电子科技大学信息与软件工程学院

实 验 报 告

学　　号 2018091618008

姓　　名 袁昊男

（实验）课程名称 汇编语言程序设计

理论教师 赵　洋

实验教师 赵　洋

电子科技大学教务处制表

电 子 科 技 大 学

实　验　报　告

学生姓名：袁昊男　　学号：2018091618008　　指导教师：赵洋

实验地点：在线实验　　实验时间：2020.06.15

一、实验室名称：信息与软件工程学院实验中心

二、实验名称：编写子程序实现字符串转换

三、实验学时：2 学时

四、实验原理：

1、 相当于将字符串中的每个字符在闭区间['a','z']内进行比较，依据比较结果进行转换操作；

2、 在调用时注意相关参数的传递。

五、实验目的：

1、 掌握条件转移指令的使用；

2、 掌握利用寄存器传递参数的使用。

六、实验内容：

编写一个子程序，将包含任意字符，以 0 结尾的字符串中的小写字母转变为大写字母。

```
assume cs:codesg

datasg segment
  db "Beginner's All-purpose Symbolic Instruction Code.",0
datasg ends

codesg segment
  begin:   mov ax,datasg
           mov ds,ax
           mov si,0
           call letterc

           mov ax,4c00h
```

```
            int 21h

letterc: ……              ;子程序部分[开始]

            ret           ;子程序部分[结束]

codesg ends
end begin
```

七、实验器材（设备、元器件）：

PC 微机一台

八、实验步骤：

1、 编辑源程序，建立一个以后缀.ASM 的文件。
2、 汇编源程序，检查程序有否错误，有错时回到编辑状态，修改程序中错误行，无错时继续第 3 步。
3、 连接目标程序，产生可执行程序。
4、 用 DEBUG 程序调试可执行程序，记录数据段的内容。

九、实验数据及结果分析

1、 实验思路

(1) 总体思路

在主程序中首先调用实验三编写的 show_str 子程序，将 data 段中未经转换的字符串按既定格式在屏幕中显示；再调用本实验编写的 letterc 子程序，实现将 data 段中字符串的小写字母转换为大写字母并保存在 data 段中原位置；转换工作完成后再一次调用 show_str 子程序，将 data 段中转换完成后的字符串按既定格式在屏幕中显示。在调用子程序与返回子程序时特别注意相关寄存器的保护。

(2) letterc 子程序

letterc 子程序依次读取 data 段中的字符串单个字符。使用 cmp 指令与 ja、jb 配合判断所读取字符是否小写字母。若当前字符为小写字母，则使用 and 指令将其转换为大写字母后写回 data 段中原位置；若当前字符为大写字母，则直接写回 data 段中原位置。

在每次循环处理之前，首先判断是否遇到“0”，如当前字符为“0”则退出循环，返回主程序；每次循环后指针后移。

2、 流程图

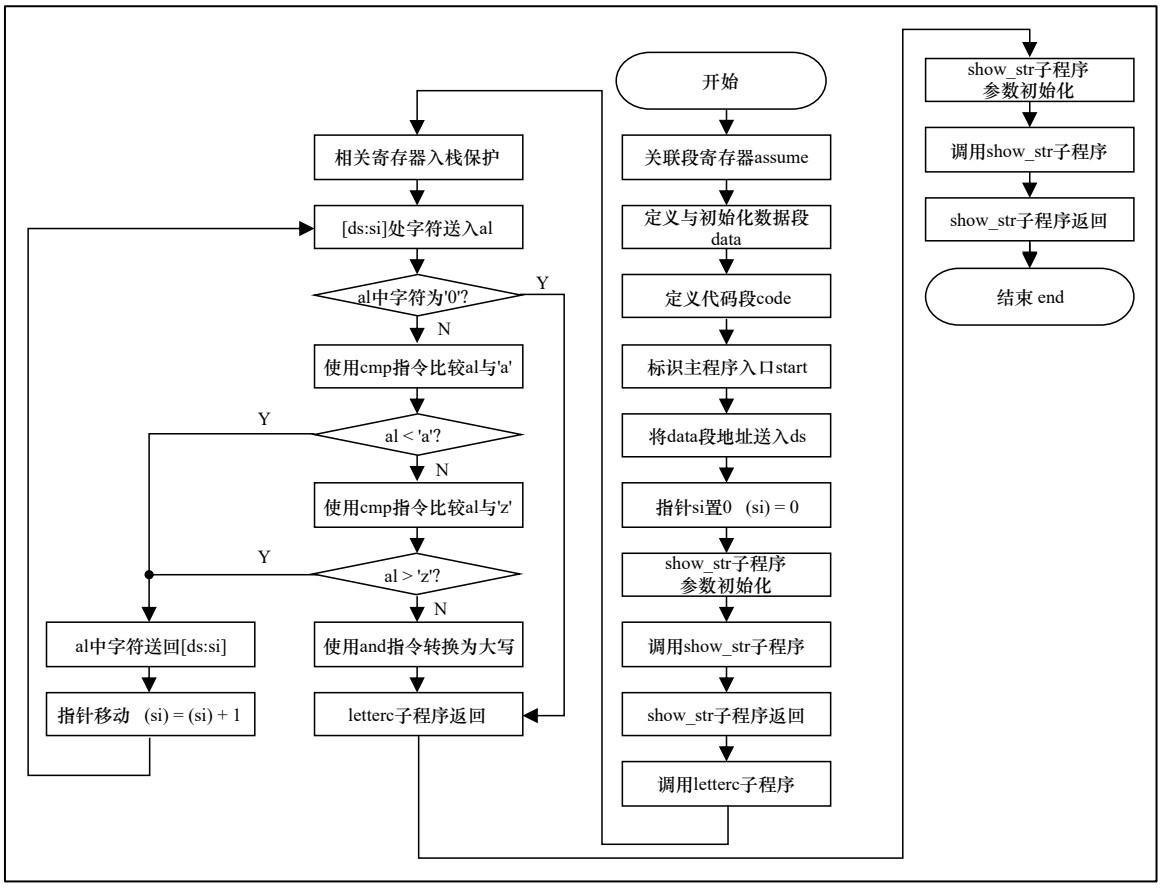

图 1 实验四流程图

3、 实验截图

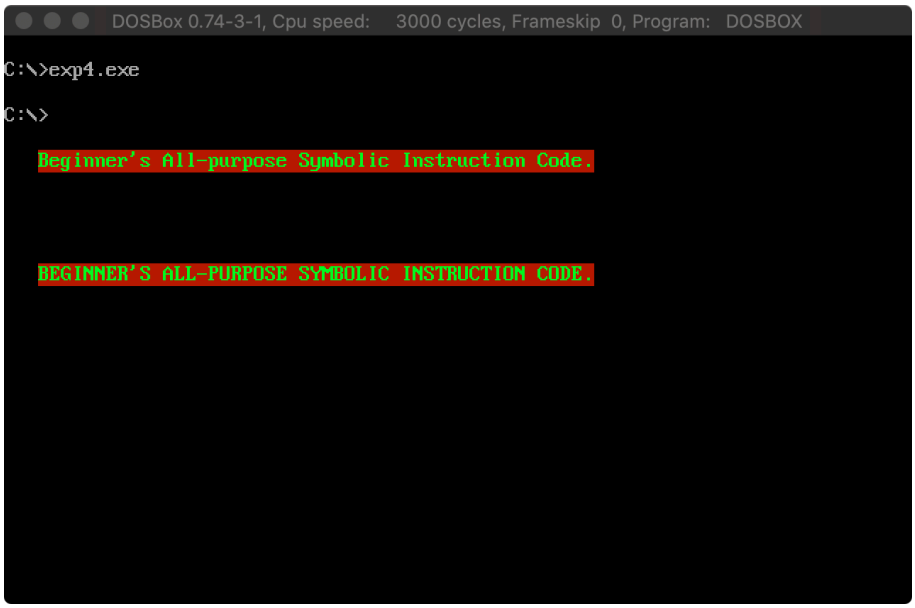

图 2 运行结果

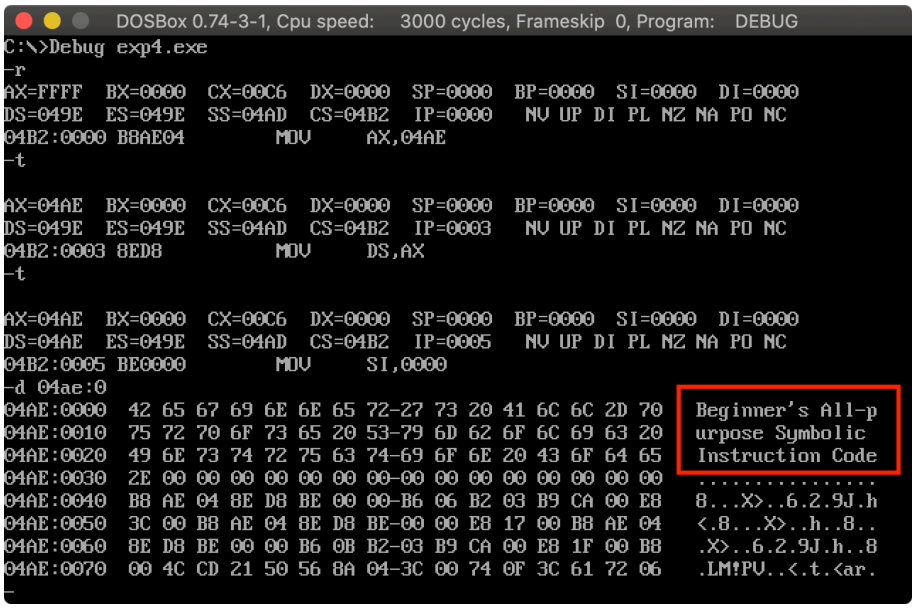

图 3　letterc 子程序执行前 data 段中数据

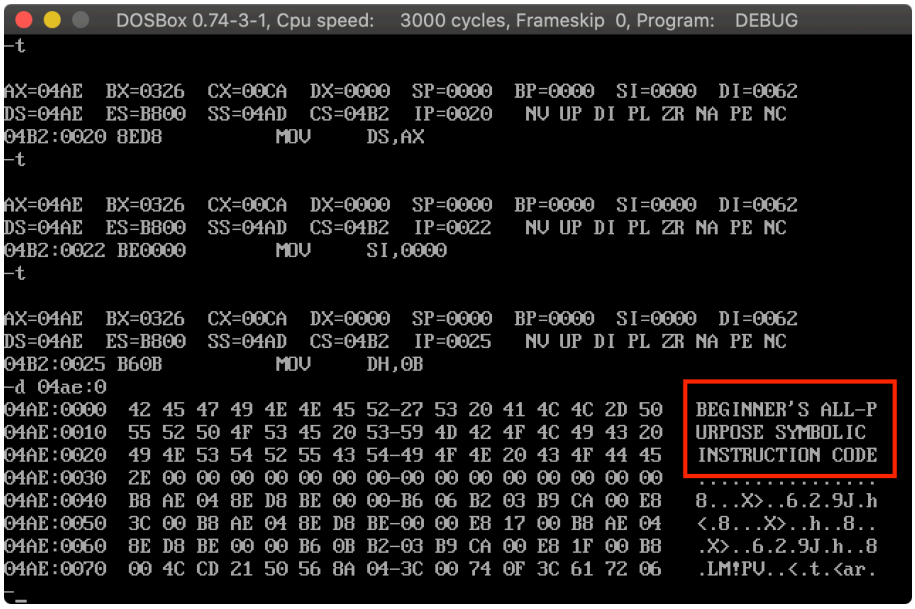

图 4　letterc 子程序执行后 data 段中数据

4、 结果分析

程序能正确编译、连接，运行后可以按实验要求在屏幕的第 6 行 3 列输出红底高亮闪烁绿色字符串'Beginner's All-purpose Symbolic Instruction Code.'；在屏幕的第 11 行 3 列输出红底高亮闪烁绿色字符串'BEGINNER'S ALL-PURPOSE SYMBOLIC INSTRUCTION CODE.'；经跟踪、调试，使用 D 命令查看执行 letterc 子程序前 data 段中数据（如图 3 所示），可以发现待显示字符串 ASCII 码正确存储且字符串以 0 作为结尾符号；使用 D 命令查看执行 letterc 子程序后 data 段中数据（如图 4 所示），可以发现之前字符串中所有小写字母已全部转换为大写字母，且其 ASCII 码正确存储且。

十、实验结论

代码经 masm 编译、link 连接成功，运行后可以按实验要求转换字符串中的大小写字母并在屏幕中输出转换前后的字符串。经 Debug 跟踪、调试，data 段中数据按预设正确存储。

结合本实验编码过程，可以得出：可以通过向子程序传递参数、调用子程序、获得子程序返回值来实现功能，体现了汇编语言编程中的模块化设计。在实际编程中遇到的现实问题往往比较复杂，对现实问题进行层次分析，进而将其分解为相互联系的子问题，再通过编写子程序进行实现，降低了代码的耦合度，相互联系、功能独立的子程序模块能在一定程度上降低程序出错的可能性，提高了代码的可维护性。

十一、总结及心得体会

1、 汇编语言中可以使用 jcxz 与 cmp 等指令实现“判断并转移”操作。如 jcxz 可以检测 cx 的值，若(cx)＝0，则转移；cmp 指令可以与配合 je（等于则转移）、jne（不等于则转移）、jb（低于则转移）、jnb（不低于则转移）、ja（高于则转移）、jna（不高于则转移）来实现对标志寄存器相关标志位的检测与转移操作，配合使用时其语法功能相当于高级语言中的 IF 语句。

2、 call 与 ret 指令共同支持汇编语言编程中的模块化设计：call 指令实现程序的转移、ret 指令实现程序的返回。利用 call 与 ret 指令可以用简便的方法，实现多个相互联系、功能独立的子程序来解决一个复杂的问题。

3、 在主程序与子程序进行切换时，由于一些寄存器起到参数传递的作用，而在主程序与子程序中，寄存器的实际意义不同，因此特别注意对发生冲突的寄存器入栈保护与出栈恢复。可单独设置代码段实现寄存器的入栈保护与出栈恢复，注意入栈与出栈时寄存器的顺序恰好相反。

十二、对本实验过程及方法、手段的改进建议

本实验逻辑较为简单，且可以借用实验三中编写好的 show_str 子程序实现字符串的打印输出。可以规定字符串的显示位置与字符属性，加深对相关内容的理解掌握。

报告评分：

指导教师签字：

附录：实验程序源码

```
assume cs:code
data segment
    db "Beginner's All-purpose Symbolic Instruction Code.",0
data ends

code segment

start:
    mov ax,data
    mov ds,ax
    mov si,0

    mov dh,6              ; 行数
    mov dl,3              ; 列数
    mov cx,0cah           ; 字符属性
    call show_str

    mov ax,data
    mov ds,ax
    mov si,0
    call letterc

    mov ax,data
    mov ds,ax
    mov si,0
    mov dh,11             ; 行数
    mov dl,3              ; 列数
    mov cx,0cah           ; 字符属性
    call show_str


    mov ax,4c00h
    int 21h

;名称：letterc
;功能：将以 0 结尾的字符串中的小写字母转变为大写字母
;参数：ds:si 指向字符串首地址
letterc:
    push ax               ; 将相关寄存器先压栈
    push si

comp:
    mov al,[si]           ; 判断是否读到字符串末尾
    cmp al,0
    je exit               ; 等于则转移
    cmp al,'a'
    jb next               ; 低于则转移
    cmp al,'z'
    ja next               ; 高于则转移
    and al,11011111b      ; 变大写

next:
    mov [si],al           ; 大写字母直接复制
    inc si
    jmp short comp
```

```
exit:
    pop si
    pop ax
    ret

show_str:
    push ax
    push ds
    push dx
    push cx
    push si                ; 保护子程序寄存器中用到的寄存器

    mov di,0               ; 显示缓存区中的偏移量
    mov bl,dh
    dec bl                 ; 第 bl 行之前有 bl-1 个完整的行
    mov al,160
    mul bl                 ; 每行 160 字节，用行数乘偏移量得到目标行的偏移量
    mov bx,ax              ; 乘积存储在 ax 中，送入 bx 中
    mov al,2               ; 列的偏移量为 2(第 1 字节为数值，第 2 字节为属性)
    mul dl                 ; 与行偏移量同理
    add bl,al              ; 将列偏移量与行偏移量相加，得到指定位置的偏移量。

    mov ax,0b800h
    mov es,ax              ; 指定显示缓存区的内存位置

    mov ax,cx              ; 现将字符属性保存
s:
    mov ch,0
    mov cl,ds:[si]         ; 首先将当前指向字符串的某个字符存入 cx 中
    jcxz okk               ; 如果 cx 为 0，则转移到 okk 标号执行相应代码
    mov es:[bx+di],cl      ; 将字符传入低地址
    mov es:[bx+di+1],ax    ; 将颜色传入高地址
    add di,2               ; 列偏移量为 2
    inc si                 ; 字符串的偏移量为 1
    loop s                 ; 不为 0，继续复制

okk:
    pop dx
    pop cx
    pop si
    pop ds
    pop ax                 ; 还原寄存器变量
    ret                    ; 结束子程序调用


code ends
end start
```