实验报告

学院：数据科学与计算机学院　专业：计算机科学与技术　年级：2016 级
实验人姓名（学号）：王锡淮（16337236）
日期：　2018 年　3 月 10 日

实验项目 1 接管裸机的控制权

[实验目的]

1．掌握配置基本虚拟机的方法和相关软件的使用方法。
2．掌握操作系统实验的工具链。
3．了解引导盘的工作原理。
4．了解并熟悉如何 x86 汇编语言的 nasm 格式。

[实验要求]

1. 制作一个用 DOS 格式化的 DOS 引导盘。
2. 用 winhex 工具将一个虚拟软盘的首扇区填满个人信息。
3. 编写一个引导扇区程序，相关细节见【实验内容】。

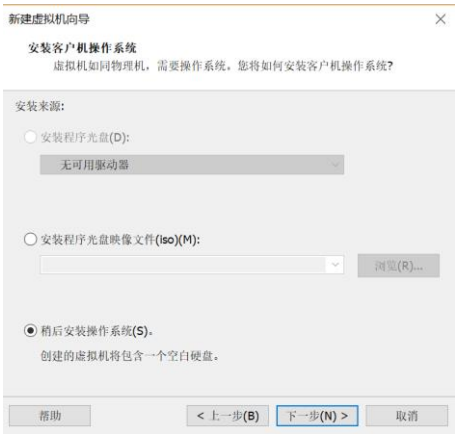
[实验内容]

引导扇区程序的功能为：用字符'O'从屏幕左边第五行位置 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；同时在左上角显示出姓名的拼音以及学号；将屏幕背景色色设置为紫色，并不断改变字符'O'的前景色，实现炫酷动态变色；同时增加交互式清屏功能，按下'c'键则将之前的轨迹清空；在进入主界面之前，还有一个字符画形式的"王锡淮"显示出来。

[实验方案]

- 虚拟机配置方法

本实验在裸机上完成，虚拟机配置时注意选择 MS-DOS 操作系统以及空白硬盘驱动。VMWare 的创建过程种注意选择稍后安装操作系统：



选择“编辑虚拟机设置”添加软盘。

- 实验原理

对于[实验要求]1，实验原理是利用成熟的操作系统（本实验中使用的是 win7）自带的 DOS 格式化功能，将一个空白的虚拟软盘格式化为 DOS 引导盘。

对于[实验要求]2，实验原理是 winhex 会将打开的文件的内容转化为 16 进制。

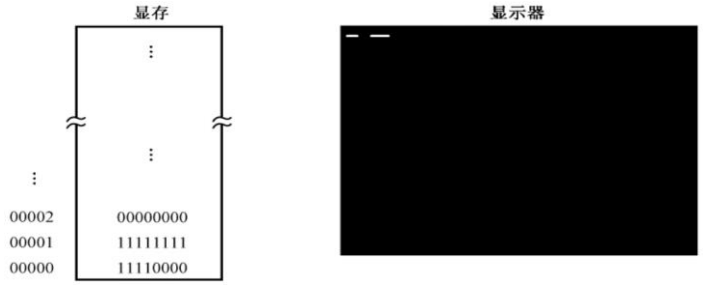
对于[实验要求]3，实验原理是 x86 汇编语言对屏幕的显示进行控制。

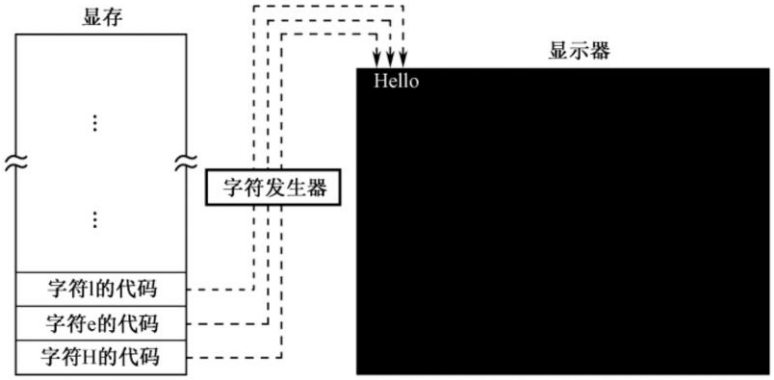
控制屏幕显示的原理是向相应的内存区域输入内容。

显示器的模式分两种：

图形方式：最小可控制单位为像素，VGA：640X400

文本方式：最小可控制单位为字符，VGA：25X80





如上面两张图片可以看出，向内存地址的 B8000~BFFFF 写入对应数据能够在屏幕上显示相关内容。而控制输出的内容的方式如下：

内存地址中没两个字节控制一个字符的内容，高字节为属性，低字节为字符的 ASCII 码。

属性字节的格式：

	7	6	5	4	3	2	1	0
含义	BL	R	G	B	I	R	G	B
	闪烁	背景			高亮	前景		

R：红色
G：绿色
B：蓝色

这就能够实现引导扇区程序的基本功能以及姓名的显示，至于炫酷颜色的改变可以通过不断修改属性字节来完成。

交互式清屏功能的实现：

a. 如何清屏：向 80X25 的屏幕不断输送与文字相同背景色的空格符号，便可以实现清屏功能。

b. 如何交互：要在汇编语言中实现和键盘的交互，可以利用硬件提供的各种中断，在本实验中使用的是 int 16h/00h 和 int 16h/01h 的成套功能，其中 int 16h/00h 的功能是不带回显地读取键盘缓冲区的内容，int 16/01h 的功能是检查键盘缓冲区是否有输入，详细功能如下：

```
AH = 01


on return:
ZF = 0 if a key pressed (even Ctrl-Break)
AX = 0 if no scan code is available
AH = scan code
AL = ASCII character or zero if special function key


- data code is not removed from buffer
- Ctrl-Break places a zero word in the keyboard buffer but does
  register a keypress.
```

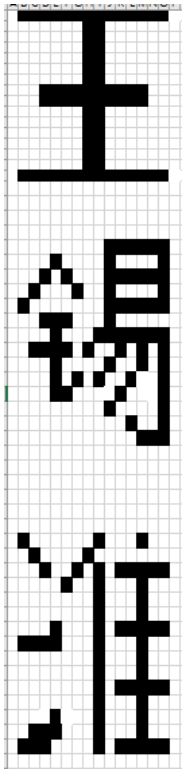
```
AH = 00


on return:
AH = keyboard scan code
AL = ASCII character or zero if special function key


- halts program until key with a scancode is pressed
```

关于如何显示中文名：

如前文所讲，显示屏有 VGA 像素模式，可以对其中的像素点进行操作，设计好正确的字模后对内存进行读写即可。字模如下：



设计为 16X16 的像素点阵。

但是这种方法的一个缺点是在虚拟机上的效果较差，所以采用了后文的字符画方式显示中文名。

关于如何突破 512 字节的限制：

使用 int 16h/02h 可以将其它扇区的内容读取到指定缓冲区，详细如下：

```
AH = 02
AL = number of sectors to read  (1-128 dec.)
CH = track/cylinder number  (0-1023 dec., see below)
CL = sector number  (1-17 dec.)
DH = head number  (0-15 dec.)
DL = drive number (0=A:, 1=2nd floppy, 80h=drive 0, 81h=drive 1)
ES:BX = pointer to buffer


on return:
AH = status  (see INT 13,STATUS)
AL = number of sectors read
CF = 0 if successful
   = 1 if error
```

使用的方法是在调用某个子程序（模块）前要将对应的内容读取到程序应该在的位置，这样就不会造成地址的错误。

- 实验工具以及环境

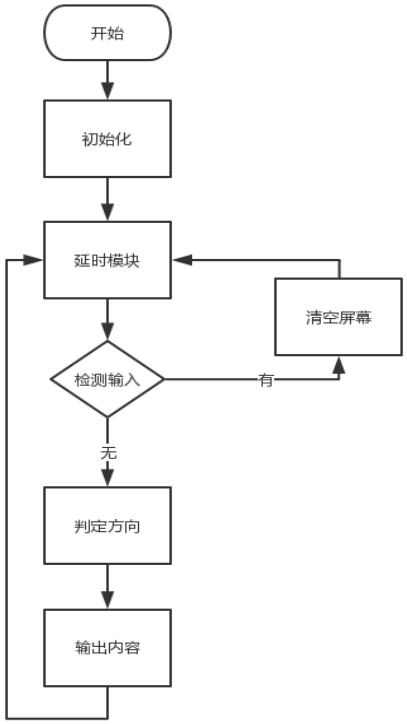
  实验工具链：nasm 编译-winhex 制作 flp 虚拟软驱-VMware 调试。

  尝试过的工具链：nasm 编译-winhex 制作 flp 虚拟软驱-bochs 调试。[1]

  环境：windows 10 以及裸机虚拟机。

- 程序流程和算法思想

  程序流程：



- 数据结构与程序模块功能

  本实验没有用上外部数据，只用内部定义的控制数据以及输出显示数据。

  主要程序模块：

  a. 延时模块：

```
loop1:
  mov ah,01h
  int 16h
  jz loop2
  mov ah,00H
  int 16h
  cmp al, 'c'
  jnz loop2

  call clearprint

  loop2:
    dec word [dcount]
        jnz loop1          ;

    mov word[dcount],delay
    dec word[ddcount]
        jnz loop1
    mov word[dcount],delay
    mov word[ddcount],ddelay
```

  其中 loop1 与 loop2 之间的内容是下面要将的键盘交互模块，延时模块其实是利用了一个双重循环来实现延时，延时的时长由 delay 和 ddelay 的值来确定。

  b. 键盘交互模块：

  如上图，在循环中不断使用 int 16h/01h 来检测是否有键盘输入，如果有，则用 int 16h/00h 来读取，如果输入的是'c'，则清空屏幕，否则继续正常执行。

  c. 显示模块：

```
show:
    ;call clearprint
    call printnames
    inc byte [color]
    and byte [color],0Fh
    or byte [color],50h
    xor ax,ax
    mov ax,word [x]
    mov bx, 80
    mul bx
    add ax,word [y]
    mov bx,2
    mul bx
    mov bx,ax
    mov ah,byte [color]
    mov al,'O'      ;  al
    mov [es:bx],ax      ;
```

显示模块主要利用的是显示屏的大小为 25X80 字符来计算显存地址。

如果当前要显示的字符位置为(x,y)，则显存地址应为 x*160+y*2，因为每个要显示的字符占 2 个字节。

在本次实验中，背景色设置为紫色，即字符属性高四位为 5Fh。

d. 个人信息显示模块：

```
mov al,1
mov bh,0
mov bl,5FH
mov cx,50
mov dx,0005h
push cs
pop es
mov bp,me
mov ah,13h
int 10h
```

该模块主要是调用中断来输出个人信息。

e. 直接写屏模块：

该模块较长，分开说明。

```
printName:
    push ds
    push cs
    pop ds

    PUSH AX
    PUSH ES
    PUSH SI
    PUSH DI
    PUSH CX

    MOV  AH, 0FH   ;读取当前显示方式.
    INT  10H
    PUSH AX
    MOV  AX, 12H   ;设置点阵显示
    INT  10H
    MOV  AX, 0A000H ;显示缓存段地址
    MOV  ES, AX
```

这一部分是切换屏幕的显示模式，切换成 640*400 的像素点模式。

```
DOT DB 7FH, 0FEH, 01H, 80H, 01H, 80H, 01H, 80H, 01H, 80H, 01H, 80H, 1FH, 0F8H
    DB 1FH, 0F8H, 01H, 80H, 01H, 80H, 01H, 80H, 01H, 80H, 01H, 80H, 7FH, 0FEH

DB 0H, 3EH, 08H, 42H, 14H, 7EH, 22H, 22H, 40H, 7EH, 1CH, 40H, 09H, 0FEH, 3DH, 2AH
DB 08H, 4AH, 0AH, 92H, 0CH, 22H, 00H, 42H, 00H, 12H, 00H, 0EH, 00H, 00H, 00H, 00H

DB 40H, 88H, 21H, 00H, 12H, 0BEH, 04H, 88H, 00H, 88H, 00H, 88H, 08H, 0BEH, 78H, 0BEH
DB 00H, 88H, 00H, 88H, 00H, 0BEH, 00H, 88H, 08H, 88H, 38H, 88H, 70H, 0BEH, 00H, 00H
```

这一部分是前文所讲的字模，每个字为 16*16，所以一个字模数据的大小为 32 字节。

```
one:
    MOVSW
    ADD  DI,78      ;相当是加上80
    LOOP one
    pop CX
    ADD DI, 2
    sub di, 16*80
    LOOP printAll
    POP  AX
```

这一部分则是输出字模到屏幕上。

注：这一部分在虚拟机上的效果很差，于是有了下面的突破 512 字节以及显示字符画名字模块。

f. 突破 512 字节模块：

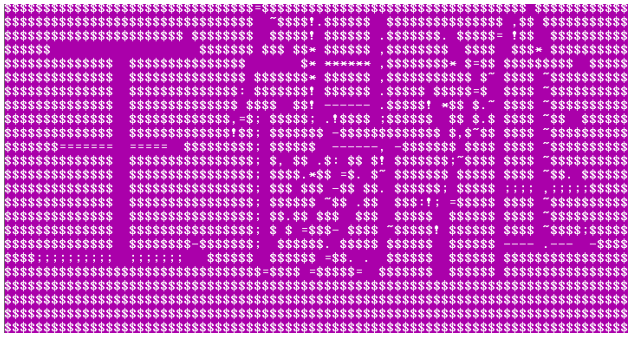
```
load_show:
    mov ah,02h                          ;读磁盘扇区
    mov al,06h                          ;读取6个扇区
    mov ch,00h                          ;起始磁道
    mov cl,02h                          ;起始扇区
    mov dh,00h                          ;磁头号
    mov dl,00h                          ;驱动器号
    mov bx,show                           ;存储缓冲区
    int 13h
    ret
```

该模块调用 int 13h 中断，将从第二号扇区起的 6 个扇区读入 show 标签处。

g. 显示字符画名字：

为了显示中文，采用字符画显示名字的方式，由于一幅字符画需要 2000 个字符，于是将其放在之后的扇区中，再读入。

显示的效果如下：



- 代码文档组成说明

Code 子文件夹下面放的是 stone.asm，为项目代码。

Img 子文件下放的是所需的三个虚拟软盘，first.flp 为 DOS 引导盘，second.flp 为姓

名信息，third..flp 为引导程序盘。

[实验过程]

- 主要流程：

任务 1：先用 vmware 创建一个空的虚拟软盘（second.flp）,随后在一个安装了 win7 系统的虚拟机上面对该软盘进行格式化，同时注意选择使该虚拟软盘成为 DOS 引导盘，就得到了一个 DOS 引导盘。[2]

运行效果如下图：

任务 2：使用如下代码汇编出来的就是所需要的虚拟软盘文件。

```
org 7c00h
times 25 db 'Wang Xihuai 16337236'
db 'Wang Xihuai '
```

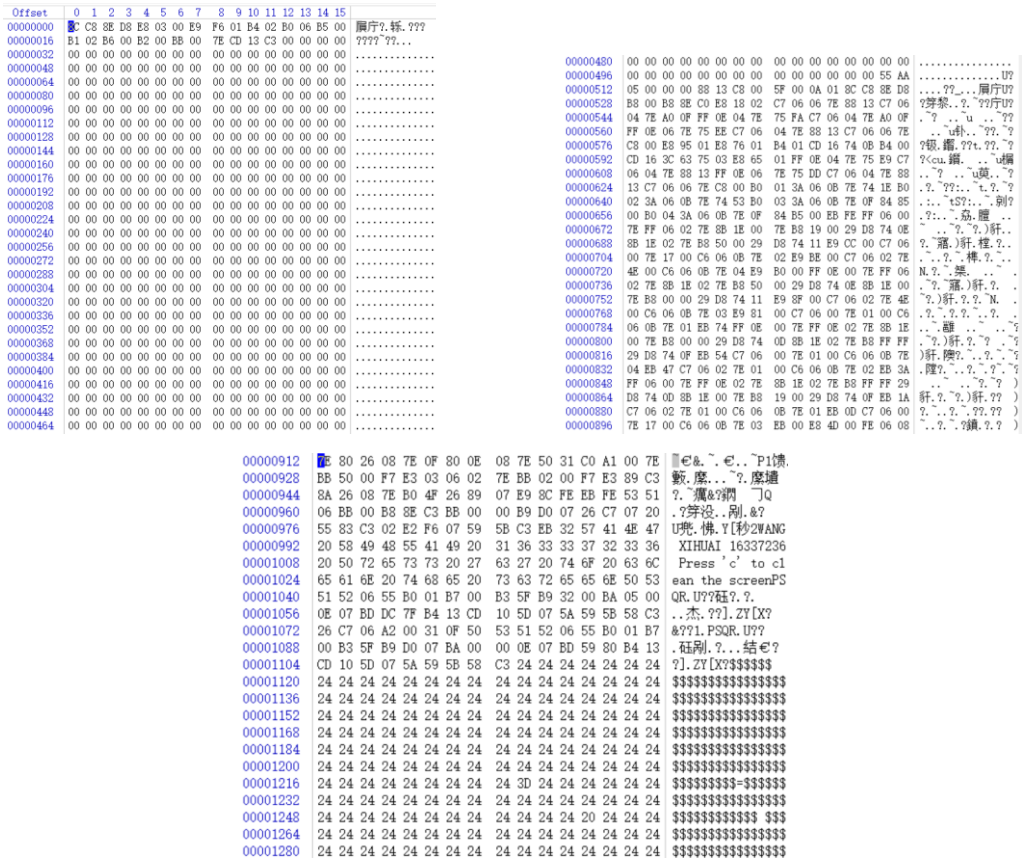
软盘文件展示如下：

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	57	61	6E	67	20	58	69	68	75	61	69	20	31	36	33	33	Wang Xihuai 1633
00000010	37	32	33	36	57	61	6E	67	20	58	69	68	75	61	69	20	7236Wang Xihuai
00000020	31	36	33	33	37	32	33	36	57	61	6E	67	20	58	69	68	16337236Wang Xih
00000030	75	61	69	20	31	36	33	33	37	32	33	36	57	61	6E	67	uai 16337236Wang
00000040	20	58	69	68	75	61	69	20	31	36	33	33	37	32	33	36	Xihuai 16337236
00000050	57	61	6E	67	20	58	69	68	75	61	69	20	31	36	33	33	Wang Xihuai 1633
00000060	37	32	33	36	57	61	6E	67	20	58	69	68	75	61	69	20	7236Wang Xihuai
00000070	31	36	33	33	37	32	33	36	57	61	6E	67	20	58	69	68	16337236Wang Xih
00000080	75	61	69	20	31	36	33	33	37	32	33	36	57	61	6E	67	uai 16337236Wang
00000090	20	58	69	68	75	61	69	20	31	36	33	33	37	32	33	36	Xihuai 16337236
000000A0	57	61	6E	67	20	58	69	68	75	61	69	20	31	36	33	33	Wang Xihuai 1633
000000B0	37	32	33	36	57	61	6E	67	20	58	69	68	75	61	69	20	7236Wang Xihuai
000000C0	31	36	33	33	37	32	33	36	57	61	6E	67	20	58	69	68	16337236Wang Xih
000000D0	75	61	69	20	31	36	33	33	37	32	33	36	57	61	6E	67	uai 16337236Wang
000000E0	20	58	69	68	75	61	69	20	31	36	33	33	37	32	33	36	Xihuai 16337236
000000F0	57	61	6E	67	20	58	69	68	75	61	69	20	31	36	33	33	Wang Xihuai 1633
00000100	37	32	33	36	57	61	6E	67	20	58	69	68	75	61	69	20	7236Wang Xihuai
00000110	31	36	33	33	37	32	33	36	57	61	6E	67	20	58	69	68	16337236Wang Xih
00000120	75	61	69	20	31	36	33	33	37	32	33	36	57	61	6E	67	uai 16337236Wang
00000130	20	58	69	68	75	61	69	20	31	36	33	33	37	32	33	36	Xihuai 16337236
00000140	57	61	6E	67	20	58	69	68	75	61	69	20	31	36	33	33	Wang Xihuai 1633
00000150	37	32	33	36	57	61	6E	67	20	58	69	68	75	61	69	20	7236Wang Xihuai
00000160	31	36	33	33	37	32	33	36	57	61	6E	67	20	58	69	68	16337236Wang Xih
00000170	75	61	69	20	31	36	33	33	37	32	33	36	57	61	6E	67	uai 16337236Wang
00000180	20	58	69	68	75	61	69	20	31	36	33	33	37	32	33	36	Xihuai 16337236
00000190	57	61	6E	67	20	58	69	68	75	61	69	20	31	36	33	33	Wang Xihuai 1633
000001A0	37	32	33	36	57	61	6E	67	20	58	69	68	75	61	69	20	7236Wang Xihuai
000001B0	31	36	33	33	37	32	33	36	57	61	6E	67	20	58	69	68	16337236Wang Xih
000001C0	75	61	69	20	31	36	33	33	37	32	33	36	57	61	6E	67	uai 16337236Wang
000001D0	20	58	69	68	75	61	69	20	31	36	33	33	37	32	33	36	Xihuai 16337236

任务 3：首先要对老师提供的 stone.asm 文件进行编辑，然后用如下指令：

Nasm.exe -f bin stone.asm -o stone.com

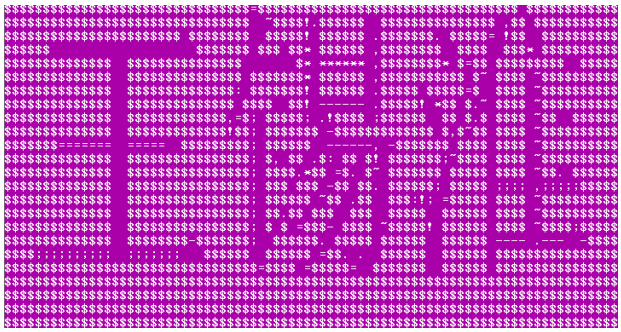
对代码进行编译，然后使用 winhex 得到一个 third.flp 文件，即为虚拟软盘，如下：



后面的是字符画的数据，不予展示。

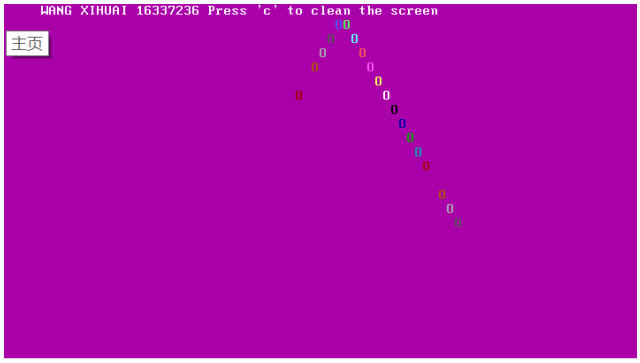
再用 VMware 运行，方法是设置=>添加软盘=>选择 third.flp=>开启虚拟机，得到的结果如下：

首先是欢迎界面：



等待数秒后进入反射球界面。

此时，按下'c'键可以清空屏幕上的小球。



- 输入输出说明：本程序的输入只有键盘的交互，当按下'c'时屏幕进行清屏操作。输出为向屏幕中输出姓名拼音、学号、提示语句、动态变色的字符'O'。
- 遇到的问题以及解决清况：
  a. NASM 格式的转变，上个学期学习 X86 汇编语言的时候学习的是 MASM 格式，在本次实验中要转变格式，通过对于 NASM 官网内容的学习，总结主要需要注意的区别。
     Masm 是微软专门为 windows 下的汇编编写的汇编器，而 nasm 是跨平台的汇编器，能在 windows 和 linux 下运行。
     i. Nasm 格式区分大小写。
     ii. 在 nasm 语法里面，对 memory 操作需要使用[]括号，否则视为标签或地址。
     iii. 指明 memory 操作数的大小的方法不需要'ptr'。
     iv. $$指所在 section 的起始偏移量，而$指当前所在指令位置的偏移量。
     v. Nasm 可以使用 times 来重复写数据，这个伪指令可以用来补全 510 个字节，在其后再加上 0x55AA。
  b. Jg 不跳转的问题。在使用 jg 来检测 dec 指令是否将操作数减少到 0 时，出现了意想不到的问题，dec 将操作数减少到零后 jg 没有跳转，查阅资料后也没有得出解决方法，将 jg 改为 jnz 后问题得到解决。
  c. 中文名字的显示效果不佳问题。如[实验方案]部分所说，通过直接写像素点的方法可以使得屏幕显示出中文，但是在 VMware 虚拟机上运行的效果不佳。
  d. Org 指令的理解。Org imm 指令的作用是将汇编出来的程序放在 imm 处起始。如果要使得 nasm 汇编出来的可执行文件能够在 windows cmd 中运行，需要 org 100h，而如果要将该可执行文件放入虚拟软盘用作引导盘，则需要 org 7c00h。
  e. 边界问题。按照老师所给的基本代码，存在字符的位置移动到屏幕的四个顶点便出界的问题。该问题可以通过修改算法，让小球在到达这些顶点的时候原路反弹，来解决；也可以通过选择适当的起始位置来避免出界情况的发生。
  f. 对于系统读取引导盘的机制的理解。系统会读取引导盘的前 512 个字节（第一扇区）来作为引导程序，而且前 512 个中必须含有 0x55AA 用作标志。所以超出 512 个字节的内容不能直接放在第一个扇区中，应该通过[实验方案]所讲的方法接口的实现放在其它扇区，而将接口的调用放在第一个扇区。
  g. 如何调试代码。在我没有尝试使用 bochs 的时候，调试的方法是输出调试，每隔一段程序向屏幕输出特定内容；而在 bochs 里面就可以像用 gdb 一样调试，较为方便，提高了效率。
  h. 突破 512 字节遇到失败。我认为这个失败的原因是不了解内部的原理，只是照搬[5]的代码，导致的错误。最开始的时候我是通过在源代码中一部分使用首扇区，

一部分使用载入的扇区的方式，call 指令不能成功；后来将几乎所有代码放在了载入的部分，就成功了。虽然问题解决了，但是原因还是不懂。

[实验总结]

对于本次实验要用到的原理，在完成本次实验的过程中，让我记忆最深刻的是从 MASM 格式到 NASM 格式的转变，可能是因为最开始学习的就是 MASM 格式的汇编语言，所以在改变语法的时候有点不适应，但是后来发现使用 NASM 格式还能省掉对于偏移量的考虑与处理，就体会到了这个格式的好处，在 NASM 格式中每隔变量名都是地址，能够剩下很多功夫。还有 times 的使用能够是我在汇编过程中就补上扇区后边的空白，而不需要手动填充。

还有就是对于如何显示中文姓名的学习。在学习如何显示中文姓名的过程中，学会了利用 excel 建立字的点阵，也学会了改变屏幕的显示方式，以及相应中断的调用。对于 VGA 的两个模式也有了了解。

关于 512 字节的限制，我刚知道系统只会将第一扇区的内容作为引导程序的时候是很疑惑的，因为我对 512 字节能够引导出一个操作系统感到不可思议，后来我的代码汇编出来的程序超过了 512 字节，但是没找到将更多的代码作为引导程序的方法，最后在同学的帮助下（同学给我提供了[5]），解决了这个问题，使得没有了 512 个字节的限制。

而在实验过程中，有几个地方花费了我最多的时间和精力。其一是繁琐的工具链，不能通过写脚本一劳永逸，只能用鼠标点击，所以后来看了[1]之后，用了 bochs 来进行调试，并且因为这是命令行能够完成的操作，能够写一个配置脚本，生下了相当多的功夫。其二是调试很困难，在使用 VMware 的时候，只能通过输出调试，然后我尝试了 dosbox，这个工具比 VMware 调试方便，但是在显示方面不如人意，最后还是 bochs 的调试功能能够让我省心。其三是我没用解决的一个问题，就是上面提到的 jg 没有跳转的问题。

关于如何将这个基本代码改编成游戏，我的想法是在最低一行显示一个横板，键盘左右键移动横板，并且计算分数。用到的原理包括屏幕显示、检测并读取键盘输入、在屏幕上显示数字。前两个原理已经在前面的基础实验上使用了，所以我只将第三个原理应用，编写了一个能够显示两位十进制数的子程序（模块），方法是逐位取商。

通过这个实验，我对于怎么样在裸机上实现一个操作系统产生了想法，以及这个操作系统应该具备哪些功能，想来实现一个操作系统会很有成就感。另外我建议老师能够教一下怎么在 windows 下使用 bochs，这方面的网络资源真的不算多。

[参考文献]

[1] Orange's 一个操作系统的实现，于渊，电子工业出版社。

[2] http://www.linuxfly.org/post/537/，VMware 中软盘映像.flp 的使用。

[3] http://www.voidcn.com/article/p-wkrjpaua-ru.html，VMware 中软盘镜像文件*.flp 使用方法。

[4] http://www.nasm.us/doc/nasmdoc0.html，NASM-The Netwide Assembler。

[5] http://blog.csdn.net/dr_flower/article/details/48318001，从 boot 读取软盘扇区中的汇编。

[6] http://www.atool.org/img2ascii.php，字符画生成在线工具。