操作系统实验报告

18 级计科超算 18340208 张洪宾

1 实验题目

接管裸机的控制权

2 实验目的

- 掌握 x86 汇编的基本方法
- 了解程序开机启动的过程并尝试用引导区引导
- 搭建和应用实验环境，熟悉各种工具的使用
- nasm 编程实践

3 实验要求

3.1 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。

3.2 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。

4 实验方案

4.1 硬件或虚拟机配置方法

- 操作系统：macOS Catalina 10.15.4
- 虚拟机：VirtualBox
- Linux 环境：Ubuntu 18.04（用于生成格式化的虚拟软盘）
- 软盘工具:Hex Fiend
- 终端：zsh
- 编译工具：NASM 2.14.02(On macOS)

4.2 搭建和应用实验环境方案

该部分较为简单。在 macOS 环境下不支持 winhex，所以我选择了 Hex Fiend 来替代它。

首先我们在 Linux 虚拟机下输入 /sbin/mkfs.msdos -C a.img 1440 来生成 a.img 这个虚拟软盘，然后用 Hex Fiend 工具对它进行修改，使得磁盘可以被我的信息填充。

4.3 接管裸机的控制权

老师给的汇编程序是采用 masm 语法，与我要采用的 nasm 不同。在读懂老师的程序的逻辑后，我决定重构老师的代码。并且在输出部分，我希望我的输出不是简单的字符‘A’，而是让我的信息在屏幕上不断跳动。为了避免整个屏幕都充满了我的个人信息，使屏幕变得十分杂乱，我在每次输出之前都进行清屏操作。所以最终我的目标是让我的个人信息看起来就像是在屏幕上不断跳动。

在网上查询资料得知：电脑在接电后，系统启动并自检完成后，会将软盘的第一个扇区中的内容读入内存地址 0000:7C00 中，并检查第一个扇区的结尾两个字节是否为 0x55 和 0xAA。所以我们首先得使用 org 07c00h，告诉汇编器将程序中的所有地址都加上 07c00h。

而数字弹跳的算法过程其实并不难，读懂了老师的代码，通过不断地循环及加减，将每次我的信息的首字符的位置坐标计算出来，再打印即可。

从细节上来说，输出字符并不是一个非常简单的事情。我在网上找到了两种方法，一种是采用中断调用的方法，直接调用 int 10h 的写功能将字符串输出，还有一种是将字符串放入显存中，它就会自动输出到屏幕。

在这里我选择了第二种方法，将字符串放进显存。在内存地址结构中，B8000H BFFFFH 共 32KB 的空间，为 80x25 彩色字符模式的显示缓冲区。向这个地址空间写入数据，写入的内容将立即出现在显示器上。

而在显存中，一个 16 位的字符可以有两个部分，一个是字符的属性，包括字符的颜色和背景色，还有一个是字符的 ASCII 码。为了使输出看起来更好看，我将要放入显存的字符串根据字符串的 x 坐标设置字符的颜色，而字符串的背景色统一设置为黑色。于是输出的字符串会随着位置的不同而显示出不同的颜色。

打印字符串的代码如下：

```
Print:;用于打印当前位置的字符串
    xor ax,ax
    mov word ax,[x]
    mov bx,WIDTH
    mul bx
    add word ax,[y]
    mov bx,2
    mul bx
    mov bx,ax
    mov dx,bx;将地址保存在dx，用于清除时使用
    mov si, message
    mov cx, LENGTH
    printStr:
      mov byte al,[si]
      and ah,15
      mov [gs:bx],ax;将字符放到显示内存段
      inc si
      inc bx
      inc bx
    loop printStr
    jmp Loop
```

而如果字符串过多，堆满了屏幕，就显得屏幕杂乱无章，所以我在每一次输出之后都会清屏。清屏的方式跟输入是一样的，我在显示内存段的响应位置放入空格字符串，达到清屏的效果。清屏操作的代码如下：

```
Print_empty:;用于清空上次打印的内容
    mov ax, 0B800h
    mov si, 160
    mov cx, 80*25
    mov dx, 0
    clsLoop:
      mov [gs:si], dx
      add si, 2
    loop clsLoop
    jmp Print

    count dd 1
    x dw 2
```

```
14      y dw 0
```

4.4 编译环境的配置

在 macOS 的终端 zsh 下使用 brew install nasm 命令安装 nasm 即可。

5 实验过程

5.1 搭建和应用实验环境

5.1.1 虚拟机的配置

- 首先在 Mac 上下载安装 VirtualBox。
- 然后打开 VirtualBox 主界面，点击 [新建]，输入名称 ZhbPC，类型选择 Other，版本选择 Other/Unknown，如图 1 所示。

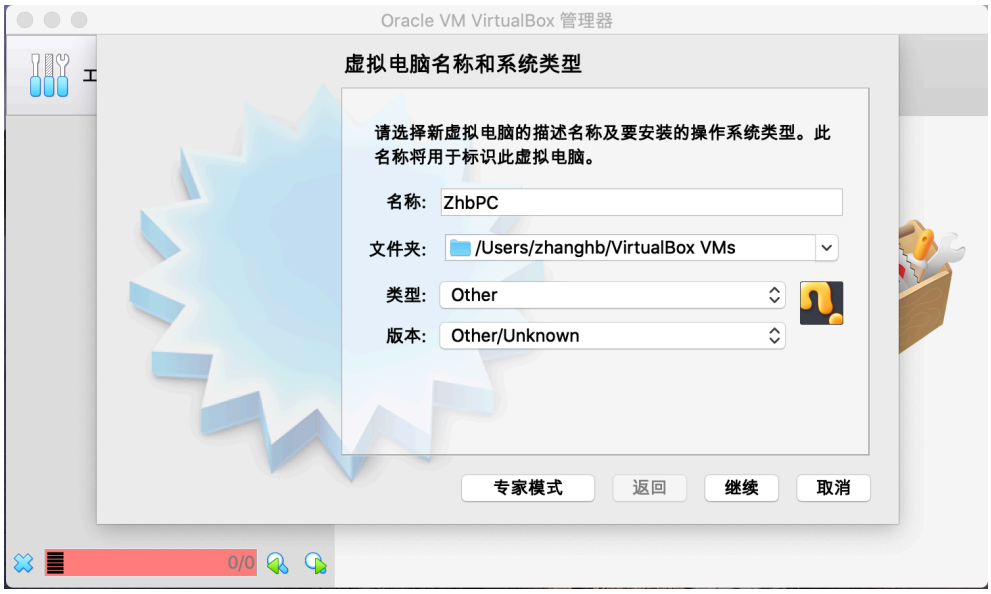


图 1: 新建虚拟机

- 分配内存大小 4MB，如图 2 所示：

图 2: 分配内存大小

- 选择“不添加虚拟硬盘”，如图 3 所示：

图 3: 不添加虚拟硬盘

5.1.2 装载由 Linux 格式化得到的软盘

在我的 Linux 虚拟机下执行/sbin/mkfs.msdos -C a.img 1440 得到 a.img，将它共享到 macOS 中。

打开虚拟机，选择 ZhbPC，再选择 [设置]-[存储]，

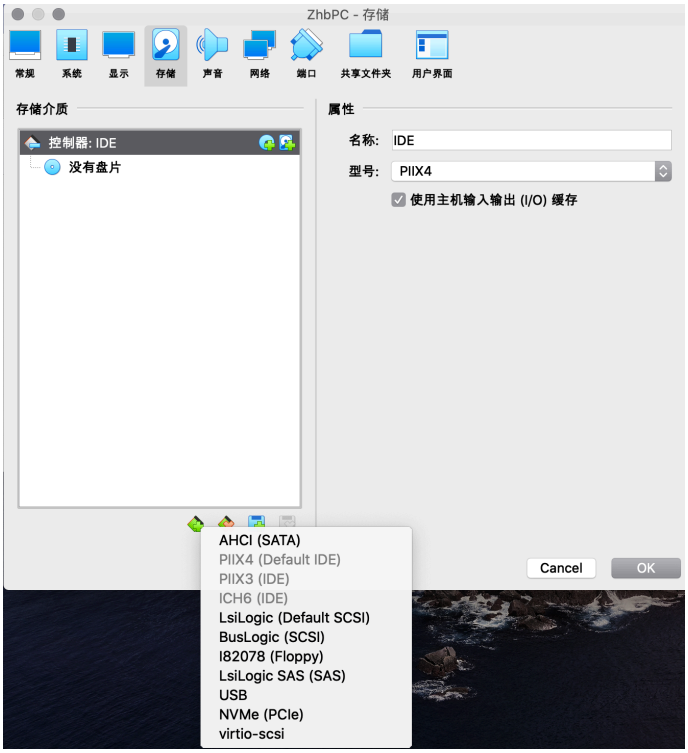


图 4: [设置]-[存储]

选择 Floppy，然后点击控制器：Floppy 右方的“+”号，然后选择注册，找到 a.img，然后选择确定，保存设置

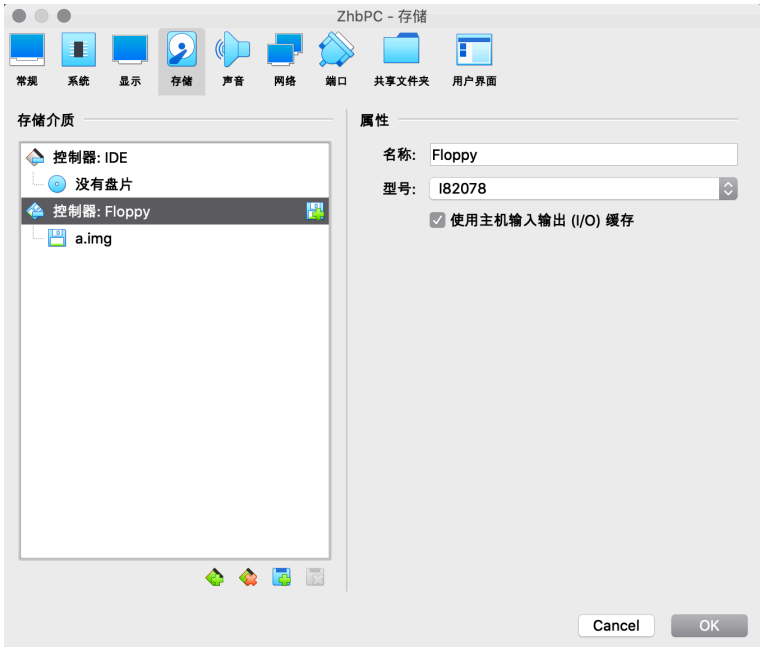


图 5: 设置软盘

启动虚拟机：

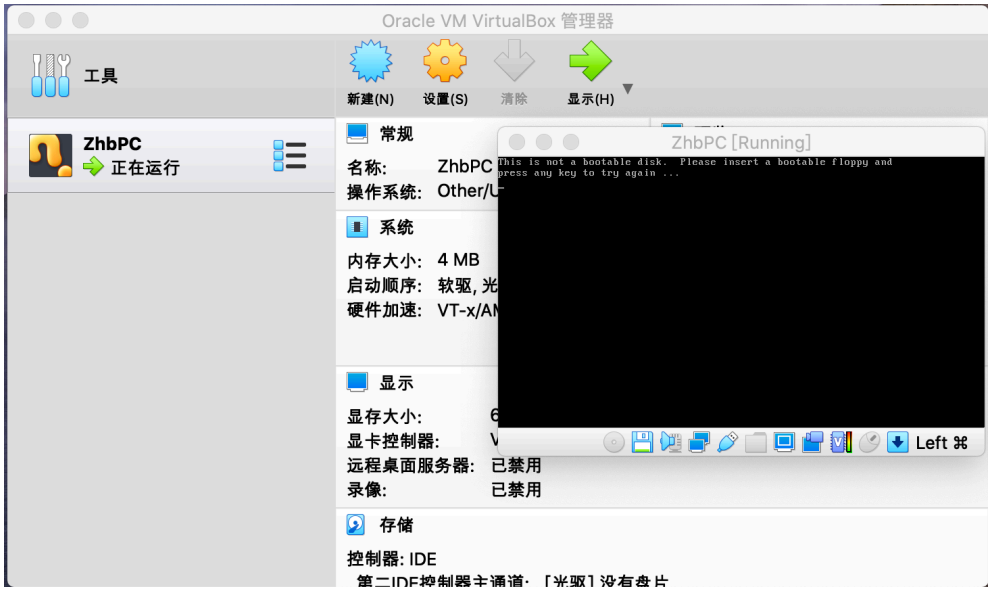


图 6: 启动虚拟机

显示这不是一个可加载的磁盘（意料之中）。

5.1.3 用软盘工具编辑软盘

用 Hex Fiend 打开 a.img，可以看到软盘的二进制信息，如图 7:

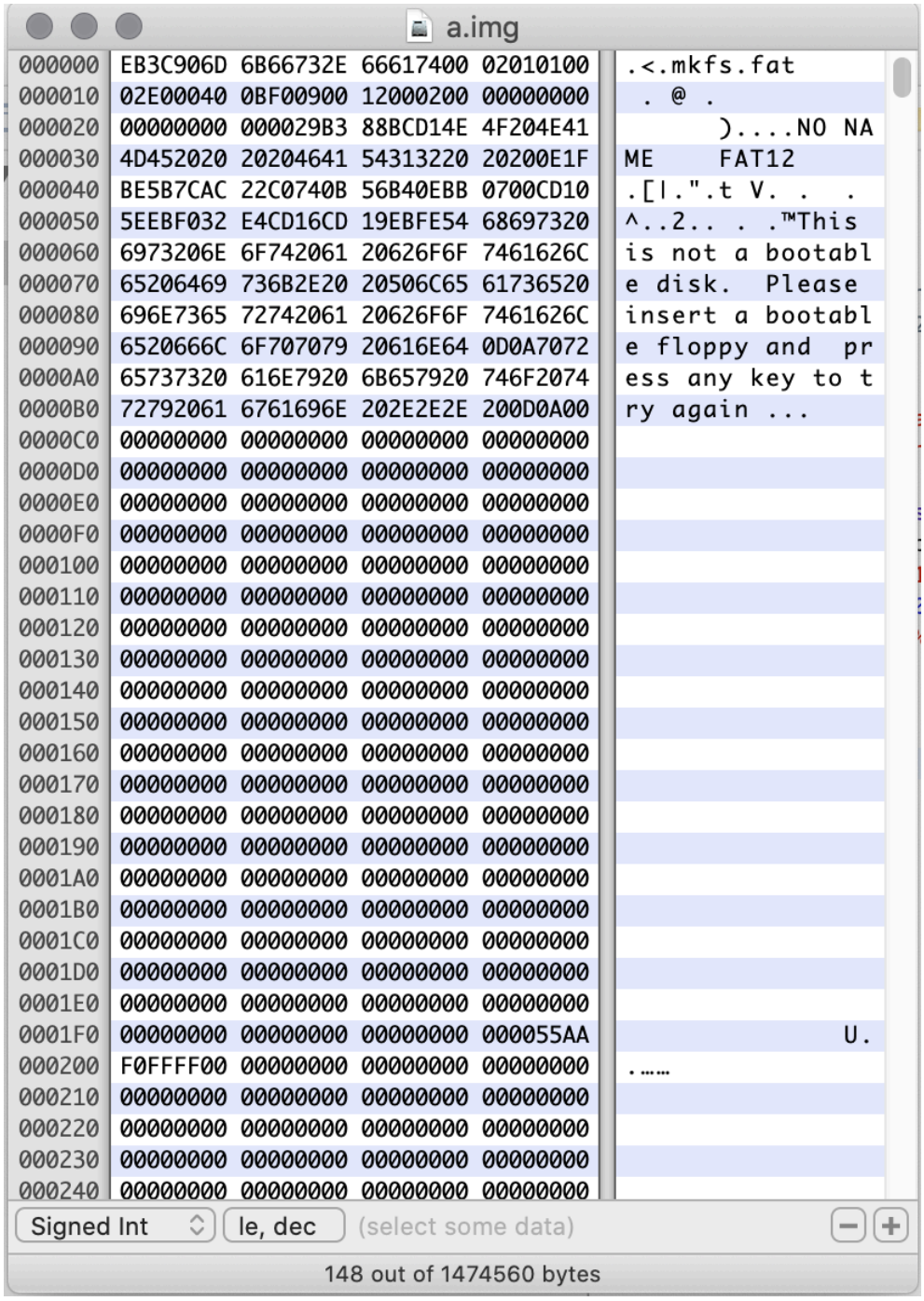


图 7: 用 Hex Fiend 打开 a.img

用 Hex Fiend 修改该扇区使得该扇区的其他部分的内容为我的个人信息:

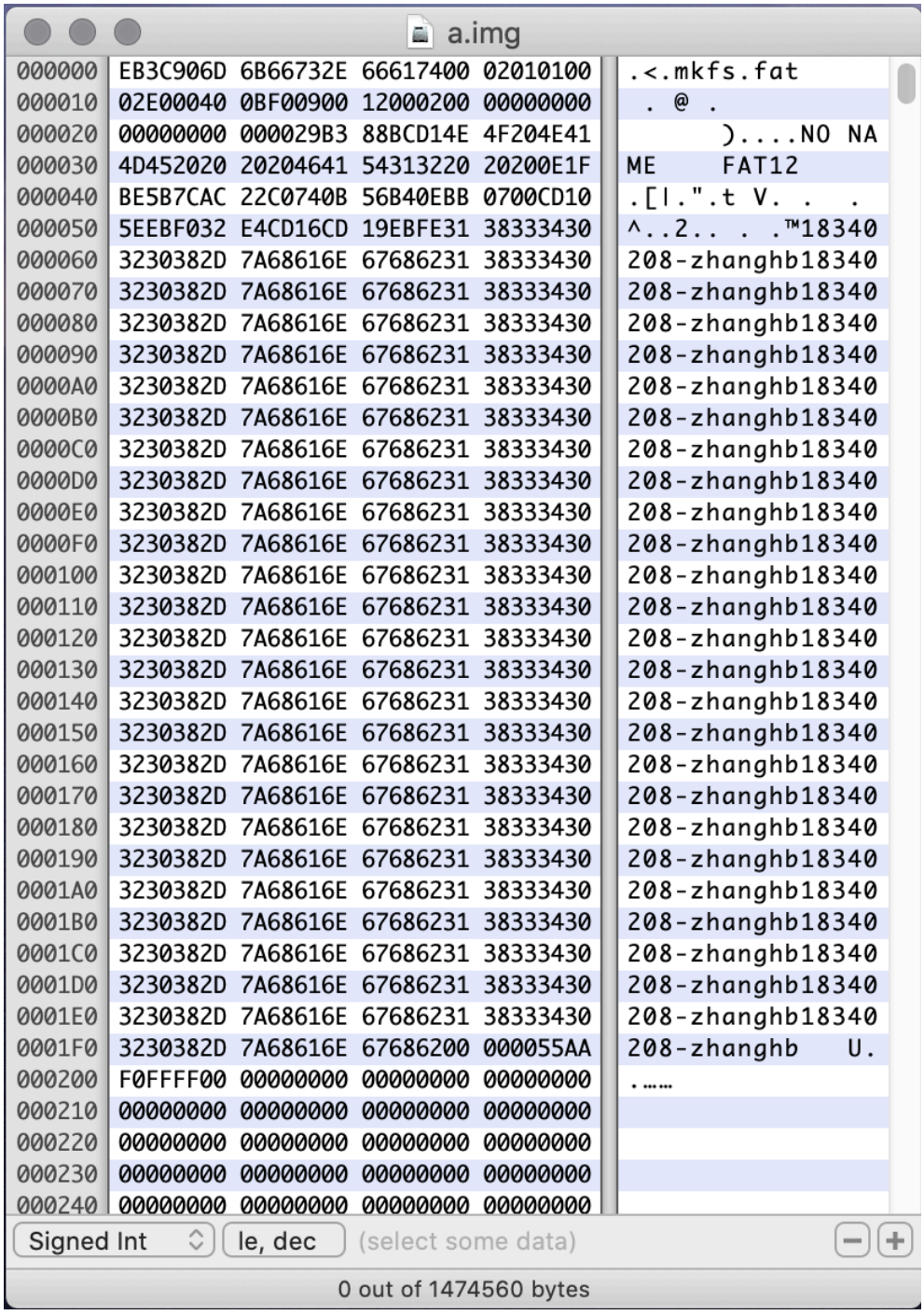


图 8: 用 Hex Fiend 打开修改后的 a.img

用 VirtualBox 虚拟机装载该软盘并运行：

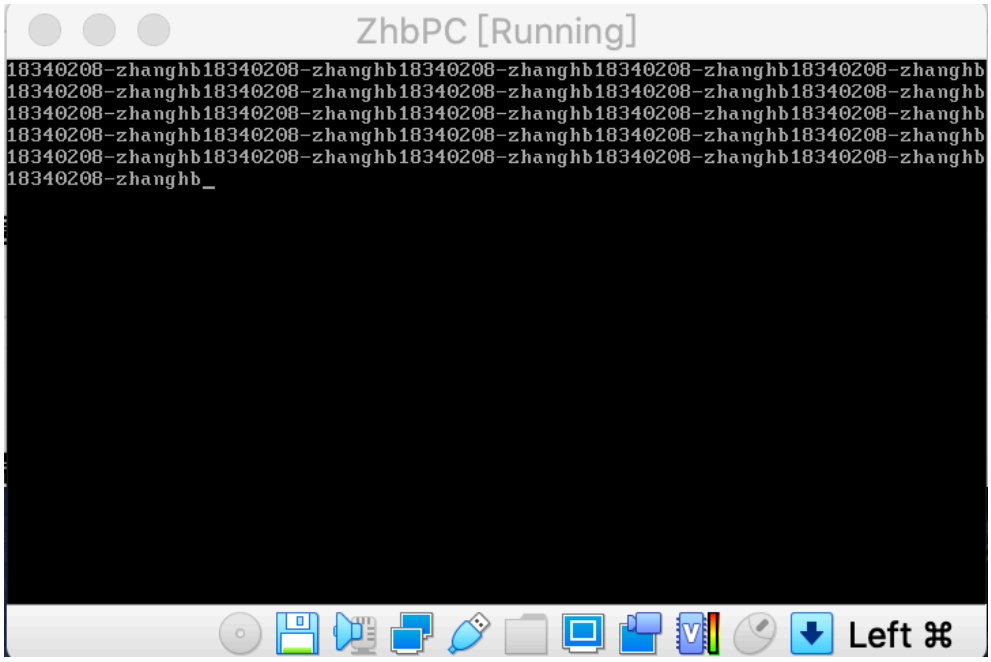


图 9: 用虚拟机运行修改后的 a.img

可以看到我的信息出现在屏幕上。

5.2 接管裸机的控制权

虽然这次的实验很简单，但是考虑到以后可能会出现更加复杂的情况，我决定使用 Makefile 来简化编译过程。

于是我写了一个简单的 Makefile：

```
IMG = zhb.img
BIN = zhb.bin

all:    pro img clean

img:$(BIN)
ifeq ($(IMG), $(wildcard $(IMG)))
    rm $(IMG)
endif
    mkfile -n 2880b $(IMG)
    dd if=$(BIN) of=$(IMG) conv=notrunc

pro:$(BIN)
$(BIN):zhb.asm
    nasm  $< -o $@

.PHONY:clean
clean:
    rm -rf *o *bin

```

图 10: Makefile

打开 macOS 的终端，进入文件所在目录，输入 make：

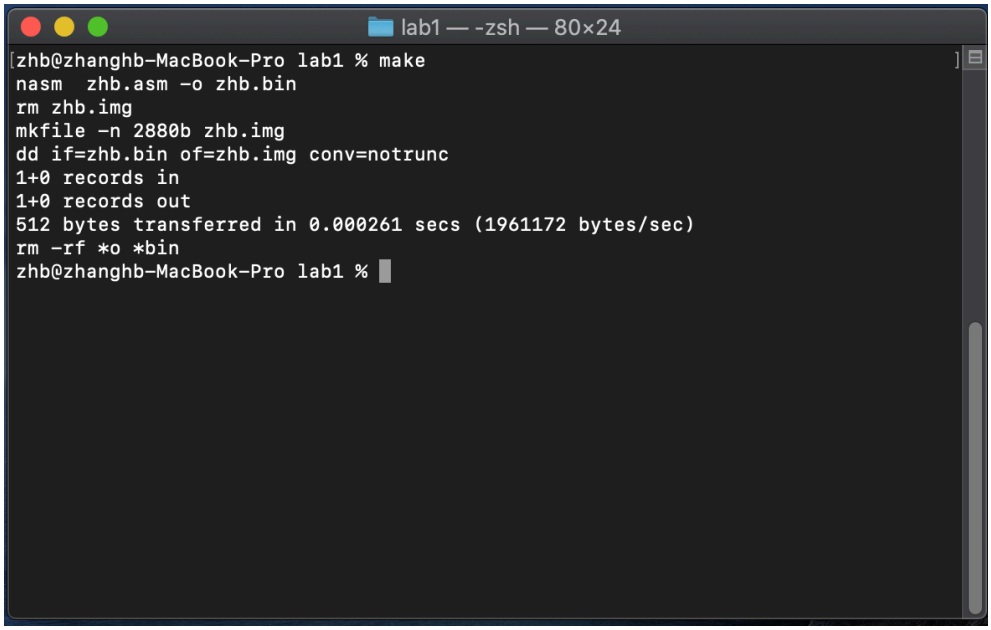
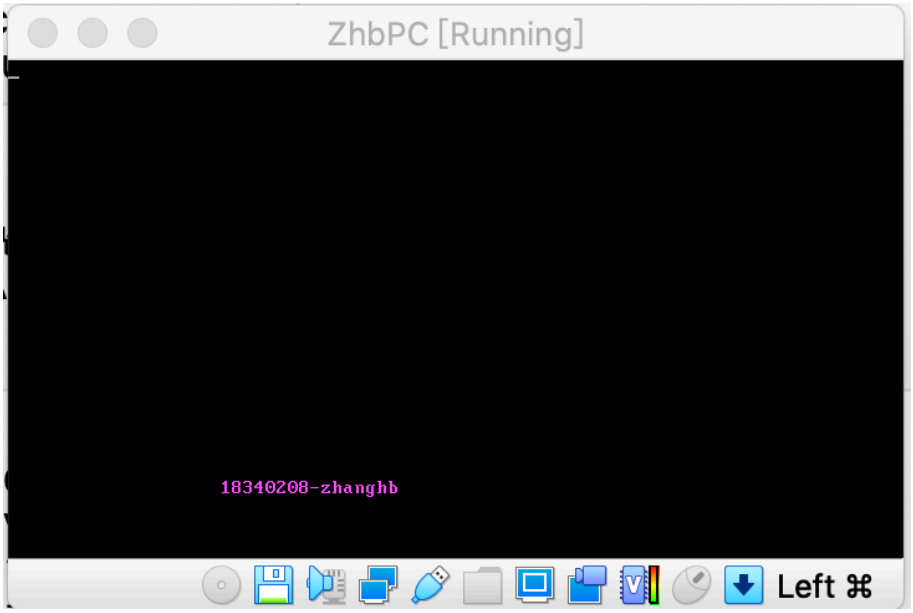


图 11: 执行 make

得到了 zhb.img，在虚拟机上装载运行：（因为一个时刻只能有一个信息在屏幕上，所以我随机选择了几个时刻截图）



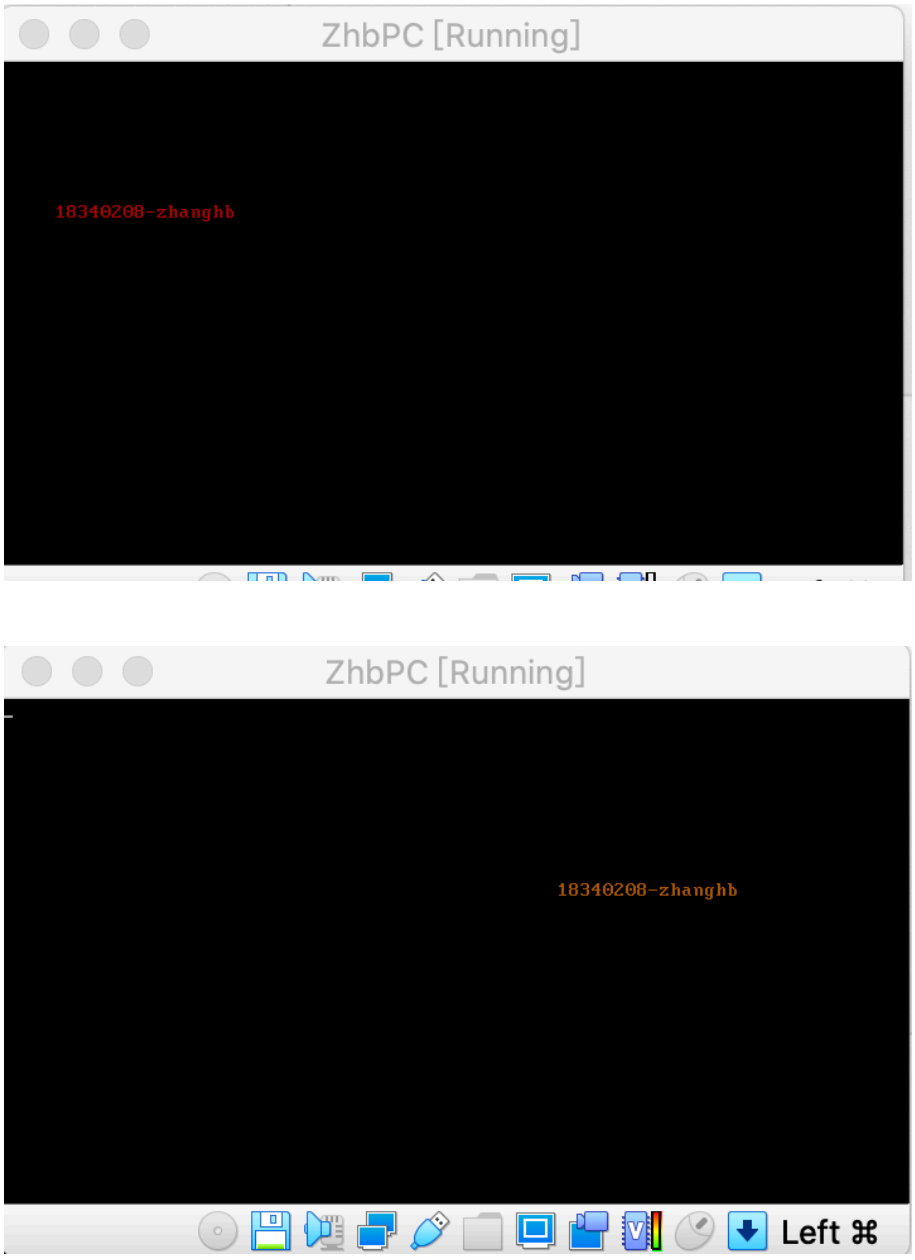


图 12: 几个不同的时间点虚拟机的运行状态

可以看出，在不同的位置，我的个人信息呈现出不同的颜色。而为了更直观的说明虚拟机运行状态，我截了一段虚拟机的运行状态的视频放在文件夹中。

6 实验总结

6.1 遇到的问题

- 3.5 寸软盘的大小是 1.44MB，我主观认为它是 1.44 * 1024 * 1024B，这样子算出来不是整数，而事实上它应该是 2880 * 512B = 2880 * 0.5KB = 1440KB = 1.44MB 容量的虚拟软盘。
- macOS 似乎没有提供类似于 Linux 的/sbin/mkfs.msdos 命令，所以我无法直接通过系统生成指定大小的格式化软盘。于是我在实验的第二部分在 Makefile 中采用了 mkfile 命令，生成指定大小的空白文件，然后再用 dd 命令将生成的文件写入指定扇区。
- 为了让首扇区以 0x55，0xAA 结尾，我在汇编文件尾部加上了两条伪指令

  times 510-($-$$) db 0 ; 重复 n 次每次填充值为 0

  dw 55aah

  这样子就保证了第一个扇区是以 0x55，0xAA 结束。

6.2 心得体会

第一次操作系统实验，对于没有接触过 x86 汇编的我是一个比较大的挑战。需要学习的东西很多，而实验的要求很高。虽然在计算机组成原理里面稍微接触过 x86，但是自己动起手来发现 bug 一个接着一个。

不过操作系统实验还是非常有趣的。虽然花了很多时间，但是当我看到在虚拟机上正确运行的结果的时候，顿时觉得成就感满满。在这个过程中觉得还是学到了很多东西，特别是如何去使用 nasm 汇编工具，对整个计算机的理解也得到了很大的提升。

为了提早适应做大的项目，并适应 macOS/Linux 下的编译，我尝试学习 Makefile，也发现了这是个非常好用的工具。相比以前编译逐条命令输入，使用 Makefile 可以提高我对结果的认知，并更好地理解我要输入的命令。

总而言之，该实验非常有趣，让我学到了很多有意思的东西，我对接下来对操作系统实验充满了期待。