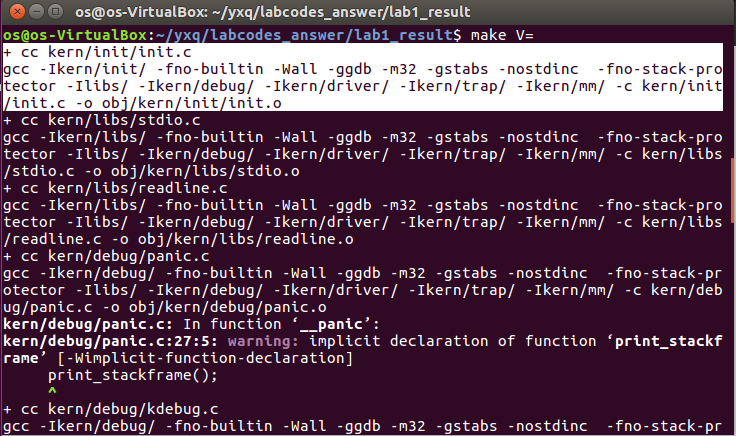
实验一

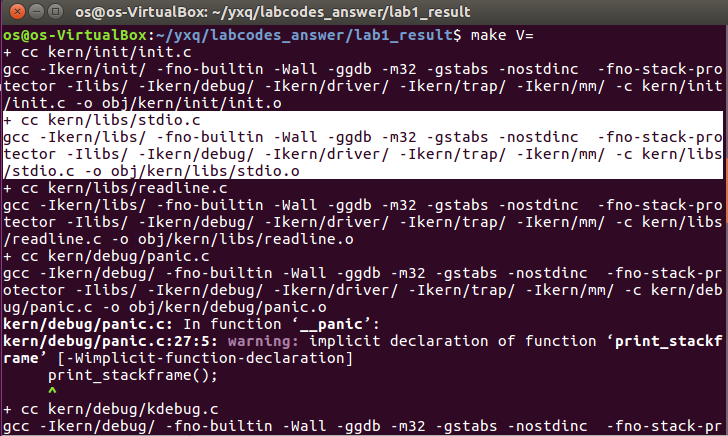
练习1.1 操作系统镜像文件 ucore.img 是如何一步一步生成的？（需要比较详细地解释makefile中每一条相关命令的含义，以及说明命令导致的结果）

生成bin/kernel

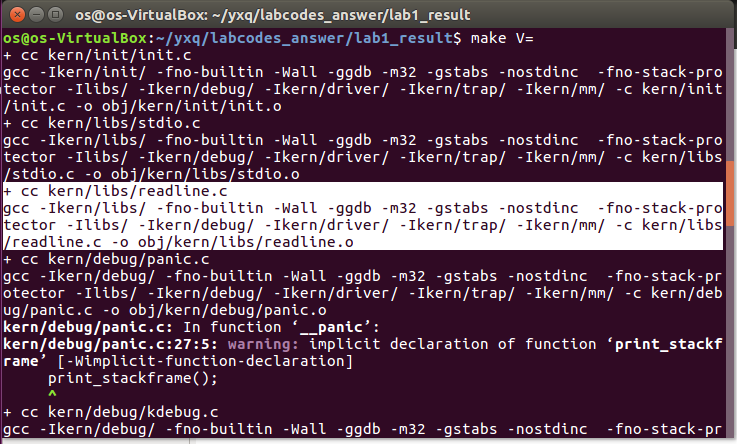
生成init.o

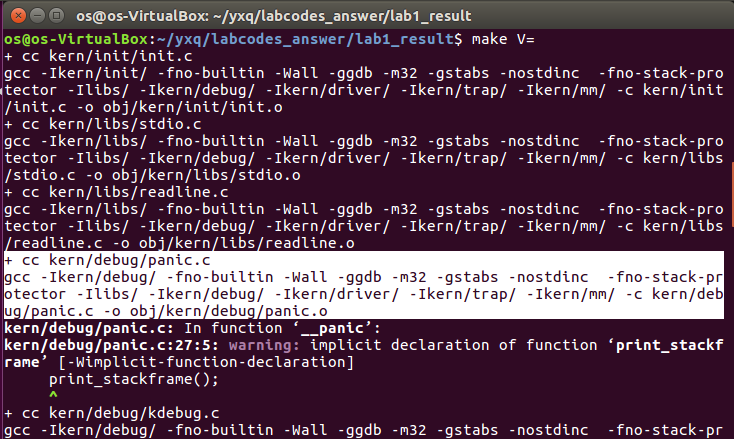


生成stdio.o

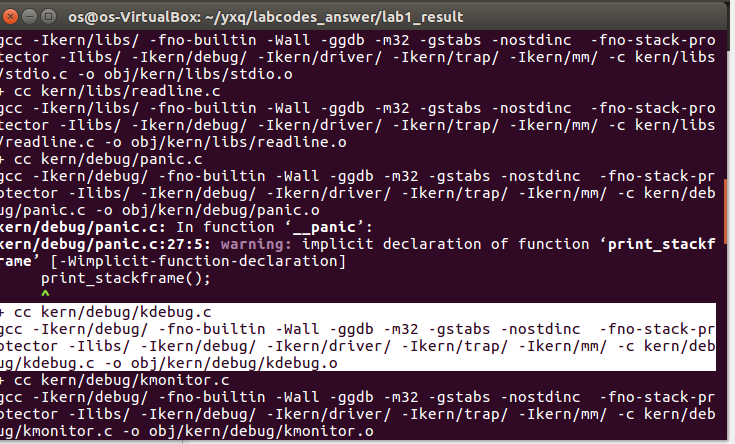


生成readline.o

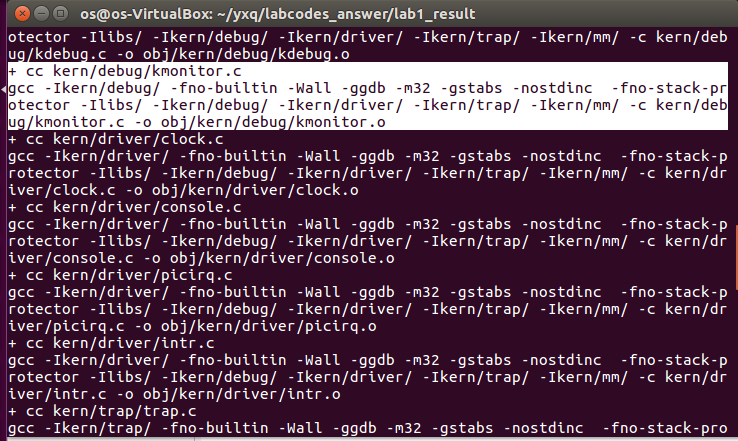
生成panic.o



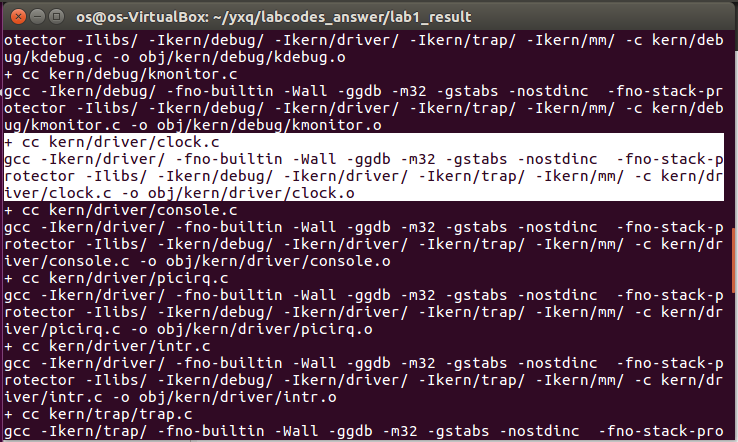
生成kdebug.o



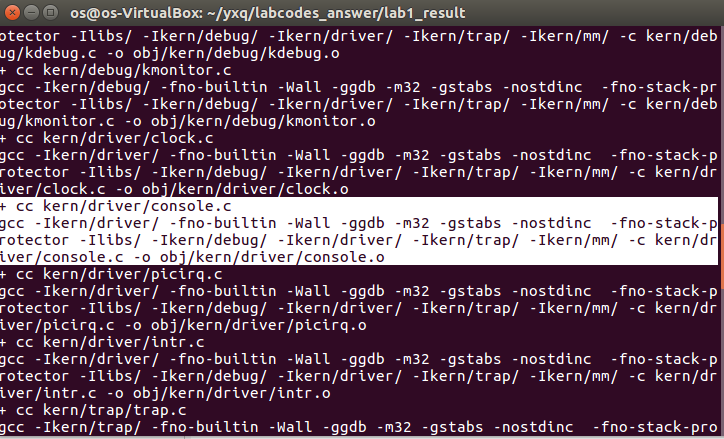
生成kmonitor.o



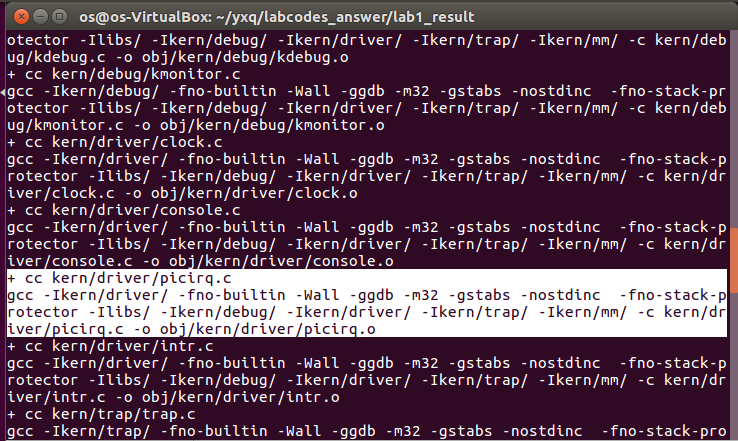
生成clock.o



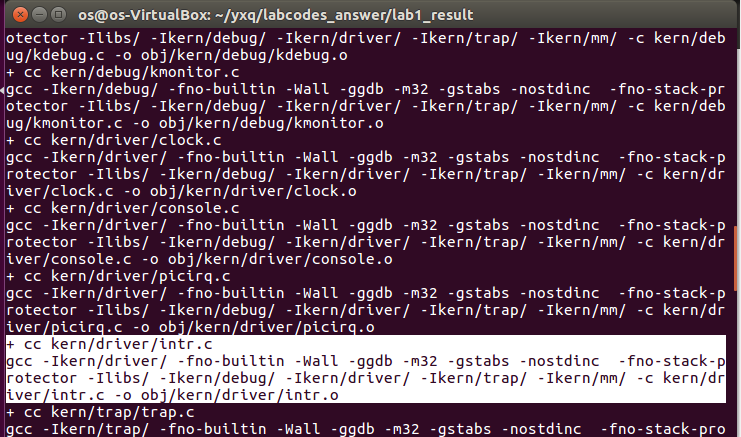
生成console.o



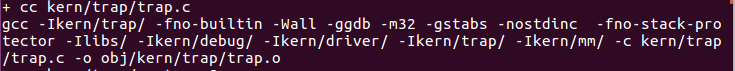
生成picirq.o



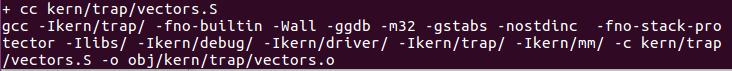
生成intr.o



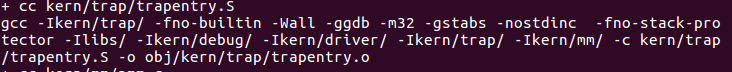
生成trap.o



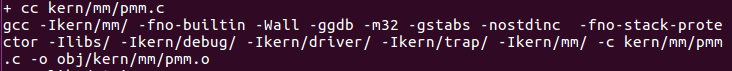
生成vectors.o



生成trapentry.o



生成pmm.o



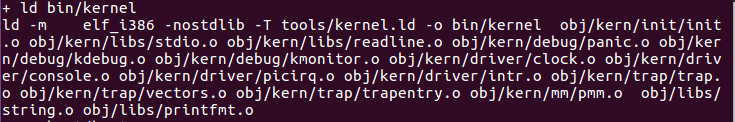
生成string.o



生成printfmt.o



连接.o文件生成bin/kernel



生成bin/bootasm.S



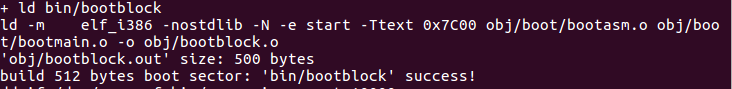
生成bootmain.o



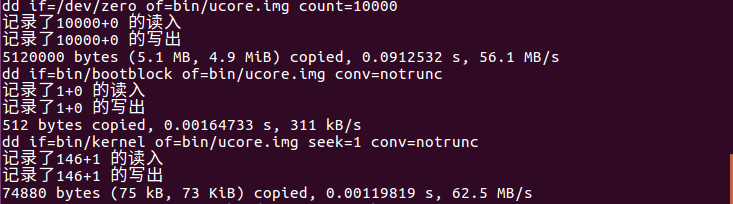
生成bin/sign



连接，生成bootblock



生成ucore.img



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。

2. 在初始化位置0x7c00设置实地址断点,测试断点正常。

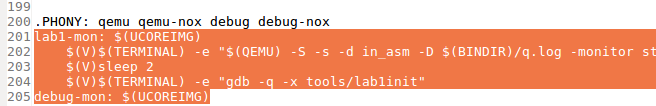
3. 从0x7c00开始跟踪代码运行,将单步跟踪反汇编得到的代码与bootasm.S和 bootblock.asm进行比较。

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。

makefile labi-mon

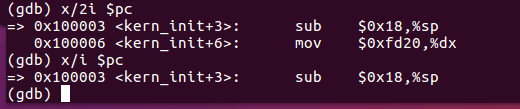
让qume把它执行的指令记录下来放到q.log

和GDB结合调试正在执行的Bootloader

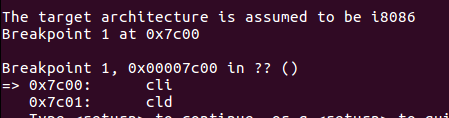


显示当前的十条指令

1.



1. 设置断点



3.单步执行，将gdb的结果与bootblock.asm比较，直到0x7ccf,除一些表达上的差异，仍没有发现明显区别。

练习3：分析bootloader进入保护模式的过程。

为何开启A20，以及如何开启A20

如何初始化GDT表

如何使能和进入保护模式

答：

bootloader首先屏蔽所有中断，之后将段寄存器清零，打开A20地址线，加载

GDT的基地址，切换到保护模式，跳转到32位代码，在32位代码中，bootloader重新设置保护模式下的段寄存器，然后设置栈顶指针，之后跳转到C代码。