# 操作系统实验报告

计科（大数据、人工智能方向）17341155 王永康

## （实验一：引导扇区程序设计）

### 一、实验目的

初步认识操作系统，实现直接从裸机（虚拟裸机）上运行程序，并掌握x86汇编初步编程能力。

### 二、实验内容

- 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。

- 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PC，直到成功。

### 三、实验方案

#### 操作系统实验工具与环境

- 实验支撑环境
  - 硬件：个人计算机
  - 主机操作系统：Linux(Ubuntu 18.04)
  - 虚拟机软件：Bochs
- 实验开发工具：
  - 汇编语言工具：x86汇编语言
  - 汇编编译工具：NASM+GCC
  - 调试工具：Bochs

## 硬件或虚拟机配置方法

- Bochs的配置以及虚拟软盘的创建

  在Linux操作系统下，去网站下载bochs-2.6.8.tar.gz文件，然后在Linux下打开终端，切换进入所下载文件的目录下，输入命令行：

  ```
  sudo tar zxvf bochs-2.6.8.tar.gz
  ```

  然后再进入到解压的文件里面去，进行编译安装，依次输入命令行：

  ```
  ./configure
  sudo make
  sudo update install
  ```

  编译安装是为了使Bochs具有调试功能，之后Bochs就安装完毕了。

  之后，创建虚拟软盘，在终端中输入ximage（这是Bochs配套的一个制作虚拟盘的软件），然后依据软件提示创建一个大小为1.44MB的、名字为a.img的虚拟软盘。

  之后还要修改Bochs的bochsrc配置文件，这个文件决定Bochs虚拟机从哪读取虚拟磁盘等属性，自己创建一个bochsrc.txt文件，内容见附件。然后再终端里面输入

  ```
  bochs -f bochsrc.txt
  ```

  然后，Bochs配置完毕。

- NASM安装

  由于个人Linux操作系统下已安装GCC，仅需再安装NASM,直接在终端下输入命令行：

  ```
  sudo apt-get install nasm
  ```

  然后就安装好了。

## 软件工具与作用

- Bochs作用

  它用于虚拟裸机环境，操作系统开发运行都是在Bochs上，而且Bochs可以用来调试。

- NASM作用

  它用于编译汇编源程序.asm成二进制文件。使用方法，直接在终端下输入：

  ```
  nasm xxx.asm -o xxx.bin
  ```

## 方案的思想

- **引导扇区程序的制作**：利用计算机BIOS的固有特点，把所需要运行的程序的二进制文件写入软盘，并填满软盘的首扇区512字节，并在倒数两个字节写入0xAA55，这样BIOS就会认为这是一个引导扇区，把它加载到计算机的内存07c00h位置之中去，从而运行程序。
- **飞翔弹跳字符程序的实现**：计算机的屏幕输出主要依据显示器和显卡，一般来说直接对显卡的储存器（即显存）赋值，就可以在相应屏幕位置显示相应的字符，也可以通过调用BIOS的中断来实现对屏幕的显示功能。由于调用BIOS中断比直接对显存赋值稍微简单一些，所以我使用BIOS的中断来实现弹跳字符的实现。基本思想是，不断对dh,dl(分别控制字符显示位置的行和列的两个寄存器)进行递增或递减，每次判断行是否到达上界或下界，若到达则反向，列也同理，每改变一次值，调用一次中断来显示字符，同时调用延时程序（通过运行空循环实现），能够被人眼看到动态变化。

## 相关原理

- 引导扇区的原理：

  计算机开机后会首先加电自测(POST)，然后寻找启动盘，因为配置时选择a.img虚拟软盘启动，计算机就会检测**软盘的0面0磁道1扇区**，如果发现它以**0xAA55结束**，则BIOS认为它是一个**引导扇区**，就会把这段512字节内容加载到内存地址0000：7c00处，然后跳转到该地址，BIOS将控制权彻底交给这段引导代码，从此计算机不再由BIOS固有程序控制，从而变成了操作系统一部分来控制。

- 飞翔字符的显示原理：

  计算机显示文本，需要两种硬件：**显示器**和**显卡**。显卡的作用是为显示器提供要显示的内容，并且控制显示器的状态和模式；显示器的作用则是把内容以人可见方式呈现在屏幕上。

  显卡最基本的两种工作模式是文字（也称为文本）模式和图形模式。

  显卡有自己的储存器，称为显示储存器，简称 ”显存” 。

    - 显示字符的方法一：**操作显存**

  通过向显存中写入相应数据，可以在显示器上显示相应内容。显存物理地址是0xB8000到0xBFFFF，共32KB，计算机默认情况下，屏幕可以显示25行*80个字符；

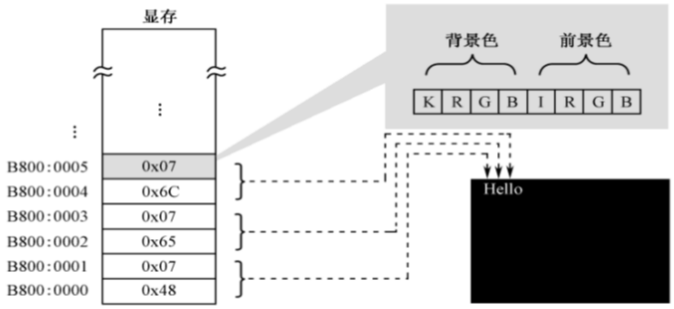
  其中显示包含两部分，一部分是显示的字符，另一部分是字符的属性（前、背景色）(如下图)；

  根据这个显存特点，假设字符要在第x行(0行开始标号)，第y列(0列开始标号)显示，则对应的物理内存映射为：

  写入字符（一个字节）的偏移地址 = (x*80 + y)*2

  写入字符相应属性（一个字节）的偏移地址 = (x*80+y)*2+1

## 显存与屏幕上字符的对应






这是其中一个方法，通过如此映射，操作显存可以完成，stoneM.asm基于此完成。

- 显示字符的方法二：**调用BIOS中断**

  通过调用BIOS的10h中断实现，个人的飞翔字符采用这种方式实现，因为可以实现闪烁，而且直接对dh,dl寄存器的赋值可以直接定位行、列显示，比较方便。

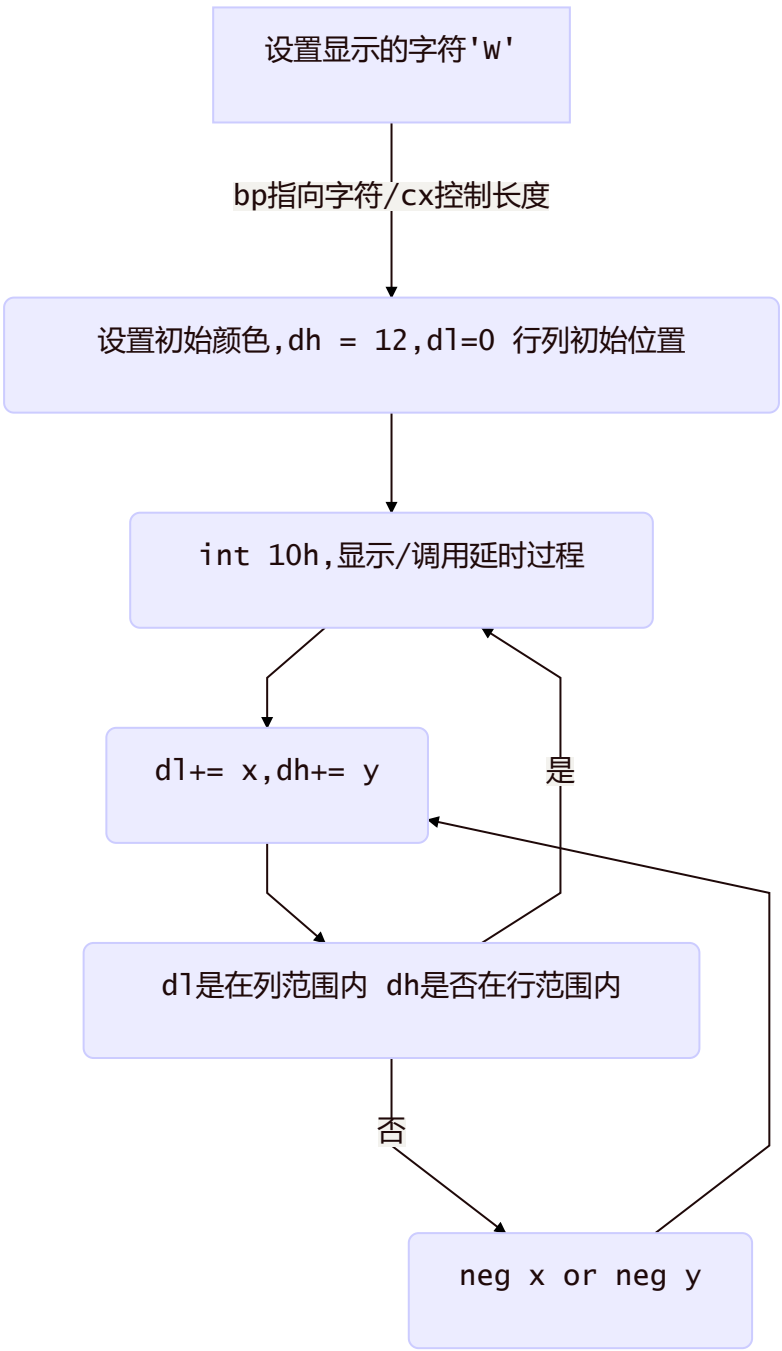
### BIOS的10H调用

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



**程序流程**




**程序关键模块**

判断dl,dh是否到达边界以及循环过程，具体见下列代码及注释：

```
LOO:
    inc bl

    add dl,byte[col]
    add dh,byte[row]
    int 10h

    cmp dh,DOWN ;判断是否到下界
    jne LOO1
    neg byte[row]

LOO1:   cmp dh,UP ;判断是否到上界
    jne LOO2
    neg byte[row]

LOO2:   cmp dl,LEFT ;判断是否到左边
    jne LOO3
    neg byte[col]

LOO3:   cmp dl,RIGHT ;判断是否到右边
    jne LOO4
    neg byte[col]

LOO4:

    call LOOPS


    jmp LOO
```

## 四、实验过程和结果

- 编写汇编源文件.asm
- 利用nasm进行编译，终端下输入

```
nasm test2_2.asm -o test2_2.bin
```

生成test2_2.bin文件；

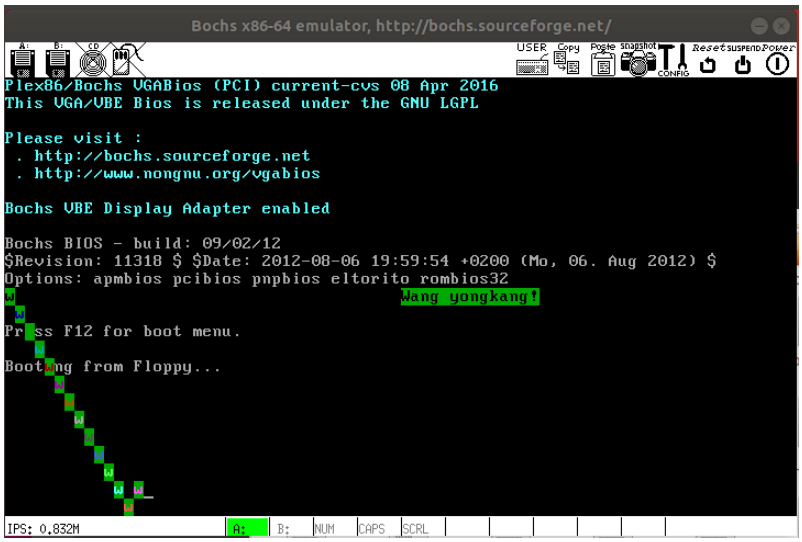
- 然后再终端利用Linux的dd命令，

```
dd if=test2_2.bin of=a.img seek=0 bs=512
count=1 conv=notrunc
```

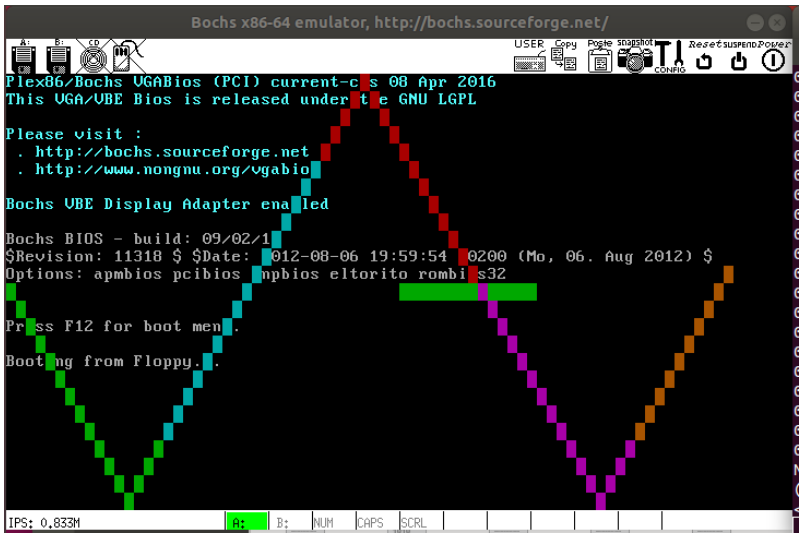
把test2_2.bin文件写入a.img软盘的首扇区；

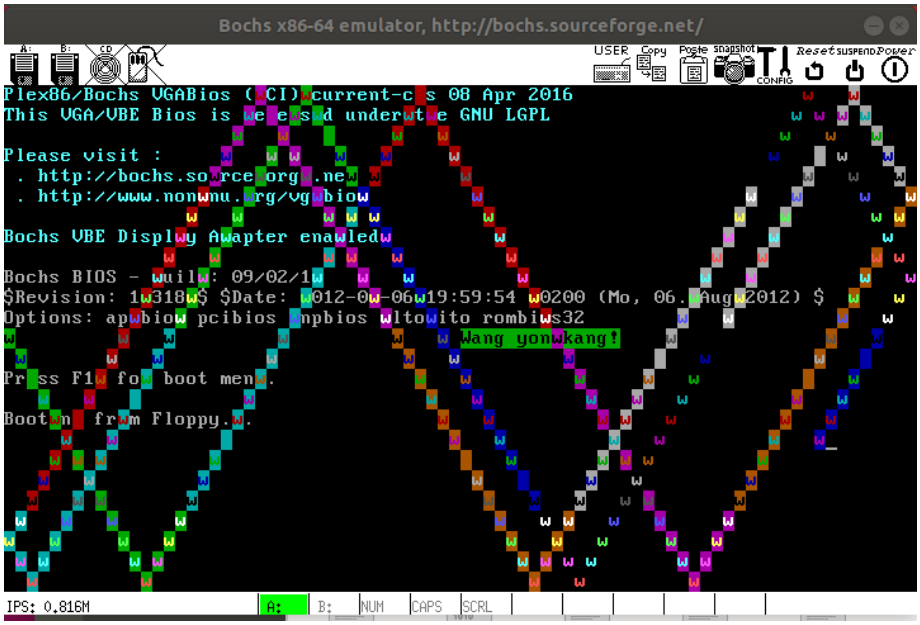
- 之后运行Bochs，终端输入Bochs

  输入c,运行虚拟机




后面的效果如图：






上述过程中，出现问题最多的是编译汇编源程序，因为不熟悉x86汇编语言，会有许多语法错误，不过编译器会提示bug的行号，可以解决。

其次是每次都需要运行虚拟机来debug，解决各种运行的错误，经过不断尝试、查阅资料最后成功使程序达到理想效果。

## 五、实验总结

经历这次实验，主要有以下几方面体会：

- **自主查阅资料，查阅文献能力非常重要**

  开始做此实验时，遇到最大挑战实际上是入门过程，因为之前对于操作系统的接触非常少，有非常多东西不了解，无从下手，在网上查阅各种资料和老师的课件后，找到了于渊的《一个操作系统的实现》的书籍，才渐渐明白这个实验的基本过程，后面实验过程中，环境的配置，遇到很多问题也是在网上查阅资料一一解决，所以自主查阅资料的能力还是非常重要的。

- **需要非常耐心去接触未知领域的知识，要认真仔细看**

  做实验时，由于急于求成，阅读资料过程中出现跳读现象，对于入门过程来讲，这是非常不可取的，我在做这个实验过程中就跳读了Bochs的配置文件修改一步，导致后面找错误浪费了许多时间；自己做实验过程中遇到的许多问题，比如在显示器上如何显示字符的原理，资料非常多，但其实只需要耐心、仔细阅读完一篇[3] 就足以解决问题了。

- **关于x86汇编语言和Linux下开发的感受**

  计算机组成原理没有讲x86体系，所以x86汇编也就没有学了，所以也建议下一届开课的时候计算机组成原理尽量选择x86体系内容来讲。

Linux下来做这个实验，可以说非常方便，虽然熟悉Linux需要一定时间，但是配置开发软件、环境的过程是非常简单、方便的。

总之，万事开头难，入门过了，也相信后面的挑战也是可以完成的！

## 参考文献


[1] 于渊.《一个操作系统的实现》[M].电子工业出版社，2009-6-1

[2] 王爽.《汇编语言(第3版)》[M].清华大学出版社，2013-9

[3] 凌应标. “01实验课.ppt”，中山大学计算机科学系，2015-3

[4] https://blog.csdn.net/longintchar/article/details/70183677，2019-3-14

[5] https://www.cnblogs.com/jiftle/p/8453106.html， 2019-3-14