操作系统原理实验

实验一

接管裸机的控制权

姓名： 吴侃

学号： 14348134

班别： 2014 级计算机系一班

日期： 2016.02.29 – 2016.03.02

目录

一、实验目的

学习如何搭建和应用实验环境,编写一个简单的汇编程序,让它接管裸机的控制权。

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，并用此软盘引导你的

XXXPC，直到成功。

三、实验方案

1. 虚拟机配置方法

由于使用虚拟机做操作系统实验有方便快速安全的特点,因此使用 VMWare Workstation 构建一个虚拟机平台. 该虚拟机平台的配置为： 单核单线程 CPU, 4MB 内存, 32MB 硬盘。

2. NASM 汇编工具及编辑器 vim 的设置：

NASM 是一个汇编工具, 能将汇编代码编译成对应二进制代码。使用命令 nasm xxx.asm 即可对 xxx.asm 汇编文件进行编译, 生成对应二进制文件 xxx。

为了使得操作方便,可以在 vim 的配置上编写: 按 F5 时执行

exec "!nasm ".file_name

十六进制编辑器：

WinHex 和 gHex 分别是 Windows 平台和 Linux 平台的十六进制编辑器。它们可以以二进制的方式打开任意文件,并且修改文件中的二进制值。

这里使用十六进制编辑器查看软盘映像的二进制值(是否存在0x55aa)，以及将编译出的文件粘贴到软盘镜像中。

使用 vim 的%!xxd 和 %!xxd -r 也能很容易地在十六进制和文本模式切换.

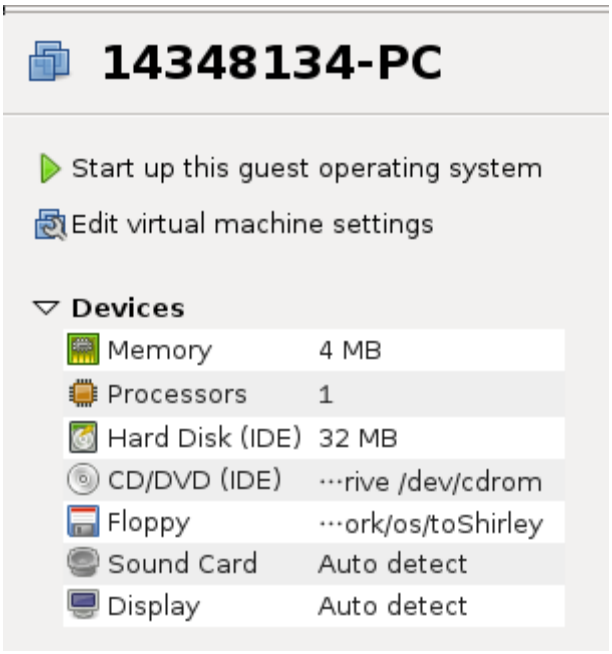
DiskWriter 能将二进制文件一键写入到软盘映像中,其原理为将二进制文件直接拷贝到软盘映像的首部。

四、实验操作

1、搭建和应用实验环境

选项卡 File -> New Virual Machine, 在"New Virual Machine Wizard"中可以设置虚拟机硬件参数。

在完成创建后,移除网络适配器(本次实验不会使用到网络),创建虚拟机完成。其中,虚拟机的名字为 14348134-PC。如图一：



图一：虚拟机配置

2、创建多个 1.44MB 容量的虚拟软盘

1.44MB 虚拟软盘参数：

> 2 面、80 道/面、18 扇区/道、512 字节/扇区
>
> 扇区总数=2 面 X 80 道/面 X 18 扇区/道 = 2880 扇区
>
> 存储容量= 512 字节/扇区 X 2880 扇区 = 1440 KB =1474560B

2.1 两种创建虚拟软盘的方式

a. 使用 dd 命令:

dd if=/dev/zero of=floppy.img bs=512 count=2880

其中, bs 为读/写缓冲区大小,单位为 byte; count 为写入的块数.

if 为输入文件名, 这里填入 0

b. 使用 WinImage 生成

File -> New, 在 Standard format 中选择 1.44MB, 选择 OK。

然后 Save 保存, 此时文件类型选择 Image file(*.ima), 文件名为 xxx.img，用 gHex 打开该文件,可以发现 510~511 字节偏移处, 十六进制分别为 55, AA. 这是一个可以在裸机环境执行的镜像.

2.2 装载软盘镜像到虚拟机:

点击虚拟机中的 Edit virtual machine settings -> Add -> Floppy Drive -> Use a floppy image, 选择镜像文件即可; 或者在已创建的软盘设备上选择镜像文件。需要勾选 Connect at power on.

使用WinImage软件 ,创建或打开一个1.44MB的软盘，选择Image -> Boot sector properties， 选择 DOS，保存即可。
或者下载 DOS 镜像，加载镜像运行虚拟机，输入 FORMAT A:/S 格式化。

3、 用 WinHex 工具将其中一个虚拟软盘的首扇区填满个人信息

一个虚拟软盘的首扇区为软盘映像文件的前 510 个字节+0x55aa.
使用 WinHex 打开软盘映像后, 选择

4、接管裸机的控制权

4.1 算法

为了使一个字符能从屏幕左边某行 45 度角下斜射出,保持一个

适当速度移动,并且碰到边缘能够反射.这里使用位置-速度的方法，即位置代表字符当前在屏幕的位置，速度代表字符下一步的位移。

初始时,位置 s 为(x,y),速度 v 为(1,1)

每次运动时, new_s = s + v,

当触碰到边缘时，速度中指向边缘的方向取反,即下一步要反弹.

4.2 文字显示

显存段地址 B800h，从这个位置开始，每两个字节的低位字节决定要显示的字,高位字节决定这个字的颜色等显示状态。文本显示的范围为 25 行 80 列。

偏移量 = ((行数(从 0 开始) * 80) + 25) * 2，即要显示的字的内存位置。

4.3 数据段 DS 设置

为了能够直接使用[内存变量]访问内存，有两种解决方法：

a. mov ds, 07c0h

b. 在汇编代码的开头写 org 7c00h，然后对 ds 赋值为 0h

原因：

主引导扇区数据为 512 字节，处理器会将主引导扇区的数据

加载到逻辑地址 0x0000:0x7c00 中 ,然后检测最后两字节是否为 0x55 和 0xAA, 若存在则主引导扇区有效, 以一个段间转移指令 jmp 0x0000:0x7c00 跳到那里继续执行 ,数据段的位置也要作相应修改。

4.4 存储

使用内存中的两个字节存储粒子(以下称会动的字符为粒子)的坐标 pos = (x,y)，另两个字节存储粒子的速度 vel = (vx,vy)两个字节即一个字可用 dw 表示, 再分别用一个字节表示粒子的字符与颜色。

当计算数据时,将内存中的数据写入寄存器中,并作相应计算,最终将结果写入内存。

其中, 寄存器 AX 主要负责累加, 存储乘法结果；BX 可用于基址偏移, 用 ES 记录偏移地址；CX 可用于循环计数。

4.5 主引导区有效标记 55aa

查阅资料可知,在汇编代码末尾加入:

```
times 510-($-$$) db 0
dw    0xaa55
```

其中 times 语句用于为空位填充 0, 0xaa55 的高位为 aa, 低位为 55

可以为 512 字节的最后两位填写 55AA，即主引导扇区有效字符.

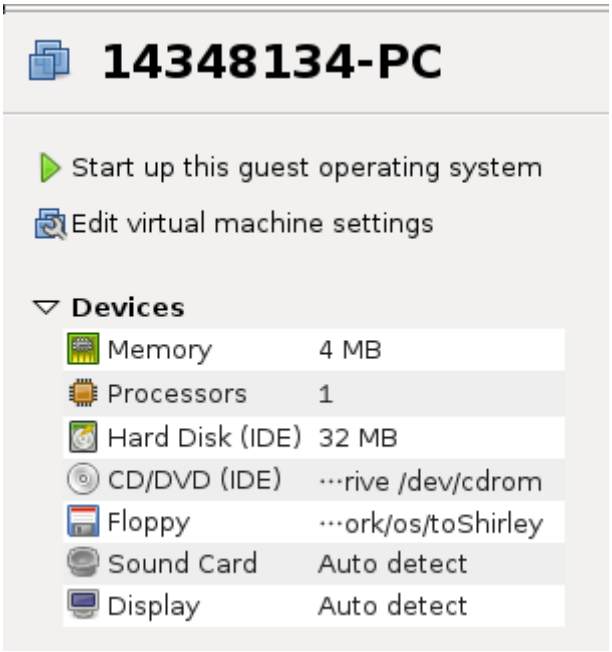
4.6 特色

使用例程(call，ret)，类似 C 语言中的调用函数,可以同时控制多个粒子移动；使用 nasm 宏，简化代码的编写；使用循环结构打印字符串如何；个人名字学号以彩色形式滑动。

五 实验过程

1. 配置虚拟机

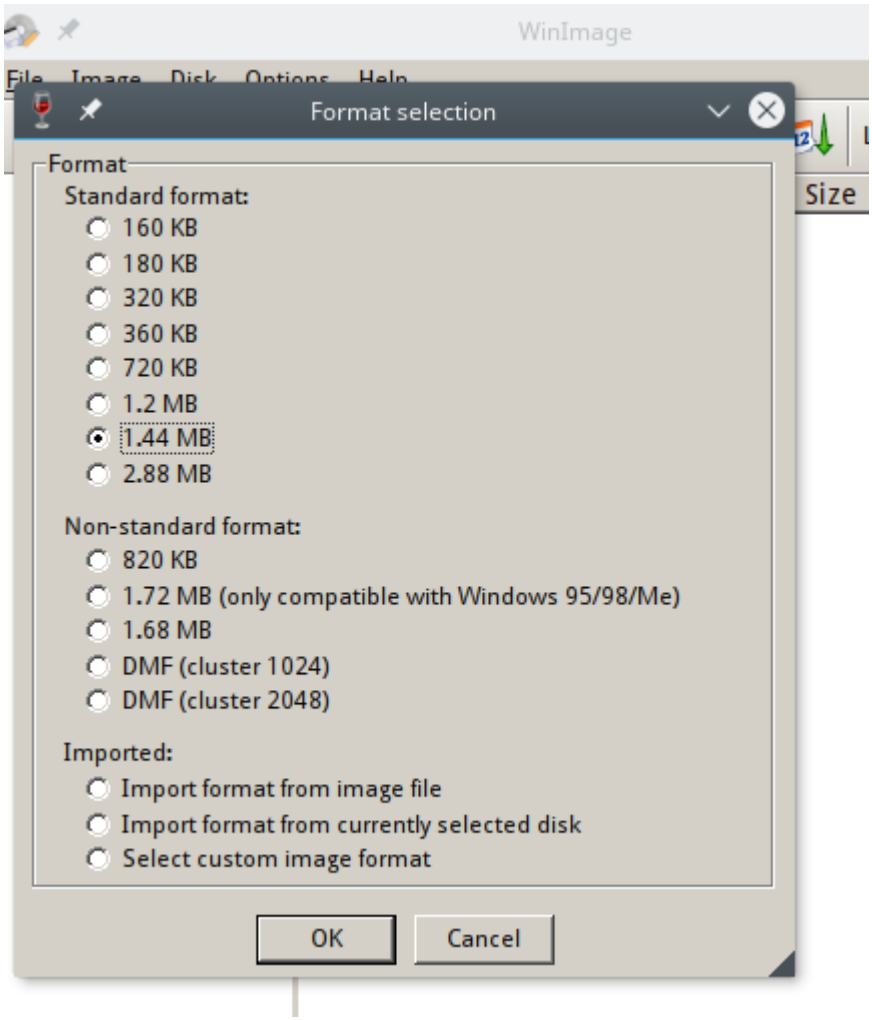
建立成功后如图二：



图二：虚拟机参数

2. 创建 1.44M 虚拟软盘

使用 WinImage 创建 1.44M 虚拟软盘如图三：



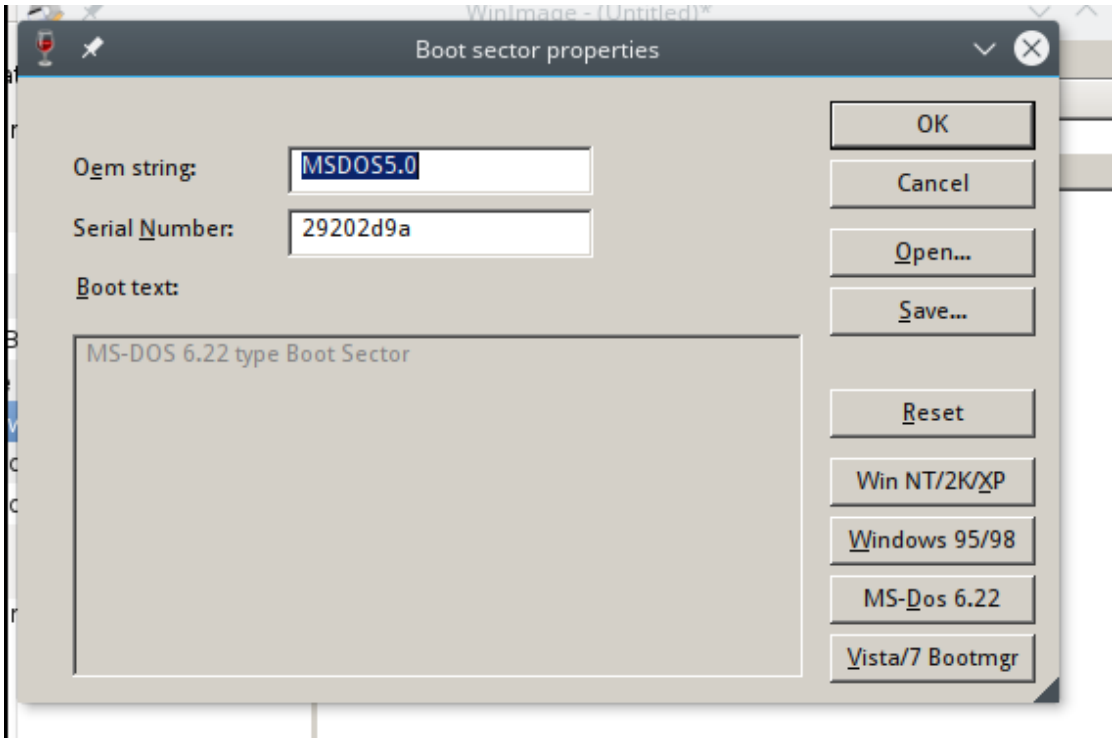
图三：使用 WinImage 创建虚拟软盘

也可以使用 dd 命令创建虚拟软盘。

WinImage 创建虚拟软盘时，要选择 ima 格式，保存的文件名的扩展名为 img。生成的文件在 510-511 字节偏移时有 55AA，而用 dd 命令创建的虚拟软盘全部为 0。

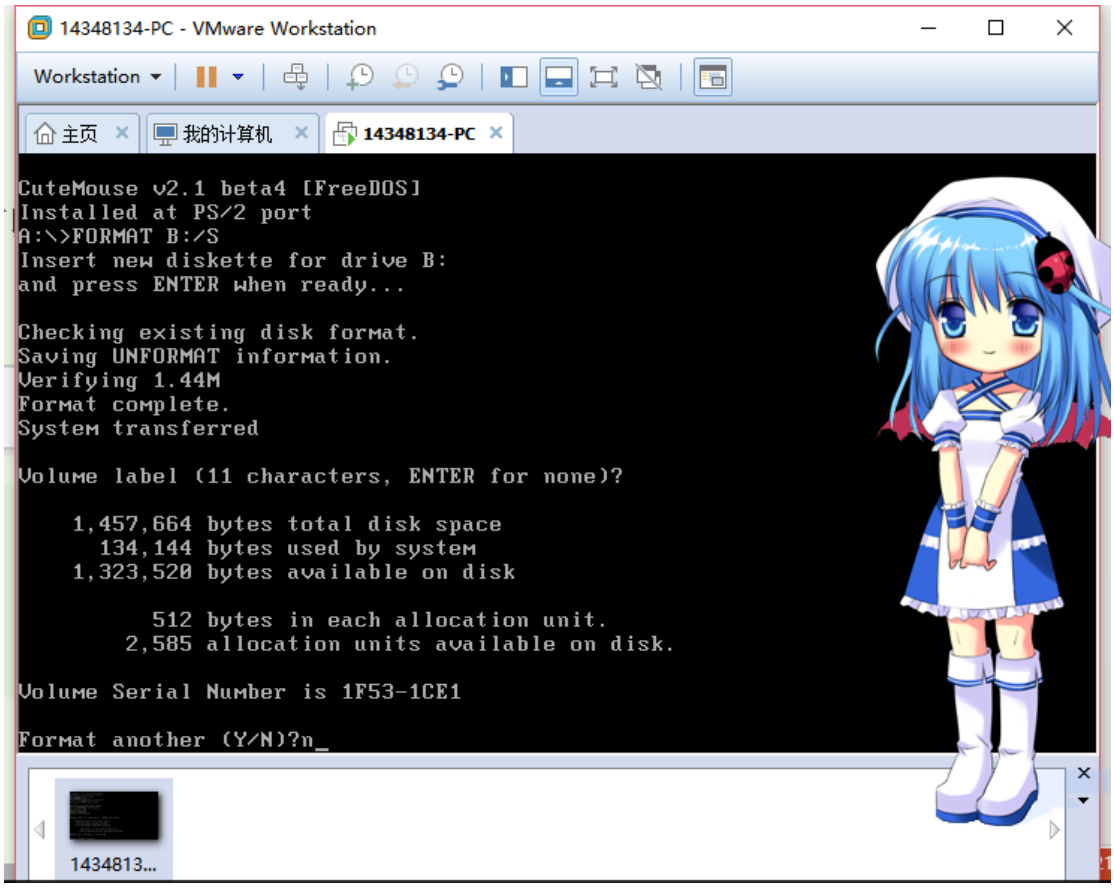
3. 格式化为 DOS 引导盘

使用 WinImage 格式化为 DOS 引导盘为图四，使用 MSDOS 格式化为 DOS 引导盘为

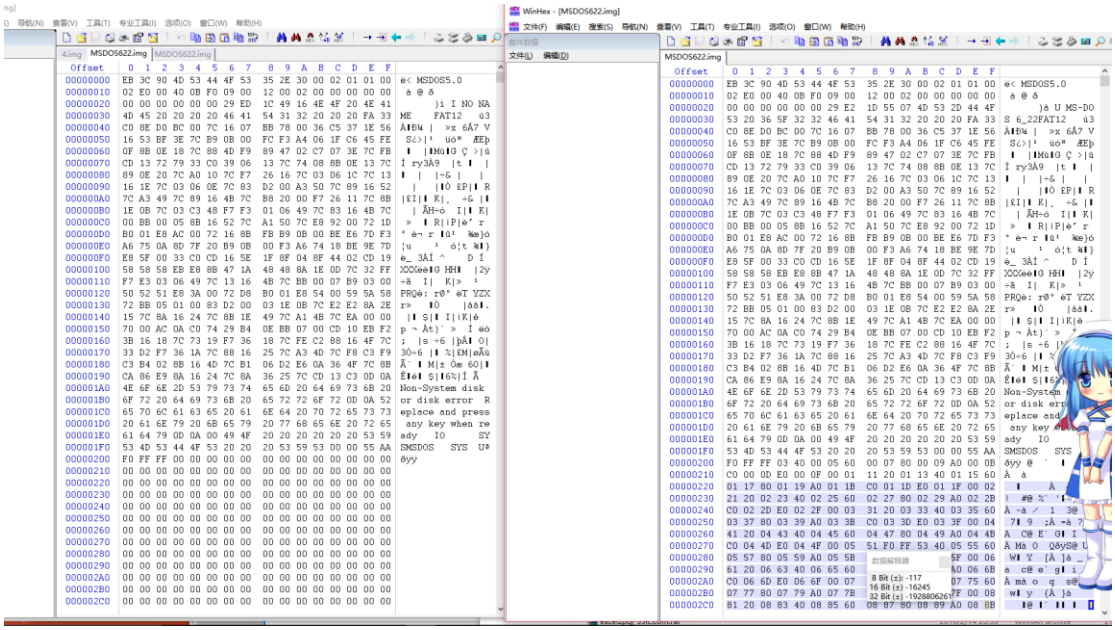


图五。

图四：用 WinImage 将软盘格式化为 DOS 引导盘



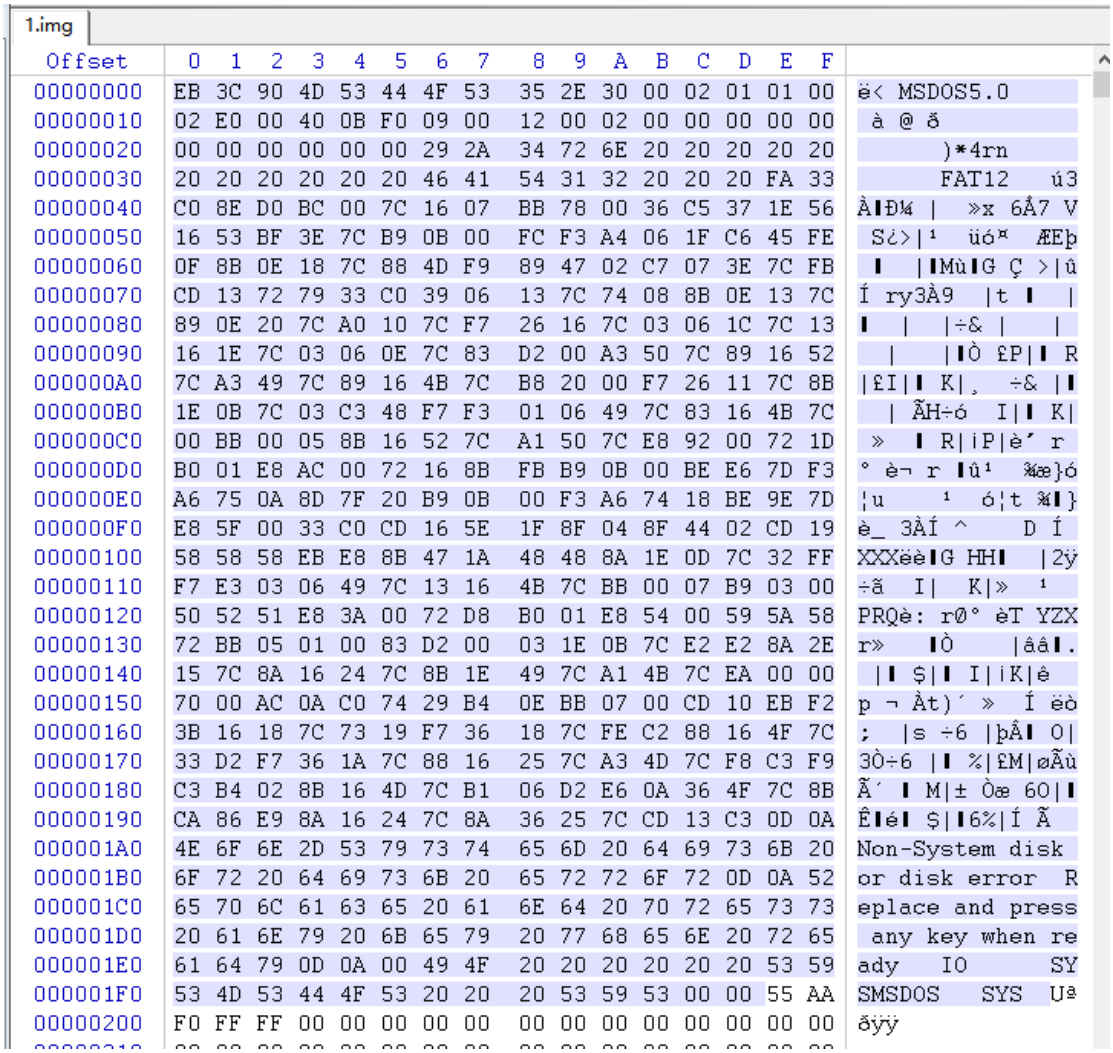
图五：用 MSDOS 将软盘格式化为 DOS 引导盘



图六：MSDOS 软盘被格式化后、格式化前的区别

如图六所示，左边和右边分别为格式化后和格式化前的文件内容，可见格式化后 0203H 后的数据被 0 覆盖。

4. 填充个人信息



1.img

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	EB	3C	90	4D	53	44	4F	53	35	2E	30	00	02	01	01	00	ë< MSDOS5.0
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	2A	34	72	6E	20	20	20	20	20	)*4rn
00000030	20	20	20	20	20	20	46	41	54	31	32	20	20	20	FA	33	FAT12 ú3
00000040	C0	8E	D0	BC	00	7C	16	07	BB	78	00	36	C5	37	1E	56	À▌Ð¼ | »x 6Å7 V
00000050	16	53	BF	3E	7C	B9	0B	00	FC	F3	A4	06	1F	C6	45	FE	S¿>|¹ üó¤ ÆEþ
00000060	0F	8B	0E	18	7C	88	4D	F9	89	47	02	C7	07	3E	7C	FB	▌ |▌Mù▌G Ç >|û
00000070	CD	13	72	79	33	C0	39	06	13	7C	74	08	8B	0E	13	7C	Í ry3À9 |t ▌ |
00000080	89	0E	20	7C	A0	10	7C	F7	26	16	7C	03	06	1C	7C	13	▌ | |÷& | |
00000090	16	1E	7C	03	06	0E	7C	83	D2	00	A3	50	7C	89	16	52	| |▌Ò £P|▌ R
000000A0	7C	A3	49	7C	89	16	4B	7C	B8	20	00	F7	26	11	7C	8B	|£I|▌ K| ¸ ÷& |▌
000000B0	1E	0B	7C	03	C3	48	F7	F3	01	06	49	7C	83	16	4B	7C	| ÃH÷ó I|▌ K|
000000C0	00	BB	00	05	8B	16	52	7C	A1	50	7C	E8	92	00	72	1D	» ▌ R|¡P|è' r
000000D0	B0	01	E8	AC	00	72	16	8B	FB	B9	0B	00	BE	E6	7D	F3	° è¬ r ▌û¹ ¾æ}ó
000000E0	A6	75	0A	8D	7F	20	B9	0B	00	F3	A6	74	18	BE	9E	7D	¦u ¹ ó¦t ¾▌}
000000F0	E8	5F	00	33	C0	CD	16	5E	1F	8F	04	8F	44	02	CD	19	è_ 3ÀÍ ^ D Í
00000100	58	58	58	EB	E8	8B	47	1A	48	48	8A	1E	0D	7C	32	FF	XXXëè▌G HH▌ |2ÿ
00000110	F7	E3	03	06	49	7C	13	16	4B	7C	BB	00	07	B9	03	00	÷ã I| K|» ¹
00000120	50	52	51	E8	3A	00	72	D8	B0	01	E8	54	00	59	5A	58	PRQè: rØ° èT YZX
00000130	72	BB	05	01	00	83	D2	00	03	1E	0B	7C	E2	E2	8A	2E	r» ▌Ò |ää▌.
00000140	15	7C	8A	16	24	7C	8B	1E	49	7C	A1	4B	7C	EA	00	00	|▌ $|▌ I|¡K|ê
00000150	70	00	AC	0A	C0	74	29	B4	0E	BB	07	00	CD	10	EB	F2	p ¬ Àt)´ » Í ëò
00000160	3B	16	18	7C	73	19	F7	36	18	7C	FE	C2	88	16	4F	7C	; |s ÷6 |þÂ▌ O|
00000170	33	D2	F7	36	1A	7C	88	16	25	7C	A3	4D	7C	F8	C3	F9	3Ò÷6 |▌ %|£M|øÃù
00000180	C3	B4	02	8B	16	4D	7C	B1	06	D2	E6	0A	36	4F	7C	8B	Ã´ ▌ M|± Òæ 6O|▌
00000190	CA	86	E9	8A	16	24	7C	8A	36	25	7C	CD	13	C3	0D	0A	Ê▌é▌ $|▌6%|Í Ã
000001A0	4E	6F	6E	2D	53	79	73	74	65	6D	20	64	69	73	6B	20	Non-System disk
000001B0	6F	72	20	64	69	73	6B	20	65	72	72	6F	72	0D	0A	52	or disk error R
000001C0	65	70	6C	61	63	65	20	61	6E	64	20	70	72	65	73	73	eplace and press
000001D0	20	61	6E	79	20	6B	65	79	20	77	68	65	6E	20	72	65	any key when re
000001E0	61	64	79	0D	0A	00	49	4F	20	20	20	20	20	20	53	59	ady IO SY
000001F0	53	4D	53	44	4F	53	20	20	20	53	59	53	00	00	55	AA	SMSDOS SYS Uª
00000200	F0	FF	FF	00	00	00	00	00	00	00	00	00	00	00	00	00	ðÿÿ

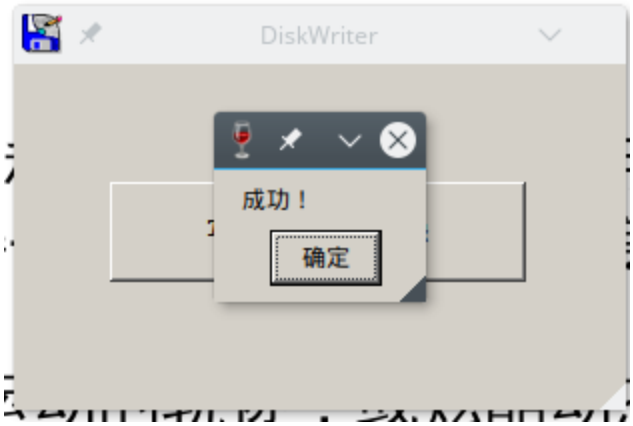
图七：填充前信息

4.img | MSDOS622.img | MSDOS622.img | 1.img

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
00000010	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
00000020	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
00000030	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
00000040	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
00000050	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
00000060	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
00000070	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
00000080	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
00000090	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
000000A0	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
000000B0	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
000000C0	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
000000D0	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
000000E0	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
000000F0	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
00000100	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
00000110	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
00000120	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
00000130	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
00000140	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
00000150	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
00000160	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
00000170	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
00000180	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
00000190	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
000001A0	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
000001B0	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
000001C0	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
000001D0	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
000001E0	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	20	20	WuKan 14348134
000001F0	57	75	4B	61	6E	20	31	34	33	34	38	31	33	34	55	AA	WuKan 14348134Uª
00000200	F0	FF	FF	00	00	00	00	00	00	00	00	00	00	00	00	00	ðÿÿ

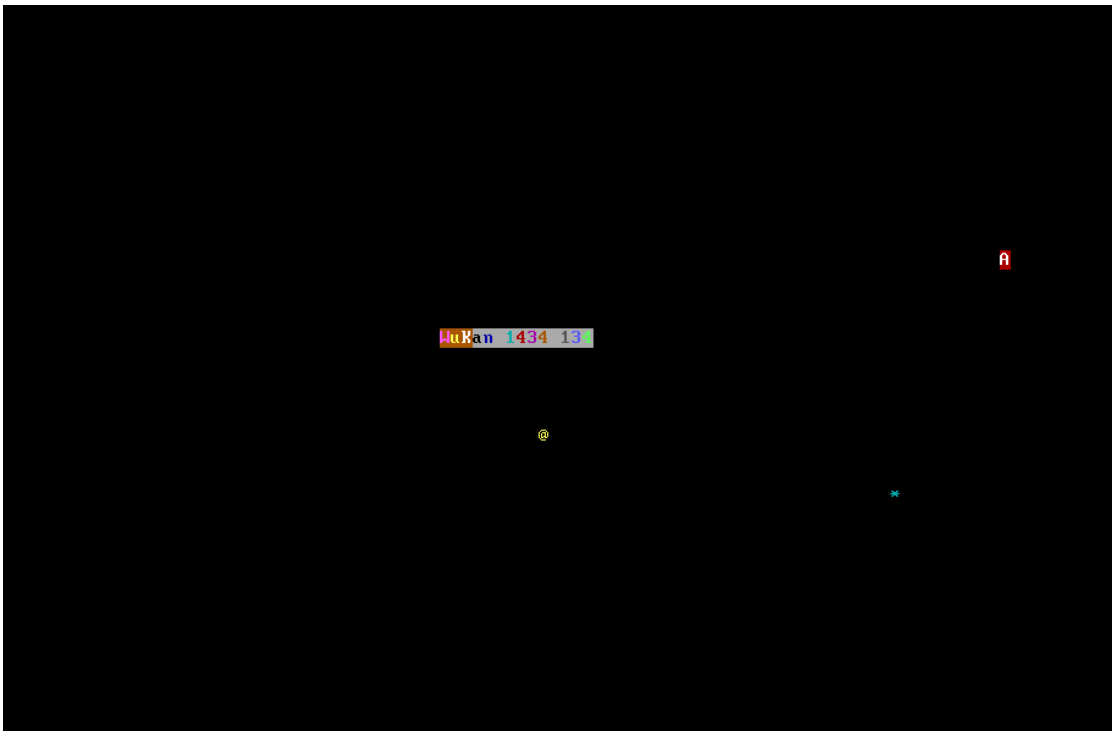
图八：填充后信息

5. 将程序写入软盘镜像



图九：将程序写入软盘镜像

6. 45 度飞翔字符实验



图十：45 度飞翔字符实验

可以看到，三个字符在按实验规则运动，中间的姓名学号信息在不停地闪烁。在这个实验中，我遇到的问题有：将一个字节的内存数据赋给两个字节的 CX 寄存器，导致显示文字出现乱码。

六 实验总结

这次实验很有趣，让我第一次能够在虚拟的裸机上运行自己的程序。

对于我来说,这次实验的主要难点在于了解什么是主引导扇区。

主引导扇区中的程序是如何加载到内存并且执行的，并且明白刚开始执行时寄存器 CS,IP，DS 的初始值. 除此之外，我也重点去了解汇编中各个寄存器的特殊作用、如何组合各个寄存器。

我的实验的特色在于实现了多个字符同时移动，其中用到了调用例程的语句、使用循环结构显示字符串、使用了 nasm 宏简化代码。我也因为兴趣做了另一个动态逐点绘制字体的程序，它的原理是通过 CX 寄存器控制循环次数。 在这个程序中，我还通过栈来保护一些寄存器的数值，即在执行某个例程之前将某个寄存器的值压入栈中，例程将结束时出栈恢复寄存器原始数据。

这里描述我探究裸机平台上，程序运行时指令和数据在内存中的位置:

经查阅资料:

主引导扇区数据为 512 字节，处理器会将主引导扇区的数据加载到逻辑地址 0x0000:0x7c00 中，然后检测最后两字节是否为 0x55 和 0xAA，若存在则主引导扇区有效，以一个段间转移指令 jmp 0x0000:0x7c00 跳到那里继续执行。

裸机平台上,程序(指令与数据)被载入 7c00 的位置。

org 指令可以规定目标程序的起始位置。

加上 org 指令后，编译出的文件中, jmp 和 call 非立即数的偏移量不会改变，运行时会加上 org 的偏移量。jmp 和 call 使用立即数时偏移量不受影响。当使用 mov 且参数为引用地址值时，编译出的文件偏移量会加上 org 的偏移量；当我不使用 org 规定程序的起始

位置时，起始的 CS:IP 为 0000h:7c00h，而 cs 为 0000h. 而使用[变量名]访问内存时,偏移量为编译出的二进制文件中的数据偏移量。

因此只有将 ds 赋值 07c0h 时，才能访问到所需数据。

另一种方式为在程序的开头加上 org 7c00h，说明之后的程序会被载入到 7c00h 这个位置。并且要将 ds 赋值为 0000h. 在这个测试中，我也发现 ax 的初始化值不一定是 0.

这次的实验,也让我发现了虚拟机与物理机的细微差距:

比如物理机将主引导区的程序写入内存后 ,要先判断主引导程序最后两个字节是否为 55aah. 若存在，则主引导区有效，执行程序。而虚拟机(VMware Station)，即使不满足 55aah 这个条件，也执行了这段程序。

另外，我想，主引导区的大小只有 512 个字节，要想运行更多的指令，可以使用 jmp xxxx:xxxx 的方式同时修改 cs 和 ip 值，实现段间跳转。

当生成的代码超过 512 个字节时，(VIM 的插件 You Complete Me 会提示)，要精简代码时，真的很刺激。

参考文献：

1. NASM 的 ORG 0100h 的实际含义 - ruyanhai
http://blog.csdn.net/ruyanhai/article/details/7177904
2. Linux 下制作虚拟软盘镜像
http://wenix.blog.51cto.com/874806/364816
3. linux 下 dd 命令详解
http://www.cnblogs.com/licheng/archive/2008/03/21/1116492.html

4. <x86 PC 汇编语言，设计与接口> - 作者：（美国）马兹迪（Muhammad Ali Mazidi）（美国）考西（Danny Causey）（美国）马兹迪（Janice Gillispie Mazidi） 译者：高升 合著者：王筱珍
5. <x86 汇编语言-从实模式到保护模式> - 李忠 王晓波 余 洁 著
<操作系统原理实验课件 - 实验 1> - 凌应标
6. <NASM 的汇编指令>
7. <nasm 介绍>

附录

文件描述：

文件名	描述
wkcn.asm	汇编文件, 45 度飞翔的字符
1.img	填充个人信息的镜像
2.img	WinImage 创建的普通 1.44M 软盘镜像
3.img	飞翔的字符镜像
4.img	全部为 0 的 1.44M 软盘镜像
5.img	WinImage 创建的 1.44M 软盘镜像 （DOS 引导盘）
MSDOS622.img	被格式化的 DOS 引导软盘镜像


;Running Ball
;Jackie Wu (Wu Kan)
;14348134
;wkcn@live.cn
;totalDelay = outDelay * inDelay
outDelay equ 50000
inDelay equ 1800
MAX_X equ 80
MAX_Y equ 25
;set data segment
mov ax,07c0h
mov ds,ax
;text window
;25 * 80
mov ax,0B800h
mov es,ax
%macro SINGLE 4
;pos dw 0000h ;from 0,0
;vel dw 0101h ;v = (1,1)
;char db '*'
;color db 07H
mov ax, [%1]
mov [pos], ax
mov ax, [%2]
mov [vel], ax
mov al, [%3]
mov [char], al
mov al, [%4]
mov [color], al
call PLAY
mov ax, [pos]
mov [%1], ax
NORMAL
<s/wkcn.asm nasm
0% : 1: 13
! trailing[26]
经查阅资料：
主引导扇区数据为512字节
处理器会将主引导扇区的数据加载到逻辑地址0x0000:0x7c00中，
然后检测最后两字节是否为0x55和0xAA，若存在则主引导扇区有效，以
一个段间转移指令 jmp 0x0000:0x7c00 跳到那里继续执行
裸机平台上，程序(指令与数据)被载入7c00的位置．
org指令可以规定目标程序的起始位置．
加上org指令后，编译出的文件中，jmp和call非立即数的偏移量不会改
变．
运行时会加上org的偏移量．jmp和call使用立即数时偏移量不受影响．
当使用mov且参数为引用地址值时，编译出的文件偏移量会加上org的偏
移量．
当我不使用org规定程序的起始位置时，起始的CS:IP为0000h:7c00h，而
cs为0000h．
而使用[变量名]访问内存时，偏移量为编译出的二进制文件中的数据偏移
量．
因此只有将ds赋值07c0h时，才能访问到所需数据．
另一种方式为在程序的开头加上org 7c00h，说明之后的程序会被载入到
7c00h这个位置．
并且要将ds赋值为0000h．在这个测试中，我也发现ax的初始化值不一定
是0．
这次的实验，也让我发现了虚拟机与
比如物理机将主引导区的程序写入
要先判断主引导程序最后两个字
效，执行程序．
而虚拟机(VMware Station)，即使
程序．
另外，我想，主引导区的大小只
以使用jmp
xxxx:xxxx的方式同时修改cs和ip值
当生成的代码超过512个字节时，
要精简代码时，
真的很刺激．
<rk/os/document/report.txt
tex
41 words
100%
195: 11

图十一： 开发环境