将小写字符串转化为大写

```
assume cs:code
data segment
        db 'conversation'
data ends
code segment
    start:
            mov  ax,data
            mov  ds,ax
            mov  si,0                       ;ds:si指向字符串的首地址
            mov  cx,12                      ;cx存放字符串的长度
            call captical
            mov  ax,4c00h
            int  21h
    captical:
            and  byte ptr [si],11011111b
            inc  si
            loop captical
            ret
code ends
end start
```

设计一个子程序：将一个全是字母，以0结尾的字符串转化为大写

```
assume cs:code
data segment
        db 'conversation',0
data ends
code segment
    start:
            mov  ax,data
            mov  ds,ax
            mov  si,0                       ;ds:si指向字符串的首地址

            call captical
            mov  ax,4c00h
            int  21h
    captical:
            mov  cl,[si]
            mov  ch,0
            jcxz ok
            and  byte ptr [si],11011111b
            inc  si
            jmp  short captical
    ok:     ret
code ends
end start
```

将data段中字符串全部转化为大写

```
assume cs:code,ds:data
data segment
        db 'word',0
        db 'unix',0
        db 'wind',0
        db 'good',0
data ends
code segment
    start:
            mov  ax,data
            mov  ds,ax
            mov  bx,0
            mov  cx,4
    s:      mov  si,bx
            call captical
            add  bx,5
            loop s
            mov  ax,4c00h
            int  21h
    captical:
            mov  cl,[si]
            mov  ch,0
            jcxz ok
            and  byte ptr [si],11011111b
            inc  si
            jmp  short captical
    ok:     ret
code ends
end start
```

显示字符串

```
assume cs:code, ds:data
data segment
        db 'Hello, World!',0
data ends
code segment
    start:
            mov  dh,8               ;dh装行数范围1-25
            mov  dl,3               ;dl装列数范围1-80
            mov  cl,2
            mov  ax,data
            mov  ds,ax
            mov  si,0
            call show_str
            mov  ah,4ch
            int  21h
    show_str:                       ;显示字符串
            push cx
            push si
            mov  al,0a0h            ;每行有80*2==160字节==0a0h字节内容
            dec  dh
```

```
            mul  dh
            mov  bx,ax
            mov  al,2                ;每个字符占两个字节
            mul  dl
            add  bx,ax
            mov  ax,0b800h           ;显存开始的地址
            mov  es,ax               ;es中存放的是显存的第0页（共0--7页）的起始的段地址
            mov  di,0
            mov  al,cl
            mov  ch,0
    s:      mov  cl,ds:[si]          ;ds:[si]指向"Hello, World!"" 取到0后跳出循环
            jcxz ok
            mov  es:[bx+di],cl       ;偶地址存放字符
            mov  es:[bx+di+1],al     ;奇地址存放字符颜色属性
            inc  si
            add  di,2
            jmp  short s
    ok:
            pop  si
            pop  cx
            ret
code ends
end start
```

定义最多20个字符，大写变小写，小写变大写

```
DATA SEGMENT                ; 数据段及变量定义
    MAXLEN DB 20            ; 最大输入长度
    ACTLEN DB ?             ; 实际输入长度
    STR1   DB 21 DUP (?)    ; 字符串存储区（20字符 + 1终止符）
DATA ENDS

CODE SEGMENT                                ; 代码段定义
            ASSUME CS:CODE, DS:DATA         ; ASSUME 指令

    BEGIN:
            MOV    AX, DATA                 ; DS 初始化
            MOV    DS, AX

    ; ========== 1. 输入字符串 ==========
            MOV    DX, OFFSET MAXLEN        ; DS:DX -> MAXLEN
            MOV    AH, 0AH                  ; DOS 功能：缓冲区输入
            INT    21H

    ; ========== 2. 输出回车换行 ==========
            MOV    DL, 0DH                  ; 输出回车
            MOV    AH, 02H
            INT    21H

            MOV    DL, 0AH                  ; 输出换行
            MOV    AH, 02H
            INT    21H

    ; ========== 3. 给字符串添加 '$' 结尾 ==========
            LEA    BX, STR1                 ; BX 指向字符串起始地址
```

```
                MOV     CL, [ACTLEN]            ; AL/CL 中保存实际输入长度
                XOR     CH, CH                  ; CH = 0
                ADD     BX, CX                  ; BX += 实际长度
                MOV     BYTE PTR [BX], '$'      ; 在末尾添加 '$'

    ; ========== 4. 大小写互换处理 ==========

    ; 再次让 BX 指向字符串的开头
                LEA     BX, STR1

    ; 重新加载实际长度到 CX
                MOV     CL, [ACTLEN]
                XOR     CH, CH                  ; CH = 0 保证 CX=ACTLEN

    LOP1:
    ; 读取当前字符
                MOV     AL, [BX]

    ; 检查是否为大写字母 A~Z
                CMP     AL, 'A'
                JB      CHECK_LOWER             ; 如果小于 'A'，跳转检查小写
                CMP     AL, 'Z'
                JA      CHECK_LOWER             ; 如果大于 'Z'，跳转检查小写

    ;---------- 当前是大写字母 ----------
                ADD     [BX], 20H               ; A~Z => a~z
                JMP     NEXT_CHAR

    CHECK_LOWER:
    ; 检查是否为小写字母 a~z
                CMP     AL, 'a'
                JB      NEXT_CHAR               ; 如果小于 'a'，直接不管
                CMP     AL, 'z'
                JA      NEXT_CHAR               ; 如果大于 'z'，也不管

    ;---------- 当前是小写字母 ----------
                SUB     [BX], 20H               ; a~z => A~Z

    NEXT_CHAR:
    ; 移动到下一个字符
                INC     BX
                DEC     CX
                JNZ     LOP1

    ; ========== 5. 显示转换后的字符串 ==========
                LEA     DX, STR1                ; DX 指向 STR1
                MOV     AH, 09H                 ; DOS 功能：显示字符串(以$结束)
                INT     21H

    ; ========== 6. 结束程序 ==========
                MOV     AH, 4CH                 ; 4CH = 结束程序
                INT     21H

CODE ENDS
END BEGIN
```

从键盘输入一组以$为结束符的字符串，对其数字字符计数并求得所有数字之和，显示

```
DATAS SEGMENT
    string db 100 dup('$')     ; 存放输入字符，直到遇到 '$'
    line   db 0dh,0ah,'$'      ; 换行字符(可选)
    count  db 0                ; 用于存“数字个数”(0~255)
    sum    db 0                ; 用于存“数字和”(0~255)，不考虑溢出
DATAS ENDS

STACKS SEGMENT
           dw 128 dup(?)     ; 简易堆栈区(128 个字)
STACKS ENDS

CODES SEGMENT
               ASSUME CS:CODES, DS:DATAS, SS:STACKS

    START:
    ; ========== 0. 初始化数据段、堆栈段 ==========
               MOV    AX, DATAS
               MOV    DS, AX

               MOV    AX, STACKS
               MOV    SS, AX
               MOV    SP, 128*2                          ; 堆栈顶(256 字节)。根据实际需要可调整

    ; ========== 1. 准备指针，循环逐字符读入直到 '$' ==========
               LEA    SI, string
               MOV    count, 0                           ; 数字个数 清0
               MOV    sum, 0                             ; 数字和 清0

    input:
               MOV    AH, 1                              ; DOS 功能号1：逐字符输入(带回显)
               INT    21H                                ; 返回字符在 AL
               MOV    [SI], AL
               INC    SI
               CMP    AL, '$'
               JNZ    input                              ; 若不是 '$' 则继续输入

    ; ========== 2. 重新从头扫描 string，统计数字个数与和 ==========
               LEA    SI, string

    next:
               MOV    CL, [SI]
               INC    SI
               CMP    CL, '$'
               JZ     disp                               ; 若遇到 '$' 则结束统计

               CMP    CL, '0'
               JB     next                               ; 若 < '0' 不是数字
               CMP    CL, '9'
               JA     next                               ; 若 > '9' 不是数字
```

```
    ;----- 当前字符是 '0'~'9' -----
                INC     count                           ; 数字个数 +1
                SUB     CL, '0'                         ; 转成数值(0~9)
                ADD     sum, CL                         ; sum += (CL)

                JMP     next

    ; ========== 3. 显示结果：先数字个数，再数字和 ==========
    disp:
    ; -- (可选) 输出换行 --
                MOV     DL, 0Dh
                MOV     AH, 2
                INT     21H
                MOV     DL, 0Ah
                MOV     AH, 2
                INT     21H

    ; --- 显示“数字个数” ---
                MOV     AL, count
                CALL    Print2Dec                       ; 最多两位十进制输出
    ; 换行
                MOV     DL, 0Dh
                MOV     AH, 2
                INT     21H
                MOV     DL, 0Ah
                MOV     AH, 2
                INT     21H

    ; --- 显示“数字和” ---
                MOV     AL, sum
                CALL    Print2Dec
    ; 换行
                MOV     DL, 0Dh
                MOV     AH, 2
                INT     21H
                MOV     DL, 0Ah
                MOV     AH, 2
                INT     21H

    ; ========== 4. 结束程序 ==========
                MOV     AH, 4CH
                INT     21H

    ;-------------------------------------------------------------
    ; 子程序: Print2Dec
    ;   - 入口: AL = 0~255 的数（本示例仅演示 0~99）
    ;   - 功能: 以十进制(最多两位)打印到屏幕 (不带回车)
    ;   - 若需支持 3位或以上，请自行扩展
    ;-------------------------------------------------------------
Print2Dec PROC
                CMP     AL, 100
                JA      PD_3DIGITS                      ; 若想支持 3位，可自行扩展

                CMP     AL, 10
                JB      PD_1DIGIT                       ; 若 < 10, 只打印一位
```

```
    ;--------- 两位数打印 (10~99) ---------
    ; AL ÷ 10 = 商(0~9)，余(0~9)
                XOR     AH, AH
                MOV     BL, 10
                DIV     BL                                      ; AL=商，AH=余
    ; 显示(商+'0')
                ADD     AL, '0'
                MOV     DL, AL
                MOV     AH, 2
                INT     21H
    ; 显示(余+'0')
                ADD     AH, '0'
                MOV     DL, AH
                MOV     AH, 2
                INT     21H
                RET

    PD_1DIGIT:
    ;------ 一位数字(0~9) ------
                ADD     AL, '0'
                MOV     DL, AL
                MOV     AH, 2
                INT     21H
                RET

    PD_3DIGITS:
    ; 如果你需要打印三位(100~255)，可在此扩展
    ; 目前直接当作两位来演示(将不正确显示)
    ; ...
                RET
Print2Dec ENDP

CODES ENDS
END START
```

题目要求：设变量DATA中存有20位学生的高等数学成绩（百分制表示），试用子程序编写一个程序统计0-59分、60—69分，70—79分，80—89分，90—99分和100分的人数并分别存放到T5、T6，T7，T8，T9，T10单元中并输出。

```
DATA SEGMENT
    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
DATA ENDS
STACK SEGMENT
          DW  10 DUP(?)
          TOP LABEL WORD
STACK ENDS
```

```
CODE SEGMENT
        ASSUME CS:CODE,SS:STACK,DS:DATA
MAIN PROC FAR
        MOV    AX,STACK
        MOV    SS,AX
        LEA    SP,TOP
        MOV    AX,DATA
        MOV    DS,AX
        LEA    SI,SCORE
        MOV    CX,10
        CLD
    L1: LODSB
        CMP    AL,60
        JB     L5
        CMP    AL,69
        JA     L2
        INC    S6
        JMP    L6
    L2: CMP    AL,79
        JA     L3
        INC    S7
        JMP    L6
    L3: CMP    AL,89
        JA     L4
        INC    S8
        JMP    L6
    L4: CMP    AL,99
        JA     L5
        INC    S9
        JMP    L6
    L5: INC    S10
    L6: LOOP   L1
        MOV    AH,4CH
        INT    21H
MAIN ENDP
CODE ENDS
     END MAIN
```

试编写一汇编语言程序，要求从键盘接收一个 4 位的十六进制数，并在终端上显示与它等值的二进制数和八进制数。

```
code segment
          assume cs:code
main proc far
    start:
          mov    cx,4          ;四个数字循环四次 loop指令循环次数放在cx中，cx==0时顺序向下
执行
          mov    bx,0          ;用bx来存放四位十六进制数，先清零
    ;执行输入过程
          mov    dh,0
    input:
          mov    ah,01h        ;读入字符，放在al中
          int    21h
          mov    ah,0h
```

```
    ;判断是否是字母
            cmp     al,39h
            jbe     L2              ;如果从<=9说明符合0~9之间，不用减07h
            sub     al,07h
    L2:
            sub     al,30h
            add     bx,ax
    ;判断是否是最后一位输入
            cmp     cx,1
            jz      L3              ;如果==0说明已经是最后一位，不用*16
            mov     ax,bx
            mov     dx,16           ;左移一位*16 字乘字
            mul     dx
            mov     bx,ax
    L3:
            loop    input
            mov     dl,'='
            mov     ah,02h
            int     21h
    ;输出,十六进制转二进制一位4个数
            mov     cx,16
    output:                         ;每次左移输出一位
            rol     bx,1
            mov     dl,bl
            and     dl,01h
            add     dl,30h
            mov     ah,02h
            int     21h
            loop    output
            mov     ax,4c00h
            int     21h
main endp
code ends
end start
```