**实验2­­­：加载用户程序的监控程序**

姓名：姜洋帆 学号：17341068

院系：数据科学与计算机学院 专业：17级计算机科学（大数据）

【**实验题目**】加载用户程序的监控程序

**【实验目的】**

了解BIOS中断，学习监控程序如何获取计算机硬件系统的控制权、如何控制用户程序的执行，并开发出最原始的操作系统。

【**实验要求】**

设计四个有输出的用户可执行程序，分别在屏幕1/4区域动态输出字符，如将用字符‘ A’从屏幕左边某行位置45度角下斜射出，保持一个可观察的适当速度直线运动，碰到屏幕相应1/4区域的边后产生反射，改变方向运动，如此类推，不断运动；在此基础上，增加你的个性扩展，如同时控制两个运动的轨迹，或炫酷动态变色，个性画面，如此等等，自由不限。还要在屏幕某个区域特别的方式显示你的学号姓名等个人信息。

【**实验方案】**

1. 硬件及虚拟机配置：Lenovo PC ；Oracle Virtual Box
2. 软件工具及作用：

Notepad++ ：写代码

Nasm：编译asm文件到bin文件

Sublime ：查看、编辑二进制文件，将编译完成的程序机器码写入软盘

C程序：自己编写的一个C语言程序，用来将编译生成的二进制文件填充为1.44M大小，并将后缀改为.img，生成1.44M的软盘

1. 程序功能

Myos1作为引导，键入A执行程序A，功能与实验一相同，显示区域限制在左上角1/4区域内，程序内部设一个时钟，达到时间后返回引导程序，键入B执行程序B，同样显示一个字符移动并反弹，取消了控制字符长度的功能，保留变色，显示区域为右上角1/4个区域。第三个程序是显示一个字符串（个人信息）并改变前景和背景颜色，一共改变十次，然后结束程序跳回引导。第四个程序为发射一个字符（屏幕上仅保留一个字符在移动），并显示字符串。字符每次碰撞到上下边界时会改变中间字符串的颜色，同时字符自身的颜色也在不断改变。

1. 程序设计

在myos显示字符之前进行清屏，显示字符后加入键入选择模块和跳转模块，其中第二个程序的跳转中将cl置为3，将第二段程序的bin文件放入虚拟软盘的第三个扇区，也就是400h开始。

在原有的stoneM代码里面添加loop循环指令和跳转模块回跳到myos

第三个程序（b.asm）是在stoneM基础上把显示的字符改成B，并且换掉字符颜色。

第四个是编写的新的程序(try.asm)，显示出一个菱形并且用10h中断的06号功能上滚和07号功能下滚让它上下运动，并且设置循环让他运动五个来回。之后输入字符‘m’退出程序回到引导程序。

五、主要代码

Myos1引导程序：

;程序源代码（myos1.asm）

org 7c00h ; BIOS将把引导扇区加载到0:7C00h处，并开始执行

OffSetOfUserPrg1 equ 0A100h

OffSetOfUserPrg2 equ 0A200h

OffSetOfUserPrg3 equ 0A300h

OffSetOfUserPrg4 equ 0A400h

Start:

mov ax, cs ; 置其他段寄存器值与CS相同

mov ds, ax ; 数据段

mov bp, Message ; BP=当前串的偏移地址

mov ax, ds ; ES:BP = 串地址

mov es, ax ; 置ES=DS

mov cx, 44 ; CX = 串长（=9）

mov ax, 1301h ; AH = 13h（功能号）、AL = 01h（光标置于串尾）

mov bx, 0007h ; 页号为0(BH = 0) 黑底白字(BL = 07h)

mov dh, 0 ; 行号=0

mov dl, 0 ; 列号=0

int 10h ; BIOS的10h功能：显示一行字符

;press a key

mov ah,0

int 16h

cmp al,'a'

je LoadnEx1

cmp al,'b'

je LoadnEx2

cmp al,'c'

je LoadnEx3

cmp al,'d'

je LoadnEx4

LoadnEx1:

;读软盘或硬盘上的若干物理扇区到内存的ES:BX处：

mov ax,cs ;段地址 ; 存放数据的内存基地址

mov es,ax ;设置段地址（不能直接mov es,段地址）

mov bx, OffSetOfUserPrg1 ;偏移地址; 存放数据的内存偏移地址

mov ah,2 ; 功能号

mov al,1 ;扇区数

mov dl,0 ;驱动器号 ; 软盘为0，硬盘和U盘为80H

mov dh,0 ;磁头号 ; 起始编号为0

mov ch,0 ;柱面号 ; 起始编号为0

mov cl,2 ;起始扇区号 ; 起始编号为1

int 13H ; 调用读磁盘BIOS的13h功能

; 用户程序a.com已加载到指定内存区域中

jmp OffSetOfUserPrg1

LoadnEx2:

;读软盘或硬盘上的若干物理扇区到内存的ES:BX处：

...

mov bx, OffSetOfUserPrg2 ;偏移地址; 存放数据的内存偏移地址

...

jmp OffSetOfUserPrg2

LoadnEx3:

;读软盘或硬盘上的若干物理扇区到内存的ES:BX处：

...

mov bx, OffSetOfUserPrg3 ;偏移地址; 存放数据的内存偏移地址

...

jmp OffSetOfUserPrg3

LoadnEx4:

...

mov bx, OffSetOfUserPrg4 ;偏移地址; 存放数据的内存偏移地址

...

jmp OffSetOfUserPrg4

AfterRun:

jmp $ ;无限循环

Message:

db 'Hello, MyOs is loading user program...'

MessageLength equ ($-Message)

times 510-($-$$) db 0

db 0x55,0xaa

程序中返回引导程序的模块：

return:

mov ah,0 ; 功能号

int 16h ; 调用16H号中断

; 按任一键：退出返回DOS

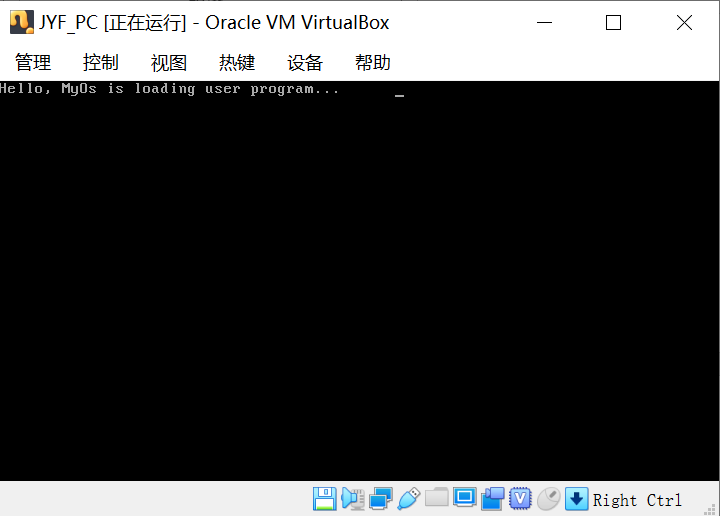
jmp 07c00h

实现时，设置为满足一定条件触发调用return，返回引导程序

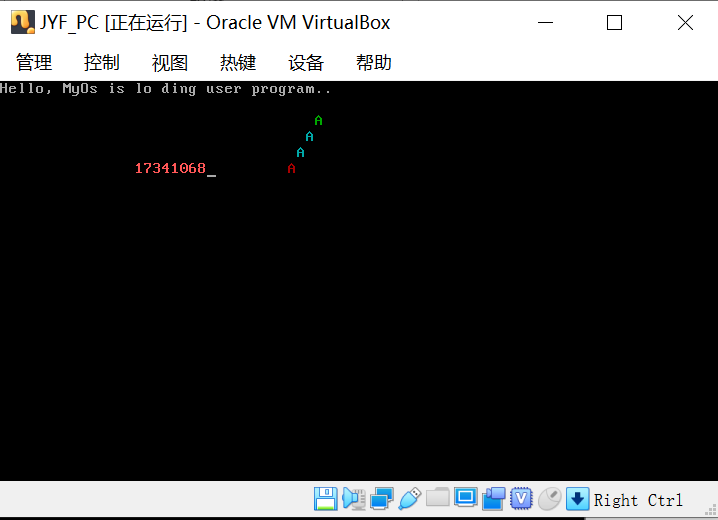
**【实验过程】**

功能展示

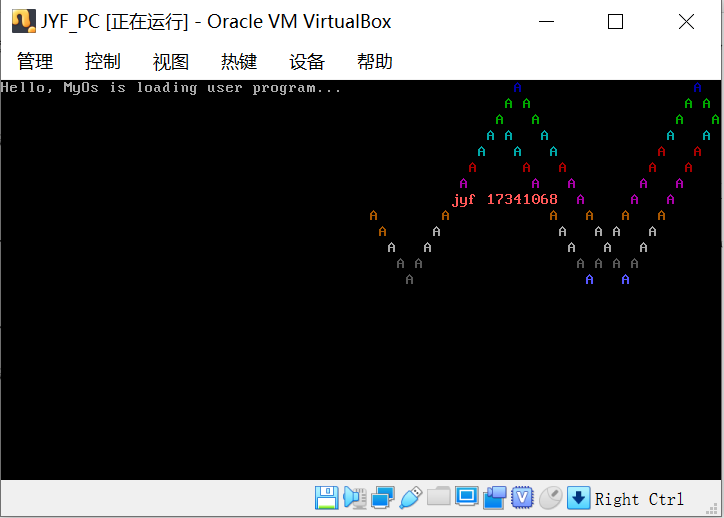
引导程序界面，打印信息，在键盘键入ABCD四个不同字母后，会分别进入第1、2、3、4个程序显示不同信息。不同程序分别显示在屏幕的左上、右上左下、右下四个1/4区域内。



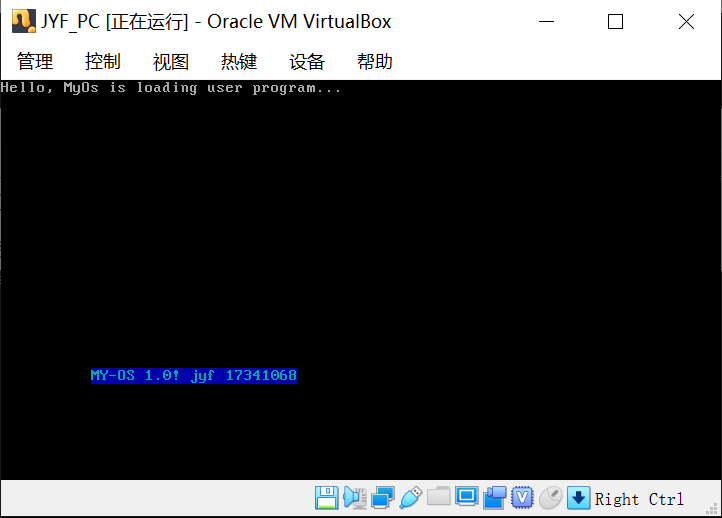
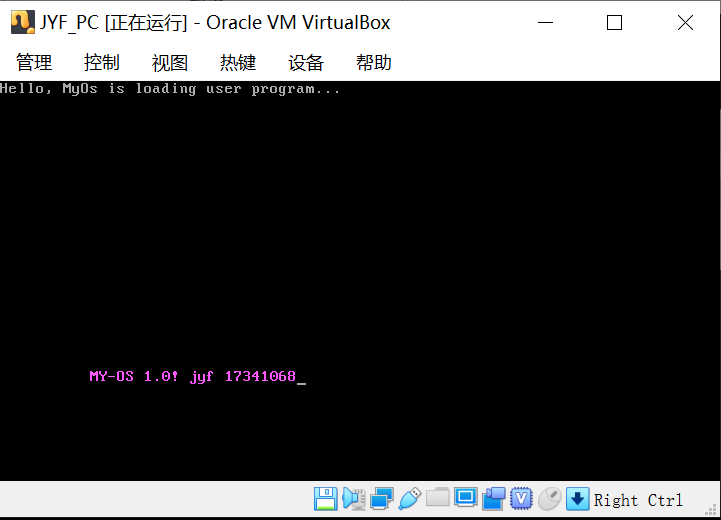
键入A，开始执行第一个程序，在左上1/4个区域内显示

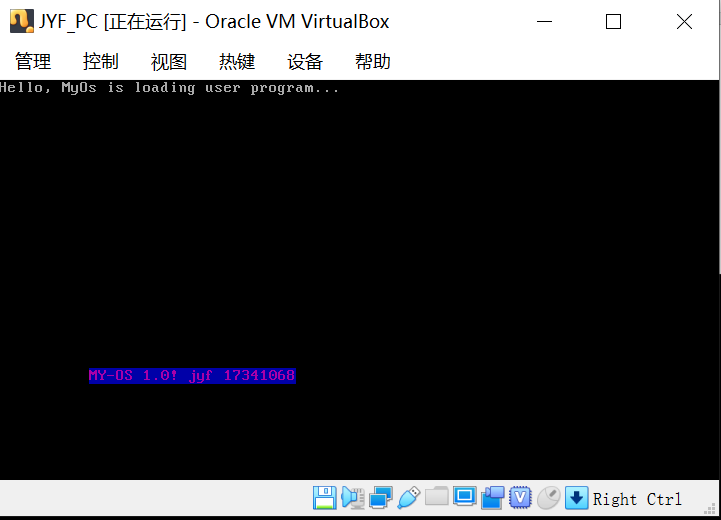
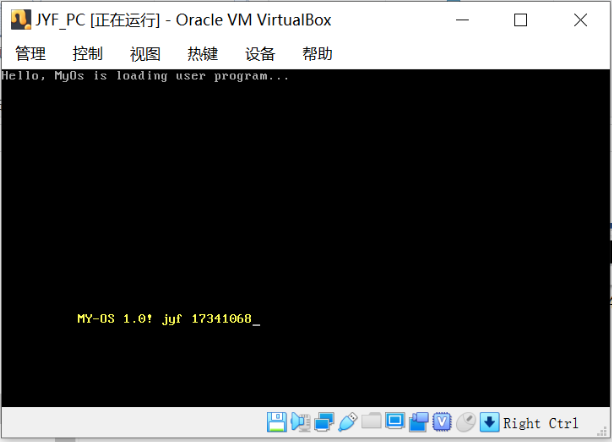


键入B执行第2个程序，在右上方1/4区域内显示

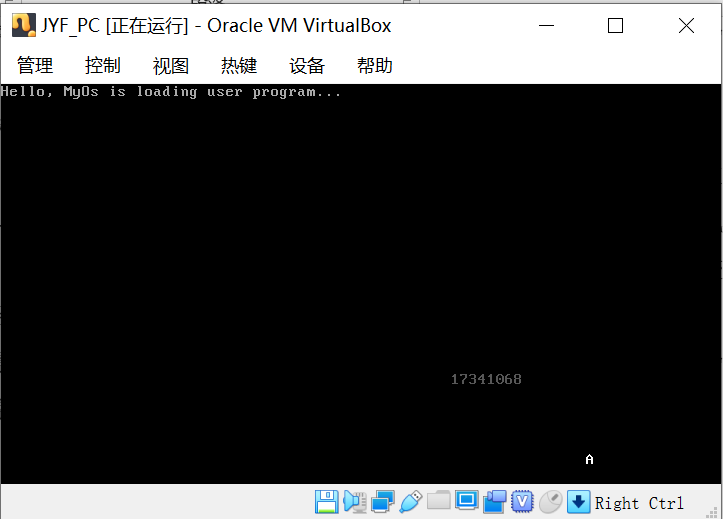


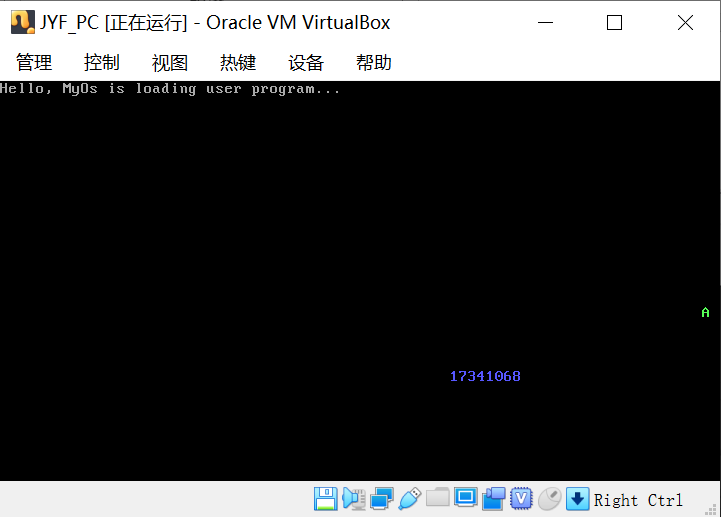
键入C，进入第三个程序，显示区域为左下，功能为改变字符的前景颜色和背景色，共有十个变化，之后回到引导程序

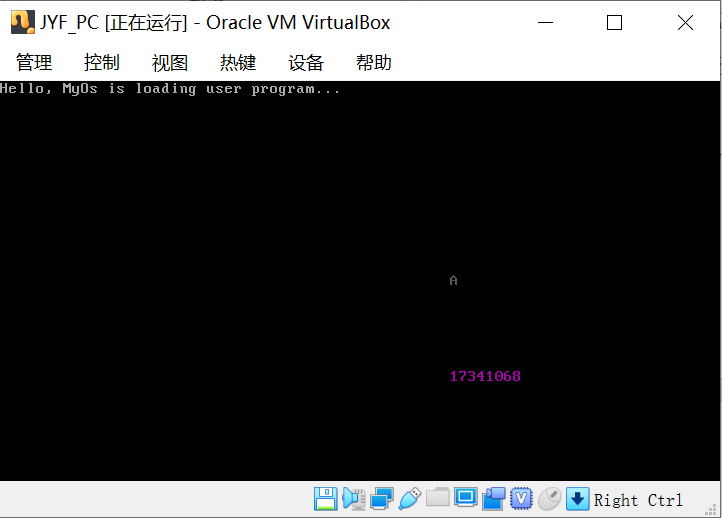




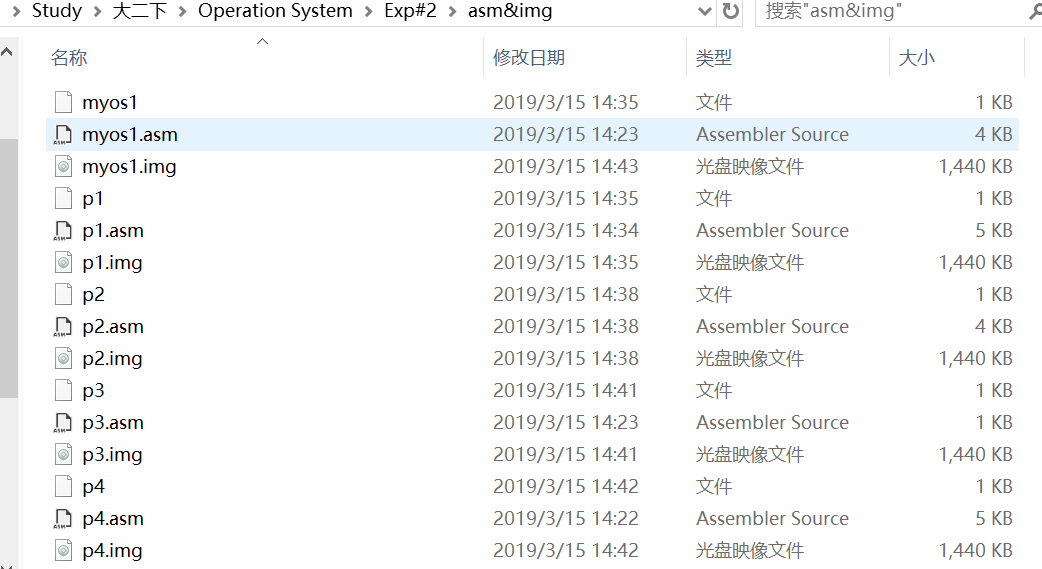
键入D进入第4个程序，显示区域限制在右下方，功能为一个字符在区域内不断移动、反弹，中间显示一个字符串，字母每在上下边框反弹一次，中间的字符就改变一次颜色，改变10次颜色后退出，返回引导程序。







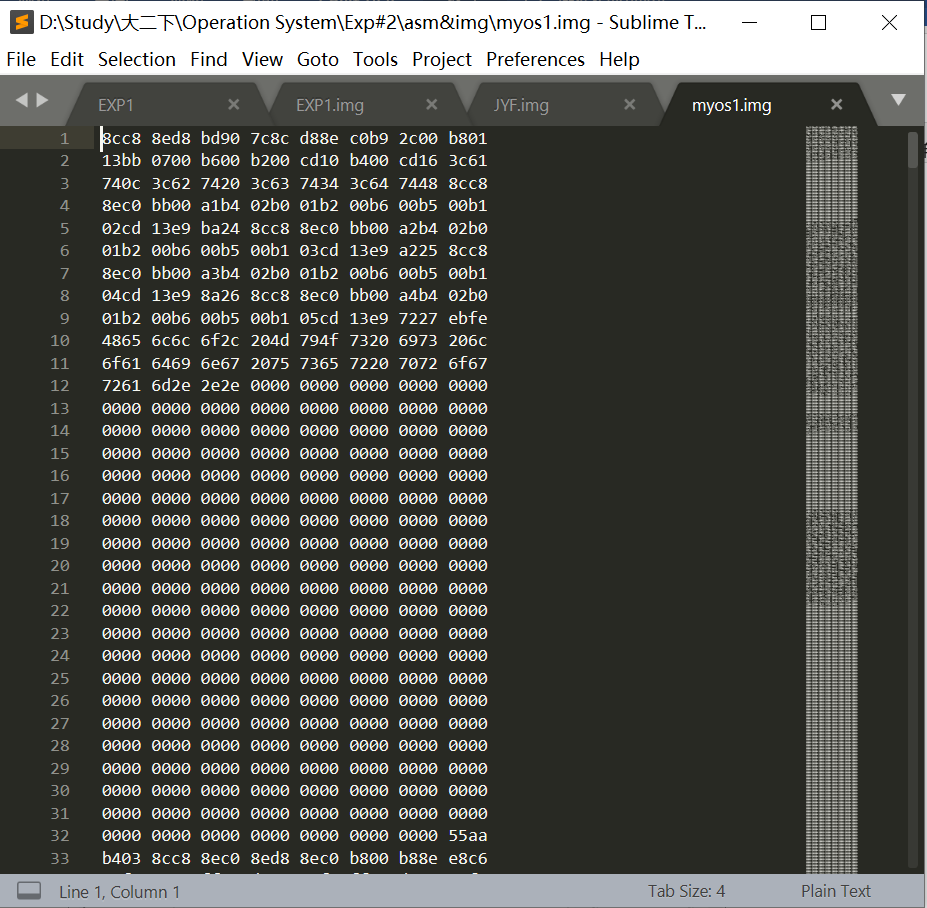
中间文件：

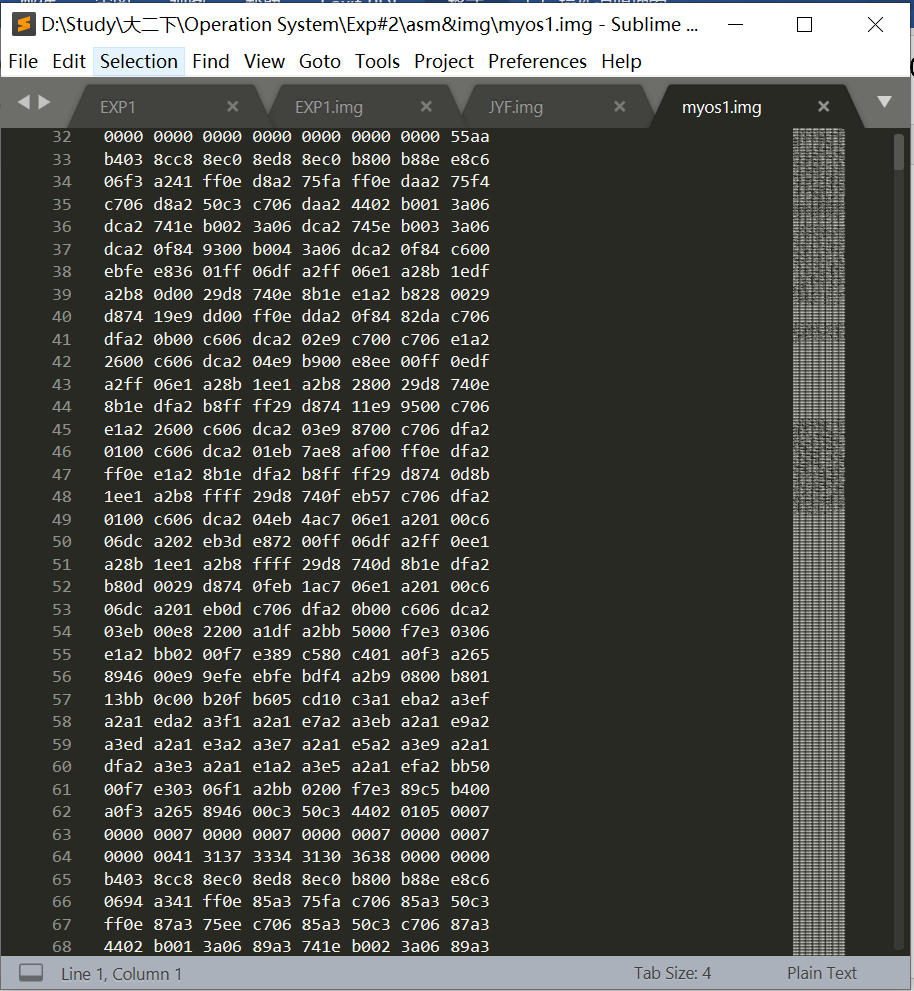


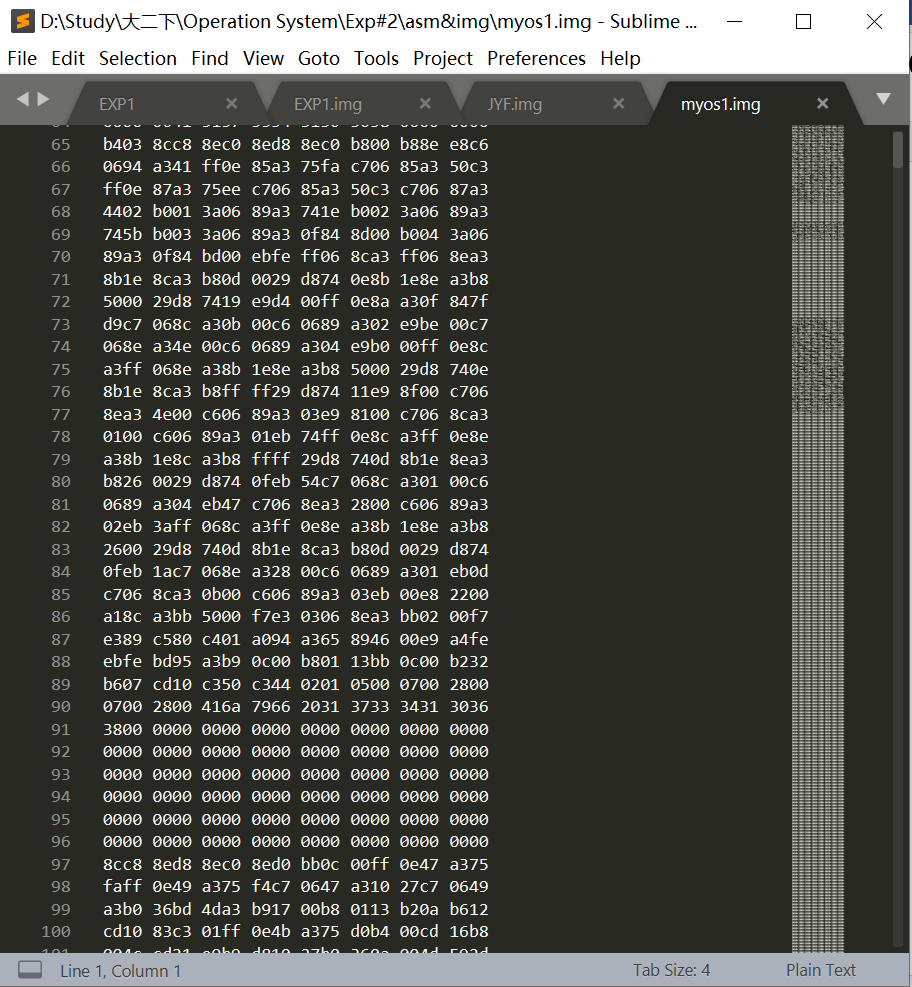
p1-p4为4个程序，在写入装载引导的软盘之前，都进行过独立的测试。附件内的程序文件org值以已改为引导程序设定的偏移量，若要测试，请手动将org偏移值修改为07c00h，重新编译制作软盘映像。

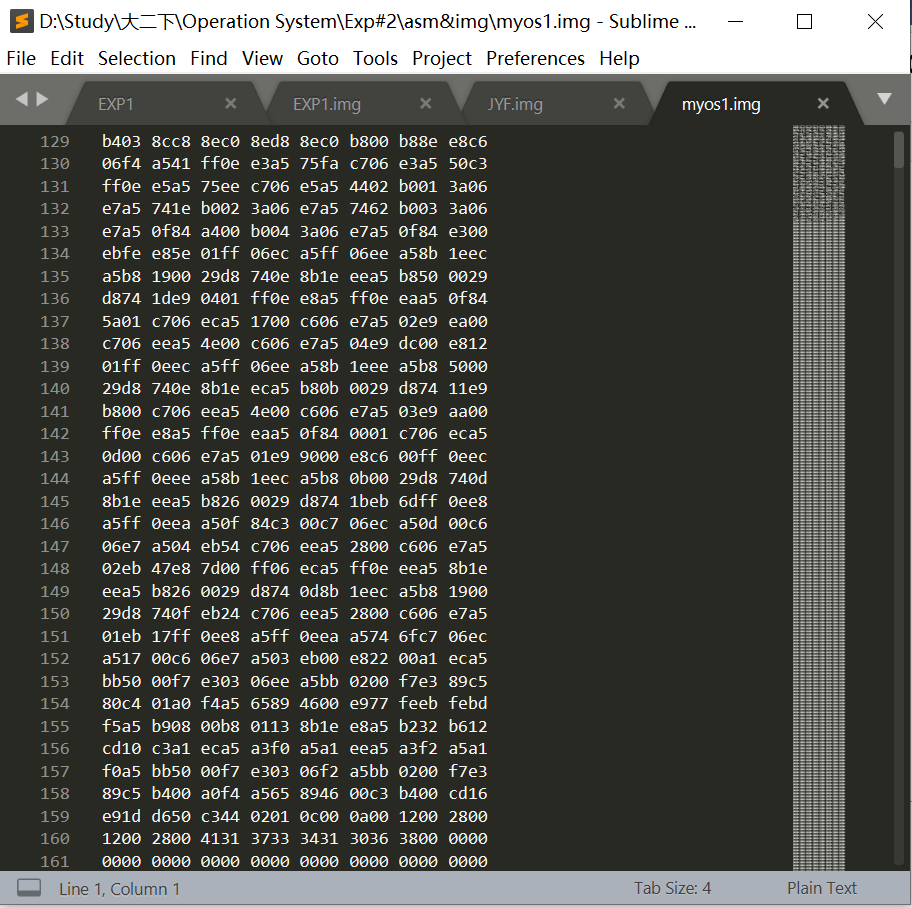
映像写入：

直接在myos1的软盘映像文件进行操作，将程序1、2、3、4分别写入第2、3、4、5个扇区，对应的启示地址为513、1025、1537、2049字节。先将4个程序正确设定内存的偏移值，编译成功后将对应的数据复制即可。

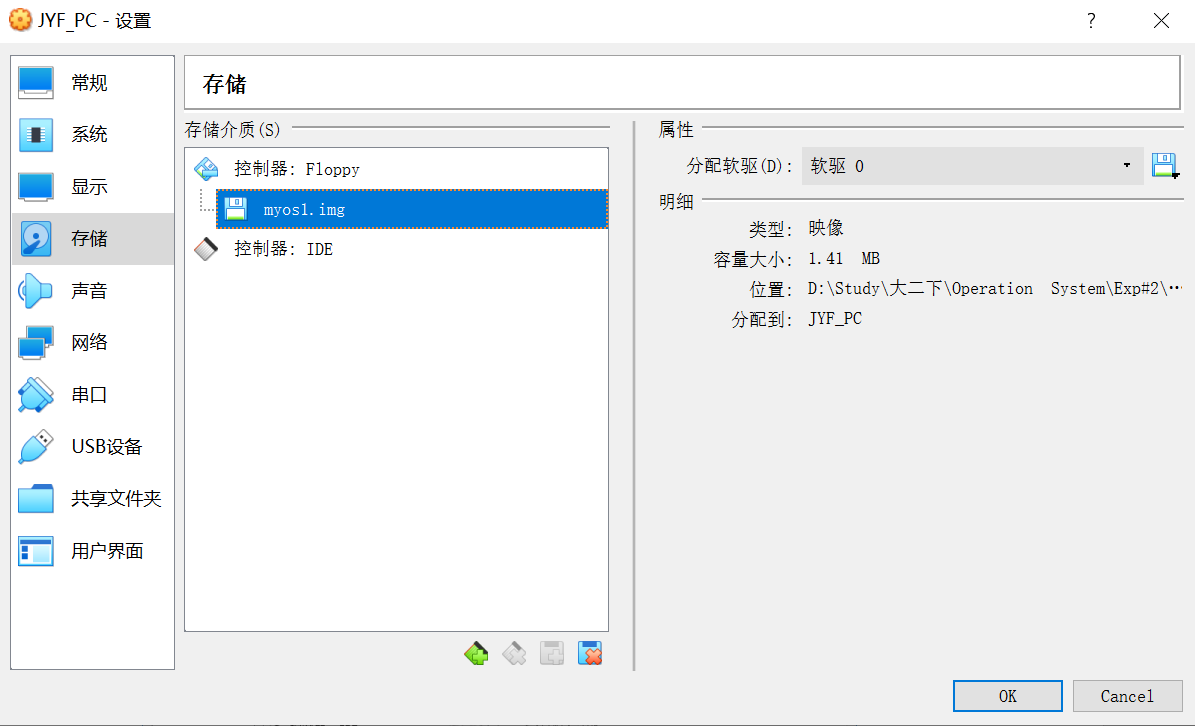








虚拟机设置



**【实验总结】**

该实验是以实验一已经老师给出的样例代码为基础，需要编写一个引导程序，使得操作系统可以根据用户输入信息来选择加载内存中不同的程序运行。实验的关键在于理解监控程序运行的原理。在将所有程序编译出的二进制数据放入软盘后，机器从软盘启动，会根据汇编代码指定的org值，将程序数据放在内存的不同位置，然后默认从07c00h启动第一个程序，所以引导程序一定要将org设置为07c00h。而剩余的待执行程序则根据汇编代码，放在内存的其他区域。通过jmp指令，可以从引导程序跳到相应的程序并执行。要注意，在设定不同程序的偏移地址时，地址要足够大，以确保能够容纳下整个程序，同时还要保证每个程序大小不超过512字节，这样才可以放在同一个扇区中，方便执行指令。

本次实验困难的地方，是最后将不同程序的机器码整合到同一个软盘映像中的步骤，本打算使用notepad++，安装插件后进行二进制文件的操作，后来发现实现起来并不容易，对文件编辑操作的支持也并不友好，遂改为使用sublime修改二进制、磁盘映像文件。由于单独测试每个程序时的org偏移量与作为被引导的层序的偏移量不同，所以每次测试成功后都要修改org值重新编译。另外，设置偏移量也需要考虑清楚，不能随意设定一个值。自己设定的偏移量分别为0A100h、0A200h、0A300h、0A400h。

、