

引导扇区程序设计（纯汇编）

引导扇区程序设计（纯汇编）

个人信息

实验题目

实验目的

实验要求

实验方案

所用工具

虚拟机配置

实验原理

代码关键部分

伪指令org

定义常量

初始化寄存器

在屏幕上显示个人信息

实现字符反弹

填充引导程序块

用个人信息填充首扇区

实验过程

安装vmware workstation pro

安装nasm

下载

安装

创建虚拟机

使用nasm生成boot.bin文件

使用winHex把boot.bin写到floppy_0.flp的首扇区

启动虚拟机

用winHex把floppy_1.flp首扇区填满个人信息

实验总结

参考资料

个人信息

- 院系：数据科学与计算机学院
- 年纪：2018
- 姓名：王天龙
- 学号：18340168

实验题目

引导扇区程序设计（纯汇编）

实验目的

接管裸机的控制权，在屏幕上显示信息

实验要求

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. 建立自己的软件项目管理目录，管理实验项目相关文档








实验方案

所用工具

实验平台是windows10系统，使用的虚拟机软件是vmware workstation pro，使用的汇编器是nasm，使用的可视化编译十六进制文件内容工具是winhex，使用visual studio 2019编写汇编程序代码。

虚拟机配置

虚拟机为1cpu，内存为4MB，使用一个1.44MB大小的软盘，选择从软盘启动

设备	摘要
 内存	4 MB
 处理器	1
 CD/DVD (IDE)	自动检测
 软盘	正在使用文件 D:\computer_op...
 网络适配器	NAT
 声卡	自动检测
 显示器	自动检测

实验原理

要获取通过引导程序获得裸机的控制权，可以把自己的程序卸载软盘的首扇区，让计算机从软盘启动。这是因为计算机启动时，先把软盘首扇区加载到内存0x7c00处，并且跳转到哪里执行。这样我们就可以获取裸机的控制权了。而要在屏幕上显示字符，可以有多种方法，比如BIOS调用。这里我们把要显示字符送到显示卡的内存（0xB8000-0xBFFFF），以在屏幕上显示出来。

代码关键部分

伪指令org

在引导程序开头，有一个伪指令

```
org 0x7c00; 监控程序被加载到0x7c00处
```

作用是让程序里的地址标号加一个偏移量，否则就会出错，比如

```
mov si,mmsg
mmsg:
    db '18340168 wangtianlong'
    db 0x00
```

有了伪指令之后，si寄存器的值为程序开头到mmsg偏移量+0x7c00，这样cpu根据ss:si才能正确找到mmsg

定义常量

对于一些经常用到的常量，我们可以用equ给它起个有意义的"别名"，可以增加代码可读性，减少bug

```
Dn_Rt equ 1
Up_Rt equ 2
Up_Lt equ 3
Dn_Lt equ 4
delay equ 5000
ddelay equ 5000
```

Dn_Rt、Up_Rt、Up_Lt、Dn_Lt为字符移动的四个方向，delay*ddelay是延时

初始化寄存器

然后是初始化寄存器，要注意到不能直接使用cs的值赋给ds、ss，同时不能直接赋值给es，否则会报错

```
mov ax,cs
mov ss,ax
mov ds,ax
mov ax,0xb800
mov es,ax;要注意不能字节赋值给es
```

在屏幕上显示个人信息

要通过显存在屏幕上第x行，第y列显示字符，首先要把字符ASCII送到0xB800:(x*80+y)* 2处，然后把字符的属性，字符的属性包括颜色、背景颜色、是否闪烁、是否加亮，编码送到0xB800:(x*80+y)* 2+1处。

```
mov si,mmsg
xor bx,bx
print:
    mov al,[si];获取字符
    mov ah,0x0f;属性编码，白字黑底，加亮
    add si,1;指向下一个字符
    cmp al,0x00
    jz start
    mov [es:bx],ax
    inc bx
    inc bx
    jmp print
```

其中mmsg定义如下

```
mmsg:
    db '18340168 wangtianlong'
    db 0x00
```

实现字符反弹

首先，是关于字符的一些信息，比如它的移动方向、位置、颜色，这些信息定义在msg字段

```

msg:
    db 1;position x si;行
    db 0;position y si+1; 列
    db 'A';char si+2
    db Dn_Lt;direction si+3
    db 0;color si+4, 多种颜色循环显示

```

获取这些信息先要把msg赋值给si，然后用[si+offset]来获取

```

start:
    mov si,msg

```

要让字符动起来，就要判断字符的移动方向，以判断右下为例

```

next1:
    mov cx,delay
    mov al,Dn_Rt
    cmp al,[si+3];direction
    jz D_R

```

如果字符的移动方向为右下方，则字符位置行加一，列也加一，同时还要判断是否到达边界，要不要改变方向，然后才能打印

```

D_R:
    mov al,[si];x行
    inc al;x+1
    cmp al,25;判断是否到底
    jnz next3;没有到底
    mov al,Up_Rt;到底就反弹成右上
    mov [si+3],al;改变方向
    mov al,23;反弹
next3:
    mov [si],al;更新位置
    mov al,[si+1];y列
    inc al;y+1
    cmp al,80;判断是否到达右边界
    jnz next4
    mov al,Dn_Lt;到了有边界就反弹成左上
    mov [si+3],al;改变方向
    mov al,78;反弹
next4:
    mov [si+1],al;更新位置
    jmp next2

```

为了让字符移动得不要太快，我们要让字符在一个位置停顿一段时间才可以移动到下一个位置，用双层循环来实现延时，一共循环delay*ddelay次

```

loop1:
    jmp show
next1:
    mov cx,delay
    .....

next2:    ;外层循环
    mov ax,delay
next11:   ;内层循环
    dec ax
    jnz next11
    dec cx
    jnz next2
    jmp loop1

```

打印字符，打印方法同样是送显存来打印

```

show:
    mov al,[si];x行
    xor ah,ah
    mov bx,80;x*80
    mul bx
    mov bl,[si+1];y列
    xor bh,bh
    add ax,bx
    shl ax,1;(x*80+y)*2
    mov bx,ax
    mov ah,[si+4];color
    and ah,0x0f
    or ah,0x08;高亮
    mov al,[si+2];字符
    mov [es:bx],ax
    inc ah
    mov [si+4],ah;改变颜色
    jmp next1

```

填充引导程序块

一个盘块大小为512个字节，上面得程序远远没有达到，所以我们要用0来填充，同时，引导扇区要求用0x55,0xaa结束

```

times 510-($-$$) db 0x00
db 0x55,0xaa

```

用个人信息填充首扇区

这个部分主要用python脚本实现，具体如下

```

data=list('18340168 wangtianlong ')
print(data)
dataSize=len(data)
with open('test.bin','wb') as f:
    for i in range(512):
        f.write(bytes(data[i%dataSize],encoding='UTF-8'))

```

执行上述脚本，即可以获得一个大小为512byte的test.bin文件，再用winHex写到软盘首扇区即可。

实验过程

安装vmware workstation pro

由于vmware workstation pro在大一的时候就已经安装了，所以不能重现安装过程，这里附上参考链接<https://blog.csdn.net/happymagic/article/details/84668719>

安装nasm

下载

下载链接<https://www.nasm.us/pub/nasm/releasebuilds/2.14.02/win32/nasm-2.14.02-installer-x86.exe>

安装

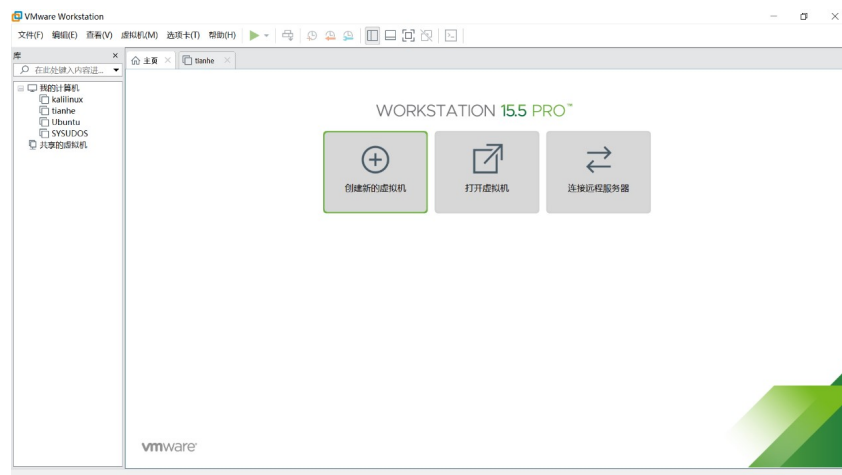
以管理员身份运行

 nasm-2.14.02-installer-x86	2020/4/29 16:09	应用程序	913 KB
--	-----------------	------	--------

然后一路点默认即可

创建虚拟机

打开vmware station，点击创建新的虚拟机



安装类型选择经典，点击下一步



选择稍后安装操作系统，点击下一步



客户机操作系统选择其他，版本选MS-DOS，点击下一步

新建虚拟机向导

选择客户机操作系统

此虚拟机中将安装哪种操作系统？

客户机操作系统

☐ Microsoft Windows(W)

☐ Linux(L)

☐ VMware ESX(X)

☒ 其他(O)

版本(V)

MS-DOS

帮助

< 上一步(B)

下一步(N) >

取消

输入虚拟机名称，选择安装位置，点击下一步

新建虚拟机向导

命名虚拟机

您希望该虚拟机使用什么名称？

虚拟机名称(V):

MS-DOS

位置(L):

D:\computer_operation_system\MS-DOS

浏览(R)...

在“编辑”>“首选项”中可更改默认位置。

< 上一步(B)

下一步(N) >

取消

指定磁盘大小，点击下一步

新建虚拟机向导

×

指定磁盘容量

磁盘大小为多少？

虚拟机的硬盘作为一个或多个文件存储在主机的物理磁盘中。这些文件最初很小，随着您向虚拟机中添加应用程序、文件和数据而逐渐变大。

最大磁盘大小 (GB)(S):

针对 MS-DOS 的建议大小: 2 GB

☒ 将虚拟磁盘存储为单个文件(O)

☐ 将虚拟磁盘拆分成多个文件(M)

拆分磁盘后，可以更轻松地在计算机之间移动虚拟机，但可能会降低大容量磁盘的性能。

帮助

< 上一步(B)

下一步(N) >

取消

点击完成

新建虚拟机向导

×

已准备好创建虚拟机

单击“完成”创建虚拟机。然后可以安装 MS-DOS。

将使用下列设置创建虚拟机：

名称：	MS-DOS
位置：	D:\computer_operation_system\MS-DOS
版本：	Workstation 15.x
操作系统：	MS-DOS
硬盘：	1 GB
内存：	4 MB
网络适配器：	NAT
其他设备：	CD/DVD, 声卡

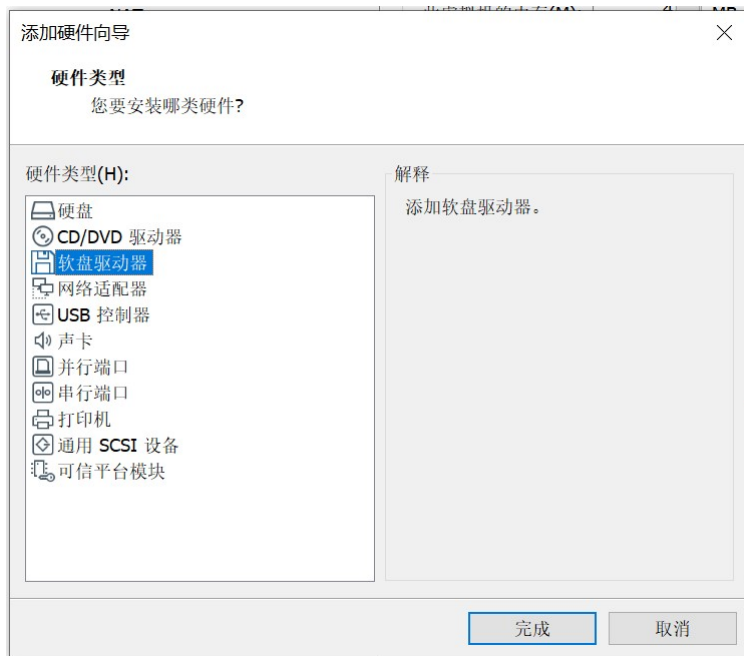
自定义硬件(C)...

< 上一步(B)

完成

取消

创建虚拟软盘floppy.flp



至此，虚拟机创建完成

使用nasm生成boot.bin文件

在命令行(windows 用win+R输入cmd打开)里输入

```
nasm boot.asm -o boot.bin
```

使用winHex把boot.bin写到floppy_0.flp的首扇区

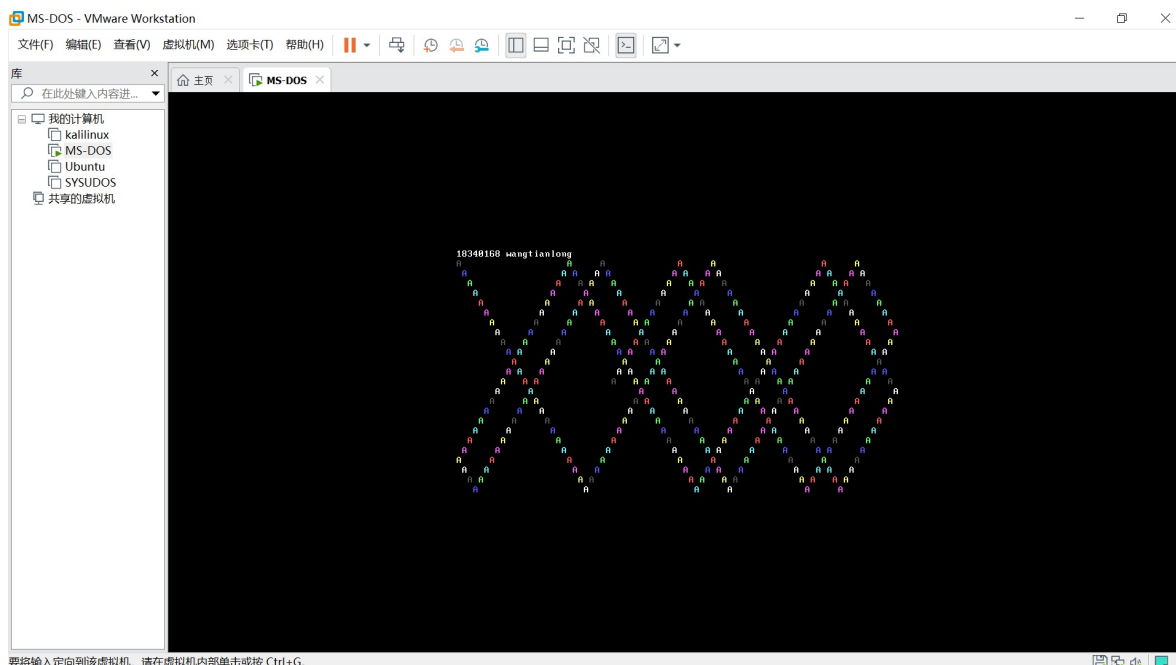
boot.bin

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
00000000	5C	C8	8E	D0	8E	D8	B8	00	B8	8E	C0	BE	12	7D	31	DB	8A	04	B4	0F	81	C6	01	00	3C	00	74	08	26	89	07	43
00000032	43	E9	EC	FF	BE	28	7D	E9	BD	00	B9	50	C3	B0	01	3A	44	03	74	3E	B0	04	3A	44	03	74	5E	B0	02	3A	44	03
00000064	74	7E	8A	04	FE	C8	3C	00	75	07	B0	04	88	44	03	B0	02	88	04	8A	44	01	FE	C8	3C	FF	75	07	B0	02	88	44
00000096	03	B0	01	88	44	01	B8	88	13	48	75	FD	49	75	F7	E9	B5	FF	8A	04	FE	C0	3C	19	75	07	B0	02	88	44	03	B0
00000128	17	88	04	8A	44	01	FE	C0	3C	50	75	07	B0	04	88	44	03	B0	4E	88	44	01	E9	CD	FF	8A	04	FE	C0	3C	19	75
00000160	07	B0	03	88	44	03	B0	17	88	04	8A	44	01	FE	C8	3C	FF	75	07	B0	01	88	44	03	B0	01	88	44	01	E9	A6	FF
00000192	8A	04	FE	C8	3C	00	75	07	B0	01	88	44	03	B0	02	88	04	8A	44	01	FE	C0	3C	50	75	07	B0	03	88	44	03	B0
00000224	4E	88	44	01	E9	7F	FF	8A	04	30	E4	BB	50	00	F7	E3	8A	5C	01	30	FF	01	D8	D1	E0	89	C3	8A	64	04	80	E4
00000256	0F	80	CC	08	8A	44	02	26	89	07	FE	C4	88	64	04	E9	18	FF	31	38	33	34	30	31	36	38	20	77	61	6E	67	74
00000288	69	61	6E	6C	6F	6E	67	00	01	00	41	04	00	00	00	00	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	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	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	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	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	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	00	00	00	00	00	00	00	00	00	00	00	00	00	00	55	AA

floppy_0.flp

00000000	B8	00	7C	8E	D0	D8	B8	00	B8	8E	C0	BE	13	01	31	DB	8A	04	B4	0F	81	C6	01	00	3C	00	74	08	26	89	07	
00000032	43	43	E9	EC	FF	BE	29	01	E9	BD	00	B9	50	C3	B0	01	3A	44	03	74	3E	B0	04	3A	44	03	74	5E	B0	02	3A	44
00000064	03	74	7E	8A	04	FE	C8	3C	00	75	07	B0	04	88	44	03	B0	02	88	04	8A	44	01	FE	C8	3C	FF	75	07	B0	02	88
00000096	44	03	B0	01	88	44	01	B8	88	13	48	75	FD	49	75	F7	E9	B5	FF	8A	04	FE	C0	3C	19	75	07	B0	02	88	44	03
00000128	B0	17	88	04	8A	44	01	FE	C0	3C	50	75	07	B0	04	88	44	03	B0	4E	88	44	01	E9	CD	FF	8A	04	FE	C0	3C	19
00000160	75	07	B0	03	88	44	03	B0	17	88	04	8A	44	01	FE	C8	3C	FF	75	07	B0	01	88	44	03	B0	01	88	44	01	E9	A6
00000192	FF	8A	04	FE	C8	3C	00	75	07	B0	01	88	44	03	B0	02	88	04	8A	44	01	FE	C0	3C	50	75	07	B0	03	88	44	03
00000224	B0	4E	88	44	01	E9	7F	FF	8A	04	30	E4	BB	50	00	F7	E3	8A	5C	01	30	FF	01	D8	D1	E0	89	C3	8A	64	04	80
00000256	E4	0F	80	CC	08	8A	44	02	26	89	07	FE	C4	88	64	04	E9	18	FF	31	38	33	34	30	31	36	38	20	77	61	6E	67
00000288	74	69	61	6E	6C	6F	6E	67	00	01	00	41	04	00	00	00	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	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	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	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	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	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	00	00	00	00	00	00	00	00	00	00	00	00	00	00	55	AA
00000512	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	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	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000576	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000608	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000640	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000672	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

启动虚拟机



用winHex把floppy_1.flp首扇区填满个人信息

00000000	31 38 33 34 30 31 36 38	20 77 61 6E 67 74 69 61	6E 6C 6F 6E 67 20 31 38	33 34 30 31 36 38 20 77	18340168 wangtianlong 18340168 w
00000032	61 6E 67 74 69 61 6E 6C	6F 6E 67 20 31 38 33 34	30 31 36 38 20 77 61 6E	67 74 69 61 6E 6C 6F 6E	angtianlong 18340168 wangtianlon
00000064	67 20 31 38 33 34 30 31	36 38 20 77 61 6E 67 74	69 61 6E 6C 6F 6E 67 20	31 38 33 34 30 31 36 38	g 18340168 wangtianlong 18340168
00000096	20 77 61 6E 67 74 69 61	6E 6C 6F 6E 67 20 31 38	33 34 30 31 36 38 20 77	61 6E 67 74 69 61 6E 6C	wangtianlong 18340168 wangtianl
00000128	6F 6E 67 20 31 38 33 34	30 31 36 38 20 77 61 6E	67 74 69 61 6E 6C 6F 6E	67 20 31 38 33 34 30 31	ong 18340168 wangtianlong 183401
00000160	36 38 20 77 61 6E 67 74	69 61 6E 6C 6F 6E 67 20	31 38 33 34 30 31 36 38	20 77 61 6E 67 74 69 61	68 wangtianlong 18340168 wangtia
00000192	6E 6C 6F 6E 67 20 31 38	33 34 30 31 36 38 20 77	61 6E 67 74 69 61 6E 6C	6F 6E 67 20 31 38 33 34	nlng 18340168 wangtianlong 1834
00000224	30 31 36 38 20 77 61 6E	67 74 69 61 6E 6C 6F 6E	67 20 31 38 33 34 30 31	36 38 20 77 61 6E 67 74	0168 wangtianlong 18340168 wangt
00000256	69 61 6E 6C 6F 6E 67 20	31 38 33 34 30 31 36 38	20 77 61 6E 67 74 69 61	6E 6C 6F 6E 67 20 31 38	ianlong 18340168 wangtianlong 18
00000288	33 34 30 31 36 38 20 77	61 6E 67 74 69 61 6E 6C	6F 6E 67 20 31 38 33 34	30 31 36 38 20 77 61 6E	340168 wangtianlong 18340168 wan
00000320	67 74 69 61 6E 6C 6F 6E	67 20 31 38 33 34 30 31	36 38 20 77 61 6E 67 74	69 61 6E 6C 6F 6E 67 20	gtianlong 18340168 wangtianlong
00000352	31 38 33 34 30 31 36 38	20 77 61 6E 67 74 69 61	6E 6C 6F 6E 67 20 31 38	33 34 30 31 36 38 20 77	18340168 wangtianlong 18340168 w
00000384	61 6E 67 74 69 61 6E 6C	6F 6E 67 20 31 38 33 34	30 31 36 38 20 77 61 6E	67 74 69 61 6E 6C 6F 6E	angtianlong 18340168 wangtianlon
00000416	67 20 31 38 33 34 30 31	36 38 20 77 61 6E 67 74	69 61 6E 6C 6F 6E 67 20	31 38 33 34 30 31 36 38	g 18340168 wangtianlong 18340168
00000448	20 77 61 6E 67 74 69 61	6E 6C 6F 6E 67 20 31 38	33 34 30 31 36 38 20 77	61 6E 67 74 69 61 6E 6C	wangtianlong 18340168 wangtianl
00000480	6F 6E 67 20 31 38 33 34	30 31 36 38 20 77 61 6E	67 74 69 61 6E 6C 6F 6E	67 20 31 38 33 34 30 31	ong 18340168 wangtianlong 183401
00000512	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	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	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000576	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000608	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000640	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000672	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	

实验总结

通过这次试验，我加深了对计算机启动过程、引导程序、汇编语言的理解。最大的收获是对org指令有了进一步的理解。一开始，我以为是因为org 0x7c00指令所以程序才会被以到内存的0x7c00处，其实不然，计算机启动时，操作系统的bios把硬盘的首扇区读到内存的0x7c00，而org 0x7c00的作用只是让标量加上一个偏移量0x7c00。该开始做实验时，由于没有org伪指令，屏幕上显示的都是一些乱码，原因就是没有加上0x7c00这个偏移量，加上org 0x7c00就完美解决问题了。

参考资料

<https://blog.csdn.net/nethanhan/article/details/8095556>

<https://blog.csdn.net/u011894856/article/details/44513947>