**实验二**

**实验目的**：

操作系统是一个软件，也需要通过某种机制加载并运行它。在这里我们将通过另外一个更加简单的软件-bootloader来完成这些工作。为此，我们需要完成一个能够切换到x86的保护模式并显示字符的bootloader，为启动操作系统ucore做准备。lab1提供了一个非常小的bootloader 和ucore OS，整个bootloader执行代码小于512个字节，这样才能放到硬盘的主引导扇区中。 通过分析和实现这个bootloader和ucore OS，读者可以了解到：

1.计算机原理

CPU的编址与寻址: 基于分段机制的内存管理

CPU的中断机制

外设：串口/并口/CGA，时钟，硬盘

2.Bootloader软件

编译运行bootloader的过程

调试bootloader的方法

PC启动bootloader的过程

ELF执行文件的格式和加载

外设访问：读硬盘，在CGA上显示字符串

3.ucore OS软件

编译运行ucore OS的过程

ucore OS的启动过程

调试ucore OS的方法

函数调用关系：在汇编级了解函数调用栈的结构和处理过程

中断管理：与软件相关的中断处理

外设管理：时钟

**实验内容：**

**练习 1**：理解通过**make**生成执行文件的过程。（要求在报告中写出对下述问题的回答）

1. 操作系统镜像文件ucore.img是如何一步一步生成的？(需要比较详细地解释Makefile中每 一条相关命令和命令参数的含义，以及说明命令导致的结果)

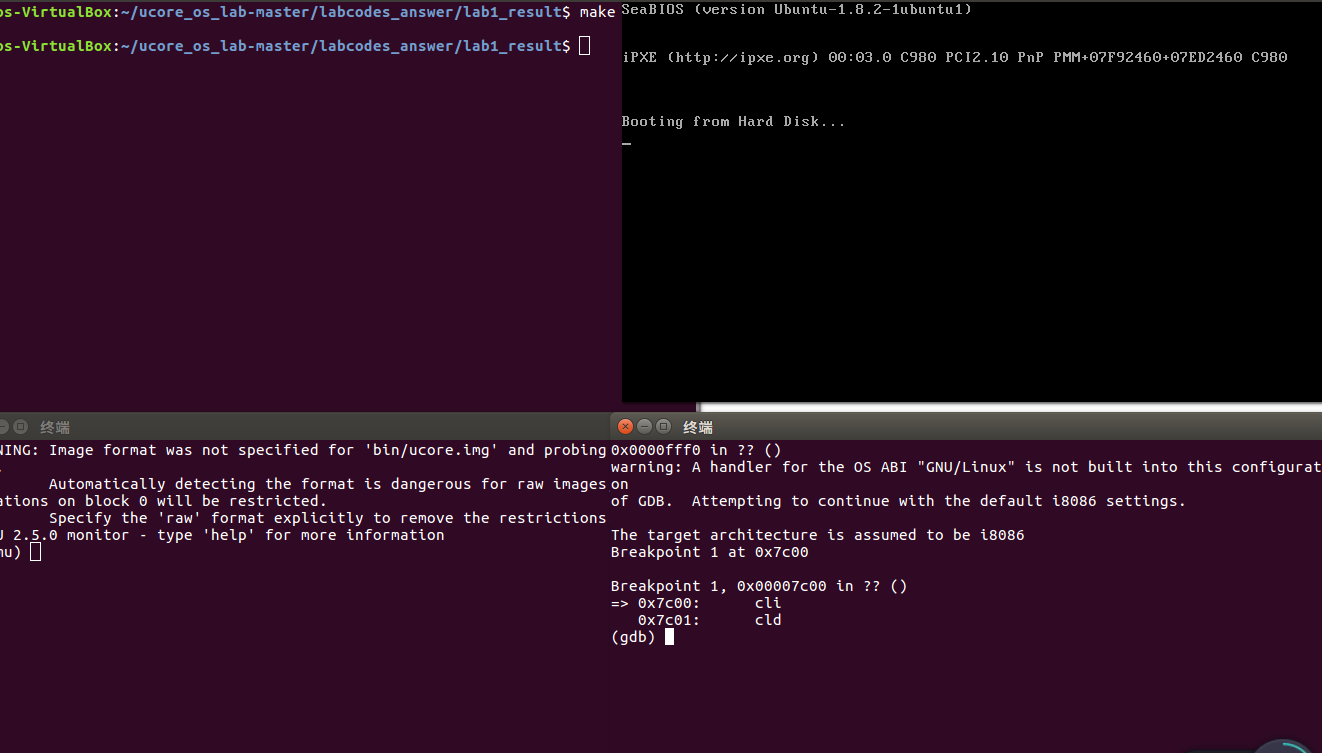
----通过GCC编译器将Kernel目录下的.c文件编译成OBJ目录下的.o文件。

----ld命令根据链接脚本文件kernel.ld将生成的\*.o文件，链接成BIN目录下的kernel文件。

----通过GCC编译器将boot目录下的.c,.S文件以及tools目录下的sign.c文件编译成OBJ目录下的\*.o文件。

----ld命令将生成的\*.o文件，链接成BIN目录下的bootblock文件。

----dd命令将dev/zero, bin/bootblock,bin/kernel 写入到bin/ucore.img



2）一个被系统认为是符合规范的硬盘主引导扇区的特征是什么？

引导扇区的大小为512字节，最后两个字节为标准性结束字节0x55,0xAA，做完这样的检查才能认为是符合规范的磁盘主引导扇区

**练习2：**使用**qemu**执行并调试**lab1**中的软件。（要求在报告中简要写出练习过程）

1. 从CPU加电后执行的第一条指令开始，单步跟踪BIOS的执行。

（1）进入~/moocos/ucore\_lab/labcodes/lab1/bin，即ucore.img所在文件夹。  
（2）输入指令：qemu -S -s -hda ucore.img -monitor stdio

（3）打开gdb。

（4）输入命令，使gdb与qemu通过1234端口进行通信，qemu会停止状态听从gbd命令。

target remote 127.0.0.1:1234

（5）输入命令，单步跟踪。s  
（6）查看BIOS代码

x/2i $pc

1. 在初始化位置0x7c00设置实地址断点,测试断点正常。

（1）输入命令，设置断点。

b \*0x7c00

（2）continue后，测试断点。

c  
x/2i $pc

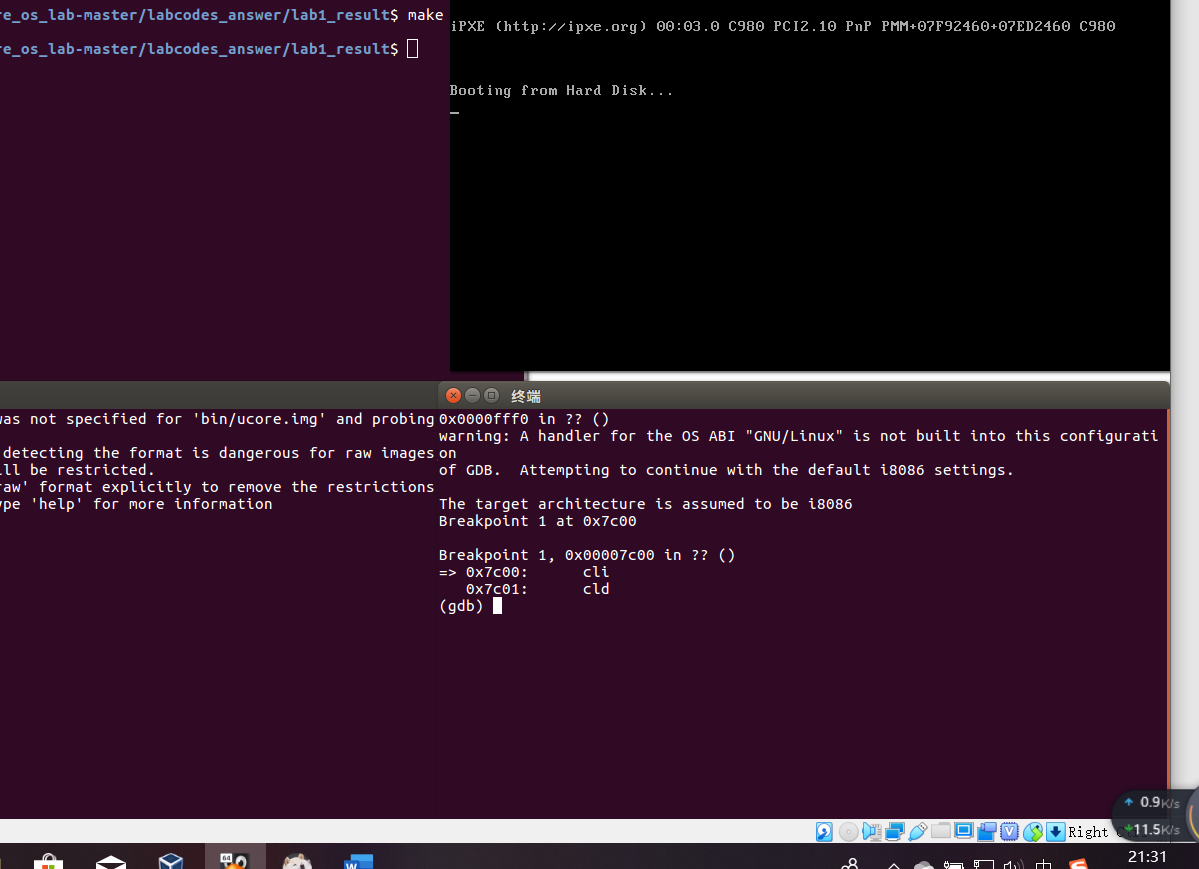
3. 从0x7c00开始跟踪代码运行,将单步跟踪反汇编得到的代码与bootasm.S和

bootblock.asm进行比较。

（1）单步跟踪。输入两次si。  
（2）使用meld对比bootasm.S和bootlock.asm的代码

4. 自己找一个bootloader或内核中的代码位置，设置断点并进行测试。

查看bootloader，选择0x7c06位置。



**练习3：**分析**bootloader**进入保护模式的过程。（要求在报告中写出分析）

从实模式进入保护模式只需要经历三个步骤，第一步加载gdt第二部，，打开A20,第三部置ro为1，开启A20的原因是在保护模式下如果A20开启可以访问连续内存，关闭就只能访问偶数内存。当A20地址线控制禁止时，则程序就像在8086中运行，1M以上的地址是不可访问的。开启A20有3种方法1.通过键盘控制器。2.通过BIOS提供的中断。3.通过系统的I/O端口。初始化GDT：lggt gdtdesc

Mov1 %cr0,%eax

Orl $CR0\_PE\_ON,%eax

Movl %eax,%cr0

将cr0寄存器置1,cro的第0位为1表示处于保护模式