Полтавський національний технічний університет імені Юрія Кондратюка

Л е к ц і я № 11
з навчальної дисципліни “Архітектура комп'ютерів”

Модуль 3. Призначення, класифікація та характеристики процесорів. Архітектура і система команд МП x86

Тема лекції:
Організація циклів та оброблення масивів МП I80X86

План лекції

1. *Організація циклів за допомогою команд переходів.*
2. *Команди організації циклу з лічильником CX.*
3. *Організація вкладених циклів.*
4. *Оброблення масивів МП I80X86*

Факультет інформаційних та телекомунікаційних технологій і систем

Кафедра комп'ютерної інженерії
К.т.н., доцент Тиртишніков О.І.

1. Організація циклів за допомогою команд переходів

1.1. Схема реалізації циклів з передумовою

- оператор мови ***C:*** **while (умова продовження) {оператори};**
- оператор мови ***Pascal:*** **while умова продовження do begin оператори end.**

```
RPT:    (Команди перевірки умови продовження)
        Jcc AWAY; Умова продовження не виконана
        Оператори
        Jmp RPT
AWAY:   nop
```

1.2. Схема реалізації циклів з післяумовою

- оператор мови ***C:*** **do оператори while (умова продовження);**
- оператор мови ***Pascal:*** **repeat оператори until умова продовження.**

```
RPT:    Оператори
        (Команди перевірки умови продовження)
        Jcc RPT; Умова продовження виконана
```

1. Організація циклів за допомогою команд переходів

Приклад 1 (цикл з післяумовою): необхідно вивести (записати у стек) непарні числа, менше за 10

Програма мовою Pascal

```
program pr10;
  var i: integer;
   begin
    i:=1;
    repeat
      writeln(i);
      i:=i+2;
    until i>10;
   end.
```

Алгоритм

Рис. 1

Програма мовою Assembler

```
#MAKE_EXE#
      .stack 256
      .data
.code
      mov bx,1
Cycl: push bx
      add bx,2
      cmp bx,0Ah
      jb Cycl ; або jnae
      HLT
```

1. Організація циклів за допомогою команд переходів

Приклад 2 (цикл з передумовою): необхідно визначити мінімальну кількість членів ряду X^2+1, сума яких буде більше за 100, значення суми та останнього члена отриманої послідовності.

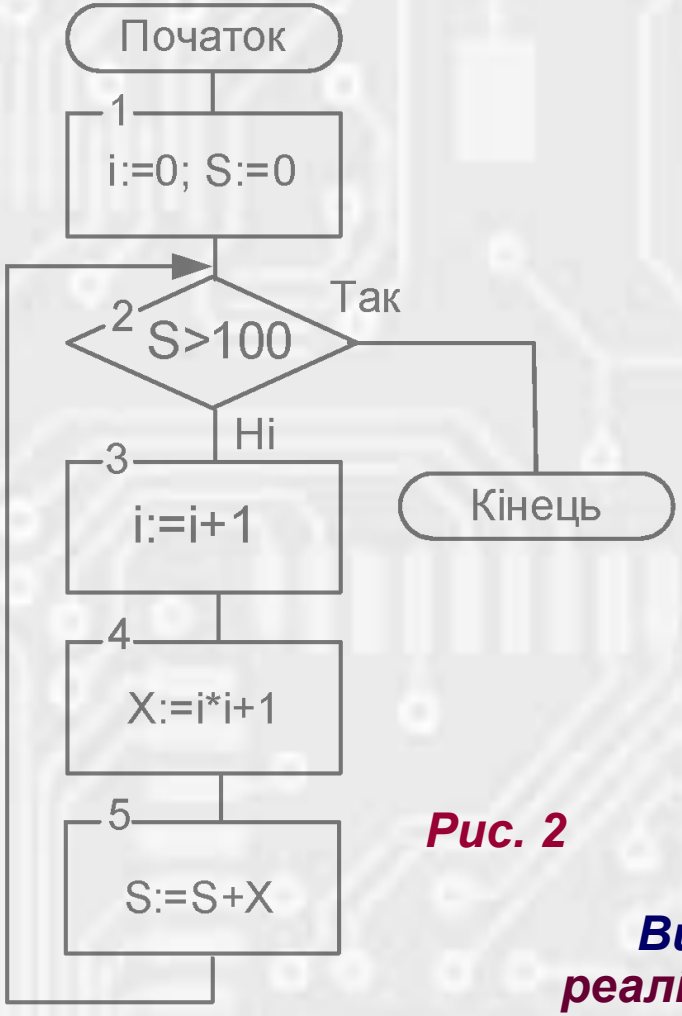


Рис. 2

```
#MAKE_EXE#
      .stack 256
      .data          .code
     mov bx,0
     mov cl,0
Cycl: cmp bx,100
     ja away
     inc cl; кількість членів
     mov al,cl
     mul al
     add ax,1; останній член
     add bx,ax; сума
     jmp Cycl
away: hlt
```

Висновок: цикл з передумовою складніше реалізується (необхідно дві команди переходів)

1. Організація циклів за допомогою команд переходів

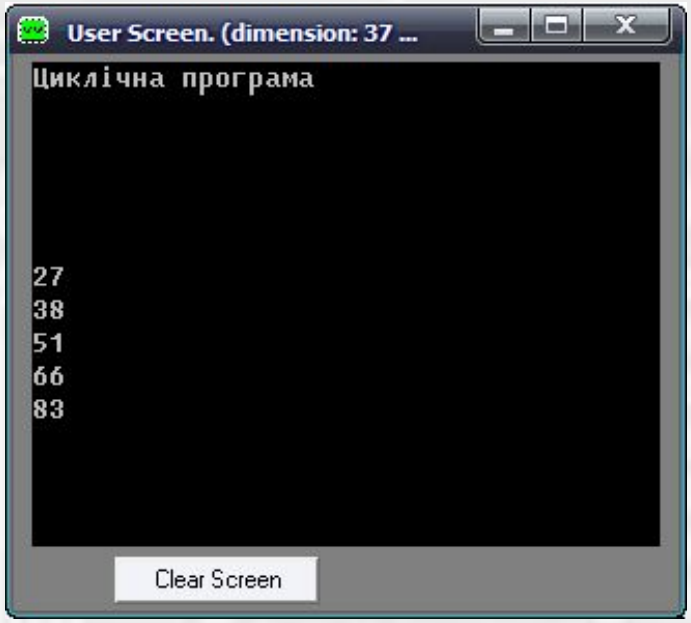
1.3. Організація циклу з лічильником

Приклад 3: обчислити значення функції $Y=X^2-2X+3$ при всіх цілих X з діапазону [6 - 10].

GOTOXY col, row – макрокоманда бібліотеки загальних функцій, має 2 параметри (стовпчик та рядок), встановлює курсор у вказану позицію.

```
#make_COM#
include 'emu8086.inc'
ORG 100h
PRINT 'Циклічна програма'
      MOV BL,6h
      MOV CX,0005
M1:   MOV AL,BL
      MUL AL
      MOV DX,AX
      MOV AX,0000
      MOV AL,02
      MUL BL
      SUB DX,AX
      ADD DX,03
      MOV AX,DX
GOTOXY 0,BL
CALL print_num
      INC BL
      DEC CX
      JCXZ ENDC
      JMP M1
ENDC: HLT
DEFINE_PRINT_NUM
DEFINE_PRINT_NUM_UNS
END
```

Команди організації циклу (варіант з післяумовою)



2. Команди організації циклу з лічильником CX

2.1. Команда організації циклу з лічильником у регістрі CX loop

Регістр ***ecx/cx*** виконує роль лічильника у командах управління циклами і при роботі з ланцюжками символів.

Синтаксис команди: loop мітка (адреса переходу). Команда призначена для організації циклу з лічильником у регістрі CX (у вигляді циклу з післяумовою). Кількість повторень циклу задається значенням у регістрі ***ecx/cx*** перед входом у послідовність команд, що складають тіло циклу.

Алгоритм виконання:

- декремент (зменшення на одиницю) вмісту регістра ***ecx/cx***;
- аналіз вмісту регістра ***ecx/cx:***
- якщо ***ecx/cx*** = *0*, передати управління керування наступній після ***loop*** команді;
- якщо ***ecx/cx*** > *0*, передати управління команді, мітка якої зазначена в якості операнда команди ***loop.***

Фактично команда ***loop мітка*** еквівалентна послідовності команд:

DEC CX

JNZ мітка

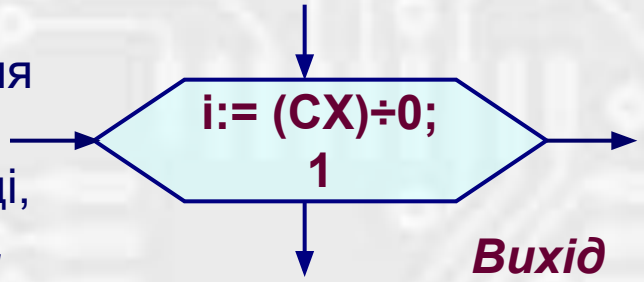


Рис. 3

2. Команди організації циклу з лічильником CX

2.2. Команди організації циклу з лічильником у регістрі CX з можливістю дострокового виходу з циклу

Команда ***LOOPE/LOOPZ*** вводить додаткову умову повторення цикла: ***ZF=1***. Це дозволяє завершувати цикл як по виконанні визначеної кількості повторень, так і при отриманні ненульового результату (або нерівності операндів, що порівнюються.)

Команда ***LOOPNE/LOOPNZ*** вводить додаткову умову повторення цикла: ***ZF=0.*** Це дозволяє завершувати цикл як по виконанні визначеної кількості повторень, так і при отриманні нульового результату (або рівності операндів, що порівнюються.)

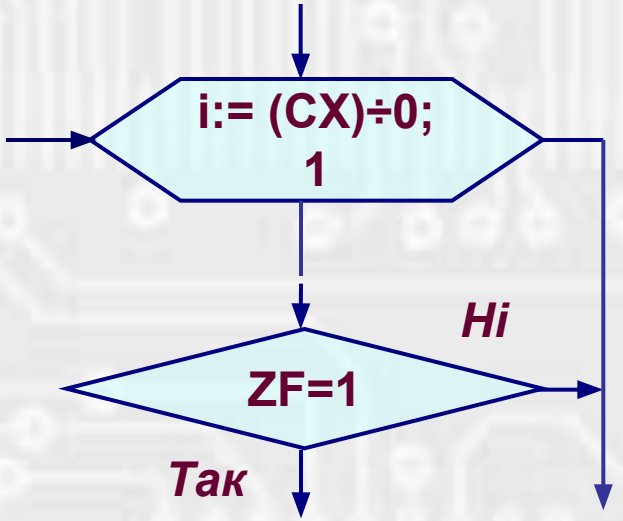


Рис. 4

Команди ***LOOP, LOOPZ, LOOPNZ*** реалізовують цикли мов високого рівня типу for (з лічильником або з відомою кількістю повторень).

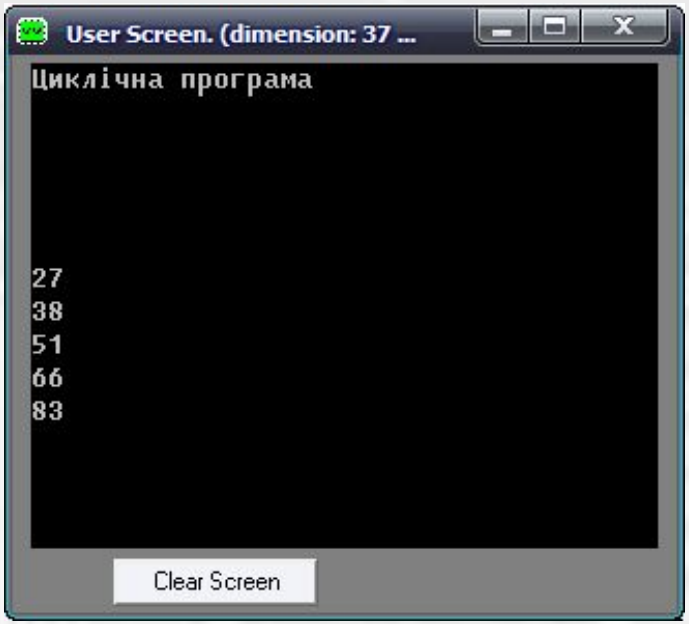
2. Команди організації циклу з лічильником CX

2.3. Приклад циклічної програми

Приклад 4: Обчислити значення функції $Y=X^2-2X+3$ при всіх цілих X з діапазону [6 - 10].

```
#make_COM#
include 'emu8086.inc'
ORG 100h
PRINT ‘Циклічна програма‘
      MOV BL,6h
      MOV CX,0005
M1:   MOV AL,BL
      MUL AL
      MOV DX,AX
      MOV AX,0000
      MOV AL,02
      MUL BL
      SUB DX,AX
      ADD DX,03
      MOV AX,DX
GOTOXY 0,BL
CALL print_num
      INC BL
      LOOP M1
      HLT
DEFINE_PRINT_NUM
DEFINE_PRINT_NUM_UNS
END
```

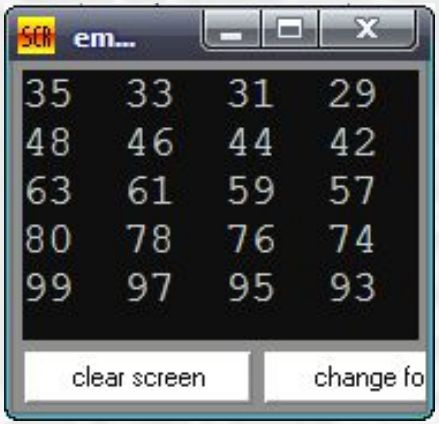
Команда організації циклу



3. Організація вкладених циклів

- При організації вкладених циклів будь-яких типів необхідно враховувати, що вони не мають перетинатися, тобто внутрішній цикл необхідно завершувати раніше зовнішнього.
- При організації вкладених циклів з відомою кількістю повторень виникає проблема, яка обумовлена наявністю тільки одного лічильника повторень ***CX***. Тому необхідно або організовувати один з лічильників на основі іншого регістра (змінної), або при входженні у внутрішній цикл зберігати вміст лічильника та потім відновлювати його при виході з внутрішнього циклу.

Приклад 5: Обчислити та вивести на екран у вигляді матриці чисел значення функції $Z=X^2-2Y+3$ при всіх цілих X з діапазону [6 - 10] та Y з діапазону [2 - 5]



Реалізуємо лічильник внутрішнього циклу в змінній У, зовнішнього – в CX.

3. Організація вкладених циклів

```
#make_COM#
include 'emu8086.inc'
ORG 100h
.data
x1 db 6;
y1 db 2
y db 4
i db 0
j db 0
.stack 256
.code
…
HLT
DEFINE_PRINT_NUM
DEFINE_PRINT_NUM_UNS
END
```

```
      MOV CX,5;   Кількість повторень по x
C1: XOR AX,AX; Початок ЗЦ
      MOV AL,x1;  Обчислення x^2
      MUL AL
      MOV DX,AX
C2: XOR AX,AX; Початок ВЦ
      MOV AL,y1;  Обчислення 2y
      SAL AL,1
      MOV BX,DX
      SUB BX,AX;  Обчислення значення виразу
      ADD BX,3
      MOV AX,BX
GOTOXY i,j;  Позиціювання курсору
CALL print_num;  Друк числа
      ADD i,4;  Збільшення позиції стовпця
      INC y1;   Збільшення значення y
      DEC y;    Зменшення лічильника ВЦ
      CMP y,0;Перевірка умови завершення ВЦ
      JNLE C2;      Завершення ВЦ
      MOV i,0; Повернення на нульовий стовпчик
      MOV y,4;      Відновлення лічильника ВЦ
      MOV y1,2;     Відновлення початкового значення y
      INC x1;   Збільшення значення x
      INC j;    Збільшення номера рядка
      LOOP C1;      Завершення ЗЦ
```

x1, y1 – початкові значення x та y відповідно; y – кількість повторень по y; i та j – координати стовпця та рядка для виведення на екран

4. Оброблення масивів МП I80X86

- ***Масив – сукупність елементів одного типу. Ця сукупність має одне ім'я, доступ до її окремих елементів здійснюється за допомогою одного або декількох індексів.*** Оброблення масивів організується за допомогою циклів, оброблення двомірних масивів – за допомогою вкладених циклів.

- ***Масив у асемблері – декілька елементів одного типу (однакової довжини), розташованих у пам'яті послідовно.*** Кількість елементів у масиві програмою не контролюється, ім'я визначається тільки для першого елементу (символічне ім'я адреси першого байта першого елемента масиву).

- Іноді масиви байтів називають рядками. В асемблері присутня група спеціалізованих команд для оброблення рядків (***MOVS, CMPS, STOS,LODS***).

- Індексація елементів масива організовується за допомогою індексних регістрів ***SI, DI***.

- ***Для подвійного індексування елементів двомірного масиву можна застосовувати пари регістрів (BX,SI), (BX,DI), (BP,SI), (BP,DI), але не можна (SI,DI) і (BX,BP)!***

4. Оброблення масивів МП I80X86

Приклад 6 (оброблення одномірного масиву): Маємо два масиви X та Y. Необхідно отримати масив Z, елементи якого є сумами відповідних елементів масивів X, Y, а також скалярний добуток X та Y.

```
#MAKE_EXE#
      .stack 256
      .data
n equ 10                    ;Кількість елементів масивів
x dw 1,2,3,4,n-4 dup (1)        ;Масив: 1,2,3,4,1,1,1,1,1,1
y dw 5,6,7,8,9,n-5 dup (2)      ;Масив: 5,6,7,8,9,2,2,2,2,2
z dw n dup (?)                ;Для масиву суми
s dw 0                      ;Для скалярного добутку
      .code
    MOV     AX, @data       ;Встановлення DS через акумулятор
    MOV     DS, AX
    mov SI,0000                 ;Обнуління індексного регістра
    mov CX,n                ;Завантаження лічильника
Cycl: mov AX,x[SI]              ;AX=Xi
    add AX,y[SI]                ;AX=Xi+Yi
    mov z[SI],AX                ;Zi=Xi+Yi
    mov AX,x[SI]                ;AX=Xi
    imul y[SI]              ;AX=Xi*Yi
    add s,AX                ;S=S+Xi*Yi
    add SI,2                ;Збільшення індексу на 2
    loop Cycl               ;Перехід на початок циклу
    HLT
```

4. Оброблення масивів МП I80X86

Приклад 7 (оброблення двомірного масиву):

Маємо матрицю слів a розміром 4x3. Необхідно ввести з клавіатури її елементи. Якщо елементи матриці позначаються індексами *i = 0,1,2,3 і j =0,1,2*, то зміщення (*i,j*) – го елементу матриці відносно її початку буде $2*i+ j*2^3$.

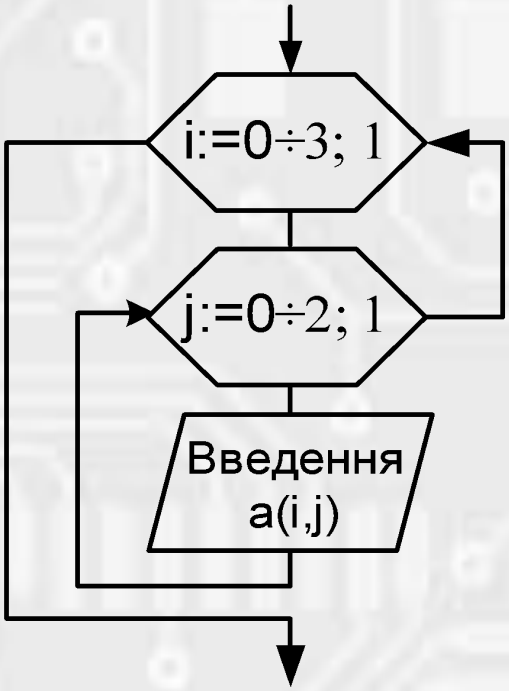
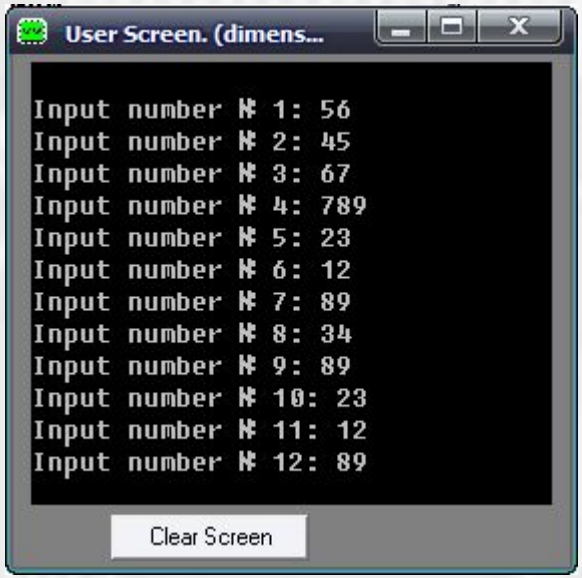


Рис. 5. Організація введення за допомогою вкладених циклів



Відображення введення елементів матриці на екрані користувача

Реалізуємо лічильник обох вкладених циклів в CX

4. Оброблення масивів МП I80X86

```
 #MAKE_EXE#
include 'emu8086.inc'
      .stack 128
      .data
m equ 4
n equ 3
a dw m dup(n dup(?)) ;Двовимірний масив
      .code
       mov AX, @data
       mov DS, AX
       xor bx,bx     ;Обнуління BX
       xor ax,ax
       mov cx,m      ;Завантаження лічильника
                     ;зовнішнього циклу
C_ext: push cx       ;Збереження лічильника
                     ;зовнішнього циклу
       mov cx,n      ;Завантаження лічильника
                     ;внутрішнього циклу
       xor si,si     ;Обнуління SI
C_int: inc ax
GOTOXY 0,al
PRINT "Input number № "
CALL print_num_uns
PRINT ": "
       push cx
CALL SCAN_NUM ;Введення числа
       mov a[bx][si],cx ; через CX
       pop cx
       add si,2          ;Збільшення
                  ;внутрішнього індексу
       loop C_int        ;n разів
       add bx,2*n        ;Збільшення
                  ;зовнішнього індексу
       pop cx            ;Відновлення
        ;лічильника зовнішнього циклу
       loop c_ext        ;m разів
       hlt
DEFINE_SCAN_NUM
DEFINE_PRINT_NUM
DEFINE_PRINT_NUM_UNS
END
```

4. Оброблення масивів МП I80X86

```
InMatr proc
      pusha
      mov cx,ax
C_ext: push cx
      mov cx,dx
      xor si,si
C_int: inc t
GOTOXY 0,t
PRINT "Input number № "
      push ax
      xor ax,ax
      mov al,t
CALL print_num_uns
      pop ax
PRINT ": "
      push cx
CALL SCAN_NUM
      mov a[bx][si],cx
      pop cx
      add si,2
      loop C_int
      shl dx,1
      add bx,dx
      shr dx,1
      pop cx
      loop c_ext
      popa
      ret
```

```
 #MAKE_EXE#
include 'emu8086.inc'
      .stack 128
      .data
m equ 4
n equ 3
t db 0
a dw m dup(n dup(?))
ОПИС ПРОЦЕДУРИ
.code
      mov AX, @data
      mov DS, AX
      mov ax,m
      mov dx,n
      lea bx,a
CALL InMatr
      hlt
DEFINE_SCAN_NUM
DEFINE_PRINT_NUM
DEFINE_PRINT_NUM_UNS
END
```

Процедура введення матриці з клавіатури (значення m передається в регістрі AX, значення n передається в регістрі DX, адреса матриці a – в регістрі BX.

Рекомендована література

1. *Юров В.И. Assembler. Учебник для вузов. 2-е изд. – СПб.: Питер, 2003.*
2. *Зубков С.В. Assembler для DOS, Windows и Unix. – М.: ДМК Пресс, 2000.*
3. *Митницкий В.Я. Архитектура IBM PC и язык Ассемблера: Учеб. Пособие. – М: МФТИ, 2000.*
4. *Схемотехніка електронних систем: У 3 кн. Кн. 3. Мікропроцесори та мікроконтролери: Підручник / В.І. Бойко, А. М. Гуржий, В.Я. Жуйков та ін. – К.: Вища шк., 2004.*
5. *Микропроцессорный комплект К1810: Структура, программирование, применение: Справочная книга / Ю.М. Казаринов, В.Н. Номоконов, Г.С. Подклетнов, Ф.В. Филиппов; Под ред. Ю.М. Казаринова. – М.: Высш. шк., 1990.*