实验 6　循环分支结构程序设计

一、 实验目的

熟悉程序框架；理解单重循环与多重循环程序以及分支的结构及执行过程；掌握循环控制指令的用法；熟练掌握循环结构程序的设计方法和利用设置断点调试程序的方法。

二、 示例

1、内存自 BUF 单元开始的缓冲区(数组,变量)连续存放着 10 个学生的英文分数，编程序统计其中 90～100，60～89，60 以下者各有多少人?并把 90-100 分人数结果存放在 P1 单元中、60－89 分人数结果存放在 P2 单元中，60 分以下人数结果存放在 P3 单元中（P1、P2、P3 为数据段的单元）

```
DATA    SEGMENT
   BUF    DB(字节型)          70H,58H,50H,91H,99H,62H,75H,82H,74H,60H(没加 H 的话会转化为对应的 16 进制,到时候不好观察,所以我们使用 H 表示)
ORG    0020H (定位伪指令,指定后面的变量存放的起始偏移地址,教材第四章,伪指令,没有的话就是偏移地址就是 A,B,C)
P1    DB    0 (或者设置为?,用于存放人数)
ORG    0030H
   P2    DB    0
ORG    0040H
   P3    DB    0
ORG    0050H
COUNT      DW         10
DATA    ENDS
;设置内存里的数据段
CSEG    SEGMENT
        ASSUME CS:CSEG,DS:DATA

START:MOV       AX,DATA
        MOV       DS,AX
        MOV       CX,COUNT(也可以直接赋为 10,但是在数据段中设置更加通用,也可以使用$-BUF 更加通用)存储器中的直接寻址方式
        MOV       SI,0
LP1:
        MOV    AL,BUF[SI](相对寻址方式,BUF 可以理解为对应的初始地址)
        CMP    AL,90H (第三章减法指令中的比较指令,两个数做减法,不存储结果当时影响标志位)
        JNB    LP2(条件转移指令,无符号数的比较,不低于>=就执行)
        CMP      AL,60H
        JC       LP3(<就转移,>=就执行)
        INC      P2
        JMP      LP5
```

```
LP3:   INC     P3
       JMP     LP5
LP2:   INC     P1(将 p1+1 操作)

LP5:   INC     SI(修改地址指针)
       LOOP    LP1(count 计数器-1 隐含在这段代码中,不要自己去减)

       MOV     AH,4CH
       INT     21H
CSEG   ENDS
       END     START
```

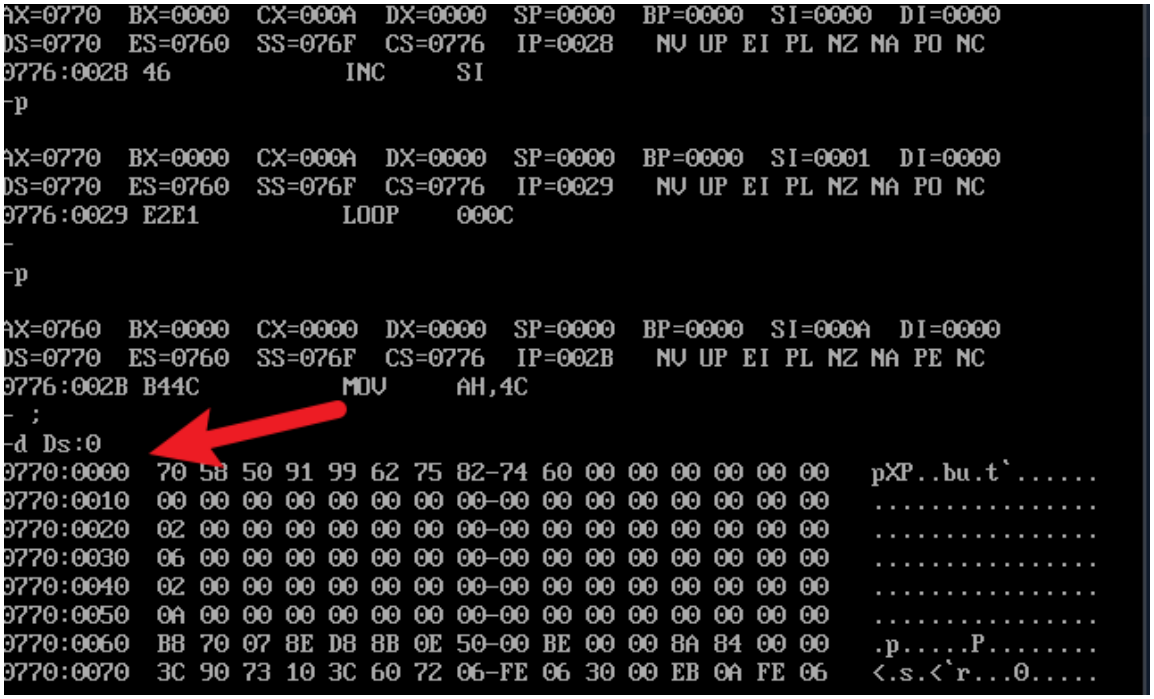
- 使用-u 命令查看程序是否正确加载

```
(pt2) D:\code\EXPERIMENTAL_REPORT\汇编语言\E6_循环分支结构程序设计\c1>debug c1.exe
-u
2053:0000 B84D20        MOV     AX,204D
2053:0003 8ED8          MOV     DS,AX
2053:0005 8B0E5000      MOV     CX,[0050]
2053:0009 BE0000        MOV     SI,0000
2053:000C 8A840000      MOV     AL,[SI+0000]
2053:0010 3C90          CMP     AL,90
2053:0012 7312          JNB     0026
2053:0014 3C60          CMP     AL,60
2053:0016 7207          JB      001F
2053:0018 FE063000      INC     BYTE PTR [0030]
2053:001C EB0C          JMP     002A
2053:001E 90            NOP
2053:001F FE064000      INC     BYTE PTR [0040]
```

- 对应成绩 0050 对应 count 的值

```
-d 204D:0
204D:0000  70 58 50 91 99 62 75 82-74 60 00 00 00 00 00 00   pXP..bu.t`......
204D:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
204D:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
204D:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
204D:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
204D:0050  0A 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
204D:0060  B8 4D 20 8E D8 8B 0E 50-00 BE 00 00 8A 84 00 00   8M .X..P.>......
204D:0070  3C 90 73 12 3C 60 72 07-FE 06 30 00 EB 0C 90 FE   <.s.<`r.~.0.k..~
```

- T 命令和 p 命令之间的区别，遇到循环体执行完



人数分别为 2,6,2

三

1. **阅读下面程序**

```
data segment
    db '1. display.....'
data ends
code segment
assume cs:code,ds:data
start:    mov ax,data
          mov ds,ax
          mov bx,0
          mov cx,4
          mov si,0
   s:     mov al,[bx+si+3]
          and al,11011111b      ；作用（ 将字符转换为大写字母（清除第 5 位）
                  ）
          mov [bx+si+3],al
          inc si
          loop s

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

感觉是从内存里取一个数放 AL，和 11011111B 做与操作，和 1 相与是 0 为 0 是 1

为 1，和 0 与都为 0，将第 3 为清零

- -u 命令查看程序是否加载正确

```
(pt2) D:\code\EXPERIMENTAL_REPORT\汇编语言\E6_循环分支结构程序设计\c2>debug c2.exe
-u
204E:0000 B84D20        MOV     AX,204D
204E:0003 8ED8          MOV     DS,AX
204E:0005 BB0000        MOV     BX,0000
204E:0008 B90400        MOV     CX,0004
204E:000B BE0000        MOV     SI,0000
204E:000E 8A4003        MOV     AL,[BX+SI+03]
204E:0011 24DF          AND     AL,DF
204E:0013 884003        MOV     [BX+SI+03],AL
204E:0016 46            INC     SI
204E:0017 E2F5          LOOP    000E
204E:0019 B8004C        MOV     AX,4C00
204E:001C CD21          INT     21
204E:001E 0000          ADD     [BX+SI],AL
```

- 与操作执行前(全为小写)

```
-D dS:0
204D:0000  31 2E 20 64 69 73 70 6C-61 79 2E 2E 2E 2E 2E 2E   1. display......
204D:0010  B8 4D 20 8E D8 BB 00 00-B9 04 00 BE 00 00 8A 40   8M .X;..9..>...@
204D:0020  03 24 DF 88 40 03 46 E2-F5 B8 00 4C CD 21 00 00   .$_.@.Fbu8.LM!..
204D:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
204D:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
204D:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
204D:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
204D:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

- 运行结果

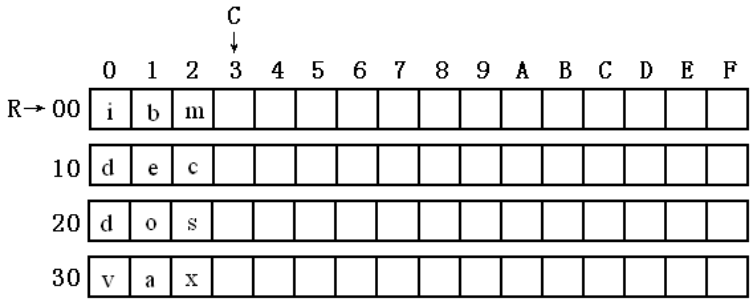
```
-g

Program terminated normally
-d dS:0
204D:0000  31 2E 20 44 49 53 50 6C-61 79 2E 2E 2E 2E 2E 2E   1. DISPlay......
204D:0010  B8 4D 20 8E D8 BB 00 00-B9 04 00 BE 00 00 8A 40   8M .X;..9..>...@
204D:0020  03 24 DF 88 40 03 46 E2-F5 B8 00 4C CD 21 00 00   .$_.@.Fbu8.LM!..
204D:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
204D:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
204D:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
204D:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
204D:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

解释程序功能

将数据段中存储的字符串 `'1. display.....'` 的第 4 到第 7 个字符（即 `d`, `i`, `s`, `p`）依次转换为大写字母。程序通过循环依次读取这 4 个字符，将它们的 ASCII 码的第 5 位清零（从而实现小写转大写），再写回原字符串对应位置，最后程序返回 DOS。这样，原字符串中的 `display` 会变成 `DISPlay`，其余部分保持不变。

2 执行下面程序。




```
data segment
    db 'ibm                '
    db 'dec                '
    db 'dos                '
    db 'vax                '
data ends
code segment
assume cs:code,ds:data
start: mov ax,data
       mov ds,ax
       mov bx,0
       mov cx,4
 s0:
       mov si,0
       mov cx,3
 s:    mov al,[bx+si]
       and al,11011111b
       mov [bx+si],al
       inc si
       loop s
       add bx,16

       loop s0
       mov ax,4c00h
       int 21h
code ends
end start
```

双重循环,先行再列,有功能上的错误

➢ 修改后的代码

```
data segment
   db 'ibm              '    ; 定义数据段，存储 4 个字符串，每个占 16 字节
   db 'dec              '
   db 'dos              '
   db 'vax              '
data ends
```

```
code segment
assume cs:code,ds:data      ; 设置代码段和数据段
start: mov ax,data          ; 初始化数据段
       mov ds,ax
       mov bx,0             ; bx 为基址
       mov cx,4             ; 外层循环4 次（4 个字符串）
s0:    mov si,0             ; si 为偏移量
       mov dx,3             ; dx 为内层循环计数器（每个字符串前3 个字符）
s:     mov al,[bx+si]       ; 取字符串中的字符到al
       and al,11011111b     ; 将字符转换为大写字母（清除第5 位）
       mov [bx+si],al       ; 将转换后的字符存回原位置
       inc si               ; 偏移量加1
       dec dx               ; 内层循环计数器减1
       jnz s                ; dx 不为0 则继续内层循环
       add bx,16            ; 基址加16，指向下一个字符串
       loop s0              ; 外层循环
       mov ax,4c00h         ; 程序返回DOS
       int 21h
code ends
end start
```

➢ 使用-u 命令查看程序加载是否正确

```
(pt2) D:\code\EXPERIMENTAL_REPORT>debug 汇编语言\E6_循环分支结构程序设计\c3\c3.exe
-u
2051:0000 B84D20        MOV     AX,204D
2051:0003 8ED8          MOV     DS,AX
2051:0005 BB0000        MOV     BX,0000
2051:0008 B90400        MOV     CX,0004
2051:000B BE0000        MOV     SI,0000
2051:000E B90300        MOV     CX,0003
2051:0011 8A00          MOV     AL,[BX+SI]
2051:0013 24DF          AND     AL,DF
2051:0015 8800          MOV     [BX+SI],AL
2051:0017 46            INC     SI
2051:0018 E2F7          LOOP    0011
2051:001A 83C310        ADD     BX,+10
2051:001D E2EC          LOOP    000B
2051:001F B8004C        MOV     AX,4C00
```

➢ 运行前数据

```
AX=204D  BX=0000  CX=0004  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=204D  ES=203D  SS=204D  CS=2051  IP=000E   NV UP DI PL NZ NA PO NC
2051:000E BA0300        MOV     DX,0003
-d ds:0
204D:0000  69 62 6D 20 20 20 20 20-20 20 20 20 20 20 20 20   ibm
204D:0010  64 65 63 20 20 20 20 20-20 20 20 20 20 20 20 20   dec
204D:0020  64 6F 73 20 20 20 20 20-20 20 20 20 20 20 20 20   dos
204D:0030  76 61 78 20 20 20 20 20-20 20 20 20 20 20 20 20   vax
204D:0040  B8 4D 20 8E D8 BB 00 00-B9 04 00 BE 00 00 BA 03   8M .X;..9..>..:.
204D:0050  00 8A 00 24 DF 88 00 46-4A 75 F6 83 C3 10 E2 EB   ...$_..FJuv.C.bk
204D:0060  B8 00 4C CD 21 00 00 00-00 00 00 00 00 00 00 00   8.LM!...........
204D:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

➢ 运行后数据

```
-g

Program terminated normally
-d ds:0
204D:0000  49 42 4D 20 20 20 20 20-20 20 20 20 20 20 20 20   IBM
204D:0010  44 45 43 20 20 20 20 20-20 20 20 20 20 20 20 20   DEC
204D:0020  44 4F 53 20 20 20 20 20-20 20 20 20 20 20 20 20   DOS
204D:0030  56 41 58 20 20 20 20 20-20 20 20 20 20 20 20 20   VAX
204D:0040  B8 4D 20 8E D8 BB 00 00-B9 04 00 BE 00 00 BA 03   8M .X;..9..>..:.
204D:0050  00 8A 00 24 DF 88 00 46-4A 75 F6 83 C3 10 E2 EB   ...$_..FJuv.C.bk
204D:0060  B8 00 4C CD 21 00 00 00-00 00 00 00 00 00 00 00   8.LM!...........
204D:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

➢ 上述程序为什么发生错误

内外层循环都用 cx，导致外层循环计数被覆盖，实际只处理第一个字符串。

在保留源程序指令的基础上，如何修改程序，并给出最终程序实现的功能。

如何修改程序:内层循环用 dx 作为计数器，外层循环用 cx，互不干扰。

```
code segment
assume cs:code,ds:data      ; 设置代码段和数据段
start: mov ax,data          ; 初始化数据段
       mov ds,ax
       mov bx,0             ; bx 为基址
       mov cx,4             ; 外层循环 4 次（4 个字符串）
s0:    mov si,0             ; si 为偏移量
       mov dx,3             ; dx 为内层循环计数器（每个字符串前 3 个字符）
s:     mov al,[bx+si]       ; 取字符串中的字符到 al
       and al,11011111b     ; 将字符转换为大写字母（清除第 5 位）
       mov [bx+si],al       ; 将转换后的字符存回原位置
       inc si               ; 偏移量加 1
       dec dx               ; 内层循环计数器减 1
       jnz s                ; dx 不为 0 则继续内层循环
       add bx,16            ; 基址加 16，指向下一个字符串
       loop s0              ; 外层循环
       mov ax,4c00h         ; 程序返回 DOS
       int 21h
code ends
```

3 设 X，Y，Z 分别放着三个 16 位数，如果三个数都不是 0，则求出三个数之和，并存放在 W 单元，如果其中一个数为 0，则把其他两个单元也清零，请实现该程序。

相关代码(这里只提供了 x,y,z 为 4,1,0 的代码,其他情况可以设置数据段的值即可)

```
DATA  SEGMENT
  X  DW  4    ; 定义变量 x，初始值为 4
  Y  DW  1    ; 定义变量 y，初始值为 1
  Z  DW  0    ; 定义变量 z，初始值为 0
  W  DW  0    ; 定义变量 w，初始值为 0
```

```
DATA  ENDS

CODE  SEGMENT
  ASSUME CS:CODE,DS:DATA    ; 设置代码段和数据段
START:
  MOV  AX,DATA              ; 初始化数据段
  MOV  DS,AX
  MOV  AX,X                 ; 将x的值送入ax
  CMP  AX,0                 ; 比较x是否为0
  JZ   HASZERO              ; 如果x为0，跳转到HASZERO
  MOV  AX,Y                 ; 将y的值送入ax
  CMP  AX,0                 ; 比较y是否为0
  JZ   HASZERO              ; 如果y为0，跳转到HASZERO
  MOV  AX,Z                 ; 将z的值送入ax
  CMP  AX,0                 ; 比较z是否为0
  JZ   HASZERO              ; 如果z为0，跳转到HASZERO
  MOV  AX,W                 ; 将w的值送入ax
  ADD  AX,Z                 ; w = w + z
  ADD  AX,Y                 ; w = w + y
  ADD  AX,X                 ; w = w + x
  MOV  W,AX                 ; 将计算结果存回w
  JMP  TOEND                  ; 跳转到程序结束
HASZERO:
  MOV  AX,0                 ; 将0送入ax
  MOV  X,AX                 ; x = 0
  MOV  Y,AX                 ; y = 0
  MOV  Z,AX                 ; z = 0
TOEND:
  MOV  AX,4C00H             ; 程序返回DOS
  INT  21H
CODE  ENDS
  END  START
```

➢ 使用-u 命令查看程序是否成功加载

```
(pt2) D:\code\EXPERIMENTAL_REPORT\汇编语言\E6_循环分支结构程序设计\c4>debug c4.exe
-u
204E:0000 B84D20        MOV     AX,204D
204E:0003 8ED8          MOV     DS,AX
204E:0005 A10000        MOV     AX,[0000]
204E:0008 3D0000        CMP     AX,0000
204E:000B 7425          JZ      0032
204E:000D A10200        MOV     AX,[0002]
204E:0010 3D0000        CMP     AX,0000
204E:0013 741D          JZ      0032
204E:0015 A10400        MOV     AX,[0004]
204E:0018 3D0000        CMP     AX,0000
204E:001B 7415          JZ      0032
204E:001D A10600        MOV     AX,[0006]
```

➢ 情况 1:初始数据 4, 1, 0

```
AX=0004  BX=0000  CX=0053  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=204D  ES=203D  SS=204D  CS=204E  IP=0008   NV UP DI PL NZ NA PO NC
204E:0008 3D0000        CMP     AX,0000
-d ds:0
204D:0000  04 00 01 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
204D:0010  B8 4D 20 8E D8 A1 00 00-3D 00 00 74 25 A1 02 00   8M .X!..=..t%!..
204D:0020  3D 00 00 74 1D A1 04 00-3D 00 00 74 15 A1 06 00   =..t.!..=..t.!..
204D:0030  03 06 04 00 03 06 02 00-03 06 00 00 A3 06 00 EB   ............#..k
204D:0040  0D 90 B8 00 00 A3 00 00-A3 02 00 A3 04 00 B8 00   ..8..#..#..#..8.
204D:0050  4C CD 21 00 00 00 00 00-00 00 00 00 00 00 00 00   LM!.............
204D:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
204D:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-
```

➢ 执行结果(有一个含 0,全为 0)

```
-g

Program terminated normally
-d ds:0
204D:0000  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
204D:0010  B8 4D 20 8E D8 A1 00 00-3D 00 00 74 25 A1 02 00   8M .X!..=..t%!..
204D:0020  3D 00 00 74 1D A1 04 00-3D 00 00 74 15 A1 06 00   =..t.!..=..t.!..
204D:0030  03 06 04 00 03 06 02 00-03 06 00 00 A3 06 00 EB   ............#..k
204D:0040  0D 90 B8 00 00 A3 00 00-A3 02 00 A3 04 00 B8 00   ..8..#..#..#..8.
204D:0050  4C CD 21 00 00 00 00 00-00 00 00 00 00 00 00 00   LM!.............
204D:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
204D:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

➢ 情况 2:初始数据 6, 6, 6

```
-d ds:0
204D:0000  06 00 06 00 06 00 00 00-00 00 00 00 00 00 00 00   ................
204D:0010  B8 4D 20 8E D8 A1 00 00-3D 00 00 74 25 A1 02 00   8M .X!..=..t%!..
204D:0020  3D 00 00 74 1D A1 04 00-3D 00 00 74 15 A1 06 00   =..t.!..=..t.!..
204D:0030  03 06 04 00 03 06 02 00-03 06 00 00 A3 06 00 EB   ............#..k
204D:0040  0D 90 B8 00 00 A3 00 00-A3 02 00 A3 04 00 B8 00   ..8..#..#..#..8.
204D:0050  4C CD 21 00 00 00 00 00-00 00 00 00 00 00 00 00   LM!.............
204D:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
204D:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

➢ 执行结果(w=(12)H=(18)D)

```
-g

Program terminated normally
-d ds:0
204D:0000  06 00 06 00 06 00 12 00-00 00 00 00 00 00 00 00   ................
204D:0010  B8 4D 20 8E D8 A1 00 00-3D 00 00 74 25 A1 02 00   8M .X!..=..t%!..
204D:0020  3D 00 00 74 1D A1 04 00-3D 00 00 74 15 A1 06 00   =..t.!..=..t.!..
204D:0030  03 06 04 00 03 06 02 00-03 06 00 00 A3 06 00 EB   ............#..k
204D:0040  0D 90 B8 00 00 A3 00 00-A3 02 00 A3 04 00 B8 00   ..8..#..#..#..8.
204D:0050  4C CD 21 00 00 00 00 00-00 00 00 00 00 00 00 00   LM!.............
204D:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
204D:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```