**操作系统实验一**

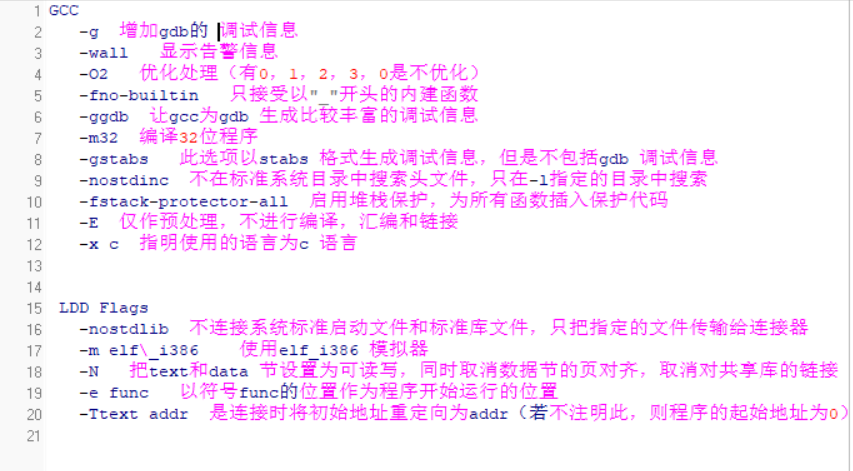
**练习1：理解通过make生成执行文件的过程。（要求在报告中写 出对下述问题的回答）**

**列出本实验各练习中对应的OS原理的知识点，并说明本实验中的实现部分如何对应和体现了 原理中的基本概念和关键知识点。**

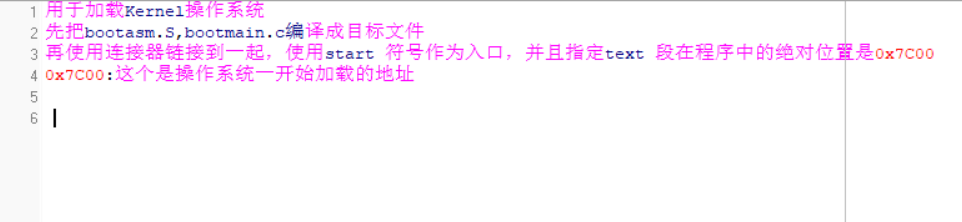
在此练习中，大家需要通过静态分析代码来了解：

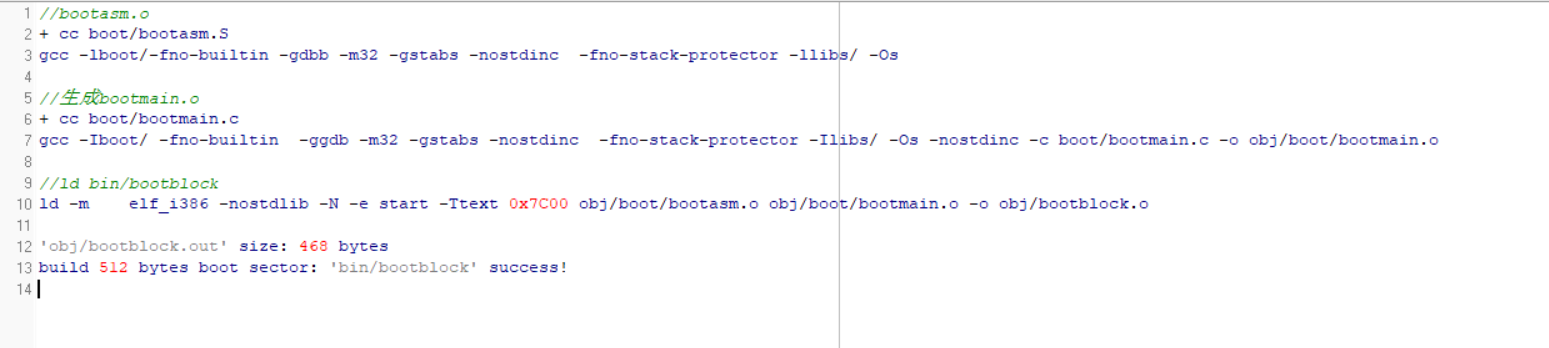
1. 操作系统镜像文件ucore.img是如何一步一步生成的？(需要比较详细地解释Makefile中每 一条相关命令和命令参数的含义，以及说明命令导致的结果)

分析：1.GCC相关编译选项

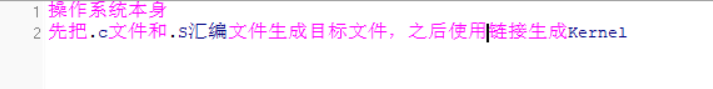


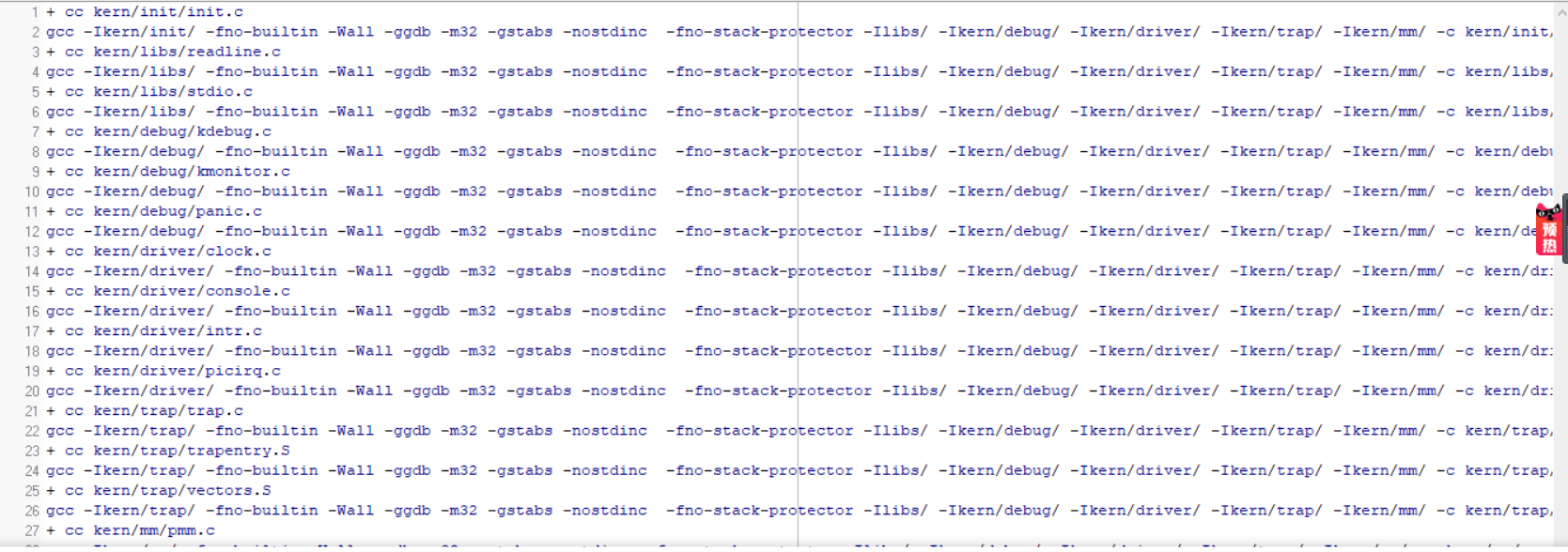
1. 编译bootloader



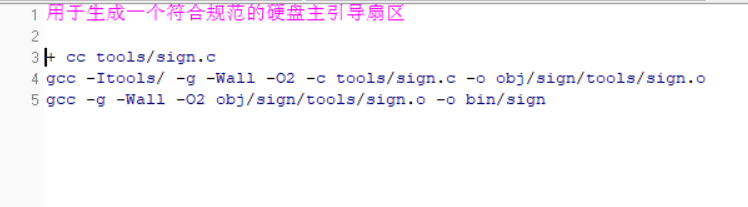


1. 编译Kernel



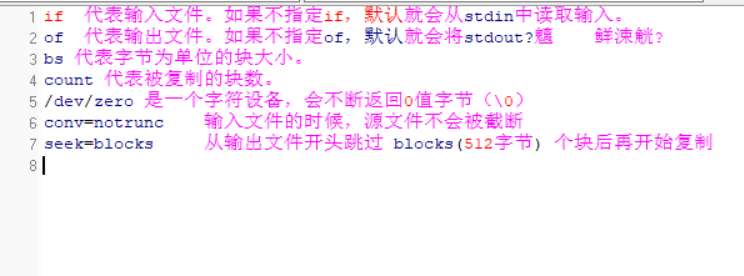


1. 编译sign

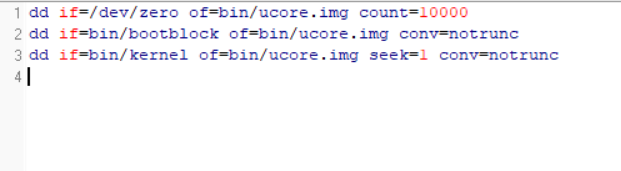


1. 生成ucore.ing

dd -转换和拷贝文件



过程：生成一个空的软盘镜像，然后把bootloader以不截断的方式填充到开始的块中，然后Kernel跳过bootloader所在的块，再填充



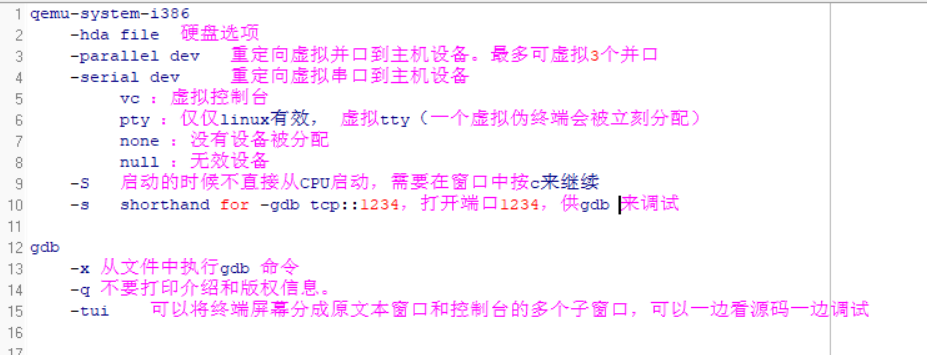
二、 一个被系统认为是符合规范的硬盘主引导扇区的特征是什么？

从sign.c的代码来看，一个磁盘主引导扇区只有512字节，而且第510个字节是0x55，第511个字节是0xAA

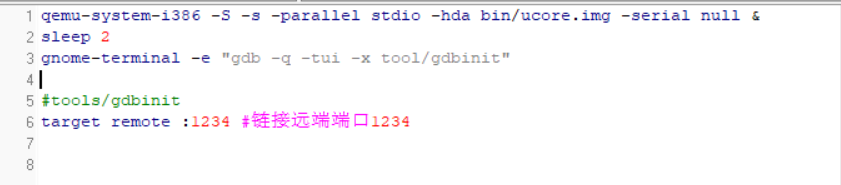
**练习2：使用qemu执行并调试lab1中的软件。（要求在报告中简 要写出练习过程）**

**为了熟悉使用qemu和gdb进行的调试工作，我们进行如下的小练习：**

1. **从CPU加电后执行的第一条指令开始，单步跟踪BIOS的执行。**



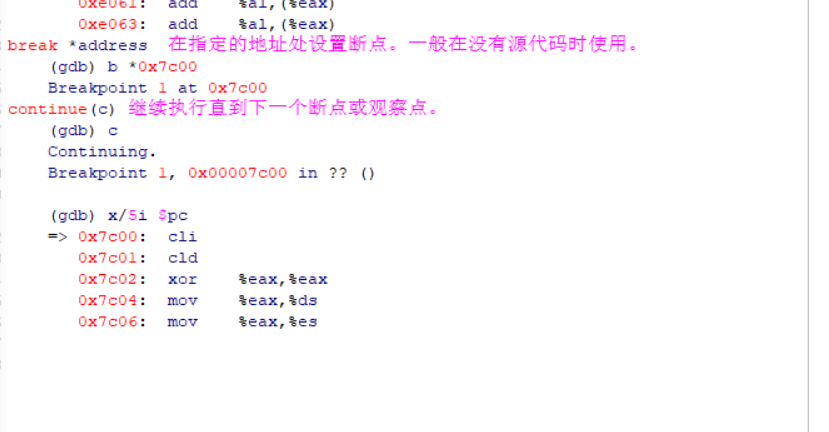
**-S -s是使得qemu在执行第一条指令之前停下来，然后sleep两秒是给qemu充分的时间准备等待连接，然后使用GDB调试工具，-tui提供了代码与命令行分屏查看的界面，tools/gdbinit中存放的是gdb调试**



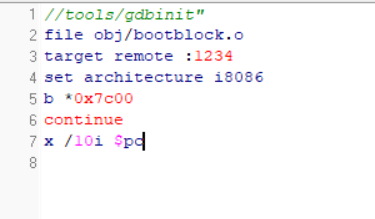
1. **在初始化位置0x7c00设置实地址断点,测试断点正常。**



1. **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。**

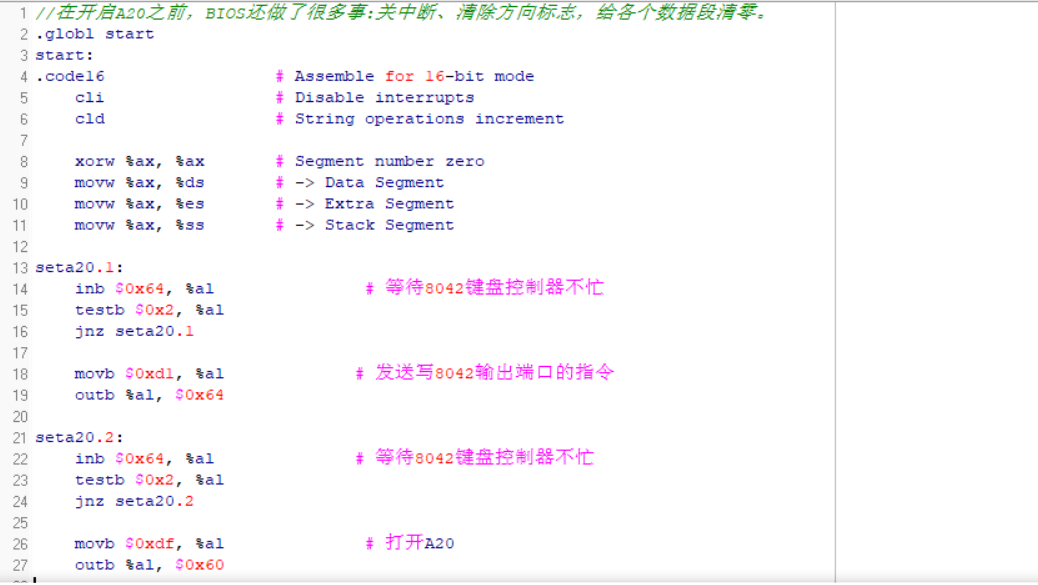


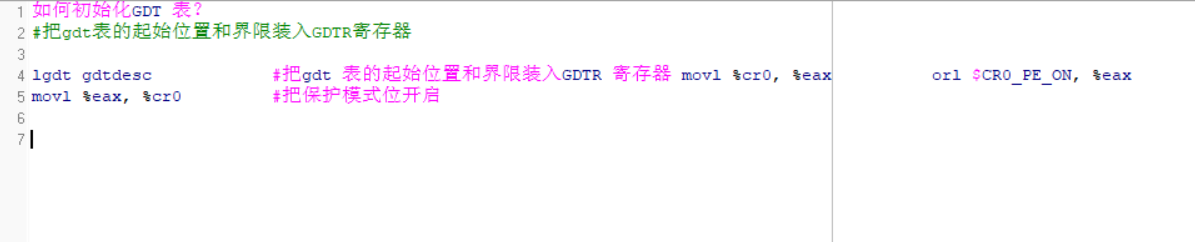
1. **自己找一个bootloader或内核中的代码位置，设置断点并进行测试**。



**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）**

**BIOS将通过读取硬盘主引导扇区到内存，并转跳到对应内存中的位置执行bootloader。请分 析bootloader是如何完成从实模式进入保护模式的。**





**工作在保护模式下。复位PE将返回到实模式工作。**

**此外，gdtdesc指出了全局描述符表在符号gdt 处，**

**上面四句话实现了打开保护模式位。**

**3、如何使能进入保护模式？ 通过长跳转指令**

**ljmp $PROT\_MODE\_CSEG, $protcseg 进入了保护模式。**

**进入保护模式之后还有一个步骤：把所有的数据段寄存器指向上面的GDT描述符表中的数据段（0x10）**