

# 第六章 多段程序

## 使用数据段

问题：编程计算以下8个数据的和，结果存在ax寄存器中。

```
1 | 0123h, 0456h, 0789h, 0abch, 0defh, 0fedh, 0cbah,  
   0987h
```

按照以学的知识在代码中应该有以下类型的数据。

```
1 | mov ax, 0123h  
2 | add ax, 0456h  
3 | ....  
4 | add ax, 0987h副了
```

非常的麻烦，可以使用数据段将这些数据存放在某一段内存空间中。

```
1 | assume cs:code  
2 |  
3 | code segment  
4 |         dw 0123h, 0456h, 0789h, 0abch, 0defh,  
   0fedh, 0cbah, 0987h ; 定义数据  
5 |  
6 |         mov bx, 0  
7 |         mov ax, 0  
8 |  
9 |         mov cx, 8  
10 | a:  
11 |         add ax, cs:[bx]  
12 |         add bx, 2 ; 处理的是字型数据，每次向后移动两个  
   字节
```

```

13         loop a
14
15         mov ax, 4c00h
16         int 21h
17 code ends
18 end

```

`dw`是`define word`的缩写，用来定义字型数据。位于程序的最开始的地方，也就是`cs`段寄存器执行的位置，此时我们可以使用`cs`作为他的段前缀。

但是以上程序有问题，就是第一行不是程序语句而是数据。在计算机内部，指令和数据都是二进制数据，所以会将数据作为指令来执行。（`cs:ip`指向这些数据）。

我们可以通过一下的方式修改`ip`初始的指向。

```

1  assume cs:code
2
3  code segment
4      dw 0123h, 0456h, 0789h, 0abch, 0defh,
      0fedh, 0cbah, 0987h ; 定义数据
5  start: ; 添加start标号，定位指令的入口
6      mov bx, 0
7      mov ax, 0
8
9      mov cx, 8
10 a:
11     add ax, cs:[bx]
12     add bx, 2 ; 处理的是字型数据，每次向后移动两个
      字节
13     loop a
14
15     mov ax, 4c00h
16     int 21h
17 code ends
18 end start ; 修改此处为 end start

```

# 简单程序的框架

```
1  assume cs:code
2  code segment
3      定义数据
4  start:
5      代码
6  code ends
7  end start
```

## 使用栈

定义栈的关键是要有一段内存空间可以用作栈来使用。这和定义数据是一样的。

```
1  assume cs:code
2
3  code segment
4      dw 0123h, 0456h, 0789h, 0abch, 0defh,
      0fedh, 0cbah, 0987h ; 定义数据
5      dw 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0 ; 定义栈, 大小为16个字
6  start: ; 添加start标号, 定位指令的入口
7      mov ax, cs
8      mov ss, ax
9      mov sp, 30h ; 将栈顶设置为cs:30h
10
11      mov ax, 4c00h
12      int 21h
13  code ends
14  end start ; 修改此处为 end start
```

## 将数据，代码，栈放入不同的段

上面将数据，栈，代码全都放到了代码段中，看起来十分的拥挤，而且使用起来也不舒服。

通过下面的方式可以将数据，栈，代码放入不同的段中。

例： 将给定的八个数据倒转

使用栈可以非常容易的解决这个问题

```
1  assume cs:code, ds:data, ss:stack
2
3  data segment
4      dw 0123h, 0456h, 0789h, 0abch, 0defh,
5      0fedh, 0cbah, 0987h
6
7  data ends
8
9  stack segment
10     dw 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
11     0, 0, 0, 0
12 stack ends
13
14 code segment
15 start:
16
17     mov ax, stack
18     mov ss, ax
19     mov sp, 20h
20
21     mov ax, data
22     mov ds, ax
23
24     mov bx, 0
25     mov cx, 8
26
27 a:
28     push [bx]
29     add bx, 2
30     loop s ; 将这八个数据入栈
31
32     mov bx, 0
```

```

28         mov cx, 8
29 b:
30         pop [bx]
31         add bx, 2
32         loop b ; 将八个数据出栈，此时完成倒转
33
34         mov ax, 4c00h
35         int 21h
36 code ends
37 end start

```

## 多段程序的框架

```

1  assume cs:code, ds:data, ss:stack
2
3  data segment
4      定义数据
5  data ends
6
7  stack segment
8      定义栈空间
9  stack ends
10
11 code segment
12 start:
13     mov ax, data
14     mov ds, ax ; 数据段
15
16     mov ax, stack
17     mov ss, ax
18     mov sp, xxh ; 栈段
19
20     程序代码
21
22     mov ax, 4c00h
23     int 21h ; 程序返回

```

24	code ends
25	end start