暨南大学本科实验报告专用纸

课程名称__汇编语言实验__实验项目名称__直接定址表__实验项目编号__15__实验项目类型__上机__实验地点__517__成绩评定_____

学生姓名_陈文笛__学号_2021103285__学院__网络空间安全__系_网络空间安全__专业__网络空间安全_实验时间__2023__年_5_月_29_日_下_午~__月__日__午

温度__℃ 湿度__指导教师__张银炎_____

实验简介

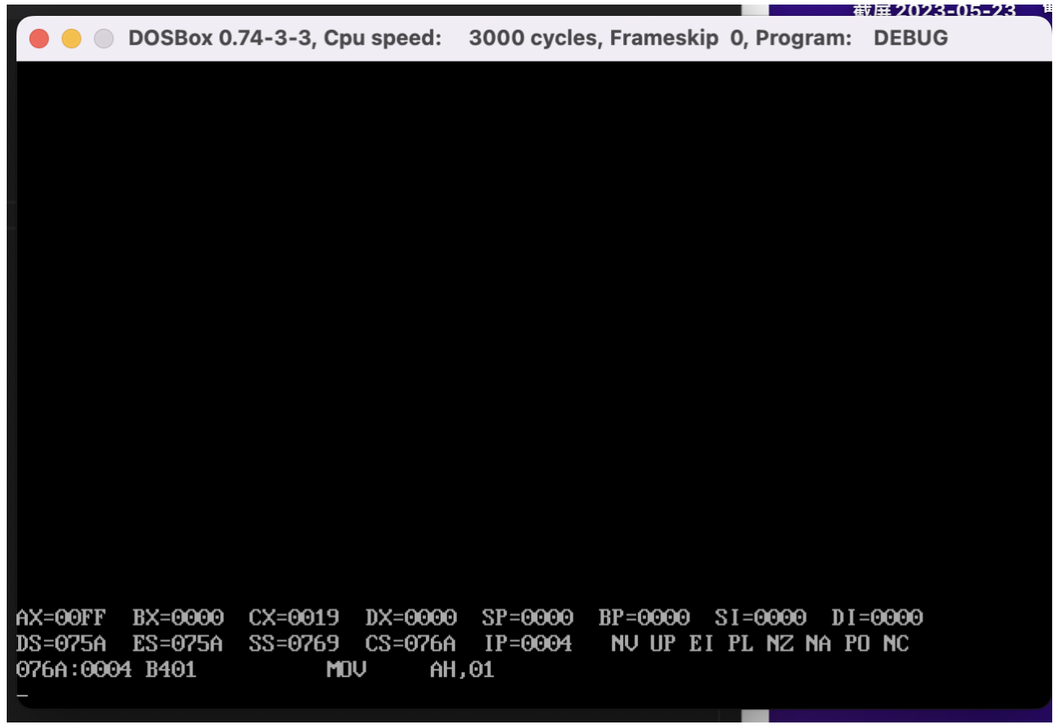
安装一个新的 int 7ch 中断例程，为显示输出提供如下功能子程序。 (1)清屏;(2)设置前景色; (3)设置背景色; (4)向上滚动一行。

入口参数说明如下。

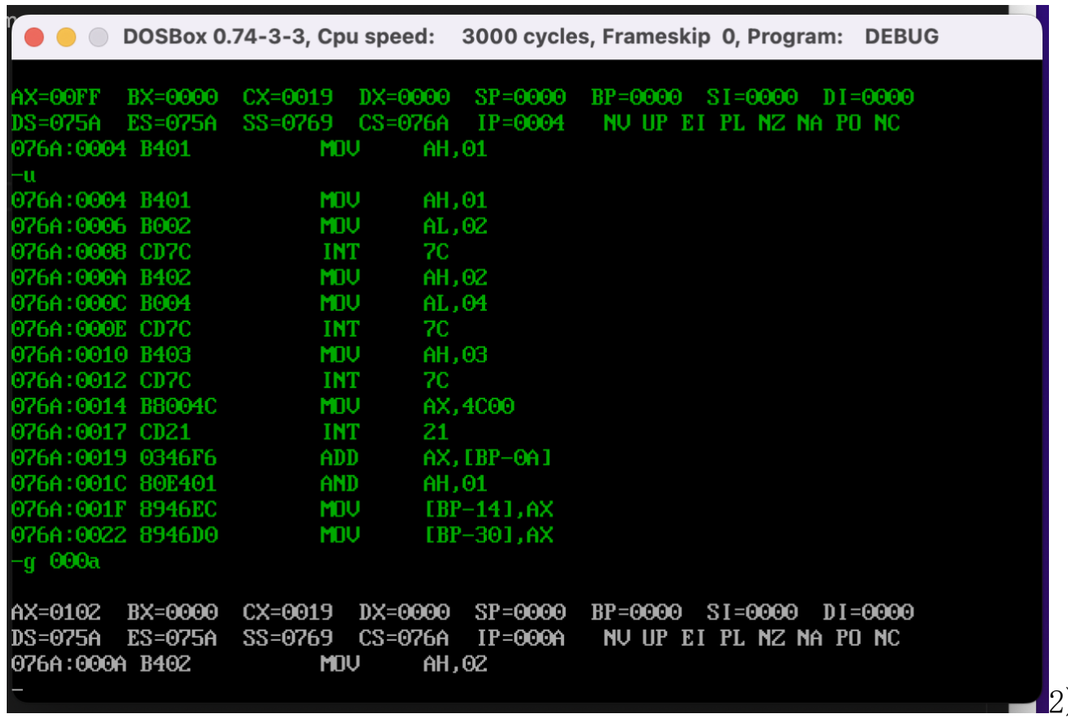
(1) 用 ah 寄存器传递功能号:0 表示清屏;1 表示设置前景色;2 表示设置背景色;3 表示向上滚动一行;

(2) 对于 1、2 号功能，用 al 传递颜色值，(al)∈ {0,1,2,3,4,5,6,7}。

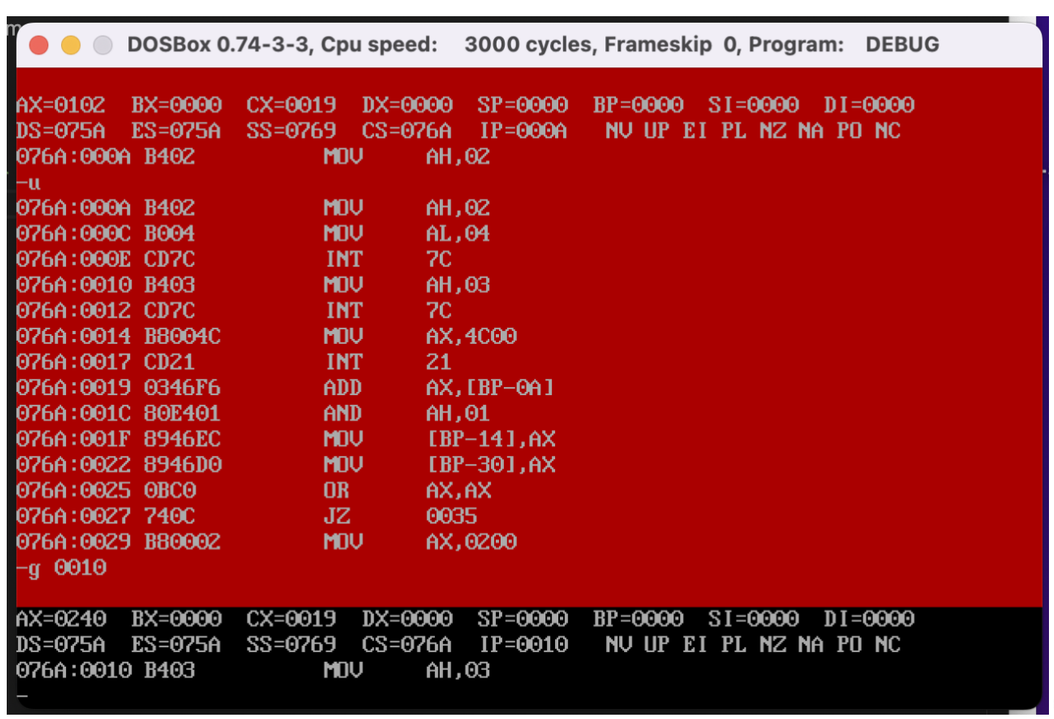
实验结果截图



(1)清屏;



2)设置前景色



(3)设置背景色

```
DOSBox 0.74-3-3, Cpu speed:   3000 cycles, Frameskip  0, Program:   DEBUG
076A:0010 B403          MOV     AH,03
076A:0012 CD7C          INT     7C
076A:0014 B8004C        MOV     AX,4C00
076A:0017 CD21          INT     21
076A:0019 0346F6        ADD     AX,[BP-0A]
076A:001C 80E401        AND     AH,01
076A:001F 8946EC        MOV     [BP-14],AX
076A:0022 8946D0        MOV     [BP-30],AX
076A:0025 0BC0          OR      AX,AX
076A:0027 740C          JZ      0035
076A:0029 B80002        MOV     AX,0200
076A:002C 2B46EC        SUB     AX,[BP-14]
076A:002F 8946EC        MOV     [BP-14],AX
-g 0014


AX=0340  BX=0000  CX=0019  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=0014   NV UP EI PL NZ NA PO NC
076A:0014 B8004C        MOV     AX,4C00
-t

AX=4C00  BX=0000  CX=0019  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=0017   NV UP EI PL NZ NA PO NC
076A:0017 CD21          INT     21
-
```

(4)向上滚动一行

实验结果分析

（1）～（3）从显示状态可以看出实验成功。

（4）执行完 int7 后与寄存器状态表中间隔了两行，说明实现了向上滚动，并且最后一行设为空格

遇到的问题及解决方法

1.

```
;这里一定要重设 es，因为前面传输完 es 可能被改变了
    mov ax,0
    mov es,ax
;记录下原始的中断程序地址,写入新程序前
```

push es:[7ch*4]

pop es:[200h]

push es:[7ch*4+2]

pop es:[202h]

2.

call word ptr table[bx];注意，虽然 table 是 dw 但是实际存的只是偏移地址，只有 dd 时才是偏移地址加段地址

;所以这里一定要用 call word ptr，不能用 call dword ptr

;因为 call dword ptr 后，高字是 CS，低字是 IP

3.

转跳到子程序时必须手动加上偏移地址！

正确：call word ptr table[bx+204h]（中断程序写在最前面时，如果写在其他地方加的数不一样）**或者 call word ptr cs:[bx+206h]**

由于 table[bx]转跳的地址是相对于 table 的偏移地址，而 table 在第一次编译时的地址明显不同于我们后来把 table 复制到的地址（0:204h）

这个时候，如果采用 table[bx]，取到的是 table 在第一次编译时的地址附近，并不是我们想要的 0:204h

call word ptr cs:[bx+206h] cs:[bx+206h]是 table 的新地址

代码

中断安装程序

```
assume cs:code

code segment

screen:
```

```
    jmp short set; 此指令占 2 字节
    ; 此 dw 数据开始地址为 0:204h 入口地址+2 即 0:20h
    ;由于中断向量表中一个表项是占两个字的，所以 int7 是从 0:204h 开始，前面两个字是原中断程序地址
    table dw 204h+a1-screen, 204h+sub2-screen, 204h+sub3-screen,204h+sub4-screen
set:
    push bx

    ; pushf;标志寄存器入栈
    ; call dword ptr cs:[200h];执行旧程序

    cmp ah,3;ah 代表要执行第几个，从 0 开始，因为地址从 0 开始
    ja sret
    mov bl,ah;ah 表示想执行第几个程序
    mov bh,0
    add bx,bx

    ;call word ptr table[bx+204h];注意，虽然 table 是 dw 但是实际存的只是偏移地址，只有 dd 时才是偏移地址加段地址
    ;所以这里一定要用 call word ptr，不能用 call dword ptr
    ;因为 call dword ptr 后，高字是 CS，低字是 IP
    call word ptr cs:[bx+206h];cs:[bx+206h]是 table 的地址

sret:pop bx
```

```
    iret
    
    
    
    
    
a1:;清屏
    push bx
    push si
    push es
    push cx

    mov bx,0b800h
    mov es,bx
    mov si,0
    mov cx,2000
sub1s:mov byte ptr es:[si],' '
    add si,2
    loop sub1s

    pop cx
    pop es
    pop si
    pop bx
    ret
```

```
sub2:;设置前景色
    push bx
    push es
    push si
    push cx


    mov bx,0b800h
    mov es,bx
    mov si,1
    mov cx,2000
sub2s:and byte ptr es:[si],11111000b
or es:[si],al     ;al 是执行中断前用户选择的
add si,2
loop sub2s


pop cx
pop si
pop es
pop bx
ret
```

```
sub3:;设置背景色
    push bx
    push es
    push si
    push cx

    mov cl,4
    shl al,cl;左移 4 位补 0
    mov bx,0b800h
    mov es,bx
    mov si,1
    mov cx,2000
sub3s:and byte ptr es:[si],1000111b
or es:[si],al ;al 是执行中断前用户选择的
add si,2
loop sub3s

    pop cx
    pop si
    pop es
    pop bx
    ret
```

```
sub4:
    push bx
    push ds
    push si
    push es
    push di
    push cx

    mov bx,0b800h
    mov ds,bx
    mov si,160;ds:si=第 n+1 行
    mov es,bx
    mov es,bx
    mov di,0;es:di=第 n 行

    cld
    mov cx,24;循环 24 次（一共 25 行）
    sub4s:
    push cx
    mov cx,160;每次移动 160 字节,一行复制完，si+=160，di+=160
    rep movsb
    pop cx
    loop sub4s
```

```
    mov cx,80
    mov di,0
    s:
    mov byte ptr es:[160*24+di],' '
    add di,2
    loop s

    pop cx
    pop di
    pop es
    pop si
    pop ds
    pop bx
    ret

screenend:nop

start:

;将程序写入
        mov ax,cs
        mov ds,ax
```

```
        mov ax,0
        mov es,ax

        mov si,offset screen;ds:si 为源地址
        mov di,204h;es,di 为目标地址，一段安全的空间 0:200，但前面[200]和[202]要存放原 int9 的 IP 和 CS

        mov cx,offset screenend-offset screen
        cld;传输方向为正
        rep movsb

;这里一定要重设 es，因为前面传输完 es 被改变了
    mov ax,0
    mov es,ax
;记录下原始的中断程序地址,写入新程序前
        push es:[7ch*4]
        pop es:[200h]
        push es:[7ch*4+2]
        pop es:[202h]

;改变中断向量表
    mov ax,0
    mov es,ax
        cli
```

```
        mov word ptr es:[7ch*4],204h

        mov word ptr es:[7ch*4+2],0 ;这里[7*4+2]或[7ch 均可*4+2]

        sti




        mov ax,4c00h

        int 21h




code ends

end start
```

中断测试程序

```
assume cs:code

code segment

start:

    ; 测试清屏

    mov ah,0;ah 代表要执行第 0 个

    int 7ch



    ; 测试绿色字
```

```
mov ah,1

mov al,2

int 7ch



; 测试红底

mov ah,2

mov al,4

int 7ch



; 向上滚动一行

mov ah,3

int 7ch



    int 7ch



    mov ax,4c00h

    int 21h

code ends

end start
```