《操作系统实验》

实验报告

（实验二）

学 院 名 称 ：	数据科学与计算机学院
专业（班级）：	17 级计算机类教务 3 班
学 生 姓 名 ：	姚森舰
学　　号 ：	17341189
时　　间 ：	2019　年　3　月　20　日

一. 实验题目

加载用户程序的监控程序

二. 实验目的

设计四个（或更多）有输出的用户可执行程序及监控程序，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，通过放在在引导扇区的监控程序加载运行，以加深对监控程序的认识和理解。

三. 实验要求

设计监控程序，显示必要的提示信息后，从引导盘的特定扇区加载一个他人开发的COM格式的可执行程序到指定的内存位置，然后启动这个程序，实现操作系统执行用户程序这一项基本功能。

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。

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

自行组织映像盘的空间存放四个用户可执行程序

四. 实验方案

实验环境：Windows10 +VMware

实验工具：NASM+Winhex+Notepad++

大概流程：用Notepad++编写汇编程序，通过nasm编译成二进制代码，通过VMware创建带软盘的虚拟机，利用winhex将编译好的监控程序的二进制代码写到虚拟机的软盘的第一个扇区，然后将用户程序一次放到紧接着的扇区，然后在Vmware上运行测试虚拟机和程序。

大概思路：在监控程序里，先清屏，显示一些提示信息，等待输入，根据输入做一个类似 switch 的语句做对应操作，即加载对应的用户程序，因为用户程序比较简单，所以写的时候，给了每个程序几秒的时间，用户程序执行完后在跳回 7C00h，再次重复执行监

控程序。

关键代码：

1.监控程序：

```
mov ax,0600h                ;清窗口
mov ch,0                    ;左上角行
mov cl,0                    ;左上角列
mov dh,24                   ;右下行
mov dl,79                   ;右下列
mov bx,0007h                ;黑底白
int 10h
```

调用int 10h清屏功能

```
read:
    mov cl,al
    inc cl
    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
AfterRun:
    jmp bx
```

调用int 13h将对应扇区的程序加载到内存

```
    mov ax,0
loop1:
    mov ah,00h                  ;检测键盘
    int 16h                     ;字符在al
    cmp al,0
    jz loop1                    ;字符在al

    mov bp,152
    mov ah,07h                  ;
    mov word[gs:bp],ax
    add bp,2
    sub al,48
    mov bx, OffSetOfUserPrg1    ;偏移地址;
    cmp al,1
    jz read
    cmp al,2
    jz p2
    cmp al,3
    jz p3
    cmp al,4
    jz p4
p2:
    add bx,200h
    jmp read
p3:
    add bx,400h
    jmp read
p4:
    add bx,600h
```

调用int 16h检测键盘，直到有有效输入后跳转到 对应的用户程序

```
    mov cx,4                ;显示几行
    mov ax, mess1
    mov si,ax
    mov bp,2400+84          ;2个字节
row1:
    push cx
    mov cx,8                ;一行的字符数

char1:
    mov ah,07h
    mov al,[si]             ; AL=显示字符值
    mov word[gs:bp],ax
    inc si
    add bp,2
    loop char1

    add bp,144              ;(80-字符数)*2
    pop cx
    loop row1
```

通过循环显示定义的字符串

```
times 510-($-$$) db 0
db 0x55,0xaa
```

将扇区未占用的空间填满0，最后以0x55，0xaa结尾，表示到达了引导扇区的最后。

2．用户程序1：

```
    mov cx,25           ; 进度条长度
loop1:                  ;延时
    push cx
    mov ah,86h
    mov cx,02h
    mov dx,0x8480
    int 15h
    pop cx

    mov dx,cx
    test dx,01h         ;偶数跳show2
    jz show2
```

```
show1:                  ;loading..
    push cx
    mov al,1
    mov ah,13h
    mov bl,1fh
    mov bh,0
    mov dh,11           ;行
    mov dl,14           ;列
    mov cx,11           ;长度
    mov bp,strc
    int 10h
    pop cx
    jmp show
```

先调用15h延时，通过cx是奇是偶来决定显示，“LOADING..” 或 ”LOADING….”，从而显示出动态的效果。延时是用来显示进度条的，所谓进度条，其实就是一些绿色背景的空格组成的长条形（让我们假装它是一个真实的进度条）。

3. 用户程序2：

```
loop1:
    push cx
    mov ah,6
    mov al,0
    mov ch,3
    mov cl,40
    mov dh,12
    mov dl,79
    mov bh,17h
    int 10h

    mov ah,86h
    mov cx,01h
    mov dx,4240h
    int 15h
```

```
    mov ah,6
    mov al,0
    mov ch,3
    mov cl,40
    mov dh,12
    mov dl,79
    mov bh,27h
    int 10h
    jmp con

helpout:
    pop cx
    loop loop1
    jmp 7c00h

con:
    mov ah,86h
    mov cx,01h
    mov dx,4240h
    int 15h
```

多次用不同背景色清屏，清屏后延时短时间，效果就是不同颜色填充完整个屏幕，闪动，让用户如同身处舞厅一般。其中会遇到的问题是，loop跳转范围太长，无法直接跳转，所以在程序引入一个helpout标号，起到跳板的作用，先跳到helpout处，再跳回loop处，或者返回监控程序。(关灯效果更好)

4. 用户程序3：

```
    mov cx,12       ;循环12行

loop1:              ;外层循环

    push cx         ;保护
    mov dl,0        ;属性，列
    mov cx,8        ;循环8次

    push cx
    push dx
    push ax         ;int 13h
    mov ah,86h
    mov cx,01h
    mov dx,0x8480
    int 15h
    pop ax
    pop dx
    pop cx

loop2:              ;内层循环
    push cx         ;保护
    mov cx,5        ;字符串长度
    int 10h         ;BIOS中断调用
    inc bl          ;属性值增加1
    add dl,5        ;列值增加5
    pop cx
    loop loop2      ;内层循环

    pop cx
    inc dh          ;改变行，行增加1
    loop loop1      ;外层循环
```

每次都调用int 13h中断，内循环8次，一个字符串5byte，外循环12次即12行，每次内循环都会增加bl的值，即改变了字符显示的属性，从而得到彩色显示的效果。

5. 用户程序4：

```
datadef:
str0 db "                  _oo0oo_                "
str1 db "                 o88888888o              "
str2 db "  17341189       88' . '88               "
str3 db "    Yao          (|  - -  |)              "
str4 db "    Sen           0\  =  /0               "
str5 db "    Jian        ___/`---'\___             "
str6 db "              .' \\|  :  |// '.           "
str8 db "           / _||||| /:\ |||||- \          "
strq db "          | \_|  ''\---/''  |_/ |         "
stri db "                                          "
strp db " ~~The buddha bless you away from bugs~~"
```

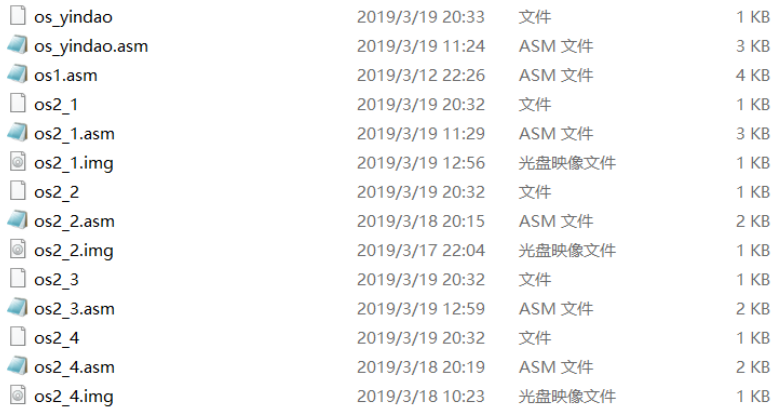
把字符串依次显示，和前面显示字符串的方式相同，不再赘述，显示完成后，调用int 15h 延时几秒后返回监控系统。

五. 实验过程与思想

1.安装 VMware 并按要求创建一个无操作系统的裸机。

2. 编写汇编程序：
关键代码见实验方案，具体代码见代码文件

3. 编译：



利用 Winhex 将监控程序代码文件生成的二进制文件粘贴到所用软盘的第一个扇区，再依次粘贴用户程序的二进制文件到紧接着的扇区。值得注意的是不要改变软盘大小，同时引导程序也不要大于 512B，否则部分程序指令无法加载，使得程序无法正确运行。

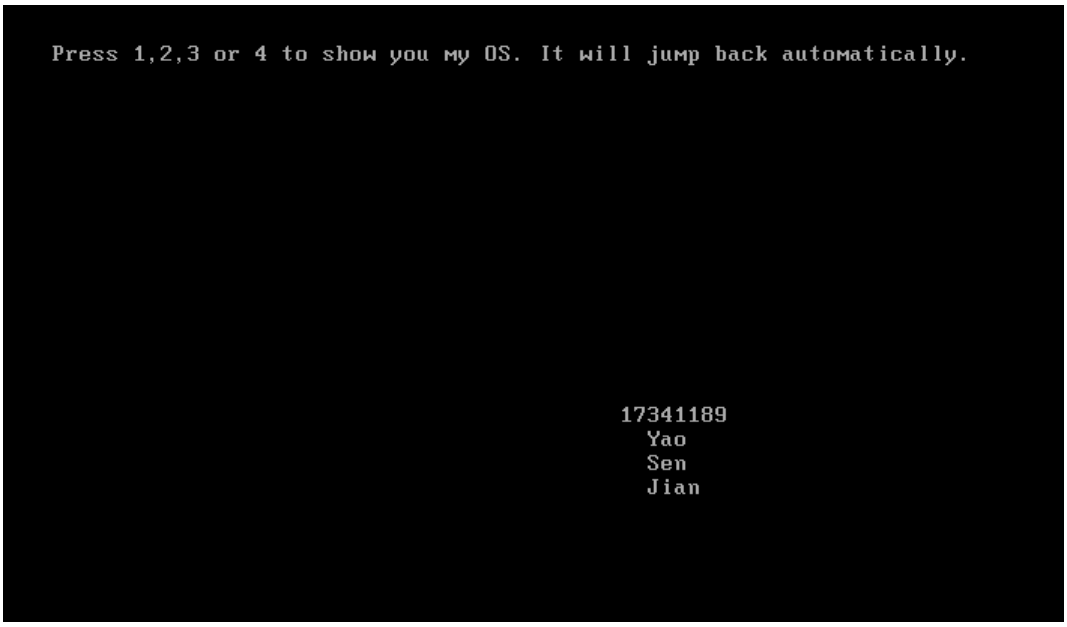
以下是完成后的软盘的部分截图：

PC_OS2.flp

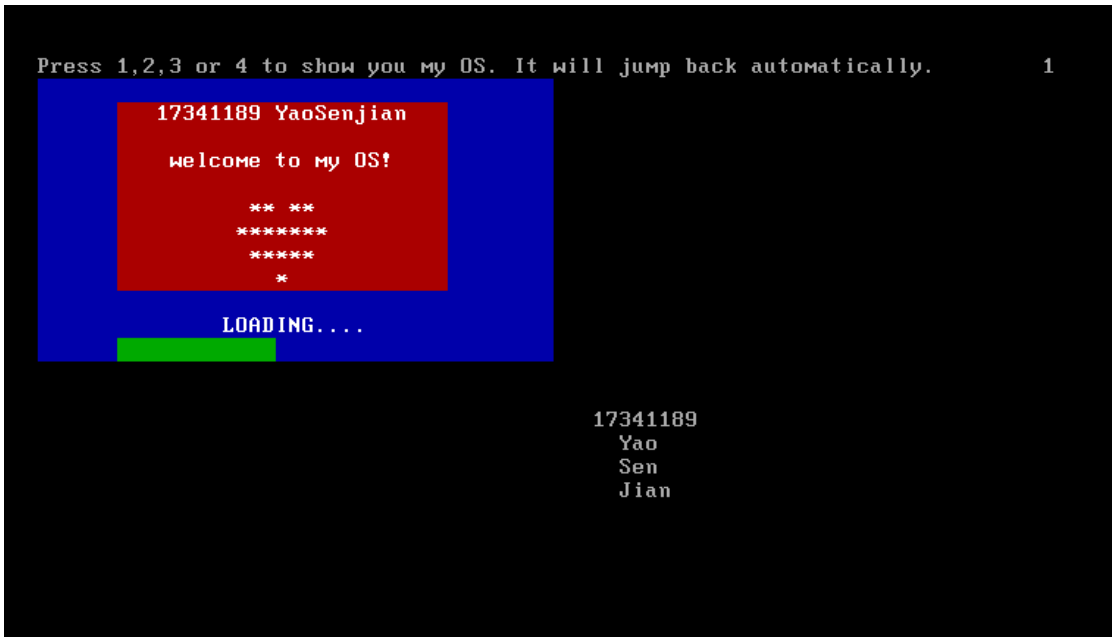
Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI ASCII
00000000	8C	C8	8E	D8	B8	00	06	B5	00	B1	00	B6	18	B2	4F	BB	ŒÈŽØ¸ µ ± ¶ ²O»
00000010	07	00	CD	10	BD	A3	7C	8C	D8	8E	C0	B9	42	00	B8	01	Í ½£|ŒØŽÀ¹B ¸
00000020	13	BB	07	00	B6	00	B2	00	CD	10	BA	00	B8	8E	EA	B9	» ¶ ² Í ° ¸Žê¹
00000030	04	00	B8	E5	7C	89	C6	BD	B4	09	51	B9	08	00	B4	07	¸å|‰Æ½´ Q¹ ´
00000040	8A	04	65	89	46	00	46	83	C5	02	E2	F2	81	C5	90	00	Š e‰F FƒÅ âò Å
00000050	59	E2	E7	B8	00	00	B4	00	CD	16	3C	00	74	F8	BD	98	Yâç¸ ´ Í < tø½˜
00000060	00	B4	07	65	89	46	00	83	C5	02	2C	30	BB	00	A1	3C	´ e‰F ƒÅ ,0» ¡<
00000070	01	74	1E	3C	02	74	08	3C	03	74	0A	3C	04	74	0C	81	t < t < t < t
00000080	C3	00	02	EB	0C	81	C3	00	04	EB	06	81	C3	00	06	EB	Ã ë Ã ë Ã ë
00000090	00	88	C1	FE	C1	B4	02	B0	01	B2	00	B6	00	B5	00	CD	ˆÁþÁ´ ° ² ¶ µ Í
000000A0	13	FF	E3	50	72	65	73	73	20	31	2C	32	2C	33	20	6F	ÿãPress 1,2,3 o
000000B0	72	20	34	20	74	6F	20	73	68	6F	77	20	79	6F	75	20	r 4 to show you
000000C0	6D	79	20	4F	53	2E	20	49	74	20	77	69	6C	6C	20	6A	my OS. It will j
000000D0	75	6D	70	20	62	61	63	6B	20	61	75	74	6F	6D	61	74	ump back autcmat
000000E0	69	63	6C	79	2E	31	37	33	34	31	31	38	39	20	20	59	icly.17341189 Y
000000F0	61	6F	20	20	20	20	20	53	65	6E	20	20	20	20	20	4A	ao Sen J
00000100	69	61	6E	20	20	00	00	00	00	00	00	00	00	00	00	00	ian
00000110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000001B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000001C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000001D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000001F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	55	AA	Uª
00000200	31	C0	8C	C8	8E	C0	8E	D8	8E	C0	B8	00	B8	8E	E8	B4	1ÀŒÈŽÀŽØŽÀ¸ ¸Žè´
00000210	06	B0	00	B5	01	B1	00	B6	0C	B2	26	B7	17	CD	10	C7	° µ ± ¶ ²&· Í Ç
00000220	06	BE	A1	06	00	C7	06	BC	A1	0C	00	B9	08	00	B8	C1	¾¡ Ç ¼¡ ¹ ¸Á
00000230	A1	89	C6	BD	4C	01	51	B9	19	00	B4	4F	8A	04	65	89	¡‰Æ½L Q¹ ´OŠ e‰
00000240	46	00	46	83	C5	02	E2	F2	83	C5	6E	59	E2	E8	B9	19	F FƒÅ âòƒÅnYâè¹

运行测试：

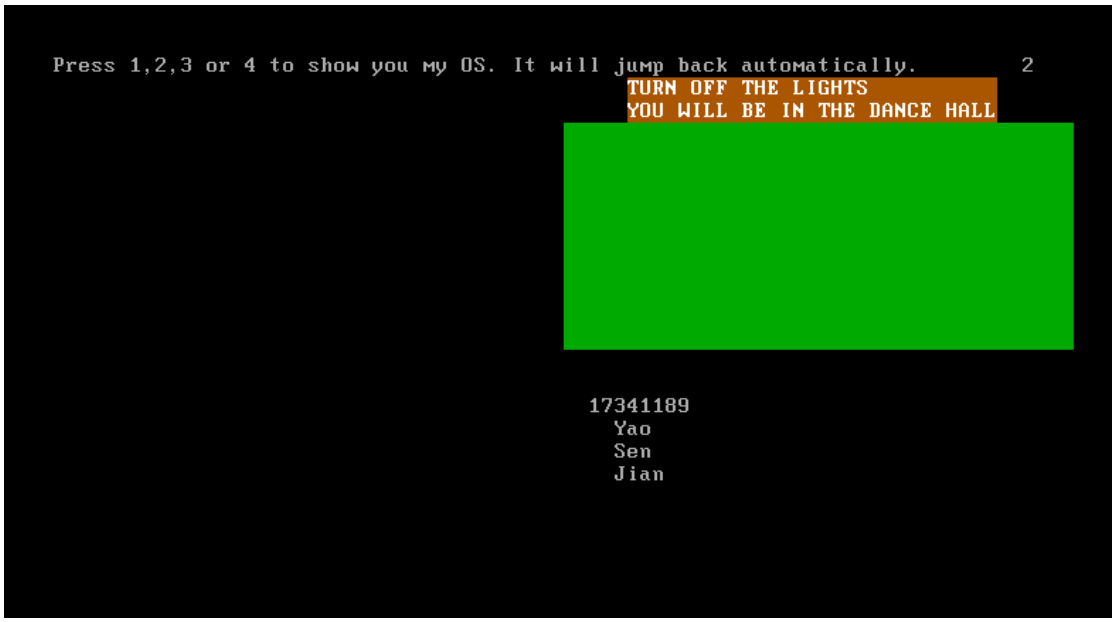
软盘编辑完成后，就可以启动虚拟机，启动后将运行我们的监控程序，如下图：



按下“1”后，执行用户程序 1,进度条加载完成后回到监控程序：



按下“2”后，执行用户程序 2，屏幕会来回闪烁，几秒后回到监控程序：



按下“3”后，执行用户程序 3,显示不同颜色的“SYSU”字符串，完成后回到监控程序：

按下“4”后，执行用户程序 4，显示如图，延时几秒后回到监控程序：

六. 实验心得和体会

通过这个实验，我对监控程序有了更深的理解，对监控程序调用用户程序并返回这个过程有了大致的了解，也熟悉了一些 BIOS 的功能调用，在这次实验里也遇到了很多问题，但

我感觉这些问题大多是在写用户程序时，和汇编语言和系统本身有关，单从整个流程来看，还是比较顺利。下面就是碰到的一些问题：

1. Int 16h 的问题，从一些资料中看到功能号为 00h，01h 的功能似乎没有太大的区别，但是用的时候才发现区别挺大：00h 号功能调用，从键盘读入字符送 AL 寄存器。执行时，等待键盘输入，一旦输入，字符的 ASCII 码放入 AL 中。若 AL＝0，则 AH 为输入的扩展码。 01h 号功能调用，用来**查询键盘缓冲区**，对键盘扫描但不等待，并设置 ZF 标志。若有按键操作（即键盘缓冲区不空），则 ZF＝0，AL 中存放的是输入的 ASCII 码，AH 中存放输入字符的扩展码。若无键按下，则标志位 ZF＝1。所以如果调用 01h 号功能，按下一次键后，该键就会在键盘缓冲区里，执行完对应用户程序后，系统会以为又按下了相同的键，导致重复执行同一个用户程序。由于没有找到清空键盘缓冲区的方法，直接用 00h 号功能调用更好。

2. 寄存器数据保护的问题。有些寄存器可能会重复用到，特别是 loop 时会用到 cx 寄存器和一些中断功能相关的寄存器，所以务必将要重复用到的寄存器的值压栈保护起来，同时注意 push，pop 指令应当是成对出现的。

3. nasm 错误 “label or instruction expected at start of line.”

编码的问题，UTF8 编码 ，默认是加上 BOM 的，也即是 0xFEFF 这 2 个字节在文件头部，导致 nasm 不能识别，选择 UTF-8 无 BOM 编码，问题解决。

4. “发生错误，导致虚拟 CPU 进入关闭状态。如果虚拟机外部发生此错误，则可能已导致物理机器重新启动。错误配置虚拟机、客户机操作系统中的错误或 VMware Workstation 中的问题都可以导致关闭。”

搜索这个问题，有人说是没有开启 CPU 虚拟化，但是按教程进入 BIOS 设置，发现此项是开启了的。试了一些其他方法也未能解决，最后把代码重新粘贴到一个新文件之后就解决了，但不知道原因。

5. 最初想做一个显示时间的用户程序，但是不知道为何我的 BIOS 调用 INT 1AH 读系统时间的中断时，无论如何都读不出正确的时间，只有秒数是正确的。可能是中断调用的时候在某些地方出来错，但是没有找出来，后面学习中断时可以再回过头来看看。还有其他一些点子，迫于课程紧迫，也没有太多时间去实现。

总的来说，从写代码到组织扇区，做过一遍之后就会对整个流程有个清晰的认识，踩过过了那些坑，花费了时间，留下了印象，才能减小以后犯错的概率，为以后的实验继续积累经验。实验有点难，但完成后受益匪浅。

七. 参考资料

[1] BIOS 功能调用之滚屏与清屏

[2] BIOS 中断大全

[3]显存文本模式详解

[4]键盘 I/O 中断调用（INT 16H）