МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

В. Г. Городецький, М. П. Осадчук

МІКРОПРОЦЕСОРНІ ПРИСТРОЇ
ЛАБОРАТОРНИЙ ПРАКТИКУМ

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра за освітньою програмою
«Електромеханічні та мехатронні системи енергоємних виробництв»

Київ
КПІ ім. Ігоря Сікорського
2019

Рецензент *Лебедєв Л. М.*, канд. техн. наук, доц. кафедри автоматизації управління електротехнічними комплексами

Відповідальний редактор *Лістовщик Л. К.*, канд. техн. наук, доц. кафедри електромеханічного обладнання енергоємних виробництв

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (Протокол № 7 від 01.04.2019 р.) за поданням Вченої ради ІЕЕ (протокол № 9 від 25.02.2019 р.)

Електронне мережне навчальне видання

Городецький Віктор Георгійович, канд. фіз.-мат. наук, доц.
Осадчук Микола Павлович, канд. фіз.-мат. наук, ас.

МІКРОПРОЦЕСОРНІ ПРИСТРОЇ

ЛАБОРАТОРНИЙ ПРАКТИКУМ

Мікропроцесорні пристрої: Лабораторний практикум [Електронний ресурс]: навч. посіб. для здобувачів ступеня бакалавра за освітньою програмою «Електромеханічні та мехатронні системи енергоємних виробництв» / В. Г. Городецький, М.П. Осадчук; КПІ ім. Ігоря Сікорського. – Електронні текстові данні (1 файл: 0.541 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2019. – 45 с.

У представленому посібнику викладено основні положення щодо виконання лабораторних робіт з курсу «Мікропроцесорні пристрої». Мета циклу робіт – засвоєння на практиці теоретичних знань, отриманих на лекціях. Навчальне видання містить основні теоретичні відомості, програму роботи, вказівки для її виконання та контрольні запитання.

Розглянуто основні принципи побудови мікропроцесорних систем, призначення та взаємодію основних вузлів мікропроцесора. Викладаються основи програмування мікропроцесорних систем з використанням мови програмування низького рівня - Асемблер.

Навчальне видання призначене для здобувачів ступеня бакалавра за спеціальністю 141 «Електроенергетика, електротехніка та електромеханіка», освітньою програмою «Електромеханічні та мехатронні системи енергоємних виробництв».

Вступ

Даний курс лабораторних робіт призначено для вивчення елементів апаратного та програмного забезпечення комп'ютерних систем керування, які побудовані на базі мікропроцесорів фірми Intel. Особлива увага приділяється основам складання програм на мові Асемблер, що є однією із складових наскрізної комп'ютерної підготовки студентів спеціальності 141 «Електроенергетика, електротехніка та електромеханіка», освітньою програми «Електромеханічні та мехатронні системи енергоємних виробництв».

Розглянуто методи та особливості підготовки програм з оглядом на архітектуру конкретного мікропроцесора. Через це в практикумі приділяється значна увага основним методам адресації, які використовує дана мікропроцесорна система.

Лабораторні роботи організовані таким чином, що їх виконання відбувається після проходження відповідних тем на лекціях, що сприяє кращому розумінню та засвоєнню матеріалу.

При підготовці до роботи студент зобов'язаний засвоїти теорію згідно інструкції та лекції з відповідної теми. Крім конкретних вимог до виконання та оформлення звіту до кожної роботи студент повинен заповнювати словник з новими командами асемблера, які були використані в даній роботі. Словник створюється на окремих сторінках конспекту лекцій і надається викладачу під час захисту кожної роботи. Цей же словник в кінці семестру є однією з вимог допущення до заліку.

Після виконання роботи студент повинен захистити її на цьому ж або на наступному занятті. Своєчасні відпрацювання та захист лабораторних робіт заохочуються рейтинговою системою оцінювання.

ЛАБОРАТОРНА РОБОТА № 1

Дослідження мікропроцесорної системи "Мікролаб"

Мета роботи - вивчити будову мікролабораторії, її основні характеристики, вивчити розподіл пам'яті в "Мікролаб", призначення портів введення - виведення, отримати основні навички роботи з мікролабораторією.

КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Мікролабораторія - мікро-ЕОМ, розроблена спеціально для цілей навчання. Вона використовує МП (мікропроцесор) КР580ИК80, ПЗП (постійний запам’ятовуючий пристрій) ємністю 1 К і ОЗП (оперативний запам’ятовуючий пристрій) ємністю 1 К. У мікро-ЕОМ є клавіатура, з якої можна вводити програми, дані для зберігання, подавати команди для управління роботою мікрокомп'ютера і дисплея, що дозволяє спостерігати вміст пам'яті і регістрів, вісім світлодіодних індикаторів і три тумблера, пов'язаних з МП за допомогою порту введення/виводу. Генерація звуків також проводиться процесором. ПЗП містить програми для зчитування з клавіатури, виконання команд і виведення даних на дисплей, тобто вся робота системи управляється програмою монітора, записаною в ПЗП.

Зберігання даних в пам'яті, перевірка вмісту пам'яті і виконання програм - основні функції мікролабораторії. Програми можуть виконуватися безперервно або в крокovому режимі, що дозволяє простежити їх роботу докладно.

В табл.1 наведена карта пам'яті мікролабораторії. На ній показано розподіл адрес пам'яті для кожного пристрою. ПЗП резервує адреси з 0000 по 03FF. Монітор (службова програма мікролабораторії) використовує адреси з 0000 по 02FF, а 0300-03FF (265 байт) можуть бути використані для розширення можливостей монітора. Область з 0400 до 05FF також може бути використана

для запису додаткових програм. Слід тільки пам'ятати, що програма в ці області може бути записана тільки програмуванням ПЗП. Програма, яка займає область адрес з 0400 до 05FF, записується (шляхом програмування) в ПЗП, що знаходиться в адаптері.

ОЗП резервує адреси з 8000 по 83FF. Зауважимо, однак, що не всі комірки ОЗП доступні для програм користувача. Адреси з 83C7 до 83FF використовуються програмою монітора і не можуть бути використані програмами користувача. Область користувача займає адреси з 8000 по 83C6.

Таблиця 1. Карта пам’яті «Мікролаб»

Адреси	Ємність пам’яті	ПЗП/ОЗП	Використання
FFFF | 8400	31К		Область що не використовується
83FF | 83C7	57	ОЗП	Робоча область монітора
83C6 | 8000	967	ОЗП	Область користувача
7FFF | 0600	30,5К		Область, що не використовується
05FF | 0400	512	ПЗП	Область користувача
03FF | 0300	256	ПЗП	Додаткова область монітора
02FF | 0000	768	ПЗП	Область монітора

ПЕРЕВІРКА ПАМ'ЯТІ, ЗАПИС І ЗБЕРЕЖЕННЯ ДАНИХ

Проведемо експеримент з перевірки вмісту пам'яті.

1. Натисніть кнопку СКИДАННЯ.

2. Натисніть кнопки 8, 0, 0, 0 і УСТ.АД. Тепер чотири лівих цифри дисплея показують адресу, яку щойно ввели, а дві крайні праві цифри показують дані, що зберігаються за цією адресою. Адреси і дані, що висвічуються на дисплеї, представлені в шістнадцятковому коді.

3. Натисніть кнопку АД.+. На лівих індикаторах буде спостерігатися прирощення адреси, а на двох крайніх правих індикаторах висвітиться вміст наступної комірки пам'яті. Багаторазово натискаючи кнопку АД.+, можна переглянути вміст пам'яті.

4. Натисніть кнопку АД.-. На лівих індикаторах станеться зменшення адреси на 1, а на двох крайніх правих індикаторах висвітиться вміст комірки пам'яті, що відповідають цій адресі. Таким чином, натискаючи кнопку АД.-, можна переглянути вміст пам'яті в зворотному порядку.

Проведемо експеримент з запису даних в пам'ять "Мікролаб".

1. Натисніть кнопки 8, 0, 0, 0 і УСТ.АД. Це визначить адресу 8000, яка є початковою адресою ОЗП.

2. Натисніть кнопки 0, 0, ЗП, тобто запишіть 00 в комірку пам'яті з адресою 8000. Сталося прирощення адреси, і на лівих індикаторах видно адресу 8001. На двох крайніх правих індикаторах тепер видно вміст комірки з адресою 8001, а те, що було записано в комірку з адресою 8000, змістилося на два індикатори лівіше.

3. Натисніть кнопки С, 3 і ЗП, тобто запишіть дані СЗ за адресою 8001. Відбувається прирощення адреси.

4. Запишіть в наступну комірку з адресою 8002 дані 00, а в комірку з адресою 8003 - 80.

5. Натисніть кнопку АД.-. Це зменшить на одиницю адресу, і на двох крайніх правих індикаторах з'являться дані, які щойно ввели.

6. Повторіть крок 5, доки не перевірите, що весь введений вами лістинг міститься в пам'яті.

Можна також перевірити дані, натиснувши кнопки 8, 0, 0, 0 і УСТ.АД. і потім використовуючи кнопку АД.+ для перевірки послідовних комірок.

Проведемо експеримент з виправлення помилок.

1. Натисніть кнопки 0, 0, 0, 1 і УСТ.АД. - Вибрали першу адресу ПЗУ.

2. Натисніть кнопки 0, 0 і ЗП. Це означає, що ви намагаєтеся записати нулі у вибрану комірку, а ця комірка знаходиться в ПЗУ і дані туди не запишуться.

3. Щоб переконатися в цьому, перевірте вміст комірки з адресою 0001. Для цього натисніть кнопку АД.-. Ви побачите, що в комірці з адресою 0001 записано 92, а не 00.

Таким чином, ваша помилкова дія не була сприйнята.

Тепер розглянемо, як бути з такими помилками, як введення неправильних адрес чи даних. Проведемо наступний експеримент.

1. Натисніть кнопки 0, 9, 0. Припустимо, що в цей момент ви згадали, що адреса, яку ви хотіли записати, 8020. І, хоча ви вже почали вводити адресу, помилку можна виправити, натиснувши кнопки 8, 0, 2, 0 і УСТ.АД.

2. Тепер, коли ви ввели правильну адресу, можна ввести потрібні дані. Натисніть кнопки 7, 9. Ці дані з'являться на правій стороні дисплея.

3. Припустимо, що це не ті дані, які ви хотіли ввести: вам потрібно було ввести 6 9. Тоді натисніть кнопки 6, 9 і введіть правильні дані, а неправильні змістяться лівіше і будуть загублені. Можна вводити дані безперервно, але "Мікролаб" зберігає тільки дві останні цифри. Після цього, якщо натиснути кнопку ЗП, дані запишуться в пам'ять.

4. Натисніть кнопку ЗП. Тепер записали дані 69 за адресою 8020.

5. Припустимо, що дані треба змінити на 68. Натисніть кнопку АД.-. На дисплеї з'явиться адреса 8020 і дані 69. Наберіть потрібні дані (68) і натисніть кнопку ЗП. Таким чином дані змінені.

ПРОСТА ПРОГРАМА

Дані, записані в ОЗП, наведені в табл. 2, є реальною програмою. Код 00, що зберігається за адресою 8000, має мнемонічне позначення NOP, яке визначає відсутність операції. Ця команда, як вказує її ім'я, нічого не робить (не виконується ніяких дій). Однак вона може бути використана в якості короткої часової затримки. Але найбільш важливо використання її в якості просторового заповнювача. Якщо пізніше треба повернутися назад і ввести нові команди в програму, яка вже написана. Можна замінити команди NOP потрібними командами. Якщо ви не помістили команди NOP в програму, вам доведеться зсовувати великий шматок програми для того, щоб помістити одну додаткову команду.

Таблиця 2. Дані, записані в ОЗП

Адреса	Вміст
8000	00
8001	C3
8002	00
8003	80

Інша команда програми - перехід (JMP - мнемонічне позначення). Код операції - C3. Ця команда означає перехід до адреси, вказаної в наступних двох байтах пам'яті. У першому байті пам’яті міститься молодший адресний байт, а в другому - старший адресний байт. Таким чином, адреса переходу 8000 зберігається наступним чином: 00 - в комірці 8002 і 80 - у комірці 8003.

Ця програма, звичайно, нічого не робить, але вона забезпечує нескінченний цикл. Структурна схема програми показана на рис. 1.

Проведемо експеримент з виконання програми користувача в автоматичному і кроковому режимах.

1. Переключіть тумблер режиму в положення КРОК. Програма виконуватиметься у кроковому режимі.

2. Введіть програму, вказану в табл. 2.

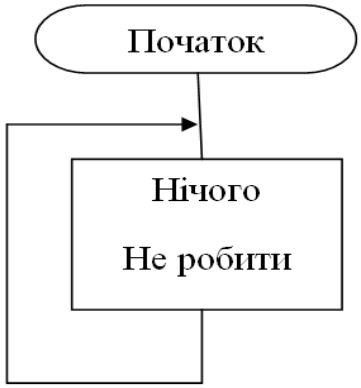

Рис.1. Структурна схема програми "нічого не робити"

3. Встановіть початкову адресу програми 8000.

4. Натисніть кнопку ПУСК. Команда, що показана на дисплеї, виконається, і дисплей покаже наступну команду. Зауважимо, що, хоча може здатися, що кнопка ПУСК (в кроковому режимі) працює так само, як кнопка АД.+, їх функції зовсім різні. Коли використовуєте кнопку АД.+, ви просто перевіряєте вміст пам'яті. Натискаючи кнопку ПУСК (в кроковому режимі), ви інтерпретуєте вміст комірки пам'яті, висвітлений на дисплеї, як команду, яка виконується МП.

Перед тим, як "Мікролаб" виконає команду, дисплей повинен показувати комірку, яка вміщує код операції. Комірка, з якої починається виконання програми, не повинна містити дані або адресу. Наприклад програма (табл. 1) може починатися з адрес 8000 або 8001, але не з адрес 8002 або 8003.

5. Тепер на дисплеї показана команда переходу СЗ. Натисніть кнопку ПУСК. Через те, що це команда переходу, програма перейде до адреси 8000. Зауважимо, що комірки пам'яті, які містять адресу переходу, в цьому разі не висвічуються, тому що, коли ви натискаєте кнопку ПУСК (в кроковому

режимі), "Мікролаб" виконує цілком всю команду, включаючи зчитування двох байтів адреси переходу.

6. Натисніть кнопку ПУСК ще два рази, щоб цикл повторився.

7. Переключіть тумблер режиму в положення АВТ. Тепер програма буде виконуватися в автоматичному режимі.

8. Встановіть початкову адресу 8000 й натисніть кнопку ПУСК. Програма виконується (безперервно) на максимальній швидкості (приблизно 3 мкс на команду). На дисплеї нічого не змінюється, через те, що при виконані даної програми (табл. 2) не передбачено виведення даних на індикацію. (Як і якими командами відбувається виведення даних на індикацію, буде розглянуто нижче). "Мікролаб" не реагує також при виконанні даної програми на натиснення всіх кнопок, крім кнопки СКИДАННЯ. Це відбувається внаслідок того, що в даній програмі не передбачено опитування клавіатури. А з програми монітора, в якій таке опитування передбачено, ми вийшли в момент натискання кнопки ПУСК.

9. Натисніть кнопку СКИДАННЯ. Задана програма припинить виконуватися, і почне виконуватися програма монітора. На індикаторах висвітяться всі нулі, і "Мікролаб" буде готовий до введення даних.

На відміну від всіх інших кнопок, кнопка СКИДАННЯ не опитується програмним способом, а сигнал з неї заведений на вхід СКИДАННЯ МП.

ПОРТИ ВВЕДЕННЯ / ВИВЕДЕННЯ

"Мікролаб" має порт введення/виведення, що реалізований на програмованому периферійному інтерфейсі (мікросхема КР580ВВ55). Порти цього інтерфейсу можуть бути або портами виводу, або портами введення в залежності від того, як інтерфейс запрограмований. До трьох розрядів порту С цього інтерфейсу підключені тумблери. Цей порт буде програмуватися як порт введення. А до 8 розрядів порту В підключені світлодіодні індикатори. Цей

порт буде програмуватися як порт виводу, тобто МП зможе зчитувати дані з тумблерів і виводити данні на світлодіодні індикатори.

На рис. 2, показана структурна схема програми, яка демонструє використання цих портів. Ця програма зчитує дані з порту введення та записує їх в порт виводу.

Лістинг програми наведено в табл.3.

Рис. 2. Структурна схема програми для перепису даних з порту введення в порт виводу

Щоб запрограмувати інтерфейс, як було передбачено (С - порт введення, В - порт виводу), необхідно подати на нього код 81 за адресою FB. Перша команда MVI A, 81 означає завантаження акумулятора кодом 81. Наступна команда OUT записує вміст акумулятора в порт за адресою FB, тобто порт (інтерфейс) програмується. Далі йде команда IN FA, яка забезпечує запис даних з порту з адресою FA, привласненою порту С, в акумулятор. Команда OUT F9 означає виведення даних з акумулятора, в порт виводу за адресою F9. Оскільки ця адреса в нашому випадку присвоєна порту В, відбувається виведення даних з

акумулятора в порт В (на світлодіодні індикатори). Таким чином, програма пересилає дані з порту введення в порт виводу. Остання команда - перехід (JMP). Вона завершує цикл, який виконується безперервно. Отже, дані вихідного порту будуть відповідати даним порту вводу під час виконання програми. МП КР580ИК80 може адресувати до 256 портів введення і стільки ж портів виводу. Тож для адресації портів достатньо два шістнадцяткових розряди, а не чотири, як це потрібно для пам'яті.

Таблиця 3. Лістинг програми для перепису даних з порту введення в порт виводу

Адреса	Вміст	Мітка	Команда	Коментар
8000	3E		MVI A, 81	Запис в акумулятор даних
8001	81			для програмування інтерфейсу
8002	D3		OUT FB	Програмування інтерфейсу
8003	FB			
8004	DB	START:	IN FA	Читання даних з порту вводу
8005	FA			
8006	D3		OUT F9	Запис даних в порт виводу
8007	F9			
8008	C3		JMP START	Перехід на початок циклу
8009	04			
800A	80			

Проведемо експеримент по виконанню програми для передачі даних з порту введення в порт виводу.

1. Введіть програму, наведену в табл.3.

2. Почніть виконання програми з адреси 8000.

3. Поставте тумблер порту вводу в будь-яке положення. Верхнє відповідає 1, нижнє - 0.

4. Подивіться на світлодіоди 2-4, рахуючи зправа. Вони показують ті ж дані, які встановлені на тумблерах порту введення.

5. Змініть положення тумблерів порту введення. Дані порту виводу повинні змінитися відповідно.

6. Натисніть кнопку СКИДАННЯ. Програма зупиниться, управління перейде до монітора.

7. Змініть положення тумблерів порту введення. Дані порту виводу не зміняться, оскільки введена програма більше не виконується.

Слід відмітити, що при бажанні, перед записом даних у порт виводу, можна зробити над ними будь-які дії, передбачені системою команд МП. Наприклад, можна зчитати дані з порту введення, інвертувати їх і ввести в порт виводу. Для цього в програму (табл. 3) необхідно додати команду, що забезпечує інвертування даних (СМА).

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1. На які області ділиться пам'ять Мікролаба?
2. Яка різниця між командами АД+ та ПУСК у покроковому режимі?
3. Поясніть, скільки команд і які занесені в табл. 2.
4. Поясніть всі команди, які використані в цій роботі.

ЛАБОРАТОРНА РОБОТА №2

Вивчення регістрів мікропроцесора

Мета роботи - вивчити регістри загального призначення і спеціальні регістри МП, отримати навички використання зазначених регістрів при програмуванні.

КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

МП КР580ИК80 містить ряд внутрішніх регістрів, частина з яких може використовуватися для зберігання і обробки даних. Ці регістри відрізняються від регулярних комірок пам'яті тим, що вони знаходяться всередині МП і вибираються конкретною командою (наприклад, команда MVI A вибирає акумулятор). Управління регістрами здійснюється безпосередньо (без-використання шин) управляючою логікою всередині процесора. Вони найбільш підходять для тимчасового зберігання даних і зберігання проміжних результатів.

Деякі регістри в МП не використовуються для зберігання даних загального призначення, а виконують спеціальні функції. Найбільш важливим з них є програмний лічильник. Це шістнадцятибітний регістр, який зберігає адресу команди, яка буде виконуватися. Коли команди зчитуються з пам'яті, вміст лічильника команд поступає на шину адреси. Обрана команда з'являється на шині даних. Коли команда буде зчитана МП, вміст лічильника збільшиться на одиницю і знову поступить на шину адреси, потім зчитується наступна команда.

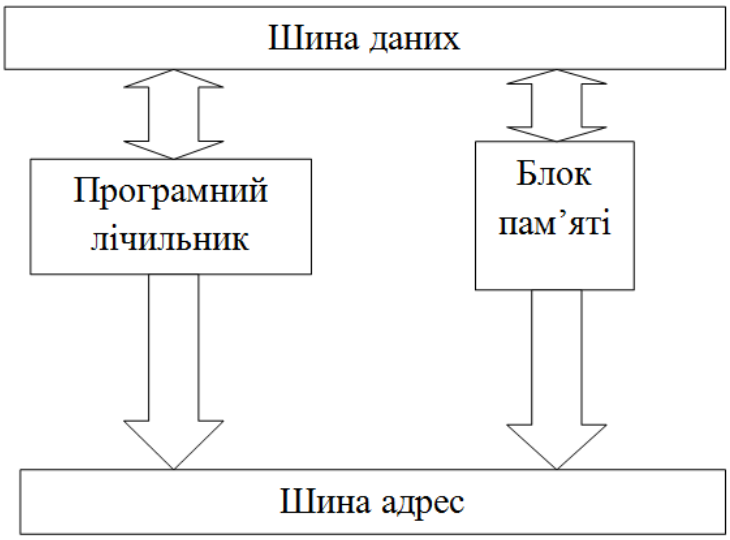

Рис. 1. Робота програмного лічильника

ПРОГРАМА МОНІТОРА "МІКРОЛАБ"

ПЗП мікролабораторії містить програму монітора, яка зчитує з клавіатури, виконує обрану операцію і керує дисплеєм. Мікролабораторія весь час виконує програму монітора, за винятком випадку, коли вона виконує програму користувача.

Коли натискається кнопка ПУСК, програма монітора змушує процесор перейти за адресою, вказаною на дисплеї "Мікролаб". Коли натискається кнопка СКИДАННЯ, мікролабораторія повертається до програми монітора. Програма монітору дозволяє перевіряти вміст регістрів в крокovому режимі після виконання кожної команди (тобто після кожного кроку). На третьому і четвертому індикаторах, рахуючи з права, після виконання кожної команди (крокovий режим) висвічується вміст акумулятора. Крім того, після кожного кроку команди програма монітора записує вміст регістрів в спеціальні комірки ОЗП. Отже, можна перевірити вміст регістрів на кожному кроці, переглянувши відповідні комірки ОЗП (табл. 1).

Таблиця 1. Адреси регістрів МП

Адреса	Регістр
83E8	Акумулятор
83EA	Регістр ознак
83E9	B регістр
83E8	C регістр
83E7	D регістр
83E6	E регістр
83E5	H регістр
83E4	L регістр
83E3	Покажчик стека (старший байт)
83E2	Покажчик стека (молодший байт)
83E1	Програмний лічильник (старший байт)
83E0	Програмний лічильник (молодший байт)

ПРОГРАМА РАХУНКУ

На рис.2. показана структурна схема програми, яка змушує комірку пам'яті рахувати в двійковому коді від 0 до 255 і потім повторювати цей рахунок.

Спочатку один регістр (в даному випадку акумулятор) встановлюється в 0. Потім вміст акумулятора переписується в комірку пам'яті з адресою 8020 і збільшується на 1. Далі запис у комірку пам'яті повторюється знову. У табл. 2, наводиться лістинг програми.

Програма починається з адреси 8004 замість 8000, так що потім можна додати кілька команд на початку програми (в "Мікролаб" 8000 - перша комірка ОЗП, програма користувача не може починатися раніше цієї адреси).

Перша команда - MVI A, 0. Вона завантажує в акумулятор нулі, інша команда - STA 8020 пересилає вміст акумулятора в комірку пам'яті 8020. Код 32 в комірці з адресою 8006 вказує, що це команда STA. Коли процесор зчитує цей код, він "розуміє", що наступні два байта (адреси 8007 і 8008) містять адресу, за якою має бути записано вміст акумулятора (в даному випадку 8020). Слід пам'ятати, що байти адреси записуються в зворотньому порядку. Ця команда не змінює вмісту акумулятора, вона просто копіює дані в комірку

пам’яті. За командою STA йде команда NOP, щоб зарезервувати місце для подальшого використання. Наступна команда - INR A, яка збільшує вміст акумулятора на одиницю. Коли досягається максимальна величина рахунку (в двійковому коді - 1111111, в десятковому – 255), вміст акумулятора скидається і подальший рахунок йде з нуля. Це звичайний режим роботи двійкового лічильника.

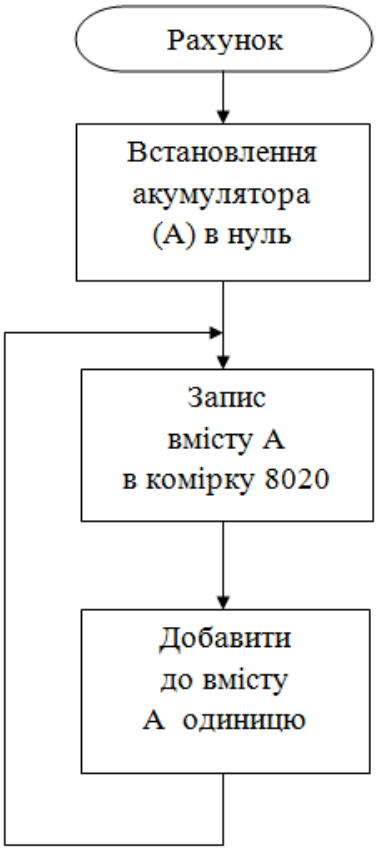

Рис. 2. Структурна схема програми рахунку.

Остання команда - перехід. Ця команда повертає програму до команди STA, адреса 8006.

Проведемо експеримент по виконанню описаної програми у кроковому режимі.

1. Введіть в пам'ять "Мікролаб" програму, наведену в табл. 2, перевірте правильність її запису в пам'ять.

Таблиця 2. Лістинг програми рахунку

Адреса	Вміст	Мітка	Команда	Коментар
8004	3E		MVI A, 0	Запис "0" в A
8005	00			
8006	32	LOOP:	STA 8020	A -> комірка пам'яті
8007	20			
8008	80			
8009	00		NOP	
800A	3C		INR A	Збільшити A
800B	C3		JMP LOOP	Перехід до LOOP
800C	06			
800D	80			

2. Переключіть тумблер режиму в положення КРОК.

3. Встановіть початкову адресу програми (8004). На двох крайніх правих індикаторах з'явиться команда MVI A (код 3E).

4. Натисніть кнопку ПУСК. Програма почала виконуватися в кроковому режимі. Перша команда виконана. Тепер на двох крайніх правих індикаторах висвітлений вміст регістру ознак, а вміст акумулятора з'явився на третьому і четвертому індикаторах, рахуючи справа. На цих індикаторах видно нулі, тому що перша команда виконана, і в акумулятор занесені нулі.

5. Натисніть кнопку ПУСК. Виконана команда STA 8020. Вміст акумулятора не змінився, в комірці пам'яті 8020 повинні з'явитися нулі. Щоб перевірити вміст цієї комірки, натисніть кнопки 8, 0, 2, 0, а також кнопку УСТ.АД. На двох крайніх правих індикаторах - нулі. Це вміст комірки 8020. Набираючи будь-яку іншу адресу, можна проглянути будь-яку комірку пам'яті після кожного кроку виконання програми. Набираючи відповідні адреси, можна також переглянути і вміст регістрів МП (див. табл. 1).

6. Натисніть кнопку ВОЗВР. Тим самим ви повернулися до виконання вашої програми і виконали наступну команду (NOP). На індикаторах знову з'явився вміст регістра ознак і акумулятора.

7. Натисніть кнопку ПУСК. Виконана команда INR A. На індикаторі з’явилася одиниця, що свідчить про збільшенні вмісту акумулятора.

8. Перевірте вміст комірки 8020. У ній знаходяться нулі, тому що перепис одиниці з акумулятора в цю комірку ще не відбувся.

9. Натиснувши кнопку ВОЗВР., поверніться до виконання вашої програми.

10. Натисніть кнопку ПУСК, виконається наступна команда (STA 8020), тобто вміст акумулятора переписався в комірку 8020.

11. Перевірте вміст комірки 8020. Тепер там знаходиться одиниця.

12. Натиснувши кнопку ВОЗВР і далі натискаючи кнопку ПУСК, виконайте програму по кроках. Прослідкуйте, як змінюється вміст акумулятора.

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1. Які регістри існують в мікропроцесорі КР580ИК80?
2. Яку роль відіграє в мікропроцесорі акумулятор?
3. Які ознаки має регістр ознак?
4. Поясніть числа, отримані в результаті експерименту.
5. Для чого використовуються регістри загального призначення?
6. Поясніть всі команди, які використані в цій роботі.

ЛАБОРАТОРНА РОБОТА №3

Вивчення основних принципів організації програм в МП

Мета роботи - вивчити основні принципи організації програм в МП, отримання навичок у розробці програм з підпрограмами.

КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

В різних випадках застосування перед МП ставляться різні завдання. Наприклад, вольтметр з вбудованим МП повинен виміряти вхідну напругу і послати дані на дисплей. Процесор повинен перевести дані в вольти і автоматично підтримувати нуль шкали. Вимірювач може мати клавіатуру (для введення користувачем спеціальних запитів). Для автоматичної установки діапазону процесор може бути пов'язаний з атенюатором. Кожна з задач, що стоять перед МП, є в основному незалежною і може бути представлена відносно простою програмою. Потім одна спільна (основна) програма може об'єднати всі спеціалізовані програми, не вдаючись у подробиці кожної задачі.

Проведемо аналогію зі структурою великого підприємства. Директор не може виконати всю роботу, але він може приймати всі головні рішення. Існують ряд заступників, які відповідають за певні напрямки роботи. У них в підпорядкуванні знаходяться начальники рангом нижче, які, в свою чергу, керують іншими службовцями. На підприємстві кожен має завдання для вирішення, і чим нижче ви на службовій драбині, тим більше ви вникаєте в деталі. Директор має час приймати важливі рішення, тому що він може змусити інших людей виконувати менш важливі. Вони, в свою чергу, змушують інших, які залучають ще людей, якщо це вимагається, потім людина доповідає своєму начальнику, і нарешті, директор отримує відповідь.

Це дуже схоже на роботу процесорної системи. Через те, що задачі можуть бути дуже різними і складними, то існує багато програм. Одна програма, яку часто називають виконавчою діє, як директор. Програма-

виконавець має інші програми, які працюють на неї, і які, в свою чергу, мають інші програми, працюючі на них і т.д. Програми, що працюють на інші програми, називають підпрограмами. Підпрограми можуть мати інші підпрограми, що працюють на них і т.д.

Розглянемо приклад організації програми. На рис. 3 наведена проста програма, яка забезпечує запис у комірку пам'яті 8020 то нулів, то одиниць. В цій програмі використовуються підпрограми. Одна підпрограма використовується для запису в комірку пам'яті одиниць, інша - для запису нулів.

Основна програма містить тільки три команди: одну - для виклику підпрограми "нулі", іншу - для виклику підпрограми - "одиниці", і третю для переходу до початку. Основна програма визначає всі функції програми, але їй не потрібно знати адресу комірки пам'яті. Потреби визначаються підпрограмами.

Команда CALL змушує МП перейти на підпрограму. Мітки "0" і "1" визначають підпрограми і розміщуються за реальною адресою, коли мова Асемблер переводиться в машинний код. Перші дві команди кожної з підпрограм мають структуру, аналогічну команді JMP.

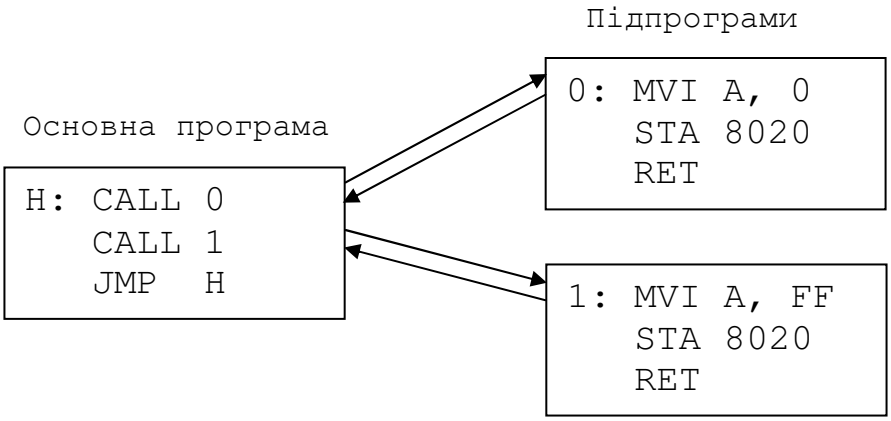

Рис. 3. Приклад використання підпрограм.

Остання команда RET (повернення), визначає кінець підпрограми і повертає МП до виконання основної програми. В табл. 3 показана, повна

програма з машинними кодами.

Таблиця 3. Лістинг програми записи в комірку пам’яті нулів і одиниць

Адреса	Вміст	Мітка	Команда	Коментар
Основна програма				
8000	CD	H:	CALL 0	Виклик підпрограми запису нулів
8001	09			
8002	80			
8003	CD		CALL 1	Виклик підпрограми запису одиниць
8004	0F			
8005	80			
8006	C3		JMP H	Повтор
8007	00			
8008	80			
Підпрограма запису нулів в комірку пам’яті 8020				
8009	3E	0:	MVI A, 00	Записати в А нулі
800A	00			
800B	32		STA 8020	Записати вміст А в комірку 8020
800C	20			
800D	80			
800E	C9		RET	Повернення в основну програму
Підпрограма запису одиниць в комірку пам’яті 8020				
800F	3E	1:	MVI A, FF	Записати в А одиниці
8010	FF			
8011	32		STA 8020	Записати вміст А в комірку 8020
8012	20			
8013	80			
8014	C9		RET	Повернення в основну програму

Команда CALL складається з коду операції CD, за яким іде адреса підпрограми. Адреса підпрограми вказується таким самим чином, як і адреса в командах STA та JMP. Команда RET складається тільки з коду C9. Ніякої адреси команда не містить. Однак МП "знає", куди повертатися, коли він зустрічає команду повернення в кінці підпрограми. Коли виконується команда CALL, адреса повернення команди, наступної після виклику підпрограми, зберігається в спеціальному місці пам'яті, яке називається СТЕК. Коли в кінці підпрограми зустрічається команда RET, МП одержує із стека адресу повернення до основної програми. Робота стека здійснюється майже повністю

автоматично.

Проведемо експеримент з використання програми, наведеної в табл. 3.

1. Введіть дану програму в пам'ять "Мікролаб".

2. Поставте тумблер режиму в положення КРОК.

3. Встановіть початкову адресу програми (8000) та натисніть кнопку ПУСК. Виконуються перша команда програми (CALL 0), і відбувається перехід до підпрограми (адреса 8009).

4. Натисніть кнопку ПУСК. Виконується наступна команда, яка є першою в підпрограмі. В акумулятор заносяться нулі.

5. Натисніть кнопку ПУСК. Вміст акумулятора заноситься в комірку пам'яті з адресою 8020. Перевірте вміст цієї комірки. Там записані нулі.

6. Натисніть кнопку ВОЗВР. Виконується команда RET, і ви повернетесь в основну програму (адреса 8003).

7. Натисніть кнопку ПУСК. Відбудеться перехід у другу підпрограму (адреса 800F).

8. Натисніть двічі кнопку ПУСК. У акумулятор запишуться одиниці (FF), і вміст акумулятора перепишеться у комірку пам'яті з адресою 8020.

9. Перевірте, чи є в комірці 8020 число FF.

10. Натисніть кнопку ВОЗВР. Виконається команда повернення (RET), і відбудеться перехід до основної програми (адреса 8006). За цією адресою будуть записані команди безумовного переходу до адреси 8000. Після цього програма почне виконуватися знову.

11. Натисніть кілька разів кнопку ПУСК, перегляньте програму ще раз.

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1. Навіщо в мікропроцесорі використовуються підпрограми?

2. Яка різниця між командами CALL та JMP?

3. Де вказується адреса для команди RET?

4. Поясніть всі команди, які використовуються в програмі.

5. Навіщо в програмі використовуються мітки?

ЛАБОРАТОРНА РОБОТА №4

Вивчення основних видів адресації

Мета роботи - вивчити основні види адресації в МП, отримати навички в програмуванні при роботі з командами пересилання даних.

КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Серед команд пересилки даних з регістровою адресацією типу MOV r1,r2 на місці одного з регістрів може стояти літера М. Однак такого регістру не існує. Так позначається комірка пам'яті, адреса якої запам’ятовується в регістрах H і L. Така адресація називається непрямою, тобто команда вказує, де запам’ятовується адреса (в нашому випадку – в регістрах H і L), а не дійсну адресу. Наприклад, якщо регістр H містить 12, а регістр L - 37 (рис. 2), команда MOV A, M буде завантажувати в акумулятор вміст комірки пам’яті з адресою 1237. Результат буде той же, що і при команді LDA 1237. Це є прикладом того, як одна операція може бути виконана двома способами. MOV A, M являється однобайтовою командою, але потребує попереднього завантаження регістрів H і L. З іншого боку, LDA 1237 є 3-байтной командою. Однак її часто використовують, через те що вона не вимагає запам'ятовування адреси в регістрах H і L.

Проведемо експеримент для демонстрації використання регістрів загального призначення при роботі команди MOV.

1. Введіть програму, наведену в табл. 1 (програма спочатку заносить в регістр B число 37, потім переносить його вміст в регістр H, і нарешті, нарощує регістр H).

2. Впевніться, що програма правильно записана в пам'ять.

3. Встановіть тумблер АВТ-КРОК положення КРОК.

4. Натисніть послідовно кнопки 8, 0, 0, 0, УСТ.АД.

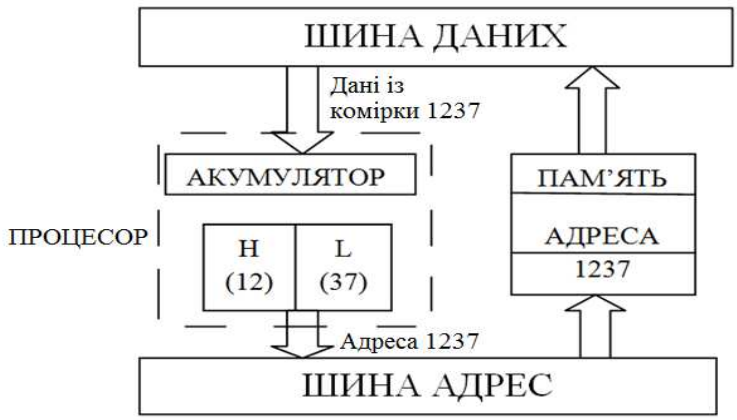

Рис. 2. Непряма адресація з використанням регістрів H і L

Таблиця 1. Параметри програми

Адреса	Дані	Команди	Коментарі
8000	06	MVI B, 37	Завантаження числа 37 в регістр B
8001	37		
8002	60	MOV H, B	Завантаження регістра H вмістом регістра B
8003	24	INR H	Прирощення вмісту регістра H

5. Натисніть кнопку ПУСК. При цьому виконується команда MVI B, 37. На індикаторі адрес висвічується число 8002 - це адреса наступної команди, яка записана у програмному лічильнику. На індикаторі даних в двох правих розрядах висвічується вміст регістра ознак, а в двох лівих - вміст акумулятора.

6. Натисніть послідовно кнопки 8, 3, E, 9, УСТ.АД. На індикаторі адрес встановиться адреса регістра B. На індикаторі даних в двох правих розрядах висвічується число 37 - вміст регістра B після виконання команди MVI B, 37.

7. Натисніть кнопку ВОЗВР. На індикаторі адрес висвічується адреса 8003. При цьому виконається команда MOV H, B.

8. Встановити на індикаторі адрес число 83E5 - адреса регістра H. На індикаторі даних в двох правих розрядах висвічується числі 37 - вміст регістра B після виконання команди MOV H, B. Встановивши на індикаторі адрес адресу регістра B, переконайтеся, що цей регістр все ще містить число 37.

9. Натисніть кнопку ВОЗВР. На індикаторі адрес висвічується число 8004 - адреса наступної команди. При цьому виконується команда INR H. Встановивши на індикаторі адресу регістра B, переконайтеся, що його вміст збільшився на 1, тобто дорівнює 38.

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1. Які основні види адресації ви знаєте?
2. Яка різниця між прямою та безпосередньою адресаціями?
3. Де вказується адреса при непрямій регістровій адресації?
4. Які вузли мікропроцесора використовуються при регістровій адресації, при непрямій регістровій адресації?
5. Поясніть всі команди, які використовуються в програмі.

ЛАБОРАТОРНА РОБОТА № 5

Вивчення команд арифметичних операцій МП

Мета роботи - вивчити склад основних команд арифметичних операцій мікропроцесора та їх застосування.

КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Додавання

Основна арифметична функція - це додавання двох чисел. Команда ADD R додає дані визначеного регістра до вмісту акумулятора і запам'ятовує результат в акумуляторі. Наприклад, ADD D додає данні регістра D до вмісту акумулятора. Нехай D=1001 0011 і A=1010 1010, тоді результат буде:

в десятковій системі	в двійковій системі
+ 147	+ 1001 0011 D
170	1010 1010 A
317	10011 1101 A

Флаг переносу

Зауважимо, що в попередньому прикладі результат більше 255_{10} тобто довше восьми біт. Для визначення цього переповнення процесор містить флаг переносу. Цей флаг працюєте як дев'ятий біт акумулятора і може бути перевірений командами JC (перехід, якщо є перенос) і JNC (перехід, якщо немає переносу). Ця перевірка подібна перевірці нульового флага.

Наприклад, програма ADD D; JC 8020 переводить вміст регістра D в акумулятор і переходить на комірку 8020, якщо генерується флаг переносу. Якщо переносу немає (результат менше 256_{10}) програма продовжується з наступної команди.

Флаг переносу застосовується також для додавання чисел довших за 8 біт. Два (або більше) регістри використовуються для зберігання кожного числа

Спочатку складаються молодші значущі байти, потім - старші. Біт переносу виконує функцію зв'язку між двома байтами.

Можна перевірити флаги, заглянувши в комірку пам'яті 83EA. Флаги зібрані в один байт, показаний на рис. 5. У цій роботі використовуються тільки флаги нуля і переносу. Не всі команди змінюють флаги. Команда MOV, наприклад, їх не змінює. Опис команд показує, які флаги генеруються кожною командою.

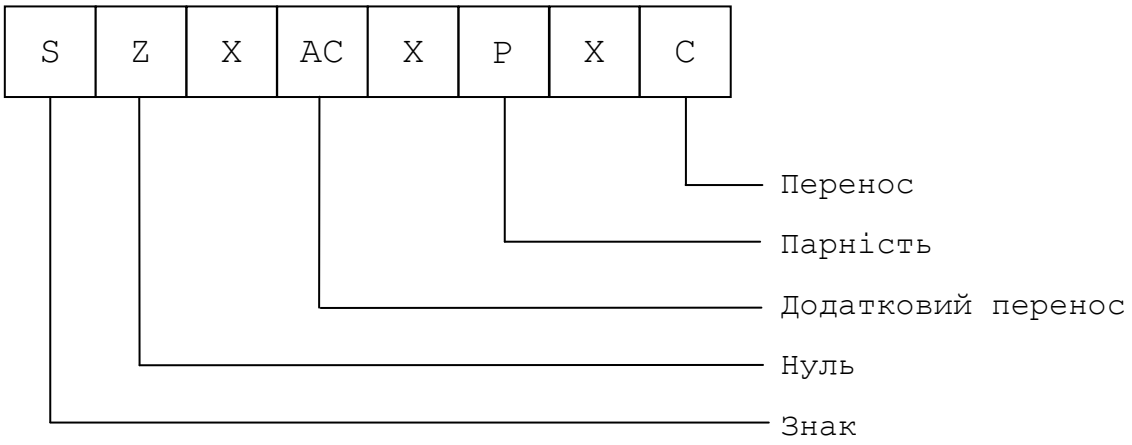

Рис.5. Ознаки МП КР580ИК80

Віднімання

Команда SUB R віднімає вміст певного регістра із вмісту акумулятора. Наприклад SUB B віднімає з акумулятора дані регістра B.

Команда віднімання використовує флаг переносу як флаг займу. Якщо флаг переносу встановлюється після команди SUB R, значить число в регістрі R більше, ніж в A.

Проведемо експеримент з виконання арифметичних команд.

1. Доповніть машинними кодами програму наведену в табл.4. Ця програма віднімає дані регістра B з акумулятора, а потім додає до акумулятора дані регістра C.

Таблиця 4. Програма демонстрації арифметичних команд

Адреса	Вміст	Мітка	Команда	Коментарі
			SUB B	A-B ->A
			ADD C	A+C ->A

2. Введіть програму в Мікролабораторію. Перейдіть в кроковий режим виконання програм.

3. Занесіть в регістр А число А7, в регістр В число 23, в регістр С число 93. Для цього необхідно встановлювати адресу комірки пам'яті, що відповідає даному регістру та записувати туди необхідні данні. Нагадаємо, що вміст регістра А зберігається в комірці 83ЕВ, регістра В - в комірці 83Е9, регістра С - в комірці 83Е8. При крововому виконанні програми монітор записує дані з цих комірок у відповідні регістри МП і після виконання кожної команди записує вміст регістра назад у відповідні комірки.

4. Встановіть початкову адресу програми й натисніть кнопку ПУСК. Команда SUB B повинна виконатися. Перевірте вміст акумулятора, який висвічується цифровими індикаторами. Він повинен дорівнювати $A7_{16}$-23_{16}=84_{16}. Перевірте вміст флагового регістра. Флаги займу та переносу не встановлені, тому що результат не дорівнює нулю і 23_{16}<$A7_{16}$.

5. Натисніть кнопку ВОЗВР. Виконується команда ADD C. Вміст акумулятора стає рівним 84_{16}+93_{16} = $l17_{16}$. Цей результат не вміщується в акумулятор, в якому залишається код 17_{16}, і формується флаг переносу. Перевірте вміст прапора переносу і переконайтесь, що він встановлений.

6. Переконайтеся, що при виконанні цієї програми вміст регістрів В і С не змінився.

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1. Які команди додавання ви знаєте?

2. Які команди віднімання ви знаєте?

3. Які команди збільшення або зменшення числа ви знаєте?

4. При яких операціях потрібні команди з переносом?

ЛАБОРАТОРНА РОБОТА № 6

Вивчення команд логічних операцій МП

Мета роботи - вивчити склад і застосування основних команд логічних операцій мікропроцесора.

КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

ЛОГІЧНІ КОМАНДИ

В основі побудови цифрових схем лежить логічний вентиль, а найбільш широко використовуються чотири базові функції вентиля: «НЕ», «І», «АБО», «Виключаюче АБО». Кожна з цих функцій може бути реалізована програмним способом.

Функцію «НЕ» виконує команда CMA (доповнення акумулятора). Кожний біт акумулятора після виконання цієї команди інвертується.

Функцію «І» виконує команда ANA R. Символ R позначає або один з регістрів, або комірку пам'яті. Наприклад, ANA D викликає об'єднання по «І» вмісту регістра D і акумулятора. Результат залишається в акумуляторі, вміст регістру (в нашому прикладі D) не змінюється. Операція з якимось певним регістром має свій код.

Рис. 1 показує апаратний еквівалент команди ANA R. Функція «І» виконується індивідуально для кожного біта акумулятора. Наприклад, якщо A=1011 0110 і D=0011 1100, результатом команди ANA D буде:

```
«I» 0011  1100 – D
    1011  0110 – A
    ----------
    0011  0100 – A
```

Функцію "АБО" виконує команда ORA R (об'єднання по "АБО" акумулятора і регістра R).

Функцію "Виключаюче АБО" виконує команда XRA R. Виконання цих команд подібно виконанню ANA R, тільки реалізуються інші логічні функції. Оскільки адресу або дані визначати не потрібно, то всі ці команди вимагають тільки один байт коду операції (див. набір команд МП КР580ИК80).

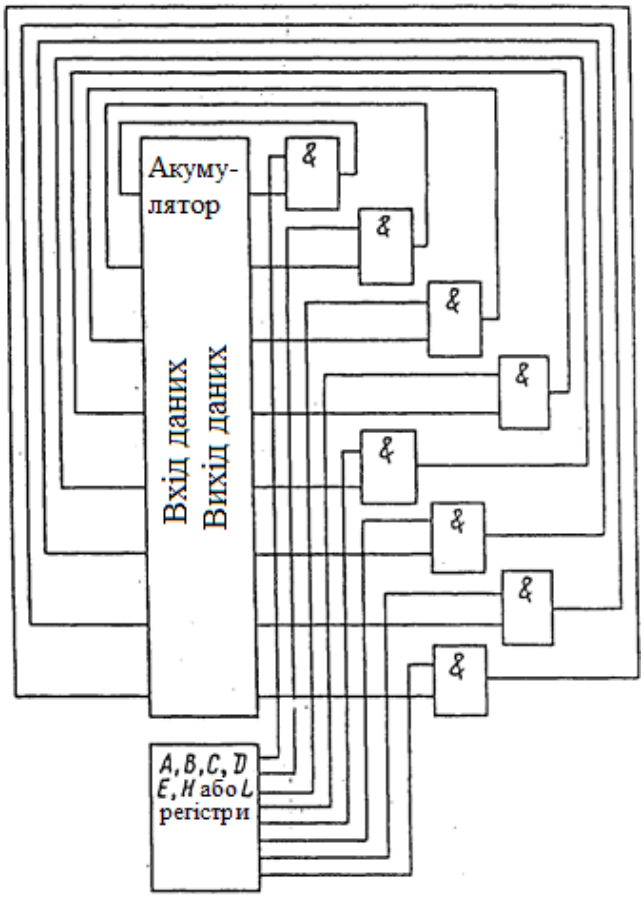

Рис. 1. Апаратний еквівалент команди ANA.

Для ілюстрації дії логічних команд приведемо наступний експеримент: будемо вводити дані з перемикачів вхідного порту, потім ці дані перетворюються логічними командами, і результат висвітиться на світлодіодних індикаторах вихідного порту. Через те, що мікролабораторія має тільки три перемикачі вхідного порту, ми будемо спостерігати тільки за трьома молодшими розрядами байта даних (див. схему).

1. Переведіть програму (табл. 1) в машинний код і введіть у мікролабораторію.

Таблиця 1. Програма, що демонструє дію логічних команд

Адреса	Вміст	Мітка	Команда	Коментарі
8000			MVI A, 81	Програмування інтерфейсу
8001				КР580ВВ55
8002			OUT FB	
8003				
8004		START:	IN FA	Зчитування даних з вхідного
8005				порту в акумулятор
8006			MVI B, 0A	0000 1010 -> B
8007				
8008			ANA B	AB -> A
8009			OUT F9	Запис даних в порт виводу
800A				
800B			JMP START	Перехід на початок циклу
800C				
800D				

Кожне мнемонічне позначення повинно бути перетворено в відповідний код і доповнено необхідно даними (якщо потрібно). Ця програма зчитує дані з перемикача вхідного порту, а потім об'єднує їх по "І" з числом 0000 1010. Результат з'являється на світлодіодних індикаторах. Потім програма повертається на початок, і процес нескінченно повторюється, так що можна змінити вхідні дані і перевірити відповідні вихідні.

2. Перевірте правильність вводу програми.

3. Встановіть початкову адресу 8000 і натисніть кнопку ПУСК в автоматичному режимі. Програма виконується.

4. Встановіть вхідні перемикача в 1. Індикатори показують 0000 1010.

5. Змініть положення перемикача та перевірте відповідні виходи. Біти, які об'єднуються по "І" з нулями, не залежать від положення перемикачів.

6. Натисніть кнопку СКИДАННЯ для повернення управління монітору. Зауважте, що перемикачі не впливають більше на стан індикаторів. Замініть команду ANA B (за адресою 8008) командою ORA B (код операції – B0).

7. Запустіть програму.

8. Подивіться на відповідність між різними положеннями вхідних

перемикачів і станами індикаторів. Зауважте відмінності між функціями "І" і "АБО". Виходи бітів, об'єднаних по "АБО" з нулями, залежать від положення перемикача.

9. Натисніть кнопку СКИДАННЯ. Замініть команду ORA B на XRA B (код операції - A8) . Повторіть кроки 7 і 8. Біти, об’єднання по "Виключаючому АБО" з одиницями, інвертуються.

МАСКУВАННЯ

Типове застосування логічних команд - виділення певних бітів слова (або маскування). Для прикладу припустимо, що нам потрібно перевірити стан, одного з перемикачів, підключених до порту введення, а стан інших проігнорувати. На рис. 2 показана структурна схема програми, яка перевіряє перемикач, з'єднаний з другим бітом вхідного порту. Якщо ключ замкнений, індикатор загоряється і - навпаки.

Табл.2 показує лістинг програми. Спочатку програма робить програмування інтерфейсу. Потім вона зчитує дані з вхідного порту в акумулятор, засилає в регістр B маскуюче слово і об'єднує по "І" регістр B і регістр A. В результаті всі біти, крім другого, приводяться до нуля. Значення другого біта результату буде залежати від положення перемикача. Програма використовує команду JZ для переходу, якщо встановлено флаг нуля. Нульовий флаг показує, що всі біти байта (в т.ч. біт 2) дорівнюють нулю.

Проведемо експеримент по виконанню цієї програми.

1. Переведіть програму у машинний код і введіть у мікролабораторію. Перевірте правильність вводу програми.

2. Запустіть програму і перевірте, чи вхідні перемикачі правильно управляють індикаторами. Всі індикатори запаляться тільки в тому випадку, якщо включений другий перемикач. Стан інших перемикачів байдужий.

3. Змініть програму, щоб перевірити інший перемикач. Якщо програма не працює, перевірте правильність машинних кодів і правильність введення програми.

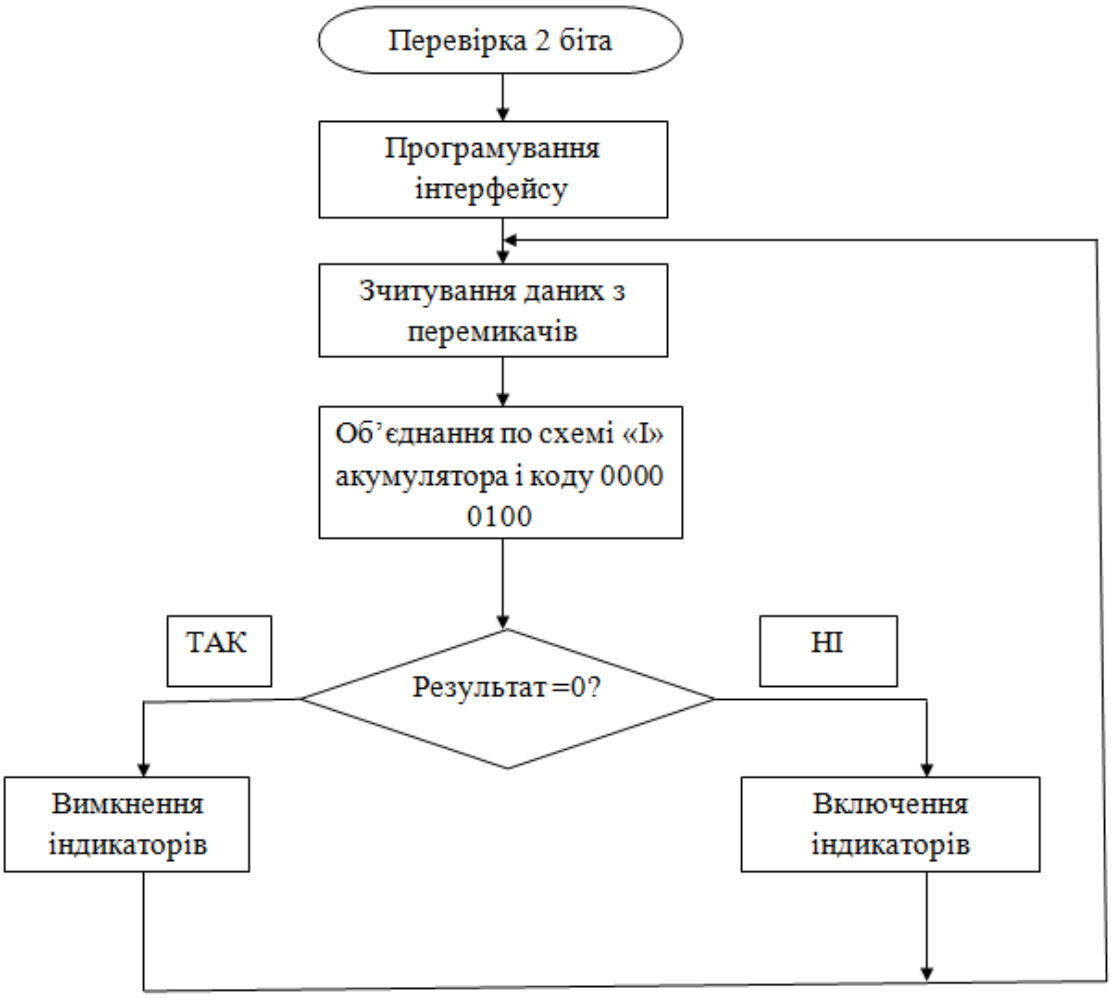

Рис. 2 Структурна схема програми для перевірки другого біту вхідного порту.

СКИДАННЯ АКУМУЛЯТОРА

Команда XRA A об'єднує по "Виключаючому АБО" акумулятор з самим собою. Через те, що логічна функція «Виключаючому АБО» будь-якого числа з самим собою дає в результаті нуль, ця команда очищає акумулятор, використовуючи тільки один байт пам'яті. Інша команда MVI A, O вимагає два байти пам'яті.

Таблиця 2. Програма перевірки другого біта

Адреса	Вміст	Мітка	Команда	Коментарі
8000			MVI A, 81	Програмування інтерфейсу
8001				
8002			OUT FB	
8003				
8004		START:	IN FA	Зчитування даних з перемикачів
8005				в акумулятор
8006			MVI B, 04	Запис в регістр B маскуючого
8007				слова (0000 0100)
8008			ANA B	AB -> A, виділення другого біту
8009			JZ OFF	Перевірка акумулятора на 0
800A				
800B				
800C		ON:	MVI A, FF	Ввімкнення індикаторів
800D				
800E			OUT F9	
800F				
8010			JMP START	
8011				
8012				
8013		OFF	MVI A, 00	Вимкнення індикаторів
8014				
8015			OUT F9	
8016				
8017			JMP START	
8018				
8019				

ЗСУВ ДАНИХ

В деяких програмах використовується зсув даних вправо або вліво. В апаратурі це виконується за допомогою регістрів зсуву. МП КР580ИК80 має команди, котрі зсувають дані в акумуляторі.

RRC (кругове обертання вправо) виконує зсув вправо, а RLC - вліво. Ці команди оперують тільки з акумулятором. Круговий рух означає, що молодший біт стає на місце старшого або навпаки (рис. 3).

Проілюструвати дію команд зсуву можна за допомогою програми, яка створює на індикаторах світіння типу "вогню, що біжить". Лістинг цієї програми наведено в табл. 3. Після програмування інтерфейсу програма заносить в акумулятор код 1000 0000. Потім проводиться виведення вмісту акумулятора на індикатори. Команда вивода (OUT F9) повторюється деякий час, тому що вона знаходиться в петлі затримки, організованої за допомогою регістрів D і E. Затримка забезпечує повільну зміну стану індикаторів, без неї зсув даних акумулятора проходив би настільки швидко, що ми б не могли простежити за ним.

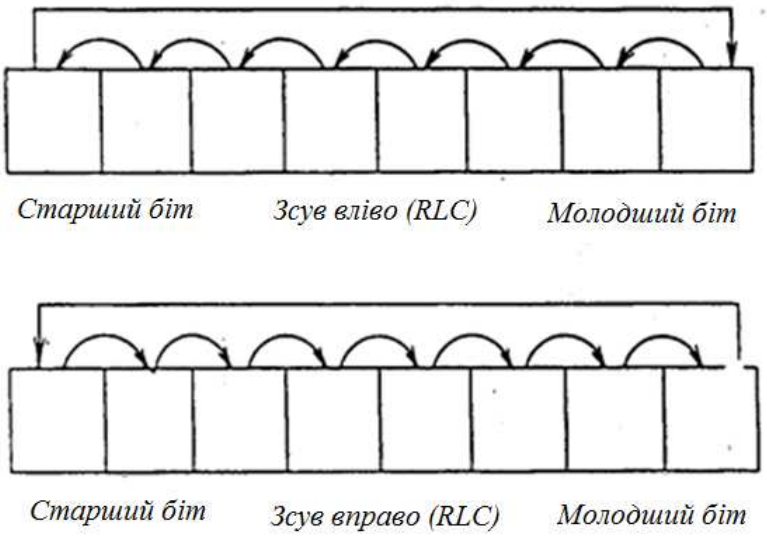

Рис. 3 Функції команд зсуву

Проведемо експеримент з виконанням цієї програми.

1. Переведіть програму в машинний код і введіть в мікролабораторію.

2. Перевірте правильність вводу програми і запустіть її. На індикаторах з'явиться ефект "вогню, що біжить" вправо. Якщо не вдалося досягти потрібного ефекту, перевірте правильність машинних кодів записаної програми.

3. Замініть команду RRC на RLC. У цьому випадку спостерігається "вогонь, що біжить" вліво.

4. Ви можете також змінити частоту перемикання індикаторів, заносячи в регістр різні числа (тобто змінюючи затримку). Для цього достатньо міняти код за адресою 8007, тобто змінювати другий байт команди MVI D. Замінюючи код за адресою 8005, ви можете змінити комбінацію «запалених індикаторів».

Таблиця 3. Програма для демонстрації функцій команд зсуву

Адреса	Вміст	Мітка	Команда	Коментарі
8000			MVI A, 81	Програмування інтерфейсу
8001				
8002			OUT FB	
8003				
8004			MVI A, 80	1000 000 -> A
8005				
8006		LOAD:	MVI D, 40	40 ->D, Завантаження регістрів
8007				D і E для організації затримки
8008			MVI E, FF	FF -> E
8009				
800A		LOOP:	OUT F9	Виведення даних на індикатори
800B				
800C			DCR E	E-1 -> E
800D			JNZ LOOP	Якщо E≠0, то перехід на LOOP
800E				(петля затримки)
800F				
8010			DCR D	D – 1 -> D
8011			JNZ LOOP	Якщо D≠0, то перехід на LOOP
8012				(петля затримки)
8013				
8014			RRC	Зсув даних в акумуляторі
8015			JMP LOAD	Перехід на початок циклу
8016				
8017				

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1. Які команди з функцією логічного додавання ви знаєте?
2. Які команди з функцією логічного множення ви знаєте?
3. Які команди з функцією «виключаючи АБО» ви знаєте?
4. Поясніть команди умовного переходу
5. Чим відрізняється команда RRC від команди RLC?

Додаток А

Мікропроцесорна система "Мікролаб"

Мікропроцесорна система "Мікролаб" призначена для використання в учбовому процесі та дозволяє виконувати лабораторні роботи з вивчення мікропроцесора КР580ИК80 (аналога Intel 8080). До складу системи входять: мікропроцесор КР580ИК80; ОЗП ємністю 1 Кб; ПЗП ємністю 1 Кб; 8-розрядний цифровий дисплей; клавіатура; 3 перемикачі та 8 світлодіодів, які відповідно з'єднані з портами введення та виведення. Конструктивно "Мікролаб" складається з плати, на якій розміщенні всі компоненти системи та блоку живлення, що знаходиться під платою.

Призначення органів керування системи "Мікролаб":

Вимикач живлення – виконаний у вигляді кнопки, що знаходиться на лівій бічній поверхні блока живлення;

Скидання – кнопка подачі сигналу скидання на мікропроцесор КР580ИК80. При натисканні "Скидання" вміст ОЗП не змінюється.

0-9, A-F – клавіші для введення шістнадцяткових цифр;

Уст. АД. – встановлення поточної адреси пам'яті;

АД+, АД- – відповідно збільшення або зменшення поточної адреси на 1;

Запис – запис байта за поточною адресою;

Авт/Крок – перемикач для вибору режиму виконання програми:

Авт – програма виконується автоматично;

Шаг – програма виконується покроково;

Пуск – запуск програми на виконання, починаючи з поточної адреси;

Возвр. – повернення до виконання основної програми. Використовується, якщо під час покрокового виконання програми були використані команди монітора;

Ввод, Вивод – відповідно зчитування програми з магнітофона та запис на магнітофон (не використовуються під час виконання лабораторних робіт).

ПЗП системи "Мікролаб" містить програму-монітор, яка надає можливість вводити та запускати програми користувача. Під час роботи з системою можна виконувати наступні дії:

1. **Перегляд та редагування вмісту ОЗП.** Для перегляду вмісту ОЗП слід ввести з клавіатури шістнадцяткову адресу та натиснути "Уст. АД.". Під час введення адреса відображується в 5-8 розрядах дисплея (нумерація зліва направо). Після натискання "Уст. АД." введена адреса стає поточною та відображується в розрядах 1-4 дисплея. Вміст байта пам'яті за поточною адресою відображується в розрядах 7-8 дисплея. Для зміни вмісту пам'яті слід ввести нове значення байта та натиснути "Запис". При цьому введене значення записується за поточною адресою, а поточна адреса збільшується на 1. Для зміни поточної адреси можна використовувати не тільки "Уст. АД.", але і "АД+", "АД-", які збільшують або зменшують поточну адресу на 1. Якщо під час введення адреси або даних була допущена помилка, то для її виправлення достатньо повторно ввести адресу або дані, не натискаючи "Уст. АД." або "Запис". Запис даних можливий лише в ОЗП, яке розташоване у діапазоні адрес 8000-83FF (табл. А.1).

2. **Перегляд та редагування вмісту регістрів процесора.** Після виконання програми користувача монітор копіює вміст регістрів процесора в комірки пам'яті (табл. А.2). Тому для перегляду або редагування вмісту регістрів необхідно переглядати або редагувати відповідні комірки пам'яті.

3. **Запуск програми на виконання.** Для запуску програми слід встановити поточну адресу, з якої починається виконання програми, та натиснути "Пуск". Якщо встановлений режим покрокового виконання програми, то при кожному натисканні "Пуск" виконується одна команда програми. Після виконання команди на дисплей виводиться: в розрядах 1-4 – адреса наступної команди; в розрядах 5-6 – вміст акумулятора; в розрядах 7-8 – вміст регістра ознак. Якщо в покроковому режимі між натисканнями "Пуск" використовувалися команди монітора (наприклад, для перегляду вмісту ОЗП),

то для виконання наступної команди слід використати "Возвр.". Якщо встановлений режим автоматичного виконання, то після натискання "Пуск" керування передається програмі користувача, а для повернення в монітор необхідно натиснути "Скидання".

4. **Використання портів вводу-виводу.** Доступ до портів вводу-виводу здійснюється за допомогою команд IN та OUT. Порт вводу, який з'єднаний з 3 перемикачами, має адресу FA. Порт виводу, який з'єднаний з 8 світлодіодами, має адресу F9. Перед використанням портів F9, FA необхідно ввести в порт FB байт 81_{16}.

Таблиця А.1. Карта пам’яті системи «Мікролаб»

Адреси	Ємність пам’яті	ПЗП/ОЗП	Використання
FFFF | 8400	31К		Область що не використовується
83FF | 83C7	57	ОЗП	Робоча область монітора
83C6 | 8000	967	ОЗП	Область користувача
7FFF | 0600	30,5К		Область, що не використовується
05FF | 0400	512	ПЗП	Область користувача
03FF | 0300	256	ПЗП	Додаткова область монітора
02FF | 0000	768	ПЗП	Область монітора

Таблиця А.2. Адреси регістрів МП

Адреса	Регістр
83E8	Акумулятор
83EA	Регістр ознак
83E9	B регістр
83E8	C регістр
83E7	D регістр
83E6	E регістр
83E5	H регістр
83E4	L регістр
83E3	Покажчик стека (старший байт)
83E2	Покажчик стека (молодший байт)
83E1	Програмний лічильник (старший байт)
83E0	Програмний лічильник (молодший байт)

Додаток Б

Емулятор мікропроцесорної системи "Мікролаб"

Програма-емулятор `cpukr580` призначена для використання в учбовому процесі замість мікропроцесорної системи "Мікролаб" та дозволяє виконувати лабораторні роботи з вивчення мікропроцесора КР580ИК80 (аналога Intel 8080). Програма емулює мікропроцесорну систему, до складу якої входять: мікропроцесор КР580ИК80; ОЗП ємністю 4 Кб; 8-розрядний цифровий дисплей; клавіатура; 3 перемикачі та 8 світлодіодів, які відповідно з'єднані з портами введення та виведення. Головне вікно програми розділене на три частини: ліва частина – відображення вмісту ОЗП; права верхня частина – відображення вмісту регістрів процесора; права нижня частина – органи керування системи "Мікролаб".

Призначення органів керування:

Скидання – переводить емулятор у вихідний стан. При натисканні "Скидання" вміст ОЗП не змінюється.

0-9, A-F – клавіші для введення шістнадцяткових цифр;

Уст. АД. – встановлення поточної адреси пам'яті;

АД+, АД- – відповідно збільшення або зменшення поточної адреси на 1;

Запис – запис байта за поточною адресою;

АВТ/ШАГ – кнопка вибору режиму виконання програми:

АВТ – програма виконується автоматично;

ШАГ – програма виконується покроково.

Напис на кнопці вказує режим, який увімкнений;

Пуск – запуск програми на виконання, починаючи з поточної адреси;

К1-К3 – перемикачі, з'єднані з портом вводу;

VD0-VD7 – світлодіоди, з'єднані з портом виводу.

Під час роботи з програмою можна виконувати наступні дії:

1. **Перегляд та редагування вмісту ОЗП.** Для перегляду вмісту ОЗП використовується ліва частин вікна програми. Для встановлення поточної адреси необхідно натиснути "Уст. АД." та ввести з клавіатури шістнадцяткову адресу. Введена адреса відображується в 1-4 розрядах дисплея (нумерація зліва направо). В 7-8 розрядах дисплея відображується вміст відповідного байта пам'яті. Для зміни вмісту пам'яті слід натиснути "Запис" та ввести нове значення байта. При цьому введене значення записується за поточною адресою та відображується в 7-8 розрядах дисплея. Далі слід встановити нову поточну адресу за допомогою "Уст. АД." або "АД+", "АД-" та ввести значення байта і т.д.. Якщо використовуються кнопки "АД+", "АД-", то поточна адреса відповідно збільшується або зменшується на 1, а перед введенням значення байта не потрібно натискати "Запис". Якщо поточна адреса була встановлена за допомогою "Уст. АД.", то перед введенням значення байта потрібно натиснути "Запис".

2. **Перегляд та редагування вмісту регістрів процесора**. Вміст регістрів процесора відображується в правій верхній частині вікна. Для зміни вмісту регістра необхідно двічі клацнути на значенні регістра. При цьому в 5-6 розрядах дисплея виводиться назва регістра (наприклад, для акумулятора – "А"), а в 7-8 розрядах – вміст регістра. Далі слід ввести з клавіатури новий вміст регістра, який одночасно відображується в правій верхній частині вікна та в 7-8 розрядах дисплея.

3. **Запуск програми на виконання.** Для запуску програми слід встановити поточну адресу, з якої починається виконання програми, та натиснути "Пуск". Якщо встановлений режим покрокового виконання програми, то при кожному натисканні "Пуск" виконується одна команда програми. Після виконання команди на дисплей виводиться: в розрядах 1-4 – адреса наступної команди; в розрядах 7-8 – вміст відповідного байта пам'яті. Якщо встановлений режим автоматичного виконання, то після натискання "Пуск" дисплей вимикається та керування передається програмі користувача.

Для переривання виконання програми необхідно натиснути "Сброс". Незалежно від режиму виконання програми, зміна вмісту ОЗП та регістрів відразу відображується на екрані.

4. **Використання портів вводу-виводу.** Доступ до портів вводу-виводу здійснюється за допомогою команд IN та OUT. Порт вводу, який з'єднаний з 3 перемикачами K1-K3, має адресу FA. Порт виводу, який з'єднаний з 8 світлодіодами VD0-VD7, має адресу F9. Перед використанням портів F9, FA необхідно вивести в порт FB байт 81_{16}.

Література

1. Локазюк В.М. Мікропроцесори та мікроЕОМ у виробничих системах, - Київ.- Академія, 2002
2. Олссон Густав, Пиани Джангуидо. Цифровые системы автоматизации и управления. – Санкт-Петербург – Невский диалект, 2001
3. Дирксен А., ред. МикроЭВМ. – Москва. – Энергоиздат, 1982.
4. Токхайм Р. Микропроцессоры. – Москва. – Энергоатомиздат, 1987.
5. Лихтциндер Б.Я., Кузнецов В.Н. Микропроцессоры и вычислительные устройства в радиотехнике. – Киев. – Выща школа, 1988.
6. Погорелый С.Д., Слободянюк Т.Ф. Программное обеспечение микропроцессорных систем – Киев. – Тэхника, 1985.

Зміст