《操作系统实验》

实验报告

（实验一）

学 院 名 称 ：	数据科学与计算机学院
专业（班级）：	16 计算机类 4 班
学 生 姓 名 ：	陈泓仰
学　　号	15303009
时　　间 ：	2018　年　　月　　日

成 绩 ：

实验一 ： 裸机图案引导

一. 实验目的

1. 搭建一套开发操作系统的实验环境及工具链
2. 掌握虚拟机的使用方法
3. 掌握计算机加载引导程序的过程

二. 实验要求

实验的具体内容与要求。

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. 实验工具和环境：

本次实验是在Mac OS系统上进行。所有的工具都是自己根据老师在Windows下所需要的软件功能自己在网上查阅资料收集而来。

a) 汇编器：NASM

NASM 是一个为可移植性与模块化而设计的一个 80x86 的汇编器。它可以将汇编代码编译成二进制文件。

在mac上安装只需要在终端上输入代码：

```
MacBook-Air:~ chen$ brew install nasm
```

等待运行完毕即可。

b) 虚拟机：VmWare Fusion

使用虚拟机来做操作系统实验，方便快捷安全，不会存在对现有操作系统造成损害的风险。

mac上的虚拟机VmWare有自己的对应版本Fusion，在网上可以找到破解版及其破解教程

c) 十六进制编辑器：

老师所使用的十六进制编辑器是WinHex，它可以以16进制的方式打开文件，

并且修改文件中的16进制值。

由于WinHex并不支持Mac OS系统，因此在本机实验使用了软件HexFiend。

d) 空白磁盘映像文件生成：

在Windows系统下，VmWare就可以直接生成一个空白的磁盘映像。然而在Mac OS系统中并不支持这种操作。通过查阅了相关资料发现，可以使用dd命令行，来建立多个1.44M的磁盘映像。

命令行如下：

```
hongyangchendeMacBook-Air:desktop chen$ dd if=/dev/zero of=floppy.img bs=512 co
unt=2880
2880+0 records in
2880+0 records out
1474560 bytes transferred in 0.042038 secs (35076865 bytes/sec)
```

2. 相关基础原理

在搭建好了实验环境之后，我们需要了解一些有关于本次操作系统实验所需要的基本知识。

a) 电脑开机是如何加载磁盘的？

当计算机通电之后，它最开始执行的是BIOS（基本输入/输出系统）程序，它本质上是一个内置在系统中的操作系统。BIOS执行一些基本的硬件检测（如内存检查等），并会绘制特殊图形到屏幕上。在这之后，BIOS会开始从某个可以找到的媒介上加载操作系统。然后大部分计算机会跳转到硬盘驱动器处开始执行主引导区（MBR，即一个硬盘驱动器最开始的512bytes）的代码。有些计算机则会在一个软盘（启动扇区）或CD-ROM上找可执行代码。

计算机具体会去哪里寻找引导程序，依赖于引导顺序，我们可以在BIOS中的选项中（硬盘或软盘或CD-ROM）指定。所以为了让计算机能够在我们所指定的位置中识别出引导扇区，我们需要将引导程序的最后面俩个字节改为55AA。

所以，启动过程如下：

1) 计算机启动并执行BIOS代码。

2) BIOS程序寻址媒介来加载操作系统。

3) BIOS从指定的媒介中加载512bytes的引导扇区，开始执行。

4) 引导扇区去加载操作系统本身，或者更复杂的引导程序。

b) org 07C00H的作用

BIOS会将引导程序固定加载到07C00h。而nasm在编译时的地址是从一开始就用0000H开始相对计算的，如果我们在内存中定义了一个变量的话，在传址的时候，由于他是相对寻址，他的实际偏移地址与我们程序数据偏移地址不同。我们必须在07C00H处，去进行内存数据传址。

因此org07C00h实际上就是改变数据的偏移地址7C00H，让他从07c00h处开始编译。

c) nasm语法

上学期在计算机组成原理中有学习过masm汇编，它与nasm有一些细节上的不同，需要我们浏览nasm手册才能知道。

3. 程序实现的功能描述：

一个字符在屏幕里不断的跳跃反弹，我的名字以字符串的形式在屏幕中显示。同时字符在撞到边缘时会改变自己以及字符串的颜色。也就是一个粒子以及字符串能够随着运动不断变色。

4. 程序流程：

a) 前期的传址传值操作。

b) 单一字符运动模块

c) 字符显示模块

d) 字符串显示模块

e) 变色操作

f) 数据声明

5. 算法说明

a) 运动模块

为了模拟一个字符从屏幕左边某行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。用x，y定义字符的位置，每次字符的移动，位置也会随之改变。我们将字符运动轨迹定义为四个方向函数：右下，右上，左下，左上。字符在每个方向上运动的时候如果碰到了边缘则会产生俩种不一样的结果，此时又可以定义8个轨迹方向改变函数：右下到右上，右下到左下，右上到右下，右上到左上，左下到左上，左下到右下，左上到右上，左上到左下。每次运动轨迹改变函数又会调用方向函数。在反射的过程中使用数学知识。从而实现字符的运动。

b) 字符显示模块

显存段地址是0B800H，显存段是ES。从这个位置开始，每俩个字节的低位字节决定要显示的字，高位字节决定字的颜色等显示状态。文本显示范围是25*80

偏移量 = （（行数*80）+ 最后一行列数）*2，即可表示字的显示位置。

c) 字符串显示模块

使用了BIOS的10H中断下的13号来显示字符串，具体参数如下：

i. AH＝13H

ii. AL＝显示方式

如果AL＝0，表示目标字符串仅仅包含字符，属性在BL中包含，不移动光标

如果AL＝1，表示目标字符串仅仅包含字符，属性在BL中包含，移动光标

如果AL＝2，表示目标字符串包含字符和属性，不移动光标

如果AL＝3，表示目标字符串包含字符和属性，移动光标

总之，可以归纳为：

| BIT7 | BIT6 | BIT5 | BIT4 | BIT3 | BIT2 | BIT1 | BIT0 | AL

BIT0为0表示不移动光标，为1表示移动光标

BIT1为0表示字符串仅包含字符，为1表示字符串包含属性

BIT2~BIT7未使用

iii. BH表示视频区页数

iv. 如果AL的BIT1为0，则BL表示显示属性。属性为：

| BIT7 | BIT6 | BIT5 | BIT4 | BIT3 | BIT2 | BIT1 | BIT0 | BL

BIT7：背景是否闪烁。0不闪烁，1闪烁

BIT6~BIT4为背景色，分别为RGB，000为黑色，111为白色

BIT3为1，则前景色加亮，为0则不加亮

BIT2－BIT0为前景色，意义同背景色

v. CX为字符串长度

vi. DH表示在第几行显示（0为第一行）

vii. DL表示在第几列显示（0为第一列）

viii. ES：BP指向字符串

四. 实验过程与结果

说明：根据需要书写相关内容，如：

1. 工具配置过程，小项目细节

a) 虚拟机配置


选择安装方法
从光盘或映像中安装
将您的 ISO 文件拖到此处以开始安装
迁移您的 PC
从恢复分区中安装 macOS
导入现有虚拟机
从 Boot Camp 安装
创建自定虚拟机
在远程服务器上创建虚拟机
取消
继续

选择创建自定虚拟机。

选择操作系统为：“其他”

一直默认选项，最后得到虚拟机。

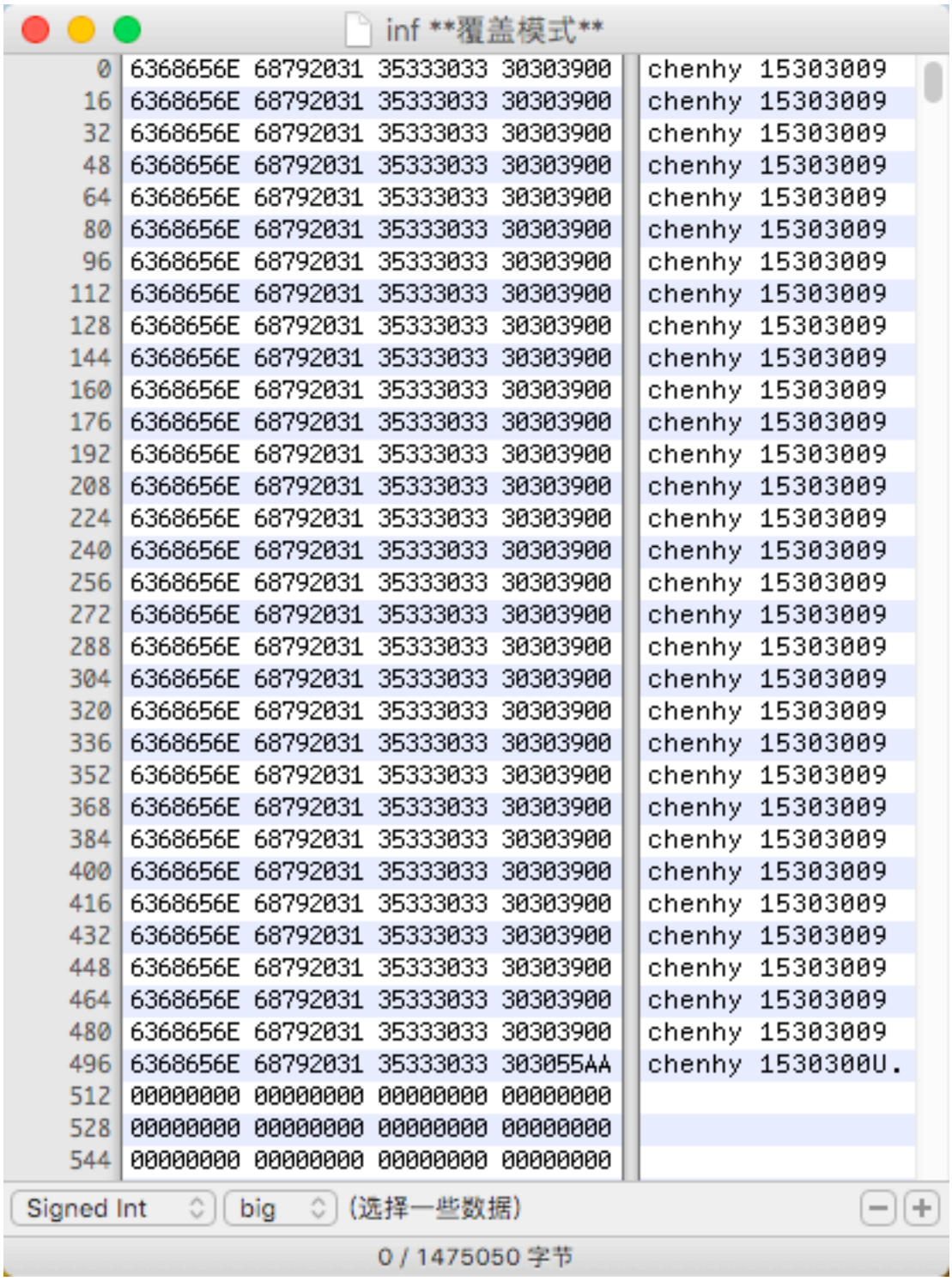
b) 其他安装过程上述已提及

c) 创建1.44M软盘

输入命令行：

```
hongyangchendeMacBook-Air:desktop chen$ dd if=/dev/zero of=floppy.img bs=512 co
unt=2880
2880+0 records in
2880+0 records out
1474560 bytes transferred in 0.042038 secs (35076865 bytes/sec)
```

d) 用HexFiend在1.44M软盘填充个人信息：



2. 字符运动实验

a) 模块划分：

i. 运动模块：使用老师的stone.asm模块，但是加上一个消除字符（kill）函数以及一个变色函数。

下面附上程序及其解释：

```
DnRt:
    call kill
    inc word [x]
    inc word [y]
    mov bx, word [x]
    mov ax, 25
    sub ax,bx
   jz dr2ur
    mov bx,word [y]
    mov ax,80
    sub ax,bx
   jz dr2dl
    jmp show
dr2ur:
    mov byte [color],BLUE
    mov word [x],23
    mov byte [rdul],Up_Rt
    jmp show
dr2dl:
    mov byte [color],sR
    mov word [y],78
    mov byte [rdul],Dn_Lt
    jmp show
```

字符运动函数分为四个函数，右下、右上、左下与左上。每个函数如图。显示处理下一次点的坐标，然后判断是否有接触边缘。然后调用显示函数。

其中通过这一函数：

```
    mov byte [color],BLUE
```

来改变运动字符的颜色。

在每次调用运动函数时都会调用kill函数，它的作用就是消除运动轨迹，使得屏幕只有一个字符在不断的运动。

```
kill:
    xor ax,ax
    mov ax,word [x]
    mov bx,80
    mul bx
    add ax,word [y]
    mov bx,2
    mul bx
    mov bx,ax
    mov ah,byte [kcolor]
    mov al,byte [kchar]
    mov [es:bx],ax
    ret
```

这个函数其实实现原理很简单，就是每次运动之后，将该坐标字符的颜色改变为黑色即可。

显示函数（show）

```
show:
    call print
    xor ax,ax
    mov ax,word [x]
    mov bx,80
    mul bx
    add ax,word [y]
    mov bx,2
    mul bx
    mov bx,ax
    mov ah,byte [color]
    mov al,byte [char]
    mov [es:bx],ax
    jmp loop1
```

通过字符在屏幕上的显示原理编写函数。显存段地址是0B800H，显存段是ES。从这个位置开始，每俩个字节的低位字节决定要显示的字，高位字节决定字的颜色等显示状态。文本显示范围是25*80。

偏移量 = （（行数*80）+ 最后一行列数）*2，即可表示字的显示位置。

ii. 个人信息显示模块：

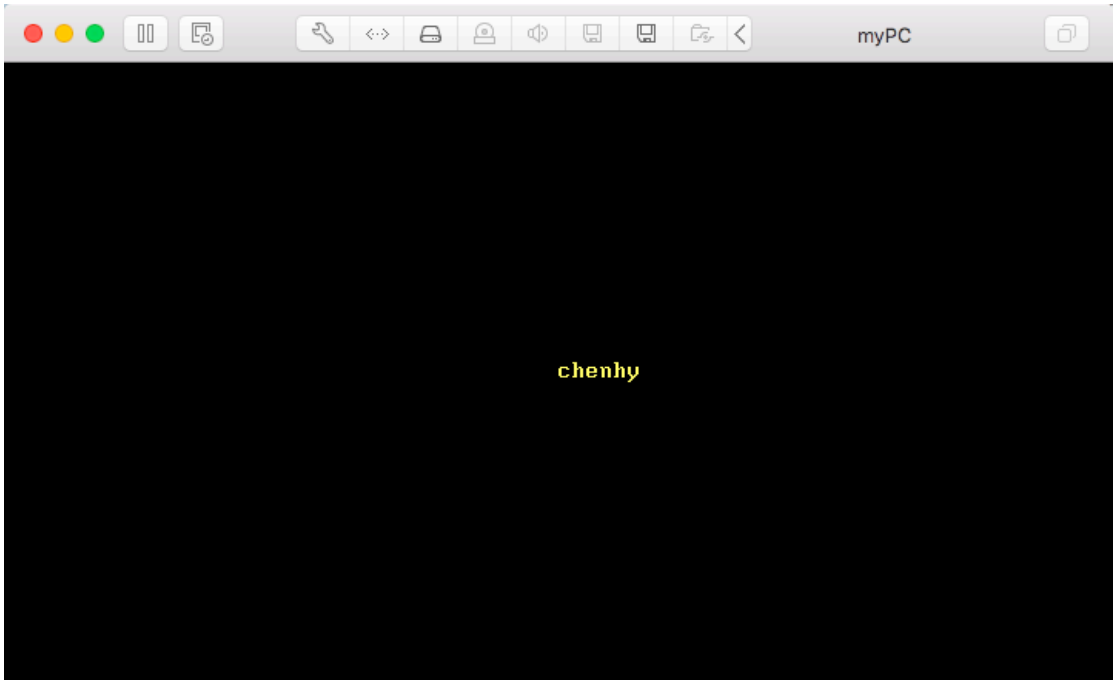
在写这个模块的时候自己在另一个程序上面写了一个可以循环快速变色的模块。查阅了资料，发现有显示字符串的bios中断命令。然后写了一个可以不断刷新变色的一个显示程序。程序如图：

```
again:
    mov cx,3
    sub cx,1
    cmp cx,2
    mov bx,000Ch
    call print
    sub cx,1
    cmp cx,1
    mov bx,000Ah
    call print
    sub cx,1
    cmp cx,0
    mov bx,000Eh
    call print
    jmp again

print:
    push cx
    mov cx,12
    int 10h
    pop cx
    call delayf
    ret

    jmp $
delayf:
    dec word [count]
    jnz delayf
    mov word [count],delay
    dec word [dcount]
    jnz delayf
    mov word [count],delay
    mov word [dcount],ddelay
    ret
```

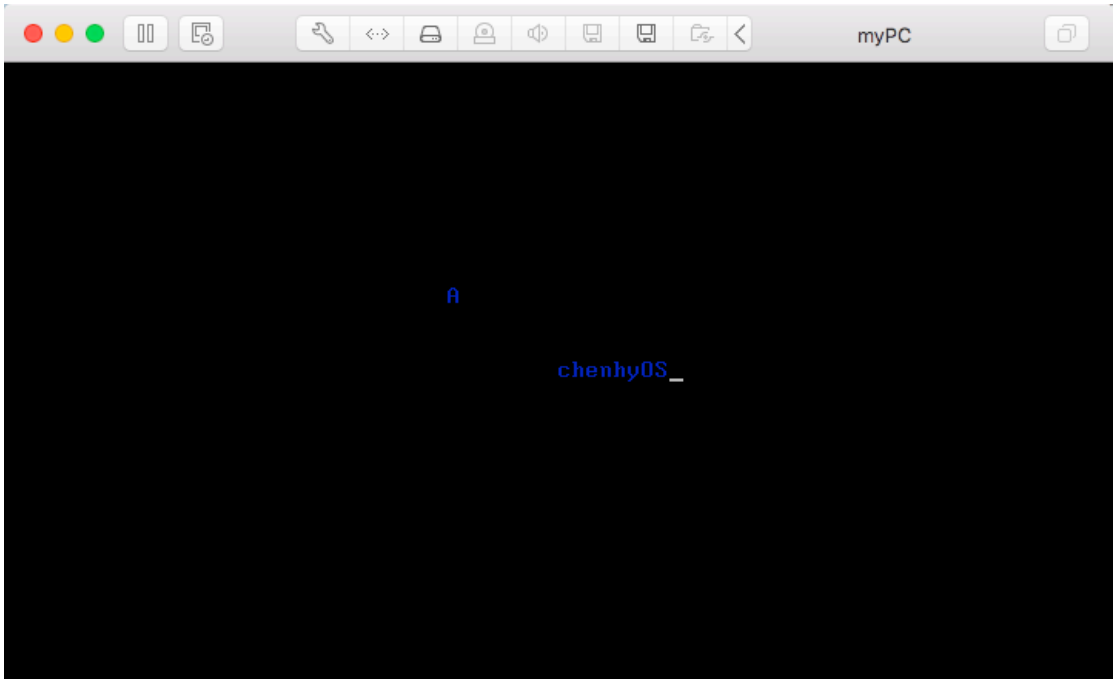
效果(见视频1)插图：




```
print:
    push es
    push bx
    mov ax,BOOTSEG;if es is 0B800H,we can't have the word in the screen
    mov es,ax
    mov ax,msg
    mov bp,ax
    mov ax,01301h
    mov dh,12
    mov dl,40
    mov cx,8
    mov bh,0
    mov bl,byte [color]
    int 10h
    pop bx
    pop es
    ret
```

在这里要注意的是，用这个指令的时候es的段值需要是0B800H，否则会显示不出字符串。所以我们必须先把es的段值压栈保护起来，然后在最后的时候才恢复。

效果见视频2:



五. 实验心得

体会和建议。（**必须认真写，若过于简单，扣分！**）

这次实验是一次以前从来没有进行操作过的实验，不过好在上学期学习了计算机组成原理这一门课，懂了一些计算机开机之后的一些操作顺序。同时其实在这次实验中，我花时间最长的并不是写代码，而是去搭建一个工具链以及其环境，同时去学习一些基础的知识，去深究操作系统开机引导程序的知识。因为我觉得，只有把一个部分的基础打好了，才能把那一个部分真正的做好。

由于这次实验我实在mac OS下完成的，所以老师的很多推荐软件，老师写的脚本呀，推荐的软件都没能用上。所以自己动手，丰衣足食。自己在网上找到了很多软件来代替老师的软件。

然后看到qq群里大家一直在讨论了一个org 7c00H的问题，我就上网查阅了很多资料，在这个问题上面停留了很久，最后才搞明白。在这个查阅的过程中，我了解到了什么是主引导扇区，这里面的程序是如何加载到内存里面执行的。正是在不断的查阅的过程中，我知道了要想调用到数据，段值什么的真的很重要。再者，在写程序的时候，我还要显示字符串，

但是自己并没有向老师说的那样用循环表示，直接显示字符的方法。在查阅资料之后，取而代之的是，使用了中断向量10H来显示。在写函数的时候，我体会到了像C++里面我们写函数一样，也是可以通过call，ret来实现，减少代码量。

在写程序中，有个最大的问题，就是如何将超过512字节的扇区程序通过不断的优化，最后减少到只有512个字节。在这个不断的减少代码，但同时又要确保基本程序不变，这一过程非常的艰辛。但是当你做出来之后，又会非常的有成就感。

在程序中，老师在课堂上要求我们必须要在扇区的511和510字节将这俩个字节修改为55AA，然而在一开始，程序在虚拟机运行的时候，我忘了修改这俩个字节的时候，程序并没有无法引导运行。这让我产生了很大的疑惑，最后在查资料才发现，原来，虚拟机和真正的物理机不同。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55AA. 若存在，则主引导区有效，执行程序。 而虚拟机，即使不满足55AA这个条件，也执行了这段程序。

最后贴上，我的参考资料：

1. **操作80*25彩色字符显示缓冲区**

http://blog.csdn.net/lulipeng_cpp/article/details/8161982

2. 使用BIOS中断显示字符串笔记

http://blog.csdn.net/pdcxs007/article/details/43378229

3. 怎样写一个操作系统（How to wirte a simple OS）

http://blog.csdn.net/magictong/article/details/6164008

4. ORG指令详解

http://blog.csdn.net/yuduoluogongwu/article/details/7359242

5. NASM手册