汇编语言与组成原理实验报告

学号：E21714049 姓名：梅世祺

一、实验名称 实验三算术及位串处理实验

二、实验目的

掌握多位数的算术运算、移位操作、字符串操作等程序的设计，统计学习使用分支的与循环等基本编程方法，熟练使用 Debug。

三、实验内容

在数据段定义缓冲区，从键盘接收两串字符到两个缓冲区，将第二串中与第一串字符不一致的字符显示在屏幕。

再从键盘输入一位字符到 BX，查找第一串字符串中有几个相同的字符，并将次数显示在屏幕上。

四、实验过程

首先在数据段开辟两个字符串缓冲区（origin_buffer 和 target_buffer ），用于接收即将输入的两个字符串:

```
; Written by lolimay <lolimay@lolimay.cn>
; Last updated: 2019.04.16

    data segment
    origin_buffer db 19 ; need add an extra byte for tab Enter
    origin_length db ? ; in other words, Enter also takes up one character.
    origin_content db 18 dup(?)
    target_buffer db 19
    target_length db ?
    target_content db 18 dup(?)
    data ends
    code segment
    assume cs:code, ds:data, es:data
    ; main function
    ; I specify it as the program's entrance manually here.
    main proc
    mov ax, data
    mov ds, ax
```

```
mov es, ax ; es=ds
lea dx, origin_buffer
call read_buffer
call println
lea dx, target_buffer
call read_buffer
call println
lea si, origin_content
lea di, target_content
mov cx, 12h ; 18
cld
repe cmpsb
jz eql
mov dl, 'N'
call printc
mov dl, ' '
call printc
mov dl, 12h ; 18
sub dl, cl ; dl = 19 - (cl + 1) = 18 - cl
xor bx, bx
mov bl, dl
dec bl
add dl, 30h ; convert it to ascii code
call printc
mov dl, ' '
call printc
mov dl, origin_content[bx]
call printc
mov dl, ' '
call printc
mov dl, target_content[bx]
call printc
jmp dsp
eql:
mov dl, 'Y'
call printc
dsp: ; scan start
xor bl, bl ; use bx to store times
call println
call getc ; store it into al
call println
lea di, origin_content
mov cx, 12h ; 18
cld
```

字符串比较核心逻辑
通过字符串操作指令
cmpsb 来对比两个字
符串是否相同

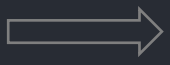
两个字符串中若有不同之处，
则第一处不同的字符位置是
dl = 18 - cl

分别打印两个不同的字符：
origin_content[bx]
和
target_content[bx]

字符串比较核心逻辑
通过字符串操作指令
cmpsb 来对比两个字
符串是否相同

字符个数统计核心逻辑

```
lop:
repne scasb
jz yes
jmp ext
yes:
inc bl
jmp lop
ext:
mov dl, bl
add dl, 30h
call printc
call exit ; exit the program,
```



字符个数统计核心逻辑：
主要通过 scasb 指令完成字符串的扫描

字符个数统计核心逻辑：
主要通过 scasb 指令完成字符串的扫描

下面是相应的子程序的实现：

read_buffer: 读取缓冲区
getc: 从标准输入流获取一个字符
printc: 将 dl 的值作为字符打印出来
println: 打印换行
exit: 退出程序

```
; read buffer from the standard input stream.
; It's a wrapper for 0a function of INT21H.
; But it can protect ah register's value from overwriting.
read_buffer proc
push ax
mov ah, 0ah
int 21h
pop ax
ret
read_buffer endp
; get a char, which stores into *al* register.
; actually, it's a wrapper for 01 function of INT 21H.
getc proc
mov ah, 1
int 21h
ret
getc endp
; print a char, which need parameter from *dl* register.
; actually, it's a wrapper for 02 function of INT 21H.
; But it can protect ah register's value from overwriting.
printc proc
push ax
mov ah, 2
int 21h
```

```
    pop ax
    ret
    printc endp
    ; print '\n' (aka. CRLF)
    println proc
    push dx
    mov dl, 0dh ; CR
    call printc ; because printc protects ax register aready
    mov dl, 0ah ; So here we don't to protect it again.
    call printc ; LF
    pop dx
    ret
    println endp
    ; exit program
    ; In fact, it's a wrapper for 4c function of INT 21H.
    ; But it can protect ah register's value from overwriting.
    exit proc
    push ax
    mov ah, 4ch
    int 21h
    pop ax
    ret
    exit endp
    code ends
    end main
```

汇编源码

```
-q

C:\WORKSHOP\4.16>debug MAIN.EXE
-g
computer software
computer software
Y
t
2
Program terminated normally
-g
computer software
comkuter software
N 4 p k
c
1
C:\WORKSHOP\4.16>debug MAIN.EXE
-g
abcdefefe
abddefefe
N 3 c d
e
3
Program terminated normally
```

实验结果

五、实验小结

通过本次实验，我们对字符串操作指令 cmpsb、scasb、repe 和 repne 有了更加深入的理解。在不借助 loop 循环的情况下，仅依靠 repe 和 repne 来对字符串进行扫描和比较，锻炼了我们对这些指令使用能力。同时，这也为我们以后开发更加复杂的分支、循环等程序打下了坚实的基础。

在本次实验中，我们需要注意的是这里使用了 mov ds, ax; mov es, ax (即 es=ds) 简化了字符串操作的过程，如果没有上述显式的指定的话，需要注意 di 寄存器对应的段寄存器是 es, si 寄存器对应的段寄存器是 ds，不能弄混。