## 实验目的

* 熟悉 hit-oslab 实验环境；
* 建立对操作系统引导过程的深入认识；
* 掌握操作系统的基本开发过程；
* 能对操作系统代码进行简单的控制，揭开操作系统的神秘面纱。

## 实验内容

此次实验的基本内容是：

1. 阅读《Linux 内核完全注释》的第 6 章，对计算机和 Linux 0.11 的引导过程进行初步的了解；
2. 按照下面的要求改写 0.11 的引导程序 bootsect.s
3. 有兴趣同学可以做做进入保护模式前的设置程序 setup.s。

改写 bootsect.s 主要完成如下功能：

1. bootsect.s 能在屏幕上打印一段提示信息“XXX is booting...”，其中 XXX 是你给自己的操作系统起的名字，例如 LZJos、Sunix 等（可以上论坛上秀秀谁的 OS 名字最帅，也可以显示一个特色 logo，以表示自己操作系统的与众不同。）

改写 setup.s 主要完成如下功能：

1. bootsect.s 能完成 setup.s 的载入，并跳转到 setup.s 开始地址执行。而 setup.s 向屏幕输出一行"Now we are in SETUP"。
2. setup.s 能获取至少一个基本的硬件参数（如内存参数、显卡参数、硬盘参数等），将其存放在内存的特定地址，并输出到屏幕上。
3. setup.s 不再加载 Linux 内核，保持上述信息显示在屏幕上即可。

## 实验报告

有时，继承传统意味着别手蹩脚。x86 计算机为了向下兼容，导致启动过程比较复杂。请找出 x86 计算机启动过程中，被硬件强制，软件必须遵守的两个“多此一举”的步骤（多找几个也无妨），说说它们为什么多此一举，并设计更简洁的替代方案。

1） 计算机上电后会将引导扇区读入0x7c00处，然后为了方便程序执行，要搬运到高地址如0x90000

解决方案：修改硬件，或者采用新方法比如UEFI（UEFI固件DXE阶段运行在64位模式，PEI阶段运行在32位模式）

2） 计算机上电后，中断向量表当到了0x000-0x3ff处，导致载入操作系统时要先载入到其他位置，再移入0地址处

解决方案：BIOS初始化中断向量时放置到较高位置，这样就能直接从0地址处载入操作系统，或者同上，采用UEFI

## 实验步骤

1. 修改 bootsect.s 更改字符串及其长度，运行并显示

|  |
| --- |
| 修改文件 |
|  |
| 运行效果 |
|  |

1. 修改bootsect.s和setup.s显示Now we are in SETUP

|  |
| --- |
| bootsect.s文件 |
|  |
| setup.s文件 |
|  |
| 运行效果 |
|  |

1. 修改 build.c 方便使用

|  |
| --- |
|  |

1. 读取内存大小、光标位置并显示

|  |
| --- |
| 代码 |
| INITSEG = 0x9000  entry \_start  \_start:  mov ah,#0x03  xor bh,bh  int 0x10  mov cx,#25  mov bx,#0x0007  mov bp,#msg2  mov ax,cs  mov es,ax  mov ax,#0x1301  int 0x10  mov ax,cs  mov es,ax  mov ax,#INITSEG  mov ss,ax  mov sp,#0xFF00  mov ax,#INITSEG  mov ds,ax  mov ah,#0x88  int 0x15  mov [2],ax  mov ax,cs  mov es,ax  mov ax,#INITSEG  mov ds,ax  mov ah,#0x03  xor bh,bh  int 0x10  mov cx,#14  mov bx,#0x0007  mov bp,#msg\_memory  mov ax,#0x1301  int 0x10  mov dx,[2]  call print\_hex  mov ah,#0x03  xor bh,bh  int 0x10  mov cx,#2  mov bx,#0x0007  mov bp,#msg\_kb  mov ax,#0x1301  int 0x10  inf\_loop:  jmp inf\_loop  print\_hex:  mov cx,#4  print\_digit:  rol dx,#4  mov ax,#0xe0f  and al,dl  add al,#0x30  cmp al,#0x3a  jl outp  add al,#0x07  outp:  int 0x10  loop print\_digit  ret  print\_nl:  mov ax,#0xe0d ! CR  int 0x10  mov al,#0xa ! LF  int 0x10  ret  msg2:  .byte 13,10  .ascii "NOW we are in SETUP"  .byte 13,10,13,10  msg\_memory:  .byte 13,10  .ascii "Memory Size:"  msg\_kb:  .ascii "KB"  .org 510  boot\_flag:  .word 0xAA55 |
| 运行效果 |
| 2020-05-02 23-36-13 的屏幕截图 |