第五章参考答案

5-1

(1) STAD DB 78,-40,0D6H,49H

(2)ARRAY DB 45H, 12H,64H,00H,0D2H,04H,0C7H,00H

(3)ALPHA DB 12H,0FCH,0E4H,65H

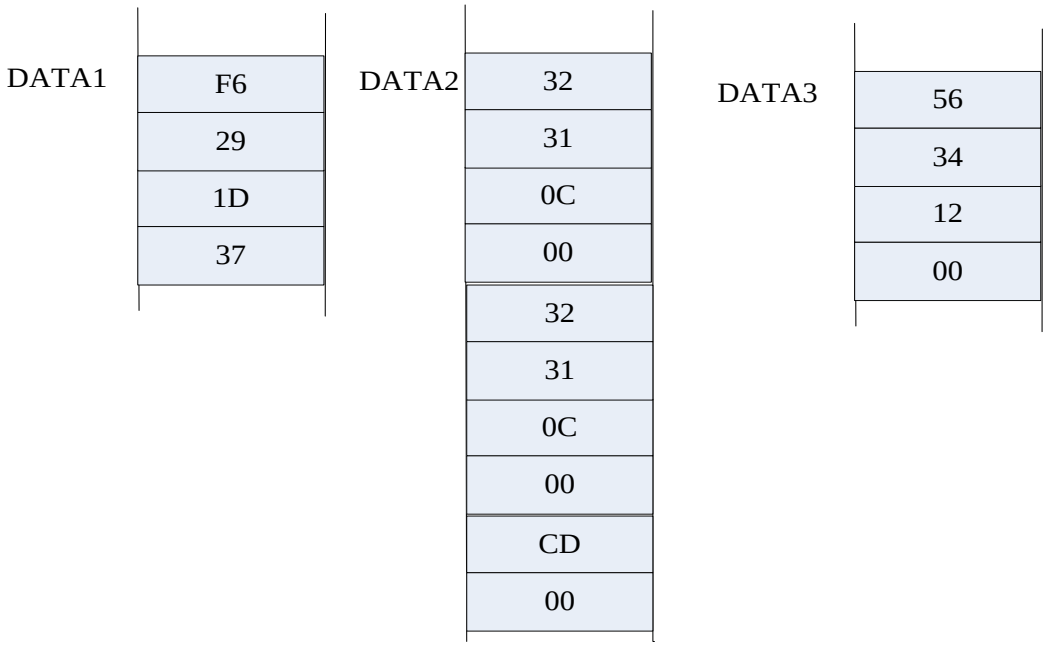
(4)BETA DB 4 DUP(8),6 DUP('S'),20 DUP(' '),10 DUP(1,3)

(5)STRING DB 'THIS IS A EXAMPE'

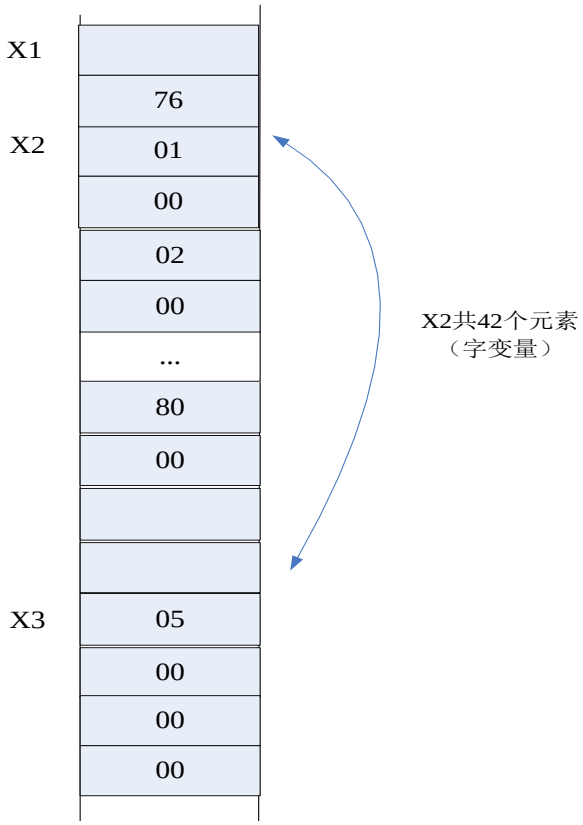
(6) TOTAL EQU 780

TOTAL= 780

5-2

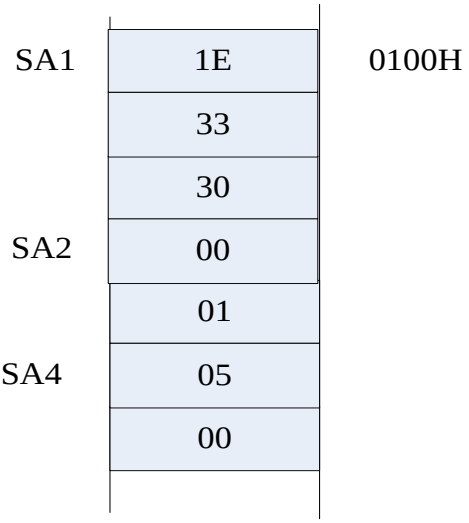


5-3



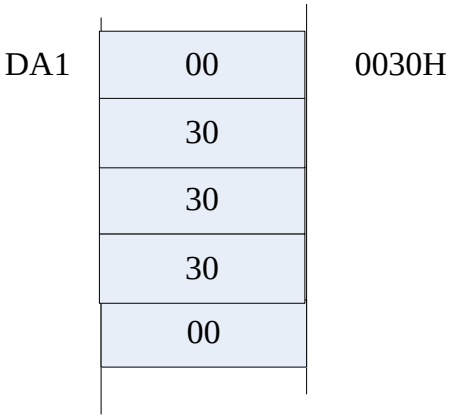
COUNT 的值为 42*2+4　　表示变量 X1 和 X 共占用的字节存储单元数

5-4



注：SA3 不占用内存

5-5



3000H 为字数据的偏移地址为 0030H

5-6

(1) MOV BX ,OFFSET BUF1　或 LEA　BX, BUF1

(2) MOV CL ,BYTE PTR BUF2+3

(3) MOV BUF3+7,0C6H

(4)CNT EQU BUF4-BUF2

(5)MOV AX,BUF2+2

ADD AL,BUF4+3(当成无符号数)

ADC AH,0

MOV　WORD PTR BUF3,AX

5-7

(1)　7CH　(2)6752H (3)C0H (4)31H　(5)04H

5-8

[DA2]=0D5B3H　　CF=0

5-9

第二个 AND 为伪指令，在汇编时给出值。第一个 AND 为指令，可改写成

AND AX,06H

5-10

```
DATA　SEGMENT
BCD1　DB　31H,32H　；十进制数 1 和 2 的 ASCII
BCD2　DB　? ;最终的结果应为 21H
DATA　ENDS
CODE　SEGMENT
　　ASSUME　CS:CODE, DS:DATA
START:　MOV AX,DATA
　　MOV DS,AX
　　MOV AL,BCD1
　　SUB AL,30H
　　MOV AH,AL;低 4 位暂存 AH
　　MOV AL,BCD1+1
　　SUB AL,30H
　　MOV CL,4
　　SHL　AL,CL ;高 4 位的左移 4 位
```

```
        OR AL,AH
        MOV BCD2,AL
        MOV AH,4CH
        INT 21H
CODE    ENDS
        END START
```

5-11

采用左移指令

```
MOV BX,AX
SHR BX,1        ;乘 2
MOV DX,AX
MOV CL,2
SHR AX ,CL    ;乘 4
ADD AX,BX    ;3 倍
ADD AX,DX    ;7 倍
```

5-12

1E00H

5-13

(1) AL=00　(2)AL=01H　(3)AL=FFH

5-14

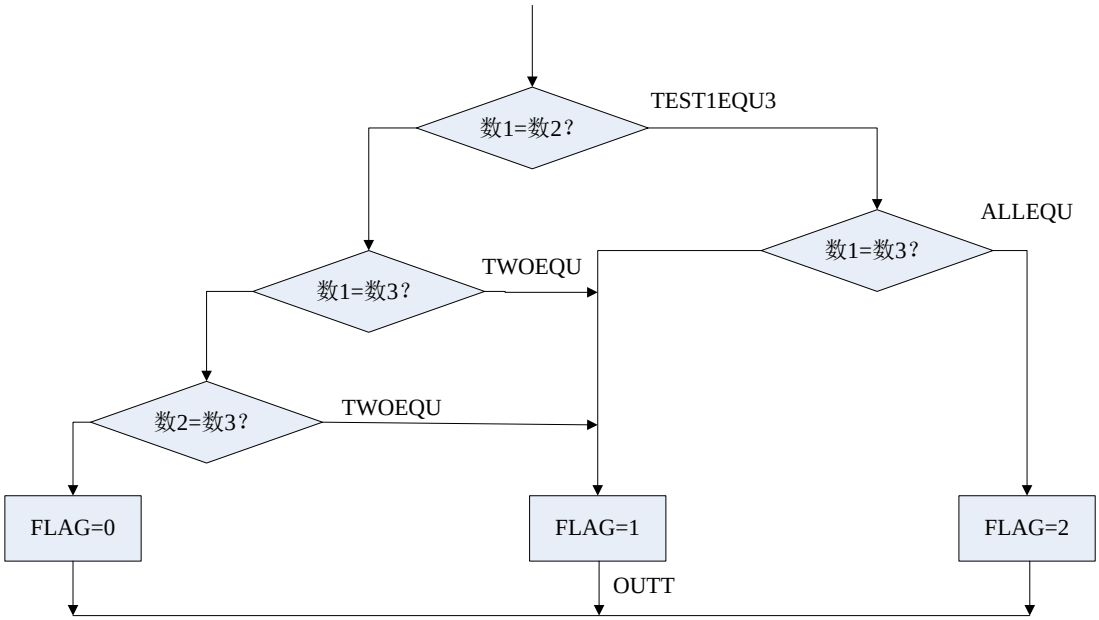
编程思路有两种方法：1.将给出的 BCD 码首先转换成二进制数（十六进制数），按二进制的算术运算，将最终结果再转换为 BCD 码。2.按照 BCD 码进行运算，其中 2*A 按压缩 BCD 的加法（A+A）。

此题用方法 2 比较简单

```
DATA    SEGMENT
BUFF    DB    34H    ; BCD 码数据为十进制 34
DES    DB    ?    ;最终的结果应为 21H
DATA    ENDS
CODE    SEGMENT
        ASSUME    CS:CODE, DS:DATA
START:    MOV AX, DATA
        MOV DS, AX
        MOV AL, BUFF
        CMP    AL,20H    ; 十进制 20H
        JB    ADDSELF
        CMP AL, 60H
        JB    SUB20H
        MOV DES, 80H
        JMP    NEXT

SUB20H:    MOV AL,BUFF
          SUB AL,20H
          DAS
          JMP OUTT
```

```
ADDSELF: MOV AL,BUFF
         ADD AL,BUFF     ;自加代替乘 2
         DAA
   OUTT: MOV DES,AL
    NEXT:MOV AH,4CH
         INT 21H
CODE     ENDS
         END START
```

5-15




```
DATA      SEGMENT
BUFER   DW 1200H,1200H,1200H
FLAG     DB   ?
DATA      ENDS
CODE      SEGMENT
          ASSUME   CS:CODE ,DS: DATA
START:    MOV    AX, DATA
          MOV    DS, AX
          MOV AX,BUFER
          CMP AX,BUFER+2        ;判断 1 和 2 是否相等
          JZ TEST1EQU3
          CMP AX,BUFER+4        ;判断 1 和 3 是否相等
          JZ   TWOEQU
          MOV AX,BUFER+2
          CMP   AX,BUFER+4
          JZ TWOEQU
          MOV FLAG, 0
          JMP OUTT
TEST1EQU3:CMP AX,BUFER+4
```

```
            JZ    ALLEQU
TWOEQU: MOV    FLAG,01H       ;两个数相等
            JMP    OUTT
ALLEQU: MOV FLAG,02H          ;三个数相等
   OUTT: MOV AH, 4CH
          INT 21H
CODE    ENDS
        END        START
```

5-16

(1) AX=55H CX=0

(2) 5,6,7,8,9

5-17

```
DATTA       SEGMENT
DATA         DW 5, 7, 1900h, 2300h, 0a0h, 000BH, …,- 1
NUM          EQU      $-BUF/2
MAX DW     ?
MIN    DW      ?
DATTA        ENDS
CODE         SEGMENT
             ASSUME DS:DATTA,      CS:CODE
START:       MOV    AX, DATTA
             MOV    DS, AX
             MOV    CX, NUM            ;序列数据个数
             MOV    SI, 0
             MOV     DX,     DATA              ;大数存于 DX 中
             MOV     BX,     DATA              ;小数存于 BX 中
             MOV     CX,     NUM-1
             MOV     SI,     2
LOP:         CMP     DX,     DATA[SI]
             JL      BIG_CHG
             CMP     BX,     DATA[SI]
             JG      LITTLE_CHG
             JMP     OUTT
BIG_CHG:     MOV     DX,     DATA[SI]
             JMP     OUTT
LITTLE_CHG: MOV     BL,     DATA[SI]
OUTT:        INC     SI
             INC     SI
             LOOP    LOP
             MOV    MAX, DX
             MOV    MIN , BX
            MOV    AH,     4CH
            INT     21H
CODE         ENDS
```

```
                END     START
```

5-18

```
DATTA     SEGMENT
NUMBER    DB 5, 7, 19h, 23h, 0a0h, 0BH, …,- 1
CNT       EQU      $-NUMBER
PLUS      DB   CNT   DUP(?)
DATTA     ENDS
CODE      SEGMENT
          ASSUME DS:DATTA,   CS:CODE
START:    MOV    AX, DATTA
          MOV    DS, AX
          MOV    CX, CNT              ;序列数据个数
          MOV    SI, OFFSET NUMBER
          LEA    DI, PLUS
LOP:      MOV    AL, [SI]
          CMP    AL ,0
          JG     PUT_PLUS
          JMP    OUTT
PUT_PLUS: MOV [DI],AL
          INC    DI
OUTT:     INC    SI
          LOOP   LOP
          MOV    AH,    4CH
          INT    21H
CODE      ENDS
          END    START
```

5-19

```
DATA     SEGMENT
MARK    DB    98, 23, 90 ,77, 68 ,78 ,84 ,78, 79 ,98
COUNT    EQU $-MARK
A_MARK    DB    0                    ;90 分
B_MARK    DB    0
C_MARK    DB    0
D_MARK    DB    0
E_MARK    DB    0                    ;不及格
AVERAGE    DB    0
DATA      ENDS
CODE      SEGMENT
          ASSUME CS:CODE ,DS: DATA
START: MOV    AX, DATA
          MOV    DS, AX
          MOV    CX, COUNT
          MOV    SI, OFFSET     MARK
```

```
        XOR     AX, AX          ;成绩和清零
LOP:    CMP     BYTE   PTR   [SI], 60
        JAE     MARK_60
        INC     E_MARK                          ;低于 60 分
        JMP     OUTT
MARK_60: CMP    BYTE   PTR    [SI], 70
        JAE     MARK_70
        INC     D_MARK                          ;低于 70 分
        JMP     OUTT
MARK_70: CMP    BYTE   PTR    [SI], 80
        JAE     MARK_80
        INC     C_MARK
        JMP     OUTT
MARK_80: CMP    BYTE   PTR   [SI], 90
        JAE     MARK_90
        INC     B_MARK
        JMP     OUTT
MARK_90: INC    A_MARK
OUTT:   ADD     AL, [SI]
        ADC     AH, 0
        INC     SI
        LOOP    LOP
        MOV     BL, COUNT
        DIV     BL                              ;求平均值
        MOV     AVERAGE, AL
        MOV     AH, 4CH
        INT     21H
CODE    ENDS
        END     START
```

5-20

;;;1)通过 INT21 的 01H 功能输入一个字符，大小写的 ASCII 码相差 20H

;;;2）显示过程中注意回车换行，或者采用空格，否则后出现在屏幕的字符会覆盖前面的字符

```
DATA    SEGMENT
STRING   DB   ?
DATA    ENDS
CODE    SEGMENT
        ASSUME   CS:CODE ,DS: DATA
START:   MOV    AX, DATA
        MOV    DS, AX
   LOP:  MOV AH,01H
        INT 21H                 ;输入字符并回显在屏幕
        CMP AL,'*'
```

```
            JZ OUTT                 ;为结束符退出循环
            MOV STRING,AL           ;将输入字符的 ASCII 码存入内存单元
            MOV DL ,0DH             ;显示回车字符
            MOV AH ,02H
            INT 21H
            MOV DL ,0AH             ;显示换行字符
            MOV AH ,02H
            INT 21H
           MOV     AL ,STRING
           SUB AL,20H               ;输入字符为小写，减 20H 为大写
           MOV    DL ,AL
           MOV AH ,02H
           INT 21H
           MOV DL ,0DH              ;显示回车字符
           MOV AH ,02H
           INT 21H
           MOV DL ,0AH              ;显示换行字符
           MOV AH ,02H
           INT 21H
           JMP LOP                  ;死循环
    OUTT: MOV AH, 4CH
           INT 21H
CODE    ENDS
          END          START
```

5-21

```
DATA       SEGMENT
PKK      DB      'adgdggdaad#ghg#uioo'
CNT    EQU    $-PKK     ;题目要求是 100 个字符，此处自己定义非 100,
CUT      DB      ?
NUM_OFFSET DW    0 ;存放和 PKK 的距离，100 个字符，实际字节数据就可以表示距离值
DATA       ENDS
CODE       SEGMENT
           ASSUME CS: CODE ,DS: DATA
START:     MOV AX, DATA
           MOV DS, AX
           MOV CX,CNT
           MOV CUT,0
           MOV BX,0                         ;采用相对寄存器间接寻址
    AGAIN: CMP PKK[BX] ,'#'
           JNZ NEXT
           INC CUT
           MOV NUM_OFFSET,BX                ;和首个字符的距离值
  NEXT: INC BX
           LOOP AGAIN
```

```
    OUTT: MOV AH, 4CH
          INT 21H
CODE ENDS
          END          START
```

5-22

;;;程序中没有考虑大小写混合的情况，只考虑单一情况

;;;排序采用冒泡法，具体思路参考教材上的 191-193 页

```
DATA       SEGMENT
STRING     DB    'DKSNTEYERTY',0DH,0AH,'$'; 0DH 为回车字符，0AH 为换行字符，
                                                ; '$'显示到此结束
CNT EQU $-STRING-3                          ;去掉最后三个字符
CHAGE_STRING    DB    CNT DUP(?)

DATA       ENDS
CODE       SEGMENT
           ASSUME    CS:CODE ,DS: DATA
START:     MOV    AX, DATA
           MOV    DS, AX

           LEA DX,STRING
           MOV AH,09H
           INT 21H

          MOV DX,CNT-1

  LOP:    MOV CX,DX
          MOV SI,0
          MOV AH,0

AGAIN:    MOV AL ,STRING[SI]
          CMP AL,STRING[SI+1]
          JLE NEXT
           XCHG    AL ,STRING[SI+1]
           XCHG    AL,STRING[SI]
           MOV AH,01H
  NEXT:    INC SI
           LOOP AGAIN
           DEC DX
           OR    AH ,AH
           JNZ    LOP
           LEA DX,STRING
           MOV AH,09H
           INT 21H
```

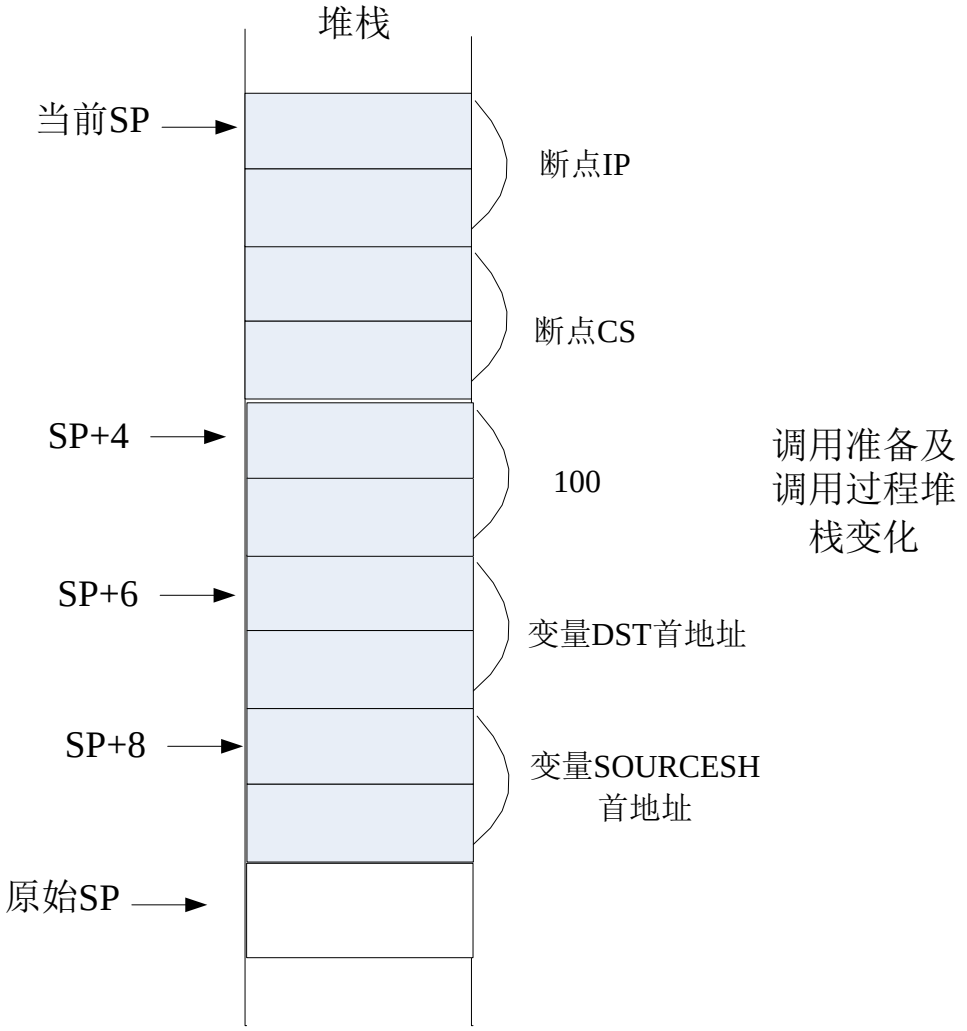
```
    JMP $
    MOV AH, 4CH        ;为了长时间显示,可采用 JMP $ 代替返回 DOS
    INT 21H

CODE  ENDS
      END      START
```

5-23

1）基本功能完成从 SOURCE 开始的地址 100 个字节传送到 DST 开始的空间。子程序中采用了串指令（未讲），可以不用管。

2）子程序的输入参数是通过堆栈传递的，调用子程序前压栈三次，占用了 6 个字节，此处子程序属于段间调用，断点（CS 和 IP）都要压栈保护，由 CPU 完成。

3）子程序调用结束后，没有出栈，所以会浪费堆栈的 6 个字节（在子程序中进行了三次压栈操作）。修改的方法：调用子程序结束后将堆栈指针加 6，或采用 RET 6 返回，或者不采用堆栈传递而通过寄存器传递输入参数。



5-24

;;;;;题意不太明确

```
DATA      SEGMENT
```

```
DATA1    DB    30H,31H,32H,33H,34H,35H,36H,37H,38H,39H, 41H,42H,43H,44H,45H,46H
                                                          ;0-9,A-F 的 ACII 码
DATA2     DB    5   ;待查十六进制个位数
DATA3     DB    ?   ;查表结果

DATA      ENDS
CODE      SEGMENT
          ASSUME    CS:CODE ,DS: DATA
START:    MOV    AX, DATA
          MOV    DS, AX
              ...
          CALL HEX_ASCII
          MOV AH ,4CH
          INT 21H
 HEX_ASCII:PUSH AX
            PUSH BX
            MOV AL ,DATA2
            MOV BX,OFFSET DATA1
            XLAT
            MOV DATA3,AL
            POP BX
            POP AX
            RET
CODE   ENDS
       END        START
```

5-25

;;;;采用 INT21 的 01H 功能输入一个字符，只考虑了 0-9（ASCII 码和数字间相差 30H），未考虑 A-F(ASCII 码和数字间相差 37H)的数据.

;;;;;查找一位数里最大数和最小数为子程序形式

;;;;;;显示字节数据采用子程序的形式，可以显示字节数高低的数可以为 0-F。

```
DATA      SEGMENT
STRING    DB    ?             ;存放输入字符的中间变量
NUM       DB    10 DUP(?) ;存放输入 10 个个位数据
CNT   EQU $-NUM
MAX   DB    ?
MIN   DB    ?
DATA      ENDS
CODE      SEGMENT
          ASSUME    CS:CODE ,DS: DATA
START:    MOV    AX, DATA
          MOV    DS, AX
          MOV CX,10
          LEA    SI,NUM
 LOP:     MOV AH,01H
```

```
        INT 21H                 ;输入字符并回显在屏幕
        MOV STRING,AL           ;将输入字符的 ASCII 码存入内存单元
        MOV DL ,20H             ;显示空格字符
        MOV AH ,02H
        INT 21H
        MOV AL, STRING
        SUB AL,30H              ;字符转换成数字
        MOV [SI],AL
        INC   SI
        LOOP   LOP
        MOV DL ,0DH             ;显示回车字符
        MOV AH ,02H
        INT 21H
        MOV DL ,0AH             ;显示换行字符
        MOV AH ,02H
        INT 21H
        LEA SI,NUM
        MOV CX,CNT
       CALL MAX_MIN             ;调用查个位数中最大和最小数
       MOV MAX,DH
       MOV MIN,DL
       MOV BL ,MIN
       CALL DIS_BYTE        ;调用字节显示子程序显示最小值

       MOV DL ,20H              ;显示空格字符
       MOV AH ,02H
       INT 21H

       MOV BL ,MAX
       CALL DIS_BYTE        ;调用字节显示子程序显示最大值
       JMP   $                  ;为了长时间显示,可采用 JMP $ 代替返回 DOS
      MOV AH, 4CH
      INT 21H
;显示字节子程序
;输入参数（调用前提）:要显示数据-->BL
;输出参数（调用结果）：无
DIS_BYTE:
        PUSH   CX
        PUSH   DX
        PUSH   AX
        MOV    AL, BL
        AND    AL, 0F0H
        MOV CL, 4                   ;取高 4 位
        SHR    AL, CL
```

```
        CMP    AL, 0AH          ;是否是 A 以上的数
        JB     ADD_30
        ADD    AL, 07H          ; A-F 字符加 37H
ADD_30: ADD    AL, 30H          ; 0-9 字符加 30H
        MOV    DL, AL           ;显示字符
        MOV    AH, 02H
        INT    21H
        MOV    AL, BL
        AND    AL, 0FH          ;取低 4 位
        CMP    AL, 0AH
        JB     ADD_30_1
        ADD    AL, 07H
ADD_30_1: ADD    AL, 30H
          MOV    DL, AL         ;显示字符
          MOV    AH, 02H
          INT    21H
               POP    AX
               POP    DX
               POP    CX
          RET
;   查找无符号字节序列数据中最大和最小值
 ;输入参数（调用前提）：  数据的首地址-->SI ,元素个数-->CX
 ;输出参数（调用结果）：  最大数-->DH ,最小数-->DL
 MAX_MIN    PROC NEAR
          MOV DH,[SI]
          MOV DL,[SI]
          DEC CX
          INC SI
 AGAIN: CMP [SI] ,DL
          JAE CHECK_MAX
          MOV [SI], DL
        JMP NEXT
CHECK_MAX: CMP [SI],DH
        JBE NEXT
        MOV DH,[SI]
NEXT: INC SI
        LOOP AGAIN
        RET
MAX_MIN ENDP
CODE ENDS
          END          START
```