Язык **ASSEMBLER**

Команды пересылки данных

Лекция
доцента кафедры ИВТ ГрГУ
кандидата технических наук
Ливак Е.Н.

Команды пересылки данных общего назначения

mov <операнд назначения>,<операнд-источник>

xchg <операнд1>,<операнд2>

mov – основная команда пересылки данных

- Схема команды:

  mov приемник**,** источник

- Назначение:

  пересылка данных между регистрами или регистрами и памятью.

- Алгоритм работы:

  копирование второго операнда в первый операнд.

- Состояние флагов после выполнения команды:

  выполнение команды не влияет на флаги

ПРИМЕРЫ

Per1 dw 100 ; 0000 – адрес **Per1**
Per2 dw 200 ; 0002 – адрес **Per2**
Per3 dd 10257h ; 0004 – адрес **Per3**

mov cx, Per1 cx 00 64 $_{10}$=64$_{16}$

mov bx, offset Per1 bx 00 00

mov bx, offset Per2 bx 00 02

mov ax, bx ax 00 02 00 02

ПРИМЕРЫ

Per1 dw 100 ; 0000 – адрес **Per1**
Per2 dw 200 ; 0002 – адрес **Per2**
Per3 dd 10257h ; 0004 – адрес **Per3**

mov cx, Per2 cx 00 c8 $00_{10}=c8_{16}$

mov ch, 20h cx

ch cl

20 c8

ПРИМЕРЫ

Per1 dw 100 ; 0000 – адрес **Per1**
Per2 dw 200 ; 0002 – адрес **Per2**
Per3 dd 10257h ; 0004 – адрес **Per3**

mov ax, offset Per2 ax 00 02

mov al,5 ax 00 05
ah al

mov ah,10+15 ax 19 05 $_{10}$=19$_{16}$

mov ax, -1 ax FF FF

ПРИМЕРЫ ОШИБОК

```
Per1  dw 100      ; 0000 – адрес Per1
Per2  dw 200      ; 0002 – адрес Per2
Per3  dd 10257h   ; 0004 – адрес Per3
```

~~mov dh, Per1~~ - *constant too large*

⇒ **mov dx, Per1**

~~mov dh, 1254h~~ - *constant too large*

⇒ **mov dx, 1254h**

mov ah, Fh - *Undefined symbol*

Fh - идентификатор с точки зрения транслятора

⇒ **mov ah, OFh**

Особенности применения команды **mov**

1. нельзя осуществить пересылку из одной области памяти в другую

mov Per1, Per2

illegal memory reference

need register in expression

⇒ нужно использовать в качестве промежуточного буфера любой доступный в данный момент регистр общего назначения

mov ax, Per2

mov Per1, ax

Фрагмент программы

```
masm
model   small
.data
x     db      5
y     db      ?
.code
start:
...
        mov     al,x
        mov     y,al
        ...
end     start
```

Особенности применения команды **mov**

2. нельзя загрузить в сегментный регистр значение непосредственно из памяти

~~**mov ds, Perem**~~

⇒ для выполнения такой загрузки нужно использовать промежуточный объект.

Это может быть регистр общего назначения или стек

mov ax, Perem

mov ds, ax

Стандартное начало программы

```
masm
Model small
.stack   100h
.data
    <описание данных>
.code
start:
 mov   ax,@data          ;@data - переменная
 mov   ds,ax
      <команды>
 mov    ax,4c00h         ; стандартный выход - ah=00h
 int  21h
end start
```

Особенности применения команды **mov**

3. нельзя переслать содержимое одного сегментного регистра в другой сегментный регистр.

(в системе команд нет соответствующей операции)

~~**mov es, ds**~~

⇒ использовать в качестве промежуточных все те же регистры общего назначения

```
mov     ax, ds
mov     es, ax
```

Особенности применения команды **mov**

4. нельзя использовать сегментный регистр cs в качестве операнда назначения.

mov cs, ax

mov cs, 100

Пара cs:ip всегда содержит адрес команды, которая должна выполняться следующей.

⇒ изменение командой mov содержимого регистра cs фактически означало бы операцию перехода, а не пересылки, что недопустимо.

Особенности применения команды **mov**

Совет

желательно использовать в качестве одного из операндов регистр al/ax/eax

в этом случае TASM генерирует более быструю форму команды mov

```
mov     al,5
mov     bl,al
```

Команда **XCHG (eXCHanGe)**

- для двунаправленной пересылки данных

xchg ax, bx

обменять содержимое регистров **ax** и **bx**

- можно, конечно,

```
mov dx, ax
mov ax, bx
mov bx, dx
```

- но операция обмена используется довольно часто, разработчики системы команд микропроцессора посчитали нужным ввести отдельную команду обмена **xchg**

Команда XCHG

- !! Операнды должны иметь один тип

  ~~**xchg ax, bl**~~

- !! Не допускается (как и для всех команд ассемблера) обменивать между собой содержимое двух ячеек памяти

  ~~**xchg Per1, Per2**~~

Пример

```
;поменять порядок следования байт в слове
ch1     label   byte                    5c f 8
        dw      0f85ch      [ch1]=5c, [chl+1]=f8
...
        mov     al,ch1          al = 5c
        xchg    ch1+1,al        al = f8, [chl+1]=5c
        mov     ch1,al           [ch1]=f8
                                        f 8 5 c
```