汇编第三次作业

4.4

```
int 21h
cmp al,'a'
jb done
cmp al,'z'
ja done
sub al,20h
mov dl,al
mov ah,2
int 21
done:   mov ax 4c00h
        int 21h
```

4.6

```
data segment
bufx db 10
bufy db 20
bufz db ?
data ends

stack segment
stack db 100 dup(?)
stack ends

code segment 'code'
assume cs:code,ds:data,ss:stack

start:
mov ax,data
mov ds,ax
mov al,bufx
mov ah,bufy
cmp al,ah
jbe gz
mov bufz,ah
jmp done

gz:
mov bufz,al

done:
mov ax,4c00h
int 21h

code ends
end start
```

4.23

参数传递有四种方法：寄存器参数传递，约定存储单元参数传递，利用 CALL 后续区进行参数传递，利用堆栈进行参数传递。

一、寄存器参数传递

优点是实现简单方便，调用方便，但是寄存器个数有限，且寄存器往往还要存放其他数据，所以只适合用于要传递的参数较少的情况。

程序示例，这里就不举例了，利用到这种参数传递的例子有很多。

例如，利用 ax 传入一个 16 位数，转化为字符串并输出。

二、利用约定存储单元传递参数

这种数据传递方式，数据在内存中，通常在数据段中，相当于全局变量。例如，使用一个字符串变量作为参数，传入函数，输出一个 16 进制数。

三、利用堆栈传递参数

实现方法：主程序在调用子程序之前，将需要传递的参数依次压入堆栈，子程序从堆栈中取入口参数；子程序调用结束之前，将需要返回的参数依次压入堆栈，主程序在堆栈中取出参数。

4.25

子程序的嵌套是指在一个子程序中调用另一个子程序，而递归是指一个子程序直接或间接地调用自身。重入是指在一个子程序执行期间，另一个子程序被调用并开始执行，而在该子程序执行完毕后，原来的子程序可以恢复执行。这三个概念都是指在程序中调用其他子程序的方式，但它们的区别在于调用的方式和执行的顺序。

4.28

```
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
```