操作系统实验报告

实验二：加载程序的监控系统的设计

学院：　　数据科学与计算机学院

班级：　　16级计算机科学与技术 教务2班

姓名：　　郑映雪

学号：　　16337327

实验时间：　　2018.3.15~2018.3.18

一、实验目的

掌握监控系统的原理，并设计一个监控系统，功能为运行不同的程序并可以返回监控系统。

二、实验要求

1、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。

2、修改参考原型代码，允许键盘输入，用于指定运行这四个有输出的用户可执行程序之一，要确保系统执行代码不超过 512 字节，以便放在引导扇区

3、自行组织映像盘的空间存放四个用户可执行程序

三、实验方案

1、相关原理:

a.监控程序

教材中提到，监控程序的使用是简单批处理方案的中心思想。通过监控系统，用户不再直接访问机器，相反，用户把卡片或磁带中的作业提交给计算机操作员，由操作员把这些作业按顺序组织成批，并将整个批作业放在输入设备上，供监控程序使用。每个程序完成处理后返回监控程序，同时监控程序自动加载下一程序。

在本实验中，监控程序作为主引导程序放在首扇区，偏移量为 7c00h。

b.BIOS 原理

老师提供了一个介绍 BIOS 的文档，文中提及，BIOS 是英文"Basic Input Output System"的缩略语，直译过来后中文名称就是"基本输入输出系统"。其主要功能是为计算机提供最底层的、最直接的硬件设置和控制。而在本实验中，多处需要用到 BIOS 中的中断程序。

c. 监控程序调用用户程序，结束后返回

首先，将用户程序编写好，写入软盘的第二、三……个扇区。然后利用 BIOS 的 13H 号中断，使用磁盘读写功能，将用户程序从扇区中读出。具体操作如下图：

读扇区	13H	02H	AL：扇区数(1~255) DL：驱动器号(0 和 1 表示软盘，80H 和 81H 等表示硬盘或 U 盘) DH：磁头号(0~15) CH：柱面号的低 8 位 CL：0~5 位为起始扇区号(1~63)，6~7 位为硬盘柱面号的高 2 位(总共 10 位柱面号，取值 0~1023) ES：BX：读入数据在内存中的存储地址	返回值： ■ 操作完成后 ES：BX 指向数据区域的起始地址 ■ 出错时置进位标志 CF=1，错误代码存放在寄存器 AH 中 ■ 成功时 CF=0、AL=0

在使用中断时，我们按上表设置好各个参数，但要特别注意，磁头号、柱面号起始是 0，扇区号的起始是 1。我们既然将监控程序放在了引导扇区，即第一个扇区，那接下来要调用的几个程序当然是从第 2 个扇区开始读起了。中断开始后，对应扇区的用户程序将会读入我们提前设置好的内存中。然后让计算机跳转到用户程序所存放的内存地址，从而开始运行用户程序。用户程序运行中，计算机开始以非阻塞的方式判断是否有输入，如果输入为退出指令，则返回到监控程序。具体对于键盘输入的原理阐述详见下一条。

d. 键盘输入跳转的判断

本实验中还使用了读取键盘输入的 BIOS 的 16H 中断。具体操作如下表：

读下一个按键	16H	00H	无具体参数	返回值：AL=ASCII 码 AH=扩展码/扫描码

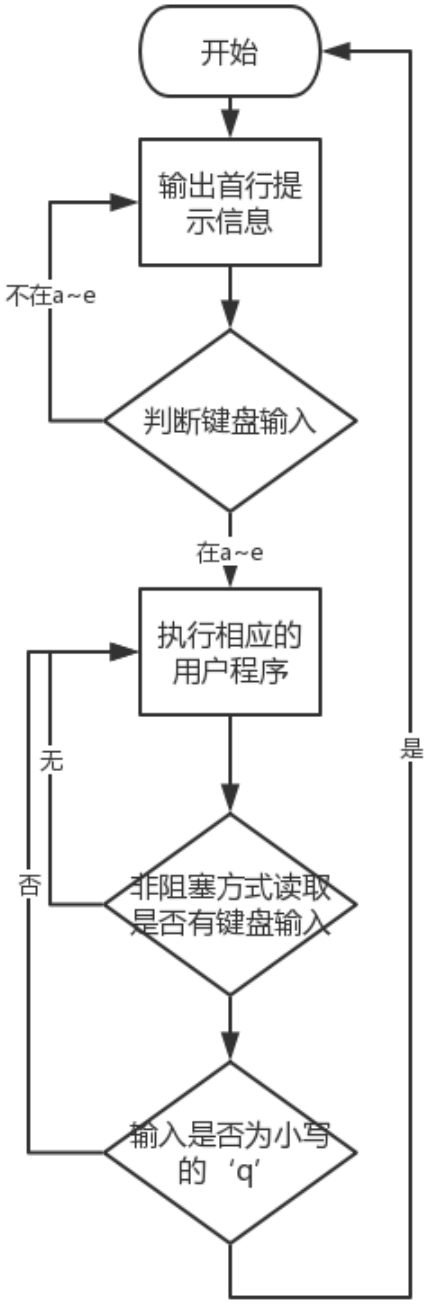
当监控程序选择要运行哪个程序时，我们进行输入，通过判断 AL 里的 ASCII 码来判断输入的内容指向哪一个用户程序。

另一个非阻塞式读取键盘的方法是调用 BIOS 的 01 号（普通键盘）或 11

号（扩展键盘）功能的 16H 中断，来读取键盘是否有输入，并将结果放入 ZF 标志位中。通过对 ZF 的判断计算机可以判断是否有键盘输入。将 00 号功能与 01 号功能结合起来，我们可以实现从用户程序返回监控程序。

2、程序流程

下面是监控程序运行的流程图：



3、程序关键模块

a.调用 BIOS 的 10h 中断，以显示第一行提示字符。这段代码修改自老师的 PPT。在本实验中，它被修改多次以用于显示不同的字符串。

```
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, MessageLength   ; CX = 串长
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
```

b.键盘判断输入及后续模块。我在网上查阅了 BIOS 中断大全后使用下列代码实现了键盘判断输入的功能。这段代码在本实验中也出现两次，以实现进入用户程序和退出返回监控程序。在用户程序中，我对这段进行了修改，使用了 01H 的功能，用非阻塞的模式判断当前有没有输入，这样也不会影响子程序字母移动的过程。

```
mov ah,00h
int 16h
mov ah, 0eh
mov bl, 0
int 10h    ;读取键盘输入

cmp al,'a'
jl Start
cmp al,'e'
jg Start      ;如果输入的字符是除了 a~e（注意是小写）以外的字符，就回到开头，
继续等待输入
```

c.读取磁盘模块，利用了 BIOS 的 13h 功能，在老师的代码上修改成了跳转多个程序的指令。

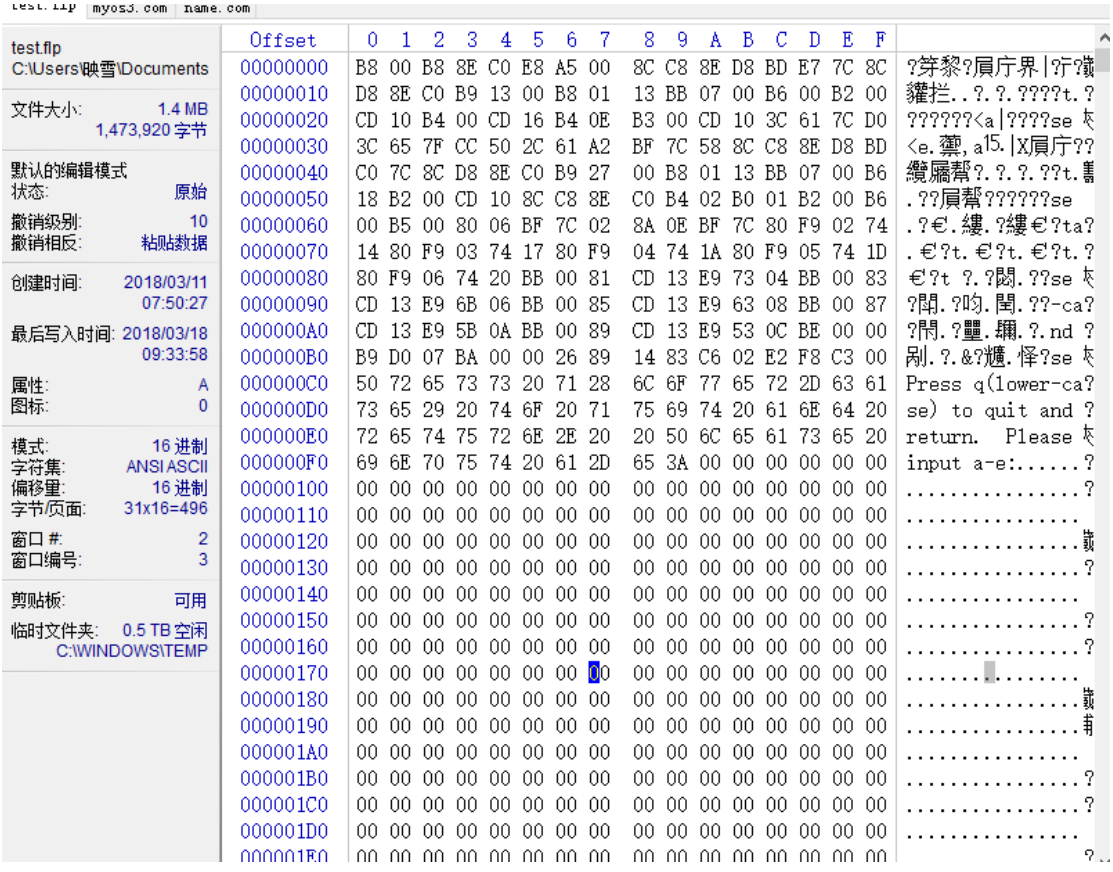
```
        mov ax,cs                       ;段地址 ； 存放数据的内存基地址
        mov es,ax                       ;设置段地址（不能直接 mov es,段地址）

        mov 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
        add byte[x],2                   ;根据前面输入计算出 x 的值，以这个值来判断应该读取
第几扇区
        mov cl,byte[x]
        cmp cl,2            ;根据前面输入计算出 x 的值，以这个值来判断应该跳转到哪个子
程序
        jz j1
        cmp cl,3
        jz j2
        cmp cl,4
        jz j3
        cmp cl,5
        jz j4
        cmp cl,6
        jz j5
j1:
        mov bx, OffSetOfUserPrg1
        int 13H ;
        jmp OffSetOfUserPrg1
j2:
        mov bx, OffSetOfUserPrg2
        int 13H ;
        jmp OffSetOfUserPrg2
j3:
        mov bx, OffSetOfUserPrg3
        int 13H ;
        jmp OffSetOfUserPrg3
j4:
        mov bx, OffSetOfUserPrg4
        int 13H ;
        jmp OffSetOfUserPrg4
j5:
        mov bx, OffSetOfUserPrg5
        int 13H ;
        jmp OffSetOfUserPrg5
```

d. 清屏模块。因为从用户程序跳转至监控程序时如果屏幕没有全清，则会出现之前子程序的显示内容还出现在屏幕上，所以我们需要 call 清屏模块。

```
clear:      ;清屏模块
        mov si,0
        mov cx,80*25    ;逐个将屏幕上每个字符 ASCII 码置零。用 cx 计数。
        mov dx,0
clear1:
        mov [es:si],dx    ;逐个置零
        add si,2
        loop clear1 ;
        ret
```

四、实验过程

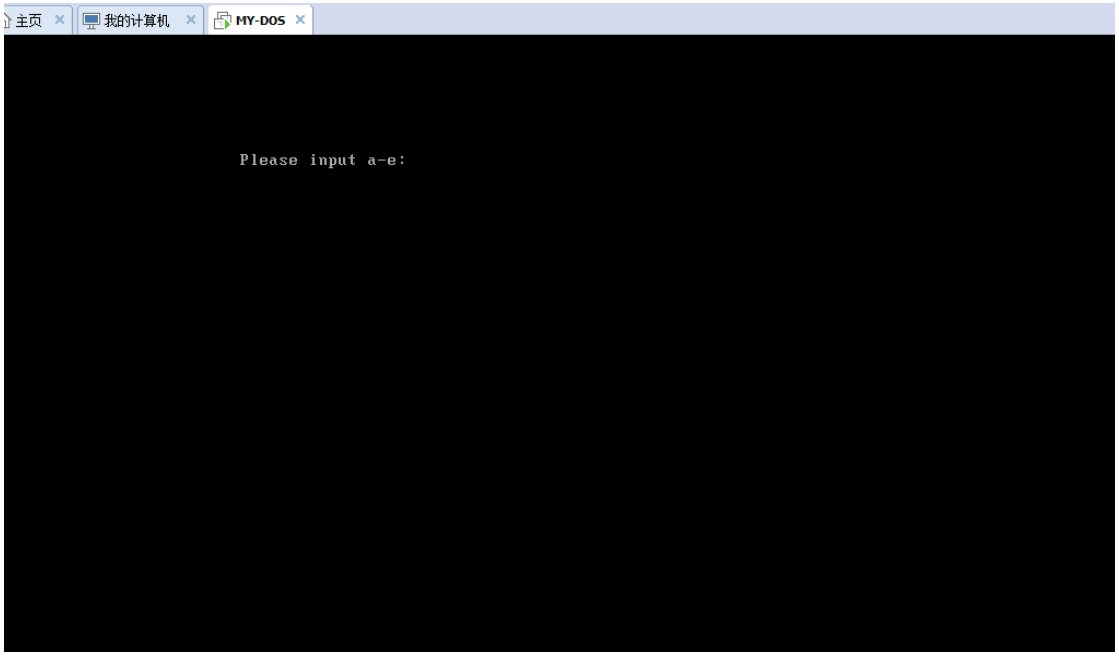
1、将监控程序编译后的二进制码复制进一张空软盘的首扇区



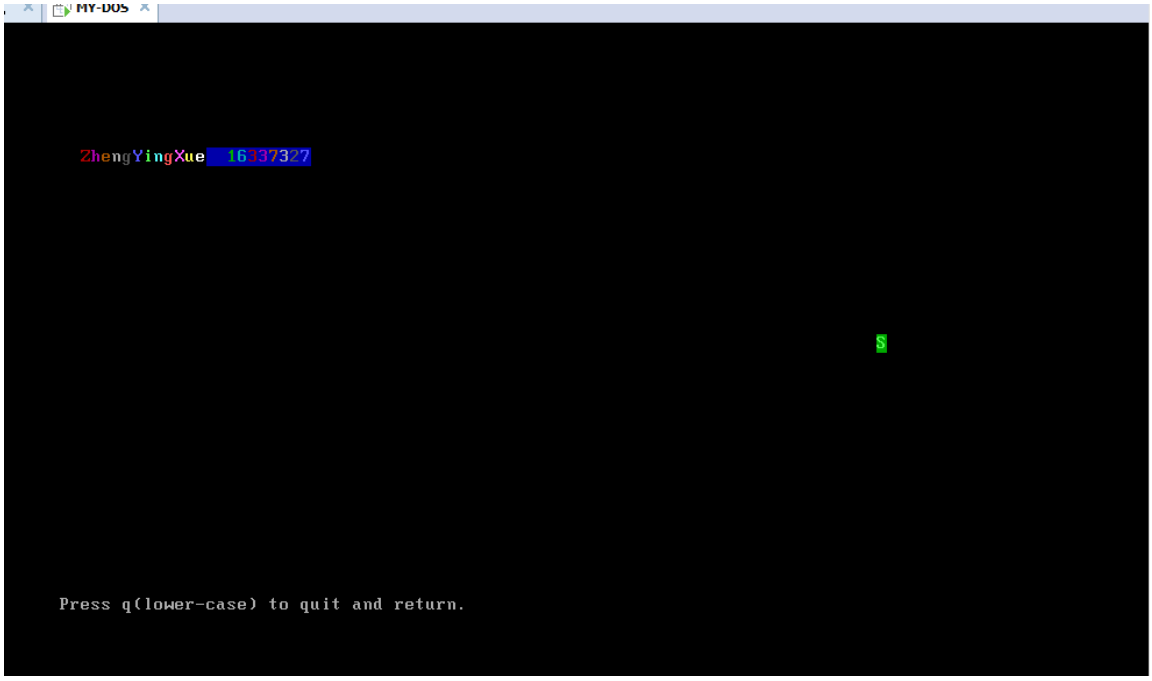
2、将用户程序编译后按顺序将它们的二进制码分别放入第 2、3、4、5、6 个扇区。

```
00000200  8C C8 8E C0 8E D8 8E C0  C6 06 F5 82 53 C6 06 F6
00000210  82 20 C6 06 F7 82 FA C7  06 D9 82 17 00 FF 0E D4
00000220  82 75 FA C7 06 D4 82 50  C3 FF 0E D6 82 75 EE B8
00000230  00 B8 8E E8 C7 06 D4 82  50 C3 C7 06 D6 82 44 02
00000240  B4 07 A0 F6 82 65 89 46  00 B0 01 3A 06 D8 82 74
00000250  1E B0 02 3A 06 D8 82 74  5D B0 03 3A 06 D8 82 0F
00000260  84 9A 00 B0 04 3A 06 D8  82 0F 84 D4 00 EB FE FF
00000270  06 F1 82 FF 06 F3 82 8B  1E F1 82 B8 0C 00 29 D8
00000280  74 0E 8B 1E F3 82 B8 50  00 29 D8 74 16 E9 F5 00
00000290  C6 06 F7 82 9A C7 06 F1  82 0A 00 C6 06 D8 82 02
000002A0  E9 E2 00 C6 06 F7 82 AA  C7 06 F3 82 4E 00 C6 06
000002B0  D8 82 04 E9 CF 00 FF 0E  F1 82 FF 06 F3 82 8B 1E
000002C0  F3 82 B8 50 00 29 D8 74  0E 8B 1E F1 82 B8 FF FF
000002D0  29 D8 74 16 E9 AE 00 C6  06 F7 82 F9 C7 06 F3 82
000002E0  4E 00 C6 06 D8 82 03 E9  9B 00 C6 06 F7 82 EF C7
000002F0  06 F1 82 01 00 C6 06 D8  82 01 E9 88 00 FF 0E F1
00000300  82 FF 0E F3 82 8B 1E F1  82 B8 FF FF 29 D8 74 0D
00000310  8B 1E F3 82 B8 28 00 29  D8 74 14 EB 68 C6 06 F7
00000320  82 0F C7 06 F1 82 01 00  C6 06 D8 82 04 EB 56 C6
00000330  06 F7 82 9E C7 06 F3 82  2A 00 C6 06 D8 82 02 EB
00000340  44 FF 06 F1 82 FF 0E F3  82 8B 1E F3 82 B8 28 00
00000350  29 D8 74 0D 8B 1E F1 82  B8 0C 00 29 D8 74 14 EB
00000360  24 C6 06 F7 82 A1 C7 06  F3 82 2A 00 C6 06 D8 82
00000370  01 EB 12 C6 06 F7 82 F5  C7 06 F1 82 0A 00 C6 06
00000380  D8 82 03 EB 00 31 C0 A1  F1 82 BB 50 00 F7 E3 03
00000390  06 F3 82 BB 02 00 F7 E3  89 C5 8A 26 F7 82 A0 F5
000003A0  82 65 89 46 00 8B 0E D9  82 BE DA 82 BF 02 00 B4
```

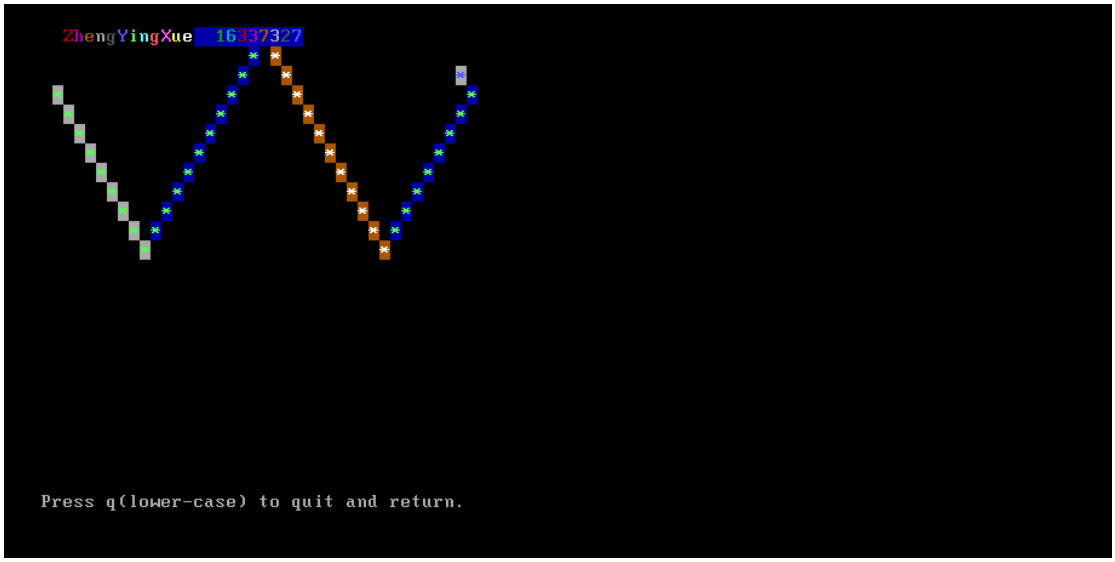
3、用虚拟机运行该软盘。可以看到如下的监控程序界面。



4、监控程序提示输入 a~e 中的字母。输入 a 进入第一个用户程序，第一象限字符“S”运动的的画面。同时左上角显示变色的个人信息，左下角显示按下小写“q”字母则退出，回到监控程序的提示。

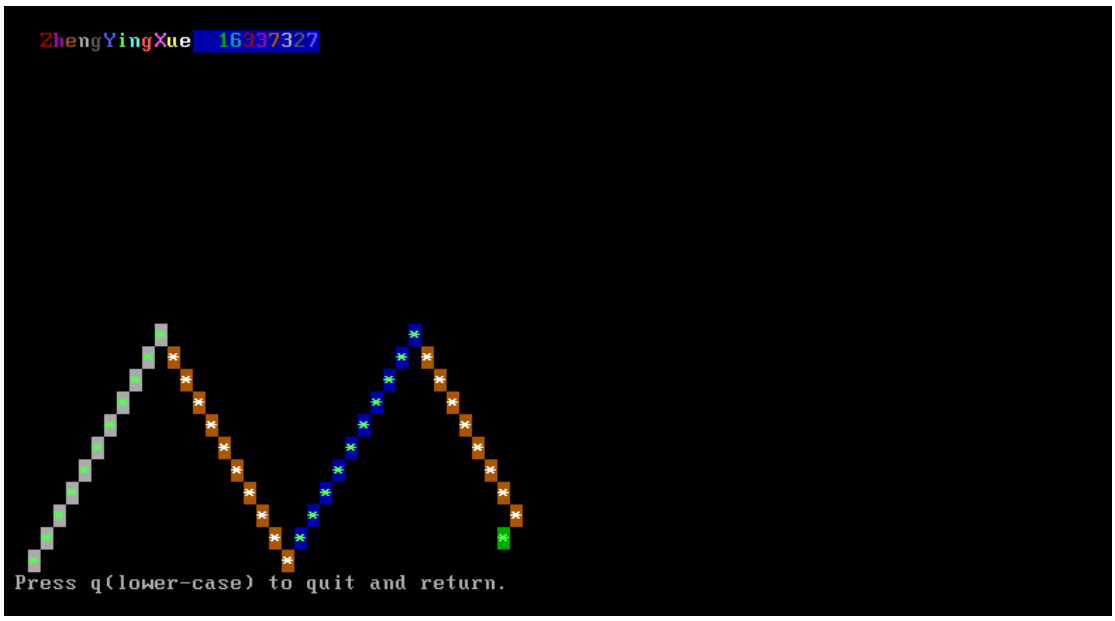


5、按下字母 q 回到监控程序，输入 b 进入第二个用户程序，第二象限字符“*”连续运动的画面。同时左上角显示变色的个人信息，左下角显示输入小写 q 则返回的信息。

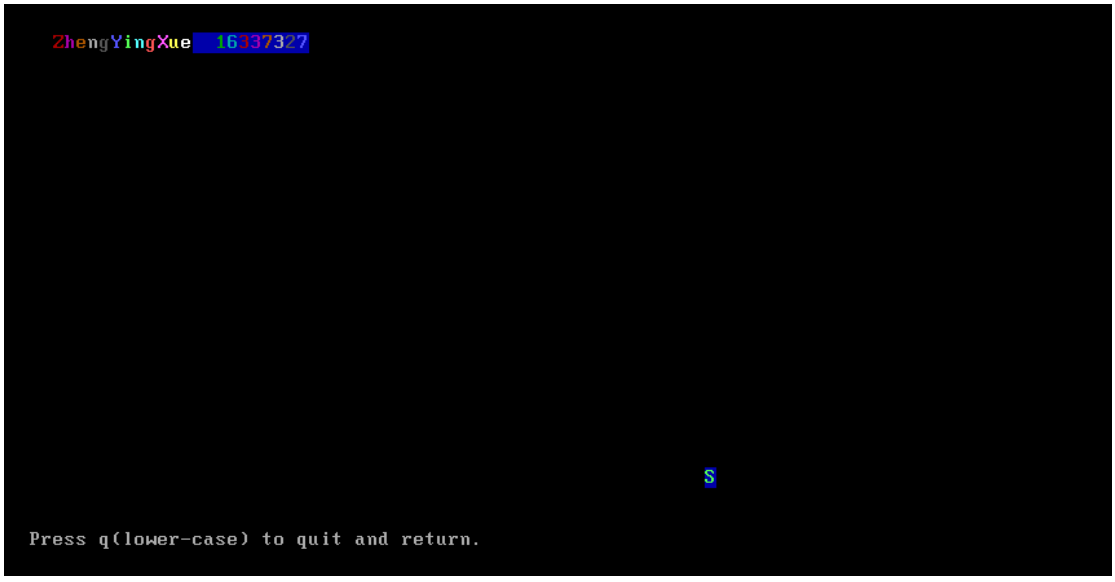


6、按下字母 q 回到监控程序，输入 c 进入第三个用户程序，第三象限字符

“*”连续运动的画面。同时左上角显示变色的个人信息，左下角显示输入小写 q 则返回的信息。



7、按下字母 q 回到监控程序，输 d 进入第四个用户程序，第四象限字符“S”连续运动的画面。同时左上角显示变色的个人信息，左下角显示输入小写 q 则返回的信息。



8、按下字母 q 回到监控程序，输入字母 e，进入第五个用户程序，在屏幕中央显示我的个人信息，同时左下显示输入小写 q 则返回的信息。

五、实验总结

这次实验是基于上次的实验的，所以会比第一次“连要干什么都一头雾水”的情况要好得多。这次的实验主要分为两个部分。

第一个部分，是修改上周实验的代码，由在全屏幕进行运动变为分别在四个象限进行运动。

四个象限是比较好改的，只要修改一下边界值就行了（如第二象限，从左上角射入，x 只要碰到 25 整除 2=12 就转向，y 只要碰到 80/2=40 就转向即可）。

另外，上星期的代码我只做到了碰壁变色，且保留了之前的轨迹，时间久了就满屏花花绿绿，看着不好看。这星期我对上周的代码进行了改进，在四个象限中，两个象限的字符保留了轨迹。另外两个象限进行了轨迹消除，看上去实际上只是一个字符在两个象限飞翔。其实轨迹消除这个点我想了很久，包括清屏，但是实际操作总是不如人意。正当我苦恼的时候，我突然发现——只要让下一个字母显示之前，在还没有改变的坐标显示空格就可以了嘛！这么简单的方法，我之前没有想到，我真的是挺傻的。

然后在左上角显示个人信息时，我上周一度焦头烂额，因为按照老师显示单个字符的方法，代码长度会超过一个扇区，所以上周我显示的个人信息只是名字而已。这一次我成功将个人信息不是通过单个字符，而是通过字符串显示的方式变色显示在了左上角。

可以说以上是我对第一个实验的改进之处，并且作为了我第二个实验的用户程序。

第二个部分是本次实验的主角——监控程序。

其实监控程序的原理并不难，老师上理论课的时候就已经讲过，但是理论课的时候还是不知道怎么实现。在实验课的时候看到老师的 ppt，我才知道 BIOS 的中断有读取磁盘数据到内存的功能，这在以前计组课上时没有用过的。我通过老师给的基础原码进行修改，使用了两个 BIOS 中断以实现显示一行字符和读写软盘上的用户程序到内存。我又按照老师的 PPT 上的资料，利用 16H 号中断来读取键盘，判断要进入哪一个子程序。进入子程序之后我又在子程序中调用了一次中断，想要实现返回监控程序的功能。

这时候麻烦出现了——我停留在监控程序的界面，怎么也进不去了。我这次实验采用自底向上的方法，五个用户程序都首先测试过的，没有问题。这时候我发现一个很容易被忽略的地方——org 指令，它被用来实现程序的起始地址。我在测试五个子程序代码时，指令是 org 7c00h，但作为子程序，它就要跟监控程序保持一致才可以成功跳转。我把五个程序首行的 org 改为 8100h、8300h、8500h、8700h、8900h，每一个程序都可以成功跳转了。

然而我高兴地太早了，这时候麻烦又来了——字符显示出来以后，它不动了！我按下原本设置的字母 q 键，它返回了监控程序，可是原本的字符还在。这下我得解决两个问题了。关于字符出来以后停止不动我是这么解决的：我上网仔细查阅了 BIOS 中断的功能，发现我重复调用了的是功能号为 00H 的 16H 号中断，它实际上是一个阻塞中断，只要你不按键，它是不会让你运行下面的内容的。那么，有没有按键与否都不会影响到程序运行的呢？有！功能号为 01H 和 11H 的 16H 号中断就是一个非阻塞中断，通过 ZF 标志位来显示你有没有按键。这下问题得到了解决——只要我调用这个中断后，通过判断 ZF 标志位，若有按键则再判断按的是不是字母 q 就好啦。原本的字符还在这个问题，其实就是没有清屏。于是我写了一个清屏的代码，在监控程序开始时便调用，这样返回到监控程序时一开始便会调用清屏模块，原本的字符就不会再出现了。

问题总算是都解决了。从学习到查资料，试写代码，到排错，直到最后全部完成，我花了好几天的课余时间。可以说这次实验做得比上一次更加得心应手一

些，也许我正从一个操作系统的小白一点点进步吧。当然我还是很后悔上学期计算机组成原理的课程没有课外多学习一些汇编语言而是只学习了基本指令，导致现在每做一个实验都要查很多资料。但是时间是回不去的，只能从现在开始跟着老师的脚步一步一步来啦，看着裸机一步步掌握在自己手里，还是很开心的。

六、参考文献

1、《x86 汇编语言-从实模式到保护模式》 李忠 王晓波 余洁 著
2、《BIOS 知识》 来自老师提供的参考资料
3、bios 中断大全-zhe_wang-ChinaUnix 博客
http://blog.chinaunix.net/uid-27033491-id-3239348.html