操作系统实验一

裸机控制权与引导程序

实验目的：

1、了解原型操作系统设计实验教学方法与要求
2、了解计算机硬件系统开机引导方法与过程
3、掌握操作系统的引导程序设计方法与开发工具
4、复习加强汇编语言程序设计能力

实验要求：

1、 知道原型操作系统设计实验的两条线路和前 6 个实验项目的差别
2、 掌握 PC 电脑利用 1.44MB 软驱的开机引导方法与过程的步骤
3、 在自己的电脑上安装配置引导程序设计的开发工具与环境
4、 参考样版汇编程序，完成在 PC 虚拟机上设计一个 1.44MB 软驱的引导程序的完整工作。
5、 编写实验报告，描述实验工作的过程和必要的细节，以证实实验工作的真实性

实验内容：

(1)在自己的电脑上安装一种虚拟机软件，在实验报告中记录主要的安装步骤和截屏。
(2)利用虚拟机软件，生成有 1.44MB 软驱的一个 PC 虚拟机，列出 PC 虚拟机的配置，并生成有 1.44MB 软盘映像文件 3 个。
(3)安装 winHex 等可视化编辑十六进制文件内容的工具，对第一个软盘映像文件的首扇区填满个人学号姓名拼音。
(4)安装一种 x86 汇编程序和一种编辑汇编/C 源程序代码的工具或集成环境。
(5)程序用 x86 汇编语言编写，参考字符反弹运动示范程，修改或重写程序，直接对文本方式的显存进行操作，以某种运动轨迹或几何图像在屏幕一个区域显示字符或字符串，还可以有各种个性化变化效果，能看到个人学号或姓名拼音。
(6)程序汇编后满足引导扇区程序的要求，利用工具将其制作写入 1.44mb 软盘映像的引导扇区中，保证在虚拟机中能引导执行，观察到效果。
(7)建立自己的软件项目管理目录，管理实验项目相关文档

实验环境：

- Windows 10-64bit
- Vmware WorkStation 15 pro 15.5.1 build-15018445：虚拟机软件
- NASM version 2.13.02：汇编程序的编译器，在 linux 下通过 sudo apt-get install nasm 下载
- Oracle VM VirtualBox：一款开源的虚拟机软件
- Ubuntu-18.04.4:安装在 Vmware 的虚拟机上
- WinHex：二进制文件编辑器

实验过程：

一、安装虚拟机：

我首先使用搜索引擎下载了两种支撑虚拟机的软件：VMware 和 VirtualBox。并且在 Vmware 里面配置好了 linux 环境方便实验。

二、创建裸机：

在这里只阐述 Vmware 下的裸机创建：

点击创建虚拟机→一直点击下一步直到完成即可。其中需要设置暂不安装操作系统、以及选择客户端操作系统和版本，如图所示。

三、创建并格式化虚拟软盘

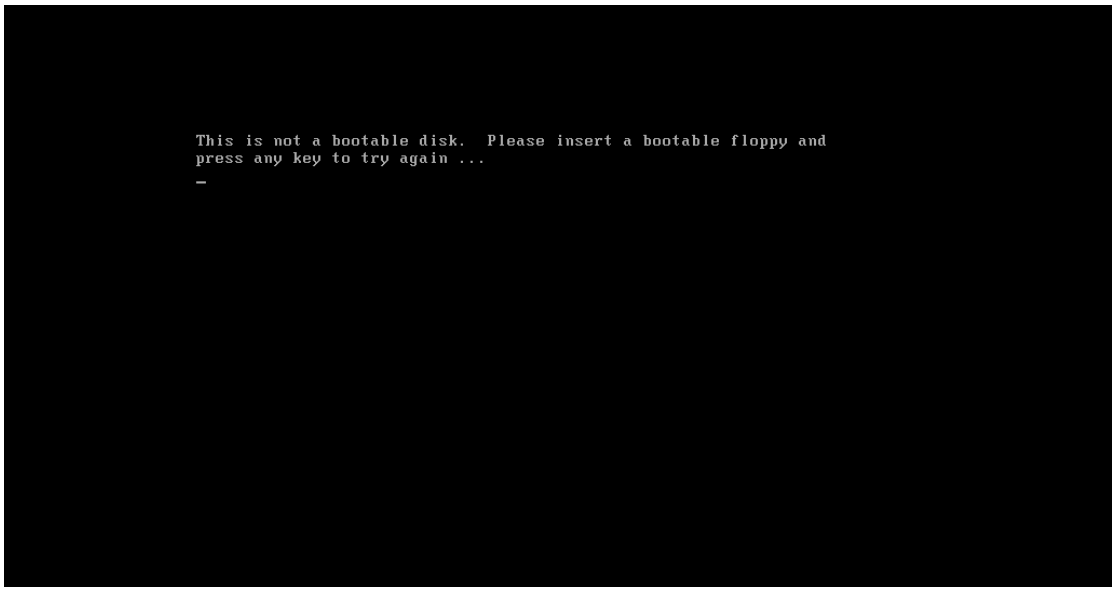
在 linux 环境下，我们可以通过使用以下命令来创建并格式化软盘：

```
/sbin/mkfs.msdos -C Condor.img 1440
```

同时，如果终端返回如下反馈，那么我们的软盘便创建并格式化成功：

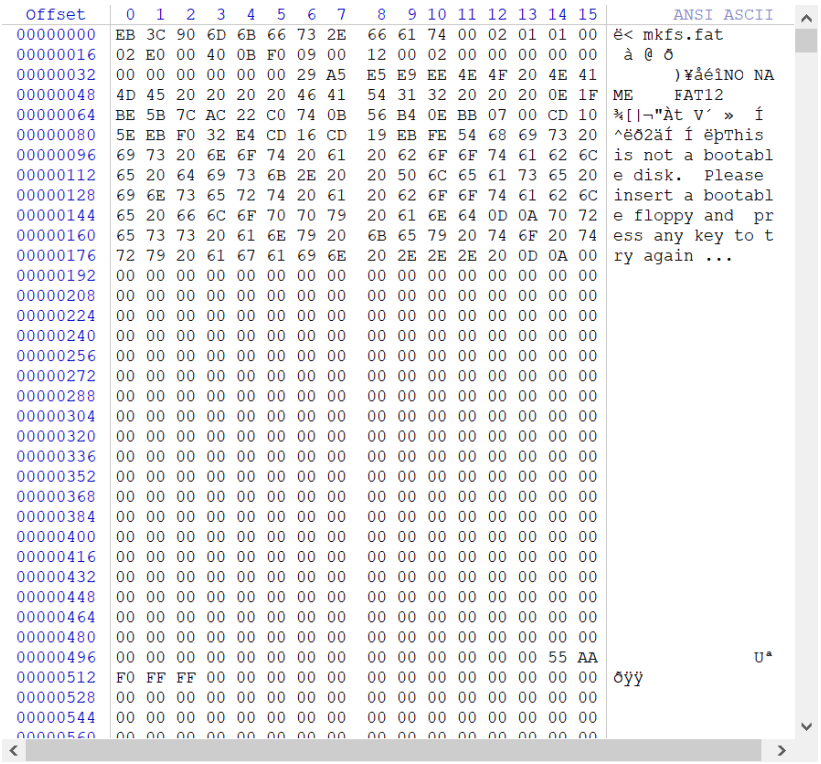
```
mkfs.fat 4.1 (2017-01-24)
```

我们将该软盘添加到我们的裸机的虚拟软驱里面，打开裸机可以得到以下显示：



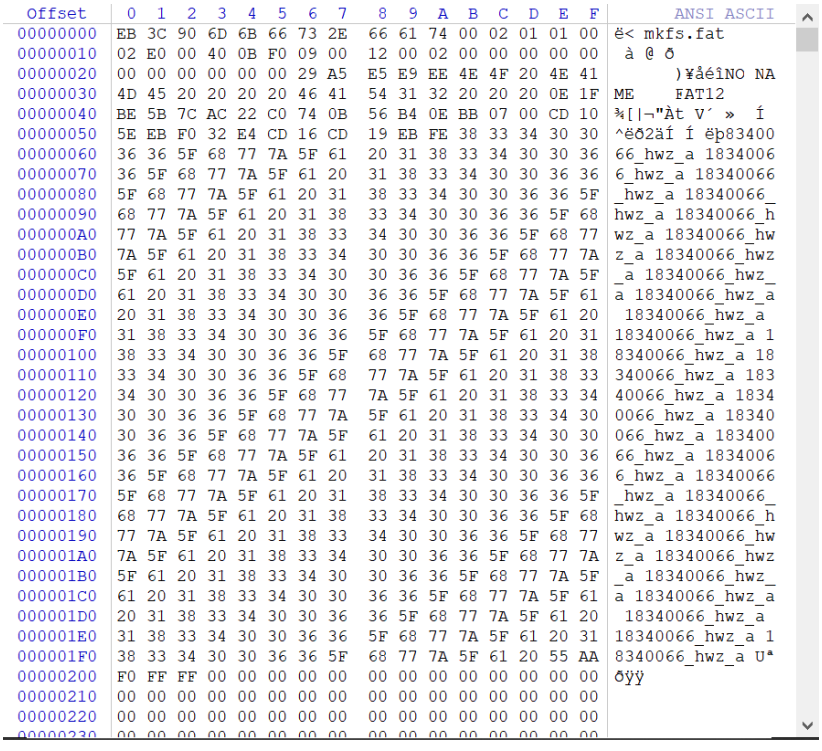
四、在扇区中填满个人信息

使用 WinHex 打开我们刚刚创建的软盘，可以得到其信息如下：



Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	ANSI ASCII
00000000	EB	3C	90	6D	6B	66	73	2E	66	61	74	00	02	01	01	00	ë< mkfs.fat
00000016	02	E0	00	40	0B	F0	09	00	12	00	02	00	00	00	00	00	à @ ð
00000032	00	00	00	00	00	00	29	A5	E5	E9	EE	4E	4F	20	4E	41	)¥åéîNO NA
00000048	4D	45	20	20	20	20	46	41	54	31	32	20	20	20	0E	1F	ME FAT12
00000064	BE	5B	7C	AC	22	C0	74	0B	56	B4	0E	BB	07	00	CD	10	¾[|¬"Àt V´ » Í
00000080	5E	EB	F0	32	E4	CD	16	CD	19	EB	FE	54	68	69	73	20	^ëð2äÍ Í ëþThis
00000096	69	73	20	6E	6F	74	20	61	20	62	6F	6F	74	61	62	6C	is not a bootabl
00000112	65	20	64	69	73	6B	2E	20	20	50	6C	65	61	73	65	20	e disk. Please
00000128	69	6E	73	65	72	74	20	61	20	62	6F	6F	74	61	62	6C	insert a bootabl
00000144	65	20	66	6C	6F	70	70	79	20	61	6E	64	0D	0A	70	72	e floppy and pr
00000160	65	73	73	20	61	6E	79	20	6B	65	79	20	74	6F	20	74	ess any key to t
00000176	72	79	20	61	67	61	69	6E	20	2E	2E	2E	20	0D	0A	00	ry again ...
00000192	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000208	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000224	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000240	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000256	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000272	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000288	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000304	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000320	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000336	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000352	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000368	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000384	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000400	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000416	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000432	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000448	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000464	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000480	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000496	00	00	00	00	00	00	00	00	00	00	00	00	00	00	55	AA	U²
00000512	F0	FF	FF	00	00	00	00	00	00	00	00	00	00	00	00	00	ðÿÿ
00000528	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000544	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

接下来，我们对其进行个人信息的填充操作，结果如下：



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI ASCII
00000000	EB	3C	90	6D	6B	66	73	2E	66	61	74	00	02	01	01	00	ë< mkfs.fat
00000010	02	E0	00	40	0B	F0	09	00	12	00	02	00	00	00	00	00	à @ ð
00000020	00	00	00	00	00	00	29	A5	E5	E9	EE	4E	4F	20	4E	41	)¥åéîNO NA
00000030	4D	45	20	20	20	20	46	41	54	31	32	20	20	20	0E	1F	ME FAT12
00000040	BE	5B	7C	AC	22	C0	74	0B	56	B4	0E	BB	07	00	CD	10	¾[|¬"Àt V´ » Í
00000050	5E	EB	F0	32	E4	CD	16	CD	19	EB	FE	38	33	34	30	30	^ëð2äÍ Í ëþ83400
00000060	36	36	5F	68	77	7A	5F	61	20	31	38	33	34	30	30	36	66_hwz_a 1834006
00000070	36	5F	68	77	7A	5F	61	20	31	38	33	34	30	30	36	36	6_hwz_a 18340066
00000080	5F	68	77	7A	5F	61	20	31	38	33	34	30	30	36	36	5F	_hwz_a 18340066_
00000090	68	77	7A	5F	61	20	31	38	33	34	30	30	36	36	5F	68	hwz_a 18340066_h
000000A0	77	7A	5F	61	20	31	38	33	34	30	30	36	36	5F	68	77	wz_a 18340066_hw
000000B0	7A	5F	61	20	31	38	33	34	30	30	36	36	5F	68	77	7A	z_a 18340066_hwz
000000C0	5F	61	20	31	38	33	34	30	30	36	36	5F	68	77	7A	5F	_a 18340066_hwz_
000000D0	61	20	31	38	33	34	30	30	36	36	5F	68	77	7A	5F	61	a 18340066_hwz_a
000000E0	20	31	38	33	34	30	30	36	36	5F	68	77	7A	5F	61	20	18340066_hwz_a
000000F0	31	38	33	34	30	30	36	36	5F	68	77	7A	5F	61	20	31	18340066_hwz_a 1
00000100	38	33	34	30	30	36	36	5F	68	77	7A	5F	61	20	31	38	8340066_hwz_a 18
00000110	33	34	30	30	36	36	5F	68	77	7A	5F	61	20	31	38	33	340066_hwz_a 183
00000120	34	30	30	36	36	5F	68	77	7A	5F	61	20	31	38	33	34	40066_hwz_a 1834
00000130	30	30	36	36	5F	68	77	7A	5F	61	20	31	38	33	34	30	0066_hwz_a 18340
00000140	30	36	36	5F	68	77	7A	5F	61	20	31	38	33	34	30	30	066_hwz_a 183400
00000150	36	36	5F	68	77	7A	5F	61	20	31	38	33	34	30	30	36	66_hwz_a 1834006
00000160	36	5F	68	77	7A	5F	61	20	31	38	33	34	30	30	36	36	6_hwz_a 18340066
00000170	5F	68	77	7A	5F	61	20	31	38	33	34	30	30	36	36	5F	_hwz_a 18340066_
00000180	68	77	7A	5F	61	20	31	38	33	34	30	30	36	36	5F	68	hwz_a 18340066_h
00000190	77	7A	5F	61	20	31	38	33	34	30	30	36	36	5F	68	77	wz_a 18340066_hw
000001A0	7A	5F	61	20	31	38	33	34	30	30	36	36	5F	68	77	7A	z_a 18340066_hwz
000001B0	5F	61	20	31	38	33	34	30	30	36	36	5F	68	77	7A	5F	_a 18340066_hwz_
000001C0	61	20	31	38	33	34	30	30	36	36	5F	68	77	7A	5F	61	a 18340066_hwz_a
000001D0	20	31	38	33	34	30	30	36	36	5F	68	77	7A	5F	61	20	18340066_hwz_a
000001E0	31	38	33	34	30	30	36	36	5F	68	77	7A	5F	61	20	31	18340066_hwz_a 1
000001F0	38	33	34	30	30	36	36	5F	68	77	7A	5F	61	20	55	AA	8340066_hwz_a U²
00000200	F0	FF	FF	00	00	00	00	00	00	00	00	00	00	00	00	00	ðÿÿ
00000210	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000220	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

由于是使用 for 循环对文件进行读写，因此得到的填充看上去不够整齐……(主要代码如下)

```
    char s[] = "18340066_hwz_a ",b[1440 << 10];
for (int j = 91; j < 510; j++)
    b[j] = s[j % (sizeof(s) - 1)];
```

可知填充的个人信息为 18340066_hwz_a。将该文件导出，得到第二个虚拟软盘

Condor2.img，将其设置为裸机的虚拟软驱，打开裸机可以看到个人信息在屏幕上得以显示。

五、字符反弹运动

我先把老师给的代码（stoneN.asm）通过 nasm 编辑器将其转换为二进制（bin）文件，然后修改其后缀为 img，并导入裸机中，但是在打开裸机后发现裸机毫无反应。经过同学们在群里面的讨论，我得知老师所给的样例存在一些错误。但是又由于我在 linux 下对其编译没有报错，这导致我修改的难度很大。（如图没有报错）

```
condor@condor-virtual-machine:~$ nasm stoneN.asm -o stoneN.bin
condor@condor-virtual-machine:~$
```

因此我决定结合老师的样例，并且在相关资料的帮助下自己完成类似的操作，具体技术细节如下：

首先我在老师给的参考代码里发现了一段注释：

```
; 程序源代码（stone.asm）
;本程序在文本方式显示器上从左边射出一个*号,以 45 度向右下运动，撞到边框后反射,如此类推.
;  凌应标 2014/3
;   MASM 汇编格式
```

第五行中的 MASM 汇编格式，结合 PPT 中介绍的 nasm 工具，我一开始认为这可能是老师的注释出了一点小差错，但通过搜索引擎我了解到，这两者是有所区别，我想这可能就是导致裸机毫无反应的原因。同时，我解决了本来应该报错的信息没有显示的问题。其报错的信息如下：

```
condor@condor-virtual-machine:~$ nasm stone.asm -o stone.com
stone.asm:11: warning: label alone on a line without a colon might be in error [
-w+orphan-labels]
stone.asm:11: error: attempt to define a local label before any non-local labels
stone.asm:13: error: parser: instruction expected
stone.asm:14: warning: macro `SEGMENT' exists, but not taking 0 parameters [-w+m
acro-params]
stone.asm:14: error: parser: instruction expected
stone.asm:162: error: symbol `code' redefined
stone.asm:162: error: parser: instruction expected
stone.asm:163: error: parser: instruction expected
```

随后我想着看下能否通过 masm 进行程序的编写，但发现 masm 的功能较于 nasm 来说相当薄弱：不报错、不支持 64 位、不自带反汇编工具等等，遂放弃该念头。

因此，我逐步对老师的代码进行了修改：

①将原来的字符串‘A’改成学号_姓名_aa（aa 是为了和第二个软盘的信息形成对比，这样既包含了个人信息，又包含了字符 a）；

②去掉了一些不必要的代码：例如老师用了 delay 和 ddelay 进行延迟显示，我将

其改成了统一的常数时间延迟显示；

③增加一些界面优化：将原代码中的

```
mov ah,0Fh          ;  0000：黑底、1111：亮白字（默认值为07h）
```

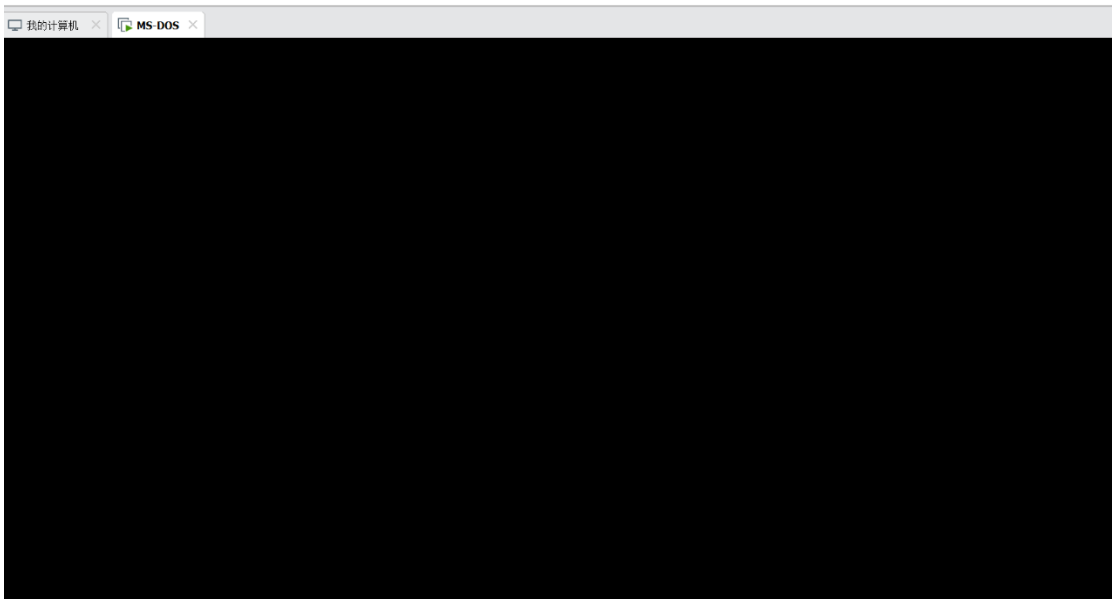
给去掉即可，这样打印出来的字符串便可以随着移动而变换颜色了。

④剩下的都是一些语句的语法和逻辑修改了（例如打印字符串的语句等），在此便不在详细赘述。

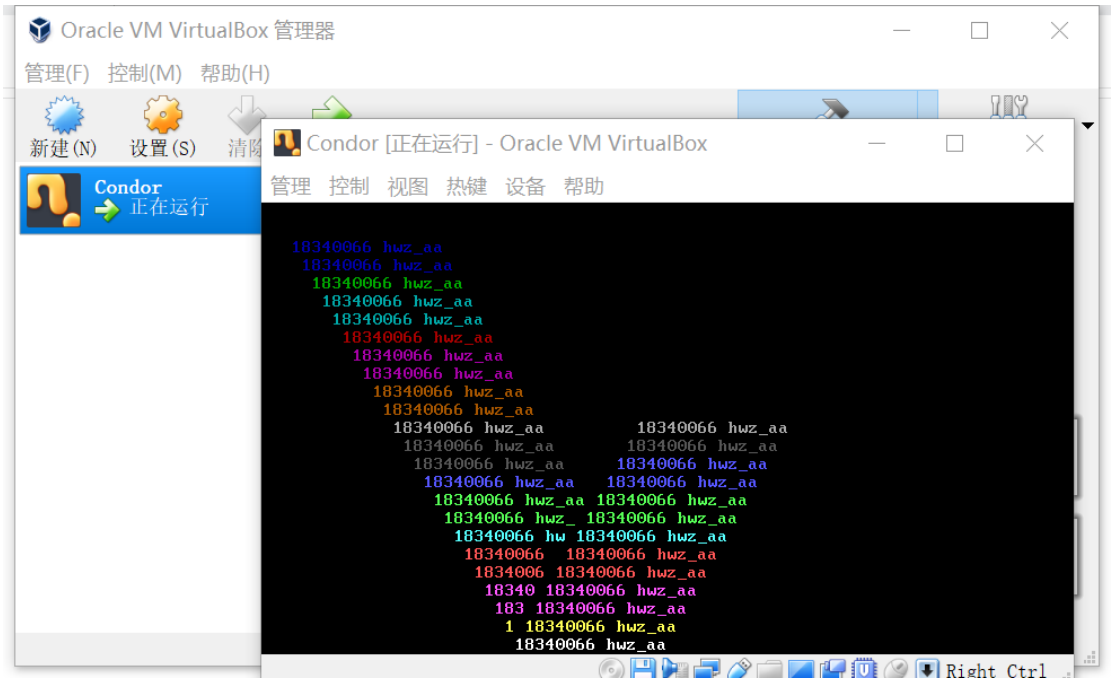
最终我得到了想要的软驱 Condor3.img，操作如下：

```
condor@condor-virtual-machine:~$ nasm Condor3.asm -o Condor3.com
condor@condor-virtual-machine:~$ /sbin/mkfs.msdos -C Condor3.img 1440
mkfs.fat 4.1 (2017-01-24)
condor@condor-virtual-machine:~$ dd if=Condor3.com of=Condor3.img bs=1440k conv=notrunc
记录了0+1 的读入
记录了0+1 的写出
290 bytes copied, 0.00181652 s, 160 kB/s
condor@condor-virtual-machine:~$
```

将得到的软驱导入到 Vmware 的裸机中，开启裸机，却没有任何反应。



但是当我使用 VirtualBox 进行相同的操作时，实验结果正是如我想要的：



六、实验心得

1、 在进行本次实验之前，我任何准备工作都没有做……这导致我在配环境等准备工作上花费了大量的时间，非常感谢群里的同学的指引，否则我可能还得踩多好几个坑。

2、 查阅相关的资料真的很重要，在做本次实验中，我打开最多的网站也许就是 csdn 了，实验过程中的大多数问题都能在上面找到答案。不过这在另一方面也表现出我在汇编语言方面的不足（在做实验之前没有接触，上学期的计算机组成原理学的是 MIPS 而不是 x86），可以说在做这个实验的时候我是查一次资料写一行代码了。不过我相信随着时间的延续我可以在这方面有所提升。

3、 最后的疑问是不知道为什么同样的操作，我在 Vmware 上和 VirtualBox 上得到截然不同的结果了，至今我仍然未能解决，可能是 Vmware 的缺陷？