学院：**信息学院**　学期：**2022 年秋季学期**　课程名称：**汇编语言程序设计**　指导教师：**周丽华**

专业：**计算机科学与技术**　年级：**2020**　姓名：**胡诚皓**　学号：**20201060330**

实验四 字符处理程序设计

一、实验目的

1. 学习字符处理的相关指令，加深对这些指令的理解和运用。
2. 掌握字符处理程序设计、编写及调试方法。
3. 掌握在程序设计中合理利用字符串的尾符。

二、实验内容

1. 复习教材中与字符处理指令相关的内容。

2. 已知字符串“aBCEFghi15OXyZ”以 0DH 作为结束标志，编写程序 1：从头搜索字符串的结束标志，统计搜索的字符个数，并画出流程图。

3. 编写程序 2：从键盘读入一个小写字母，输出字母表中倒数与该字母序号相同的那个字母，并画出流程图。例如输入首字母 a，则输出最后一个字母 z，输入第 4 个字母 d 则输出倒数第 4 个字母 w。

4. 分析程序的结果，并准备好上机调试。

5. 修改程序 2 实现从键盘读入一个大写字母，输出字母表中倒数与该字母序号相同的那个大写字母；从键盘读入一个小写字母，输出字母表中倒数与该字母序号相同的那个大写字母

三、实验结果（截图）

1. 搜索字符个数

使用循环可以简单地实现这个功能，本程序中以 SI 为循环变量，最后将结果存在 CX 中。从下图中可见 CX 的值为 14，与题目中给出的“aBCEFghi15OXyZ”长度为 14 一致。

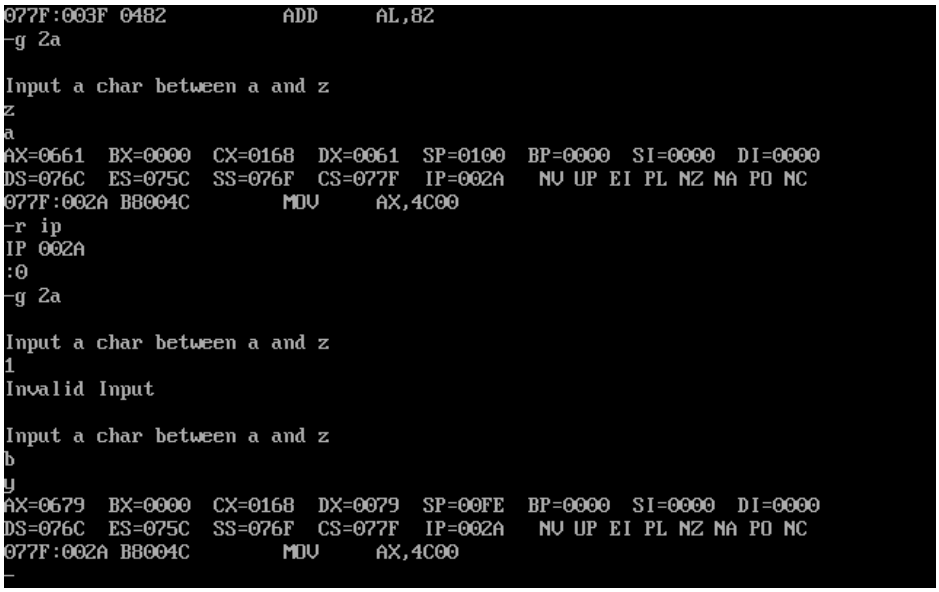
```
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.


Y:\>DEBUG PROJECT.EXE
-u
077D:0000 B86C07        MOV     AX,076C
077D:0003 8ED8          MOV     DS,AX
077D:0005 BE0000        MOV     SI,0000
077D:0008 8AB40000      MOV     AL,[SI+0000]
077D:000C 3C0D          CMP     AL,0D
077D:000E 7403          JZ      0013
077D:0010 46            INC     SI
077D:0011 EBF5          JMP     0008
077D:0013 8BCE          MOV     CX,SI
077D:0015 B8004C        MOV     AX,4C00
077D:0018 CD21          INT     21
077D:001A 16            PUSH    SS
077D:001B 3C21          CMP     AL,21
077D:001D 8987BE22      MOV     [BX+22BE],AX
-g 15

AX=070D  BX=0000  CX=000E  DX=0000  SP=0100  BP=0000  SI=000E  DI=0000
DS=076C  ES=075C  SS=076D  CS=077D  IP=0015   NV UP EI PL ZR NA PE NC
077D:0015 B8004C        MOV     AX,4C00
-_
```

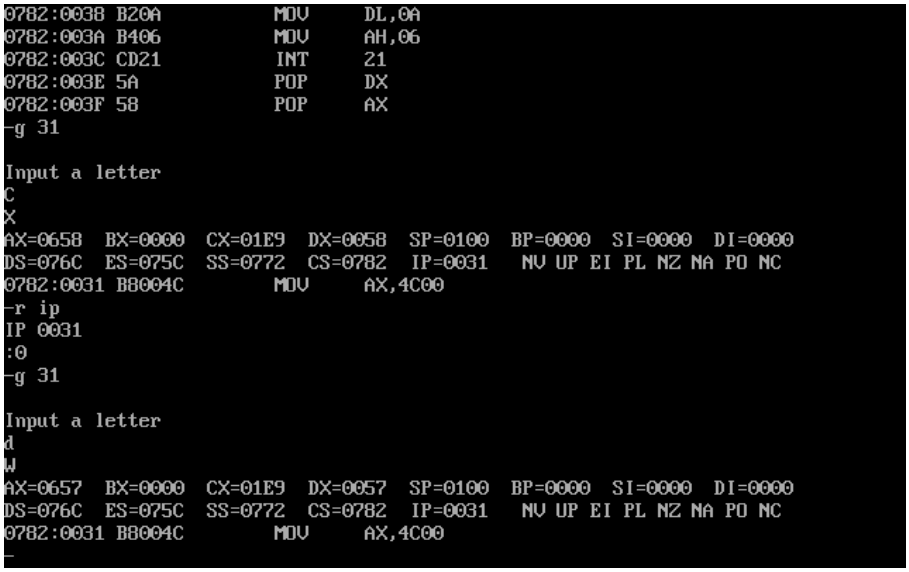
2. 字母倒数

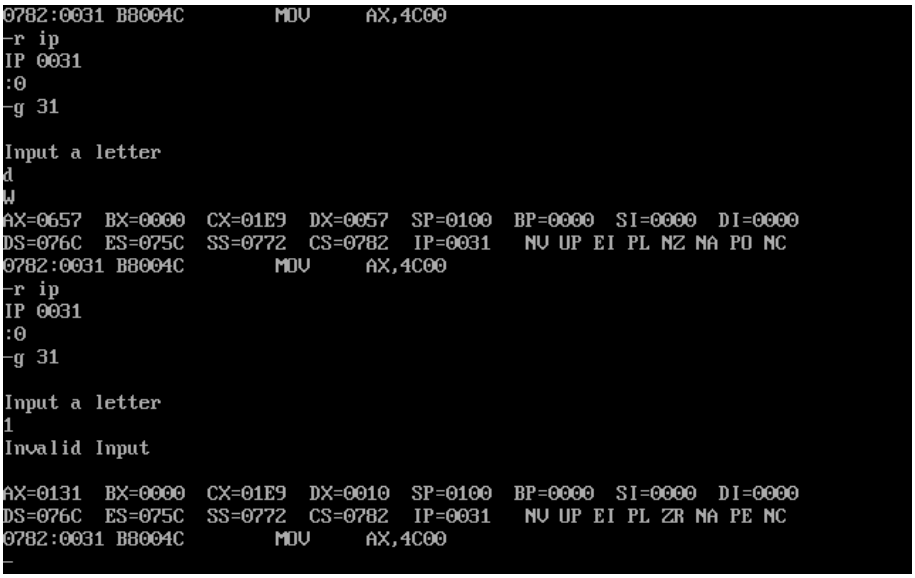
先求得输入小写字母相对于‘a’的偏移量，再在‘z’的 ASCII 码中减去即可，最后输出即可。从下图可见，对于不在范围内的输出做出了正确的处理，输入 z 确实得到了 a，输入 b 确实得到了 y。



3. 字母转大写倒数

计算方法与上题一致，在验证输入的有效性上，为了简便使用了枚举法。另外，若输入无效，显示提示信息后直接退出程序。本题使用了多个子过程进行处理。下面第一张截图中，输入 C、d，输出的分别为 X、W，结果正确。下面第二张截图中，输入了字符 1，未通过输入有效性检验，程序直接退出。





四、实验报告要求（习题）

1. 程序 1

程序中，AL 暂存字节、SI 作为循环变量、CX 中保存最后的结果。

```
DATA SEGMENT
    ; 定义好要检查的字符串
```

```
    STRING DB 'aBCEFghi15OXyZ', 0DH
DATA ENDS

STACK SEGMENT STACK
    ; 堆栈段清零
    DW 128 DUP(0)
STACK ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA, SS:STACK
START:
    ; 定义数据段
    mov ax, DATA
    mov ds, ax
    ; si 作为循环变量，先清零
    mov si, 0
EXAM:  ; 循环体，用于查看是否到了字符串末
    ; 一个个字符读入到 AL
    mov al, STRING[si]
    ; 检查是否为 0DH
    cmp al, 0DH
    ; 若为 0DH，直接跳转到退出段的代码
    jz EXIT
    ; si++
    inc si
    ; 进行循环
    jmp EXAM
EXIT:  ; 退出段
    ; 将结果保存到 CX 中
    mov cx, si
    ; 程序返回的功能码
    MOV AX, 4C00H
    INT 21H

CODE ENDS

END START
```

2. 程序 2

此程序中有多个 Label，在 INPUT 段中，AL 暂存输入的内容；在 CALCU 段中，AL 暂存输入字符相对于‘a’的偏移量、DL 保存最后要输出的内容。

```
DATA SEGMENT
    ERROR_MSG DB 10,'Invalid Input',10,'$'
    INPUT_MSG DB 10,'Input a char between a and z',10,'$'
DATA ENDS

STACK SEGMENT STACK
    DW 128 DUP(0)
STACK ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA, SS:STACK
START:
    mov ax, DATA
    mov ds, ax

INPUT: ; 输入读取段代码
    ; 显示输入提示信息
    mov ah, 9
    mov dx, OFFSET INPUT_MSG
    int 21h
    ; 从键盘读取一个字符作为输入
    mov ah, 1
    int 21h

    ; 由于 AL 中存了用户输入的字符，入栈进行保护
    push ax

    ; 检查输入是否在 a-z 范围内
    cmp al, 'a'
    jb ERROR_OUTPUT
    cmp al, 'z'
    ja ERROR_OUTPUT

    ; 输出个换行符
```

```
    mov dl, 10
    mov ah, 6
    int 21h

CALCU:  ; 计算转换字母代码
    ; 将 AX 弹出，重新得到用户输入的字符
    pop ax
    ; 计算结果保持在 DL 中
    mov dl, 'z'
    ; 偏移量存在 AL
    sub al, 'a'
    ; 倒着从 z 减回来
    sub dl, al
    ; 输出（DL）
    mov ah, 6
    int 21H
EXIT:  ; 退出代码
    MOV AX, 4C00H
    INT 21H

ERROR_OUTPUT: ; 输入非法提示信息输出代码
    ; 显示错误提示信息
    mov ah, 9
    mov dx, OFFSET ERROR_MSG
    int 21h
    ; 输入非法，要求用户重新输入
    jmp INPUT

CODE ENDS

END START
```

3. 对程序 2 的修改

本程序使用多个子过程实现全部功能，各子过程的具体功能在代码注释中均有体现。

```
DATA SEGMENT
    ERROR_MSG DB 10,'Invalid Input',10,'$'
```

```
    INPUT_MSG DB 10,'Input a letter',10,'$'
    LETTERS DB
'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'
    INPUT_CHAR DB 0;
    IS_CHAR DB 0;
DATA ENDS

STACK SEGMENT STACK
    DW 128 DUP(0)
STACK ENDS

CODE SEGMENT

main PROC FAR  ; 主函数
    ASSUME CS:CODE, DS:DATA, SS:STACK

START:
    ; 数据段定义
    mov ax, DATA
    mov ds, ax
    ; 显示输入提示信息
    mov ah, 9
    mov dx, OFFSET INPUT_MSG
    int 21h
    ; 读入一个字符
    mov ah, 1
    int 21h
    mov INPUT_CHAR ,al
    ; 调用 judge 判断输入字符是否合法
    call judge
    ; 不为大小写字母直接退出
    cmp IS_CHAR, 0
    je OUTPUT_ERROR
    ; 调用 calcu 进行转换
    call calcu
    ; 换行显示结果并正常退出
    call clrf
```

```
    ; 此时 INPUT_CHAR 中的内容已经转换好了，显示即可
    mov dl, INPUT_CHAR
    mov ah, 6
    int 21h
    jmp EXIT2
OUTPUT_ERROR: ; 显示错误信息
    call outerror
EXIT2:  ; 主函数的退出位置
    mov ax, 4C00H
    int 21h
main ENDP

clrf PROC NEAR  ; 此过程用于显示换行
    ; 保护此过程中要使用的 ax 与 dx
    push ax
    push dx

    ; 输出个换行符（ASCII 为 10）
    mov dl, 10
    mov ah, 6
    int 21h

    ; 还原现场
    pop dx
    pop ax
    ret
clrf ENDP

calcu PROC NEAR
    ; 保护 ax 寄存器
    push ax
    ; 使用内存单元进行传参
    mov al, INPUT_CHAR
    cmp al, 'a'
    ; 大于等于 a 的就是小写，直接-97；大写字母先加 32，+32-97=-65，
    jae LOWER
    add al, 32
```

```
LOWER:  ; 小写则直接跳转此处
    sub al, 97  ; 此时 al 中为偏移量
    mov INPUT_CHAR, 'Z'
    sub INPUT_CHAR, al  ; 此时 INPUT_CHAR 中为转换后的字符

    pop ax
    ret
calcu ENDP


outerror PROC NEAR  ; 用于显示错误信息
    ; 保护子过程中要使用到的寄存器
    push ax
    push dx

    ; 显示输入无效的提示信息
    mov ah, 9
    mov dx, OFFSET ERROR_MSG
    int 21h

    pop dx
    pop ax
    ret
outerror ENDP


judge PROC NEAR  ; 用于判断输入字符是否符合要求
    ; 保护子过程中要用到的寄存器
    push cx
    push si
    mov si, 0
    ; CL 作为判断条件
    mov cl, 0
    ; CH 存要判断的字符
    mov ch, INPUT_CHAR
REPEAT1:
    ; 最多 52 次判断循环
```

```
    cmp si, 52
    je EXIT1
    ; 与 LETTERS 中的字符一个个比较看看是否有相同的
    cmp ch, LETTERS[si]
    jne TMP1; 不等则跳过“cl=true”
    mov cl, 1
    jmp EXIT1
TMP1: ; 推进循环
    inc si
    jmp REPEAT1
EXIT1:
    ; 使用内存单元传参，IS_CHAR 保存判断的结果
    ; 1 表示是字母、0 表示不是
    mov IS_CHAR ,cl
    pop si
    pop cx
    ret
judge ENDP

CODE ENDS

END START
```

4. 如何用 debug 查看程序运行结果

下面叙述本次报告使用 debug 查看运行结果的过程

① 使用 debug [exe 程序名]，进入程序的调试

② 使用 g [断点]运行程序至断点处，断点可以使用相对地址指定，需要注意的是，指定的断点位置的语句并不会运行

③ 使用 r 命令观察寄存器，计算过程中把结果存在了 AX 中

④ 使用 d [内存地址]查看指定内存单元以后一段的内容，该命令中的地址也可以使用相对地址进行指定，默认查看的是 DS:指定偏移地址的位置

5. 遇到的问题及如何解决

相较于之前几个实验，本次实验要求实现的功能更为复杂，需要进行输入输出以及中间对字符的处理。在编写代码的过程中，由于没有高级程序设计语言中那样的分支语句，实现条件分支较为麻烦。在程序 2 中，使用 Label 标记各个功能块，各块之间进行跳转；但在修改程序 2 改造其功能时，发现如此操作还是较

为不便，需要完全自己管理 IP 寄存器的内容。因此，使用了多个子过程完成所需功能，CALL 和 RET 指令会帮我们实现 IP 指针内容的管理。

五、个人体会与总结

此次实验尝试使用子过程解决问题，在调试的过程中，发现堆栈指针对子过程的跳转影响非常大，直接关系到了入栈出栈保护通用寄存器。另外，在低级语言中，仍需要注意对程序代码功能的分解，利用好模块化思想编写代码，既便于调试，又便于编写。