МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Запорізький національний технічний університет

Факультет післядипломної освіти

“Мікропроцесори в системах обробки даних та управлінні”

Програма, методичні вказівки
до лабораторних робіт і самостійної роботи

слухачів спеціальності 8.080403

“Програмне забезпечення автоматизованих систем”

2002

Мікропроцесори в системах обробки даних та управлінні. Програма, методичні вказівки до лабораторних робіт і самостійної роботи слухачів спеціальності 8.080403 “Програмне забезпечення автоматизованих систем” /Укл.: С.К.Корнієнко, Л.П.Скачко. – Запоріжжя: ЗНТУ, 2002. - 66 с.

Укладачі:

С.К.Корнієнко, доцент, к.т.н.,
Л.П.Скачко, асистент

Рецензент: Рисіков В.П., доцент, к.т.н

Відповідальний за випуск: С.К.Корнієнко

Затверджено
на засіданні кафедри КВР

Протокол № 1 від 17.09.2001

Методичні вказівки, призначені для використання на лабораторних заняттях із дисципліни "Мікропроцесори в системах обробки даних та управлінні", а також у процесі самостійної роботи слухачів факультету післядипломної освіти.

Методичні вказівки окрім указівок із виконання лабораторних робіт містять програму дисципліни, основні відомості з архітектури 8-розрядного мікропроцесора I8080 та практичні рекомендації з його програмування.

ЗМІСТ

1 ПРОГРАМА КУРСУ

1.1 Загальна характеристика мікропроцесорних засобів

Основні поняття цифрової схемотехніки. Прилади з "жорсткою" та програмованою логікою. Основні визначення курсу. Класифікація мікропроцесорних засобів. Принципи побудови МПС. Поняття архітектури. Фоннеймановська та гарвардська архітектура.

1.2 Архітектура однокриштального мікропроцесора I8080

Блок регістрів. Організація стека. Блок АЛП. Слово стану. Системний контролер. Машинні цикли та такти. Виконання команд.

1.3 Загальні питання побудови мікропроцесорних систем

1.3.1 Система пам’яті МПС. Основні характеристики запам’ятовуючих пристроїв (ЗП). Статичні та динамічні оперативні запам’ятовуючі пристрої (ОЗП). Постійні запам’ятовуючі пристрої (ПЗП). Організація кеш-пам’яті. Зовнішні запам’ятовуючі пристрої. Магнітні, компактні та цифрові відеодиски.

1.3.2 Інтерфейси МПС. Призначення. Основні принципи організації. Класифікація. Інтерфейси периферійних пристроїв. Інтерфейси ПЕОМ. Апаратні засоби інтерфейсу. Шинні формувачі. Порти вводу-виводу.

1.3.3 Режими обміну інформацією. Протоколи обміну. Арбітраж. Програмний обмін. Обмін інформацією в режимі переривань. Прямий доступ до пам’яті.

1.4 Організація МПС на базі сімейства iAPX86

1.4.1 Архітектура МП i8086/8088. Операційні пристрої. Шинний інтерфейс. Сегментація пам’яті. Логічні та фізичні адреси. Система переривань. Мінімальний та максимальний режими роботи. Система команд Керування ресурсами.

1.4.2 Особливості МП i80286, i80386, i80486, Pentium, Pentium Pro, Pentium MMX, Pentium II.

1.5 RISC-архітектура

Особливості RISC-архітектури. Трансп'ютери.

1.6 Сучасний стан та перспективи розвитку мікропроцесорної техніки

2 ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

2.1 Архітектура типової системи на базі мікропроцесора i8080

Функції, які реалізуються мікропроцесором, визначаються його структурою (сукупністю елементів, які складають МП, та зв'язків між ними) і послідовністю управляючих слів (команд), що надходять із запам'ятовуючого пристрою на входи МП. Для комплексної характеристики можливостей МП використовують поняття архітектури.

Архітектура МП – це його логічна організація, яка визначається можливостями МП з апаратної та програмної реалізації функцій, необхідних для будування мікро-ЕОМ (МПС). Поняття архітектури МП відображає його структуру, способи звертання до всіх доступних для користувача елементів структури, а також систему команд, режими адресації, формати та довжину даних, тобто архітектура будь-якої обчислювальної системи – це її представлення з точки зору користувача.

Структурна схема ЦПЕ представлена на рисунку 2.1. В ній виділені доступні користувачу регістри. МП містить шість 8-розрядних регістрів загального призначення (РЗП) (***B, C, D, E, H, L***), регістр-акумулятор, чотири 8-розрядних буферних регістра (***БР1, БР2, W, Z***), 8-розрядний регістр ознак (***F***), 8-розрядний арифметико-логічний пристрій (***АЛП***); 16-розрядні спеціалізовані регістри: лічильник команд (***ЛК***), покажчик стека (***ПС***). Окрім цього, до складу МП входять: регістр команд (***РК***), буфер даних (***БД***), буфер адреса (***БА***), схема вибірки регістрів (***СВР***), схема десяткової корекції (***СДК***), схема інкремент-декремент (***СІД***), дешифратор команд (***ДШК***), схема керування машинним циклом (***СКМЦ***) та пристрій управління (***ПУ***). Більш детальніший опис складових частин ЦПЕ наводиться у літературі.

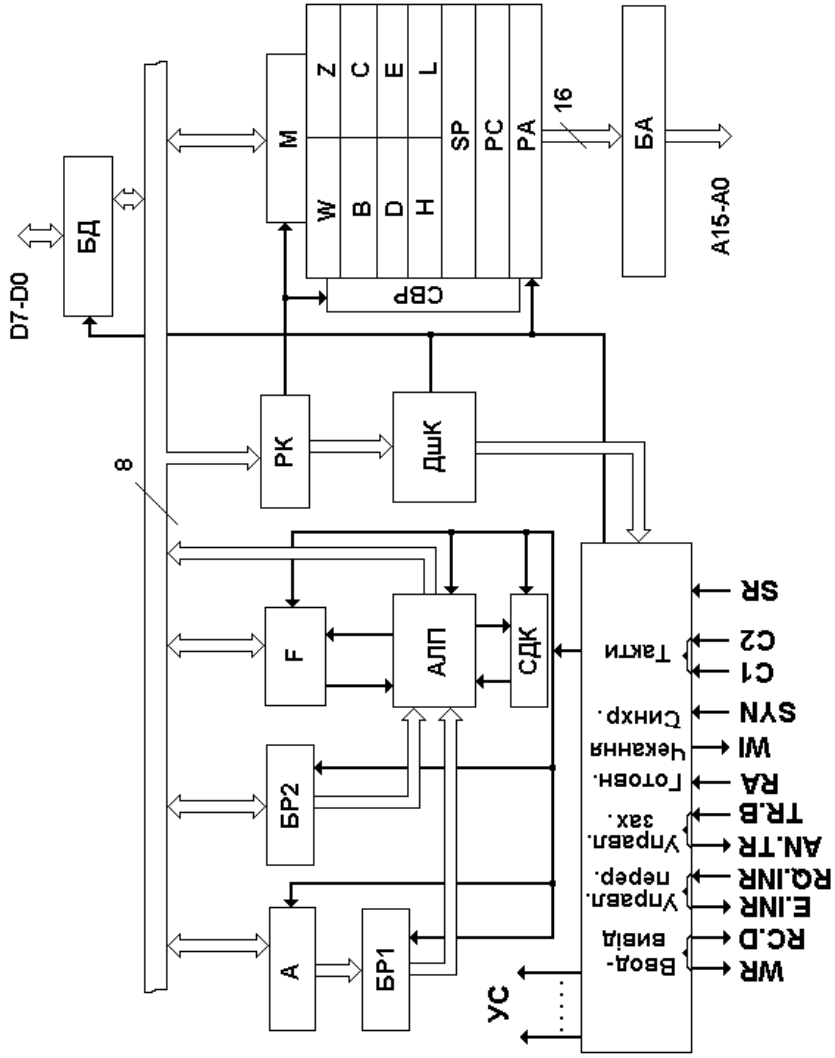


Рисунок 2.1 - Структурна схема ЦПЕ I8080

Внутрішні регістри мають наступне призначення:

A - регістр-акумулятор або накопичувач, призначений для збереження результату, а також здійснення вводу та виводу даних по командах ***IN*** та **OUT**;

РЗП - регістри загального призначення (**У**, **С**, **D**, **E**, **H**, **L**), які виконують роль надоперативної пам'яті ЦПЕ; можуть використовуватися як окремо, так і парами (тобто в якості 16-розрядних регістрів). В операціях звернень до пам'яті парний регістр HL використовується особливо: у ньому зберігається (записується) адреса ОЗП, по якій відбувається звернення;

PC - 16-розрядний програмний лічильник (лічильник команд), який визначає адресу ОЗП чи ПЗП, по якій зберігається чергова команда;

SP - 16-розрядний покажчик стека, який визначає поточну адресу вершини стека в ОЗП;

F - 8-розрядний регістр ознак (прапорців), розряди (прапорці) якого відповідають різноманітним ознакам і використовуються при виконанні операцій умовних переходів, умовного виклику підпрограм та умовного повернення з підпрограм. В результаті виконання окремих команд ознаки можуть змінюватися (встановлюватися в одиницю).

Структурна схема МПС представлена на рисунку 2.2 й містить мінімальну сукупність елементів, які повинні бути у складі будь-якої мікро-ЕОМ.

Типова МПС складається з наступних основних блоків:

- центрального процесорного елемента (**ЦПЕ**) I8080;
- системного контролера (**СК**);
- генератора тактових імпульсів (**ГТІ**);
- постійного запам'ятовуючого пристрою (**ПЗП**);
- оперативного запам'ятовуючого пристрою (**ОЗП**);
- портів уводу;
- портів виводу.

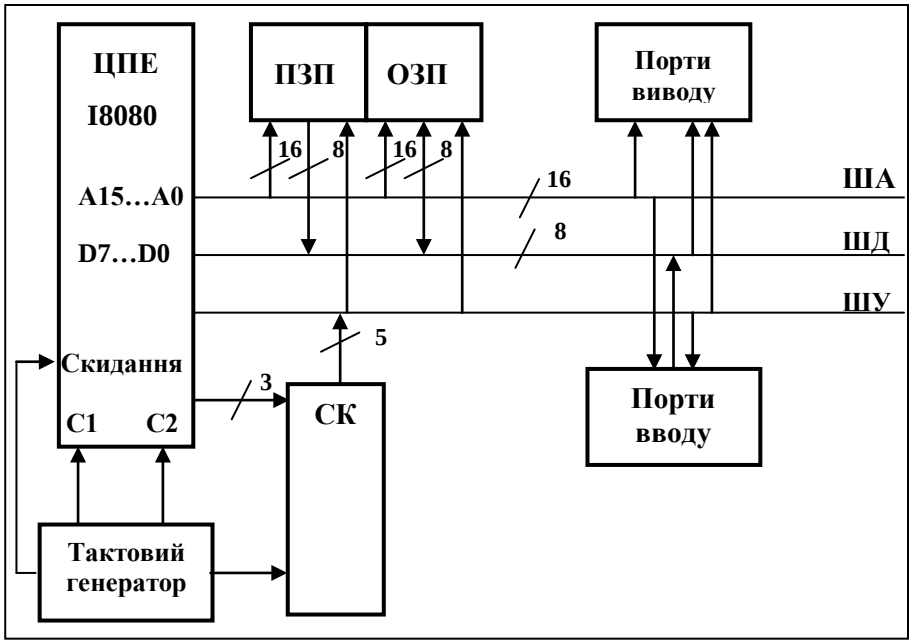


Рисунок 2.2 - Структурна схема мікропроцесорної системи

2.2 Типи і формати даних

Типи й формати даних, які обробляються мікропроцесором (МП), показані на рисунку 2.3.

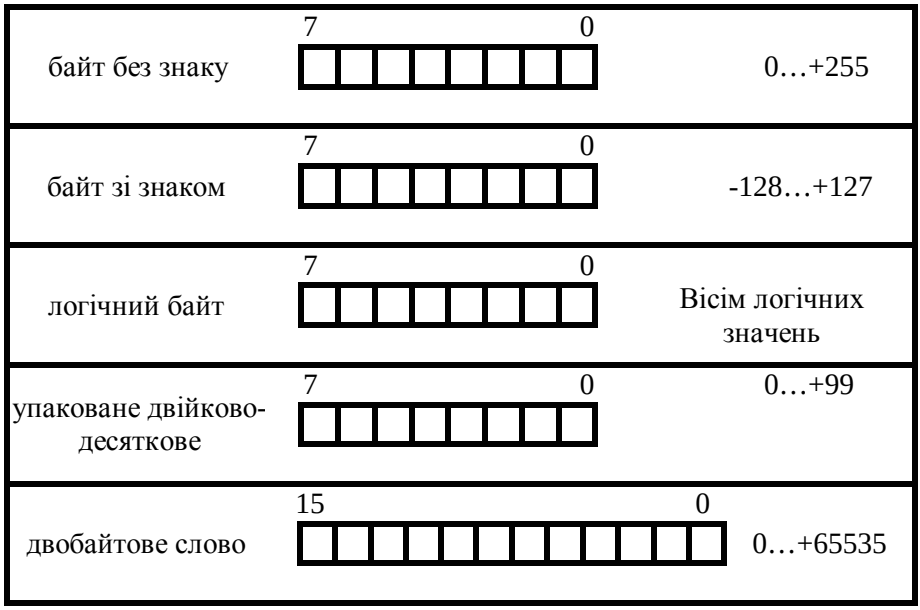


Рисунок 2.3 - Типи і формати даних

Над двійковими байтами виконуються операції складання, віднімання, прирощення, зменшення на одиницю; над двійково-десятковими - це операції складання, прирощення на одиницю. Над логічними даними виконуються логічні операції **І**, **АБО**, **ВИКЛЮЧНЕ АБО**, **НІ**, циклічні зсуви. Над двобайтовими словами виконуються арифметичні операції складання, прирощення й зменшення на одиницю. Арифметичні операції виконуються за правилами двійкової арифметики над числами у додатковому коді. Двійково-десяткові числа складаються у двійковому коді, а потім за допомогою операції корекції код результату із двійкового перетворюється у двійково-десятковий. Логічні операції виконуються за правилами двійкової логіки.

За результатами виконування операцій формуються ознаки: знаку **S**, нульового результату **Z**, переносу **C**, додаткового перенесення **AC** і парності **P**. Ознаки використовуються в якості умов управління обчислювальними процесами. Для їхнього зберігання використовується спеціальний регістр **F**, який зветься ***регістром ознак або прапорців*** (рисунок 2.4).

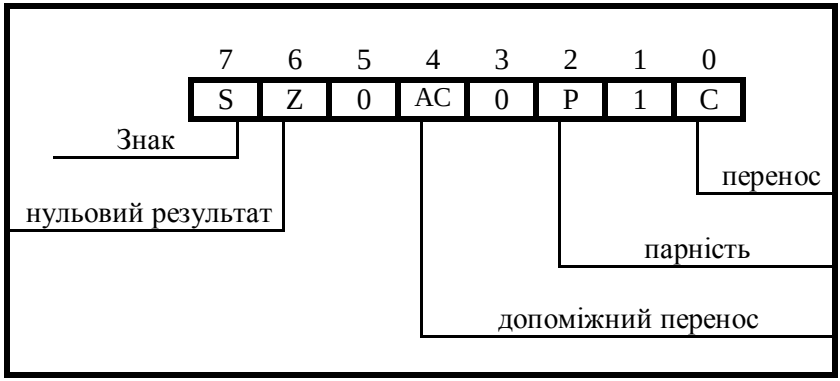


Рисунок 2.4 - Формат регістру ознак F

Ознаку переносу можна використовувати при виконанні операцій складання (віднімання) в якості додаткового додатка (від'ємника), який характеризує результат попередньої операції складання (віднімання). В операціях зсуву ознака переносу **C** дозволяє збільшити розрядність зсувного слова на один біт.

2.3 Адресний простір

Адресний простір МП складають простір запам'ятовуючих пристроїв (ЗП), зовнішніх пристроїв (ЗвП) і надоперативного запам'ятовуючого пристрою (НОЗП). Простір ЗП логічно та фізично організований у вигляді послідовності байтів об'ємом 64 Кбайт (рисунок 2.5), які адресуються 16-розрядними словами.

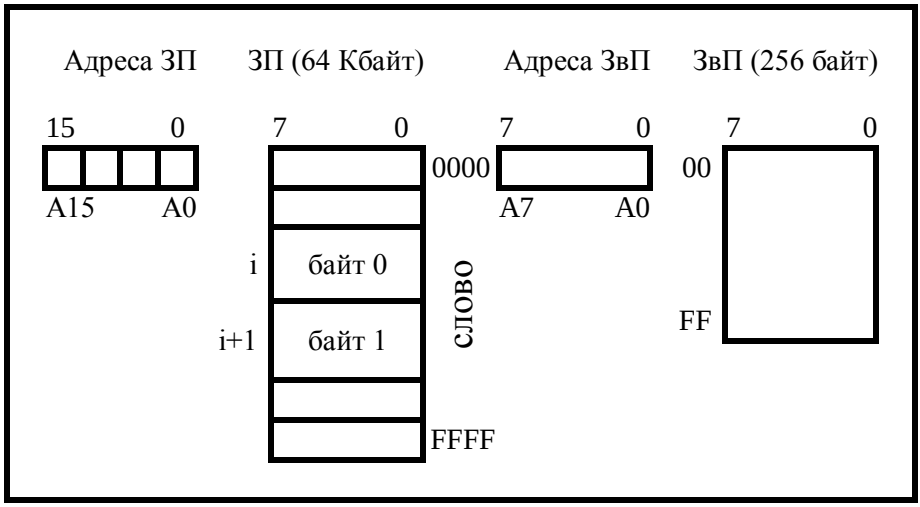


Рисунок 2.5 - Склад та організація адресного простору ЗП та ЗвП

Двобайтові слова розташовуються у порядку зростання адрес пам'яті та адресуються по молодшому байту. В адресному просторі розташовуються ***ОЗП*** і ***ПЗП***, де зберігаються програми і дані. В будь-якому місці простору можна організувати стек із послідовним доступом за принципом **LIFO (Last In - First Out)** об'ємом до 64Кбайт і логічною організацією 32К×16. Стек адресується 16-розрядними двійковими словами, які вказують поточну ***вершину стека*** - адресу старшого байта останнього завантаженого в стек слова.

Зовнішні пристрої розташовуються в окремому просторі об'ємом 256 байт, який адресується 8-розрядним адресним словом.

Внутрішній надоперативний запам'ятовуючий пристрій (***НОЗП)***

мікропроцесора I8080 (рисунок 2.6) уявляє собою статичну пам'ять з довільним доступом, організовану як шість 16-розрядних регістрів.

Три регістри загального призначення (**BC**, **DE**, **HL**) можна адресувати побайтно або як регістрові 16-розрядні пари послівно. У першому випадку регістри використовуються для зберігання байтових даних, у другому - для зберігання адрес або двобайтних слів даних.

Старший байт регістру **слова стану програми PSW (Program Status Word)** використовується у якості акумулятора, молодший байт - для зберігання ознак.

16-розрядний **покажчик стека SP (Stack Pointer)** призначається для зберігання адреси ***вершини стека***.

16-розрядний **програмний лічильник PC** використовується для природної адресації команд. Примусова адресація команд здійснюється шляхом використання команд безумовного та умовних переходів та команд безумовного та умовних викликів підпрограм.

Рисунок 2.6 - Організація НОЗП

2.4 Формат команд

Мікропроцесор I8080 - одноадресний. Команди мають одно-, двох- або трибайтовий формат (рисунок 2.7). Однобайтовий формат використовується для кодування команд звертання до регістрів НОЗП, непрямого адресування ЗП, при роботі зі стеком. Двох- та трихбайтовий формати використовуються для команд із безпосередньою та прямою адресацією.

У другому байті двобайтового формату вказується байт даних або 8-розрядна адреса зовнішнього пристрою, а в трибайтовом форматі у другому та третьому байтах вказується двобайтове слово даних або 16-розрядна адреса ЗП.

Багатобайтові команди зберігаються у сусідніх чарунках пам'яті та адресуються по першому байту, причому двобайтові слова даних та адрес розташовуються у порядку зростання.

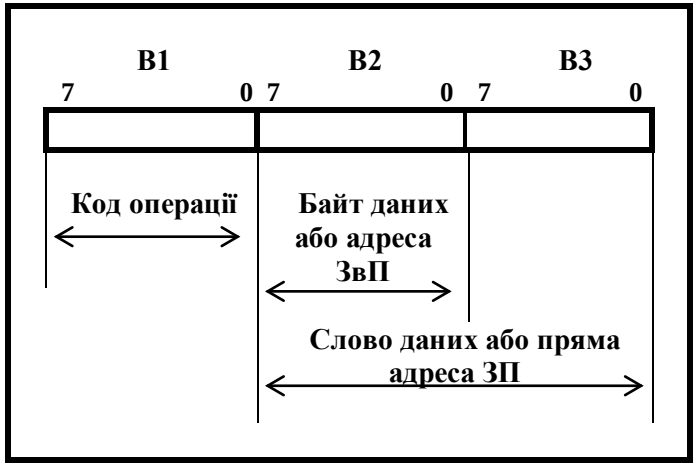


Рисунок 2.7 - Формати команд МП I8080

2.5 Методи адресації

В МП І8080 використовуються наступні методи адресації:

- неявна;
- пряма;
- безпосередня;
- непряма (опосередкована).

2.5.1 Неявна адресація

При цьому методі адресації адреса модулю МПС, із яким працює дана команда, ніде не згадується та визначається кодом операції.

Неявна адресація використовується для :

– звертання до акумулятора. Наприклад, команда **CMA** (інвертувати зміст акумулятора);

– звертання до бітів регістру ознак, наприклад, **STC** (установити біт переносу);

– для адресації тригера переривань (команди заборони та дозволяння переривань).

2.5.2 Пряма адресація

При цьому методі адресації адреса елементу МПС, який використовується підчас виконання операції, задається у команді. Пряме адресування використовується для адресування :

1) пам'яті; в цьому випадку виконавча адреса записується у другому та третьому байтах команди, наприклад, **STA 1000H** - завантажити чарунку з адресою 1000Н умістом акумулятора;

2) регістрів зовнішніх пристроїв; у цьому випадку виконавча адреса записується у другому байті команди, наприклад, **OUT 05H** - завантажити регістр зовнішнього пристрою під номером 5 вмістом акумулятора;

3) регістрів і регістрових пар МП; у цьому випадку виконавча адреса записується у полі коду операції, наприклад, **MOV A, B** -

передати вміст регістру **B** до акумулятора.

2.5.3 Безпосереднє адресування

При цьому методі адресації адреса елементу, який використовується під час виконання операції МП, вказана безпосередньо у команді.

Безпосередня адресація використовується в операціях безпосереднього завантаження регістрів та регістрових пар, таких як **MVI B,05H, LXI H,0016H**.

2.5.4 Непряме адресування

При цьому методі адресації виконавча адреса попередньо формується в одній з регістрових пар **B, D, H**. Наприклад,

STAX D - записати вміст акумулятора згідно з адресою, розміщеною в регістровій парі **D-E**;

MOV M,A - записати вміст акумулятора згідно з адресою, розміщеною в регістровій парі **H-L.**

Різновидом непрямого адресування є адресація з автоіндексуванням, при якому значення адреси, яка зберігається у покажчику стека **SP**, автоматично збільшується або зменшується на 2 після звертання до стеку, наприклад, **PUSH PSW**.

Система команд МП I8080 наведена у додатку Б.

3 ПРОГРАМУВАННЯ МІКРОПРОЦЕСОРНИХ СИСТЕМ

3.1 Елементарні заходи програмування

Нижче приведені приклади застосування окремих команд та складання невеликих програм.

Приклад 1. Результат дії операції **DCR A**, якщо **(A)=00H**, складає **(A)=FFH**.

Приклад 2. Для якої мети можна використовувати команду **XRA**?

Відповідь :

а) для установки акумулятора в нульовий стан **(XRA A)** ;

б) для інвертування вмісту регістру **(MVI A, FFH; XRA B)**.

Приклад 3. Установити прапорці регістру ознаки згідно зі значенням числа, яке зберігається у чарунці **0456H:**

```
LDA A, 0456H
ANA A
```

Приклад 4. Отримати допоміжний код числа, яке зберігається в акумуляторі.

```
CMA
INR A
```

Приклад 5. Замінити число, яке зберігається у чарунці з адресою **0200H**, на те ж саме число у допоміжному коді :

```
LXI H,0200H
MOV A,M
CMA
INR A
MOV M,A
```

Приклад 6. Онулити 0-й та 5-й розряди вмісту регістру **B :**

```
MOV A,B
ANI, 11011110B; ( ANI DEH)
MOV B,A
```

Приклад 7. Установити в одиницю 2-й, 3-й та 7-й розряду вмісту регістру **D**:

```
MOV A,D
ORI, 01001100B; (  ANI, 4CH)
MOV D,A
```

Приклад 8. Перевірити стан нульового біта регістру **C**:

```
MVI A, 00000001B
ANA C
```

Якщо нульовий розряд регістра **C** дорівнює нулю, то прапорець **Z** регістру ознак установлюється в одиничний стан і до потрібного фрагменту програми можна перейти, наприклад, по команді **JZ.**

Приклад 9. Змінити вміст чарунки пам'яті з адресою **0400H** у розряді **D5**:

```
LXI H,0400H
MVI A,00100000B
XRA M
MOV M,A
```

Приклад 10. Завантажити у регістр ознак (тригер **C**) значення розряду **D5** регістру **D**:

Спосіб 1:

```
MOV A,D
RLC
RLC
RLC
```

Спосіб 2:

```
MOV A,D
RRC 4
RRC 3
RRC 2
RRC1
RRC 0
RRC C
```

Приклад 11. Відняти вміст чарунки пам'яті з адресою **AB9EH** із умісту чарунки пам'яті з адресою **AB9FH** та здійснити перехід до чарунки пам'яті з адресою **ADR**, якщо результат виявиться від'ємним:

```
LXI H,AB9FH
MOV A,M
DCRR L
SUB M
JM ADR
```

Приклад 12. Прийняти інформацію з порту з адресою **PORT1** та записати її в порт із адресою **PORT2**:

```
IN PORT1
        OUT PORT2
```

Приклад 13. Перевірити значення розряду **D3** порту **05H** та перейти до програми **A** у тому випадку, якщо **D3=0**, та до програми **B**, якщо **D3=1**:

```
IN 05H
ANI 08H      ; виділення розряду D3
JZ A
B:    ...    ; перша команда програми B
A:    ...    ; друга команда програми A
```

3.2 Організація циклів

Цикл організується за наступною схемою :

```
        MVI r,N     ; ініціалізація циклу, де N - кількість
повторень тіла циклу
LABEL:  ...
        ...         ; тіло циклу
        ...
        DCR R; зменшення лічильника на
                    одиницю
        JNZ LABEL   ; перевірка виходу з циклу
```

Якщо кількість повторень більше 256 D, належить організувати вкладені цикли або використати для організації лічильника регістрову пару **rp**.

> ***Увага!!!***
> *В останньому випадку після команди **DCX rp** не можна одразу використати команду тестування регістру ознак, оскільки команда **DCX rp** не модифікує прапори.*

3.3 Програмна обробка масивів

***Приклад** 1*. Маємо масив чисел з адресою 04FFH. Потрібно скласти програму складання цих чисел одне з одним до появи ознаки перенесення. Адреса останнього числа, яка приймає участь у складанні, повинна бути видана на порти з адресами 04H та 05H.

```
        LXI H,04FFH
        MOV A,M
AGAIN:  INX H
        ADD M
        JNC AGAIN
        MOV A,H
        OUT 04H
        MOV A,L
        OUT 05H
        HLT
```

***Приклад** 2*. Маємо масив чисел з адресою 13ECH довжиною 25D. Підрахувати кількість парних елементів у масиві.

```
  LXI H,13ECH ; задають початкову адресу масиву
  MVI C,19H   ; ініціалізація циклу
  MVI B,0H    ; ініціалізація лічильника парних елементів
LAB:MOV A,M   ; вибірка числа з масиву
  ANI 01H     ; виділення 0-го біта
  JNZ L1      ; перехід на вибір наступного числа
```

```
INR B          ; збільшення лічильника парних елементів
L1:INX H       ;формування адреси наступного елементу масиву

DCR C          ; перевірка закінчення циклу
JNZ LAB
HLT
```

3.4 Робота з підпрограмами

Організація підпрограм ставить деякі проблеми використовування внутрішніх регістрів мікропроцесора. Правильно складена підпрограма не повинна змінювати жодного з регістрів, які будуть потрібні головній програмі після виконання підпрограми.

Таким чином, вміст регістрів, виконаних підпрограмою, необхідно тимчасово запам'ятати, з тим, щоб після виконання підпорами його можна було відновити без змін. Найбільш зручною пам'ятаю для тимчасового запам'ятовування вмісту регістрів є стек. Запам'ятати вміст регістрів можна у головній програмі перед викликом підпрограми, або у самій підпрограмі.

Практика програмування показує, що доцільно завжди запам'ятовувати вміст регістрів на початку і відновити його наприкінці підпрограм.

Наприклад, якщо програма **SUBR** використовує регістри **A**, **B** і **C**, вона повинна мати такий вигляд :

```
SUBR:
PUSH PSW       ; Запам'ятовування вмісту регістрів до стека
PUSH B
...                    ; Команди підпрограми
...
...
POP B          ; Відновлення колишнього вмісту регістрів
POP PSW
RET                    ; Повернення з підпрограми
```

У деяких підпрограмах зустрічаються команди умовного повернення. При явності команд **PUSH** і **POP** такі підпрограми повинні бути трохи модифіковані. Нехай, наприклад, заключний фрагмент підпрограми має такий вигляд :

```
...

DCR A
RZ
INX B
        RET
```

Коли у підпрограмі вимагається тимчасове запам'ятовування та відновлення вмісту регістрів, цей фрагмент трохи змінюється :

```
SUBR:   PUSH PSW
        PUSH H
          ...
        DCR A
        JZ NOT
        INX B
NOT:    POP H
        POP PSW
        RET
```

3.5 Організація переривань

Обслуговування вводу/виводу по перериванням є альтернативною програмно-керуємому обміну. Якщо при чисто програмному керуванні як початок процедури, так і безпосередньо її виконання знаходиться під керуванням програми, то обслуговування по перериванням ініціюється апаратними засобами. Сукупність цих засобів, команд і програм їхнього обслуговування зветься системою переривань.

Процес обробки запиту на переривання значно подібний процесам виклику та повернення з підпрограм. Але у випадку переривань виклик здійснюється командою **CALL,** яка формується та підставляється у загальну команду послідовність за допомогою апаратури системи переривань. З цієї причини запити на переривання часто називають

апаратними викликами підпрограм.

Усі запити на переривання можуть бути заборонені або дозволені одночасно за допомогою команд **DI** та **EI**, відповідно.

Для прискорення процедури обробки переривань застосовується спеціальний скорочений варіант команди **CALL adr m- RST n**, де **adr=8n, n=0-7.**

Використання команди **RST n** припускає резервування перших 64 (8×8) байтів пам'яті під таблицею входів у підпрограми обслуговування переривань (таблиця).

Зазвичай за адресами **8n, n = 0-7**, розташовані команди **JMP**, передаючи керування на підпрограми обслуговування переривань. Модифікування адресної частини команд **JMP** дозволяє оперативно здійснювати входи у підпрограми.

Таблиця 3.1 - Розподіл адрес входів до підпрограм обслуговування переривань

Номер області ЗП	Початкова адреса	Номер області ОЗП	Початкова адреса
0	0000h	4	0020h
1	0008h	5	0028h
2	0010h	6	0030h
3	0018h	7	0038h

При виконанні команди **RST** також, як і у команді **CALL,** адреса повернення запам'ятовується у стеку. Стан програмно-приступних регістрів може бути збережено у пам'яті, а потім відновлено безпосередньо перед поверненням у перервану програму. Цей процес називається контекстним перемиканням і виконується як програмними, так і апаратними засобами.

Повернення із підпрограм обслуговування переривань звичайно, виконується за допомогою командної послідовності

EI ; **INTE<= 1** (дозволяння переривання)

RET

4 ЛАБОРАТОРНІ РОБОТИ

Лабораторна робота №1 “Вивчення архітектури типової МПС на базі мікропроцесора I8080”

1 Мета роботи

Метою роботи є ознайомлення з архітектурою мікропроцесора I8080 та типової мікропроцесорної системи на його базі, а також отримання навичок практичної роботи з програмою-емулятором, вивчення принципів складання та вводу найпростіших програм до оперативної пам’яті МПС та способу їхньої корекції.

2 Завдання до лабораторної роботи

2.1 Вивчити архітектуру мікропроцесора i8080 і мікропроцесорної системи на його основі.

2.2 Вивчити структуру емулятора мікропроцесорної системи (ЕМПС) та призначення його основних команд.

2.3 Увести програму та перевірити її виконання.

3 Методичні вказівки до виконання лабораторної роботи

3.1 Вказівки до виконання пп. 2.1 та 2.2 завдання.

Для виконання пп. 2.1 та 2.2 завдання необхідно ознайомитись з розділом 2.1, конспектом лекцій, а також призначенням основних команд ЕМПС (додаток А).

3.2 Вказівки до виконання п. 2.3 завдання.

Для виконання п. 2.3 завдання необхідно отримати у викладача текст програми, ввід якої здійснити у такій послідовності.

3.2.1 Запустити програму ЕМПС, двічі клацнувши на ярлик програми.

3.2.2 Вибрати в пункті меню Файл підменю ***Новий***, щоб створити нове вікно для вводу програми. Основні команди ЕМПС описані у додатку А.

4 Зміст звіту

Звіт повинен мати наступні матеріали.
– мета роботи;
– структурна схема МПС;
– текст програми.

5 Контрольні запитання

5.1 Які типи МПС існують?
5.2 Укажіть склад і функції МПС.
5.3 Сформулюйте основні етапи створення програмної частини МПС.
5.4 Архітектура МП і8080.
5.5 Яка послідовність налагодження програми та її запуску?
5.6 Основні функції програми-емулятора МПС.

Лабораторна робота №2 “Вивчення принципів реалізації лінійних програм”

1 Мета роботи

Метою роботи є вивчення групи команд організації пересилань, арифметичних операцій, вводу-виводу даних, роботи з ОЗП, організації роботи зі стеком та придбання практичних навичок у складанні та налагодженні простих лінійних програм.

2 Завдання на лабораторну роботу

2.1 Вивчити призначення та склад внутрішніх регістрів ЦПЕ та формат команд.

2.2 Вивчити позначення і зміст команд, що входять до групи команд пересилань даних, арифметичних операцій, вводу-виводу даних, звертання до ОЗП та роботи зі стеком.

2.3 Скласти та реалізувати програму складання (віднімання) двох чисел з використанням ОЗП.

2.4 Скласти та реалізувати програму обробки даних з використанням ОЗП.

2.5 Скласти та реалізувати програму обміну даними між регістрами з використанням стекової пам’яті.

Примітка:
при складанні програми використовувати різні методи адресації з метою порівнювального аналізу та вибору оптимального варіанту.

3 Методичні вказівки до виконання лабораторної роботи

Для виконання завдань лабораторної роботи необхідно передчасно ознайомитися зі змістом лабораторної роботи №1.

3.1 Вказівки до виконання п.2.1 завдання

Для виконання п.2.1 лабораторного завдання необхідно ознайомитися з розділами 2.1 та 2.2 методичних указівок.

3.2 Вказівки до виконання п.2.2 завдання

Виконання п.2.2 вимагає вивчення вказаних у завданні груп команд згідно табл.Б.1 Додатку Б.

3.3 Вказівки до виконання п.2.3 завдання

По отриманому у викладача завданню скласти, ввести та налагодити програму, користуючись вказівками попередньої лабораторної роботи, а також даними додатків А, Б.

3.4 Вказівки до виконання п.2.4 завдання

Виконання п.2.4 завдання здійснюється в послідовності, аналогічній п.2.3. При цьому звертання до конкретної чарунки ОЗП виконується за адресою, яка дорівнює змісту регістрової пари H-L.

3.5 Вказівки до виконання п.2.5 завдання

Реалізацію обміну між регістрами з використанням стека необхідно здійснювати в такій послідовності.

3.5.1 Сформувати стек, для цього до парного регістру SP завантажити константу, що дорівнює адресі чарунки ОЗП, яка є вершиною стека.

3.5.2 Виконати пересилку змісту РЗП до стека у необхідному порядку, а потім завантажити РЗП інформацією зі стека. В обміні між РЗП та стеком приймають участь пари регістрів. При цьому переміщення стекової інформації відбувається автоматично, та реалізується принцип стекової організації "останнім прийшов – першим вийшов".

3.5.3 Виконати перевірку змісту регістрів до та після обміну інформацією.

4 Зміст звіту

Звіт повинен включати наступні матеріали:

4.1 мету роботи;
4.2 тексти програм по пп.2.3, 2.4, 2.5;
4.3 схеми алгоритмів програм;
4.4 таблиці змісту регістрів загального призначення на кожному кроку виконання програми;
4.5 аналіз отриманих результатів та висновки.

5 Контрольні запитання

5.1 Назвіть склад внутрішніх регістрів ЦПЕ.
5.2 Що таке формат команди?
5.3 Яке призначення окремих груп команд?
5.4 Який зміст основних команд?
5.5 Як виконується ввід (вивід) інформації у МПС?
5.6 Як здійснюється адресація чарунок ОЗП?
5.7 Що таке стек, яким чином він організується?

Лабораторна робота №3 “Вивчення принципів реалізації розгалужених та циклічних програм”

1 Мета роботи

Метою роботи є вивчення групи команд, які використовуються при організації переходів, формуванні лічильників, операціях зсувів та надбання практичних навичок у використанні регістра ознак при складанні й налагодженні розгалужених та циклічних програм.

2 Завдання на лабораторну роботу

2.1 Вивчити призначення та склад регістра ознак.

2.2 Вивчити позначення і зміст команд, що входять до групи зсувів, переходів та інкремент-декремент.

2.3 Скласти та реалізувати програму обчислення складної функції.

2.4 Скласти та реалізувати програму формування часових інтервалів.

3 Методичні вказівки щодо виконання лабораторної роботи

Для виконання завдання по даній роботі необхідне попереднє ознайомлення зі змістом лабораторних робіт №1, 2.

3.1 Вказівки до виконання пп.2.1 та пп.2.2 завдання

Для виконання пп.2.1 та пп.2.2 необхідно вивчити розділ 2.1 та вказані у завданні групи команд по таблиці Б.1 Додатку Б. При цьому потрібно зосередити особливу увагу на графу 3 таблиці Б.1, в якій наведено час виконання різноманітних команд у машинних тактах. Час одного машинного такту ***t =0,5 мкс.***

3.2 Вказівки до виконання п.2.3 завдання

За отриманим у викладача завданням скласти схему реалізації

заданої функції. Для подальшої роботи потрібно виділити конкретний спосіб організації розгалуження програми, користуючись додатком А. Після цього можна приступити до написання тексту програми. Текст програми рекомендується оформляти у вигляді таблиці, в якій повинні бути передбачені графи для адреси ОЗП(ПЗП), шістнадцяткового коду команди, мітки, мнемонічних позначень команд та коментарів.

Ввід тексту програми, налагодження, ввід початкових даних і отримання результатів здійснюється у відповідності з методикою, викладеною у попередніх роботах.

Результати обчислень у точках, вказаних викладачем, а також кінцевий результат необхідно фіксувати в протоколі виконання лабораторної роботи.

3.3 Вказівки до виконання п.2.4 завдання

Для виконання п.2.4 завдання необхідно отримати у викладача завдання, у відповідності з яким визначити варіант реалізації алгоритму (див. додаток Б). Розробити необхідні розрахунки, враховуючи час виконання команд (див. графу 3 табл.2.1) і варіанти реалізації. Скласти детальну схему та текст програми у відповідності зі вказівками по п.3.2.

4 Зміст звіту

Звіт повинен мати наступні матеріали.

4.1 Мету роботи.

4.2 Завдання, схему алгоритму та текст програми, результати розрахунків за п.2.3 завдання.

4.3 Завдання, необхідні розрахунки, схему, текст програми і результати експерименту за п.2.4.

4.4 Аналіз отриманих результатів та висновки.

5 Контрольні запитання

5.1 Що таке регістр ознак?

5.2 Як реалізуються команди умовних переходів?

5.3 Як організується лічильник циклів?

5.4 Яке призначення розгалужених та циклічних програм?

Лабораторна робота №4 “Дослідження принципів програмної реалізації логічних функцій”

1 Мета роботи

Метою роботи є вивчення групи команд логічних операцій і принципів програмної реалізації логічних функціональних вузлів.

2 Завдання на лабораторну роботу

2.1 Вивчити позначення та склад команд, що входять до групи логічних операцій.

2.2 Проаналізувати задану принципову схему цифрового пристрою, скласти логічну функцію і таблицю істинності.

2.3 Скласти схему алгоритму, що реалізує логічну функцію.

2.4 Скласти текст і налагодити робочу програму.

2.5 Виконати експериментальну перевірку таблиці істинності.

3 Методичні вказівки до виконання лабораторної роботи

Для виконання завдання по даній роботі необхідно освіжити в пам’яті основні відомості з цифрової техніки та булевої алгебри та виконати лабораторні роботи №1,2,3.

3.1 Вказівки до виконання п.2.1 завдання

Для виконання завдання за п.2.1 необхідно вивчити відповідний розділ таблиці Б.1 Додатку Б. Потрібно зосередити увагу на таблицю істинності логічної операції “виключне АБО” (команди ***XRA r, XRI***).

3.2 Вказівки до виконання п.2.2 завдання

Отримав у викладача принципову схему цифрового приладу, необхідно зрозуміти, які логічні функції повинні бути реалізовані, використовуючи правила булевої алгебри. Потім, використовуючи таблиці істинності основних операцій булевої алгебри, скласти таблиці істинності отриманої логічної функції (див. додаток Д).

3.3 Вказівки до виконання п.2.3 завдання

Складання схеми, що реалізує логічну функцію, необхідно виконувати, використавши таблиці істинності основних логічних операцій “**І**”, “**АБО**”, “**НІ**”. При цьому складання схеми алгоритму рекомендується починати зі здійснення перевірки на рівність одиниці кожної частини логічної функції (див. додаток Д).

3.4 Вказівки до виконання п.2.4 завдання

3.4.1 Розроблення програми маємо проводити таким чином, щоб після отримання кожної частини логічної функції прийняття рішення про рівність або нерівність одиниці відбувалося згідно зі станом того чи іншого прапорця регістра ознак.

3.4.2 Необхідно пам’ятати, що ввід інформації про значення незалежних змінних X_i здійснюється побайтно. Отже, потрібно замаскувати “зайві”, тобто не приймаючі участь у формуванні значення функції, розряди. У випадку, якщо цей байт інформації приймає участь у формуванні значень других частин функції, то його треба зберігати в одному з РЗП (див. додаток Д).

3.5 Вказівки до виконання п.2.5 завдання

Виконання п.2.5 завдання здійснюється за методикою, яка описана у попередніх роботах.

4 Зміст звіту

Звіт повинен мати наступні матеріали.

4.1 Мету роботи.
4.2 Завдання на лабораторну роботу.
4.3 Отриману логічну функцію, яка реалізує заданий пристрій.
4.4 Схему алгоритму.
4.5 Текст програми.
4.6 Таблиці істинності (теоретичну та експериментальну).
4.7 Аналіз отриманих результатів та висновки.

5 Контрольні запитання

5.1 Який зміст команд, що входять до групи логічних операцій? Таблиці істинності логічних операцій.

5.2 Яким чином складається таблиця істинності логічної функції?

5.3 У чому полягає принцип програмної реалізації логічних функцій?

5.4 Для чого і яким чином проводиться маскування?

5.5 Як здійснюється інвертування потрібних розрядів у байті інформації?

5.6 У чому перевага програмного способу реалізації логічних функцій у зрівнянні з апаратним?

Лабораторна робота №5 "Вивчення принципів реалізації програм управління зовнішніми пристроями"

1 Мета роботи

Метою роботи є вивчення прийому блочної реалізації програм із використанням команд ***CALL*** та ***RET*** на прикладі програми керування портами індикації.

2 Завдання на лабораторну роботу

2.1 Вивчити позначення та зміст команд ***CALL*** та ***RET*** за різноманітними умовами, що входять до групи команд переходів.

2.2 Скласти схему алгоритму, який дозволяє на вихідних портах отримати “бігаючі вогні”.

2.3 Реалізувати програму “бігаючі вогні” з керуванням із портів **00** і **01.**

2.4 Внести необхідні зміни у програму для сумісного використання індикаторів **00** і **01.**

3 Методичні вказівки до виконання лабораторної роботи

Для виконання завдання по даній роботі необхідно попередньо ознайомитися з виконанням попередніх лабораторних робіт.

3.1 Вказівки до виконання п.2.1 завдання

Для виконання п.2.1 лабораторного завдання необхідно вивчити групу команд переходів за таблицею Б.1 Додатку Б.

Потрібно звернути увагу, що трибайтна команда ***CALL*** є командою звернення до підпрограми, яка починається з деякої адреси. За командою ***CALL*** зміст лічильника команд центрального процесора, тобто адреса наступної команди, заноситься до стекової області ОЗП, а зміст покажчика стека зменшується на 2. Після операцій зі стеком до лічильника команд заноситься початкова адреса підпрограми, вказана другим та третім байтами команди ***CALL***, після чого починається її виконання.

У кінці підпрограми обов’язково повинна стояти команда ***RET***. Команда ***RET*** відновлює той стан процесора й стека, який був

безпосередньо перед виконанням команди ***CALL***. Зміст покажчика стека збільшується на 2, а до лічильника центрального процесора заноситься адреса, яка була передана для зберігання до стекової пам'яті по команді ***CALL***.

3.2 Вказівки до виконання п.2.2 завдання

Працююча програма повинна реагувати на такі керуючі впливи, які задаються шляхом установлення відповідних розрядів **Bi** порту **01** в одиницю (рисунок 5.1):

"***ПУСК-СТОП***", тобто при $\mathbf{B_7}$=1 – переміщення інформації на порту індикації (вогні "бігуть"), а при $\mathbf{B_7}$=0 – немає переміщення;

"***ВЛІВО-ВПРАВО***", тобто при $\mathbf{B_6}$=1 – переміщення інформації вліво, а при $\mathbf{B_6}$=0 – вправо;

"***ШВИДКІСТЬ***" – розряди $\mathbf{B_0}$-$\mathbf{B_5}$ повинні дозволяти в широких межах змінювати швидкість переміщення інформації (час переносу інформації на один розряд від 0,05с до 3-4с);

"***ВВІД ІНФОРМАЦІЇ***" – в положенні "***СТОП***" інформація, яка вводиться повинна задаватися з порту **00.**

3.3 Вказівки до виконання п.2.3 завдання.

Оскільки режим роботи програми визначають старші розряди порту **B** ($\mathbf{B_7}$ і $\mathbf{B_6}$), їхній стан доцільно аналізувати по прапорцю перенесення, здійснив попередньо зсув аналізуємого байта вліво.

Константа, що визначає час затримки між зсувами ("**ШВИДКІСТЬ**"), легко отримується в акумуляторі з використанням відповідної маски (див. лабораторну роботу №4).

"Бігаючі вогні" отримуються регулярним оновленням інформації на порту індикації. Час затримки визначається розрядами $\mathbf{B_0}$-$\mathbf{B_5}$ порту **B** (див. вище). Оновлення інформації полягає в її зсуві в акумуляторі на один розряд уліво або вправо. Після кожного зсуву інформацію необхідно запам'ятовувати в одному із РЗП і для наступного зсуву знов викликати акумулятор. Для спрощення процесу налагодження програми керування індикатором доцільно виділити дві підпрограми:

перша – підпрограма мінімальної затримки t_{min}=10мкс;

друга – підпрограма змінної затримки з кроком t_{min};

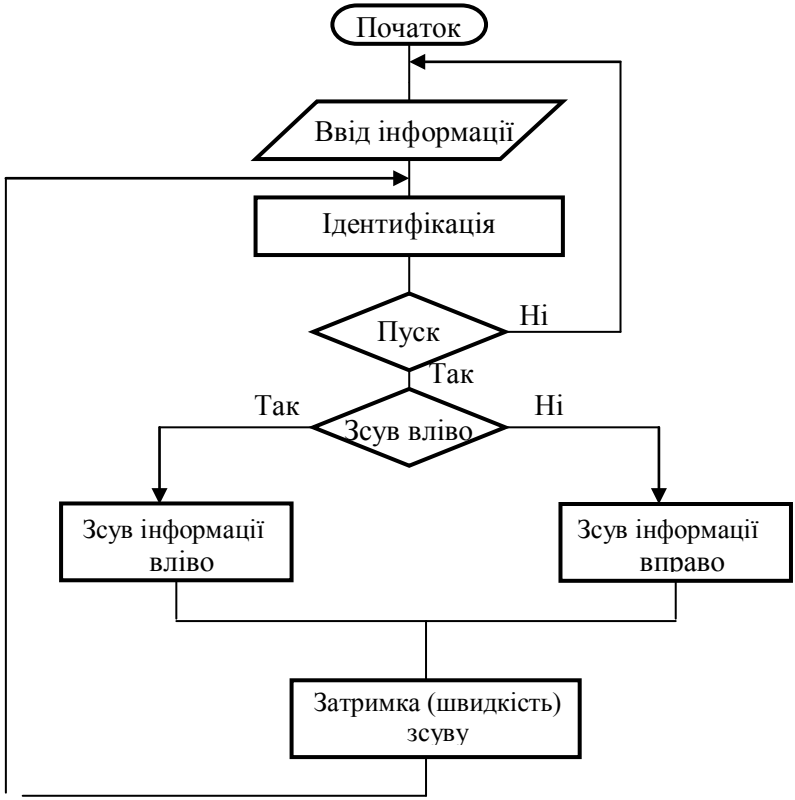


Рисунок 5.1 - Схема алгоритму керування портом виводу

3.4 Вказівки до виконання п.2.4 завдання

Зміни в програмі насамперед стосуються вводу інформації і її виводу на індикатори. При цьому треба пам'ятати, що операції ***IN*** та ***OUT*** використовують однобайтовий регістр-акумулятор. Зсув інформації в двобайтовому регістрі вліво доцільно здійснювати за допомогою команди ***DAD***, а вправо – комбінуючи команди зсуву через прапорець перенесення та пересилання з регістра до регістра.

4 Зміст звіту

Звіт повинен мати наступні матеріали.

4.1 Мету роботи

4.2 Позначення та зміст команд звернення до підпрограми і повернення з неї, команд циклічного зсуву.

4.3 Схему алгоритму програми керування.

4.4 Текст програми керування індикатором.

4.5 Текст змін програми для одночасного використання двох портів R та L .

4.6 Аналіз отриманих результатів та висновки.

5 Контрольні запитання

5.1 Який зміст команд ***CALL*** і ***RET***?

5.2 Який зміст команд циклічного зсуву?

5.3 Що таке адреса повернення? Де вона зберігається і як використовується?

5.4 Як може виконуватися аналіз окремих бітів слова?

5.5 У чому полягає різниця та схожість прийомів зсуву однобайтової та двобайтової інформації?

5.6 Як реалізується зміна величини затримки операцією з порту вводу?

Лабораторна робота №6 "Розробка програми-монітор"

1 Мета роботи

Метою роботи є отримання навичок складання програми керування мікропроцесорною системою на прикладі програми-монітор.

2 Завдання на лабораторну роботу

2.1 Вивчити схему алгоритму програми-монітор (додаток Ж).

2.2 Отримати завдання у викладача (набір функцій монітора та призначення розрядів керуючого порту).

2.3 Скласти та запустити програму на виконання.

3 Зміст звіту

3.1 Мета роботи.

3.2 Структурна схема програми-монітор.

3.3 Структурна схема робочої програми.

3.4 Текст програми.

4 Контрольні запитання

4.1 Основні функції, які виконує монітор.

4.2 Варіанти реалізації програми-монітор.

4.3 Призначення і формати команд, які використовуються у програмі-монітор.

4.4 Як можна оптимізувати розроблену програму?

ЛІТЕРАТУРА

1 Гилмор Ч. Введение в микропроцессорную технику. – М.: Мир, 1989.

2 Григорьев В.Л. Программирование однокристальных микропроцессоров. – М: Энергоатомиздат, 1987.

3 Лихтциндер В.Я., Кузнецов В.Н. Микропроцессоры и вычислительные устройства в радиотехнике. – К.: Вища школа, 1988.

4 Майоров В.Г., Гаврилов А.И. Практический курс программирования микропроцессорных систем. – М.: Машиностроение, 1989.

5 Проектирование микропроцессорной электронно-вычислительной аппаратуры: Справочник / В.Г. Артюхов, А.А. Будняк, В.Ю. Лапий и др. – К.: Техника, 1988.

6 Самофалов К.Г., Викторов О.В. Микропроцессоры. – К.: Техника, 1989.

7 Справочник по микропроцессорным устройствам / А.А. Молчанов, В.И. Корнейчук, В.П. Тарасенко. – К.: Техника, 1987.

8 Сташин В.В. Проектирование цифровых устройств на однокристальных микропроцессорах. – М.: Энергоатомиздат, 1990.

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

10 Шевкопляс Б.В. Микропроцессорные структуры. Инженерные решения: Справочник. – М.: Радио и связь, 1990

11 Щелкунов Н.Н., Дианов А.П. Микропроцессорные средства и системы. – М: Радио и связь, 1989.

ДОДАТОК А - ЕМУЛЯТОР МІКРОПРОЦЕСОРНОЇ СИСТЕМИ

Емулятор мікропроцесорної системи (ЕМПС) - це програмний комплекс, призначений для моделювання роботи МПС на ПЕОМ старших поколінь та навчання студентів основам програмування МПС на основі МП і8080. МПС, яка моделюється, складається з центрального процесорного елементу, ОЗП та портів уводу-виводу.

ЕМПС виконує наступні функції:

- запис програми користувача до ОЗП;
- внесення виправлень до програми;
- запуск програми користувача;
- зупинка у вказаних пунктах програми;
- можливість контролю стану процесора (РЗП, РС та SP), умісту ОЗП і портів уводу-виводу;
- можливість пуску програми з будь-якого місця.

Редактор ЕМПС

На рисунку А.1 наведений зовнішній вигляд редактора ЕМПС.

У верхній частині вікна вказується ім’я файлу, який редагується. Нижче розташований редактор, який утримує всі стандартні механізми, необхідні для редагування тексту (копіювання, вставка, вирізання тексту). Для пересування курсору можна використовувати як клавіатуру, так і мишку.

Команди редактора

Ctrl + Y - вилучання рядка, у якому знаходиться курсор.
Ctrl + C - копіювання виділеного фрагмента до буфера обміну.
Ctrl + V - уставити з буфера обміну.
Ctrl + X - вирізати виділений фрагмент до буфера обміну.
Ctrl + Z - відміна останньої операції.

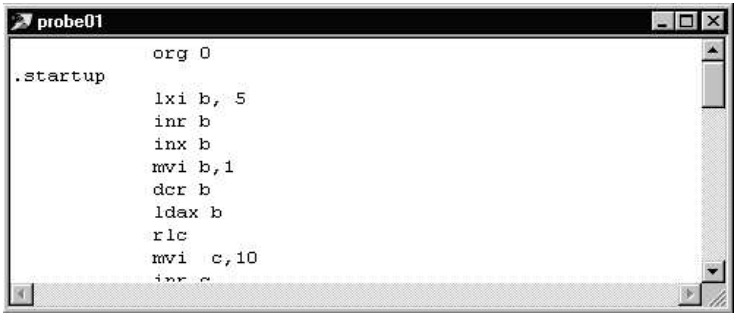


Рисунок А.1 - Редактор

Директиви компілятора

ЕМПС має ряд додаткових директив, необхідних для вірної організації коду. Розглянемо ці директиви:

ORG адреса - установлення адреси формування коду програми. Директив ORG у тілі програми може бути декілька.

мітка EQU значення - визначення констант. Мітці надається значення, вказане після директиви. При завданні значень констант можна використовувати як число, так і внутрішню змінну ЕМПС $, яка має значення поточної адреси. При використанні $, можна до $ додавати або віднімати деяке значення.

STARTUP - визначає точку входження до програми. У тілі програми може зустрічатися лише один раз.

INCLUDE "ім'я файлу - підключення до тіла програми зовнішнього файлу з текстом процедур або описів констант. Файл буде вставлений після описаної директиви. Мітки у файлах не повинні повторюватися, а також у допоміжних файлах не повинна зустрічатися директива ***.startup.***

[мітка] db значення - розміщує у пам'яті дані, які перелічені після директиви. Значення необхідно записувати через кому. Дані будуть розміщені послідовно, розмірність - байт.

[мітка] db N dup (значення) Заповнює ***N*** чарунок пам'яті заданим значенням. Якщо замість значення вказати "?", то ЕМПС зарезервує місце завдовжки ***N.***

Директива ***dw*** має такий синтаксис, як і ***db***, але розмірність - слово.

Значення внутрішньої змінної $ можна використовувати як константу при визначенні команд пересилки даних, а також у командах передачі управління. До $ можна як додавати, так і віднімати деяке значення, яке не повинне перевищувати 65535.

Для створення коментарів необхідно поставити спочатку зарезервований символ ";", після якого весь залишений рядок вважається коментарем.

Наладник ЕМПС

Наладник використовується для емуляції МПС i8080 (рис. А.2).

Стрілка вказує на поточну команду, яка підлягає виконанню.

ЕМПС дозволяє встановлювати контрольні точки, які дозволяють полегшити налагодження програми. Контрольна точка встановлюється (знищується) при натисканні клавіші ***F5.***

Для зміни формату виводу числової інформації (десятковий/шістнадцятковий) необхідно натиснути праву кнопку миші, та обрати запропонований формат уявлення. Випливаюче підменю містить ряд корисних функцій, які полегшують процес налагодження.

Для полегшення налагодження ЕМПС має ряд інспекторів, які будуть описані нижче.

Рисунок А.2 - Наладник

Команди наладника:

F7 - трасування за однією командою;
F8 - трасування за командою без входу до підпрограми;
F4 - запуск програми зі зупинкою в позиції курсору;
F5 - установлення контрольної точки;
F9 - запуск програми без відновлення вікна наладника;

Ctrl + F2 - скидання програми (установлення **PC** на початок програми).

Інспектор регістрів

Інспектор регістрів (рис. А.3) містить інформацію про регістри загального призначення, програмний лічильник, покажчик стека, а також прапорці МП.

Для зміни значення регістра необхідно двічі натиснути ліву кнопку миші або при натисканні правої кнопки обрати у випливаючому підменю "***Змінити...***". Після чого на екрані з'явиться вікно, у якому треба ввести нове значення для регістра або регістрової пари. За допомогою миші також можна змінювати значення прапорців. Після внесення змін при виконанні наступної команди вони вступають у силу.

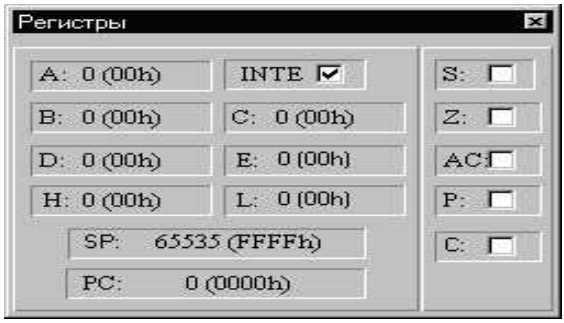


Рисунок А.3 - Інспектор регістрів

Інспектор тактів

Інспектор тактів дозволяє робити часові заміри, в міру виконання програми. Для скидання лічильника тактів необхідно натиснути праву кнопку на мишці та обрати пункт "***Скинути***", що викличе встановлення лічильника тактів у нульове значення.

Інспектор портів виводу

Інспектор портів виводу (рис. А.4) дозволяє відстежувати будь-який з портів виводу з діапазону від 0 до 255. Лівий стовпчик інспектора містить адреси портів, які відстежуються. Правий стовпчик містить значення портів, які відстежуються. Інспектор дозволяє переглядати значення портів виводу. Значення інтерпретується в трьох системах: десятковій, шістнадцятковій та двійковій. Активною вважається той рядок, який трішки втоплений.

Для активного рядка можливо проводити зміни адреси порту, який треба спостерігати. Для включення контролю за портом необхідно натиснути клавішу ***Ins*** або викликати випливаюче підменю натисненням правої кнопки миші, та обрати "***Додати***". Далі необхідно ввести адресу того порту, який треба відстежувати. Для відміни контролю за портом необхідно натиснути клавішу ***Del*** або обрати у випливаючому підменю "***Вилучити***". Для зміни адреси для вже створеного контролю необхідно натиснути ***Ctrl+Ins*** або у

випливаючому меню обрати "***Змінити***", після чого ввести нову адресу порту, який підлягає відстеженню.

Рисунок А.4 - Інспектор портів виводу

Інспектор портів уводу

Інспектор портів уводу (рис. А.5) дає можливість контролювати значення порту вводу. Зміну значення необхідно зробити за допомогою мишки, змінюючи потрібні біти значення порту. Зміни одразу ж інтерпретуються у десятковій та шістнадцятковій формі. За виключенням можливості редагування значення порту інспектор портів уводу нічим не відрізняється від інспектора портів виводу.

Рисунок А.5 - Інспектор портівуводуу

Інспектор пам’яті

Інспектор пам’яті (рис. А.6) дозволяє відстежувати послідовні області пам’яті. Лівий стовпчик містить початкові адреси областей, які підлягають контролю. Механізм налагодження інспектора нічим не відрізняється від вище описаних, але при включенні контролю необхідно вказати деяку додаткову інформацію. Для отримання цієї інформації ЕМПС виводить діалогове вікно, яке зображене на рисунку А.7.

У полі "***Адреса пам’яті***" вводиться адреса області, яка інспектується. Також необхідно обрати тип даних, які виводяться. По умовчанню встановлена розмірність один байт. Поле "***Розмірність***" задає довжину масиву, який інспектується , максимальна кількість елементів у масиві - 255.

Необхідно також обрати форму уявлення інформації. По умовчанню встановлено ввід у десятковому форматі. Це встановлення визначає формат виводу не тільки інформації чарунок пам’яті, але й представлення початкової адреси.

Рисунок А.6 - Інспектор пам’яті

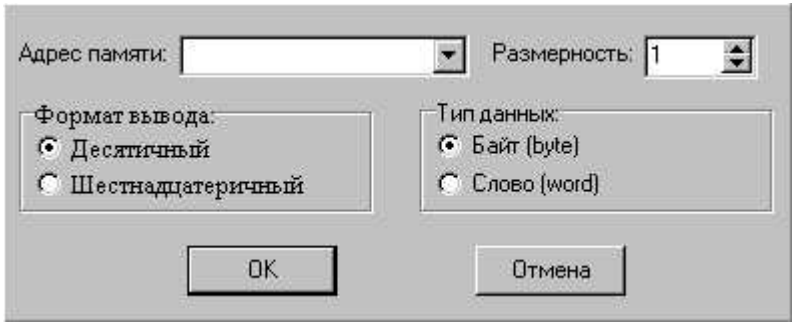


Рисунок А.7 - Додаткове вікно інформації

Протокол контрольних точок

Протокол контрольних точок (рис. А.8) містить інформацію про всі контрольні точки, які встановлені в ЕМПС. Інформація має наступну структуру: ім'я програми, яка виконується, та адреса. ЕМПС дає можливість вилучити та шукати контрольні точки.

Для вилучення контрольної точки необхідно:
- обрати контрольну точку у вікні;
- обрати кнопку "***Вилучити***";
- активізувати кнопку.

Для того щоб знайти контрольну точку, треба виконати наступні операції:

- обрати контрольну точку у вікні;
- обрати кнопку "***Знайти***";
- активізувати кнопку.

ЕМПС знайде потрібну виконувану програму та адресу розташування контрольної точки й виведе її у вікні наладника.

Кнопка "Закрити" використовується для того, щоб усунути протокол контрольних точок з екрана.

Рисунок А.8 - Протокол контрольних точок

Меню ЕМПС

Файл - підменю для роботи з файлами. У свою чергу складається з наступних пунктів:

Новий - створення нового файлу;

Відкрити - дає можливість відкривати вже існуючі тексти програм та скомпільовані програми для МП і8080;

Закрити - зачиняє активне вікно;

Зберегти - збереження тексту програми без зміни імені файлу;

Зберегти як - збереження тексту програми під новим ім'ям;

Вихід - вихід з ЕМПС;

Правка - підменю для праці з текстовим редактором. Містить такі процедури:

Відмінити операцію - відміна останньої операції у текстовому редакторі;

Вирізати - знищення виділеного фрагменту до буфера обміну;

Копіювати - копіювання виділеного фрагменту до буфера обміну;

Вставити - вставка з буфера обміну в позицію курсору;

Виділити все - виділення всього тексту програми;

Знайти - пошук зразка у тексті програми;

Замінити - заміна зразка на новий текст;

Знайти наступний - пошук наступного зразка у тексті програми;

Вид - підменю настройки ЕМПС. Містить такі пункти:

Регістри - показати (сховати) інспектор регістрів;

Такти - показати (сховати) інспектор тактів;

Інспектор виводу - показати (сховати) інспектор портів виводу;

Інспектор вводу - показати (сховати) інспектор портів вводу;

Інспектор пам'яті - показати (сховати) інспектор пам'яті;

Протокол контрольних точок - показати протокол контрольних точок;

Запуск - підменю компілятора та наладника ЕМПС:

Запуск - для редактора викликає компілятор, а потім передає управління наладнику, у якому викликає запуск програми;

Компіляція - компіляція тексту програми;

За командою - трасування програми в ***покроковому*** режимі за однією командою;

Без підпрограм - трасування за командою без входу до підпрограм;

Запуск до курсору - запуск програми із зупинкою в позиції курсору або в контрольній точці;

Рестарт програми - перезапуск програми та встановлення програмного лічильника на початок;

Контрольна точка - встановлення (знищення) контрольної точки в позиції курсору;

Вікно - підменю містить операції по настройці вікон:

Каскад - розташовує вікна каскадом;

Ланцюжок - розташовує вікна у ланцюжок, якщо вони не мінімізовані;

Зменшити все - мінімізує всі вікна;

Допомога - містить механізм роботи з довідковою системою.

ДОДАТОК Б – СИСТЕМА КОМАНД МІКРОПРОЦЕСОРА I8080

Система команд однокриштального мікропроцесора I8080 має команди трьох форматів. Перший байт команди містить інформацію про формат команди, код операції, вид адресації та про регістри або регістрові пари, якщо вони приймають участь у виконанні операцій.

У двобайтових командах другий байт (***B2***) містить 8-розрядний операнд або 8-розрядну адресу пристрою вводу чи виводу.

У трибайтових командах другий та третій байти (***B2***, ***B3***) містять 16-розрядні адреси (у командах із прямою адресацією пам'яті) або 16-розрядні операнди (у командах завантаження регістрових пар або покажчика стека). Другий байт трибайтової команди містить молодший байт числа, третій – старший.

У таблиці Б.1 наведені мнемонічні позначення та опис команд мікропроцесора i8080. При цьому використовуються наступні умовні позначення :

r - регістр загального призначення;
rp - пара регістрів загального призначення;
PC - програмний лічильник (лічильник команд);
SP - покажчик стека;
M - чарунка пам'яті;
B2,B3 - другий та третій байти команд

Таблиця Б.1 - Система команд мікропроцесора I8080

Команда	Довжина команди, байт	Число тактів	Опис команди	Ознаки
1	2	3	4	5
1 Команди пересилання даних				
MOV r1,r2	1	5	Пересилання даних із регістра **r**2 до регістра **r**1	-
MOV M,r (MOV r,M)	1	7	Пересилання даних із регістра **r** до пам'яті за адресою, що зберігається у регістровій парі **H-L** (із пам'яті до регістру **r**)	-
XCHG	1	4	Обмін даними між парами регістрів **H-L** і **D-E**	-
MVI r,<B2> (MVI M,<B2>)	2	7 (10)	Занесення байта даних до регістру **r** (до пам'яті)	-
LXI rp <2байта>	3	10	Занесення двох байтів даних у пару регістрів **(B-C, D-E, H-L, SP**). Третій байт команди заноситься в старший регістр, а другий - у молодший	-
LDAX rp	1	7	Завантаження в накопичувач вмісту чарунки, яка опосередковано адресується парою регістрів **rp (B-C, D-E)**	-
LDA <адреса>	3	13	Завантаження накопичувача вмістом чарунки за вказаною адресою. 2-й байт команди - молодший байт адреси, 3-й байт - старший	-
STAX rp	1	7	Занесення вмісту накопичувача до чарунки, яка опосередковано адресується парою **rp**	-
STA, <адреса>	3	13	Занесення вмісту накопичувача до чарунки за вказаною адресою	-

Продовження таблиці Б.1

1	2	3	4	5
LHLD <адреса>	3	16	Завантаження регістра **L** вмістом чарунки за вказаною адресою, а регістр **H** - чарунки з адресою на одиницю більше	-
SHLD <адреса>	3	16	Занесення вмісту регістрів **H** і **L** до пам'яті (аналогічно команді **LHLD**)	-
2 Арифметичні команди				
ADD r (ADD M)	1	4 (7)	Складання змісту регістра **r** (чарунки пам'яті) і накопичувача	Z,S,P,C
ADC r (ADC M)	1	4 (7)	Складання змісту регістра **r** (чарунки пам'яті) і накопичувача з бітом перенесення	Z,S,P, C,AC
SUB r (SUB M)	1	4 (7)	Віднімання змісту регістра **r** (чарунки пам'яті) від змісту накопичувача	Z,S,P, C^1,AC^2
SBB r (SBB M)	1	4 (7)	Віднімання змісту регістра **r** (чарунки пам'яті) та біта перенесення від змісту накопичувача	Z,S,P, C^1,AC^2
ADI ,<байт>	2	7	Складання байта зі змістом накопичувача	Z,S,P,C, AC
ACI ,<байт>	2	7	Складання байта зі змістом накопичувача та бітом перенесення	Z,S,P,C, AC
SUI,<байт>	2	7	Віднімання байта із змісту накопичувача	Z,S,P, C^1,AC^2
SBI,<байт>	2	7	Віднімання байта команди і біта перенесення від змісту накопичувача	Z,S,P, C^1,AC^2
DAD rp	1	10	Складання змісту пари регістрів **rp** (**B-C,D-E,H-L,SP**) зі змістом пари регістрів **H-L**	C
INR r (INR M)	1	5 (10)	Збільшення змісту регістра **r** (чарунки пам'яті) на одиницю	Z,S,P, AC
DCR r (DCR M)	1	5 (10)	Зменішення змісту регістра **r** (чарунки пам'яті) на одиницю	Z,S,P, AC^2

Продовження таблиці Б.1

1	2	3	4	5
INX rp (DCX rp)	1	5	Збільшення (зменьшення) змісту пари регістрів **rp** (**B-C,D-E,H-L,SP**) на одиницю	-
DAA	1	4	Перетворення змісту накопичувача в двійково-десятичний код	Z,S,P,C, AC
3 Логічні команди				
ANA r (ANA M)	1	4 (7)	Порозрядне 'І' над змістом регістра **r** (чарунки пам'яті) і накопичувача	Z,S,P, C=0, AC=0
XRA r (XRA M)	1	4 (7)	Порозрядне виключаюче 'АБО' над змістом регістра **r** (чарунки пам'яті) і накопичувача	Z,S,P, C=0, AC=0
ORA r (ORA M)	1	4 (7)	Порозрядне 'АБО' над змістом регістра **r** (чарунки пам'яті) і накопичувача	Z,S,P, C=0, AC=0
CMP r (CMP M)	1	4 (7)	Порівняння змісту регістра **r (**чарунки пам'яті) і накопичувача	(Z,S,P, C,AC)[3]
ANI,<байт>	2	7	Порозрядне 'І' над змістом накопичувача і байтом	Z,S,P, C=0, AC=0
XRI,<байт>	2	7	Порозрядне виключаюче 'АБО' над змістом накопичувача і байтом	Z,S,P, C=0, AC=0
ORI,<байт>	2	7	Порозрядне 'АБО' над змістом накопичувача і байтом	Z,S,P, C=0, AC=0
CPI,<байт>	2	7	Порівняння байта зі змістом накопичувача	(Z,S,P, C,AC)[3]
RLC (RRC)	1	4	Циклічний зсув змісту накопичувача вліво (вправо)	C[4]
RAL (RAR)	1	4	Циклічний зсув змісту накопичувача вліво (вправо) через перенесення	C[4]
CMA	1	4	Порозрядне інвертування накопичувача	-
STC	1	4	Встановлення ознаки перенесення в одиницю	C=1
CMC	1	4	Інвертування ознаки перенесення C	$C=\overline{C}$

Продовження таблиці Б.1

1	2	3	4	5
4 Команди переходів				
PCHL	1	5	Занесення змісту регістрів **H**, **L** в лічильник команд (зміст **H** – в старший байт, **L** – в молодший)	-
JMP,<адреса>	3	10	Безумовний перехід по вказаній адресі	-
JC/(JNC), <адреса>	3	10	Перехід при наявності (відсутності) перенесення	-
JZ/(JNZ), <адреса>	3	10	Перехід при наявності (відсутності) нуля	-
JP/(JM), <адреса>	3	10	Перехід при плюсі (мінусі)	-
JPE/(JPO), <адреса>	3	10	Перехід при парності (непарності)	-
CALL,<адреса>	3	17	Виклик підпрограми	-
CC/(CNC), <адреса>	3	11 (17)	Виклик підпрограми при наявності (відсутності) перенесення	-
CZ/(CNZ), <адреса>	3	11 (17)	Виклик підпрограми при наявності (відсутності) нуля	-
CP/(CM), <адреса>	3	11 (17)	Виклик підпрограми при плюсі (мінусі)	-
CPE/(CPO), <адреса>	3	11 (17)	Виклик підпрограми при парності (непарності)	-
RET	1	10	Повернення з підпрограми	-
RC/(RNC)	1	5 (11)	Повернення при наявності (відсутності) перенесення	-
RZ/(RNZ)	1	5 (11)	Повернення при наявності (відсутності) нуля	-
RP/(RM)	1	5 (11)	Повернення при плюсі (мінусі)	-
RPE/(RPO)	1	5 (11)	Повернення при парності (непарності)	-
RST,<номер>	1	11	Повторне запущення з адреси 8 х номер (0,8,…,56)	-
5 Команди вводу-виводу та управління				
IN,<порт>	2	10	Ввод даних із вказаного порта до накопичувача	-
OUT,<порт>	2	10	Вивід даних із накопичувача до вказаного порта	-

Продовження таблиці Б.1

1	2	3	4	5
PUSH rp	1	11	Занесення змісту пари регістрів **rp** (**B-C,D-E,H-L, PSW**) до стека	-
POP rp	1	10	Видання даних зі стека в пару регістрів **rp** (**B-C,D-E, H-L,PSW**)	(Z,S,P,C AC)[6]
XTHL	1	18	Обмін даними між верхівкою стека та парою регістрів **H-L**	-
SPHL	1	5	Занесення в показчик стека змісту регістрів **H-L**	-
DI/EI	1	5	Заборонити/дозволити переривання	-
NOP	1	4	Порожня операція	-
HLT	1	7	Зупин	-

Примітки:

1 встановлюється при наявності займу до старшого розряду, у протилежному випадку скидається;

2 встановлюється при наявності займу зі старших чотирьох розрядів в молодші, у протилежному випадку скидається;

3

– **Z** встановлюється, якщо зміст регістра та байта даних дорівнює змісту накопичувача ;

– **S,C,** якщо зміст регістра або байта даних більше змісту накопичувача;

– **AC,** якщо зміст молодших чотирьох розрядів регістра й байта даних більше змісту молодших чотирьох розрядів накопичувача;

– **P,** якщо байт різниці між змістом накопичувала та змістом регістра або байта даних містить парне число одиниць;

4 стан ознаки дорівнює значенню висунутого з накопичувача двійкового розряду;

5 у знаменнику дробу вказано кількість тактів при виконанні умов, в чисельнику – при невиконанні;

6 за командою ***POP PSW*** ознаки установлюються відповідно до значення розрядів слова, яке занесено до стека, при інших значеннях rp ознаки не змінюються

ДОДАТОК В - СПОСОБИ ОРГАНІЗАЦІЇ ЧАСОВИХ ЗАТРИМОК

При організації часової затримки використовують дані табл.Б.1 про час виконання тієї чи іншої команди. В залежності від потрібної тривалості затримки доцільно розділити всі способи організації затримки на дві групи:

способи організації затримки малої тривалості;

способи організації затримки великої тривалості;

Для обох груп затримка кратна тривалості одного машинного такту $t_{зад}$=0,5 мкс.

Для організації затримок малої тривалості використовуються команди, які не змінюють інформацію та стан ЦПЕ. Рекомендується використовувати команди NOP; MOV A,A; ADI 0 і пару команд PUSH-POP, які складають у ланцюжок.

Для організації затримок великої тривалості використовують однокаскадний або багатокаскадний лічильник. Типова схема однокаскадного лічильника показана на рисунку В.1.

Час затримки в машинних тактах розраховується за формулою:

$$N_{зад}=T_{вст}+N_в(T_{рах}+T_{дод.\,в})+T_{дод.\,к} \qquad (В.1)$$

де $N_в$ - кількість циклів лічильника;

$T_{вст}$ - кількість тактів виконання команди встановлення початкового стану лічильника;

$T_{рах}$ - мінімальна затримка у циклі рахування;

$T_{дод.\,в}$ - додаткова затримка у циклі рахування;

$T_{дод.\,к}$ - додаткова затримка після закінчення циклу;

Вибір величин, що входять до цього виразу, здійснюється методом підбору за даними (кількість тактів) табл.Б.1.

Текст програми, що реалізує схему однокаскадного лічильника, наведено у таблиці В.1.

Таблиця В.1 - Текст програми однокаскадного лічильника

Адреса ОЗП	Код команди	Мітка	Мнемоніка	Кільк. тактів	Формат	Текст коментарю
04A0 04A1	06 C8H		**MVI B,N_B**	7	2	Встановлення пачатк. стану
04A2	00	**M1:**	**NOP**	4	1	Додаткова затримка у циклі
04A3	05		**DCR B**	5	1	Мінімальна затримка лічильника
04A4 04A5 04A6	C2 A2 04		**JNZ M1**	10	3	
04A7	00		**NOP**	4	1	Додаткова затримка після закінчення рахування
04A8	7F		**MOV A,A**	5	1	

При $N_в$=200 дана програма реалізує затримку в машинних тактах, яка дорівнює

$$N_{зад}=7+200*(15+4)+9=3816$$

Звідси час затримки:

$$T_{зад}=t_{зад}*N_{зад}=0,5*3816=1908мкс.$$

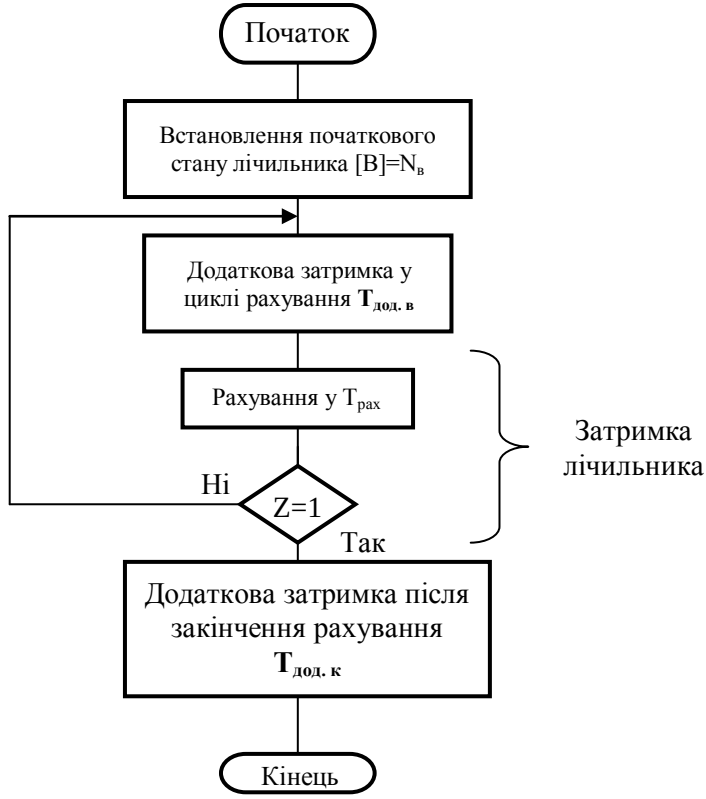


Рисунок В.1 - Типова схема однокаскадного лічильника.

Використовуючи багатокаскадні лічильники, можливо отримати практично будь-яку затримку.

ДОДАТОК Д - ПРИКЛАД РЕАЛІЗАЦІЇ ЛОГІЧНОЇ ФУНКЦІЇ

Нехай задана принципова схема логічного функціонального вузла (рисунок Д.1). Програмна реалізація даного пристрою складається з наступних етапів.

Д.1 Складання та перетворення логічної функції.

$$Y=\overline{X1*X2}+X3*X4=X1*X2*\overline{X3*X4}=X1*X2*(\overline{X3}+\overline{X4})=$$

$$=X1*X2*\overline{X3}+X1*X2*\overline{X4}=Y1+Y2$$

Д.2 Складання таблиці істинності отриманої функції.

Таблиця Д.1 - Таблиця істинності логічного функціонального вузла

X1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
X2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
X3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
X4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Y1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
Y2	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

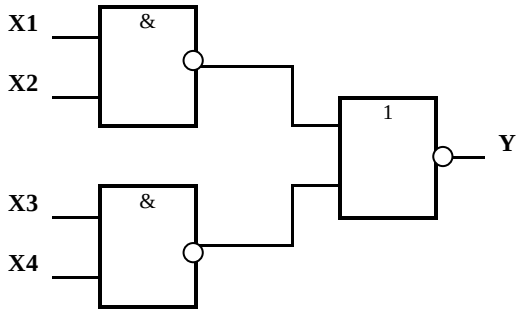


Рисунок Д.1 - Схема функціонального вузла

Д.3 Складання схеми алгоритму, що реалізує логічну функцію.

Схема повинна складатися з ланцюжка блоків, в яких відбувається обчислення значення відповідної частини функції з наступною перевіркою на рівність результату одиниці (рисунок Д.2).

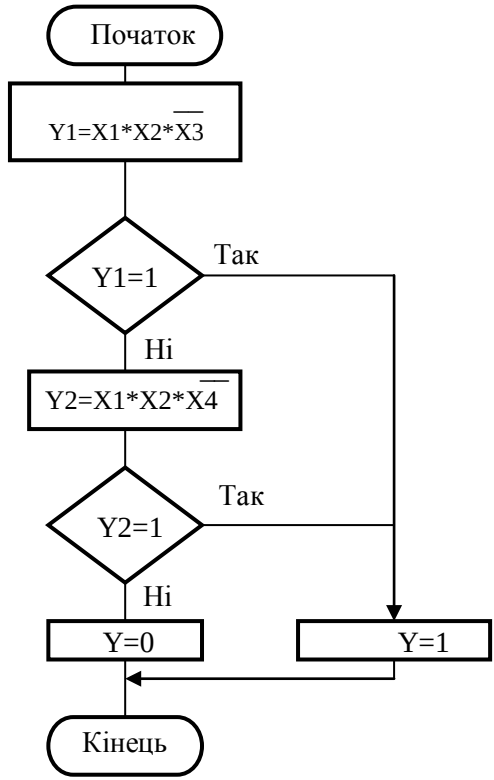


Рисунок Д.2 - Схема програми, що реалізує логічний пристрій

Д.4 Складання тексту програми.

Цей етап для кожного частини функції здійснюється у такій послідовності.

Д.4.1 Маскування розрядів, які не використовуються, шляхом логічного множення на “маску” – двійкове число, в якому на місці

розрядів,які не використовуються, стоять нулі, а на інших місцях – одиниці.

Таблиця Д.2 - Вихідна інформація

	X7	X6	X5	X4	X3	X2	X1	X0
Маска Y1	0	0	0	0	1	1	1	0
Маска Y2	0	0	0	1	0	1	1	0

Д.4.2 Ознакою умовного переходу до аналізу наступної частки обираємо прапорець Z, для цього замінимо перевірку Y1=1 і Y2=1 перевіркою Y1=0 і Y2=0.

Проведемо складання по модулю 2 “маскованої” вхідної інформації з константою, отриманою зі своєї “маски” шляхом заміни одиниць на нулі у розрядах, що містять $\overline{X_i}$.

Д.4.3 Здійснюється перехід по прапорцю Z.

Текст програми, що реалізує отриману логічну функцію, наведений у табл.Д.3.

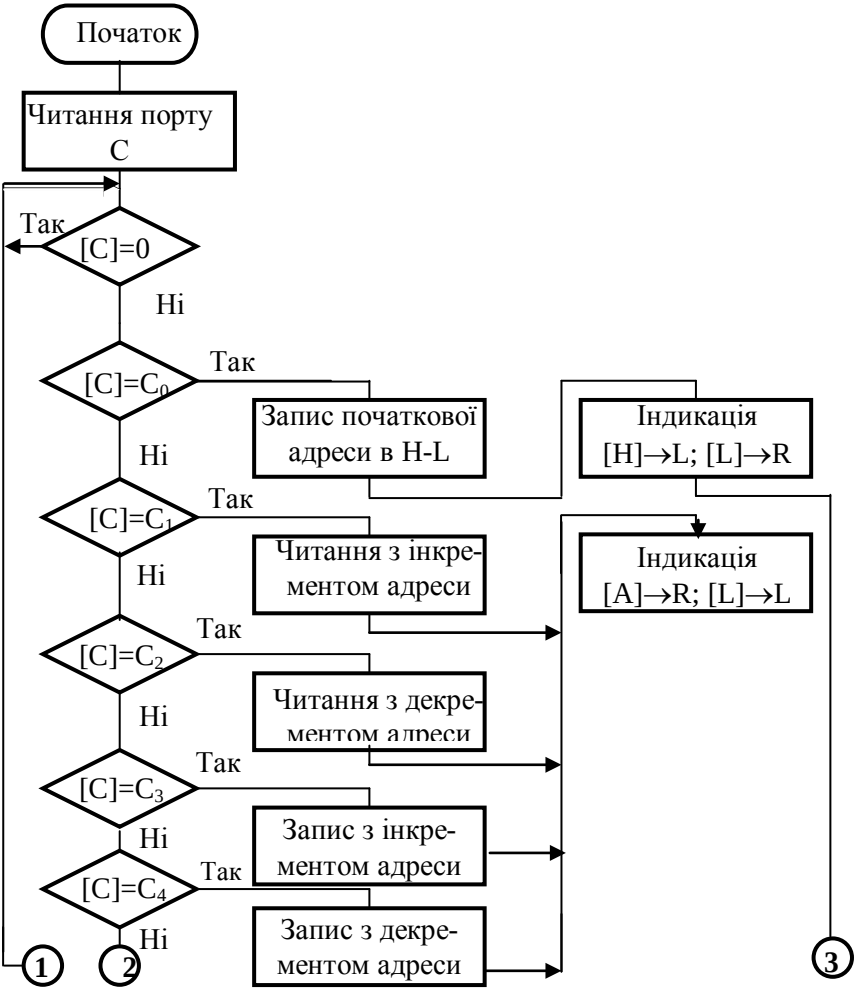
Таблиця Д.3- Текст програми

Мітка	Мнемоніка	Кільк. тактів	Формат	Текст коментарю
FUNC:	**IN PORT X**	10	2	Ввід {X} у A
	MOV C,A	5	1	Занесення X до C
	ANI MASK1	7	2	Обчислення Y1
	XRI CONST1	7	2	
	JZ M1	10	3	Перехід доY=1, тому що Y1=1
	MOV A,C	5	1	Відновлення X у A
	ANI MASK2	7	2	Обчислення Y2
	XRI CONST2	7	2	
	JZ M1	10	3	Перехід доY=1, тому що Y2=1
	XRA A	4	1	Формування Y=0
	JMP M2	10	3	
M1:	**MVI A,FF**	7	2	A=11111111
M2:	**OUT PORT Y**	10	2	Вивід Y
	RET	10	1	Повернення

ДОДАТОК Ж - ОПИС ПРОГРАМИ-МОНІТОР

Програма-монітор призначена для керування мікроконтролером (мікроЕОМ). Після налагодження програми вона записується до ПЗП та викликається при включені мікроЕОМ.

Принцип роботи програми базується на тестуванні визначених розрядів (або їхніх комбінацій) керуючого порту **С** та виклику відповідних процедур (рисунок Ж.1).



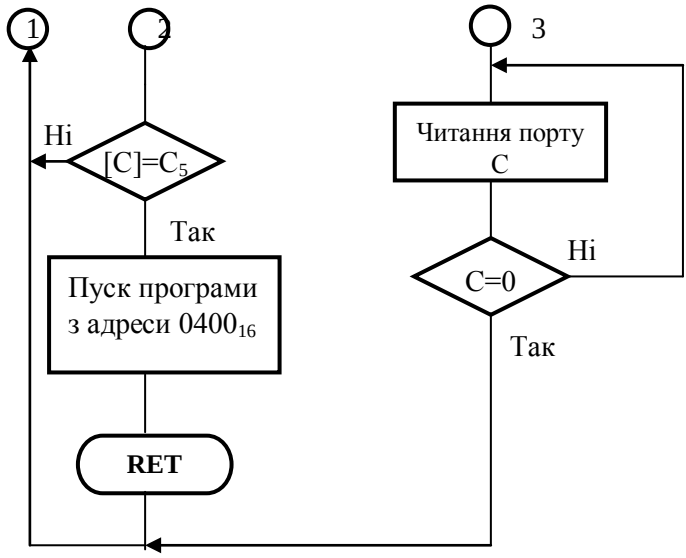


Рисунок Ж.1 - Схема алгоритму програми-монітор

ДОДАТОК К ВАРІАНТИ ЗАВДАНЬ ДЛЯ САМОСТІЙНОЇ РОБОТИ

На мові Асемблера мікропроцесора I8080 скласти програми, які реалізують наступні завдання :

1. У масиві чисел, розташованому з адреси 0150 Н до 0200 Н, встановити в "1" старші і в" 0" молодші розряди чисел.

2. Підрахувати, скільки разів у масиві чисел з адресами 0200-02FF зустрічаються числа, менші, ніж перший елемент масиву.

3. Підрахувати, скільки разів у масиві чисел з адресами 0500-05FF зустрічаються числа, більші, ніж перший елемент масиву.

4. У масиві чисел, які зберігаються в ОЗП у чарунках з адресами з 0500 до 05DE, замінити від'ємні числа нулем.

5. Підрахувати число додатних елементів у масиві з адресами 0600-06AE.

6. Здійснити обмін інформації між масивами, розташованими з адрес 0400 і 0500, які вміщають по 21 елементу.

7. Програма формування тимчасової затримки тривалістю 0,3 с (f = 1,2 МГц).

8. Зсунути вміст чарунки пам'яті з адресою 040F на число бітів, яке визначається вмістом регістру **D.**

9. Скласти два числа, які знаходяться у чарунках з адресами 0100Н та 0200Н, і вивести молодший напівбайт у порт 01, а старший напівбайт - порт 02.

10. Формування додатного коду числа, прийнятого з порту № 256.

11. Масив чисел у чарунках ОЗП з адресами з 0500 до 05CD переписати у ті ж самі чарунки у зворотному порядку.

12. З масиву чисел із адресами з 0400 до 04CD видавати вміст чарунок із непарними адресами у вихідний порт, помножуючи їх на 2.

13. Переписати 10 послідових байтів із одної області пам'яті (починаючи з адреси 2200 Н) у другу (з адреси 2300 Н).

14. Програма пересилки у старший розряд вихідного порту логічної одиниці з частотою 100 Гц (f = 2 МГц).

15. Масив чисел розмірністю 120, розташований, починаючи з адреси 0400, вивести на вихідні порти 01 (молодший напівбайт) і 02 (старший напівбайт).

16. Замінити у масиві з адресами 0500Н-05ССН числа, більші 9,

числом 10.

17. У регістрі ознак установити в одиницю ознаки **S** і **P.**

18. Масив чисел з адресами з 0500H і 05ECH розсортувати, переписав від'ємні, починаючи з адреси 0600H, а решту - з адреси 0700H.

19. В масиві чисел, які зберігаються в ОЗП, в чарунках з адресами з 0500H до 05BEH, знайти число 25D і вказати його адресу на вихідних портах 01 і 02. Число в масиві є і воно єдине.

20. В масиві чисел із адресами з 0500H до 05FFH числа, які зберігаються у чарунках із парними адресами, замінити на 0.

21. В масиві чисел з адреси 0400H до 04FFH парні числа поділити на 2, непарні - помножити на 4.