

操作系统实验报告

17341190 叶盛源

2019 年 3 月 21 日

Contents

1 实验题目:	2
2 实验目的:	2
3 实验要求:	2
4 实验方案:	2
4.1 实验环境:	2
4.2 实验工具:	2
4.3 实验思想及实验过程:	3
5 实验心得和总结:	9
6 参考文献:	10

1 实验题目:

加载用户程序的监控程序

2 实验目的:

设计四个（或更多）有输出的用户可执行程序,再设计一个监控程序放在首扇区作为裸机引导程序，再将这四个程序分别放在监控程序之后的四个扇区。利用bios功能调用，实现键盘输入指定运行这四个有输出的用户可执行程序之一。

3 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) 修改参考监控程序代码，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

4 实验方案:

4.1 实验环境:

- 1) 实验运行环境：Windows10
- 2) 虚拟机软件：VMware Function

4.2 实验工具:

- 1) 汇编语言：NASM

- 2) 文本编辑器：VScode
- 3) 软盘操作工具：WinHex

4.3 实验思想及实验过程：

- 1) 安装主要工具：

在VMware官网下载虚拟机的体验版安装包，再在网上找到破解的密钥输入永久破解。接着按照要求安装并创建一个无操作系统的裸机。

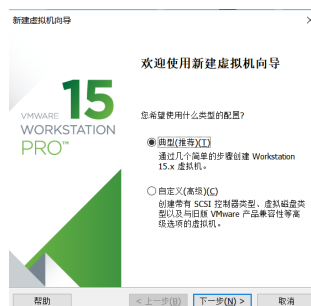


Figure 1: 安装VMware虚拟机

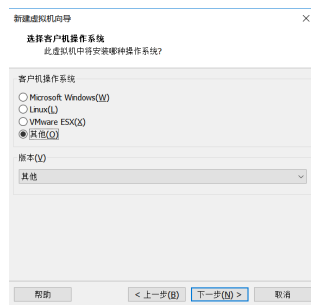


Figure 2: 创建无操作系统裸机



Figure 3: 为裸机配置属性

2) 设计监控程序:

以老师给的监控程序模板为基础进行创作，首先需要增加一段清屏功能，能让监控程序和子程序相互切换的时候把上一次显示的内容清除干净，我们利用BIOS的功能调用int 10h的6号功能，将整个显示器的值都填为0，这样就实现了清零。具体代码如figure4，之后使用int10的1号功能输出字符串，将要显示给用户的提示语句展示在屏幕上如figure5。

```
;使用int10h的清屏功能
mov ah,6
mov al,0
mov ch,0
mov cl,0
mov dh,24
mov dl,79
mov bh,7
int 10h
```

Figure 4: 清除屏幕内容

```
Message:
db "Hello, MyOS is Ready!",0AH,0DH
db "Here is the start Menu.Please Enter the serial number selection program: ",0AH,0DH
db "1.Personal information",0AH,0DH
db "2.stone",0AH,0DH
db "3.square",0AH,0DH
db "4.sand clock",0AH,0DH
```

Figure 5: 显示提示信息

利用int 16h的BIOS功能调用，读取键盘输入，再按照输入的值跳转到需要运行的程序段，使用BIOS功能13H来加载不同的扇区号内的程序段到内存中使用，如figure6 figure7。

```
;使用int16指令输入选择跳转
;使用循环来确保输入实在1, 2, 3, 4中的一个数字.
input:
mov ah,0
int 16h
cmp al,'1'
jz switch
cmp al,'2'
jz switch
cmp al,'3'
jz switch
cmp al,'4'
jz switch
jmp input
```

Figure 6: 获取键盘输入

```
switch:
mov bl,'1'
cmp al,bl
jz program1
mov bl,'2'
cmp al,bl
jz program2
mov bl,'3'
cmp al,bl
jz program3
mov bl,'4'
cmp al,bl
jz program4
```

Figure 7: 通过键盘输入跳转

3) 设计用户程序:

i) 用户程序1:

在用户程序前都要有语句`org 0A100h`,它是让计算机将程序加载到0A100开始的内存上运行,每个用户程序都加载到这个地址上,监控程序就可以统一跳转到这个地址上来运行所选扇区的内容。

利用bios提供的中断`int 10h`的13h号功能,实现在显示器上输出字符串。`bx`可以设置显示姓名序号的前景色背景色等属性。在展示学号姓名等学生信息的同时,不断改变字符的属性和显示字符串的位置,让信息在显示屏上不断的上下移动变化位置和属性,实现动态展示的效果。

因为实验要求在屏幕的四个位置分别展示四个用户程序,所以我需要先画大概的**衡量边界**,再编写程序

在程序末端,利用`int 16h`的1号功能,实现捕捉键盘输入(不阻塞等待)来检查用户是否输入的**空格**,再使用`jmp`语句返回**监控程序的内存地址7c00处**,其他几个用户程序都有类似的代码段,就不在下面赘述。代码和实验效果如figure 8和figure 9:

```
showMessage:
    inc dh
    mov al,11
    cmp dh,al
    jnz next
    mov dx,Name
    mov bp,ax ; es:bp中地址
    mov cx,20
    mov dx,1301h ; ah=13 显示字符串
    mov dx,000ah ; bl 6-4前景色 3位1前景色高亮 2-e 前景色bgc
    mov al,0ch
    int 10h

    inc dh
    mov dx,Number
    mov bp,ax
    mov cx,15
    mov dx,1301h
    mov dx,000ah
    mov al,0ch
    int 10h
    sub dh,1
    jmp next
```

Figure 8: 展示姓名学号

```
; 利用int 16h的 01号功能,输入空格后弹出程序
mov ah,1
int 16h
mov bl,20h
cmp al,bl
jz Quit
jmp loop1

Quit:
    jmp 7c00h

Name: db "Author:Ye Sheng Yuan"
Number: db "Number:17341190"
count dw delay
dcount dw ddelay
```

Figure 9: 按空格返回

ii) 用户程序2

沿用第一个实验中的stone的程序,当小球在屏幕上弹动的时候,不断改变小球显示字符的属性,实现动态展示的效果,代码类似实验一中的stone,部分截图示例如下图figure 10和figure 11:

```
loop1:
    dec word[count] ; 递减计数器 产生延迟 一共50000*500
    jnz loop1 ; 不为0,继续;
    mov word[count],delay ; 递减计数器
    dec word[count] ; 递减计数器
    jnz loop1 ; 不为0,继续;
    mov word[count],delay ; 递减计数器
    mov word[count],ddelay ; 递减计数器
    ; 类似switch case语句
    mov al,1
    cmp al,byte[rdu1]
    jz Dnlt
    mov al,2
    cmp al,byte[rdu1]
    jz Updt
    mov al,3
    cmp al,byte[rdu1]
    jz Updt
    mov al,4
    cmp al,byte[rdu1]
    jz Dnlt
    jmp $ ; rdu1读空,进入死循环
```

Figure 10: 跳转到当前方向

```
show1:
    mov ax,word[x]
    mov bx,00
    mul bx ; 把 al*bx结果 放到 ax
    add ax,word[y]
    mov bx,2
    mul bx
    mov bp,ax
    mov ah,[property] ; 设置属性
    mov al,byte[char] ; AL = 显示字符 (默认值为20h=空格符)
    mov word[px+bp],ax ; 显示字符和ASCII码值
    add byte[property],15

    ; 利用int 16h的 01号功能,输入空格后弹出程序
    mov ah,1
    int 16h
    mov bl,20h
    cmp al,bl
    jz Quit
    jmp loop1
```

Figure 11: 展示小球属性和字符

iii) 用户程序3:

第三个用户程序是在屏幕的中间显示一个程序员们刚入门一定会看到的，字符串”hello world”，在四周沿着边缘用字符0画出一个五彩的边框作为装饰，边框不断改变着它的属性和颜色，呈现出一个动态的展示效果，代码类似实验1中的弹球实验，不过要做一些修改，因为字符在矩形边框上移动只有上下左右四种运动方式。实验代码举例如下图figure 12和figure 13

```
showMessage:
    mov ax,Message
    mov bp,ax ; es:bp串地址
    mov cx,11
    mov ax,1301h ; ah=13 显示字符串
    mov bx,101ah ;b1 6-4背景色 3位1前景色高亮 2-0 前景色RGB
    mov dx,0537h
    int 10h
```

Figure 12: 展示hello world字符串

```
Dn:
    inc word[x2]
    mov bx,word[x2]
    mov ax,12
    sub ax,4
    jr DnLeft
    add byte[property],15
    jmp show
DnLeft:
    mov word[x2],11
    mov byte[rdi],R_t
    jmp show
Rt:
    inc word[y2]
    mov bx,word[y2]
    mov ax,80
    sub ax,40
    jr RtLeft
    add byte[property],15
    jmp show
RtLeft:
    mov word[y2],79
    mov byte[rdi],U_p
    jmp show
```

Figure 13: 向右和向下示例

iv) 用户程序4:

第四个程序是制作一个沙漏一样的图形，字符动态移动并在碰到预设好的边缘后发生转向，在画沙漏图形的时候，只有下，左下和左上三种运动的情况，示例代码如下图：

```

Rt:
    inc word[y]
    mov bx,word[y]
    mov ax,53
    sub ax,bx
    jz Rttoit
    jmp show1

Rttoit:
    mov bx,word[x]
    mov ax,12
    sub ax,bx
    jz Rttoit_On

Rttoit_Up:
    mov word[y],52
    mov byte[rdul],it_Up_
    jmp show1

Rttoit_On:
    mov word[y],52
    mov byte[rdul],it_On_
    jmp show1
  
```

Figure 14: 向下移动情况

```

it_On:
    inc word[x]
    dec word[y]
    mov bx,word[x]
    mov ax,25
    sub ax,bx
    jz it_Onoatt
    jmp show1

it_Onoatt:
    mov word[x],24
    mov word[y],48
    mov byte[rdul],it_
    jmp show1

it_Up:
    dec word[x]
    dec word[y]
    mov bx,word[x]
    mov ax,11
    sub ax,bx
    jz it_Onoatt
    jmp show1

it_Onoatt:
    mov word[x],12
    mov word[y],48
    mov byte[rdul],it_
  
```

Figure 15: 向左上和坐下移动情况

v) 使用NASM编译程序:

打开nasm，并输入指令nasm+文件名.asm，编译完后在同一个目录下会生成一个同名的二进制文件。重复操作将4个用户程序和监控程序都编译一次并生成各自的二进制文件。举例如图：

```

D:\NASM>nasm test.asm
D:\NASM>
  
```

Figure 16: nasm编译



 test
 test.asm

Figure 17: 生成bin文件

vi) 使用winHex修改软盘:

使用WinHex打开刚刚生成的所有二进制文件和软盘flp文件，首先将监控程序myos的二进制文件内容替换到WinHex的第一个扇区(512Byte)中。**注意：程序的大小不能超过512B，否则会出错，替换后软盘的大小也不能改变。**然后在第二个512字节的扇区中放第一个监控程序，在第三个512字节的扇区内放第二个，依次类推。**用户程序之间不能重叠在同一个扇区。**如图figure18和figure19:

logoffp	fun1	test	mys1	fun2	fun3	fun4	段头1bp	
offset	0	1	2	3	4	5	6	7
00000000	8C	08	8E	D8	BD	7C	8C	D8
00000001	B4	64	80	00	85	00	B1	00
00000002	89	98	00	88	01	13	BB	07
00000003	CD	14	6C	31	74	06	3C	32
00000004	74	02	8B	EA	B3	31	38	D8
00000005	83	33	38	D8	74	36	B3	34
00000006	BB	00	AI	B4	02	80	01	B2
00000007	13	89	8C	24	8C	08	00	88
00000008	00	86	00	85	00	B1	03	CD
00000009	BB	00	AI	B4	02	80	01	B2
00000010	13	89	8C	24	8C	08	00	88
00000011	00	86	00	85	00	B1	03	CD
00000012	13	89	8C	24	8C	08	00	88
00000013	00	86	00	85	00	B1	03	CD
00000014	BB	00	AI	B4	02	80	01	B2
00000015	13	89	8C	24	8C	08	00	88
00000016	00	86	00	85	00	B1	03	CD
00000017	00	86	00	85	00	B1	03	CD
00000018	00	86	00	85	00	B1	03	CD
00000019	00	86	00	85	00	B1	03	CD
00000020	00	86	00	85	00	B1	03	CD
00000021	00	86	00	85	00	B1	03	CD
00000022	00	86	00	85	00	B1	03	CD
00000023	00	86	00	85	00	B1	03	CD
00000024	00	86	00	85	00	B1	03	CD
00000025	00	86	00	85	00	B1	03	CD
00000026	00	86	00	85	00	B1	03	CD
00000027	00	86	00	85	00	B1	03	CD
00000028	00	86	00	85	00	B1	03	CD
00000029	00	86	00	85	00	B1	03	CD
00000030	00	86	00	85	00	B1	03	CD
00000031	00	86	00	85	00	B1	03	CD
00000032	00	86	00	85	00	B1	03	CD
00000033	00	86	00	85	00	B1	03	CD
00000034	00	86	00	85	00	B1	03	CD
00000035	00	86	00	85	00	B1	03	CD
00000036	00	86	00	85	00	B1	03	CD
00000037	00	86	00	85	00	B1	03	CD
00000038	00	86	00	85	00	B1	03	CD
00000039	00	86	00	85	00	B1	03	CD
00000040	00	86	00	85	00	B1	03	CD
00000041	00	86	00	85	00	B1	03	CD
00000042	00	86	00	85	00	B1	03	CD
00000043	00	86	00	85	00	B1	03	CD
00000044	00	86	00	85	00	B1	03	CD
00000045	00	86	00	85	00	B1	03	CD
00000046	00	86	00	85	00	B1	03	CD
00000047	00	86	00	85	00	B1	03	CD
00000048	00	86	00	85	00	B1	03	CD
00000049	00	86	00	85	00	B1	03	CD
00000050	00	86	00	85	00	B1	03	CD
00000051	00	86	00	85	00	B1	03	CD
00000052	00	86	00	85	00	B1	03	CD
00000053	00	86	00	85	00	B1	03	CD
00000054	00	86	00	85	00	B1	03	CD

Figure 18: 监控程序的二进制文件

logoffp	fun1	test	mys1	fun2	fun3	fun4	段头1bp	
offset	0	1	2	3	4	5	6	7
00000000	8C	08	8E	D8	BD	7C	8C	D8
00000001	B4	64	80	00	85	00	B1	00
00000002	89	98	00	88	01	13	BB	07
00000003	CD	14	6C	31	74	06	3C	32
00000004	74	02	8B	EA	B3	31	38	D8
00000005	83	33	38	D8	74	36	B3	34
00000006	BB	00	AI	B4	02	80	01	B2
00000007	13	89	8C	24	8C	08	00	88
00000008	00	86	00	85	00	B1	03	CD
00000009	BB	00	AI	B4	02	80	01	B2
00000010	13	89	8C	24	8C	08	00	88
00000011	00	86	00	85	00	B1	03	CD
00000012	13	89	8C	24	8C	08	00	88
00000013	00	86	00	85	00	B1	03	CD
00000014	BB	00	AI	B4	02	80	01	B2
00000015	13	89	8C	24	8C	08	00	88
00000016	00	86	00	85	00	B1	03	CD
00000017	00	86	00	85	00	B1	03	CD
00000018	00	86	00	85	00	B1	03	CD
00000019	00	86	00	85	00	B1	03	CD
00000020	00	86	00	85	00	B1	03	CD
00000021	00	86	00	85	00	B1	03	CD
00000022	00	86	00	85	00	B1	03	CD
00000023	00	86	00	85	00	B1	03	CD
00000024	00	86	00	85	00	B1	03	CD
00000025	00	86	00	85	00	B1	03	CD
00000026	00	86	00	85	00	B1	03	CD
00000027	00	86	00	85	00	B1	03	CD
00000028	00	86	00	85	00	B1	03	CD
00000029	00	86	00	85	00	B1	03	CD
00000030	00	86	00	85	00	B1	03	CD
00000031	00	86	00	85	00	B1	03	CD
00000032	00	86	00	85	00	B1	03	CD
00000033	00	86	00	85	00	B1	03	CD
00000034	00	86	00	85	00	B1	03	CD
00000035	00	86	00	85	00	B1	03	CD
00000036	00	86	00	85	00	B1	03	CD
00000037	00	86	00	85	00	B1	03	CD
00000038	00	86	00	85	00	B1	03	CD
00000039	00	86	00	85	00	B1	03	CD
00000040	00	86	00	85	00	B1	03	CD
00000041	00	86	00	85	00	B1	03	CD
00000042	00	86	00	85	00	B1	03	CD
00000043	00	86	00	85	00	B1	03	CD
00000044	00	86	00	85	00	B1	03	CD
00000045	00	86	00	85	00	B1	03	CD
00000046	00	86	00	85	00	B1	03	CD
00000047	00	86	00	85	00	B1	03	CD
00000048	00	86	00	85	00	B1	03	CD
00000049	00	86	00	85	00	B1	03	CD
00000050	00	86	00	85	00	B1	03	CD
00000051	00	86	00	85	00	B1	03	CD
00000052	00	86	00	85	00	B1	03	CD
00000053	00	86	00	85	00	B1	03	CD
00000054	00	86	00	85	00	B1	03	CD

Figure 19: 将监控程序拷贝到软盘首扇区

vii) 用软盘启动裸机:

将软盘的前五个扇区分别替换为自己编写的汇编指令的二进制代码后, 就可以开启虚拟机, 选择以虚拟软盘启动, 启动后虚拟机就会立刻将我们的程序加载到虚拟机的内存空间中并运行, 监控程序和各个用户程序效果如下图:

```

Hello, MpOS is Ready!
Here is the start Menu.Please Enter the serial number selection program:
1.Personal information
2.stone
3.square
4.sand clock
-

```

Figure 20: 监控程序

```

Author:Ye Sheng Yuan
Number:17341198

```

Figure 21: 用户程序1



Figure 22: 用户程序2

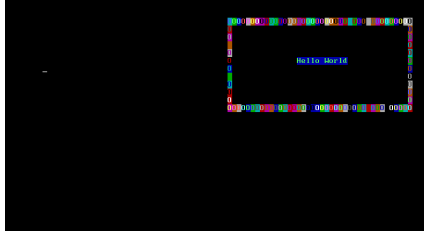


Figure 23: 用户程序3

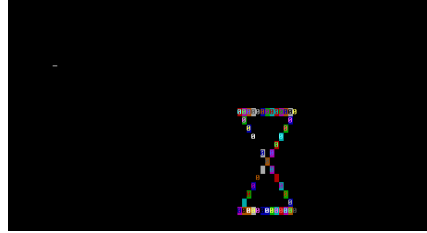


Figure 24: 用户程序4

5 实验心得和总结：

这次实验是建立在上一次实验的基础上的进阶。上一次我们在一个没有操作系统的裸机上使用软盘的首扇区引导裸机启动，而这次我们需要自己写一个监控程序，模拟最原始的操作系统的功能，可以控制执行哪个扇区里的用户程序，用户程序在执行完成或者接收到讯号后立刻将控制权交还给监控程序，这其实就是用`jmp`指令在内存中的地址间跳转的结果，这也让我对“操作系统其实就是一个程序”这句话渐渐有了更深刻的理解和体会，同时在实验中，我使用了很多的BIOS提供的中断指令，它们不同的功能帮助我更轻松了使用汇编语言编写出了想要的效果，包括`int 10h`、`int 13h`等指令，在使用过程中慢慢的熟练也不需要经常翻阅资料就懂得如何去使用和改换参数了。

这次实验因为有了上一次实验的基础，所以在设计的过程中，大致有了方向和方法，不像上次那样一直在查询资料，但这次还是和上次实验有很多不同的地方，例如监控程序的设计，如何使用键盘输入在内存中跳转，为什么要使用`org a100h`指令等等，这些都让我十分的困惑。其中让我思考最久的，是在我设计监控程序跳转到用户程序的时候，键盘输入经常无法被捕捉，之后慢慢调试才发现，可能是因为键盘缓冲区没有被清空，为了解决这个问题，我选择了使用一个循环嵌套在输入外面，要求用户一定要输入是1、2、3、4其中一个值才能离开循环，这样也不会出现键盘缓冲区有回车或者空格导致跳过输入这一部分了。

在用户程序返回监控程序的时候也遇到了问题，因为用户程序我不能用一个循环要求用户输入，因为程序必须一直在执行，所以我查看了BIOS `int 16h`的1号功能调用，发现了可以在不阻塞的条件下反复查看键盘缓冲区内容，一旦发现空格就立刻返回监控程序，但如果键盘缓冲区为空，也不会阻塞用户程序的进行，这样就成功实现了我的目的。

在使用nasm和winhex将用户的二进制代码覆盖软盘扇区的时候也经常遇到困难，因为需要很细心的删除和复制，否则很容易就会删多或者删少导致软盘大小改变被损坏，所以操作系统的实验其实是一个考验人的耐心的过程，包括汇编语言、覆盖扇区、找BUG等都是十分的困难的，也让我深刻体会了以前的程序员是多么的痛苦。总之，实验二顺利结束了，希望我能继续努力，面对下一个挑战，继续加油！

6 参考文献:

- 1) 刘海洋. L^AT_EX入门[M]. 北京: 电子工业出版社, 2013.
- 2) <https://blog.csdn.net/cielozhang/article/details/6171783/>
- 3) https://blog.csdn.net/lulipeng_cpp/article/details/8161982
- 4) <https://www.cnblogs.com/alwaysking/p/7789282.html>
- 5) <https://blog.csdn.net/cielozhang/article/details/6171783>
- 6) https://blog.csdn.net/lulipeng_cpp/article/details/8161982