

6.7 设有 10 个学生的成绩分别是 76, 69, 84, 90, 73, 88, 99, 63, 100 和 80 分。试编制一个子程序统计 60~69 分, 70~79 分, 80~89 分, 90~99 分和 100 分的人数并分别存放到 S6, S7, S8, S9 和 S10 单元中。

源代码如下:

数据段:

```
DATAS SEGMENT
;此处输入数据段代码
score db 76,69,84,90,73,88,99,63,100,80
S6 db 0
S7 db 0
S8 db 0
S9 db 0
S10 db 0
string db 'S6 S7 S8 S9 S10',0ah,0dh,'$';$终止符
space db ' ','$'
DATAS ENDS
```

堆栈段:

```
STACKS SEGMENT
db 40 dup(0) ;此处输入堆栈段代码
STACKS ENDS
```

代码段:

```
CODES SEGMENT
ASSUME CS:CODES,DS:DATAS,SS:STACKS
START:
MOV AX,DATAS
MOV DS,AX

mov si,offset score
mov cx,10
loop_:
mov al,[si]
call choice
inc si
loop loop_

mov dx,offset string;打印结果提示
mov ah,9
int 21h

mov cx,5
mov si,offset S6
```

```
mov di,offset S6
loop_check:
mov dx,[si]
add dx,30H
mov ah,02H
int 21h
inc si
mov dx,offset space
mov ah,09h
int 21h
loop loop_check

MOV AH,4CH
INT 21H
```

子程序:

```
choice proc near
    push ax
    push bx
    push cx
    push dx
    cmp al,60
    jl return
    ;jb return
    cmp al,100
    je isS10
    cmp al,69
    jbe isS6
    cmp al,79
    jbe isS7
    cmp al,89
    jbe isS8
    cmp al,99
    jbe isS9

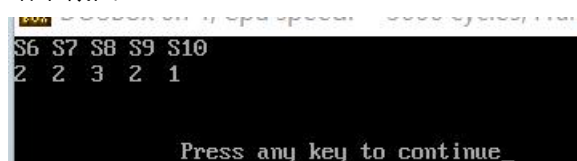
isS10:
    inc S10
    jmp return
isS6:
    inc S6
    jmp return
isS7:
    inc S7
    jmp return
isS8:
    inc S8
    jmp return
isS9:
    inc S9

return:
    pop dx
    pop cx
    pop bx
    pop ax
    ret
choice endp

;此处输入代码段代码

CODES ENDS
END START
```

结果截图:



```
C:\DOS\COMMAND.COM [4,000 bytes] - C:\DOS\SYSTEM\...
S6 S7 S8 S9 S10
2 2 3 2 1

Press any key to continue_
```

6.15 试编写一个执行以下计算的子程序 COMPUTE:

$$R \leftarrow -X + Y - 3$$

其中 X, Y 及 R 均为字数组。假设 COMPUTE 与其调用程序都在同一代码段中, 数据段 D_SEG 中包含 X 和 Y 数组, 数据段 E_SEG 中包含 R 数组, 同时写出主程序调用 COMPUTE 过程的部分。

如果主程序和 COMPUTE 在同一程序模块中, 但不在同一代码段中, 程序应如何修改?

如果主程序和 COMPUTE 不在同一程序模块, 程序应如何修改?

1. 主程序和 COMPUTE 在同一程序模块中, 在同一代码段中时代码如下:

定义数据段和堆栈段

```
D_SEG SEGMENT
    count equ 10H
    X DW count DUP(?)
    Y DW count DUP(?) ;此处输入数据段代码
D_SEG ENDS

E_SEG SEGMENT
    R DW count DUP(?) ;此处输入堆栈段代码
E_SEG ENDS

STACKS SEGMENT
    db 40 dup(0) ;此处输入堆栈段代码
STACKS ENDS
```

代码段:

```
C_SEG SEGMENT
MAIN proc far
    ASSUME CS:C_SEG,DS:D_SEG,SS:STACKS
START:
    MOV AX,DATAS
    MOV DS,AX
    ;此处输入代码段代码
    mov cx,count
    call COMPUTE
    RET
MAIN endp

COMPUTE proc near
    push ax
    push bx
    push cx
    push dx
    mov bx,0
loop_:
    mov ax,X[bx]
    add ax,Y[bx]
    sub ax,3
    mov ES:R[bx],ax
    add,bx,2
    loop loop_
    pop dx
    pop cx
    pop bx
    pop ax
    ret
COMPUTE endp

C_SEG ENDS
    END START
```

2. 主程序和 COMPUTE 在同一程序模块中，但不在同一代码段中时。

数据段和堆栈段不用更改。

代码段分为两段

```
C_SEG SEGMENT
MAIN proc far
    ASSUME CS:C_SEG,DS:D_SEG,SS:STACKS
START:
    MOV AX,DATAS
    MOV DS,AX
    ;此处输入代码段代码
    mov cx,count
    call FAR PTR COMPUTE
    RET
MAIN endp
C_SEG ENDS
```

主要更改处: `call FAR PTR COMPUTE` 段外调用

```
CN_SEG SEGMENT
COMPUTE proc far
    ASSUME CS:CN_SEG
    push ax
    push bx
    push cx
    push dx
    mov bx,0
loop_:
    mov ax,X[bx]
    add ax,Y[bx]
    sub ax,3
    mov ES:R[bx],ax
    add,bx,2
    loop loop_
    pop dx
    pop cx
    pop bx
    pop ax
    ret
COMPUTE endp
CN_SEG ENDS

END START
```

3. 主程序和 COMPUTE 不在同一程序模块时代码主要更改：
主程序在 2 的基础上增加扩展与引入

```
TITLE MAIN ;  
EXTRN COMPUTE:FAR  
PUBLIC COUNT,X,Y,R
```

COMPUTE 在 2 的基础上增加拓展与引入

```
TITLE COMPUTE  
EXTRN COUNT:WORD,X:WORD,Y:WORD,R:WORD  
PUBLIC COMPUTE
```

在将原子程序和主程序分成两个程序模块即可。