1.求2^3

```
assume cs:code

code segment
    mov ax,2
    add ax,ax
    add ax,ax

    mov ax,4c00h
    int 21h
code ends
end
```

2.loop 标号

- 每次执行到loop指令的时候，都去看一下cx寄存器中的值是否为0 如果为0，就向下继续执行其他的命令，如果不是，就返回到标号的位置

```
assume cs:code

code segment
    mov ax,2
    mov cx,11
    s:add ax,ax
    loop s
    mov ax,4c00h
    int 21h
code ends
end
```

3.把dw 1h,2h,3h,4h定义的数据，逆序存放

- 思路解决 想把数据存放到栈中，然后在利用栈的特性先进后出，把数据弹出 要给栈段设置存储空间（就直接在代码段设置一个空的占用空间）

```
assume cs:code

code segment
    dw 1h,2h,3h,4h
    dw 8 dup(0)
    start:
        mov bx,0
        mov cx,4
        mov ax,cs
        mov ss,ax
        mov sp,20h

```

```
13
14      s:push cs:[bx] ;把数据压栈
15      add bx,2  ;因为操作的是字，所以需要加2
16      loop s
17
18      mov cx,4
19      mov bx,0
20      s2:pop cs:[bx]
21      add bx,2
22      loop s2
23
24
25
26      mov ax,4c00h
27      int 21h
28
29  code ends
30  end start
```

4.把数据段的adcd数据转换成大写的

```
1   assume cs:code,ds:data
2   data segment
3     db 'abcd'
4   data ends
5
6   code segment
7   start:mov ax,data
8       mov ds,ax    ;给数据段赋值
9       mov cx,5
10      mov bx,0
11      s:mov al,[bx]   ;把数据段的数据取出来
12      and al,11011111B  ;al和11011111B做与运算，运算的结果赋值给al
13      mov [bx],al
14      inc bx
15      loop s
16      mov ax,4c00h
17      int 21h
18  code ends
19  end star
```

5.字母转换成大写的

- 实现中出现的问题 二重循环都需要cx作为计数器，但是内部循环改变cx后，外重循环就乱套了 解决 在栈中保存外层循环的数据，需要的时候栈取出

```
1   assume cs:code,ss:stack
2
3   stack segment
4    db 0,0,0,0,0,0,0,0
```

```
stack ends

code segment
    db 'abc             '
    db 'def             '
    start:
        mov ax,stack
        mov ss,ax
        mov sp,8
        mov dx,0


    mov cx,2
    s1:push cx ;把cx计数器数据保存到栈中

    mov cx,3
    mov bx,0
    s:mov al,cs:[dx+bx]
    and al,11011111b
    mov cs:[dx+bx],al
    inc bx
    loop s

    add dx,16
    pop cx   ;弹出栈中的数据，赋值给cx
    loop s1

    mov ax,4c00h
    int 21h


code ends
end start
```

6.程序在运行的时候，将s处的指令复制到s0处

```
assume cs:code
code segment
    s:mov,ax
    mov si,offset s
    mov di,offset s0
    mov ax,cs:[si]
    mov cs:[di],ax ;应为cs:[di],对应数据段，所以必须使用ax作为中间数据，赋值
    s0:nop
    nop
code ends
end
```

7.用栈来传递参数

- 把参数放到栈中，需要的时候，利用bp（这个寄存器默认是ss），去除栈中保存的数据 问题 把dw定义的数的3次方，放到dd定义的数据中

```
assume cs:code,ds:data

data segment
 dw 1,2,3,4,5,6,7,8
 dd 0,0,0,0,0,0,0,0
data ends
stack segment
  db 16 dup(0)
stack ends

code segment
        mov si,0
        mov di,16
        mov cx,8  ;需要循环8次

    mov ax,stack
    mov ss,ax
    mov sp,16;初始化栈段

    mov ax,data
    mov ds,ax  ;初始化数据段

    s:push [si] ;参数压栈，等待cube调用这个参数
    call cube









    add si,2   ;定义的数据是字，所以计算一个后加2
    add di,4    ;定义的数据是双字，所以计算一个后加4
    loop s
    cube:
    push bp  ;把bp暂时征用，保存bp之前的值
    mov ax,[sp+4]  ;获取栈中保存的数据，sp默认    ss:[bp],又因为call和push bp，都用了栈。
                    ;所以要从栈中的第4个数据才是传递的参数
    mul bp
    mul bp

    mov [di],dx  ;高位的数据
    mov [di+2],ax  ;低位的数据
    pop bp  ;归还bp
    ret 4

    mov ax,4c00h
    int 21h

code ends
end start
```

8.完成128位数的相加

```
assume cs:code,ds:data

data segment
 dw 1h,2h,3h,4h,5h,6h,7h,8h
 dw 1h,2h,3h,4h,5h,6h,7h,8h
data ends
stack segment
  db 16 dup(0)
stack ends

code segment
        start:
        mov si,0
        mov di,16


    mov ax,data
    mov ds,ax;初始化数据段

    mov ax,stack
    mov ss,stack
    mov sp,16;初始化栈段




    call add128

    mov ax,4c00h
    int 21h

    add128:
    push cx
    push ax
    push si
    push di

    mov cx,8
    sub ax,ax  ;把ax归0，也把CF归0,不能用mov,ax,0,这样不能保证CF是0

    mov ax,[si]
    add [di],ax

    s:inc si
       inc si
    inc di   ;不能用  add di,2这样就可能破坏了CF原有的值，而inc不会影响CF
        inc di

    mov ax,[si]
    adc [di],ax
    loop s

    pop di
```

```
    pop si
    pop ax
    pop cx

    ret



code ends
end start
```

9.找出data数据中，等于8的个数

```
assume cs:code,ds:data

data segment
 db 1,2,3,4,5,6,7,8
data ends
stack segment
  db 16 dup(0)
stack ends

code segment
        start:
        mov ax,data
        mov ds,ax


    mov ax,stack
    mov ss,ax
    mov sp,16
    mov ax,0

    mov si,0
    mov cx,8

    s:cmp byte ptr [si],8

    jne next
    inc ax
    next:
    inc si
    loop s

    mov ax,4c00h
    int 21h

code ends
end start
```

10.直接操控显存，在屏幕正中央显示 白底蓝字welcome to masm!

```
assume cs:code,ds:data

data segment
 db 'welcome to masm!'
data ends

code segment
        start:
            mov ax,data
            mov ds,ax


•       mov ax,0B800H ;显存首地址
•       mov ss,ax;让SS指向显存的首地址

•

•       mov cx,16
•       mov si,0
•       mov bp,1980    ;显存总共25行，每行160字节。12*160+80-16=1980.（显存显示一个字符需要一个字的大小）
•     s:mov al,[si]    ;把字符送入低位
•       mov ah,71h    ;把字符属性送入高位。71h，代表白底蓝字
•       mov [bp],ax
•       add bp,2
•       add si,1
•       loop s
•
•       mov ax,4c00h
•       int 21h

code ends
end start
```

11.在屏幕中，把数据2B用字符显示出来

- 思想 先定义一个0-E的字符表 然后把当前的数据作为偏移量，去表中查找

```
assume cs:code
code segment
    table db '0123456789ABCDE'

start:
    call showbyte

mov ax,4c00h
int 21h
showbyte:
mov al,2bh
mov ah,al
mov cl,4
shr ah,cl ;右移4位，只保留高4位的数据
```

```
and al,00001111b;只保留低4位的数据

mov bh,0
mov bl,ah
mov ah,table[bx]  ;把数字2变成字符2

•

mov bl,al
mov bh,0
mov al,table[bx] ;把数字B变成字符B

mov bx,0B800H ;显存首地址
mov es,bx;让SS指向显存的首地址

mov bh,71h
mov bl,ah

mov es:[1980],bx
mov bl,al
mov es:[1982],bx
ret

•
code ends
end start
```

12.除法错误，运行自己编写的中断处理程序

```
assume cs:code

code segment
    start:
    ;安装程序
    mov si,offset do0
    mov ax,cs
    mov ds,ax

    mov di,200h
    mov ax,0
    mov es,ax



    mov cx,offset do0end-offset do0
    cld
    rep movsb
    ;设置中断表的数据
    mov ax,0
    mov es,ax
    mov ax,200h
    mov es:[0],ax
```

```
    mov ax,0
    mov es:[2],ax

     ;触发触发中断
    mov ax,1000
    mov bh,0
    div bh
    mov ax,4c00h
    int 21h

 ;除法中断程序，完成显示Overflow!
    do0:jmp do0start
    db 'Overflow!';不能把定义的数据当做指令执行了
    do0start:
    mov si,202h
    mov ax,0b800h
    mov es,ax

    mov di,1980
    mov cx,9

    s:mov al,cs:[si]
    mov  es:[di],al
    inc si
    add di,2
    loop s
    mov ax,4c00h
    int 21h
do0end:nop
code ends

end start
```

13.出现int 7ch中断，然后执行我自己的程序。最终在屏幕显示 HELLO

```
assume cs:code,ds:data
data segment
 db 'hello',0
data ends

code segment
    start:
    ;安装程序
    mov ax,cs
    mov ds,ax
    mov si,offset do0


mov ax,0
mov es,ax
mov ax,200h
mov di,ax
```

```
18
19  mov cx,offset do0end-offset do0
20  cld
21  rep movsb
22  ;设置中断表的数据
23  mov ax,0
24  mov es,ax
25  mov ax,200h
26  mov es:[7ch*4],ax
27  mov ax,0
28  mov es:[7ch*4+2],ax
29
30  int 7ch
31
32  mov ax,4c00h
33  int 21h
34
35  do0:
36      push cx
37      push bx
38      push ds
39      push es
40      push si
41      push di
42
43
44  mov bx,0
45  mov si,bx
46  mov bx,1980
47  mov di,bx
48  mov bx,0B800H
49  mov es,bx
50  mov bx,data
51  mov ds,bx
52
53  s:mov ah,0
54  mov cl,[si]
55  jcxz change
56  and byte ptr [si],11011111b
57  mov bl,[si]
58  mov es:[di],bl
59  add di,2
60
61  inc si
62  jmp s
63
64  change:
65      pop di
66      pop si
67      pop es
68      pop ds
69      pop bx
70      pop cx
```

```
    iret
do0end:nop
code ends

end start
```

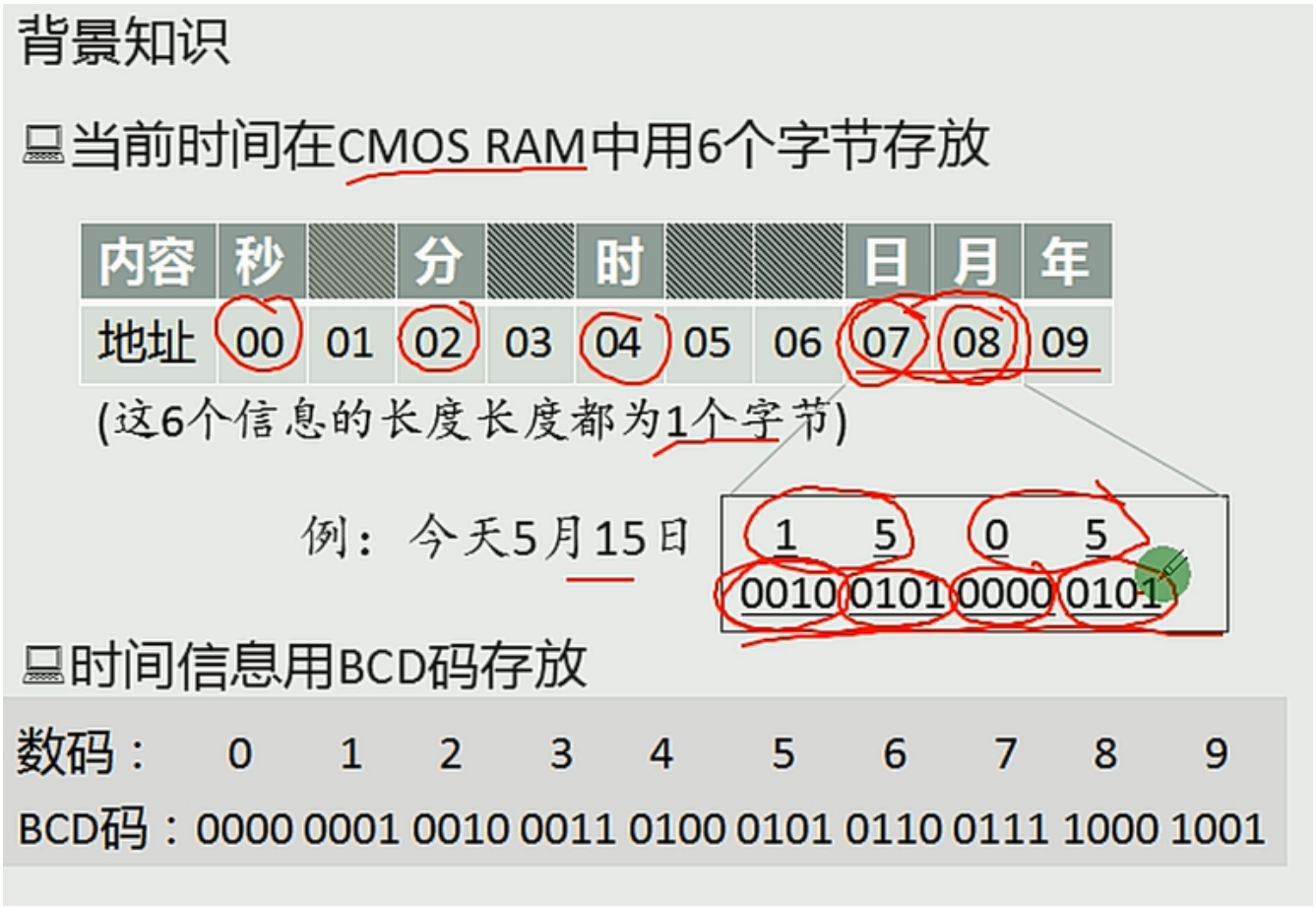
14.获取系统时间


```
assume cs:code

 code segment
 start:
        mov ah,2Ah
        int 21h

        nop
        mov ax,4c00h
        int 21h
code ends
end start
```

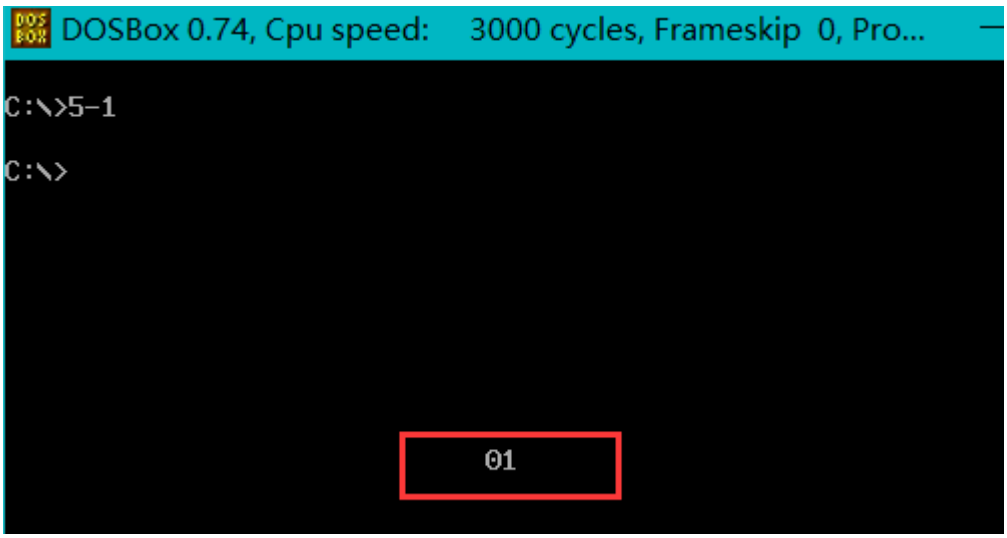
- 执行结果

```
DOSBox 0.74, Cpu speed:   3000 cycles, Frameskip  0, Pro...

AX=2AFF  BX=0000  CX=000A  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=0002   NV UP EI PL NZ NA PO NC
076A:0002 CD21          INT     21
-t

AX=2AFF  BX=0000  CX=000A  DX=0000  SP=FFFA  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=F000  IP=14A0   NV UP DI PL NZ NA PO NC
F000:14A0 FB            STI
-t

AX=2AFF  BX=0000  CX=000A  DX=0000  SP=FFFA  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=F000  IP=14A1   NV UP EI PL NZ NA PO NC
F000:14A1 FE38          ???     [BX+SI]                         DS:0000=CD
-t

AX=2A01  BX=0000  CX=07E5  DX=0104  SP=FFFA  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=F000  IP=14A5   NV UP EI PL NZ NA PO NC
F000:14A5 CF            IRET
-t

AX=2A01  BX=0000  CX=07E5  DX=0104  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=0004   NV UP EI PL NZ NA PO NC
076A:0004 90            NOP
-
```

15.访问当前月份



```
assume cs:code

 code segment
 start:
                ;需要读取8号地址的信息
          mov al,8
          out 70h,al
          in al,71h

             ;把十位和各位信息分开
          mov ah,al
          mov cl,4
          shr ah,cl
          and al,00001111b
          ;转换为ASCII码
          add ah,30h
          add al,30h
          ;操作显存，显示信息
          mov bx,0B800H
          mov es,bx
          mov byte ptr es:[1980],ah
          mov byte ptr es:[1982],al

          mov ax,4c00h
          int 21h

code ends
end start
```



16.屏幕从a-z依次显示，按下esc就改变颜色

```
assume cs:code,ss:stack,ds:data

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

```
    db 128 dup(0)
stack ends
 code segment
 start:
        mov ax,stack
        mov ss,ax
        mov sp,128
        mov ax,data
        mov ds,ax

        ;改变9号中断例程入口
        mov ax,0
        mov es,ax
        push es:[9*4]
        pop ds:[0]
        push es:[9*4+2]
        pop ds:[2]
        ;保存，9号中断例程的地址信息后，转到自己的中断例程中
        mov word ptr es:[9*4],offset int9
        mov es:[9*4+2],cs



        ;显示a~z
        mov ax,0B800H
        mov es,ax
        mov ah,'a'
        s:mov es:[160*12+40*2],ah
        call delay
        inc ah
        cmp ah,'z'
        jna s

        ;回复原来中断例程的地址
        mov ax,0
        mov es,ax
        push ds:[0]
        pop es:[9*4]
        push ds:[2]
        pop es:[9*4+2]

        mov ax,4c00h
        int 21h

        ;按esc切换颜色,这个中断例程，不是在代码中调用的。他是发生外部中断时候调用的。（键盘敲入等时候，调用）
        int9:
        push ax
        push bx
        push es

        in al,60h;读取当前键入扫描码

```

```
        ;模拟调用int 9中断。
        pushf
        pushf
        pop bx
        and ah,11111100b;修改IF,TF值为0
        push bx
        popf
        call dword ptr ds:[0];call相当于把cs,ip入栈的操作做了

        cmp al,1  ;esc的扫描码是1，查看是不是esc
        jna int9end
        mov bx,0B800H
        mov es,bx
        inc byte ptr es:[160*12+80+1]
int9end:
        pop es
        pop bx
        pop ax
        iret




        ;延迟效果
        delay:
        push cx
        push bx
        mov cx,0FFFFH
        mov bx,3
        s2:nop
        cmp cx,1
        je setcx
        loop s2

        setcx:
        cmp bx,1
        je  delayend
        mov cx,0FFFFH
        dec bx
        jmp s2
delayend:pop bx
        pop cx

        ret
        mov ax,4c00h
        int 21h

code ends
end start
```

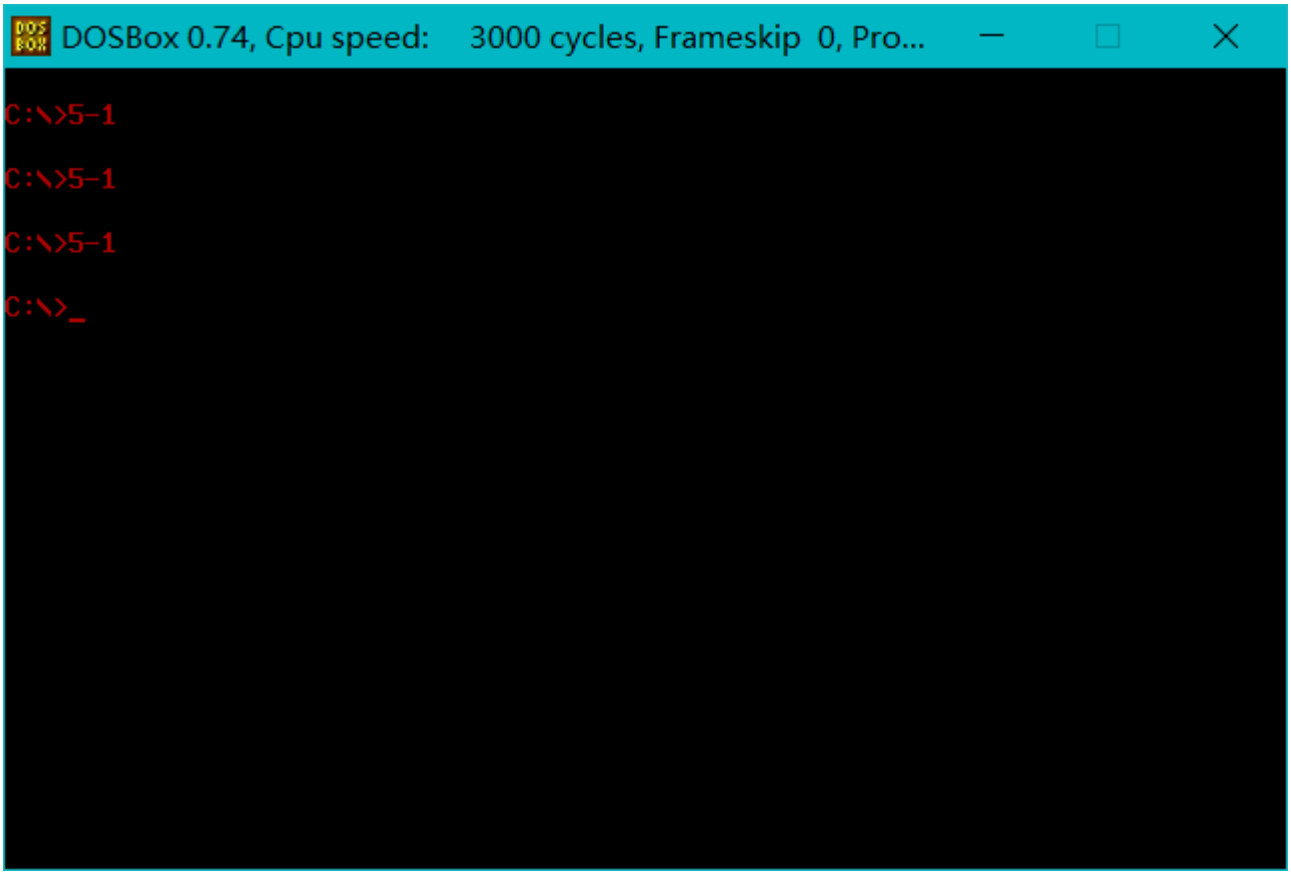
17.修改int 9中断例程，按下f1切换屏幕字体颜色

```
assume cs:code,ss:stack,ds:data
```

```
;修改int 9中断例程，按下f1切换屏幕字体颜色
data segment
    dw 0,0
data ends
stack segment
    db 128 dup(0)
stack ends
 code segment
 start:
        mov ax,stack
        mov ss,ax
        mov sp,128
        mov ax,data
        mov ds,ax
        ;安装程序
        mov ax,cs
        mov ds,ax
        mov ax,offset int9
        mov si,ax

        mov ax,0
        mov es,ax
        mov ax,204h
        mov di,ax

        mov cx,offset int9end-offset int9
        rep movsb

        ;保存原来程序的地址,并且更新新的中断例程入口
        cli
        mov ax,0
        mov es,ax
        push es:[9*4]
        pop es:[200h]
        push es:[9*4+2]
        pop es:[202h]

        mov word ptr es:[9*4],200h
        mov word ptr es:[9*4+2],0
        sti
        mov ax,4c00h
        int 21h



        ;修改int9中断例程
        int9:
        push ax
        push bx
        push cx
        push es

        pushf
```

```
        pushf
        pop bx
        and bh,11111100b
        push bx
        popf

        mov ax,0
        mov es,ax
        in al,60h;保存键入的扫描码



        call dword ptr es:[200h]

        cmp al,3bh
        jne done
        mov ax,0B800H
        mov es,ax
        mov bx,1
        mov cx,160*25
    s:
        inc byte ptr es:[bx]
        add bx,2
        loop s
    done:pop es
        pop cx
        pop bx
        pop ax
        iret
        int9end:nop




code ends
end start
```

18.实现，按下 r,g,b键，屏幕字体颜色变红，绿，蓝

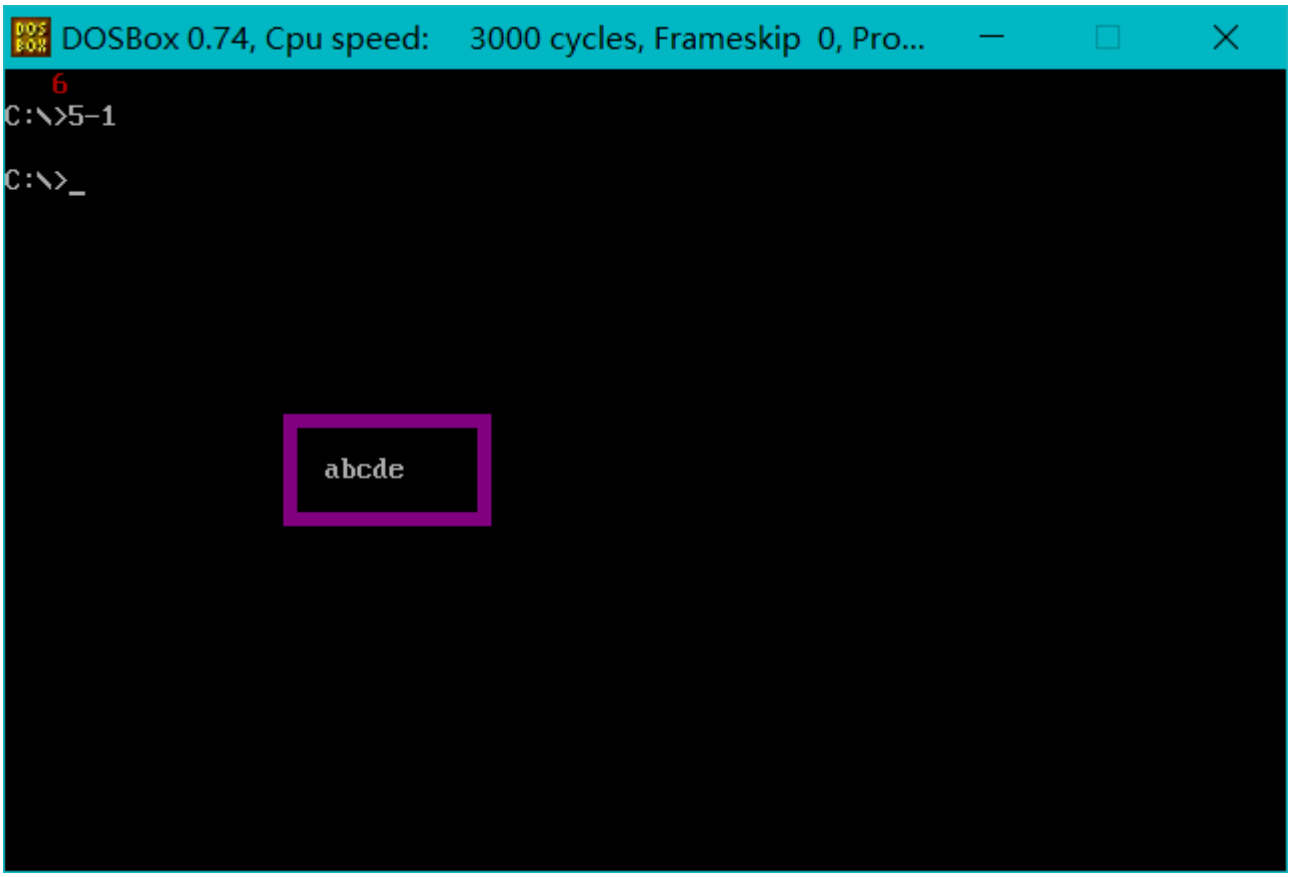



```
assume cs:code,ss:stack,ds:data
data segment
    dw 0,0
data ends
stack segment
    db 128 dup(0)
stack ends
 code segment
 start:
        mov ax,stack
        mov ss,ax
        mov sp,128
        mov ax,data
        mov ds,ax

        mov ah,0
        int 16h

        mov ah,1
        cmp al,'r'
        je red
        cmp al,'g'
        je green
        cmp al,'b'
        je blue
        jmp sret

        red:shl ah,1
```

```
        green:shl ah,1
        blue:mov bx,0B800H
        mov es,bx
        mov bx,1
        mov cx,160*25
          ;先把RGB这三个设置为0
        s:and byte ptr es:[bx],11111000b
          ;把自己设置的颜色，赋值上去
        or byte ptr es:[bx],ah
        add bx,2
        loop s


        sret:mov ax,4c00h
        int 21h



code ends
end start
```

19.字符串的输入

- 按回车键：删除一个字符
- 按字符键：输入一个字符
- 按回车键：字符输入结束，在字符最后添加一个0，然后显示字符



```
assume cs:code,ss:stack,ds:data

data segment
    db 32 dup(0)
    top db 0
data ends
stack segment
    db 128 dup(0)
stack ends
 code segment
 start:
        mov ax,stack
        mov ss,ax
        mov sp,128
        mov ax,data
        mov ds,ax

        mov si,0
        mov dh,12
        mov dl,20
        call getstr
        return:
        mov ax,4c00h
        int 21h

    getstr:
        push ax
    getstrs:
        mov ah,0
        int 16h

        cmp al,20h;获取到的字符（不用扫描码），查看是不是字符
        jb nochar

        mov ah,0
        call charstack;是字符，就调用函数，字符入栈
        jmp getstrs

    nochar:
        cmp ah,0eh;退格键扫描码
        je backspace
        cmp ah,1ch;回车键的扫描码
        je enters
        jmp getstrs
    backspace:
        mov ah,1
        call charstack
        jmp getstrs

    enters:
        mov al,0
        mov ah,0
        call charstack;结束输入，把0字符入栈
```

```
54
55          mov ah,2;显示字符串
56          call charstack
57
58
59
60          pop ax
61          ret;getstr结束
62
63
64      charstack:
65          jmp short charstart
66          chartable db '0123456789ABCDE'
67          table dw charpush,charpop,charshow
68      charstart:
69          push bx
70          push dx
71          push di
72          push es
73          push cx
74
75          cmp ah,2
76
77          ja  sret  ;大于2，就不是调用。直接返回
78
79          mov bl,ah
80          mov bh,0
81          add bx,bx
82          jmp word ptr table[bx]
83
84      charpush:
85          mov bx,0B800H
86          mov es,bx
87          ;输出当前敲的是哪个键
88          mov byte ptr es:[2],al
89          mov byte ptr es:[3],00000100b
90
91          mov bl,top
92          mov bh,0
93          mov [si+bx],al
94          add top,1
95
96
97          mov bl,top;测试  top有没有+1
98          mov bh,0
99          ;显示当前有几个字符串，top的值
100         mov bl,chartable[bx]
101         mov byte ptr es:[6],bl
102         mov byte ptr es:[7],00000100b
103
104         jmp sret
105
106     charpop:
```

```
        cmp top,0
        je sret
        dec top
        mov bl,top
        mov bh,0
        mov al,[si+bx]
        ;测试有没有调用这个
        ;mov bx,0B800H
        ;mov es,bx
        ;mov byte ptr es:[160*2],'o'
        ;mov byte ptr es:[160*2+1],00000100b
        jmp sret

    charshow:
        cmp top,0
        je sret;没有字符串了，直接回去
        mov bx,0B800H
        mov es,bx
        mov bx,0
        mov di,12*160+40

        mov cl,top
        mov ch,0

        s1:mov  al,[bx]
        mov es:[di],al
        add di,2
        inc bx


        loop s1

    sret:
        pop cx
        pop es
        pop di
        pop dx
        pop bx

        ret



code ends
end start
```

20.让计算机唱歌

```
assume cs:code,ss:stack,ds:data

data segment
    ;音乐频谱
```

```
    mus_freq dw 262,262,262,196,330,330,330,262
    dw 262,330,392,392,349,330,294
    dw 294,330,349,349,330,294,330,262
    dw 262,330,294,196,247,294,262,-1
    ;每个应付延续时间
    mus_time dw 3 dup(12,12,25,25),12,12,50
    dw 3 dup(12,12,25,25),12,12,50
data ends
stack segment
    db 100 dup(0)
stack ends
 code segment
 start:
        mov ax,stack
        mov ss,ax
        mov sp,100
        mov ax,data
        mov ds,ax
        lea si,mus_freq
        lea di,mus_time
    play:
        mov dx,[si]
        cmp dx,-1
        je end_play
        call sound
        add si,2
        add di,2
        jmp play
    end_play:
        mov ax,4c00h
        int 21h

    sound:
        push ax
        push dx
        push cx

        ;把频率传送给，8253芯片，下面的设置，都是固定的
        mov al,0b6h
        out 43h,al
        mov dx,12h
        mov ax,34dch
        div word ptr [si];si里面就是音乐频率，这个是自己给的
        out 42h,al
        mov al,ah
        out 42h,al


        ;控制8255芯片，把扬声器打开
        in al,61h
        mov ah,al;保存当前扬声器的状态，以便稍后恢复
        or al,3
        out 61h,al
```

```
58
59          ;设置扬声器开启时长
60
61          mov dx,[di]
62      wait1:
63          mov cx,20000
64      delay:
65          nop
66          loop delay
67          dec dx;这里用的是双重循环。dx判断外重循环多少次
68          jnz wait1
69          ;关闭扬声器，恢复扬声器的值
70          mov al,ah
71          out 61h,al
72
73          pop cx
74          pop dx
75          pop ax
76          ret
77
78  code ends
79  end start
```