数据科学与计算机学院 超算方向 18340161 田蕊

# 实验一实验报告

# 一、实验目的

1. 搭建和应用操作系统实验环境，熟悉该环境下的操作与使用。
2. 熟悉实验开发工具的使用，包括
    - 汇编语言工具：X86汇编语言
    - 高级语言工具
    - 磁盘映像文件浏览编辑工具
    - 调试工具
3. 熟悉并掌握虚拟机的使用
4. 体验裸机编程
5. 接管裸机的控制权，设计IBM_PC的一个引导扇区程序。

# 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XXXPC，直到成功。

# 三、实验方案

## （一）、虚拟机配置方法

这里使用的虚拟机工具是virtualbox，实现接管裸机的控制权需要配置一个类型和版本都是other的裸机。在新建虚拟电脑部分选择版本和类型，为虚拟机起名并设置虚拟机的位置。之后分配虚拟机的内存大小和硬盘大小，这一部分可以根据个人需要配置，我这里直接采用默认配置。之后在该虚拟机的设置部分添加软盘控制器，就可以添加软盘了。

## （二）、软件工具与作用

1. 记事本：编辑汇编语言
2. NASM：编译汇编语言
3. FloppyWriter：将编译好的.bin文件写入目标软盘镜像文件（其实此处应该可以使用Linux环境下的命令实现）
4. VirtualBox：管理和控制虚拟机

## （三）、方案的思想

编写一个满足条件的汇编程序，将写好的程序保存为.asm格式之后用nasm编译形成.bin文件之后，将.bin文件写入软盘中之后将软盘链接到虚拟机中，进而实现接管裸机的控制权。

如果可以，我希望称呼这个程序为世界上最小的操作系统。对于程序的实现思想大概为将需要输出的字符放入显存的适当位置中，根据变换的方向计算坐标进而计算出字符应该出现的位置。从而使屏幕输出字符

# （四）、相关原理

## BIOS

BIOS就是"基本输入输出系统"，英文全称是“Basic Input/Output System”。

BIOS的功能如下：

- 系统启动配置
- 基本的设备I/O服务
- 系统的加电自检和启动

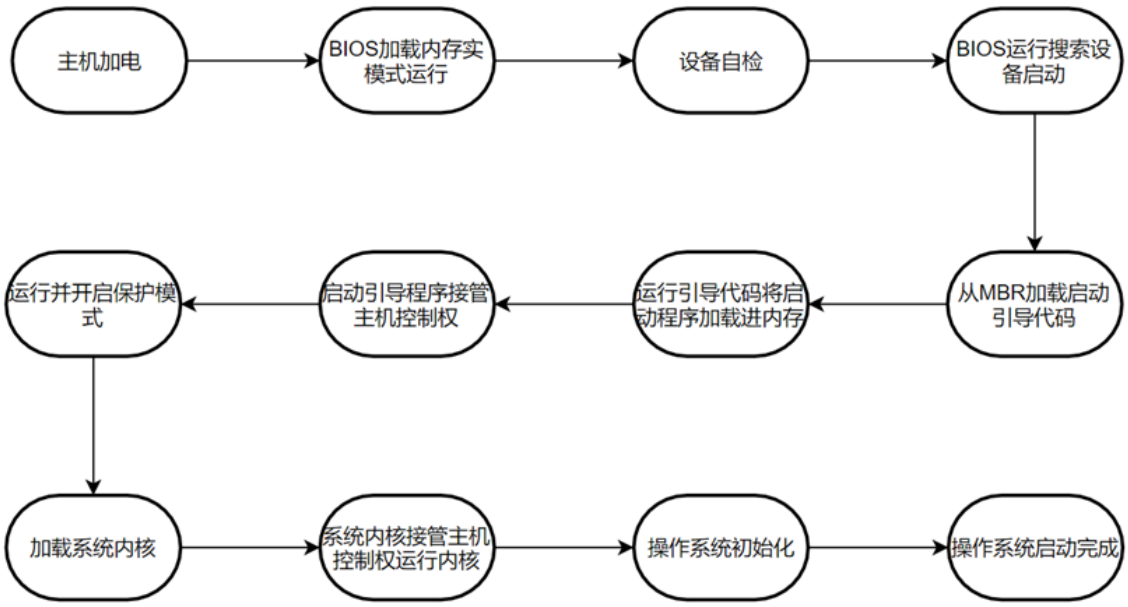
在开机过程中，我们主要关心系统的加电自检和启动。

BIOS首先检查计算机硬件能否满足运行的基本条件，这叫做“硬件自检”（Power-On Self-Test），缩写为POST。如果硬件出现问题，主板会发出不同含义的蜂鸣，启动中止，如果没有问题，屏幕就会显示出CPU、内存、硬盘等信息。

## MBR

在BIOS自取的过程中，他会去读存放在硬盘/软盘的首扇区，这个首扇区内放的内容我们称之为“主启动记录”，也就是MBR，英文全称是“Main Boot Record”，这一部分内容是和操作系统启动的相关信息。主启动记录只有512字节，放不下太多的东西，主要作用是告诉计算机硬盘到哪一个位置去寻找操作系统。整个硬盘的第一个扇区我们用来存放主启动记录MBR，我们实现的目标就是将程序写入首扇区，设计出一个MBR。

## 操作系统的启动过程




## 在屏幕上显示文字

显示器

- 将那些内容以视觉可见的方式呈现在屏幕上

显示卡

- 为显示器提供内容，并控制显示器的显示模式和状态
  - 图形方式：最小可控制单位为像素，VGA：640*400
  - 文本方式：最小可控制单位为字符，VGA：25*80
- 显示卡内存：存放像素或文字及相关属性

### 字符显示原理

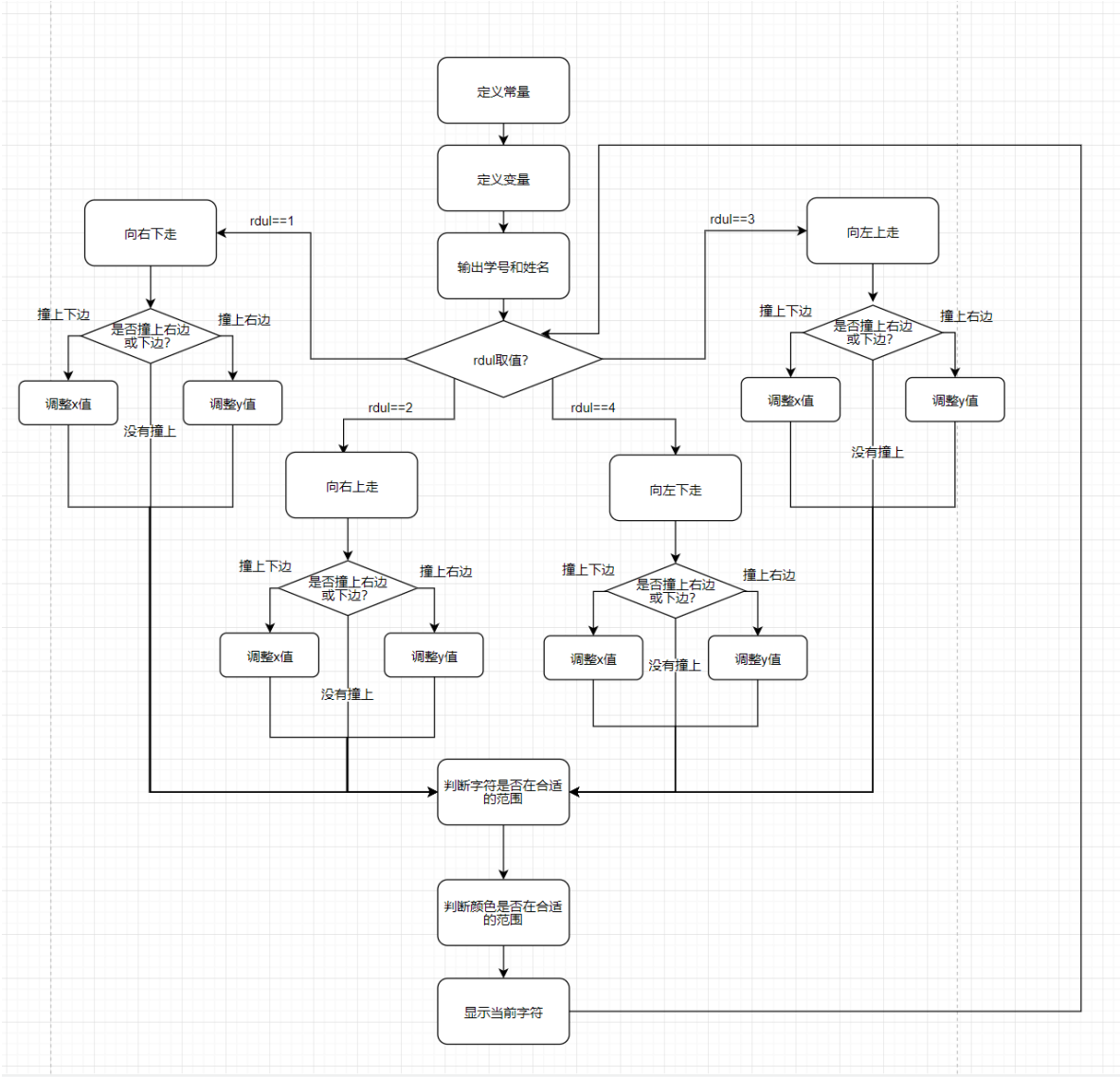
用代码来控制屏幕上的像素，使它们或明或暗以构成字符的轮廓。使用往显存发送数据的方式，就是直接把字符放到显存所在位置。显存所在的段基址是0B8000h，再根据我们想要将字符放入的位置进行算术运算即可得到实际地位置。

### 屏幕上字符的显示属性

八位数据分别为KRGBIRGB，其中KRGB四位表示背景色，IRGB四位表示前景色。具体显示属性如下表

| R | G | B | 背景色 | 前景色 | |
|---|---|---|---|---|---|
| | | | K=0 时不闪烁，K=1 时闪烁 | I=0 | I=1 |
| 0 | 0 | 0 | 黑 | 黑 | 灰 |
| 0 | 0 | 1 | 蓝 | 蓝 | 浅蓝 |
| 0 | 1 | 0 | 绿 | 绿 | 浅绿 |
| 0 | 1 | 1 | 青 | 青 | 浅青 |
| 1 | 0 | 0 | 红 | 红 | 浅红 |
| 1 | 0 | 1 | 品（洋）红 | 品（洋）红 | 浅品（洋）红 |
| 1 | 1 | 0 | 棕 | 棕 | 黄 |
| 1 | 1 | 1 | 白 | 白 | 亮白 |

## （五）、程序流程




# （六）、算法与数据结构

## 数据结构

```
name db "18340161 Tian Rui"  ;字符串变量显示学号姓名
count dw delay               ;延迟变量
dcount dw ddelay             ;延迟变量
rdul db Dn_Rt                ;运动方向标记变量
x    dw 18                   ;标记行方向上位置的变量
y    dw 0                    ;标记列方向上位置的变量
char db 'A'                  ;byte大小的变量表示显示的字符，以大写字母A为初值
color db 01h                 ;byte大小的变量表示显示的颜色
```

## 算法

程序的实现算法非常简单，大体分为计算字符输出为位置，更改字符颜色，更改字符内容几个方面。

### 字符位置

由变量x和y表示当前字符的位置，根据rdul的数值判断当前运行的方向。在计算字符位置的时候要注意是不是碰到了边框位置，如果碰到了要进行反射，即改变字符的横坐标或纵坐标。

### 更改字符颜色

如果要使字符只显示一个颜色，只需要将目标颜色放入显存的相应位置即可，但如果想要改变显示字符的颜色，改变的就是放入现存的不是数值而是一个计数变量，变量数字每次都更改使得现实的字符的颜色也每次都更改。当数值达到最大数值时，再从头开始赋值。

**更改字符内容**

如果要使字符一直不变，只要在现存位置放入常量即可，但如果要使字符不断改变，则要在相应位置放入变量，变量的数字每次都随着字符的位置变换而更改。当属值达到我们所规定的最大值时，再从头开始递增。

# （七）、程序关键模块及代码解释

## 定义变量部分

```
datadef:
    name db "18340161 Tian Rui"  ;字符串变量显示学号姓名
    count dw delay               ;延迟变量
    dcount dw ddelay             ;延迟变量
    rdul db Dn_Rt                ;运动方向标记变量
    x    dw 18                   ;标记行方向上位置的变量
    y    dw 0                    ;标记列方向上位置的变量
    char db 'A'                  ;byte大小的变量表示显示的字符，以大写字母A为初值
    color db 01h                 ;byte大小的变量表示显示的颜色
```

首先是变量定义部分

在汇编语言中我们需要明确的一点是，在汇编语言中并不像C语言中存在各种类型的变量，而是一个个空间，每一个空间都不存在数据类型，它的数据类型在于我们如何用这个变量。所以在定义变量部分不再是由变量类型来定义，而是变量名加上内存单元和初始值 来决定的。这里我们定义一个表示个人信息的变量name。另外char变量大小为一个字节，表示我们需要显示的字符，由于ASCII码的范围是0~256，所以一个字节的空间就已经足够了。同理color变量大小为一个字节，表示我们需要显示的字符的颜色，由于表示颜色为数据为一共只有八位，所以取值范围只能是0~255，所以一个字节的空间就已经结构了。

## 显示学号姓名部分

```
    org 7c00h               ;告诉编译器将程序加载到7c00h处
    mov ax,cs               ;取代码段地址
    mov es,ax               ;将附加段与代码段 指向相同的地址
    mov ds,ax               ;将数据段和代码段指向相同的地址
    mov ax,name             ;输出我的姓名和学号
    mov bp,ax               ;CPU使用ES:BP寻址字符串
    mov cx,17               ;cx保存字符串的长度
    mov ax,01301h           ;ah=13h表示输出字符串，al=01h写模式字符串中只包含字符，写完更新光标位置
    mov bx,000ch            ;bh=0h表示页数为0，BL表示字符串属性，0CH表示红底黑字
    mov dl,0                ;显示位置，DH表示行，DL表示列
    int 10h                 ;调用BIOS中断10H
```

显示学号部分和显示字符的部分的原理是不同的，这里采用BIOS终端的方式实现字符串的显示。

在NASM中，任何不被方括号[]括起来的标签或变量都被认为是地址，访问标签中的内容必须使用[].所以mov ax,name将会把“18340161 Tian Rui”这个字符串的首地址传给寄存器ax。

之后利用BIOS的中断来实现字符串的输出。

## 字母运动部分

### 延迟的设置

```
loop1:
    dec word[count]              ; 递减计数变量
    jnz loop1                    ; >0：跳转;
    mov word[count],delay
    dec word[dcount]                 ; 递减计数变量
    jnz loop1
    mov word[count],delay
    mov word[dcount],ddelay
```

这一部分是设置一个延迟用于控制画框的速度，实际上的延迟时间是delay*ddelay

### 确定字符运动的方向

```
    mov al,1
    cmp al,byte[rdul]
    jz  DnRt                    ;表示向右下走
      mov al,2
      cmp al,byte[rdul]
    jz  UpRt                    ;表示向右上走
      mov al,3
      cmp al,byte[rdul]
    jz  UpLt                    ;表示向左上走
      mov al,4
      cmp al,byte[rdul]
    jz  DnLt                    ;表示向左下走
      jmp $                     ;无限循环
```

根据前文所属，rdul变量表示的是当前走的方向，通过将rdul的数值和各个常量相比较，如果相等就跳转到像这个方向运动的代码处，继续执行下面的指令。 这里其实就相当于一个switch-case语句，根据rdul的值跳转到不同的位置处。

关于$，$表示当前被汇编后的地址，一直跳转到当前的地址，就是无限循环到现在，结束程序的含义。

接下来部分分别是向右下运动，左下运动，右上运动和左上运动部分的代码，这四个部分大同小异，我们就以向右下运动为例来进行分析。

### 字符向右下运动

```
DnRt:
    inc word[x]                 ;因为是向右下运动，所以每次运动x要递增
    inc word[y]                 ;向右下运动，所以每次运动y也要递减
    mov bx,word[x]
    mov ax,25                   ;纵向的边界值是25
    sub ax,bx                   ;判断是不是撞上了下边
      jz  dr2ur                 ;如果撞上了就要跳转到这个函数调整x的值
    mov bx,word[y]
    mov ax,80                   ;横向的边界值是80
    sub ax,bx                   ;判断是不是撞上了右面
      jz  dr2dl                 ;如果撞上了就要跳转到这个函数调整y的值
    jmp judgech
dr2ur:                          ;如果撞上了下面
      mov word[x],23            ;调整x的值来实现反弹
      mov byte[rdul],Up_Rt      ;发生反弹后运动方向也要发生改变
```

```
      jmp judgech
dr2dl:                          ;如果撞上了右面
      mov word[y],78            ;调整y的值来实现反弹
      mov byte[rdul],Dn_Lt      ;发生反弹后运动方向也要发生改变
      jmp judgech
```

这一部分是字符向右下运动的代码，核心思想就是先根据运动方向对x和y进行自增或自减，然后再判断这样你的x和y是否满足现实的需求，如果不满足再根据我们的需要进行调整。

**改变显示的字符**

```
judgech:
      inc byte[char]              ;先让表示显示的字符的ASCII码的变量自增
      mov bl,byte[char]
      mov al,91                   ;91是小写字母a的ASCII码
      sub al,bl
      jz changech                 ;如果超过范围就跳转到调整函数
```

因为我们规定字符的显示从A到Z，所以一旦超过这个范围就要进行调整。同理根据这个原理，只需要改变字符的初始值和调用调整函数是ASCII码的界限，就可以实现现实的字符在不同的范围变化。同理char变量每次增加不同的数值也可以实现字符的不规则变化。

```
changech:
      mov byte[char],'A'
      jmp judgeco
```

这一部分是显示的字符内容的调整。函数实现非常简单只需要将char重新赋为'A'即可，使之从头开始循环。但是需要在意的是接下来的跳转语句，因为我们规定每次新打印一个字符就改变颜色，所以无论字符的ASCII范围是否超出，都要在判读颜色值是否在规定的区间内，所以必须要跳转到判断颜色的函数部分去。

**改变字符的颜色**

```
judgeco:
      inc byte[color]             ;首先让color的值自增使得颜色发生改变
      mov bl,byte[color]
      mov al,10h                  ;10h就是十进制的16
      sub al,bl
      jz changeco                 ;如果超过范围就要跳转到调整颜色函数
      jmp show
changeco:
      mov byte[color],01h
      jmp show
```

我们这里希望字符的背景色恒为黑色，因为我觉得一直改变背景色花里胡哨的不好看。所以表示字符颜色的八位数据最高的四位恒为0，低四位如果全部都是零的话即为黑底黑字则会看不出来，所以前景色由0001b开始变化，当变化到0000b的时候就重新开始循环一遍。所以如果超过了范围就要跳转到调整函数重新将color的值赋为01H。

**字符的显示**

```
show:
    xor ax,ax                   ; 计算显存地址
    mov ax,word[x]
    mov bx,80
    mul bx
    add ax,word[y]
    mov bx,2
    mul bx
    mov bx,ax                   ;bx中是位置
    mov ah,byte[color]
    mov al,byte[char]
    mov [es:bx],ax
    jmp loop1
```

实现字符的显示，我们首先要计算显存的地址，首先我们知到字符是2580，*那么可以知道字符地址应该为0B800H+ (80*x+y)2,*而颜色的地址则为0B800H+ (80*x+y)*2+1.这里将color和char进行组合之后直接放入相应的地址里。

# 四、实验过程

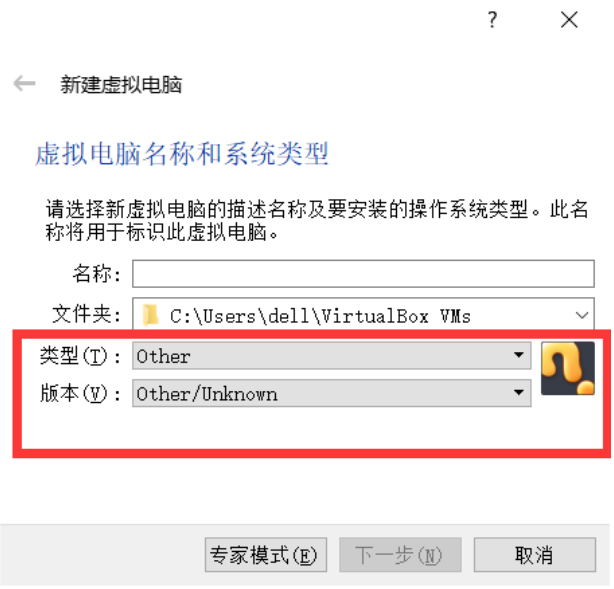
## （一）、主要工具安装使用过程

### VirtualBox

#### 创建虚拟机


图中有三个已经建好的虚拟机，分别是Linux系统，裸机操作系统何一个配置好的dos系统。

裸机的创建




在创建逻辑的过程中，这一部分的类型和版本都选择other，接下来按照默认设置配置内存和硬盘就可以得到一个创建好的虚拟机裸机。

**NASM**

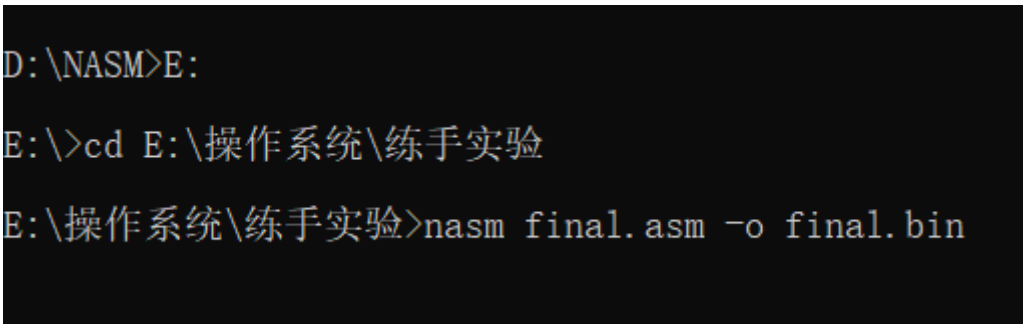
首先到NASM的官网下载安装包，解压之后该包中会包含有四个可执行文件：NASM 可拟行文件'nasm.exe'和'nasmw.exe',还有 NDISASM 可执行文件'ndisasm.exe'和'ndisasmw.exe'。文件名以'w'结尾的是'Win32' 可执行格式。是运行在'Windows 95'或'Windows NT'的 Intel处理器上的，另外的是 16 位的'DOS'可执行文件。

按照指示的流程就可以顺利安装NASM。

在本实验中NASM只需要用到一个指令，即将.asm文件编译成.bin文件的指令，指令如下：

nasm 文件名.asm -o 文件名.bin

但是在执行这一条指令之前需要先进入到.asm文件所在的目录下，操作如下



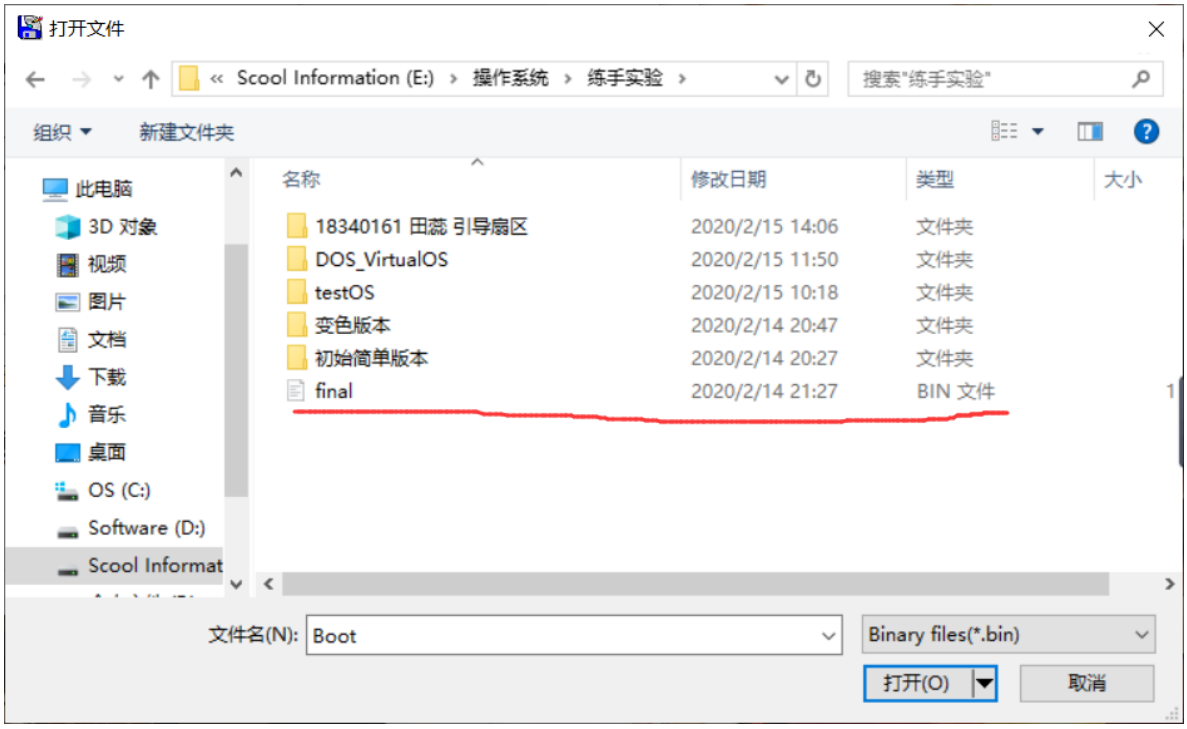

输入如上图所示的指令就可以进入到目的文件夹，然后对文件进行编译

**FloppyWriter**

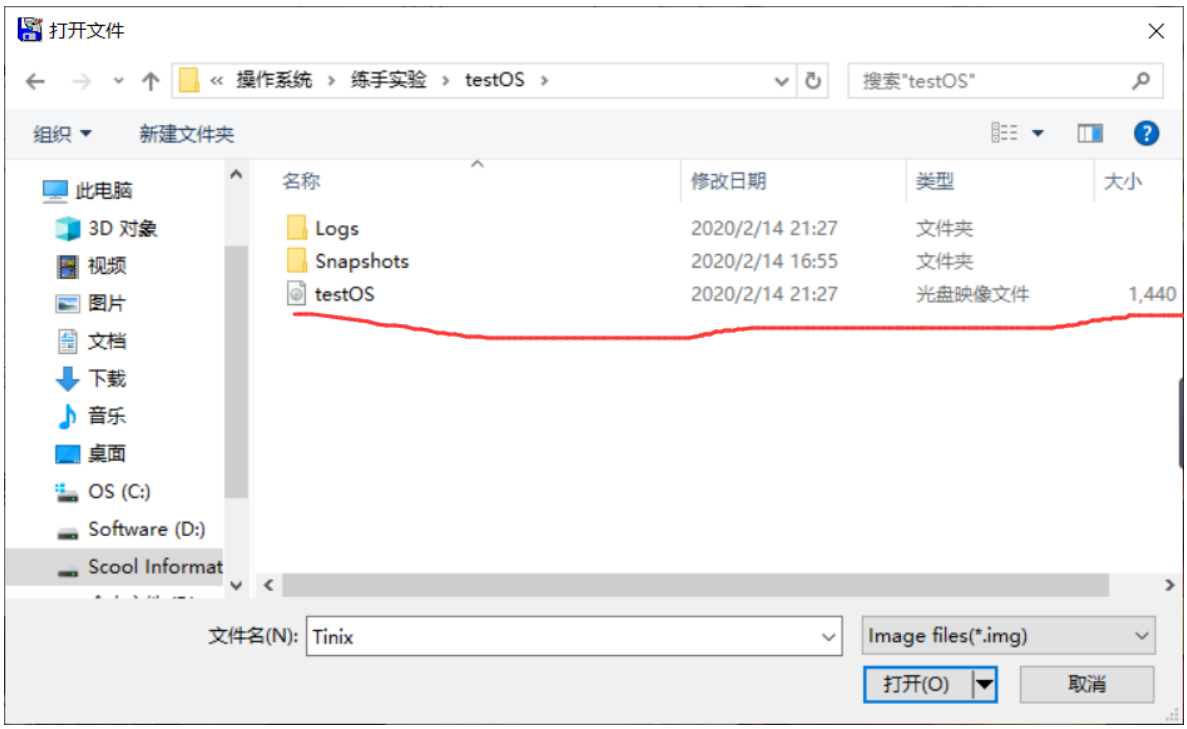
这是一个对软盘和软盘映像进行书写的一个软件，其实这里应该是可以下Linux下进行操作的，但是因为自己不能熟练掌握所以还是借助了工具。

这里操作很简单，打开软件后选择Write File to Image


之后选择我们之前编译好的.bin文件




在选择我们之前就新建好的镜像软盘，镜像软盘可以在virtualBox中直接建立并连接。




之后就可以成功将编译好的.bin文件写入镜像软盘。

## （二）、操作步骤

### 配置虚拟机

由于实验目的是在裸机上实现扇区引导，所以需要在虚拟机上配置一个没有安装任何操作系统的裸机，同时需要新建一个软盘以写入目标程序。具体操作步骤上面已经介绍过了，这里就不再赘述了。

### 编辑代码

为了实现目标程序，根据要求编写代码，实现程序要求。

### 编译代码

通过NASM编译代码，改正语法错误，生成.bin文件。

### 写入软盘

接下来的任务是将目标文件写入配置到虚拟机中的软盘。这里用的工具是floppywrite。

### 虚拟机开机

将虚拟机开机后，程序就会显示出我们想要显示的结果

## （三）、输入输出及说明

### 输入

该实验没有输入

### 输出

本实验结果将在左上角输出本人的学号和姓名。用字符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；输出的字符从‘A’到‘Z'不断变化，输出的字符颜色不断变化，但背景底色一直是黑色。

## （四）、问题及解决方案

1. 在本程序中，输出姓名和学号采用了两种不同的方式。因为输出的学号和姓名是没有规律的，所以在不定义字符串变量的前提下难以用循环的方式实现输出字符。如果手动写出各个字符放在显存的位置则太过冗长，所以在这里输出学号和姓名部分采用了中断的方式。现在程序的最开始输出姓名和学号，之后的部分再用把字符直接放入现存地址的地方的方法输出字符。
2. 在本程序中一个很细节的问题，就是在设置时间延迟的部分，为什么要设置两个时间延迟变量。根据程序我们知道实际的时间延迟应该是两个时间延迟变量相乘。我认为是相乘的结果过大，无法保存在一个变量内，通过阅读代码我们知道，两个时间延迟变量分配的内存都是word的内存，也就是十六位数字，十六位数字能够表示的数据范围-32767 ~ + 32768，而两个延迟数字相乘为29000000，远远超出了十六位数字的表示范围，所以要使用两个循环控制变量来保证延迟时间。
3. 图中红色方框全出的部分我不是很能理解。这句话的含义是将ax中的数据复制到bx中。根据前面的逻辑我们可以知道bx中保存的是显存地址，那么这句话的含义是什么呢？将ax的值赋值给bx不就会破坏掉bx的内容吗？计算好的现存地址又由谁来保存呢？根据我的理解应该是变为

   ```
   mov ax,bx
   ```

   但是通过实验我们发现事实并不是像我所想的那样。所以这个问题至今也没有解决，并不是很能明白这条语句的含义。

   ```
   160   show:
   161       xor ax,ax                    ; 计算显存地址
   162       mov ax,word[x]
   163       mov bx,80
   164       mul bx
   165       add ax,word[y]
   166       mov bx,2
   167       mul bx
   168       mov bx,ax                    ;bx中是位置
   169       mov ah,byte[color]
   170       mov al,byte[char]
   171       mov [es:bx],ax
   172       jmp loop1
   ```

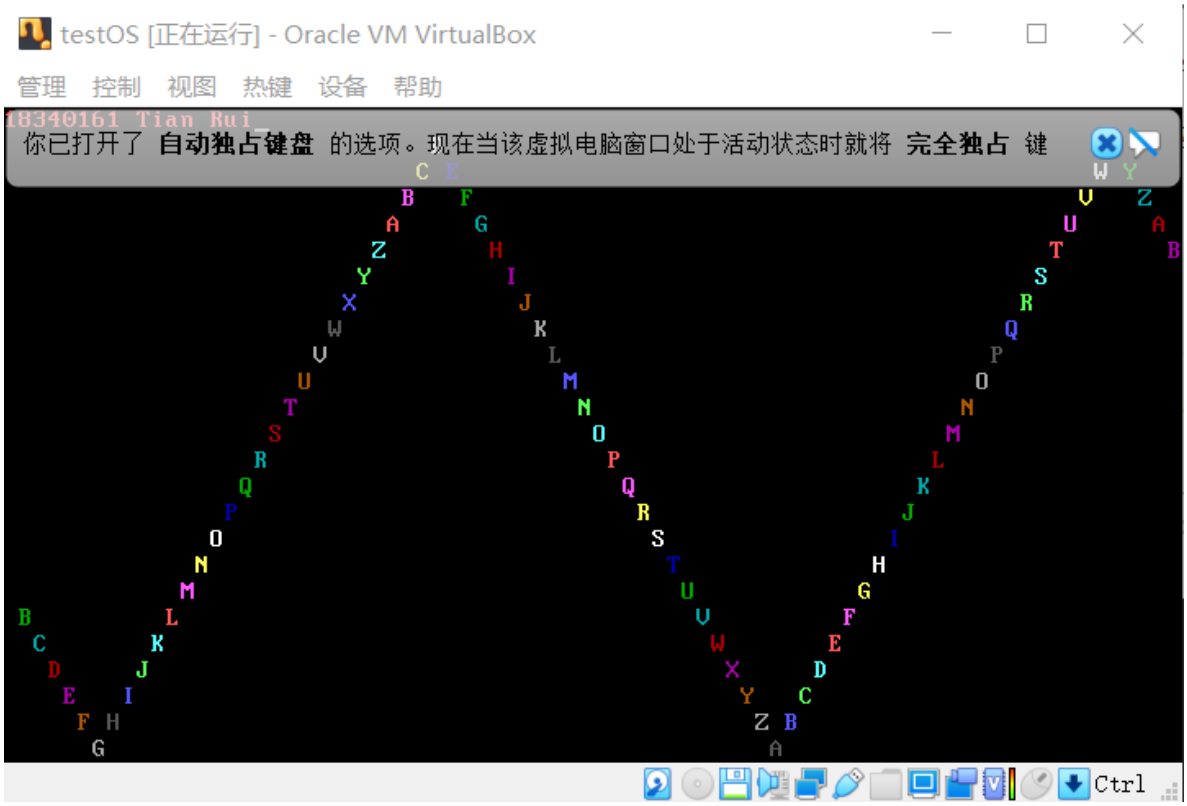
4.

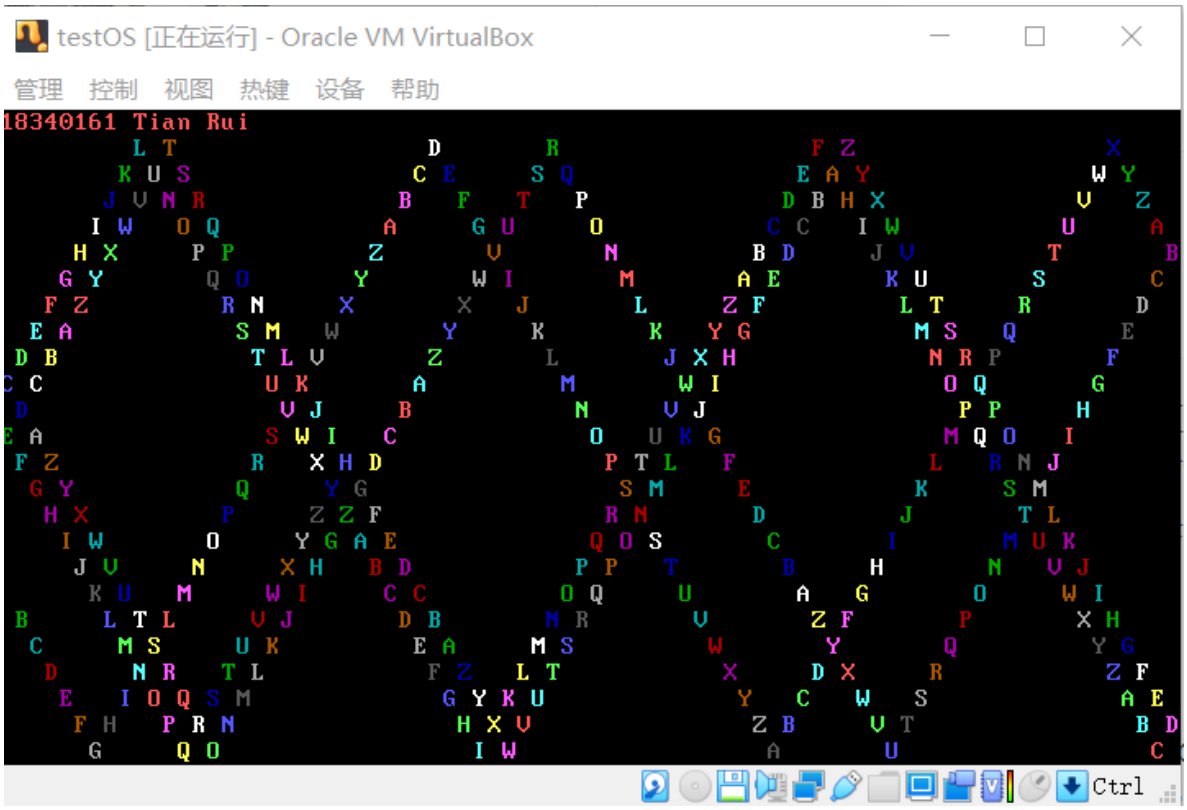
## （五）、关键操作

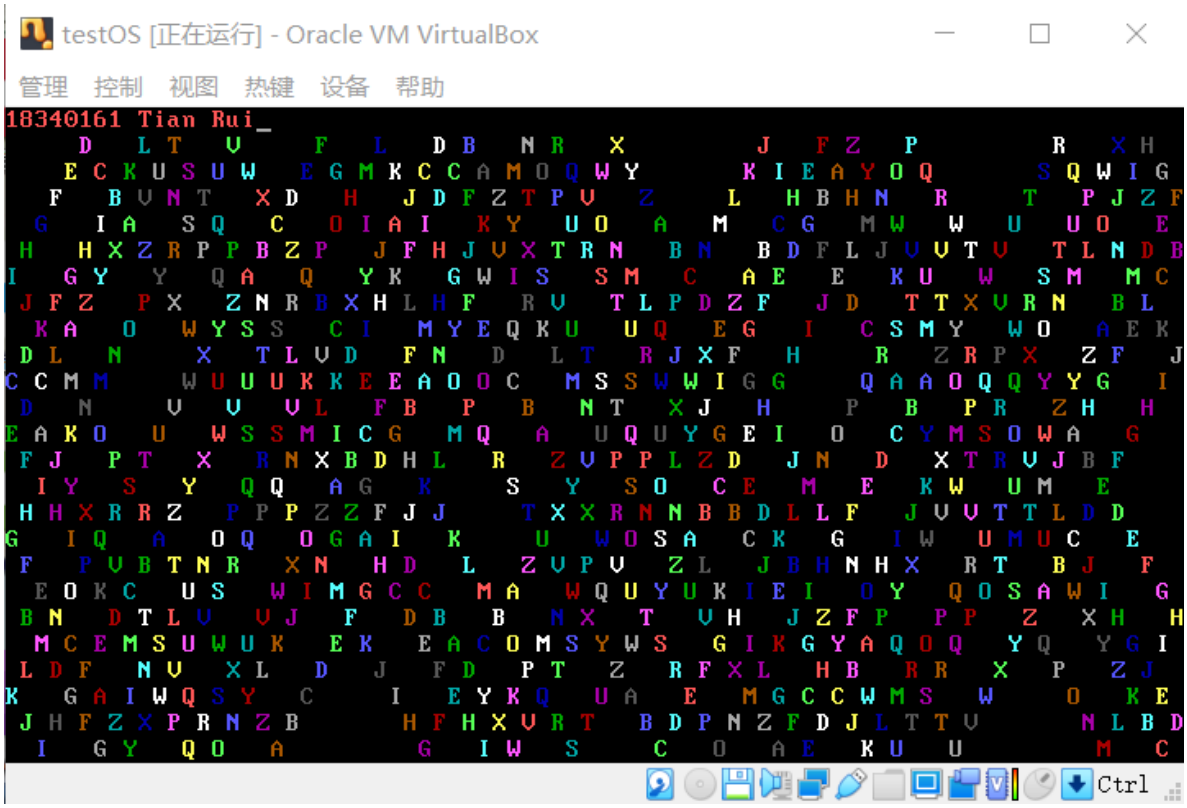
1. 编译文件。编译文件的过程需要将文件编译为二进制文件。
2. 配置虚拟机。配置虚拟机的过程需要配置一个裸机。
3. 写入软盘。写入软盘可以使用的软件有很多，这里采用floppy write

# 五、实验结果

下面只对运行中的几个时刻进行截图，具体操作过程和运行结果可以查看文件中的录屏。


testOS [正在运行] - Oracle VM VirtualBox
管理 控制 视图 热键 设备 帮助
18340161 Tian Rui
你已打开了 自动独占键盘 的选项。现在当该虚拟电脑窗口处于活动状态时就将 完全独占 键
Ctrl


testOS [正在运行] - Oracle VM VirtualBox
管理 控制 视图 热键 设备 帮助
18340161 Tian Rui
Ctrl




# 六、实验总结

这是第一次接触操作系统，试验结束后成就感满满的同时还有那么一丝刻骨铭心。在操作过程遇到了很多的问题，同时也努力的想办法去解决问题，在这过程中有很多的收获但同时也有很多不足，解决了一些问题也有一些残存的问题，接下来我们就好好梳理一下这些问题。

### 实现环境

本实验的实现环境本质上还是在Windows下完成的，事实上实验在Linux系统下实现会更加方便。希望以后可以更多地学习Linux的操作方法，能够在Linux环境下完成实验操作。

### 实现效果

本实验目前只能实现简单的字符反弹和字符变色，其实经过多方面的更改应该可以实现字符更加炫酷的运动效果。

### 实现工具

本实验采用的实现工具是封装好的工具非常的简单，不如在将文件写入软盘的时候可以采用winhex而不是floppywrite，可是实现更加简便的操作。

### 关于学号和姓名的输出

事实上学号和姓名的输出可以采用定义字符串的方式来实现，但是因为不能很好的掌握汇编语法，所以没能实现。

# 七、参考文献

[1]于渊.自己动手写操作系统[M].电子工业出版社:北京市,2005:1.
[2]枫竹梦.操作系统：实现引导扇区[EB/OL].https://blog.csdn.net/furzoom/article/details/52485090,2016-09-09.
[3]nethanhan.nasm的安装方法和使用[EB/OL].https://blog.csdn.net/nethanhan/article/details/8095556,2012-10-21.