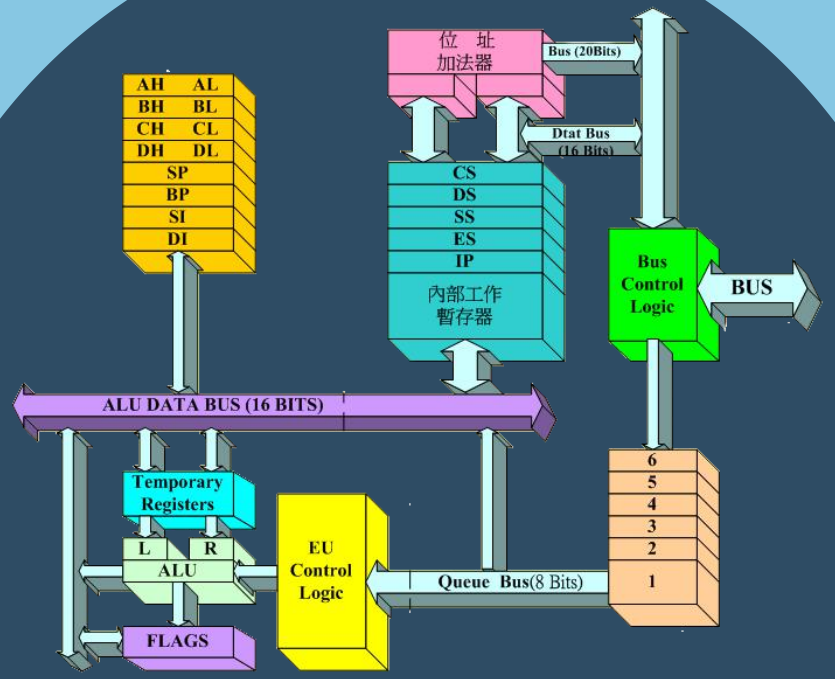
段前缀的使用

贺利坚　主讲



汇编语言程序设计

Assembly Language

引入段前缀：一个“异常”现象及对策

```
C:\>debug
-a
073F:0100 mov ax, 2000
073F:0103 mov ds, ax
073F:0105 mov al, [0]
073F:0108 mov bl, [1]
073F:010C mov cl, [2]
073F:0110 mov dl, [3]
073F:0114
-u
073F:0100 B80020        MOV     AX,2000
073F:0103 8ED8          MOV     DS,AX
073F:0105 A00000        MOV     AL,[0000]
073F:0108 8A1E0100      MOV     BL,[0001]
073F:010C 8A0E0200      MOV     CL,[0002]
073F:0110 8A160300      MOV     DL,[0003]
073F:0114 0000          ADD     [BX+SI],AL
```

Debug中，mov al, [0]的功能是

——将DS:0存储单元的值传给AL

```
1  assume cs:code
2  code segment
3          mov ax,2000h
4          mov ds,ax
5          mov al,[0]
6          mov bl,[1]
7          mov cl,[2]
8          mov dl,[3]
9
10         mov ax,4c00h
11         int 21h
12 code ends
13 end
```

编译(masm)并连接(link)后...

```
C:\>debug p5-3.exe
-u
076A:0000 B80020        MOV     AX,2000
076A:0003 8ED8          MOV     DS,AX
076A:0005 B000          MOV     AL,00
076A:0007 B301          MOV     BL,01
076A:0009 B102          MOV     CL,02
076A:000B B203          MOV     DL,03
076A:000D B8004C        MOV     AX,4C00
076A:0010 CD21          INT     21
076A:0012 00508B        ADD     [BX+SI-25
```

编译好的程序中，mov al, [0]变成了将常量0传给AL

对策：在[idata]前显式地写上段寄存器

```
mov ax,2000h
mov ds,ax
mov bx,0
mov al,ds:[bx]
```

```
mov ax,2000h
mov ds,ax
mov al,ds:[0]
```

小结（在程序中）：

mov al,[0]：(al)=0，同mov al,0

mov al,ds:[0]：(al)=((ds)*16+0)

mov al,[bx]：(al)=((ds)*16+(bx))

mov al,ds:[bx]：与mov al,[bx]相同

这些出现在访问内存单元的指令中，用于显式地指明内存单元的段地址的“ds:”、“cs:”、“ss:”或“es:”，在汇编语言中称为**段前缀**。

访问连续的内存单元——loop和[bx]联手！

- 问题：计算ffff:0~ffff:b字节单元中的数据的和，结果存储在dx中
- 分析：

（1）运算后的结果是否会超出 dx 所能存储的范围？

ffff:0 ~ ffff:b内存单元中的数据是字节型数据，范围在0 ~ 255之间，12个这样的数据相加，结果不会大于 65535，可以在dx中存放下。

（2）是否可以将 ffff:0 ~ ffff:b中的数据直接累加到dx中？

add **dx**, ds:[addr] ;(dx)=(dx)+?

期望：取出内存中的8位数据进行相加

实际：取出的是内存中的16位数据

（3）是否可以将 ffff:0 ~ ffff:b中的数据直接累加到dl中？

add **dl**, ds:[addr] ;(dl)=(dl)+?

期望：取出内存中的8位数据相加

实际：取出的是内存中的8位数据，但很有可能造成进位丢失。

- 对策：取出8位数据，加到16位的寄存器

```
mov al, ds:[addr]
mov ah, 0
add dx, ax
```

程序：计算ffff:0~ffff:b单元中的数据的和，结果存储在dx中

```
assume cs:code
code segment
    mov ax,0ffffh
    mov ds,ax

    mov dx,0

    mov al,ds:[0]
    mov ah,0
    add dx,ax

    mov al,ds:[1]
    mov ah,0
    add dx,ax

    mov al,ds:[2]
    mov ah,0
    add dx,ax

    mov al,ds:[3]
    mov ah,0
    add dx,ax

    mov al,ds:[4]
    mov ah,0
    add dx,ax

    mov al,ds:[5]
    mov ah,0
    add dx,ax

    mov al,ds:[6]
    mov ah,0
    add dx,ax

    mov al,ds:[7]
    mov ah,0
    add dx,ax

    mov al,ds:[8]
    mov ah,0
    add dx,ax

    mov al,ds:[9]
    mov ah,0
    add dx,ax

    mov al,ds:[0ah]
    mov ah,0
    add dx,ax

    mov al,ds:[0bh]
    mov ah,0
    add dx,ax

    mov ax,4c00h
    int 21h
code ends
end
```

$$sum = \sum_{x=0}^{0bh} (0ffffh \times 10h + x)$$

- 改进
  - 用loop循环
- 方法
  - 循环次数由cx控制
  - 循环中要访问的内存单元的偏移地址放到 bx中，随循环递增，访问连续的内存单元。

```
1  assume cs:code
2  code segment
3      mov ax,0ffffh
4      mov ds,ax
5
6      mov bx,0
7      mov dx,0
8      mov cx,12
9
10 s:  mov al,[bx]
11     mov ah,0
12     add dx,ax
13     inc bx
14     loop s
15
16     mov ax,4c00h
17     int 21h
18 code ends
19 end
```

段前缀的使用

问题：将内存ffff:0~ffff:b中的数据拷贝到 0:200~0:20b单元中。

```
1  ; 初始方案
2  assume cs:code
3  code segment
4          mov bx,0
5          mov cx,12
6
7    s:    mov ax,0ffffh
8          mov ds,ax
9          mov dl,[bx]
10
11         mov ax,0020h
12         mov ds,ax
13         mov [bx],dl
14
15         inc bx
16         loop s
17
18         mov ax,4c00h
19         int 21h
20 code ends
21 end
```

```
1  ;使用附加段寄存器
2  assume cs:code
3  code segment
4          mov ax,0ffffh
5          mov ds,ax
6          mov ax,0020h
7          mov es,ax
8
9          mov bx,0
10         mov cx,12
11
12   s:    mov dl,[bx]
13         mov es:[bx],dl
14         inc bx
15         loop s
16
17         mov ax,4c00h
18         int 21h
19 code ends
20 end
```