Лекция 13.

- Организация циклов
- Работа с массивом данных
- Строковые команды для работы с массивами

Организация циклов

Организовать программный цикл командами процессора можно несколькими способами:

- вести счетчик циклов – увеличивающий (в регистре или памяти)
- вести счетчик циклов - уменьшающий
- использовать команду LOOP

Пример. Организовать цикл на 10 повторов.
В качестве счетчика циклов используем регистр BL

Способ 1. Суммирующий счетчик

```
      mov  bl, 0   ; сч ←0
m1: команды        «тело цикла»
    . . .
    inc  bl        ; сч ← сч +1
    cmp  bl, 10    ; сравнение с 10
    jb short m1    ; переход, если <
     . . .
```

Способ 2. Вычитающий счетчик

```
      mov  bl, 10    ; сч ← 10
m1:  команды         «тело цикла»
     . . .
     dec  bl         ; сч ← сч -1
     jnz short m1    ; переход,
                       если  флаг ZF≠1
      . . .
```

Этот способ эффективнее, так как требует меньше команд для реализации

Использование команды LOOP

- Команда процессора LOOP метка команды использует регистр CX в качестве вычитающего счетчика
- Исполнение команды LOOP процессором:

  CX ← CX – 1

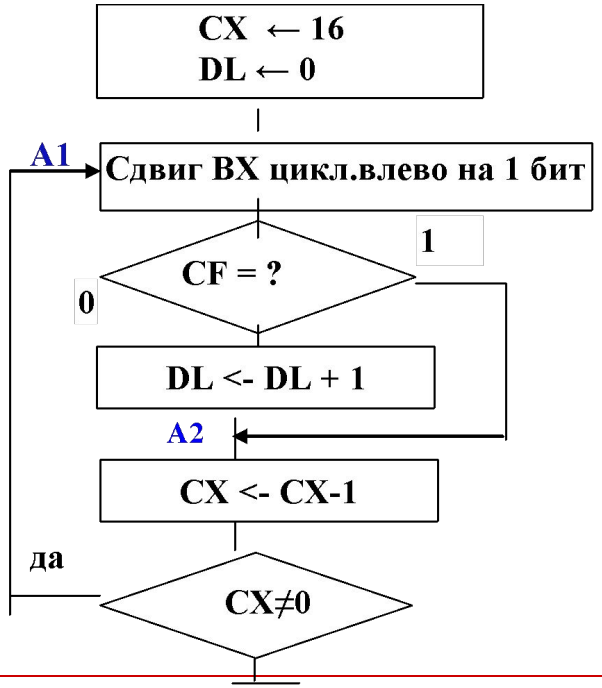
  переход по адресу, указанному меткой, если CX≠0

Пример: реализация цикла на 10 повторов

```
      mov  cx, 10
m1:   команды
      . . . .          Тело цикла
      loop  m1   ; CX ←CX-1 и
                 переход по адресу m1,
                 если CX ≠0
```

Пример: определить количество нулевых бит в регистре BX.

1. Односегментная программа
2. Размещение данных в регистрах:
   - CX – счетчик циклов
   - DL – количество нулей
3. Детальный алгоритм




```
csg segment  use16
   assume cs: csg
a1:     mov  cx, 16
        mov  dl, 0
cycl:   rol  bx, 1
        jc short a2
        inc dl
a2:     loop  cycl
; == вызов в ОС для выгрузки кода ==
        mov ah, 4ch
        int  21h
csg   ends
        end a1
```

Работа с массивами данных

«Массив» - способ размещения данных в памяти, при котором данные имеют одинаковый формат и размещены в памяти последовательно.

Такие особенности размещения позволяют организовать циклическую и единую логику обработки каждого «элемента массива»

Пример: Выделение места под массив данных в исходной программе

- массив из 5-ти конкретных однобайтных чисел

```
Mass1   db   1,  -2,  2,  3, -3
```

- резервирование памяти для массива из 10-ти двухбайтных значений

```
Mass2   dw   10 dup (?)
```

Организация цикла для работы с массивом данных

1. Организовать вычитающий счетчик для отсчета количества циклов
2. Обеспечить в каждом цикле изменение внутрисегментного адреса элемента массива с учетом формата данных.

   Это возможно только при использовании косвенной адресации
3. Косвенную внутрисегментную адресацию элементов массива можно реализовать двумя способами:

- *косвенно заданный адрес;*
- *косвенно заданное смещение от прямого адреса*

Пример: Увеличить на 1 каждый элемент в массиве из 5 двухбайтных значений

Способ 1. Косвенно задаем внутрисегментный адрес

Для занесения символического внутрисегментного адреса в регистр предназначена команда LEA. Для занесения числового внутрисегментного адреса используется MOV

Размещение данных в памяти и регистрах:
ds:mass – адрес начала массива в памяти
регистр si – для косвенного задания внутрисегментного адреса элемента
регистр cx – вычитающий счетчик циклов

```
        . . .
        lea  si, mass          ; si ← адрес начала массива
        mov cx, 5              ; счетчик циклов ← 5
m1:     inc word ptr ds:[si]   ; инкрементация значения элемента массива ,
                               находящегося по адресу ds:[si]
        inc si                 ; увеличение внутрисегментного адреса в si на 2
        inc si
        loop m1
        . . .
```

Способ 2. Косвенно задаем «смещение» в байтах от адреса начала массива до нужного элемента массива

Размещение данных в памяти и регистрах:
ds:mass – адрес начала массива в памяти
регистр si – для косвенного задания смещения от адреса начала массива
регистр cx – вычитающий счетчик циклов

```
        .   .   .
        mov  si, 0          ; si ← смещение в байтах от адреса mass
        mov  cx, 5
m1:     inc  word ptr ds: mass [si] ; инкрементация значения элемента
                                    массива, находящегося по адресу ds: mass+si
        inc  si             ; увеличение адресного смещения в массиве на 2
        inc  si
        loop  m1
        .   .   .   .
```

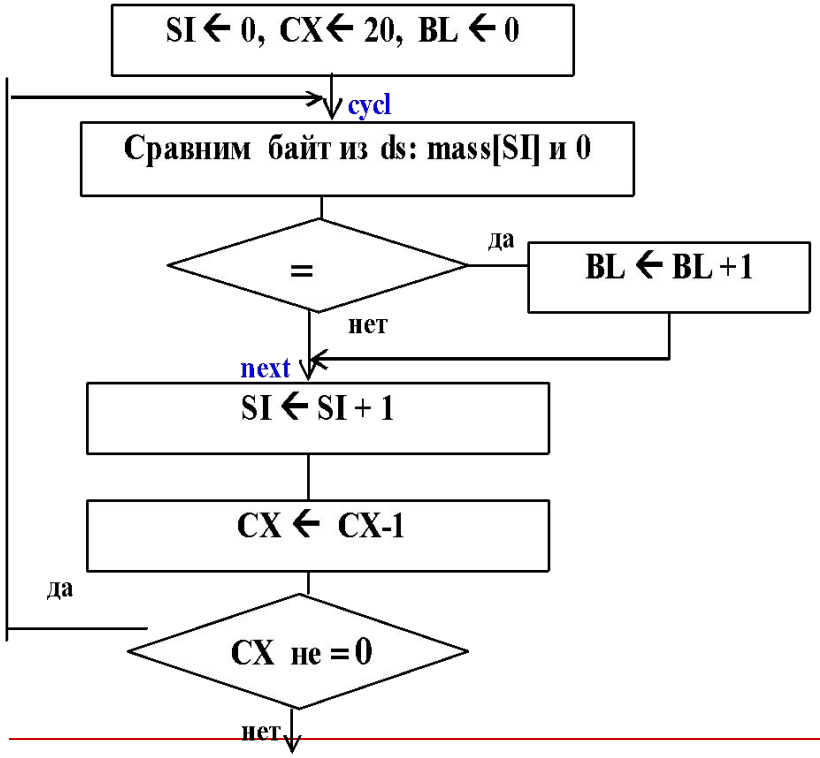
Напоминание!

- Для косвенного задания адреса или адресного смещения могут использоваться регистры: SI, DI, BX, BP и их комбинации (см.Лекцию 5)
- При косвенной адресации и косвенном задании смещения могут использоваться арифметические выражения, например: ds:adr[si-3], cs:[si+bx+1]
- При косвенной адресации через BP и отсутствии указателя сегмента, процессор по умолчанию использует указатель SS

```
mov  [bp+1], bx          ; выполнение:  ss:[bp+1]← bx
```

Пример. Определить количество нулевых байтов в массиве из 20 однобайтных кодов. Используем косвенное адресное смещение

ds:mass – адрес начала массива
Si – адресное смещение от начала массива,
CX- счетчик циклов, BL – количество нулей,




```
        .   .   .   .
mass    db   20 dup (?)
        .    .    .
        mov  bl, 0
        mov si, 0
        mov cx, 20
cycl: cmp byte ptr ds:mass[si], 0
        jnz  short  next
        inc   bl
next: inc si
        loop cykl
          .   .   .
```

Строковые команды для работы с массивами

Это короткие (однобайтные) команды с операндами «по умолчанию» для эффективной работы с большими массивами данных в памяти

- Сравнение данных с адресами DS:SI и ES:DI (память-память!)
  CMPSB – байты, CMPSW – слова, CMPSD – двойные слова
- Сравнение AL/AX/EAX с данными по адресу ES:DI
  SCASB, SCASW, SCASD
- Чтение из DS:SI в AL/AX/EAX
  LODSB, LODSW, LODSD
- Запись из AL/AX/EAX по адресу ES:DI
  STOSB, STOSW, STOSD
- Пересылка из DS:SI в ES:DI (память-память!)
  MOVSB, MOVSW, MOVSD

Выполнение строковых команд процессором

- Адреса операндов всегда - DS:SI и/или ES:DI

- Строковая команда после выполнения изменяет адрес в SI и DI на длину элемента массива в зависимости от флага DF:

  при DF=0 - увеличивает, при DF=1 - уменьшает

- Строковая команда может иметь префикс повторения REP. Например:

  REP STOSB

  Команда с префиксом повторяется столько раз, сколько записано в регистре CX. При этом CX автоматически декрементируется. Повторения закончатся при CX=0.

Пример: Пересылка 100 слов из одного сегмента памяти в другой.
Адрес источника - ds:mass, адрес приемника - es:massnew.

1) обычный MOV

```
        lea di, Massnew
        lea si, Mass
        mov cx,100
a2:     mov ax, ds:[si]
        mov es:[di], ax
        inc si
        inc si
        inc di
        inc di
        loop a2
```

В цикле процессор исполнит:
100 x 7 = 700 команд

2) строковый MOVSW

```
lea  di, Massnew
lea  si, Mass
mov cx,100
cld             ; флаг DF ← 0
rep  movsw ; пересылка слов
                    es:[di] ← ds:[si]
          с увеличением адресов на 2
```

В цикле процессор исполнит:
100 x 1 =100 команд