★★★

```
        PUSH    DS
        XOR     AX,AX
        PUSH    AX
        MOV     AX,DATA
        MOV     DS,AX
        MOV     ES,AX                   置附加段寄存器

        LEA     DX,SOURCE               数据区起始地址
        LEA     SI,NAME0                原始数据文件名
        MOV     CX,N                    字节数
        CALL    LOAD                    ;从'INPUT1.DAT' 中读取数据
;   **** BEGIN ****
        LEA     SI, RESULT
        (1)     DI,OFFSET PDATA  ;PDATA 为正数数组存放缓冲区首址
        MOV     BX,OFFSET NDATA ;NDATA  为负数数组存放缓冲区首址
        XOR     DX,DX
        MOV     CX,N
        CLD
MAIN1:  LODSB
        TEST    AL,(2)
        JZ      MAIN2
        INC     DH
        MOV     [BX],AL
        INC     BX
        (3)
MAIN2:  INC     DL
        MOV     [DI],AL
        INC     DI
MAIN3:  (4)     MAIN1
        LEA     SI,PDATA
        LEA     DI,RESULT
        MOV     [DI],DL
        INC     DI
        XOR     CX,CX
        MOV     CL,DL
MAIN4:  MOV     AL,(5)
        MOV     [DI],AL
        INC     DI
        INC     SI
```

```
            LOOP        ____(6)____
            MOV         [DI],DH
            INC         DI
            XOR         CX,CX
            MOV         CL,DH
            MOV         BX,OFFSET NDATA
MAIN5:      MOV         AL,[BX]
            MOV         [DI],AL
            INC         DI
            ____________(7)____________
            LOOP        MAIN5
;     **** END ****
            LEA         DX,RESULT       结果数据区首址
            LEA         SI,NAME1        结果文件名起始地址
            MOV         CX,N+2          字节数
            CALL        SAVE            保存结果到'OUTPUT1.DAT'  文件
            RET
START       ENDP
CODE        ENDS
            END         START
```

【答案】（1）MOV
（2）80H
（3）JMP MAIN3
（4）LOOP
（5）[SI]
（6）MAIN4
（7）INC BX

【分析】

此程序主要实现正数和负数的分类，以及统计正数和负数的个数。正数和负数的所有结果存放在寄存器 DI 中。寄存器 DL 存放正数的个数，DH 存放负数的个数。

程序的数据段中定义了六个字节型变量,原始数据区 SOURCE 、结果数据区 RESULT 、原始数据文件名 NAME0 、结果数据文件名 NAME1 ,以及正数和负数数组存放缓冲区首址。数据段下面是代码段,告诉汇编程序,代码段、数据段和堆栈段分别属于段寄存器 CS、DS、SS。

在程序的开始部分，是对返回地址的保存和数据段寄存器的初始化，这是每一个汇编程序所必须具有的操作步骤。准备工作完成以后，程序进入主体部分。

程序的开始，将原始数据偏移地址装入到 SI 中。正数和负数存放缓冲区首址分别送 DI 和 BX 中，所以第（1）空填写“MOV”。然后用 LODSB 指令将 SI 当前所指向的地址

单元（即[SI]）中内容装入到 AL 中。接着要判断 AL 的正负性，第（2）空填写“80H”，将 AL 和 80H 做 TEST 指令操作。若 TEST 指令的结果 ZF=0，则存放负数个数的 DH 加 1，然后转向 MAIN3 段，所以第(3)空填写 JMP MAIN3；若 ZF=1，则 AL 为正数，转向 MAIN2 段，存放正数个数的 DL 加 1。

MAIN3 段开始处要填写第（4）空，显然应该是“LOOP MAIN1”，即继续判断下一个数 AL 的正负性。然后将正数个数 DL 送 CL，用于控制循环次数。再依次将 DL 个正数存入到 DI 当前所指向的地址单元（即[DI]）中。第(5)空填写‘SI”，SI 装入的是正数的偏移地址。第（6）空考查的是循环，装入正数到[DI]的循环入口处是 MAIN4。

MAIN5 段是依次将负数存放到[DI]中，将负数个数 DH 赋值给 CL，控制装入负数的循环次数。因为 BX 中装入的是负数的偏移地址，所以每次存入一个负数后，偏移地址 BX 加 1，指向下一个负数。所以第（7）空填写‘INC BX”。

END 后面的指令是分别将结果数据区首址、结果文件名装入到 DX 和 SI 中，并保存结果文件。

★★★

```
STAC    ENDS

DATA    SEGMENT
SOURCE  DB      N DUP(?)        顺序存放10个字节数
RESULT  DB      N DUP(0)        存放结果
NAME0   DB      'INPUT1.DAT',0
NAME1   DB      'OUTPUT1.DAT',0
DATA    ENDS

CODE    SEGMENT
        ASSUME  CS:CODE, DS:DATA, SS:STAC
START   PROC    FAR
        PUSH    DS
        XOR     AX,AX
        PUSH    AX
        MOV     AX,DATA
        MOV     DS,AX
        LEA     SI,SOURCE       数据区起始地址
        LEA     SI,NAME0        原始数据文件名
        MOV     CX,N            字节数
        CALL    LOAD            ;从'INPUT1.DAT'中读取数据
;       **** BEGIN ****
        LEA     DI, RESULT
        LEA     SI, SOURCE
        MOV     CX,10
AGN0:   MOV     AL,[SI]
        ___(1)___
        MOV     CX,8
        MOV     BX,0
AGN1:   MOV     AH,0
        SHL     BL,1
        ___(2)___ AL,1
        RCL     AH,L
        CMP     AH,___(3)___
        _____(4)_____
        JMP     NEXT
SET_ONE: OR     BL,01H
NEXT:   MOV     ___(5)___,BL
        _____(6)_____
```

需要全本请访问 www.ertongbook.com

```
                LOOP        AGN1
                            (7)
                MOV         [DI],BL
                INC         SI
                INC         DI
                LOOP        AGN0
;     **** END ****
                LEA         DX,RESULT       结果数据区首址
                LEA         SI,NAME1        结果文件名
                MOV         CX,N            结果字节数
                CALL        SAVE            保存结果到文件
                RET
START           ENDP
CODE            ENDS
                END         START
```

【答案】(1) PUSH CX

(2) RCL

(3) BH

(4) JNE SET_ONE

(5) BH

(6) AND BH，01H

(7) POP CX

【分析】

此程序是实现将 10 个无符号 8 位格雷码转换成 10 个 8 位二进制数的功能。关键是要处理好移位的过程。移位是为了处理每一位格雷码向二进制的转换。寄存器 AX 存放的是原格雷码，BX 存放处理后的二进制码。其中 AL 与 BL 分别存放处理前的格雷码以及处理后的二进制码；而 AH 和 BH 是用作异或处理的。

程序的堆栈段定义了一个 128 字节的堆栈区。堆栈段下面是数据段。数据段中定义了四个字节型变量，原始数据区 SOURCE、结果数据区 RESULT、原始数据文件名 NAME0、结果数据文件名 NAME1。数据段下面是代码段，告诉汇编程序，代码段、数据段和堆栈段分别属于段寄存器 CS、DS、SS。

AGN0 段是给一些寄存器如 CX、BX、AH 赋初值。第(1)、(7)空分别填写压入和弹出 CX。压入堆栈寄存器，是为了保存返回的地址，而弹出操作，是当程序结束后能够正确的返回操作系统。CX 是用于控制循环次数，压入堆栈后， CX 赋值为 8，因为每个数有 8 位格雷码，需要循环 8 次处理每一位码子；每次处理完一个数后，再从栈顶弹出 CX。

第 2）空及下一条指令都是把进位标志位 CF 包含在循环移位中的左循环移位指令。

这样就实现了将的 AL 最高位移入 CF，然 后 CF 又移入到 AH 的最低位。

第（3）空则是处理格雷码，其中 AH 中存放的是当前的格雷码（如 g6），而 BH 中存放的前一位二进制码（如 d7），将它们进行异或操作，就得到了 d6。

第（4）空考查的是异或操作，若 AH 与 BH 中内容不同，则异或结果为 1，否则为 0。

第（5）空是将 BL 中二进制码子移入到 BH 中，以便处理下一个格雷码。

第（6）空是将 BH 内容的高 7 位都屏蔽掉，所以填写指令“ AND BH，01H”。因为每一位格雷码转换成二进制码时，仅与前一个二进制码有关，与其他的二进制码无关，所以只留下当前所需要的 1 位二进制码。

接着，将转换后的二进制码存入到 DI 当前所指向的地址单元（即[DI]）中，SI 和 DI 地址各加 1，循环处理下一个格雷码，循环入口处是 AGN0。

★★★

```
NAME1   DB      'OUTPUT1.DAT',0
DATA    ENDS

CODE    SEGMENT
        ASSUME  CS:CODE, DS:DATA, SS:STAC
START   PROC    FAR
        PUSH    DS
        XOR     AX,AX
        PUSH    AX
        MOV     AX,DATA
        MOV     DS,AX
        LEA     SI,SOURCE       数据区起始地址
        LEA     SI,NAME0        原始数据文件名
        MOV     CX,N*2          字数
        CALL    LOAD            ;从'DAT'中读取数据
;   **** BEGIN ****
        LEA     SI,SOURCE
        LEA     DI,RESULT
        MOV     CX,N
NEXT0:  MOV     AX,[SI]
        MOV     [DI],AX
        ADD     SI,___(1)___
        _____(2)_____
        LOOP    ___(3)___
        CLD
        MOV     BX,N-1
MAL1:   LEA     SI,RESULT
        MOV     CX,___(4)___
NEXT:   LOD___(5)___
        CMP     [SI],AX
        JAE     CONT
        XCHG    [SI],___(6)___
        MOV     [SI-2],AX
CONT:   LOOP    ___(7)___
        _____(8)_____
        ___(9)___       MAL1
;   **** END ****
        LEA     DX,RESULT       结果数据区首址
        LEA     SI,NAME1        结果文件名
```

```
        MOV     CX,N*2          结果字数
        CALL    SAVE            保存结果到文件
        RET
START   ENDP
CODE    ENDS
        END     START
```

【答案】（1）2
（2）ADD DI，2
（3）NEXT0
（4）BX
（5）SW
（6）AX
（7）NEXT
（8）DEC BX
（9）JNZ

【分析】

此程序是用冒泡排序法对一个含 20 个无符号数的序列实现升序排序的功能。冒泡排序方法的原理为：从第一个数据开始与相邻的数进行比较，若次序不对，两数交换位置。依次类推，直至第 n-1 个数和第 n 个数进行过比较为止。上述过程称为第一趟冒泡排序。第一趟冒泡排序经过 n-1 次比较后，最大的数已经到了数组尾，第二趟仅需比较 n-2 次比较就够了，一共比较 n-1 趟就完成了排序。这样共有两重循环： CX 是内循环次数，而 BX 是外循环次数。

程序的堆栈段定义了一个 128 字节的堆栈区。堆栈段下面是数据段。数据段中定义了两个字型变量：原始数据区 SOURCE 、结果数据区 RESULT ；两个字节型变量：原始数据文件名 NAME0 、结果数据文件名 NAME1 。数据段下面是代码段，告诉汇编程序，代码段、数据段和堆栈段分别属于段寄存器 CS、DS、SS。

第（1）、（2） 空 SI 和 DI 地址要分别加 2，因为原始数据和结果数据都是以字型存放的。

接着开始循环，循环入口处是 NEXT0 ，所以第（3） 空填写“ NEXT0”。第（3）空之前的几条指令是实现将原始数据区 SOURCE 中的数据依次装入到 RESULT 中。

CLD 指令是将方向标志 DF 清 0，在数据操作时，实现地址指针自动增量。

MAL1 段和 NEXT 段是实现每趟内的数据比较。第（4）空中第一趟比较 N-1 次后，最大的数已到了末尾，第二趟仅需比较 N-2 次就够了，所以 CX 是随着 BX 变化的。

第（5）空是需要将由 SI 作为地址指针的串元素装入到 AX 中，同时修改 SI ，指向下一个串元素。

然后将下一个数与当前的数进行比较，若大于，则继续循环，取再一个数；若小于，则两数交换。所以第（6）空应该填写“ AX”。

第（7）、（8）、（9 ）空则是完成一趟比较后，BX 应减 1，若 BX 不为 0，则继续循环，

进行第二趟比较,循环入口处是 NEXT。所以第(7) 空填写“ NEXT”，第(8) 空填写“ DEC BX ”,第 (9) 空填写“ JNZ”。

★★★

```
START   PROC    FAR
        PUSH    DS
        XOR     AX,AX
        PUSH    AX
        MOV     AX,DATA
        MOV     DS,AX

        LEA     DX,SOURCE       数据区起始地址
        LEA     SI,NAME0        原始数据文件名
        MOV     CX,N*2          字节数
        CALL    LOAD            ;从'INPUT1.DAT'中读取数据
;       **** BEGIN ****
        MOV     DI,0
        MOV     SI,0
        MOV     CX,20
        MOV     BX,   (1)
AGN0:   MOV     AX,SOURCE[SI]
        PUSH    SI
        PUSH    CX
        MOV     (2) , (3)
AGN1:   INC     SI
        INC     SI
        ADD     AX,SOURCE[SI]
        LOOP    AGN1
                (4)
        DIV     BX
        MOV     RESULT[DI],AX
        INC     DI
                (5)
        POP     CX
        POP     SI
        INC     SI
                (6)
        LOOP    AGN0
;       **** END ****
        LEA     DX,RESULT       结果数据区首址
        LEA     SI,NAME1        结果文件名
        MOV     CX,40           结果字节数
        CALL    SAVE            保存结果到文件
```

需要全本请访问 www.ertongbook.com

```
        RET
START   ENDP
CODE    ENDS
        END     START
```

【答案】（1）5
（2）CX
（3）4
（4）CWD
（5）INC DI
（6）INC SI

【分析】

此程序是处理 24 个无符号二进制字序列，其中前 20 个二进制字进行移动平均处理，将要处理的字用以它为开始的连续五个字的平均数代替，最后 4 个二进制字数据不变。要注意每个二进制字是占两个字节。所以，每次寻址时，数据区起始地址 SI 两次加 1，结果数据区地址 DI 也是两次加 1。

程序的堆栈段定义了一个 128 字节的堆栈区。堆栈段下面是数据段。数据段中定义了两个字型变量：原始数据区 SOURCE、结果数据区 RESULT；两个字节型变量：原始数据文件名 NAME0、结果数据文件名 NAME1。数据段下面是代码段，告诉汇编程序，代码段、数据段和堆栈段分别属于段寄存器 CS、DS、SS。

第（1）空中，由下面的程序语句可知，BX 是作为 AX 的除数，因为是求 5 个数的平均数，所以应该填“5”。

AHN0 段是将原始数据装入到 AX 中，其中 SI 是作为原始数据的偏移地址，其初值为 0。

第（2）、（3）空中 CX 是用来控制循环次数，因为是第 1 个数与后面的 4 个数相加，所以要循环四次，故 CX 赋值为 4。

第（4）空的前几条指令是将 SI 地址加 2，取下一个数据，与第一个数据相加。然后再从 AGN1 处开始循环，再将 AX 与后三个数相加。接着求平均数，将五个数的和除以 5，除法结果，商在 AX 中，余数在 DX 中。因为余数舍去，所以第（4）空是要给 DX 赋初值为 0。将求得的平均数 AX 存入到 [DI] 中。

第（5）空是将 DI 地址再加 1，第（6）空是将 SI 地址再加 1，因为不管是原始数据还是结果数据，它们都是以字的形式存放的，两个字节一共占内存中的连续两个地址。

求完一个平均数后，接着开始求下一组五个数的平均数，循环从 AGN0 开始。

★★★

即第一个数据不变，其后的数据取与前一数据的差值并用四位二进制补码表示，两个差值拼成一个字节，前一个差值放在高四位，后一个差值放在低四位。

例如：

原数据（X[n]）：23H，27H，2AH，29H，22H…

压缩后（Y[n]）：23H，　　43H，　　F9H…

编程按上述方法进行压缩，结果保存在 RESULT 开始的内存单元中。

部分程序已给出，请填空 BEGIN 和 END 之间已给出的源程序使其完整（空白已用横线标出，每个空白一般只需一条指令，但采用功能相当的多条指令亦可），或删除 BEGIN 和 END 之间原有的代码并自行编写程序片段来完成所要求的功能。

原始数据由过程 LOAD 从文件 INPUT1.DAT 中读入 SOURCE 开始的内存单元中，结果要求从 RESULT 开始存放，由过程 SAVE 保存到文件 OUTPUT1.DAT 中。

对程序必须进行汇编，并与 IO.OBJ 链接产生可执行文件，最终运行程序产生结果。调试中若发现整个程序中存在错误之处，请加以修改。

试题程序：

```
            EXTRN     LOAD:FAR,SAVE:FAR
N           EQU       10

STAC        SEGMENT   STACK
            DB        128 DUP (?)
STAC        ENDS

DATA        SEGMENT
SOURCE      DB        2*N+1   DUP(?)
RESULT      DB        N+1     DUP(0)
NAME0       DB        'INPUT1.DAT',0
NAME1       DB        'OUTPUT1.DAT',0
DATA        ENDS

CODE        SEGMENT
            ASSUME    CS:CODE, DS:DATA, SS:STAC
START       PROC      FAR
            PUSH      DS
            XOR       AX,AX
            PUSH      AX
            MOV       AX,DATA
            MOV       DS,AX
            MOV       ES,AX          置附加段寄存器

            LEA       DX,SOURCE      数据区起始地址
```

```
                LEA     SI,NAME0        原始数据文件名起始地址
                MOV     CX,2*N+1        字节数
                CALL    LOAD            ;从'INPUT1.DAT'中读取数据
;       **** BEGIN ****
                LEA     SI,SOURCE
                LEA     DI,RESULT
                CLD
                MOVSB                   ;Y[0]=X[0]
                XOR     BX,BX           ;FLAG=0
                MOV     DX,N*2          ;COUNTER
COMPRESS:
                LODSB
                SUB     (1)             ;X[n]-X[n-1]
                (2)                     ;FLAG=NOT FLAG
                J(3)    LOW_HEX
                MOV     AL,CL
                (4)     AL,CL
                MOV     AH,AL
                JMP     NEXT
LOW_HEX:
                (5)
                OR      AL,AH
                STOSB
NEXT:           (6)
                JNE     COMPRESS
;       **** END ****
                LEA     DX,RESULT       结果数据区首址
                LEA     SI,NAME1        结果文件名起始地址
                MOV     CX,N+1          字节数
                CALL    SAVE            保存结果到'OUTPUT1.DAT'文件中
                RET
START           ENDP
CODE            ENDS
                END     START
```

【答案】（1）AX，[SI-2]

（2）XOR BX，0FFH

（3）Z

（4）SHL
（5）AND AL，0FH
（6）DEC DX

【分析】

此程序是实现对 21 个八位有符号数的差分压缩。关键要分清两个差值中，有一个是存放 AX 的高 字节即 AH，另一个差值是放低字节即 AL，然后再按要求拼成一个字节。还要注意当差值为负数时，要转换成补码形式。

程序的堆栈段定义了一个 128 字节的堆栈区。堆栈段下面是数据段。数据段中定义了四个字节型变量，原始数据区 SOURCE 、结果数据区 RESULT 、原始数据文件名 NAME0、结果数据文件名 NAME1。数据段下面是代码段，告诉汇编程序，代码段、数据段和堆栈段分别属于段寄存器 CS、DS、SS。

LODSB 指令是将以原始数据区 SI 作为地址指针的数据装入到 AX 中。接着要将后一个数 AX 与前一个数，即[SI-2]中内容求差值。因为每一个数占一个字节，所以[SI-2]表示后一个数。所以第（1）空填写“ AX，[SI-2]”。

第（2）空的填写，首先要理解寄存器 BX 的作用。若 BX=0 时，表明刚才做的减法结果得到的是前一个差值，则将该差值放入结果数据的高四位；若 BX=1，则表明刚才做的减法结果得到的是后一个差值，则将该差值放入结果数据的低四位。每次做减法后，将 BX 取反，即由 1变成 0,或者由 0变成 1，所以第(2) 空填写“ XOR BX，0FFH”。这样能保证后一个差值是存放在与前一个差值不同的四位上。第（3）空填写“ Z”。由异或结果的标志状态位 ZF 来判断，当 ZF=1 时，即当 BX=1,则差值结果放低四位(这时已在 AL 中)，所以转向 LOW_HEX 段；否则，将差值存放在高四位，将差值 AL 逻辑左移四位，再赋值给 AH。所以第(4) 空应该填写“ SHL”。是将 AL 逻辑左移 4 位，再传送高四位给 AH。

第（5）空填写“ AND AL，0FH”，是将差值取补码。因为差值为正数时，其补码为原数：差值为负数时，要取补码才是正确结果。

注意指令“ OR AL,AH ”是将得到的两个差值结果结合起来，拼成一个字节，再赋值给 AL,再用 STOSB 指令，将 AL 中的内容传送到结果数据区地址[DI]中。

第（6）空是处理完一次差分压缩后，DX 次数减 1,若 DX 不为 0,继续从 COMPRESS 开始循环。

★★★

32H（'2'），…，39H（'9'）（共 20 个 ASCII 字符）

结果为 24H（'$'），00H，01H，01H，01H，24H（'$'），02H（'2'），…，09H（'9'）（共 20 个字节）

部分程序已给出，其中原始数据由过程 LOAD 从文件 INPUT1.DAT 中读入 SOURCE 开始的内存单元中。运算结果要求从 RESULT 开始存放，由过程 SAVE 保存到文件 OUTPUT1.DAT 中。

请填空 BEGIN 和 END 之间已给出的源程序使其完整（空白已用横线标出，每个空白一般只需一条指令，但采用功能相当的多条指令亦可），或删除 BEGIN 和 END 之间原有的代码并自行编程来完成所要求的功能。

对程序必须进行汇编，并与 IO.OBJ 链接产生可执行文件，最终运行程序产生结果。调试中若发现整个程序中存在错误之处，请加以修改。

试题程序：

```
        EXTRN   LOAD:FAR,SAVE:FAR
N       EQU     20

STAC    SEGMENT STACK
        DB      128 DUP (?)
STAC    ENDS

DATA    SEGMENT
SOURCE  DB      N DUP(?)
RESULT  DB      N DUP(0)
NAME0   DB      'INPUT1.DAT',0
NAME1   DB      'OUTPUT1.DAT',0
DATA    ENDS

CODE    SEGMENT
        ASSUME  CS:CODE, DS:DATA, SS:STAC
START   PROC    FAR
        PUSH    DS
        XOR     AX,AX
        PUSH    AX
        MOV     AX,DATA
        MOV     DS,AX

        LEA     DX,SOURCE       数据区起始地址
        LEA     SI,NAME0        原始数据文件名
        MOV     CX,N            字节数
        CALL    LOAD            ;从'INPUT1.DAT' 中读取数据
```

```
;     **** BEGIN ****
              (1)
              MOV       DI, 0
              (2)
CHAN:         (3)
              SUB       AL,20H
              JZ        CHANGE
              SUB       AL,   (4)
              MOV       RESULT[DI],AL
              (5)
              (6)
              DEC       CX
              JZ        EXIT
              JMP       CHAN
CHANGE:       MOV       RESULT[DI],24H
              INC       DI
              INC       SI
              JMP       (7)
EXIT:         NOP
;     **** END ****
              LEA       DX,RESULT      结果数据区首址
              LEA       SI,NAME1       结果文件名
              MOV       CX,N           结果字节数
              CALL      SAVE           保存结果到文件
              RET
START         ENDP
CODE          ENDS
              END       START
```

【答案】（1）MOV SI，0

（2）MOV CX，N

（3）MOV AL，SOURCE [SI]

（4）30H

（5）INC DI

（6）INC SI

（7）CHAN

【分析】

此程序比较简单，要实现将 20 **个数字转换的功能，即** 0~9 **之间数字的** ASCII **字符转**

需要全本请访问 www.ertongbook.com