实验一 接管裸机的控制权

何泽 18340052

一、 实验目的

- 搭建和应用实验环境
- 接管裸机的控制权

二、 实验要求

1. 搭建和应用实验环境

> 虚拟机安装，生成一个基本配置的虚拟机XXXPC和多个1.44MB容量的虚拟软盘，将其中一个虚拟软盘用DOS格式化为DOS引导盘，用WinHex工具将其中一个虚拟软盘的首扇区填满你的个人信息。

2. 接管裸机的控制权

> 设计IBM_PC的一个引导扇区程序，程序功能是：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。将这个程序的机器码放进放进第三张虚拟软盘的首扇区，并用此软盘引导你的XXXPC，直到成功。

三、 实验方案

1. 相关基础原理

- x86汇编
- 操作系统启动原理与顺序（之前交过的文档有描述，此处就不再赘述）
- 扇区引导原理

  > 把Bootloader加载到内存中的固定地址0x7C00，之后CS：IP地址改变指向0x7C00，运行Bootloader。Bootloader加载OS后，CS：IP地址指向OS在内存中的首地址，运行操作系统。

2. 实验工具和环境

- 平台：纯Windows
- 汇编工具：nasm

- 命令行工具：Cygwin
- 创建空白软盘文件：Bochs Disk Image Creation Tool
- 虚拟机：VMware Workstation

3. 实验流程与思路

四、 实验过程和结果

1. 编写引导扇区程序和操作系统

```
    org 07c00h                  ; 程序加载到100h，可用于生成COM
    Dn_Rt equ 1                 ;D-Down,U-Up,R-right,L-Left
    Up_Rt equ 2
    Up_Lt equ 3
    Dn_Lt equ 4
    delay equ 50000             ; 计时器延迟计数,用于控制画框的速度
    ddelay equ 580              ; 计时器延迟计数,用于控制画框的速度
;以下一段为引导扇区程序
    mov ax,cs
    mov ds,ax                   ; DS = CS
    mov es,ax                   ; ES = CS
    mov ax,0b800h
    mov gs,ax
    call name   ;显示学号和姓名
    call start  ;显示数字
    jmp $

name:
    mov bp, 0     ; BP=当前串的偏移地址
    mov byte[gs:bp+0],'1'
    mov byte[gs:bp+2],'8'
    mov byte[gs:bp+4],'3'
    mov byte[gs:bp+6],'4'
    mov byte[gs:bp+8],'0'
    mov byte[gs:bp+10],'0'
    mov byte[gs:bp+12],'5'
    mov byte[gs:bp+14],'2'
    mov byte[gs:bp+16],'H'
    mov byte[gs:bp+18],'e'
    mov byte[gs:bp+20],'Z'
    mov byte[gs:bp+22],'e'
    ret

start:
    mov ax,0B800h               ; 文本窗口显存起始地址
    mov gs,ax                   ; GS = B800h
    mov byte[char],'0'

loop1:
    dec word[count]             ; 递减计数变量
    jnz loop1                   ; >0：跳转;
```

```
    mov word[count],delay       ;延时

    dec word[dcount]            ; 递减计数变量
    jnz loop1
    mov word[count],delay
    mov word[dcount],ddelay     ;延时

    mov al,1
    cmp al,byte[rdul]
    jz  DnRt
    mov al,2
    cmp al,byte[rdul]
    jz  UpRt
    mov al,3
    cmp al,byte[rdul]
    jz  UpLt
    mov al,4
    cmp al,byte[rdul]
    jz  DnLt                    ;决定运动方向
    jmp $

DnRt:                           ;向右下运动
    inc word[x]
    inc word[y]
    mov ax,word[x]
    cmp ax,81                   ;判断是否已经超界,x取值范围0-79
    jz  dr2dl
    mov ax,word[y]
    cmp ax,26                   ;判断是否已经超界,y取值范围1-25
    jz  dr2ur
    jmp show

dr2ur:
    mov word[y],24
    mov byte[rdul],Up_Rt
    jmp show

dr2dl:
    mov word[x],79              ;x-2=78
    mov byte[rdul],Dn_Lt
    jmp show

UpRt:                           ;每走一步，x+1,y-1
    ;mov al,'O'
    ;mov byte[char],al
    inc word[x]
    dec word[y]
    mov ax,word[y]
    cmp ax,1
    jz  ur2dr
    mov ax,word[x]
    cmp ax,81
    jz  ur2ul
    jmp show

ur2ul:
    mov word[x],79
    mov byte[rdul],Up_Lt
```

```
100         jmp show
101
102     ur2dr:
103         mov word[y],3
104         mov byte[rdul],Dn_Rt
105         jmp show
106
107     UpLt:
108         dec word[x]
109         dec word[y]
110         mov ax,word[x]
111         cmp ax,0
112         jz  ul2ur
113         mov ax,word[y]
114         cmp ax,1
115         jz  ul2dl
116         jmp show
117
118     ul2dl:
119         mov word[y],3
120         mov byte[rdul],Dn_Lt
121         jmp show
122
123     ul2ur:
124         mov word[x],2
125         mov byte[rdul],Up_Rt
126         jmp show
127
128     DnLt:
129         dec word[x]
130         inc word[y]
131         mov ax,word[x]
132         cmp ax,0
133         jz  dl2dr
134         mov ax,word[y]
135         cmp ax,26
136         jz  dl2ul
137         jmp show
138
139     dl2dr:
140         mov word[x],2
141         mov byte[rdul],Dn_Rt
142         jmp show
143
144     dl2ul:
145         mov word[y],24
146         mov byte[rdul],Up_Lt
147         jmp show
148
149     show:
150         ; 计算显存地址
151         mov ax,word[y]
152         dec ax
153         mov bx,160
154         mul bx
155         mov cx,ax    ;存储在cx
156         mov ax,word[x]
157         dec ax
```

```
158         add ax,ax
159         add ax,cx
160         mov bp,ax
161         mov ah,byte[color]          ;  0000：黑底、1111：亮白字（默认值为07h）
162         mov al,byte[char]           ;  AL = 显示字符值（默认值为20h=空格符）
163         mov word[gs:bp],ax          ;  显示字符的ASCII码值
164         cmp ah,10
165         jnz addColor
166         mov byte[color],2
167         jmp go
168
169     addColor:
170         inc byte[color]
171
172     go:
173         mov ah,byte[char]
174         cmp ah,57
175         jnz addchar
176         mov byte[char],48
177         jmp goway
178
179     addchar:
180         inc byte[char]
181
182     goway:
183         jmp loop1
184
185     end:
186         jmp $                       ; 停止画框，无限循环
187
188     datadef:
189         count dw delay
190         dcount dw ddelay
191         rdul db Dn_Rt           ; 向右下运动
192         x    dw 3
193         y    dw 1
194         char db 97
195         color db 2
```

2. 将所写asm文件汇编为bin文件

- 用nasm完成

-

```
D:\hz\nasm>nasm boot.asm -o boot.bin
```

3. 创建空白软盘

- 用安装Bochs时一起安装的Bochs Disk Image Creation Tool创建
- 可以自由选择大小

- ```
  Disk Image Creation Tool
  ========================================================================
                                  bximage
    Disk Image Creation / Conversion / Resize and Commit Tool for Bochs
           $Id: bximage.cc 13481 2018-03-30 21:04:04Z vruppert $
  ========================================================================

  1. Create new floppy or hard disk image
  2. Convert hard disk image to other format (mode)
  3. Resize hard disk image
  4. Commit 'undoable' redolog to base image
  5. Disk image info

  0. Quit

  Please choose one [0] 1

  Create image

  Do you want to create a floppy disk image or a hard disk image?
  Please type hd or fd. [hd] fd

  Choose the size of floppy disk image to create.
  Please type 160k, 180k, 320k, 360k, 720k, 1.2M, 1.44M, 1.68M, 1.72M, or 2.88M.
   [1.44M]

  What should be the name of the image?
  [a.img]
  ```

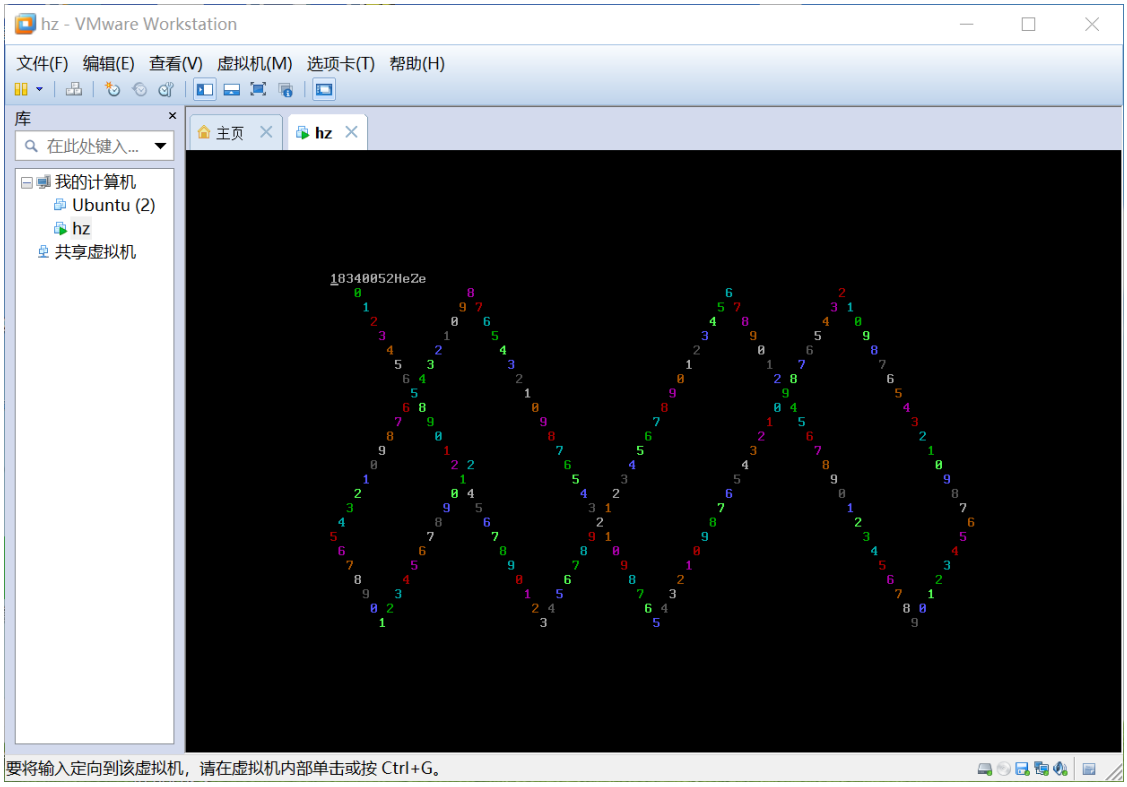
4. 将bin文件写入软盘

- 用Cygwin，使用dd命令
- ```
  /cygdrive/d/hz

  lenovo@LAPTOP-TKLJQ4VS /cygdrive/d/hz
  $ dd if=boot.bin of=boot.img bs=512 count=1 conv=notrunc
  记录了0+1 的读入
  记录了0+1 的写出
  455 bytes copied, 0.0051862 s, 87.7 kB/s
  ```
- 若不用dd命令，将文件后缀名将.bin改为.img在一些情况下也是可以的

5. 启动虚拟机

- 创建新的虚拟机，打开VMware Workstation，启动软盘设为刚刚的img文件
- CD/DVD (IDE) 正在使用文件 D:\hz\hz0.img
软盘 正在使用文件 D:\hz\hz.img
- 启动后便可看到预期界面，实验完成



五、 实验总结

这是操作系统实验的热身项目同时也是第一次实验，由于现在一直在家待着，所以这两天有相对充足的时间去学习相关知识。

当老师刚开始在群里说要去学习操作系统的启动过程的时候我便第一时间去查找的一些相关文献，了解了什么是操作系统。我发现虽然用了这么多年操作系统，现在好像才刚刚知道操作系统真正是什么。

可当老师说可以扩展写一个引导扇区的时候我就彻底懵了，虽然大体知道过程，可当时我脑袋中出现了好多疑问。这个程序该用什么去写？在什么平台上去写？开发的环境是用Windows还是Linux？写完后怎么跑自己的操作系统？该如何测试自己写的引导扇区程序以及操作系统的正确性？当时真是一头雾水。

后来看了一些书和一些文档才渐渐明白上面这些问题的答案，知道了开发自己的操作系统的基本流程。但因为我们上学期的计算机组成原理课程只学了MIPS汇编，并没有涉及到x86汇编，所以我是先看了一些x86汇编的基本语法之后才试着将引导扇区完成。再到后来开发环境的选择问题，我选择了在纯Windows环境中开发。因为汇编工具在两个平台都有，而一些命令行操作在Windows上也可以通过cygwin来实现，省去了在两个操作系统之间转换的时间，也稍微方便一些。

至于虚拟机我先是用了Bochs，并且学习了这个软件的使用方法，而且也在这个上也完成了实验，但是因为这Bochs需要提前写配置文件，相对于VMware要复杂一些，而且界面也不是很好看，所以我后来选择了VMware。

以上便是大体上实验一的总结与我的心路历程，接下来我会接着学习x86汇编以及与操作系统有关的一些知识。

总之，第一次实验让我发现自己写一个属于自己的操作系统是如此有趣的一件事，也激发了我的热情，我会在操作系统这方面一直努力下去的！