# 作业

## 22920212204392 黄勖

**4.4编写一个程序，把从键盘输入的一个小写字母用大写字母显示出来。**

答：

changealpha:

mov ah, 1 ；从键盘输入，出口l :al 存键值

int 21h

cmp al, 'a' ；判键值是小写字母

jb changealpha

cmp al, 'z'

ja changealpha

sub al,20h ；是小写字母转换为大写字母

mov dl,al

mov ah, 02h ；显示

int 21h

**4.6编制一个程序，把变量 bufX 和 bufY 中较大者存入 bufZ ；若两者相等，则把其中之一存入bufZ中。假设变量存放的是8位无符号数。**

答：

.model small

.stack 256

.data

bufX db ?

bufY db ?

bufZ db ?

.code

.startup

mov al, bufX

mov bl, bufY

cmp al,bl

ja next

mov bufZ, bl

jmp done

next: mov bufZ, al

done: .exit 0

end

**4.23子程序的参数传递有哪些方法，请简单比较。**

答：

有三种方法：寄存器、共享变量（公共存储单元）、堆栈。

1. 用寄存器传递参数是把参数存于约定的寄存器中，这种方法简单易行，经常采用；
2. 用变量传递参数指的是主程序与被调用过程直接用同一个变量名访问传递的参数。如果调用程序与被调用程序在同一个源程序文件中，只要设置好数据段寄存器 DS ，则子程序与主程序访问变量的形式相同，也就是它们共享数据段的变量；若调用程序与被调用程序不在同一个源文件中，必须利用public/extern 进行声明，才能用变量传递参数，利用变量传递参数，过程的通用性比较差，然而，在多个程序段间，尤其在不同程序的模块间，利用全局变量共享数据也是一种常见的参数传递方法；
3. 用堆栈传递参数是主程序将子程序的入口参数压入堆栈，子程序从堆栈中取出参数；子程序将出口压入堆栈，主程序弹出堆栈取得它们。

**4.25什么是子程序的嵌套、递归和重入？**

答：

1. 子程序中又调用子程序就形成子程序嵌套。
2. 子程序中直接或间接调用该子程序本身就形成子程序递归。
3. 子程序的重入是指子程序被中断后又被中断服务程序所调用，能够重入的子程序称为可重入子程序。

**4.28写一个子程序，根据入口参数 AL的值为0或1或2 ，分别实现对大写字母转换成小写、小写转换成大写或大小写字母互换。欲转换的字符串在string 中，用 0 表示结束。**

答：

alphachange proc

push bx ；保护 bx

xor bx, bx ；bx位移量清零

cmp al,0 ；根据入口参数 AL=0/1/2 ，分别处理

jz chan\_0

dec al

jz chan\_1

dec al

jz chan\_2

jmp done

chan\_0: mov al,string[bx] ；实现对大写字母转换成小写

cmp al,0

jz done

cmp al,’A’ ；是大写字母

jb next0

cmp al,’Z’ ；是大写字母

ja next0

add al, 20h ；转换为小写

mov string[bx], al

next0: inc bx ；位移量加1 ，指向下一字母

jmp chan\_0

chan\_1: mov al,string[bx] ；实现对小写字母转换成大写

cmp al,0

jz done

cmp al,’a’ ；是大写字母

jb next1

cmp al,’z’ ；是大写字母

ja next1

sub al, 20h ；转换为大写

mov string[bx], al

next0: inc bx ；位移量加 1 ，指向下一字母

jmp chan\_1

chan\_2: mov al,string[bx] ；实现对大写字母小写字母互换

cmp al,0

jz done

cmp al,’A’ ；是大写字母

jb next2

cmp al,’Z’ ；是大写字母

ja next20

add al, 20h ；转换

jmp next2

next20: cmp al,’a’ ；是小写字母

jb next2

cmp al,’z’ ；是小写字母

ja next2

sub al, 20h ；转换

mov string[bx], al

next2: inc bx ；位移量加1 ，指向下一字母

jmp chan\_2

done: pop bx ；恢复bx

ret

alphachange endp