

Міністерство освіти та науки України
Запорізький національний технічний університет

"ЕОМ ТА МІКРОПРОЦЕСОРНІ СИСТЕМИ"

**методичні вказівки
до практичних занять**

для студентів напрямку підготовки
6.050103 – Програмна інженерія
спеціальності 7.080403 "Програмне забезпечення
автоматизованих систем"
усіх форм навчання

2008

Методичні вказівки до практичних занять з курсу "ЕОМ та мікропроцесорні системи" для студентів напрямку підготовки 6.050103 "Програмна інженерія" спеціальності 7.080403 "Програмне забезпечення автоматизованих систем" усіх форм навчання/ Укладачі: С.К. Корнієнко, Л.П. Скачко. - Запоріжжя: ЗНТУ, 2008. - 26 с.

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

Рецензент: В.Ф.Онищенко, доцент, к. ф.-м.н.

Відповідальний
за випуск: А.В.Притула, доцент, к.т.н.

Затверджено
на засіданні кафедри
"Програмних засобів"

Протокол № 1
від 28.08.08 р.

Методичні вказівки містять основні відомості з архітектури 8-розрядного мікропроцесора I8080 і практичні рекомендації по його програмуванню. Призначені для використання на практичних і лабораторних заняттях по дисципліні "ЕОМ та мікропроцесорні системи", а також у процесі самостійної роботи студентів.

ЗМІСТ

1 Архітектура типової системи на базі i8080.....	4
1.1 Типи і формати даних.....	4
1.2 Адресний простір.....	6
1.3 Формат команд.....	8
1.4 Методи адресації.....	8
1.4.1 Неявна адресація.....	9
1.4.2 Пряма адресація.....	9
1.4.3 Безпосереднє адресація	9
1.4.4 Опосередкована адресація.....	10
2 Програмування мікропроцесорних систем	11
2.1 Елементарні засоби програмування.....	11
2.2 Організація циклів	15
2.3 Програмна обробка масивів.....	15
2.4 Робота з підпрограмами.....	20
2.5 Організація переривань.....	22
Література.....	24
Додаток А Варіанти завдань для самостійної роботи.....	25

1 АРХІТЕКТУРА ТИПОВОЇ СИСТЕМИ НА БАЗІ І8080

Поняття архітектури МП відображає його структуру, засоби звертання до всіх доступних для споживача елементів структури, а також систему команд, режими адресації, формати і довжину даних.

1.1 Типи і формати даних

Типи і формати даних, які обробляються мікропроцесором (МП), показані на рисунку 1.1.

байт без знаку	7 0 □ □ □ □ □ □ □ □	0...+255
байт зі знаком	7 0 □ □ □ □ □ □ □ □	-128...+127
логічний байт	7 0 □ □ □ □ □ □ □ □	Вісім логічних значень
упаковане двійково-десятькове	7 0 □ □ □ □ □ □ □ □	0...+99
двобайтове слово	15 0 □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □	0...+65535

Рисунок 1.1 - Типи та формати даних

Над двійковими однобайтовими словами виконуються операції складання, віднімання, прирощення або зменшення на одиницю; над двійково-десятьковими – 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 і 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 в двійково-десятковий. Логічні операції виконуються за правилами двійкової логіки.

За результатами виконання 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 їх зберігання використовується спеціальний регістр **F**, який зветься *регістром ознак або прапорців* (рис. 1.2).

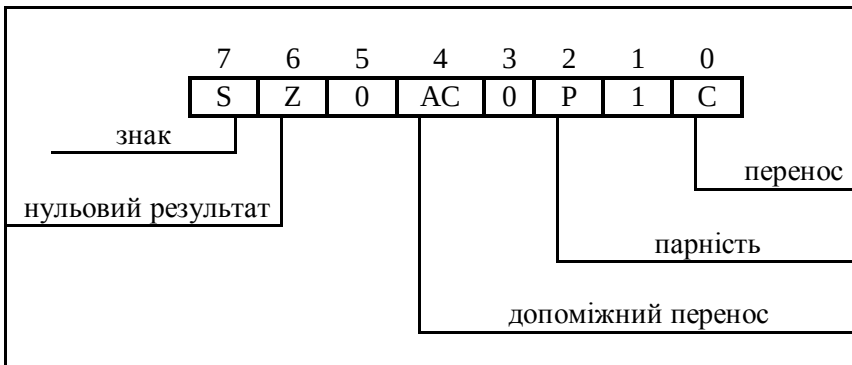


Рисунок 1.2 - Формат регістру ознак F

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.

1.2 Адресний простір

Адресний простір МП складають простір запам'ятовуючих пристроїв (ЗП), зовнішніх пристроїв (ЗвП) і надоперативного запам'ятовуючого пристрою (НОЗП). Простір ЗП логічно і фізично організовано у вигляді послідовності байтів об'ємом 64 Кбайт (рисунок 1.3), які адресуються 16-розрядними словами.

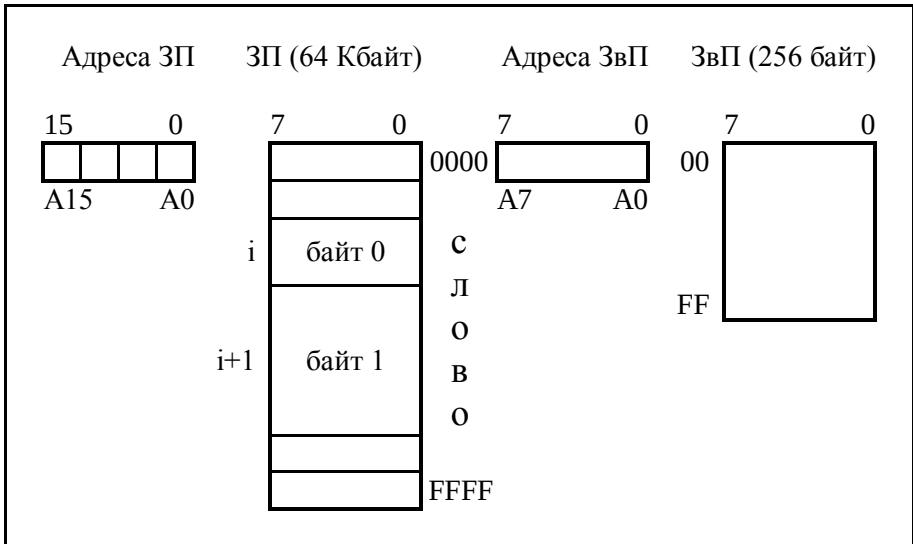


Рисунок 1.3 - Склад та організація адресного простору ЗП та ЗвП

Двохбайтові слова розташовуються у порядку зростання адрес пам'яті та адресуються за молодшим байтом. В адресному просторі розташовуються **ОЗП** і **ПЗП**, де зберігаються програми та дані.

В будь-якому місці ОЗП можна організувати стек з послідовним доступом за принципом **LIFO (Last In - First Out)** об'ємом (теоретично) до 64 Кбай. Стек адресується 16-розрядними двійковими словами, які вказують поточну **вершину стека** – адресу старшого байта останнього слова, завантаженого до стека.

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

Внутрішній над оперативний запам'ятовуючий пристрій (**НОЗ**) I8080 (рис. 1.4) 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 16-розрядні регістрові пари. У першому випадку регістри використовуються для зберігання байтових даних, у другому – для зберігання адрес або двобайтних слів даних.

Слово стану програми **PSW (Program Status Word)** використовується для зберігання умісту акумулятора (старший байт) та регістра ознак (молодший байт).

16-розрядний вказівник стеку **SP (Stack Pointer)** призначений для зберігання адреси вершини стеку.

16-розрядний програмний лічильник **PC (Program Counter)** використовується для формування адреси наступної команд (природна адресація). Примусова адресація команд здійснюється шляхом використання команд безумовного та умовних переходів або команд безумовного та умовних викликів підпрограм.

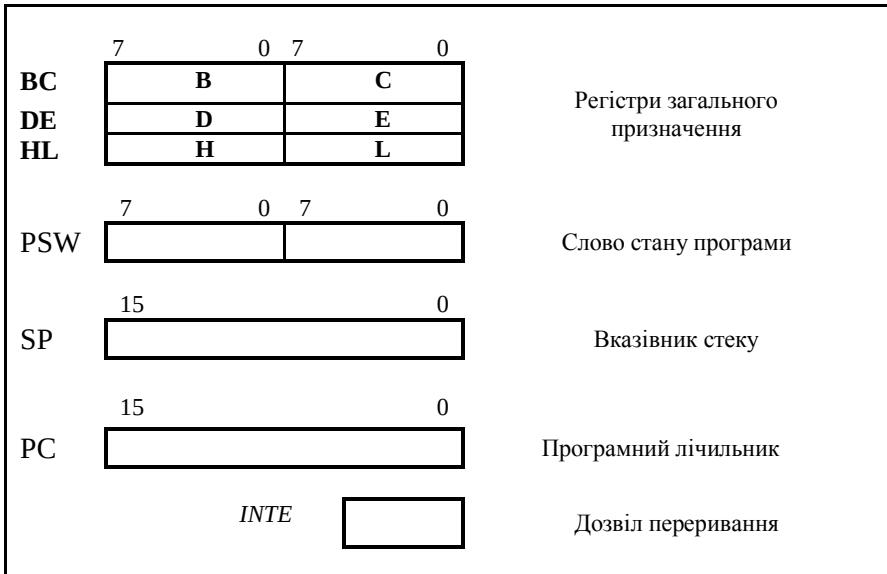


Рисунок 1.4 - Організація НОЗП

1.3 Формат команд

Мікропроцесор I8080 – одноадресний. Команди подаються одним, двома або трьома байтами (рис. 1.5). Однобайтовий формат використовується для кодування команд звертання до регістрів НОЗП, непрямого адресування ЗУ, при роботі зі стеком. Двох- та трьохбайтові формати використовуються для команд з безпосередньою та прямою адресацією.

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

Багатобайтові команди зберігаються у сусідніх чарунках пам'яті та адресуються за першим байтом, причому двобайтові слова даних та адрес розташовуються у порядку зростання.

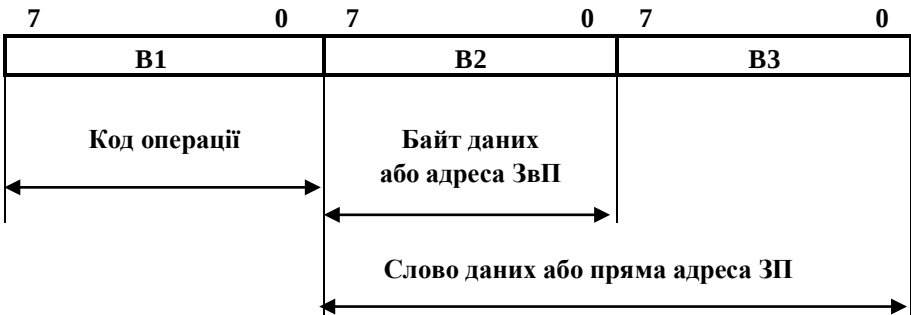


Рисунок 1.5 - Формати команд МП I8080

1.4 Методи адресації

В МП I8080 використовуються наступні методи адресації:

- неявна;
- пряма;
- безпосередня;
- опосередкована.

1.4.1 Неявна адресація

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

Неявна адресація використовується для :

- звертання до акумулятору. Наприклад, команда **CMA** (інвертувати зміст акумулятору);
- звертання до бітів регістру ознак, наприклад, **STC** (встановити біт переносу);
- для адресації триггеру переривань (команди заборони та дозволяння переривань).

1.4.2 Пряма адресація

При цьому методі адресації адреса елементу МПС, який використовується під час виконання операції, задається у команді. Пряма адресація 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 1000H вмістом акумулятора);

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

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).

1.4.3 Безпосереднє адресація

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, наприклад, **MVI B,05H, LXI H,0016H**.

1.4.4 Опосередкована адресація

При цьому методі адресації виконавча адреса попередньо формується в одній з регістрових пар **B, D, H**. Наприклад, **STAX D** (записати вміст акумулятора до чарунки пам'яті, з адресою, розміщеною в регістровій парі **D**).

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

Система команд МП І8080 приведена у додатку А.

2 ПРОГРАМУВАННЯ МІКРОПРОЦЕСОРНИХ СИСТЕМ

2.1 Елементарні засоби 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. Встановити прапорці регістру ознаки згідно із значенням числа, яке зберігається у чарунці **0456H**:

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

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

```
CMA  
INR A
```

Приклад 5. Перевести число 13 у двійково-десятичну форму

```
MVI A,13  
DAA
```

Приклад 6. Обчислити арифметичний вираз $(-17+47)$ за допомогою додаткового коду.

```
MVI A,10001B  
CMA  
INR A  
ADI 101111B
```

Приклад 7. 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 8. Обнулити 0-й та 5-й розряди вмісту регістру **B** :

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

Приклад 9. Встановити в одиницю 2-й, 3-й та 7-й розряду вмісту регістру **D**:

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

Приклад 10. Перевірити стан нульового біту регістру **C**:

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

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

```
б) MOV A,C
RAR
```

В результаті нульовий розряд буде циклічно зсунутий праворуч до триггеру **C** регістра ознак. Після цього його можна тестувати за допомогою команд умовних переходів **JC** або **JNC**.

Приклад 11. Змінити вміст чарунки пам'яті з адресою **0400H** у розряді **D5**:

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

Приклад 12. 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 13. Відняти вміст чарунки пам'ять з адресою **AF9CH** із вмісту чарунки пам'яті з адресою **B9FH** та здійснити перехід до чарунки пам'яті з адресою **ADR**, якщо результат виявиться від'ємним:

LXI H, AF9CH
LXI D, B9FH
MOV B,M
LDAX D
SUB B
JM ADR

Приклад 14. Прийняти інформацію з порту з адресою **PORT1** і записати її в порт з адресою **PORT2**:

IN PORT1
OUT PORT2

Приклад 15. 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**
CZ A
CALL B

Приклад 16. З'ясувати, чи є число, отримане з порту 00h, парним. Якщо так – вивести на порт 00h одиницю, інакше – нуль.

```

IN 00H
RAR
JNC L1
MVI A,0
OUT 0
HLT

L1:
MVI A,1
OUT 0
HLT

```

Приклад 17. Порівняти число, отримане з порту 0 із значенням регістру D.

1. Якщо значення однакові — вивести на порт 1 число 1.
2. Якщо не однакові — вивести на порт 0 їх різницю.
3. Якщо отримано від'ємне число — вивести його модуль на порт 0.

```

IN 0
SUB D
JZ L2
JP L1
CMA
INR A
OUT 0
JMP L3

L1: OUT 0
JMP L3

L2: MVI A,1
OUT 1

L3: продовження програми

```

2.2 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 із 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** неможна одразу використати команду тестування регістру ознак, оскільки команда **DCX rp** не модифікує прапори.*

2.3 Програмна обробка масивів

Приклад 1. Маємо масив чисел з адресою 04FFH. Потрібно реалізувати 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; завдання початкової адреси масиву
MVI C,19H  ; ініціалізація циклу
MVI B,0H   ; ініціалізація лічильника парних
               елементів
LAB: MOV A,M       ; вибірка числа із масива
ANI 01H         ; виділення 0-го біту
JNZ L1          ; перехід на вибір наступного числа
INR B           ; збільшення лічильника парних
               елементів
L1: INX H        ; формування адреси наступного
               елемента масиву
DCR C           ; перевірка закінчення циклу
JNZ LAB
HLT

```

Приклад 3. Підрахувати кількість чисел у масиві (адрес першого елемента FFh, кількість елементів – 100), більших за отримане з порту 00h, та вивести результат на порт 00h:

```

IN 0
MVI B,0
LXI H,00FFH
MVI C,100
LABEL1:
  CMP M
  JNC LABEL2
  INR B
LABEL2:
  INX H
  DCR C
  JNZ LABEL1
  MOV A,B
  OUT 0
  HLT

```

Приклад 4. У масиві всі від'ємні елементи замінити їх модулями. Кількість елементів в масиві – 10, адреса першого елемента A100h

```

MVI B,10
LXI H,A100H
ITERATION:
MOV A,M
ANA A
JP SKIP
CMA
INR A
MOV M,A
SKIP:
INX H
DCR B
JNZ ITERATION
MOV A,B
OUT 00H
HLT

```

Приклад 5. Масив складається з 50 елементів. Підрахувати елементи масиву, що дорівнюють 37.

```

MVI C,50
MVI A,37
MVI B,0
LXI H,1000H
ITERATION:
CMP M
JNZ SKIP
INR B
SKIP: INX H
DCR C
JNZ ITERATION
MOV A,B
OUT 0
HLT

```

Приклад 6. Масив складається з 50 елементів, перший з яких має адресу 32h. Скопіювати усі його додатні елементи в масив, що починається з адреси FAh. Кількість додатних елементів вивести на порт 0.

```

MVI B,50
LXI H,0032H
LXI D,00FAH
MVI C,0

S:
MOV A,M
ANA A
JM GH
INR C
STAX D
INX D

GH:
INX H
DCR B
JNZ S
MOV A,C
OUT 0
HLT

```

Приклад 7. Масив складається з 100 чисел. Перше знаходиться у чарунці пам'яті з адресою 100h. Скопіювати числа в масив, що починається з чарунки 200h, помножуючи непарні на 2

```

MVI B,64H
LXI H,100H
LXI D,200H

S:
MOV A,M
ANA A
RAR
JC G; перевірка парності
RAL; повернення попереднього значення числа
RAL; множення на 2

```

```

      STAX D
      INX D
G:    INX H
      DCR B
      JNZ S
      HLT

```

Приклад 8. Масив складається з 70 елементів. Адреса першого елемента: 64h. Скласти молодші 4 розряди усіх додатних елементів, та отриману суму вивести на порт 0.

```

      MVI B,70
      LXI H,0064H
      MVI D,00H
START: MOV A,M
      ANA A
      JM NEXT
      ANI 00001111B
      ADD D
      MOV D,A
NEXT:  INX H
      DCR B
      JNZ START
      MOV A,D
      OUT 0
      HLT

```

Приклад 9. В масиві замість його елементів записати значення їх молодших полубайтів. Значення старших полу байтів записати в інший масив. Перший масив починається з адреси 100h, другий масив – з адреси 200h, кількість елементів – 50.

```

      LXI H,100H
      LXI D,200H

```

```

    MVI B,50
LABEL1:
    MOV A,M
    MOV C,A
    ANI 11110000B
    RRC
    RRC
    RRC
    RRC
    STAX D
    MOV A,C
    ANI 00001111B
    MOV M,A
    INX H
    INX D
    DCR B
    JNZ LABEL1
    HLT

```

2.4 Робота з підпрограмами

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

Таким чином, вміст регістрів, виконаних підпрограмою, необхідно тимчасово запам'ятати, 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

С, вона повинна мати такий вигляд :

```

SUBR:
PUSH PSW           ; Запам'ятовування вмісту регістрів до
стеку
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

```

2.5 Організація переривань

Обслуговування вводу/виводу по перериванням є альтернативною програмно-керованому 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 їх обслуговування зветься системою переривань.

Процес обробки запиту на переривання значно подібний процесам виклику і повернення із 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).

Звичайно 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 2.1 - Розподіл адрес входів до підпрограм обслуговування переривань

Номер області ЗУ	Початкова адреса	Номер області ОЗУ	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

Література

1. Гилмор Ч. Введение в микропроцессорную технику. - М.: Мир, 1989, 365 с..
2. Конспект лекцій з дисципліни “ЕОМ та мікропроцесорні системи” для студентів професійного спрямування 6.0804 “Комп’ютерні науки” спеціальності 8.080403 “Програмне забезпечення автоматизованих систем” усіх форм навчання / Укладач С.К.Корнієнко. - Запоріжжя: ЗДТУ, 2001. - 66 с
3. Лихциндер В.Я., Кузнецов В.Н. Микропроцессоры и вычислительные устройства в радиотехнике. - К.: Вища школа, 1988, 254 с.
4. Майоров В.Г., Гаврилов А.И. Практический курс программирования микропроцессорных систем. - М.: Машиностроение, 1989, 383 с.
5. Проектирование микропроцессорной электронно-вычислительной аппаратуры: Справочник / В.Г. Артюхов, А.А. Будняк, В.Ю. Лапий и др. - К.: Техника, 1988, 379 с.
6. Самофалов К.Г., Викторов О.В. Микропроцессоры. К.: Техника, 1989, 334 с.
7. Справочник по микропроцессорным устройствам / А.А. Молчанов, В.И. Корнейчук, В.П. Тарасенко. - К.:Техника, 1987, 389с.
8. Токхайм Р. Микропроцессоры: Курс и упражнения. - М.: Энергоатомиздат, 1988, 302 с.



Додаток А

Варіанти завдань для самостійної роботи

На мові Асемблеру мікропроцесору K580 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 в масиві з адресами 0600-06AE.
6. Здійснити обмін інформації між масивами, розташованими з адрес 0400 і 0500, які вміщують 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 в зворотному порядку.
12. З масиву чисел з адресами з 0400 до 04CD видавати вміст чарунок з 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 реєстрі ознак встановити 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 05BEN, знайти число 25D і вказати його адресу на вихідних портах 01 і 02. Число в масиві є і воно єдине.

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

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