. Зубчук [и др.]. – К. : Техніка, 1990. – 448 с.

7. Ирвин К. Язык ассемблера для процессоров Intel / Кип Р. Ирвин. – М. : Вильямс, 2005. – 912 с.

Додаток

Машинні коди команд

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	NOP	LXI *B,vv*	STAX *B*	INX *B*	INR *B*	DCR *B*	MVI *B,v*	RLC
1		LXI *D,vv*	STAX *D*	INX *D*	INR *D*	DCR *D*	MVI *D,v*	RAL
2		LXI *H,vv*	SHLD *aa*	INX *H*	INR *H*	DCR *H*	MVI *H,v*	DAA
3		LXI SP,*vv*	STA *aa*	INX SP	INR *M*	DCR *M*	MVI *M,v*	STC
4	MOV *B,B*	MOV *B,C*	MOV *B,D*	MOV *B,E*	MOV *B,H*	MOV *B,L*	MOV *B,M*	MOV *B,A*
5	MOV *D,B*	MOV *D,C*	MOV *D,D*	MOV *D,E*	MOV *D,H*	MOV *D,L*	MOV *D,M*	MOV *D,A*
6	MOV *H,B*	MOV *H,C*	MOV *H,D*	MOV *H,E*	MOV *H,H*	MOV *H,L*	MOV *H,M*	MOV *H,A*
7	MOV *M,B*	MOV *M,C*	MOV *M,D*	MOV *M,E*	MOV *M,H*	MOV *M,L*	HLT	MOV *M,A*
8	ADD *B*	ADD *C*	ADD *D*	ADD *E*	ADD *H*	ADD *L*	ADD *M*	ADD *A*
9	SUB *B*	SUB *C*	SUB *D*	SUB *E*	SUB *H*	SUB *L*	SUB *M*	SUB *A*
A	ANA *B*	ANA *C*	ANA *D*	ANA *E*	ANA *H*	ANA *L*	ANA *M*	ANA *A*
B	ORA *B*	ORA *C*	ORA *D*	ORA *E*	ORA *H*	ORA *L*	ORA *M*	ORA *A*
C	RNZ	POP *B*	JNZ *aa*	JMP *aa*	CNZ *aa*	PUSH *B*	ADI *v*	RST 0
D	RNC	POP *D*	JNC *aa*	OUT *n*	CNC *aa*	PUSH *D*	SUI *v*	RST 2
E	RPO	POP *H*	JPO *aa*	XTHL	CPO *aa*	PUSH *H*	ANI *v*	RST 4
F	RP	POP *PSW*	JP *aa*	DI	CP *aa*	PUSH *PSW*	ORI *v*	RST 6
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**

В табл. Д.1 наведені коди команд у шістнадцятковому форматі. Номер рядка тут позначає першу тетраду коду, а номер стовпця другу тетраду. Умовні позначення навндені в табл. Д.2.

процесора Intel 8080/8085 Таблиця Д.1

8	**9**	**A**	**B**	**C**	**D**	**E**	**F**	
	DAD *B*	LDAX *B*	DCX *B*	INR *C*	DCR *C*	MVI *C,v*	RRC	**0**
	DAD *D*	LDAX *D*	DCX *D*	INR *E*	DCR *E*	MVI *E,v*	RAR	**1**
	DAD *H*	LHLD *aa*	DCX *H*	INR *L*	DCR *L*	MVI *L,v*	CMA	**2**
	DAD *SP*	LDA *aa*	DCX *SP*	INR *A*	DCR *A*	MVI *A,v*	CMC	**3**
MOV *C,B*	MOV *C,C*	MOV *C,D*	MOV *C,E*	MOV *C,H*	MOV *C,L*	MOV *C,M*	MOV *C,A*	**4**
MOV *E,B*	MOV *E,C*	MOV *E,D*	MOV *E,E*	MOV *E,H*	MOV *E,L*	MOV *E,M*	MOV *E,A*	**5**
MOV *L,B*	MOV *L,C*	MOV *L,D*	MOV *L,E*	MOV *L,H*	MOV *L,L*	MOV *L,M*	MOV *L,A*	**6**
MOV *A,B*	MOV *A,C*	MOV *A,D*	MOV *A,E*	MOV *A,H*	MOV *A,L*	MOV *A,M*	MOV *A,A*	**7**
ADC *B*	ADC *C*	ADC *D*	ADC *E*	ADC *H*	ADC *L*	ADC *M*	ADC *A*	**8**
SBB *B*	SBB *C*	SBB *D*	SBB *E*	SBB *H*	SBB *L*	SBB *M*	SBB *A*	**9**
XRA *B*	XRA *C*	XRA *D*	XRA *E*	XRA *H*	XRA *L*	XRA *M*	XRA *A*	**A**
CMP *B*	CMP *C*	CMP *D*	CMP *E*	CMP *H*	CMP *L*	CMP *M*	CMP *A*	**B**
RZ	RET	JZ *aa*		CZ *aa*	CALL	ACI *v*	RST 1	**C**
RC		JC *aa*	IN *n*	CC *aa*		SBI *v*	RST 3	**D**
RPE	PCHL	JPE *aa*	XCHG	CPE *aa*		XRI *v*	RST 5	**E**
RM	SPHL	JM *aa*	EI	CM *aa*		CPI *v*	RST 7	**F**
8	**9**	**A**	**B**	**C**	**D**	**E**	**F**	

Таблиця Д.2

Познач.	**Розмір**	**Зміст**	**Познач.**	**Розмір**	**Зміст**
n	1 байт	Номер порту і/о	*vv*	2 байти	Дані
aa	2 байти	Адреса	v	1 байт	Дані

Склад команд мікропроцесора Intel 8080/8085

Таблиця Д.3

Мнемоніка	**КОП**	**Назва**	**Специфікація**	**Такти**
ADD *A*	87h	Add	Додати *A* до *A* (подвоєння *A)*	4
ADD *B*	80h		Додати *B* до *A*	
ADD *C*	81h		Додати *C* до *A*	
ADD *D*	82h		Додати *D* до *A*	
ADD *E*	83h		Додати *E* до *A*	
ADD *H*	84h		Додати *H* до *A*	
ADD *L*	85h		Додати *L* до *A*	
ADD *M*	86h		Додати вміст пам’яті LOC (*HL*) до *A*	
ADI *v*	C6h	Add Immediate	Додати безпосередньо наступні дані *v* до *A*	7
ADC *A*	8Fh	Add *vv* Carry	Додати *A* до *A* з переносом (подвоєння *A*)	4
ADC *B*	88h		Додати *B* до *A* з переносом	
ADC *C*	89h		Додати *C* до *A* з переносом	
ADC *D*	8Ah		Додати *D* до *A* з переносом	
ADC *E*	8Bh		Додати *E* до *A* з переносом	
ADC *H*	8Ch		Додати *H* до *A* з переносом	
ADC *L*	8Dh		Додати *L* до *A* з переносом	
ADC *M*	8Eh		Додати вміст пам’яті LOC (*HL*) до *A* з переносом	
ACI *v*	CEh	Add *vv* Carry Immediate	Додати безпосередньо наступні дані *v* до *A* з переносом	7
ANA *A*	A7h	AND	Тест *A* *A* AND *A*	4
ANA *B*	A0h		Логічна операція *B* AND *A*	
ANA *C*	Alh		Логічна операція *C* AND *A*	
ANA *D*	A2h		Логічна операція *D* AND *A*	
ANA *E*	A3h		Логічна операція *E* AND *A*	
ANA *H*	A4h		Логічна операція *H* AND *A*	
ANA *L*	A5h		Логічна операція *L* AND *A*	
ANA *M*	A6h		Вміст пам’яті LOC (*HL*) AND *A*	
ANI *v*	E6h	AND Immediate	Безпосередньо наступні дані *v* AND *A*	7
CALL *aa*	CDh	Call	Виклик підпрограми за адресою *aa*	17

Продовження табл. Д3

Мнемоніка	**КОП**	**Назва**	**Специфікація**	**Такти**
CZ *aa*	CCh	Call if Zero	Виклик підпрограми за адресою *aa*, якщо нуль	11/17
CNZ *aa*	C4h	Call if No Zero	Виклик підпрограми за адресою *aa*, якщо не нуль	
CP *aa*	F4h	Call if Plus	Виклик підпрограми за адресою *aa*, якщо плюс	
CM *aa*	FCh	Call if Minus	Виклик підпрограми за адресою *aa*, якщо мінус	
CC *aa*	DDh	Call if Carry	Виклик підпрограми за адресою *aa*, якщо перенос	
CNC *aa*	D4h	Call if No Carry	Виклик підпрограми за адресою *aa*, якщо не перенос	
CPE *aa*	ECh	Call if Parity Even	Виклик підпрограми за адресою *aa*, якщо парно	
CPO *aa*	E4h	Call if Parity Odd	Виклик підпрограми за адресою *aa*, якщо непарно	
CMA	2Fh	Complem-ent Acc.	Інвертувати *A*	4
CMC	3Fh	Complem-ent Carry	Інвертувати перенос	4
CMP *A* CMP *B* CMP *C* CMP *D* CMP *E* CMP *H* CMP *L* CMP *M*	BFh B8h B9h BAh BBh BCh BDh BEh	Compare	Встановити індикатор нуля операцією (*A*)-(*A*) Порівняти *A* з *B* Порівняти *A* з *C* Порівняти *A* з *D* Порівняти *A* з *E* Порівняти *A* з *H* Порівняти *A* з *L* Порівняти *A* з вмістом пам’яті LOC (*HL*)	4
CPI *v*	FEh	Compare Immediate	Порівняти *A* з наступними даними безпосередньо	7
DAA	27h	Decimal Adjust Acc.	Десяткова корекція акумулятора	4
DAD *B* DAD *D* DAD *H* DAD *SP*	09h 19h 29h 39h	Double Acception Data	Додати *BC* до *HL* Додати *DE* до *HL* Додати *HL* до *HL* (подвоєння *H*) Додати *SP* до *H*	3

Продовження табл. Д3

Мнемоніка	КОП	Назва	Специфікація	Такти
DCR *A*	3Dh	Decrement	Декрементувати *A*	5
DCR *B*	05h		Декрементувати *B*	
DCR *C*	0Dh		Декрементувати *C*	
DCR *D*	15h		Декрементувати *D*	
DCR *E*	1Ch		Декрементувати *E*	
DCR *H*	25h		Декрементувати *H*	
DCR *L*	2Dh		Декрементувати *L*	
DCR *M*	35h		Декрементувати вміст пам’яті LOG (*HL*)	
DCX *B*	0Bh	Decrement Extended	Декрементувати *BC*	5
DCX *D*	1Bh		Декрементувати *DE*	
DCX *H*	2Bh		Декрементувати *HL*	
DCX *SP*	3Bh		Декрементувати *SP*	
DI	F3h	Disable Interrupt	Заборонити переривання	4
EI	FBh	Enable Interrupt	Дозволити переривання	4
HLT	76h	Halt	Зупинити мікропроцесор	7
IN *n*	DBh	In	Ввести дані з пристрою *n*	10
INR *A*	3Ch	Increment	Інкрементувати *A*	5
INR *B*	04h		Інкрементувати *B*	
INR *C*	0Ch		Інкрементувати *C*	
INR *D*	14h		Інкрементувати *D*	
INR *E*	1Ch		Інкрементувати *E*	
INR *H*	24h		Інкрементувати *H*	
INR *L*	2Ch		Інкрементувати *L*	
INR *M*	34h		Інкрементувати вміст пам’яті LOC(*HL*)	
INX *B*	03h	Increment Extended	Інкрементувати *BC*	5
INX *D*	13h		Інкрементувати *DE*	
INX *H*	23h		Інкрементувати *HL*	
INX *SP*	33h		Інкрементувати *SP*	
JMP *aa*	C3h	Jump	Перейти на адресу *aa*	10
JZ *aa*	CAh	Jump if Zero	Перейти на адресу *aa,* якщо нуль	11/17

Продовження табл. Д3

Мнемоніка	**КОП**	**Назва**	**Специфікація**	**Такти**
JNZ *aa*	C2h	Jump if No Zero	Перейти на адресу *aa,* якщо не нуль	11/17
JP *aa*	F2h	Jump if Plus	Перейти на адресу *aa,* якщо плюс	
JM *aa*	FAh	Jump if Minus	Перейти на адресу *aa,* якщо мінус	
JC *aa*	DAh	Jump if Carry	Перейти на адресу *aa*, якщо перенос	
JNC *aa*	D2h	Jump if No Carry	Перейти на адресу *aa*, якщо не перенос	
JPE *aa*	EAh	Jump if Parity Even	Перейти на адресу *aa*, якщо паритет парний	
JPO *aa*	E2h	Jump if Parity Odd	Перейти на адресу *aa*, якщо паритет непарний	
LDA *aa*	3Ah	Load Acc.	Завантажити *A* із комірки пам’яті з адресою *aa*	13
LDAX *B*	0Ah	Load Acc. Extended	Завантажити *A* із комірки пам’яті LOC (*BC*)	7
LDAX *D*	1Ah		Завантажити *A* із комірки пам’яті LOC (*DE*)	
LHLD *aa*	2Ah	Load *HL* Direct	Завантажити *HL* із комірки пам’яті з адресою *aa*	16
LXI *B, vv*	01h	Load Extended Immediate	Завантажити *BC* безпосередньо наступними даними	10
LXI *H, vv*	21h		Завантажити *HL* безпосередньо наступними даними	
LXI *SP, vv*	31h		Завантажити *SP* безпосередньо наступними даними	
MOV *A, B*	78h	Move	Переслати дані із *B* в *A*	5
MOV *A, C*	79h		Переслати дані із *C* в *A*	
MOV *A, D*	7Ah		Переслати дані із *D* в *A*	
MOV *A, E*	7Bh		Переслати дані із *E* в *A*	
MOV *A, H*	7Ch		Переслати дані із *H* в *A*	
MOV *A, L*	7Dh		Переслати дані із *L* в *A*	
MOV *A, M*	7Eh		Переслати дані із комірки пам’яті LOC (*HL*) в *A*	
MOV *B, A*	47h		Переслати дані із *A* в *B*	

Продовження табл. Д3

Мнемоніка	**КОП**	**Назва**	**Специфікація**	**Такти**
MOV *B*, C	41h	Move	Переслати дані із *C* в *B*	5
MOV *B, D*	42h		Переслати дані із *D* в *B*	
MOV *B, E*	43h		Переслати дані із *E* в *B*	
MOV *B, H*	44h		Переслати дані із *H* в *B*	
MOV *B, L*	45h		Переслати дані із *L* в *B*	
MOV *B, M*	46h		Переслати дані із комірки пам'яті LOC (*HL*) в *B*	
MOV *C, A*	4Fh		Переслати дані із *A* в *C*	
MOV *C, B*	48h		Переслати дані із *B* в *C*	
MOV *C, D*	4Ah		Переслати дані із *D* в *C*	
MOV *C, E*	4Bh		Переслати дані із *E* в *C*	
MOV *C, H*	4Ch		Переслати дані із *H* в *C*	
MOV *C, L*	4Dh		Переслати дані із *L* в *C*	
MOV *C, M*	4Eh		Переслати дані із комірки пам'яті LOC (HL) в *C*	
MOV *D, A*	57h		Переслати дані із *A* в *D*	
MOV *D, B*	50h		Переслати дані із *B* в *D*	
MOV *D, C*	51h		Переслати дані із *C* в *D*	
MOV *D, E*	53h		Переслати дані із *E* в *D*	
MOV *D, H*	54h		Переслати дані із *H* в *D*	
MOV *D, L*	55h		Переслати дані із *L* в *D*	
MOV *D, M*	56h		Переслати дані із комірки пам'яті LOC (*HL*) в *D*	
MOV *E, A*	5Fh		Переслати дані із *A* в *E*	
MOV *E, B*	5Fh		Переслати дані із *B* в *E*	
MOV *E, C*	58h		Переслати дані із *C* в *E*	
MOV *E, D*	59h		Переслати дані із *D* в *E*	
MOV *E, H*	5Ah		Переслати дані із *H* в *E*	
MOV *E, L*	5Ch		Переслати дані із *L* в *E*	
MOV *E, M*	5Dh		Переслати дані із комірки пам'яті LOC (*HL*) в *E*	
MOV *H, A*	67h		Переслати дані із *A* в *H*	
MOV *H, B*	60h		Переслати дані із *B* в *H*	
MOV *H, C*	61h		Переслати дані із *C* в *H*	
MOV *H, D*	62h		Переслати дані із *D* в *H*	
MOV *H, E*	63h		Переслати дані із *E* в *H*	
MOV *H, L*	65h		Переслати дані із *L* в *H*	

Продовження табл. Д3

Мнемоніка	КОП	Назва	Специфікація	Такти
MOV *H*, *M*	66h	Move	Переслати дані із комірки пам'яті LOC (*HL*) в *H*	5
MOV *L*, *A*	6Fh		Переслати дані із *A* в *L*	
MOV *L*, *B*	68h		Переслати дані із *B* в *L*	
MOV *L*, *C*	69h		Переслати дані із *C* в *L*	
MOV *L*, *D*	6Ah		Переслати дані із *D* в *L*	
MOV *L*, *E*	6Bh		Переслати дані із *E* в *L*	
MOV *L*, *H*	6Ch		Переслати дані із *H* в *L*	
MOV *L*, *M*	6Eh		Переслати дані із комірки пам'яті LOC (*HL*) в *L*	
MOV *M, A*	77h		Переслати дані в комірку пам'яті LOC (*HL*) із *A*	
MOV *M, B*	70h		Переслати дані в комірку пам'яті LOC (*HL*) із *B*	
MOV *M, C*	71h		Переслати дані в комірку пам'яті LOC (*HL*) із *C*	
MOV *M, D*	72h		Переслати дані в комірку пам'яті LOC (*HL*) із *D*	
MOV *M, E*	73h		Переслати дані в комірку пам'яті LOC (*HL*) із *E*	
MOV *M, H*	74h		Переслати дані в комірку пам'яті LOC (*HL*) із *H*	
MOV *M, L*	75h		Переслати дані в комірку пам'яті LOC (*HL*) із *L*	
MVI *A*, *v*	3Eh	Move Immediate	Переслати безпосередньо наступні дані *v* в *A*	7
MVI *B, v*	06h		Переслати безпосередньо наступні дані *v* в *B*	
MVI *C*, *v*	0Eh		Переслати безпосередньо наступні дані *v* в *C*	
MVI *D*, *v*	16h		Переслати безпосередньо наступні дані *v* в *D*	
MVI *E, v*	1Eh		Переслати безпосередньо наступні дані *v* в *E*	
MVI *H*, *v*	26h		Переслати безпосередньо наступні дані *v* в *H*	
MVI *L*, *v*	2Eh		Переслати безпосередньо наступні дані *v* в *L*	

Продовження табл. Д3

Мнемоніка	**КОП**	**Назва**	**Специфікація**	**Такти**
MVI *M, v*	36h	Move Immediate	Переслати безпосередньо наступні дані *v* в LOC (*HL*)	
NOP	00h	No Operation	Немає операції	4
ORA *A*	B7h	OR	Перевірити *A* та скинути перенос	4
ORA *B*	B0h		Логічна операція *B* OR *A*	
ORA *C*	B1h		Логічна операція *C* OR *A*	
ORA *D*	B2h		Логічна операція *D* OR *A*	
ORA *E*	B3h		Логічна операція *E* OR *A*	
ORA *H*	B4h		Логічна операція *H* OR *A*	
ORA *L*	B5h		Логічна операція *L* OR *A*	
ORA *M*	B6h		Вміст комірки пам’яті LOG (*HL*) OR *A*	
ORI *v*	F6h	OR Immediate	Безпосередньо наступні дані *v* OR *A*	7
OUT *n*	D3h	Out	Вивести вміст *A* на адресу *n*	10
PCHL	E9h	*PC HL*	Переслати вміст *H* та *L* в *PC* (лічильник команд)	5
POP *B*	Clh	Pull up	Добути із стека вміст пари регістрів *BC*	10
POP *D*	Dlh		Добути із стека вміст пари регістрів *DE*	
POP *H*	Elh		Добути із стека вміст пари регістрів *HL*	
POP *PSW*	Flh		Добути із стека вміст пари регістрів *PSW*	
PUSH *B*	C5h	Push	Завантажити в стек вміст пари регістрів *BC*	11
PUSH *D*	D5h		Завантажити в стек вміст пари регістрів *DE*	
PUSH *H*	E5h		Завантажити в стек вміст пари регістрів *HL*	
PUSH *PSW*	F5h		Завантажити в стек вміст пари регістрів *PSW*	
RAL	17h	Rotate Arithmetic Left	Переместити циклічно *CY+A* вліво	4

Продовження табл. Д3

Мнемоніка	КОП	Назва	Специфікація	Такти
RAR	1Fh	Rotate Arithmetic Right	Переместити циклічно *CY+A* вправо	4
RLC	07h	Rotate Left	Переместити *A* вліво з переносом	4
RRC	0Fh	Rotate Right	Переместити *A* вправо з переносом	4
RET	C9h	Return	Повернення з підпрограми	10
RZ	C8h	Return if Zero	Повернення з підпрограми, якщо нуль	5/11
RNZ	C0h	Return if No Zero	Повернення з підпрограми, якщо не нуль	
RP	F0h	Return if Plus	Повернення з підпрограми, якщо плюс	
RM	F8h	Return if Minus	Повернення з підпрограми, якщо минус	
RC	D8h	Return if Carry	Повернення з підпрограми, якщо перенос	
RNC	D0h	Return if No Carry	Повернення з підпрограми, якщо немає переносу	
RPE	E8h	Return if Parity Even	Повернення з підпрограми, якщо парний паритет	
RPO	E0h	Return if Parity Odd	Повернення з підпрограми, якщо непарний паритет	
RST 0	C7h	Restart	Повторний запуск програми з адреси 00h	4
RST 1	CFh		Повторний запуск програми з адреси 08h	
RST 2	D7h		Повторний запуск програми з адреси 10h	
RST 3	DFh		Повторний запуск програми з адреси 18h	
RST 4	E7h		Повторний запуск програми з адреси 20h	
RST 5	EFh		Повторний запуск програми з адреси 28h	
RST 6	F7h		Повторний запуск програми з адреси 30h	
RST 7	FFh		Повторний запуск програми з адреси 28h	

Продовження табл. Д3

Мнемоніка	**КОП**	**Назва**	**Специфікація**	**Такти**
SPHL	F9h	*SP HL*	Завантажити *SP* із *HL*	5
SHLD *aa*	22h	Store *HL* Direct	Розмістити *HL* в пам'яті з адресою *aa*	16
STA *aa*	32h	Store Acc.	Розмістити *A* в пам'ять на адресу *aa*	13
STAX *B*	02h	Store Acc. Extended	Розмістити *A* в пам'ять LOC (*BC*)	7
STAX *D*	12h		Розмістити *A* в пам'ять LOC (*DE*)	
STC	37h	Set Carry	Встановити індикатор переносу	4
SUB *A*	97h	Subtract	Відняти *A* із *A* (очистити акумулятор)	4
SUB *B*	90h		Відняти *B* із *A*	
SUB *C*	91h		Відняти *C* із *A*	
SUB *D*	92h		Відняти *D* із *A*	
SUB *F*	93h		Відняти *E* із *A*	
SUB *H*	94h		Відняти *H* із *A*	
SUB *L*	95h		Відняти *L* із *A*	
SUB *M*	96h		Відняти вміст пам'яті LOG (*HL*) із *A*	
SUI *v*	D6h	Subtract Immediate	Відняти безпосередньо наступні дані *v* із *A*	7
SBB *A*	9Fh	Subtract – Borrow	Відняти *A* із *A* (очистити акумулятор)	4
SBB *B*	98h		Відняти з позикою *B* із *A*	
SBB *C*	99h		Відняти з позикою *C* із *A*	
SBB *D*	9Ah		Відняти з позикою *D* із *A*	
SBB *E*	9Bh		Відняти з позикою *E* із *A*	
SBB *H*	9Ch		Відняти з позикою *H* із *A*	
SBB *L*	9Dh		Відняти з позикою *L* із *A*	
SBB *M*	9Eh		Відняти з позикою вміст пам'яті LOC (*HL*) із *A*	
SBI *v*	DEh	Subtract – Borrow Immediate	Відняти з позикою безпосередні дані *v* із *A*	7
XCHG	EBh	Exchange	Обмін вмістом пар регістрів *DE* і *HL*	4

Продовження табл. Д3

Мнемоніка	**КОП**	**Назва**	**Специфікація**	**Такти**
XTHL	E3h	Exchange *HL*	Обмін вершини стека з вмістом пари регістрів *HL*	18
XRA *A*	AFh	Exclusive OR	Логічна операція *A* XOR *A* (очищення *A*)	4
XRA *B*	A8h		Логічна операція *B* XOR *A*	
XRA *C*	A9h		Логічна операція *C* XOR *A*	
XRA *D*	AAh		Логічна операція *D* XOR *A*	
XRA *E*	ABh		Логічна операція *E* XOR *A*	
XRA *H*	ACh	Exclusive OR	Логічна операція *H* XOR *A*	4
XRA *L*	ADh		Логічна операція *L* XOR *A*	
XRA *M*	AEh		Вміст пам’яті LOC (*HL*) XOR *A*	
XRI *v*	EEh	Exclusive OR Immediate	Безпосередні дані *v* XOR *A*	7

Еквіваленти чисел у шістнадцятковій, десятковій, вісімковій та двійковій системах числення

Таблиця Д.4

16	**10**	**8**	**2**
0	0	0	0000
1	1	1	0001
2	2	2	0010
3	3	3	0011
4	4	4	0100
5	5	5	0101
6	6	6	0110
7	7	7	0111
8	8	10	1000
9	9	11	1001
A	10	12	1010
B	11	13	1011
C	12	14	1100
D	13	15	1101
E	14	16	1110
F	15	17	1111
10	16	20	10000