

操作系统原理实验二报告

姓名:任文頔

学号:14322181

院系:数据科学与计算机学院

专业年级:计算机科学与技术 14级

指导老师:凌应标

【实验题目】

加载用户程序的监控程序

【实验目的】

- 1 开发最原始的操作系统——监控程序
- 2 熟悉BIOS的调用
- 3 了解用户程序的格式和执行过程
- 4 进一步熟悉NASM汇编语言

【实验要求】

- 1设计四个（或更多）有输出的用户可执行程序

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

【实验方案】

1 硬件或虚拟机配置方法

系统环境：windows 7

虚拟机配置方法：选择稍后安装操作系统，客户操作系统选取其他，版本选择其他。为虚拟机命名为14322181 renwendi。添加软盘驱动器，选择软盘映像文件为已经创建好的myos1.img。其余使用默认设置，即最大磁盘大小为8G，将虚拟磁盘拆分为多个文件。

2 软件工具与作用

- (1) 汇编语言编译器NASM：针对Intel x86架构的汇编与反汇编程序
- (2) 二进制编辑器WinHex：编辑文件和各种磁盘、U盘的物理扇区，查看机器码。
- (3) 文本编辑器Notepad++：支持ASP、Assembly、C、C++、C#、CSS、COBOL、Fortran、HTML、Java、Javascript、LISP、Makefile、Matlab、Objective-C、Pascal、Perl、PHP、Python、Ruby、SQL、VB、XML等52种编程和描述语言的语法格式。
- (4) 虚拟机软件Bochs：自己编写的操作系统的测试环境，也可用于生成软盘映像文件。

(5) 写软盘映像小软件DiskWriter：用于将NASM汇编程序产生的二进制文件*.bin写入磁盘映像文件中。

3 方案的思想

编写操作系统监控程序 and 用户可执行程序，而后将程序依次放在软盘的不同扇区。目前我们使用IBM_PC基本内存640k，实模式的段+段内偏移的方式访问内在单元。BIOS装入引导扇区程序时，优先使用最低端的一个64k段作为引导扇区程序的存放区域，偏移量7c00h。之后跳转到操作系统监控程序，控制用户程序，本实验中我将操作系统监控程序放在偏移量8100h开始的一个区域。再将用户程序存放在同一个段中。

实现批处理，将用户输入的数字每次存储并显示，判断输入的数字是几，跳转到相应的子程序。

4 相关知识原理

(1) 监控程序

监控程序是操作系统的最早期的形式，其控制方式为：获取计算机硬件系统的控制权，提供计算机输入设备和输出的控制程序，控制用户程序的执行。

(2) 用户程序的格式：

监控程序获取计算机硬件系统的控制权的方法：如果程序不超过512字节，则可以引导扇区程序的方法实现；如果程序超过512字节，则要分解为引导扇区部分和其它功能部分。

用户程序的格式：COM，BIN

监控程序控制用户程序的执行的方法过程：加载用户程序，磁盘存储方法，磁盘读取控制编程。

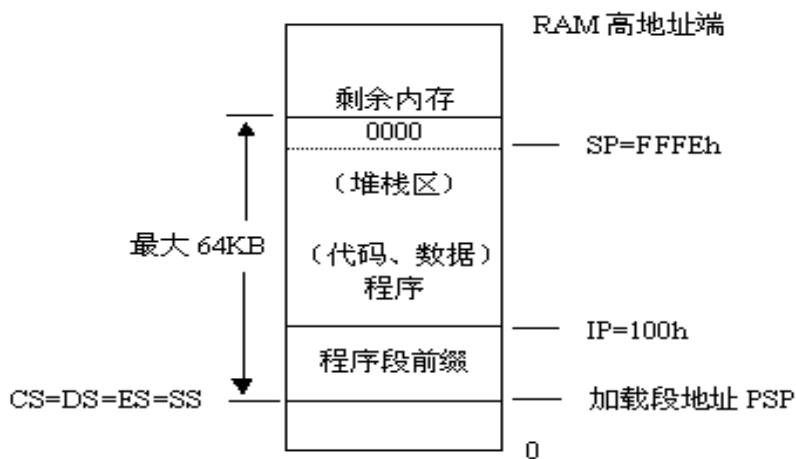
(3) COM格式：

COM (command file, 命令文件) 是CP/M和DOS的一种原始二进制可执行格式，以.com为扩展名。COM文件非常简单，没有文件头、没有元数据，只有代码和数据。

COM文件会被装载到当前段的0x100 (256) 处，不能重新定位。由于不能分段，所以COM文件的大小必须 $\leq 64KB-256B$ ，且不能有独立的数据段和堆栈段，程序的所有代码和数据都必须位于一个段中。

另外，在Windows操作系统的64位版本中，不再支持COM程序的运行。

DOS加载COM程序的内存映象图：



(4) 返回DOS

利用4ch功能调用返回DOS (.COM/.EXE)

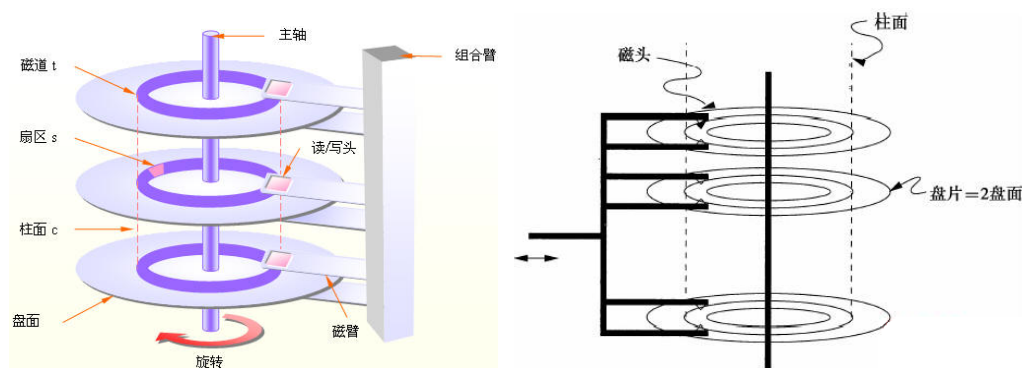
DOS功能调用的4ch子功能（返回DOS）：入口参数：AH = 4ch，AL = 返回数码。

产生终止程序执行返回操作系统的指令代码,它的可选参数是一个返回的数码，通常用0表示没有错误。例如程序结束点放置对应的代码是：

```
mov ax,4c00h
int 21h
```

(5) 磁盘分区

磁盘分区 (disk partitioning) 将硬盘分割成若干逻辑部分 (分区, partition)，可安装多个操作系统和使用不同的文件系统。在DOS和Windows中，硬盘的分区对应于文件系统中不同的卷 (volume)，如C:、D:、E: 等。由于软盘的容量小，没有分区结构。



磁盘结构

主引导扇区的构成：

主引导扇区 (Master Boot Sector) 是磁盘的第一个物理扇区 (0号扇区=0磁头/盘面、0柱面/磁道、1扇区)，由主引导记录 (MBR = Master Boot Record)、磁盘分区表 (DPT = Disk Partition Table) 和引导记录标识符 (BRID = Boot Record Identifier) 三部分组成。

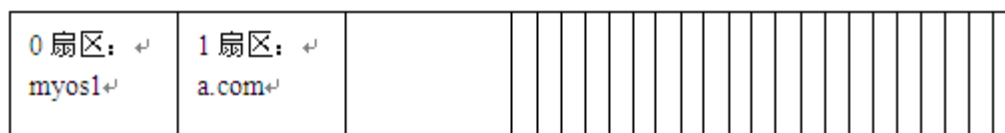
0	主引导记录	MBR(446B)	
1BD			
1BE			
1CD			
1CE	分区项1(16B)		
1DD			
1DE	分区项2(16B)		
1ED			
1EE	分区项3(16B)		
1FD			
1FE	分区项4(16B)		
1FF			
	55 AA	BRID(2B)	

主引导扇区结构图

6. 内存安排

一种简单的磁盘存储组织：

1.44MB的软盘共2880个扇区，布局如图2-2.0扇区：



1.44MB的软盘布局图

目前我们使用IBM_PC基本内存640k，实模式的段+段内偏移的方式访问内在单元。

BIOS装入引导扇区程序时，优先使用最低端的一个64k段作为引导扇区程序的存放区域，偏移量7c00h。

7. BIOS中中断例程

BIOS中断服务程序是微机系统软、硬件之间的一个可编程接口，用于程序软件功能与微机硬件实现的衔接。DOS/Windows操作系统对软、硬盘、光驱与键盘、显示器等外围设备的管理即建立在系统BIOS的基础上。程序员也可以通过INT 5、INT 13等终端的访问直接调用BIOS终端例程。

调用BIOS中断服务程序的方法：每个中断服务有特定的参数，一般使用指定的寄存器传递参数。

利用软中断指令调用BIOS中断调用的一般格式为：

mov ah,功能号

……;设置各种入口参数

int 中断号

BIOS的10H号调用功能与参数:

显示字符串	10H	13H	AL: 放置光标的方式 BL: 字符属性字节 BH: 显示页(0~3) DH: 行位置(0~24) DL: 列位置(0~79) CX: 字符串的字节数 ES:BP: 字符串的起始地址	AL=0/2 光标留在串头 AL=1/3 光标放到串尾 AL=0/1 串中只有字符 AL=2/3 串中字符和属性字节交替存储 BL: 位 7 为 1 闪烁 位 6~4 为背景色 RGB 位 3 为 1 前景色高亮 位 2~0 为前景色 RGB
-------	-----	-----	---	--

BIOS的13H号调用功能与参数:

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
-----	-----	-----	---	--

5 程序流程

1. 创建软盘映像文件
2. 编写引导扇区代码
3. 编写主操作系统代码, 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中的任何一个序列, 如123、1432等。
4. 编写四个用户可执行程序代码
5. 组织映象盘空间的存放, 用winHex将引导扇区, 操作系统和用户可执行程序的十六进制机器码分别复制到软盘的第1, 2, 3, 4, 5, 6个扇区。每个扇区的大小为512字节。

——改进方案, 直接使用incbin伪操作, 将二进制文件插入到原文件中, 可以省去手动复制机器码的过程。

6 算法和数据结构

汇编语言的基本算法: 判断, 循环

7 程序关键模块, 结合代码与程序中的位置进行解释。

(1) 引导扇区(boot.asm):

```
org 7c00h
OffsetOfUserPrg equ 8100h
```

Start:

```
    mov ax, cs
    mov ds, ax
    mov ax, ds
    mov es, ax
```

```
    mov ax, 1301h
    mov bx, 0047h
```

```
mov dh, 8
    mov dl, 28
    mov cx, str1_len
    mov bp, str1
    int 10h
```

```
mov ax, 1301h
mov bx, 0007h
mov dh, 12
mov dl, 20
mov cx, str2_len
mov bp, str2
int 10h
```

```
mov ah, 0 ;用户监听，输入任何键后跳转进入操作系统主界面
int 16h
```

LoadnEx:

```
    mov ax, cs
    mov es, ax
    mov bx, OffSetOfUserPrg
    mov ah, 2
    mov al, 5
    mov dl, 0
    mov dh, 0
    mov ch, 0
    mov cl, 2
    int 13H
    jmp OffSetOfUserPrg
```

AfterRun:

jmp \$

;=====字符串内容===== (省略)

times 510-(\$-\$) db 0

db 0x55,0xaa

incbin "os.bin"

incbin "use1.bin"

incbin "use2.bin"

incbin "use3.bin"

incbin "use4.bin"

主扇区引导代码，前半部分基本上与老师所给参考代码一致，增加了监听用户输入，按任意键跳转到操作系统主界面。incbin操作写入引导程序的最后，将依次复制二进制文件到2，3，4，5，6扇区。

(2) 监控程序(os.asm): 主要说明与第一版相比的改进部分，即批处理的实现

①: Listen_in:

mov ah,0 ;提示用户输入后，监听,输入存于al中

int 16h

cmp al,0dh ;判断输入的是否是回车键,ASC码的十六进制是0d

jne shownum;jne,如果不是回车键，跳转到shownum函数显示用户输入的数字并储存

jmp Run_pro;如果是回车键，跳转到Run_pro函数执行输入序列的程序

每次用户输入一个值后，系统就进行判断，因为al中只能存储一个数字。

②: Shownum:

mov ah,0eh

mov bl,00h

inc word [count] ;统计输入的数字序列的数字个数，每次跳转到shownum函数,count加1

cmp word [count],1

je savea

jmp compB

savea:

mov [a],al ;用户输入的数字存储在al中，将输入的数字存在a中

int 10h

je Listen_in

```

cmp word [count],2
je saveb
jmp compC
saveb:
mov [b],al
int 10h
je Listen_in

```

compE:

cmp word [count],5 ;超过四个字符不显示也不保存，继续监听直到回车

jge Listen_in ; 如果大于或等于5则跳转

中间省略comC, comD部分，与A、B类似。

在每次输入不是回车键后，即跳转进入Shounum函数。count的初始值为0，每跳转一次，增加1。用户输入第一个数字后，count变为1，count与1进行比较，现在是相等，那么跳转到savea函数，将用户输入的的第一个数字保存在a中，再次跳回Listen_in函数，等待用户输入第二个数字。若用户没有输入回车键，那么又跳回到Shounum函数，此时count又加1，变为2。先与1进行比较，不相等，跳入comB函数，count与2进行比较，现在是相等，那么跳转到saveb函数，将用户输入的第二个数字保存在b中，再次跳回Listen_in函数。如此循环反复，最多count与5进行比较，超过四个字符不显示也不保存，继续监听直到回车。最后达到的效果就是在a、b、c、d中分别存储了第一、二、三、四次用户输入的对子程序的数字。

③: Run_pro:

```

cmp word [a],0 ;第一个程序
jne proa
jmp Run_end;没有则结束
proa:
mov ax,[a]
sub ax,46    ;扇区偏移，前两个扇区中分别放着引导和操作系统,[a]-48+2=[a]-46
mov [position],ax ;用户程序的加载扇区位置
call Run

cmp word [b],0 ;第二个程序
jne prob
jmp Run_end
prob:
mov ax,[b]
sub ax,46

```



```
mov [position],ax
call Run
```

第三、第四个主程序与前两个类似，只是改变了变量值a、b、c、d。

当输入为回车键后，跳转到Run_pro函数，依次将a、b、c、d与0比较。a、b、c、d的初始值均为0，若不是0，则对应第一、二、三、四次的用户输入的程序序号。接下来计算对应子程序的扇区偏移量。共48个扇区，前两个扇区存放引导扇区和操作系统，3、4、5、6扇区存放4个子程序。用mov ax,[a]; sub ax,46 计算扇区偏移量，用mov [position],ax将用户程序的加载扇区位置存储到position中。

④： Run: ;执行某个用户的程序,具体扇区在[position] 中

```
mov ax,0
mov es,ax
mov bx,0a700h ;加载到内存
mov ax,0201h;
mov dx,0
mov ch,0
mov cl,[position];要被加载的扇区号
int 13h;加载

call word 0a700h
ret
```

计算好每次输入对应子程序的扇区偏移后，加载到内存0a700h处执行。

(3) 改变屏幕填充颜色：

与第一个版本相同。

利用循环，不断将‘ ’也就是空格填充进入屏幕，bx设置填充颜色，屏幕共25*80=2000个字符，2000*2=4000，因此设置cx=3999，bx=0，每次填充后，对bx加1，对cx减1，比较两者大小，若bx仍小于cx，则继续循环执行。

(4) 用户程序：

```
call clean_s;清屏
org a700h
mov ax,cs
mov es,ax
mov ds,ax
```

其余部分未做修改，只是修改了加载位置，四个用户程序都加载到内存a700h处执行。按照本操作系统，每次判断后加载到相应子程序的扇区位置，每个子程序所在扇区不一样，且是依次执行。a700h距离操作系统所在的8100h足够远。在os.asm中的Run函数中声明了加载位置: mov bx,0xa700。保持一致。

【实验过程】

1 主要工具安装使用过程及截图结果

1. Vmware



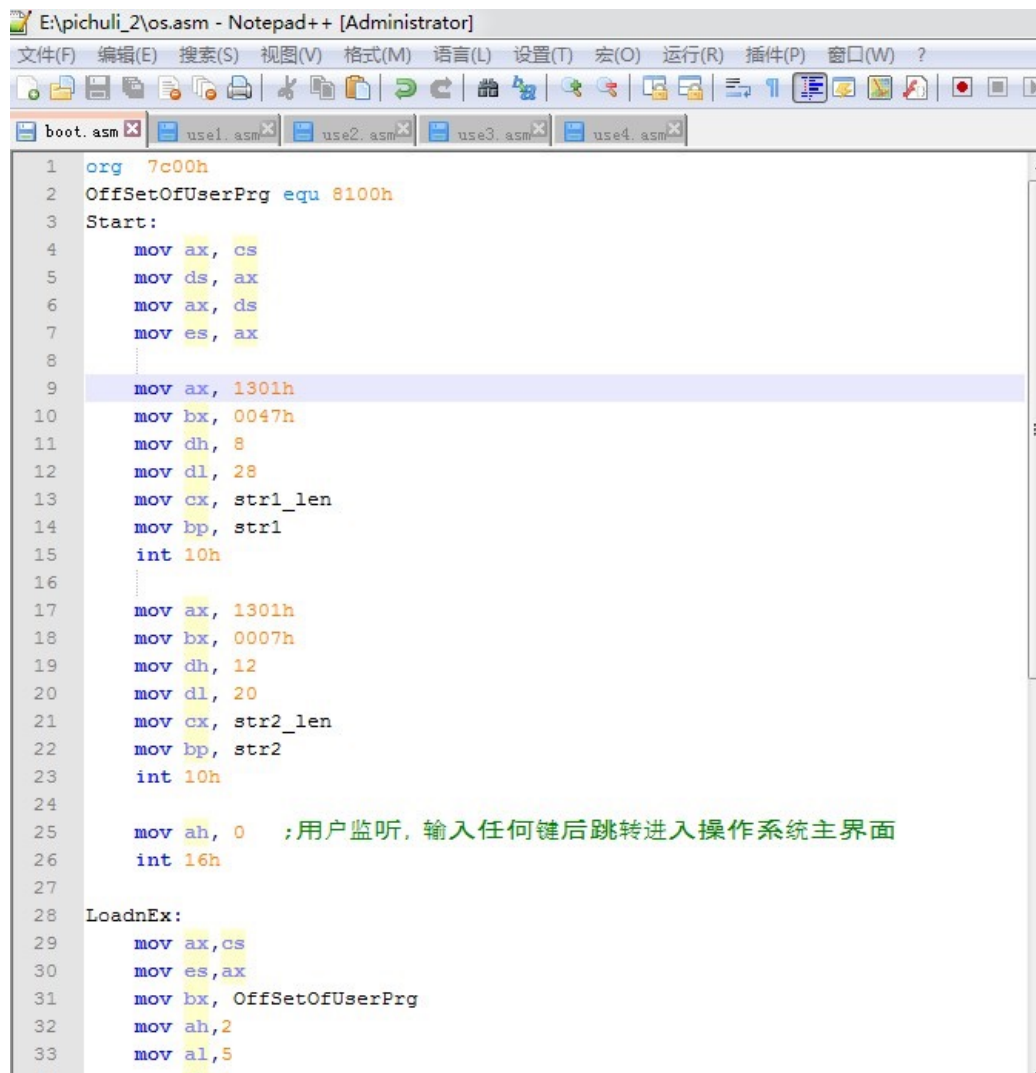
建立2个虚拟机，方便进行调试。

2. 用NASM编译汇编文件，检查语法错误

The screenshot shows a Windows command prompt window titled "管理员: nasm". The window contains a series of commands and their outputs. The commands are:

```
C:\Program Files (x86)\nasm>nasm bootw.asm -o bootw.img
C:\Program Files (x86)\nasm>nasm usr1.asm -o usr1.bin
usr1.asm:82: warning: label alone on a line without a colon might be in error
C:\Program Files (x86)\nasm>nasm usr1.asm -o usr1.bin
C:\Program Files (x86)\nasm>nasm bootw.asm -o bootw.img
C:\Program Files (x86)\nasm>nasm os.asm -o os.bin
C:\Program Files (x86)\nasm>nasm use1.asm -o use1.bin
C:\Program Files (x86)\nasm>nasm use2.asm -o use2.bin
C:\Program Files (x86)\nasm>nasm use3.asm -o use3.bin
C:\Program Files (x86)\nasm>nasm use4.asm -o use4.bin
C:\Program Files (x86)\nasm>nasm boot.asm -o boot.img
C:\Program Files (x86)\nasm>nasm use4.asm -o use4.bin
C:\Program Files (x86)\nasm>nasm use3.asm -o use3.bin
```

3. 用Notepad++编辑代码



```
1 org 7c00h
2 OffsetOfUserPrg equ 8100h
3 Start:
4     mov ax, cs
5     mov ds, ax
6     mov ax, ds
7     mov es, ax
8
9     mov ax, 1301h
10    mov bx, 0047h
11    mov dh, 8
12    mov dl, 28
13    mov cx, str1_len
14    mov bp, str1
15    int 10h
16
17    mov ax, 1301h
18    mov bx, 0007h
19    mov dh, 12
20    mov dl, 20
21    mov cx, str2_len
22    mov bp, str2
23    int 10h
24
25    mov ah, 0 ;用户监听, 输入任何键后跳转进入操作系统主界面
26    int 16h
27
28 LoadEx:
29     mov ax, cs
30     mov es, ax
31     mov bx, OffsetOfUserPrg
32     mov ah, 2
33     mov al, 5
```

2. 程序过程中的操作步骤

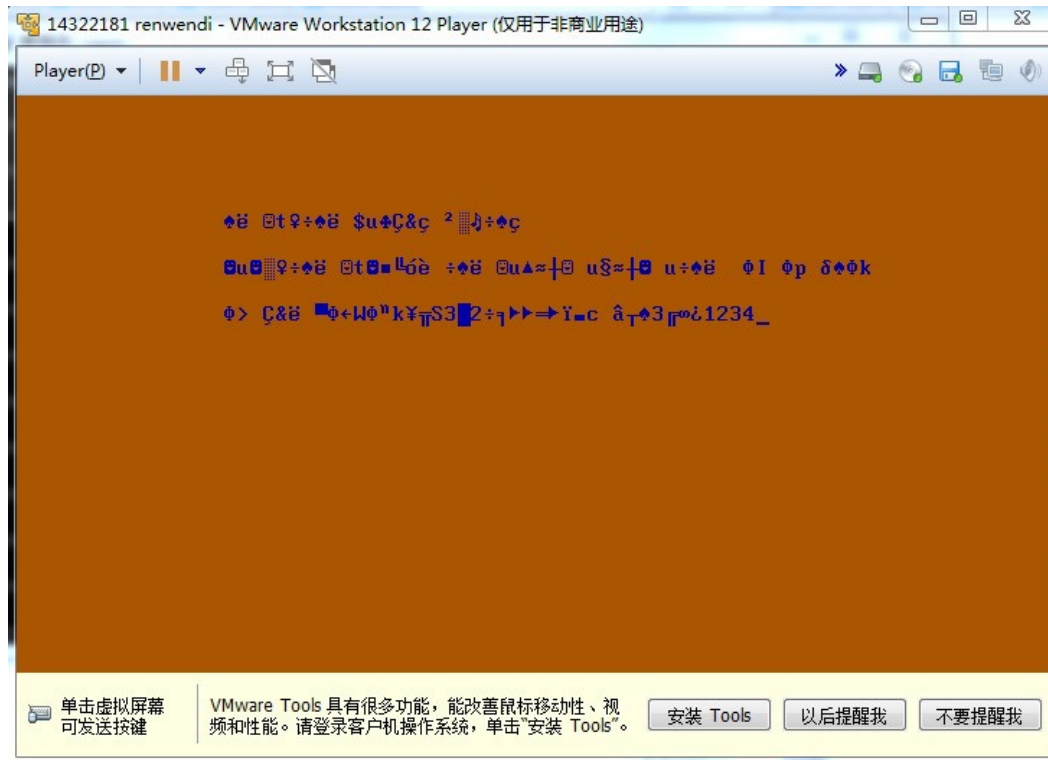
1. 编写引导扇区代码, 并在最后加入incbin "os.bin", incbin "use1.bin", incbin "use2.bin", incbin "use3.bin", incbin "use4.bin". 而后用NASM直接编译生成软盘映像文件。这一步骤的操作可以省去用Bochs创建软盘映像文件和用WinHex复制机器码的过程。程序将会自动将引导扇区, 操作系统和用户可执行程序的十六进制机器码分别复制到软盘的第1, 2, 3, 4, 5, 6个扇区。
2. 编写监控程序代码, 也就是最简单的操作系统原型, 要实现监听用户的输入, 并管理用户进程。注意不能超过512字节, 可用WinHex查看。
3. 编写四个有输出的用户可执行文件, 注意要能返回DOS。
4. 将以上汇编.asm文件用NASM调试, 直到没有错误, 生成.bin二进制文件。
5. 用NASM编译生成软盘映像文件boot.img, 在虚拟机中启动软盘, 不断进行调试。
6. 用WinHex查看软盘映像文件, 确定相应扇区填充正确的机器码, 确保每个程序没有超过一个扇区。

3遇到的问题及解决情况

1. 批处理的实现

这个问题纠结了很久，想到用高级语言里类似数组存储的实现，但是不知道循环判断要怎么实现。后来想到去年计算机组成原理的一个实验，4*4矩阵键盘输入，里面的判断循环和本次操作系统试验有点类似。通过参考后，实现了一层层逐级循环判断。

2. 操作系统选择界面乱码



在第一版的基础上改变了加载的地址，因此出现了问题，发现是少了显示字符串之前，少了初始化，加上如下的语句后正常显示。

```
1 OS_start:
2
3 org 8100h;加载到0x8100执行
4
5 ;=====initation=====
6 call clean_s;清屏, 并且显示颜色
7 call initation;变量初始化, 均为0
8
9 ;=====welcome=====
10 mov ax,cs
11 mov ds,ax
12 mov es,ax
13
14 mov ah,13h
```

3. 操作系统程序 (os.asm) 超过一个扇区

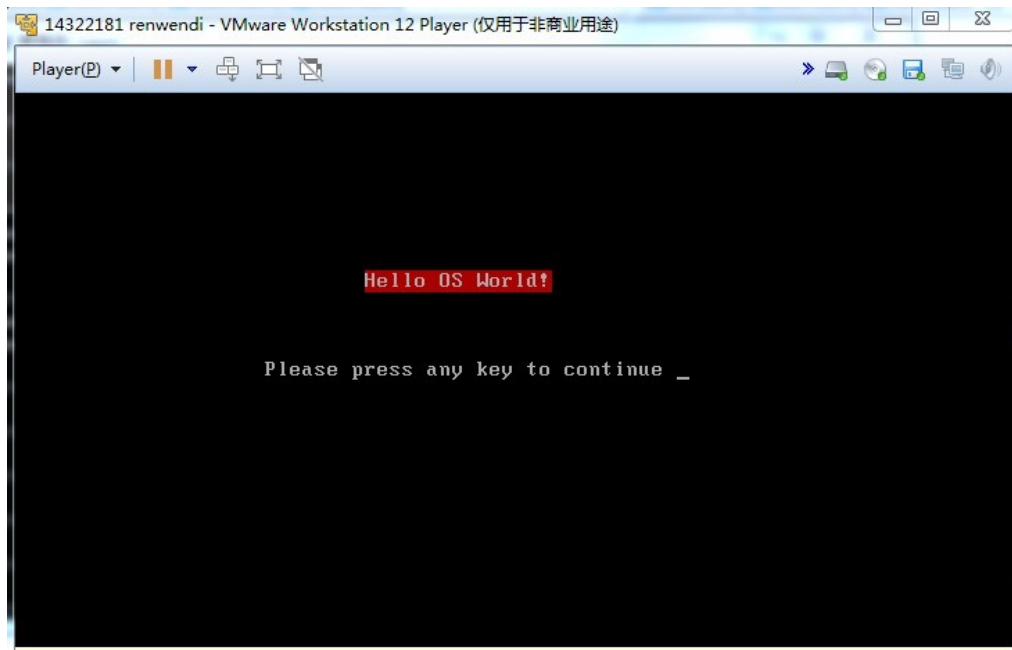
```
E:\pichuli_2>nasm osp.asm -o osp.bin
osp.asm:73: error: symbol 'd' undefined
osp.asm:216: error: TIMES value -5 is negative

E:\pichuli_2>nasm osp.asm -o osp.bin
osp.asm:212: error: TIMES value -7 is negative
```

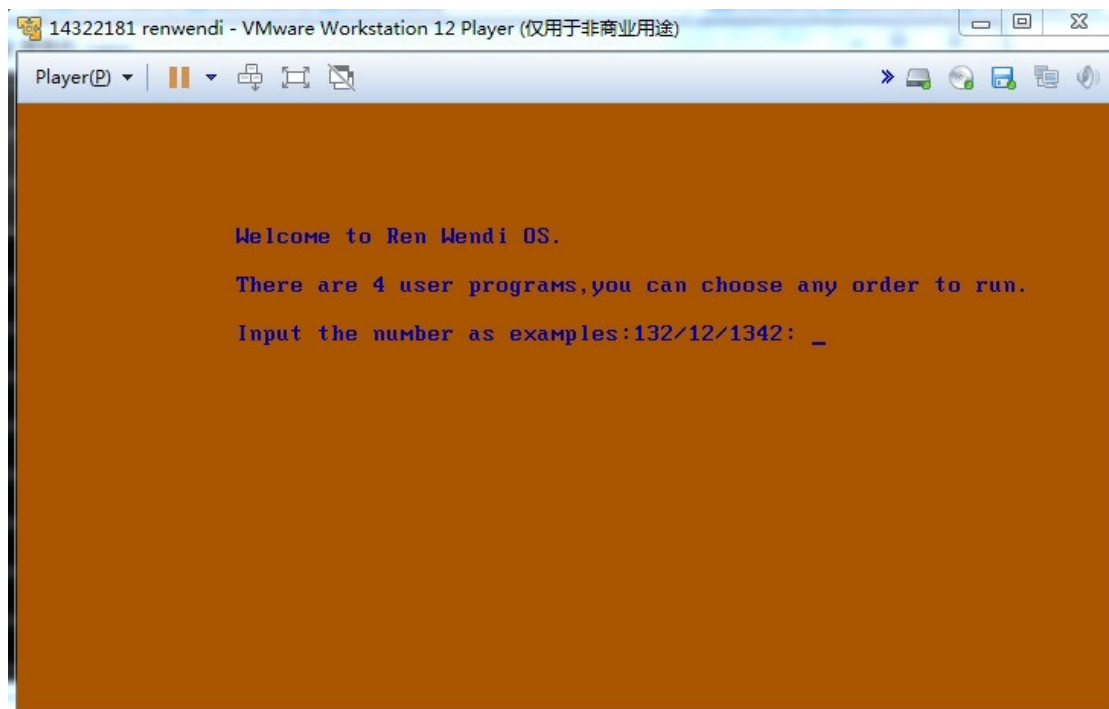
在操作系统程序的代码最后加了times 512-(\$\$\$) db 0 填充0，若前面的代码超过512个字节，则会报错。为了减少代码量，删除了一个字符串显示，因为超出的不太多，少量删除之后即可。

4 关键功能或操作的截图结果

1. 引导程序界面：



2. 欢迎以及选择界面（操作系统主界面）：

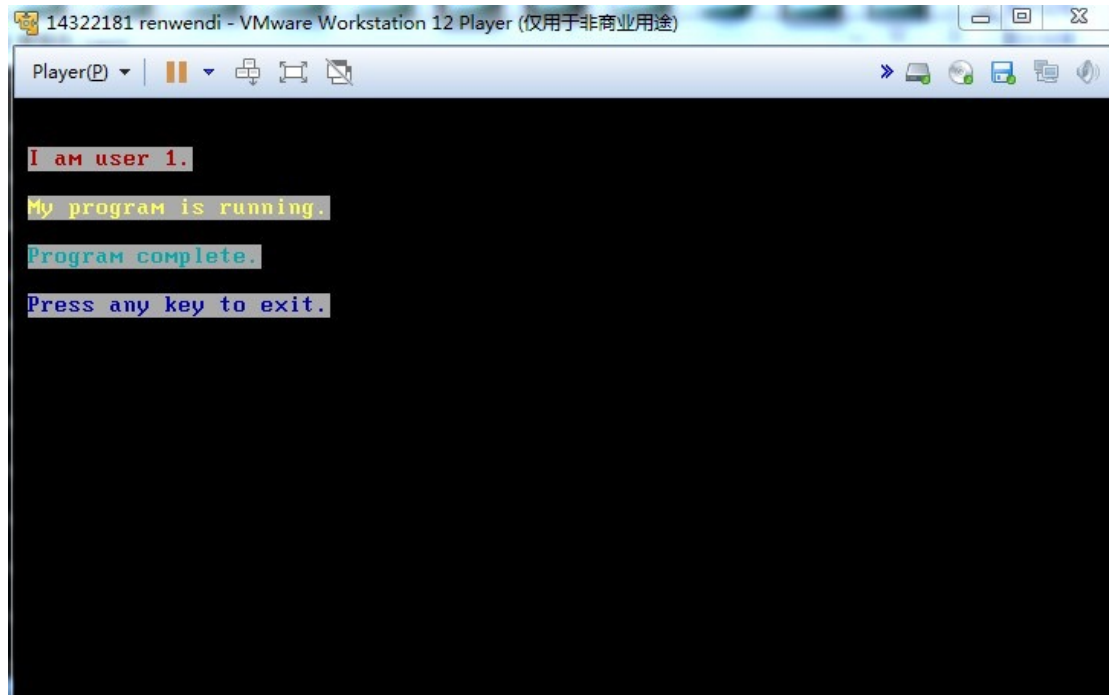


(3) 输入所选择的用户程序后显示序列标号

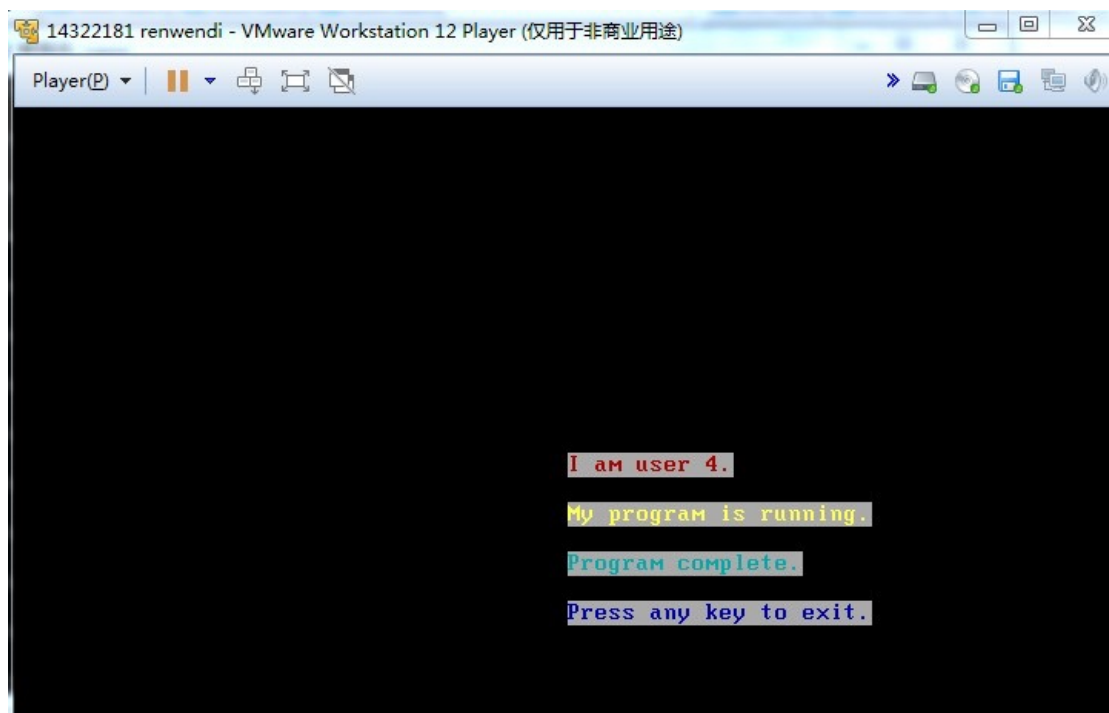


3. 四个用户程序按输入顺序依次执行(如本次输入为1423,按回车键结束)
经过验证，也可输入重复数字，如1322等。

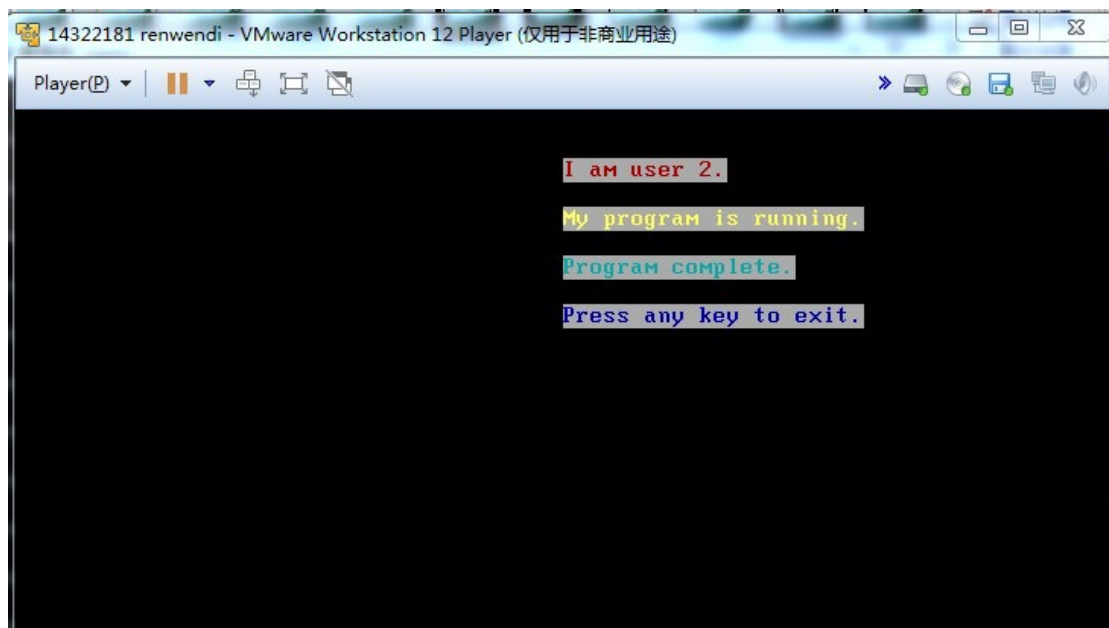
Use1:



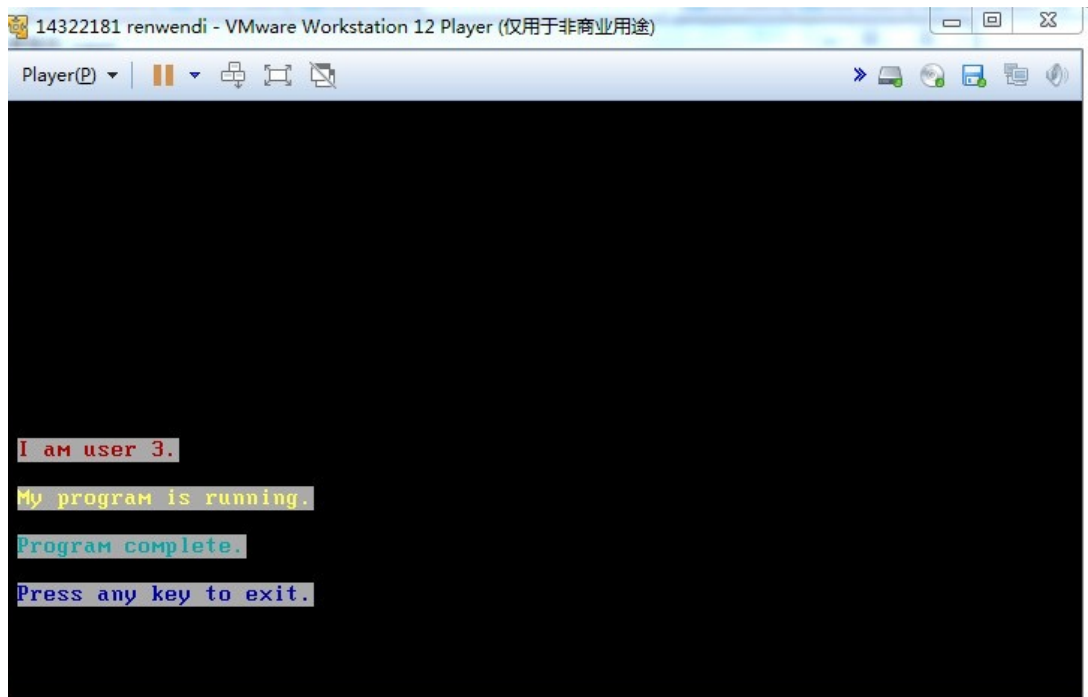
Use4:



Use2:



Use3:



(2)用WinHex验证程序填充验证程序填充扇区：

①：引导程序正确结束

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
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	55	AA		Ua
00000200	E8	19	01	E8	33	01	8C	C8	8E	D8	8E	C0	B4	13	B0	01	è è3 IEI0IA' °
00000210	B7	00	B3	61	B6	05	B2	10	BD	4C	82	B9	18	00	CD	10	· 3a 2 2L I' í
00000220	B4	13	B0	01	B7	00	B3	61	B6	07	B2	10	BD	64	82	B9	· ° · 3a 2 2d I'
00000230	3A	00	CD	10	B4	13	B0	01	B7	00	B3	61	B6	09	B2	10	: í ' ° · 3a 2
00000240	BD	9E	82	B9	2A	00	CD	10	B4	00	CD	16	3C	0D	75	5B	2I I' * í ' í < u[
00000250	EB	00	83	3E	CC	82	00	75	02	EB	4B	A1	CC	82	83	E8	è I>I' u èKiIle
00000260	2E	A3	DC	82	E8	9B	00	83	3E	D0	82	00	75	02	EB	36	.fÜIèI I>DI u è6
00000270	A1	D0	82	83	E8	2E	A3	DC	82	E8	86	00	83	3E	D4	82	iDIè.fÜIèI I>OI
00000280	00	75	02	EB	21	A1	D4	82	83	E8	2E	A3	DC	82	E8	71	u èIiOIè.fÜIèq
00000290	00	83	3E	D8	82	00	75	02	EB	0C	A1	D8	82	83	E8	2E	I>OI u è iOIè.
000002A0	A3	DC	82	E8	5C	00	E9	57	FF	EB	FE	B4	0E	B3	00	FF	fÜIè\ éwyèp' ° y
000002B0	06	C8	82	83	3E	C8	82	01	74	02	EB	07	A2	CC	82	CD	ÈI>ÈI t è çIÍ
000002C0	10	74	85	83	3E	C8	82	02	74	02	EB	09	A2	D0	82	CD	tI>ÈI t è çDIÍ
000002D0	10	0F	84	73	FF	83	3E	C8	82	03	74	02	EB	09	A2	D4	IsyI>ÈI t è çO
000002E0	82	CD	10	0F	84	61	FF	83	3E	C8	82	04	74	02	EB	09	IÍ IayI>ÈI t è
000002F0	A2	D8	82	CD	10	0F	84	4F	FF	83	3E	C8	82	05	0F	8D	çOIÍ IOyI>ÈI
00000300	46	FF	B8	00	00	8E	C0	BB	00	A7	B8	01	02	BA	00	00	Fý, IÀ» S, °
00000310	B5	00	8A	0E	DC	82	CD	13	E8	E5	24	C3	B8	00	B8	8E	µ I ÜIÍ èà\$Ã, I
00000320	C0	B8	00	00	B9	9F	0F	BB	00	00	26	C6	07	20	43	26	À, 'I » &Æ C&
00000330	C6	07	61	43	39	CB	7E	F2	C3	B8	00	00	A3	C8	82	A3	Æ aC9E~oÃ, fÈIf
00000340	CC	82	A3	D0	82	A3	D4	82	A3	DC	82	C3	57	65	6C	63	îIfDIèOI fÜIÄwelc

②：操作系统未超过1个扇区

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
000002D0	10	0F	84	73	FF	83	3E	C8	82	03	74	02	EB	09	A2	D4	IsyI>ÈI t è çO
000002E0	82	CD	10	0F	84	61	FF	83	3E	C8	82	04	74	02	EB	09	IÍ IayI>ÈI t è
000002F0	A2	D8	82	CD	10	0F	84	4F	FF	83	3E	C8	82	05	0F	8D	çOIÍ IOyI>ÈI
00000300	46	FF	B8	00	00	8E	C0	BB	00	A7	B8	01	02	BA	00	00	Fý, IÀ» S, °
00000310	B5	00	8A	0E	DC	82	CD	13	E8	E5	24	C3	B8	00	B8	8E	µ I ÜIÍ èà\$Ã, I
00000320	C0	B8	00	00	B9	9F	0F	BB	00	00	26	C6	07	20	43	26	À, 'I » &Æ C&
00000330	C6	07	61	43	39	CB	7E	F2	C3	B8	00	00	A3	C8	82	A3	Æ aC9E~oÃ, fÈIf
00000340	CC	82	A3	D0	82	A3	D4	82	A3	DC	82	C3	57	65	6C	63	îIfDIèOI fÜIÄwelc
00000350	6F	6D	65	20	74	6F	20	52	65	6E	20	57	65	6E	64	69	ome to Ren Wendi
00000360	20	4F	53	2E	54	68	65	72	65	20	61	72	65	20	34	20	OS.There are 4
00000370	75	73	65	72	20	70	72	6F	67	72	61	6D	73	2C	79	6F	user programs,yo
00000380	75	20	63	61	6E	20	63	68	6F	6F	73	65	20	61	6E	79	u can choose any
00000390	20	6F	72	64	65	72	20	74	6F	20	52	75	6E	2E	49	6E	order to Run.In
000003A0	70	75	74	20	74	68	65	20	6E	75	6D	62	65	72	20	61	put the number a
000003B0	73	20	65	78	61	6D	70	6C	65	73	3A	31	33	32	2F	31	s examples:132/1
000003C0	32	2F	31	33	34	32	3A	20	00	00	00	00	00	00	00	00	2/1342:
000003D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000003E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000003F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000400	E8	64	00	8C	C8	8E	C0	8E	D8	BD	8A	A7	B9	0C	00	B4	èd IEIÀI0%IS' °
00000410	13	B0	01	B3	F4	B7	00	B6	02	B2	01	CD	10	E8	55	00	· °ó. 2 í èU
00000420	B4	13	B0	01	BD	96	A7	B9	16	00	B3	FE	B7	00	B6	04	· ° %IS' °p. 2
00000430	B2	01	CD	10	E8	3E	00	B4	13	B0	01	BD	AC	A7	B9	11	2 í è> · ° %S'
00000440	00	B3	F3	B7	00	B6	06	B2	01	CD	10	E8	27	00	B4	13	°ó. 2 í è' °
00000450	B0	01	BD	BD	A7	B9	16	00	B3	F1	B7	00	B6	08	B2	01	· ° %S' °ñ. 2
00000460	CD	10	B4	00	CD	16	C3	B8	00	06	B7	00	B9	00	00	BA	í ' í Ã, · ' 2
00000470	4F	18	CD	10	C3	BA	00	00	B9	00	00	41	81	F9	30	75	O í Ãe ' A ùOu
00000480	75	F9	42	81	FA	30	75	75	EF	C3	49	20	61	6D	20	75	uùB úOuuiÄI am u
00000490	73	65	72	20	31	2E	4D	79	20	70	72	6F	67	72	61	6D	ser 1.My program

【实验总结】

1 本实验可改进之处

(1) 没有用Bochs进行调试，之后的实验应该学会用Bochs调试，并编写Bochs配置环境。用Vmware Player每次都启动比较麻烦，也没有办法单步调试和设置断点。

(2) 本次实验首先尝试了在Linux下进行开发，但是Bochs一直无法释放鼠标，在进入虚拟机后就无法退出，只能进入终端强制退出，因此最后还是选择了在Windows下进行。之后文件系统的实验可以尝试在Linux下用GCC+NASM开发。

(3) 用户程序显示过于简单，第二版将时间全部用在批处理的实现上，因此为对子程序做进一步的修改。汇编代码编写能力还有待提高。

(4) 没有实现分时，对分时的理论概念还不够理解，不知道分时如何在汇编中实现。

2 实验心得

本次实验通过两个版本终于实现了批处理。进行的过程比较缓慢，因为对一些比较难的功能的实现觉得很茫然。在第三个实验以及以后的实验中，最后还是要达到批处理和分时的效果。

自己解决一个新问题的过程中一定会遇到很多的bug，此时一定不能着急，对着代码一直看也是无法解决问题的，而应该不断尝试，查阅资料或与同学交流，推进实验的进程。如本实验中的incbin操作就是互相交流之中学习到的，对实验的进行提供了很多便捷。此外，批处理的实现也是在自己有了初步想法后，与同学交流发现大家都选取了类似的方法，在数组中存储，因此进一步推进实验。

操作系统的理论知识很重要，对内存和地址等的模糊理解会给实验带来很大的麻烦，因此理论知识一定要清晰。