

Системное программирование

Лекция №5

Способы адресации памяти

Способы адресации

Способом, или режимом адресации называют процедуру нахождения операнда для выполняемой команды. Если команда использует два операнда, то для каждого из них должен быть задан способ адресации, причем режим адресации первого и второго операнда могут как совпадать, так и различаться. Операнды команды могут находиться в разных местах: непосредственно в составе кода команды, в каком-либо регистре, в ячейке памяти; в последнем случае существует несколько возможностей указания его адреса. Способы адресации являются элементом архитектуры процессора, отражая заложенные в нем возможности поиска операндов.

Регистровая адресация

Операнд (байт или слово) находится в регистре. Этот способ адресации применим ко всем программно-адресуемым регистрам процессора (кроме IP).

inc	CH	;Плюс 1 к содержимому CH
push	DS	;DS сохраняется в стеке
xchg	BX, BP	;BX и BP обмениваются содержимым
mov	ES, AX	;Содержимое AX пересылается в ES
mov	AL, BH	;8-разрядный режим
mov	AX, BX	;16-разрядный режим
mov	EAX, EBX	;32-разрядный режим

Непосредственная адресация

Операнд (байт или слово) указывается в команде и после трансляции поступает в код команды; он может иметь любой смысл (число, адрес, код ASCII), а также быть представлен в виде символического обозначения.

```
mov  AH,40h      ;Число 40h загружается в AH
mov  AL,'*'      ;Код ASCII символа '*' загружается в AL
int  21h         ;Команда прерывания с аргументом 21h
limit = 528      ;Число 528 получает обозначение limit
mov  CX,limit    ;Число, обозначенное limit, загружается
                    ;в CX
```

Пересылка относительных адресов (смещений):

; Сегмент данных

```
mes  db "Урок 1" ;Строка символов
```

;Сегмент команд

```
mov  DX,offset mes ;Адрес строки засылается в DX
```

Прямая адресация памяти

Адресуется память; адрес ячейки памяти (слова или байта) указывается в команде (обычно в символической форме) и поступает в код команды:

;Сегмент данных

mem1 dw 0 ;Слово памяти содержит 0

mem2 db 230 ;Байт памяти содержит 230

; Сегмент команд

inc mem1 ;Содержимое слова mem1 увеличивается на 1

mov DX,mem1 ;Содержимое слова mem1 загружается в DX

mov AL,mem2 ;Содержимое байта mem2 загружается в AL

Команды процессора, обращающиеся к памяти, могут в качестве первого байта своего кода содержать префикс замены сегмента, с помощью которого процессор определяет, из какого сегментного регистра взять сегментный адрес. Если префикс отсутствует, сегментный адрес берется из регистра DS (хотя для него тоже предусмотрен свой префикс). В ряде случаев префикс замены сегмента должен указываться в программе в явной форме. Такая ситуация возникает, например, если данные расположены в сегменте команд, что типично для резидентных обработчиков прерываний:

mov AX, CS : mem

Вывод символов в левый верхний угол экрана

```
mov  AX,0B800h      ;Сегментный адрес видеобuffers
mov  ES,AX          ;Отправим его в ES
mov  byte ptr ES:0,'!' ;Символ на 1-е знакоместо экрана
mov  byte ptr ES:2,'!' ;Символ на 2-е знакоместо экрана
```

Настроив регистр E5 на сегментный адрес видеобuffers B800h, мы пересылаем код знака «!» сначала по относительно адресу 0 (в самое начало видеобuffers, в байт со смещением 0), а затем на следующее знакоместо, имеющее смещение 2 (в нечетных байтах видеобuffers хранятся атрибуты символов, т. е. цвет символов и фона под ними). В обеих командах необходимо с помощью обозначения ES: указать сегментный регистр, который используется для адресации памяти.

Встретившись с этим обозначением, транслятор включит в код команды префикс замены сегмента, в данном случае код 26h.

Что будет, если использовать word ptr?

```
mov  AL,'!'
mov  ES:0,AL
```

Регистровая косвенная адресация памяти

Адресуется память (байт или слово). Относительный адрес ячейки памяти находится в регистре, обозначение которого **заключается в прямые скобки**. В МП 86 косвенная адресация допустима только через регистры BX, BP, SI и DI. При использовании регистров BX или BP адресацию называют **базовой**, при использовании регистров DI или SI — **индексной**.

```
mov    AX,0B800h           ;Сегментный адрес
mov    ES,AX               ;видеобуфера в ES
mov    BX,2000             ;Смещение к середине экрана
mov    byte ptr ES:[BX],!' ;Символ на экран
```

Если косвенная адресация осуществляется через один из регистров BX, SI или DI, то подразумевается сегмент, адресуемый через DS, поэтому при адресации через этот регистр обозначение *DS*: можно опустить:

```
mov    AX,0B800h           ;Сегментный адрес
mov    DS,AX               ;видеобуфера в DS
mov    BX,2000             ;Смещение к середине экрана
mov    byte ptr [BX],!'    ;Символ на экран
```

Регистровая косвенная адресация памяти

Регистры BX, SI и DI в данном применении совершенно равнозначны, и с одинаковым успехом можно воспользоваться любым из них:

```
mov    DI,2000          ;Смещение к середине экрана  
mov    byte ptr [DI],!' ;Символ на экран
```

Регистр же BP специально предназначен для работы со стеком, и при адресации через этот регистр в режимах косвенной адресации подразумевается сегмент стека; другими словами, в качестве сегментного регистра по умолчанию используется регистр SS.

Обычно косвенная адресация к стеку используется в тех случаях, когда необходимо обратиться к данным, содержащимся в стеке, без изъятия их оттуда (например, если эти данные приходится считывать неоднократно).

Регистровая косвенная адресация со смещением

Адресуется память (байт или слово); относительный адрес операнда определяется, как сумма содержимого регистра BX, BP, SI или DI и указанной в команде константы, иногда называемой смещением. Смещение может быть числом или адресом.

```
mov  AX,0B800h           ;Сегментный адрес
mov  ES, AX              ;видеобуфера в ES
mov  DI,80*2*24          ;Смещение к нижней строке экрана
mov  byte ptr ES:[DI], 'O' ;Символ на экран
mov  byte ptr ES:2[DI], 'K' ;Символ в следующую позицию
mov  byte ptr ES:4[DI], '!' ;Символ в следующую позицию
```

Иногда можно встретиться с альтернативными обозначениями того же способа адресации, которые допускает ассемблер. Вместо, например, 4[BX] можно с таким же успехом написать [BX+4], 4+[BX] или [BX]+4.

Пример использования базовой адресации со смещением при обращении к стеку

;Основная программа

```
push DS      ;В стек загружаются значения  
push ES      ;трех регистров,  
push SI      ;передаваемых подпрограмме  
call mysub   ;Вызов подпрограммы mysub  
              ;использующей эти параметры
```

;Подпрограмма mysub

```
mov BP,SP     ;Поместим в BP текущий адрес вершины  
стека  
mov AX,2[BP]  ;Читаем в AX последний параметр (SI)  
mov BX,4[BP]  ;Читаем в BX предыдущий параметр (ES)  
mov CX,6[BP]  ;Читаем в CX первый параметр (DS)
```

Если бы подпрограмма просто сняла со стека находящиеся там параметры, она первым делом изъяла бы из стека адрес возврата, и лишила бы себя возможности вернуться в основную программу

Пример заполнения массива из 10000 слов натуральным рядом чисел

;Сегмент данных

array dw 10000 dup (?)

;Сегмент команд

mov SI,0 ;Начальное значение индекса элемента в массиве

mov AX,0 ;Первое число-заполнитель

mov CX,10000;Число шагов в цикле (всегда в CX)

fill: mov array[SI],AX ;Занесение числа в элемент массива

inc AX ;Инкремент числа-заполнителя

add SI,2 ;Смещение в массиве к следующему

слову

loop fill ;Возврат на метку fill (CX раз)

Базово-индексная адресация

Адресуется память (байт или слово). Относительный адрес операнда определяется, как сумма содержимого следующих пар регистров:

[BX] [SI]	(подразумевается DS:[BX][SI])
[BX] [DI]	(подразумевается DS:[BX][DI])
[BP][SI]	(подразумевается SS:[BP][SI])
[BP][DI]	(подразумевается SS:[BP][DI])

Это чрезвычайно распространенный способ адресации, особенно, при работе с массивами. В нем используются два регистра, при этом одним из них должен быть базовый (BX или BP), а другим — индексный (SI или DI). Как правило, в одном из регистров находится адрес массива, а в другом — индекс в нем, при этом совершенно безразлично, в каком что.

Базово-индексная адресация

;Сегмент данных

```
аггау      dw 10000 dup (?)
```

;Сегмент команд

```
mov  BX,offset аггау ;Базовый адрес массива в базовом регистре
mov  SI,0             ;Начальное значение индекса элемента в массиве
mov  AX, 0            ;Первое число-заполнитель
mov  CX,10000         ;Число шагов в цикле
fill: mov  [BX] [SI],AX ;Отправим число в массив
inc  AX               ;Инкремент числа-заполнителя
add  SI,2             ;Смещение в массиве к следующему слову
loop fill             ;На метку fill (CX раз)
```

Повышение эффективности достигается за счет того, что команда занесения числа в элемент массива оказывается короче (так как в нее не входит адрес массива) и выполняется быстрее, так как этот адрес не надо каждый раз считывать из памяти.

Базово-индексная адресация со смещением

Адресуется память (байт или слово). Относительный адрес операнда определяется как сумма содержимого двух регистров и смещения.

```
Sims db 'QWERTUIOP{'  
      db 'ЙЦУКЕНГШЩЗХЪ'
```

Последовательность команд

```
mov  BX,12    ;Число байтов в строке  
mov  SI,6  
mov  DL,Sims[BX][SI]
```

загрузит в регистр DL элемент с индексом 6 из второго ряда, т.е. код ASCII буквы Г. Тот же результат можно получить, загрузив в один из регистров не индекс, а адрес массива:

```
mov  BX,offset Sims  
mov  SI,6  
mov  DL,12[BX][SI]
```

Адресация по базе с индексированием и масштабированием

Это самая полная возможная схема адресации, в которую входят все случаи, рассмотренные ранее, как частные. Полный адрес операнда можно записать как выражение, представленное на рисунке.

$$\left[\begin{array}{l} \text{CS: EAX} \\ \text{SS: EBX} \\ \text{DS: ECX} \\ \text{ES: EDX} \\ \text{FS: EBP} \\ \text{GS: ESP} \\ \text{ESI} \end{array} + \begin{array}{l} \text{EAX} \\ \text{EBX} \\ \text{ECX} \\ \text{EDX} \\ \text{EBP} \\ \text{ESI} \\ \text{EDI} \end{array} * \begin{array}{l} 1 \\ 2 \\ 4 \\ 8 \end{array} + \text{смещение} \right]$$