实践课程项目 四

程序中定义三个段如下：

```
a segment
        dd 0123h,0456h,0789h,0abch
a ends

b segment
        dw 10h,20h,10h,20h
b ends

c segment
        dw 0,0,0,0
c ends
```

编程，把 a 段中的双字数据当做被除数，b 段中的字型数据当除数。相除后，商放在 c 段的对应位置。调试通过后，把源程序写在下面。

二、已知程序的数据段如下：

```
data segment
        db 'abCDefgHijkLmnOpqRstuVwXYZ'
data ends
```

按照下面步骤用两种方法完成数据段中内容的大小写互换（大写转换为小写，小写转换为大写）。

方法一（使用条件跳转语句）：

1.完成程序框架的书写。

```
assume cs:code, ds:data
code segment
start:
        mov ax,4c00h
        int 21h
code ends
end start
```

2.把 ds:bx 指向 data 段中字符串的首地址。

```
        mov ax,___
        mov ds,ax               ;将 ds 指向 data 段
        mov bx,___              ;将 ds: bx 指向 data 段中字符串的首地址
```

3.data 段共 26 个字母，对一个字母做处理后，ds:bx 再指向下一个字母。相应的代码如下：

```
        mov cx,26
s:      ;此处需要处理 ds:bx 中的字母
        inc  bx
```

```
        loop s
```

4.判断字母是大写还是小写的语句是：

```
        mov cl,[bx]
        and cl,00100000b
        mov ch,0
        jcxz s1
        ____________________    ;cx 不为 0，则[bx]中的字母为小写,需要变为大写
        jmp short ok

s1:     ____________________    ;cx 为 0，则[bx]中的字母为大写，需要变为小写
ok:     ;此处根据以上分析自己填写相关代码
        ;返回指令
```

5.根据上面的分析，将完整代码写出。（注意 cx 的保存与恢复。）

6.编译、连接、运行，用 debug 查看程序结果。

方法二(不使用条件跳转语句)：

1.用逻辑运算进行大小写的互换，完整 code 段定义如下：（若要程序正确执行，需要添加 data 段内容和相关的伪指令。）

```
code segment
start:  mov ax,data
        mov ds,ax
        mov bx,0

        mov cx,26
s:      add byte ptr [bx],00100000b     ;将需要修改字母的 ASCII 码加上 20h
        and byte ptr [bx],01111111b     ;将相加后的数值最高位 置为 0
        or byte ptr [bx],01000000b      ;将相加后的数值次高位 置为 1
        ;这三句指令比较有技巧，注意理解。
        inc bx
        loop s

        mov ax,4c00h
        int 21h

code ends
end start
```

2.编译、连接，运行生成的程序，观察结果。

3.注意分析理解程序标号 s 处的 3 句指令，体会逻辑运算的高效性和抽象性。

三、编程实现：找到当前屏幕中所有显示为“a”的字符，并把它的属性变为闪烁黑底红字。（第九章实验指导中有这部分内容）

完整的程序代码如下：

四、程序中定义数据段如下：

```
data segment
dw 0,0
data ends
```

用目前所学的汇编语句，编程实现算法“(1+2+3+4+...+n)>122 时的第一个 n 是多少?”并把此时的和与数据 n 分别放在上面的数据段的对应位置中。(提示：可以考虑负数的二进制数最高位为 1)。

请写出完整的程序代码：

五、根据要求编程

数据段定义如下：

```
data segment
db ‘welcome to masm!’
data ends
```

1.将数据段中的字符串以写显存的方式打印在屏幕上。

2.令打印在屏幕上的字符串的颜色不停的变化。

提示：

通过耗费 cpu 运行时间实现延时的代码如下：

```
      mov cx,0fffh   ;注意：改变此 cx 值可改变延时的时间长短
  s0:push cx
      mov cx,0ffffh
  s1:sub dx,dx
      loop s1
      pop cx
      loop s0
```