第五章 [BX] 和 loop

[BX]

```
mov ax, [bx]
```

将**bx中存放的内容作为偏移地址**，段地址默认在ds中，将组合之后的地址中的数据放入到ax寄存器中。

如果bx=1，那么上面的指令就相当于

```
mov ax, [1]
```

同样的也可以使用如下的形式

```
mov [bx], ax
```

和之前介绍`[1]`时候的用法是一样的。

loop

loop指令是用于循环的指令。

loop指令的格式为**loop 标号**

loop指令执行的步骤

1. `cx -= 1`， 将cx寄存器中的内容减1
2. 判断cx寄存器中值，如果不为0则跳转到标号处。如果为0继续执行下一条指令。

cx用于控制循环的次数

loop示例

例： 使用loop指令计算2^10次方，存放到ax寄存器中。

```
assum cs:code

code segment
        mov ax, 1
        mov cx, 10
s:
        add ax, ax
        loop s

        mov ax, 4c00H
        int 21H
code ends
end
```

其中的s就是标号。当然也可以使用其他的任何的名字，只要不和系统的保留字冲突就行了

最简单的loop框架

```
mov cx, 循环次数
标号:
            循环的内容
loop 标号
```

loop加[bx]的简单运用

例： 将内存中的`ffff:0~ffff:b`单元中的所有的数据累加，结果存放在dx中。

```
assume cs:code

```

```
code segment
        mov ax, 0ffffH ; 如果数字以字母开始必须要加上一个0
        mov ds, ax ; 定位数据的地址
        mov bx, 0
        mov dx, 0
        mov cx, 12 ; 循环的次数为12次
s:
        mov al, [bx]
        mov ah, 0 ; 这两条指令相当于将内存中的8位数据赋值到了16位寄存器ax上
        add dx, ax
        inc bx ; 相当与 add bx, 1
        loop s

        mov ax, 4c00H
        int 21H
code ends
end
```

段前缀

在`mov ax, [bx]`中，段地址默认由`ds`寄存器给出，但是我们也是可以指定寄存器的，但是必要要是段寄存器。段寄存器只有以下的四个`ds, cs, ss, es`

```
mov ax, ds:[bx]
mov ax, cs:[bx]
mov ax, ss:[bx]
mov ax, es:[bx]


mov ax, cs:[0]
mov ax, ss:[2]
mov ax, es:[4]
```

段前缀的应用

例，将`ffff:0~ffff:b`中的数据复制到`0:200~0:20b`的单元中。

使用两个段寄存器作为段前缀，这样更简单。

```
assume cs:code

code segment
        mov ax, 0ffffh
        mov ds, ax
        mov ax, 20h ; 0:200~0:20b也可以看作是20:0~20:b
        mov es, ax
        mov bx, 0
        mov cx, 12
copy:
        mov dl, [bx] ; 需要借助一个8位寄存器
        mov es:[bx], dl
        inc bx
        loop copy

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