王爽汇编第八章,数据处理的两个基本问题

本章对前面的所有内容是具有总结性的。我们知道，计算机是进行数据处理、运算的机器，那么有两个基本的问题就包含在其中：

- (1) 处理的数据在什么地方?
- (2)要处理的数据有多长?

这两个问题，在机器指令中必须给以明确或隐含的说明，否则计算机就无法工作。

1. bx、si、di和bp

- 在8086CPU中，只有这4个寄存器可以用在 `[...]` 中进行内存单元的寻址。
- 在[...]中，这4个寄存器可以单个出现，或只能以4种组合出现： `bx和si` 、 `bx和di` 、 `bp和si` 、 `bp和di` 。
- 只要在[...]中使用寄存器bp，而指令中没有显性地给出段地址，段地址就默认在ss中。

```
mov ax,[bx]
mov ax,[si]
mov ax,[di]
mov ax,[bp]
mov ax,[bx+si]
mov ax,[bx+di]
mov ax,[bp+si]
mov ax,[bp+di]
mov ax,[bx+si+idata]
mov ax,[bx+di+idata]
mov ax,[bp+si+idata]
mov ax,[bp+di+idata]
```

下面的指令是错误的：

```
mov ax,[bx+bp]
mov ax,[si+di]
```

2. 机器指令处理的数据在什么地方

数据处理指令大致分为3类： `读取` 、 `写入` 、 `运算` 。

在机器指令这一层来讲，并不关心数据的值是多少，而关心指令执行前一刻，它将要处理的数据所在的位置。指令在执行前，所要处理的数据可以在3个地方： `CPU内部、内存、端口`

机器码	汇编指令	指令执行前数据的位置
8E1E0000	mov bx,[0]	内存，ds:0 单元
89C3	mov bx,ax	CPU 内部，ax 寄存器
BB0100	mov bx,1	CPU 内部，指令缓冲器

3．汇编语言中数据位置的表达

汇编语言中用3个概念来表达数据的位置。

- 立即数(idata)

```
;直接包含在机器指令中的数据(CPU指令缓冲器中)
;在汇编语言中称为:立即数(idata)
;在汇编指令中直接给出
mov ax,1
add bx,2000h
or  bx,00010000b
mov al,'a'
```

- 寄存器

```
;指令处理的数据在寄存器中,
;在汇编指令中给出相应的寄存器名
mov ax,bx
mov ds,ax
push bx
mov ds:[0],bx
push ds
mov ss,ax
mov sp,ax
```

- 段地址(SA)和偏移地址(EA)

```
;数据在内存中，用[...]或者sreg:[...]
;1.默认ds
mov ax,[0]
mov ax,[bx]
mov ax,[di]
mov ax,[bx+di+8]
;2.默认ss
mov ax,[bp]
mov ax,[bp+8]
mov ax,[bp+si+8]
;3.显性指定段寄存器
mov ax,ds:[bp]      ;(ax)=(ds*16+(bp))
mov ax,es:[bx]      ;(ax)=(es*16+(bx))
mov ax,ss:[bx+si]   ;(ax)=(ss*16+(bx)+(si))
mov ax,cs:[bx+si+8];(ax)=(cs*16+(bx)+(si)+8)
```

4．寻址方式

当数据存在内存中的时候，可以用多种方式来给定内存单元的偏移地址，这种定位内存单元的方法一般称为 寻址方式 。

表 8.2　寻址方式小结

寻址方式	含　义	名　称	常用格式举例
[idata]	EA=idata;SA=(ds)	直接寻址	[idata]
[bx]	EA=(bx);SA=(ds)	寄存器间接寻址	[bx]
[si]	EA=(si);SA=(ds)		
[di]	EA=(di);SA=(ds)		
[bp]	EA=(bp);SA=(ss)		
[bx+idata]	EA=(bx)+idata;SA=(ds)	寄存器相对寻址	用于结构体： [bx].idata 用于数组： idata[si],idata[di] 用于二维数组： [bx][idata]
[si+idata]	EA=(si)+idata;SA=(ds)		
[di+idata]	EA=(di)+idata;SA=(ds)		
[bp+idata]	EA=(bp)+idata;SA=(ss)		
[bx+si]	EA=(bx)+(si);SA=(ds)	基址变址寻址	用于二维数组： [bx][si]
[bx+di]	EA=(bx)+(di);SA=(ds)		
[bp+si]	EA=(bp)+(si);SA=(ss)		
[bp+di]	EA=(bp)+(di);SA=(ss)		
[bx+si+idata]	EA=(bx)+(si)+idata; SA=(ds)	相对基址变址寻址	用于表格(结构)中的数组项： [bx].idata[si] 用于二维数组： idata[bx][si]
[bx+di+idata]	EA=(bx)+(di)+idata; SA=(ds)		
[bp+si+idata]	EA=(bp)+(si)+idata; SA=(ss)		
[bp+di+idata]	EA=(bp)+(di)+idata; SA=(ss)		

5．指令要处理的数据有多长

`8086CPU` 的指令，可以处理两种尺寸的数据，byte和word

- 通过寄存器名指明要处理的数据的尺寸。

例如：　`mov al, ds:[0]`　寄存器al指明了数据为1字节

- 在没有寄存器名存在的情况下，用操作符 `X ptr` 指明内存单元的长度，X在汇编指令中可以为 `word` 或 `byte` 。

例如：`mov byte ptr ds:[0], 1`　byte ptr 指明了指令访问的内存单元是一个字节单元

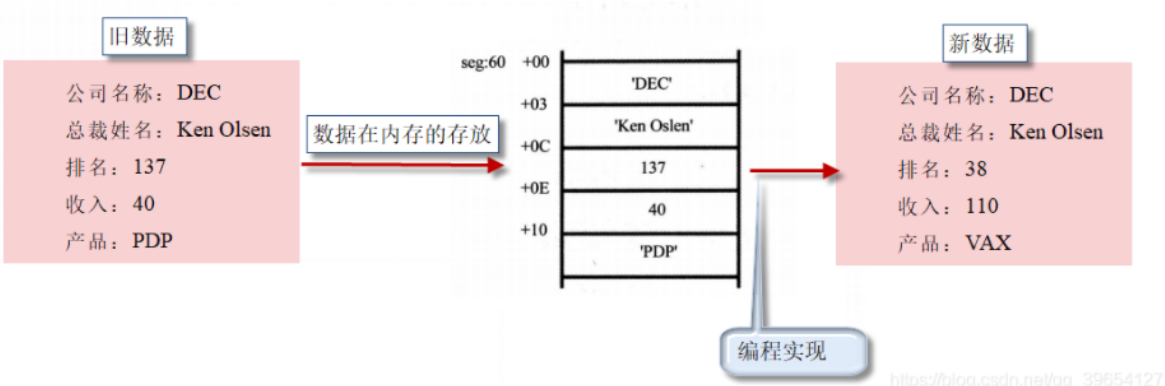
- 有些指令默认了访问的是字单元还是字节单元

例如，`push [1000H]`，push 指令只进行字操作。

插播一点 `x86` 的知识，在X86架构中一般会见到下面汇编指令

`xxx dword ptr xxx`，这种就是指明是长度是4字节,也是Win中C++的 `DWORD` 类型 `unsigned long` 。

6．寻址方式的综合应用



汇编代码：

```
code segment
main
      mov ax,data
      mov ds,ax
      mov bx,60h ;确定记录地址, ds:bx

      mov word ptr [bx+0ch],38;排名字段改为38 [bx].0ch
      add word ptr [bx+0eh],70;收入字段增加70 [bx].0eh
      mov si,0 ;用si来定位产品字符串中的字符
      mov byte ptr [bx+10h+si],'V';[bx].10h[si]
      inc si;自增
      mov byte ptr [bx+10h+si],'A'
      inc si;自增
      mov byte ptr [bx+10h+si],'X'
code ends
end main
```

c语言代码：

```c
/*定义一个公司记录的结构体*/
struct company
{
    char cn[3];/*公司名称*/
    char hn[9];/*总裁姓名*/
    int pm;/*排名*/
    int sr;/*收入*/
    char cp[3];/*著名产品*/
};

int main()
{
    /*定义一个公司记录的变量,内存中将存有一条公司的记录*/
    struct company dec = {"DEC","Ken Olsen",137,40,"PDP"};

    int i;
    dec.pm = 38;
    dec.sr = dec.sr + 70;

    i = 0;
    dec.cp[i] = 'V';
    i++;
    dec.cp[i] = 'A';
    i++;
    dec.cp[i] = 'X';
```

```
26
27      return 0;
28  }
```

7. dd、dup、mul、div指令

7.1. dd 伪指令

db和dw定义字节型数据和字型数据。

dd用来定义dword (double word,双字) 4字节类型数据的伪指令

7.2. dup操作符

dup在汇编中同db、dw、dd等一样，也是由编译器识别处理的符号。

它和db、dw、dd等数据定义伪指令配合使用，用来进行数据的重复。

```
114   ;定义了3个字节，他们的值都是0
115   ;相当于 db 0,0,0
116   db 3 dup(0);
117
118   ;定义了9个字节
119   ;相当于db 0,1,2,0,1,2,0,1,2
120   db 3 dup(0,1,2)
121
122   ;定义了18个字节
123   ;相当于db 'abc','ABC'.....
124   db 3 dup('abc','ABC')
125
126   ;总结:前面是类型 循环次数 (循环体)
```

7.3. mul乘法指令

mul乘法指令，之前在介绍寄存器章节的时候有提到过，会将结果保存在ax中。

注意：(8086中使用mul做乘法的时候：相乘的两个数：要么都是8位，要么都是16位)。

- 8位： AL 中和 8位寄存器 或 内存字节单元 中；
- 16位： AX 中和 16位寄存器 或 内存字节单元 中。

结果：

- 8位：AX中；
- 16位：DX(高位)和AX(低位)中。

```
1   mov al,100
2   mov bl,10
3   mul bl;100 * 10 = 1000
4   ;结果 (ax) = 1000(3E8H)
5
6   mov ax,100
7   mov bx,10000
8   mul bx;100 * 10000 = 1000000
9   ;结果 (dx) = 000F ,(ax) = 4240H (F4240H = 1000000)
```

7.4. div除法指令

- 除数：有8位和16位两种，在一个寄存器或内存单元中。
- 被除数：默认放在AX或DX和AX中，
  如果除数为8位，被除数则为16位，默认在AX中存放；
  如果除数为16位，被除数则为32位，在DX和AX中存放，DX存放高16位，AX存放低16位。
- 结果：
  如果除数为8位，则AL存储除法操作的商，AH存储除法操作的余数；
  如果除数为16位，则AX存储除法操作的商，DX存储除法操作的余数。

```
;利用除法指令计算100001/100。
;100001D = 186A1H
mov dx, 1
mov ax, 86A1H ;(dx)*10000H+(ax)=100001
mov bx, 100
div bx

;利用除法指令计算1001/100
mov ax, 1001
mov bl, 100
div b1
```

> 参考文献：
> https://blog.csdn.net/qq_39654127/article/details/88698911 （CSDN王爽《汇编语言》笔记）
> 《王爽汇编语言第4版 第8章》