第6章 包含多个段的程序

第6章 包含多个段的程序

- 6.1 在代码段中使用数据
- 6.2 在代码段中使用栈
- 6.3 将数据、代码、栈放入不同的段

引言

- 问题：如果程序需要用其他空间来存放数据，可以使用哪里呢？

  - 第5章中，讲到说0:200～0:300是相对安全的，这里只有256个字节，若需要的空间超过256个字节该怎么办呢？

引言

- 对于使用多个段的问题：

  数据、代码、栈

  - 在一个段中
  - 放入不同的段中？

6.1 在代码段中使用数据

- 考虑这样一个问题，编程计算以下8个数据的和，结果存在ax 寄存器中：

  0123H，0456H，0789H，0abcH，0defH，0fedH，0cbaH，0987H。

- 思路，程序开始是已经初始化数据到内存中，然后循环访存读数据进行累加。

6.1 在代码段中使用数据

- 程序6.1

```
assume cs:codesg

codesg segment
        dw 0123h,0456h,0789h,0abch,0defh,0fedh,0cbah,0987h
        mov bx,0
        mov ax,0
        mov cx,8
 s:     add ax,cs:[bx]
        add bx,2
        loop s

        mov ax,4c00h
        int 21h
codesg ends

end
```

dw（define word）：定义字型数据，数据间以逗号分隔。

可这8个数据在哪里呢？

段地址：在代码段中，可CS中得到。
偏移地址：分别为0, 2, 4, 6, 8, A, C, E。

它们的地址就是CS:0、CS:2、CS:4、CS:6、CS:8、CS:A、CS:C、CS:E。

6.1 在代码段中使用数据

- 程序6.1

```
assume cs:codesg

codesg segment
        dw 0123h,0456h,0789h,0abch,0defh,0fedh,0cbah,0987h
        mov bx,0
        mov ax,0
        mov cx,8
 s:     add ax,cs:[bx]
        add bx,2
        loop s

        mov ax,4c00h
        int 21h
codesg ends

end
```

bx存放偏移地址，在循环中对bx加2递增指向要访问的数据，并进行累加。

（1）在循环开始前，设置(bx)=0，cs:bx指向第一个数据所在的字单元。

（2）每次循环中(bx)=(bx)+2，cs:bx指向下一个数据所在的字单元。

6.1 在代码段中使用数据

- 将程序6.1编译、连接为可执行文件p61.exe，用debug加载查看一下。

```
C:\masm>debug p61.exe
-r
AX=0000  BX=0000  CX=0026  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0B2D  ES=0B2D  SS=0B3D  CS=0B3D  IP=0000   NV UP EI PL NZ NA PO NC
0B3D:0000 2301          AND     AX,[BX+DI]                    DS:0000=20CD
-u
0B3D:0000 2301          AND     AX,[BX+DI]
0B3D:0002 56            PUSH    SI
0B3D:0003 0489          ADD     AL,89
0B3D:0005 07            POP     ES
0B3D:0006 BC0AEF        MOV     SP,EF0A
0B3D:0009 0DED0F        OR      AX,0FED
0B3D:000C BA0C87        MOV     DX,870C
0B3D:000F 09BB0000      OR      [BP+DI+0000],DI
```

这时不能在系统中直接T命令运行，为什么？

——因为程序的入口处不是我们的指令。

6.1 在代码段中使用数据

- 将程序6.1编译、连接为可执行文件p61.exe，用debug加载查看一下。

```
C:\masm>debug p61.exe
-r
AX=0000  BX=0000  CX=0026  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0B2D  ES=0B2D  SS=0B3D  CS=0B3D  IP=0000   NV UP EI PL NZ NA PO NC
0B3D:0000 2301          AND     AX,[BX+DI]                    DS:0000=20CD
-u
0B3D:0000 2301          AND     AX,[BX+DI]
0B3D:0002 56            PUSH    SI
0B3D:0003 0489          ADD     AL,89
0B3D:0005 07            POP     ES
0B3D:0006 BC0AEF        MOV     SP,EF0A
0B3D:0009 0DED0F        OR      AX,0FED
0B3D:000C BA0C87        MOV     DX,870C
0B3D:000F 09BB0000      OR      [BP+DI+0000],DI
```

在Debug中将 IP 设为10h然后再用T（P、G） 命令执行。why?

（提示：dw写的数据占用了多少内存字节？）

6.1 在代码段中使用数据

- 如何让这个程序在编译后可以存系统中直接运行呢？

可以在源程序中指明程序入口，如程序6.2所示。

6.1 在代码段中使用数据

- 程序2

```
assume cs:codesg
codesg segment
        dw 0123h,0456h,0789h,0abch,0defh,0fedh,0cbah,0987h
 start: mov bx,0
        mov ax,0
        mov cx,8
      s: add ax,cs:[bx]
        add bx,2
        loop s

        mov ax,4c00h
        int 21h
codesg ends
end start
```

加上标号start ：
1. 在程序第一条指令前，
2. 伪指令end的后面出现。

伪指令end的作用：
通知编译器汇编程序的
(1) 结束位置
(2) 入口位置。

6.1 在代码段中使用数据

- 可安排程序框架如下：

```
assume cs:code
code segment
        :
      数据
        :
start:
        :
        :
      代码
        :
        :
code ends
end start
```

6.2 在代码段中使用栈

- 完成下面的程序，利用栈，将程序中定义的数据在原内存空间逆序存放。

```
assume cs:codesg
 codesg segment
   dw 0123h,0456h,0789h,0abch,0defh,0fedh,0cbah,0987h

   ?

 code ends
 end
```

- 程序大致思路

6.2 在代码段中使用栈

- 程序的思路大致如下：
  - 数据存放在cs:0~cs:15这8个字单元。依次将这些字单元入栈，然后再依次出栈到这些字单元中，即可逆序存放。
  - 存在问题：首先要有一段可当作栈的内存空间。如前所述，这段空间应该由系统来分配。可在程序中通过定义数据来取得一段空间，然后将这段空间当作栈空间来用。

6.2 在代码段中使用栈

- 程序6.3源码

```
assume cs:codesg

codesg segment

    dw 0123h,0456h,0789h,0abch,0defh,0fedh,0cbah,0987h

    dw 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                        ;用 dw 定义 16 个字型数据，在程序加载后，将取得 16 个字的
                        ;内存空间，存放这 16 个数据。在后面的程序中将这段
                        ;空间当作栈来使用

start:  mov ax,cs
        mov ss,ax
        mov sp,30h      ;将设置栈顶 ss:sp 指向 cs:30

        mov bx,0
        mov cx,8
    s:  push cs:[bx]
        add bx,2
        loop s          ;以上将代码段 0~15 单元中的 8 个字型数据依次入栈

        mov bx,0
        mov cx,8
    s0: pop cs:[bx]
        add bx,2
        loop s0         ;以上依次出栈 8 个字型数据到代码段 0~15 单元中

        mov ax,4c00h
        int 21h

codesg ends

end start               ;指明程序的入口在 start 处
```

特别提示

(1) 下面的程序实现依次用内存 0:0~0:15 单元中的内容改写程序中的数据，完成程序：

```
assume cs:codesg

codesg segment

    dw 0123h,0456h,0789h,0abch,0defh,0fedh,0cbah,0987h

start:  mov ax,0
        mov ds,ax
        mov bx,0

        mov cx,8
    s:  mov ax,[bx]

        ____________

        add bx,2
        loop s

        mov ax,4c00h
        int 21h

codesg ends

end start
```

检测点6.1（Page 129）

没有通过检测点，请不要向下学习！

6.3 将数据、代码、栈放入不同的段

- 将数据、栈和代码放于一个段中，存在问题：

  （1）程序混乱，不便于设计维护程序；

  （2）大程序中数据、栈和代码需要的空间可能超过段大小（8086CPU最大64KB）。

- 可用多个段来存放数据、代码和栈。

6.3 将数据、代码、栈放入不同的段

- （1）怎么将数据、代码和栈放入不同的段呢？

类似于定义代码段，可定义多个段，如程序6.4（和程序6.3 功能相同）所示。

6.3 将数据、代码、栈放入不同的段

- 程序6.4

程序6.4

```
assume cs:code,ds:data,ss:stack

data segment
   dw 0123h,0456h,0789h,0abch,0defh,0fedh,0cbah,0987h
data ends

stack segment
   dw 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
stack ends

code segment
start:  mov ax,stack
        mov ss,ax
        mov sp,20h    ;设置栈顶 ss:sp 指向 stack:20
code ends
end start
```

数据段

栈段

？

设置SS：SP
运行到这里CPU在知道栈的位置

stack在编译时为常数，不能

mov ss,data ✗

6.3 将数据、代码、栈放入不同的段

- 程序6.4（续）

```
        mov ax,data
        mov ds,ax       ;ds 指向 data 段

        mov bx,0        ;ds:bx 指向 data 段中的第一个单元

        mov cx,8
    s:  push [bx]
        add bx,2
        loop s          ;以上将 data 段中的 0~15 单元中的 8 个字型数据依次入栈

        mov bx,0

        mov cx,8
    s0: pop [bx]
        add bx,2
        loop s0         ;以上依次出栈 8 个字型数据到 data 段的 0~15 单元中

        mov ax,4c00h
        int 21h

code ends

end start
```

设置DS
运行到这里，程序才知道数据段位置

设置CS:IP（编译器将入口start的地址写入exe中，加载exe时根据被写入的地址设置CS：IP）

6.3 将数据、代码、栈放入不同的段

- （2）段地址的应用

程序中“data”段中的数据“0cbah”的地址为：data:6。如何将此数据送入bx中？

正确汇编指令：

```
mov ax,data
mov ds,ax
mov bx,ds:[6]
```

错误汇编指令：

```
mov ds,data

mov bx,ds:[6]
```

将data在编译器中相当于常数，8086CPU不允许mov ds, 常数

6.3 将数据、代码、栈放入不同的段

- （3）“代码段”、“数据段”、“栈段”完全是我们的安排

  - -1-. 定义了code、data、stack段，CPU知道了“代码段”、“数据段”、“栈段”在哪里？

    答： No.

6.3 将数据、代码、栈放入不同的段

- -2- 伪指令“assume cs:code,ds:data,ss:stack”将cs、ds和ss分别和code、data、stack段相连。这样做了之后，CPU是否就会将 cs指向 code，ds 指向 data，ss 指向stack呢？

答： No. assume 是由编译器执行的伪指令，CPU并不知道它们。

。

assume 的作用:

•It tell the assembler which segment register to use to access a variable.
•It tell the assembler with respect to which segment to calculate the offsets.
•The segment ASSUME ed by CS is the segment the code labels belong to.

6.3 将数据、代码、栈放入不同的段

- -3- 若要CPU如何知道代码段在哪里？

答：“end start”说明了程序的入口在标识为start的内存。

.exe文件描述信息中有入口信息（第一条指令位置），.exe被加载入内存后，CPU的CS:IP被设置指向这个入口，从而开始执行程序中的第一条指令。

6.3 将数据、代码、栈放入不同的段

- 总之，程序员在汇编程序中明确指定CS:IP、SS:SP、DS等寄存器设置，之后CPU才知道代码段、栈段、数据段的内存位置。