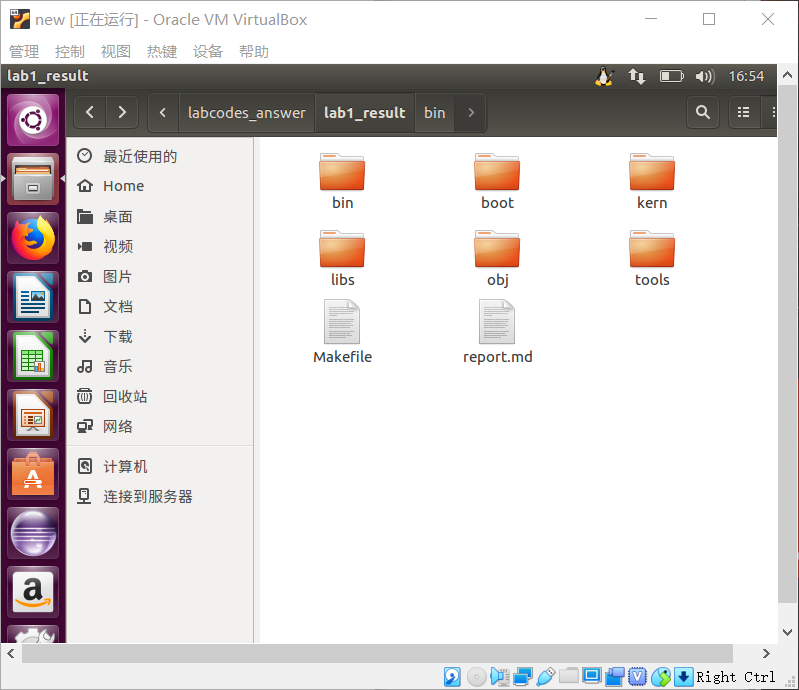
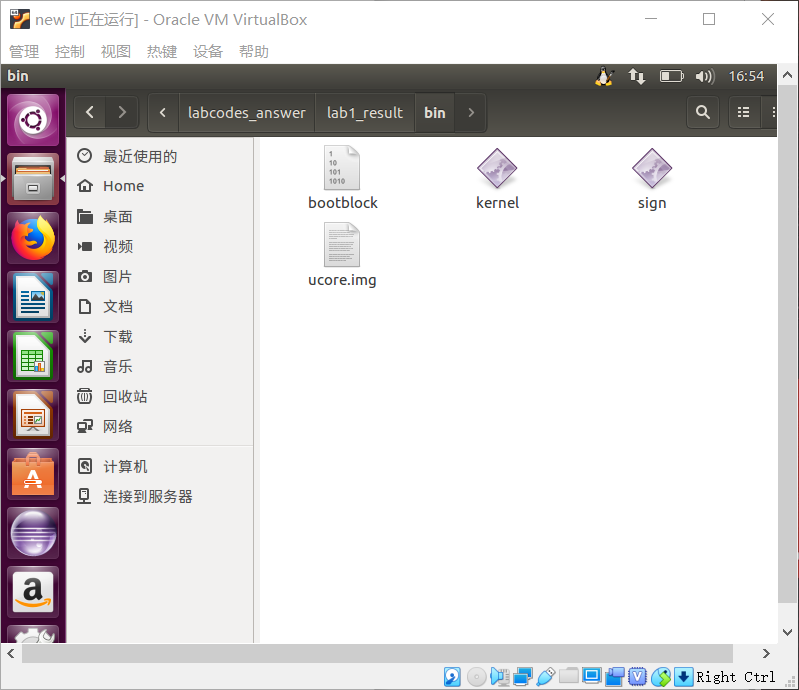
操作系统第1次实验报告

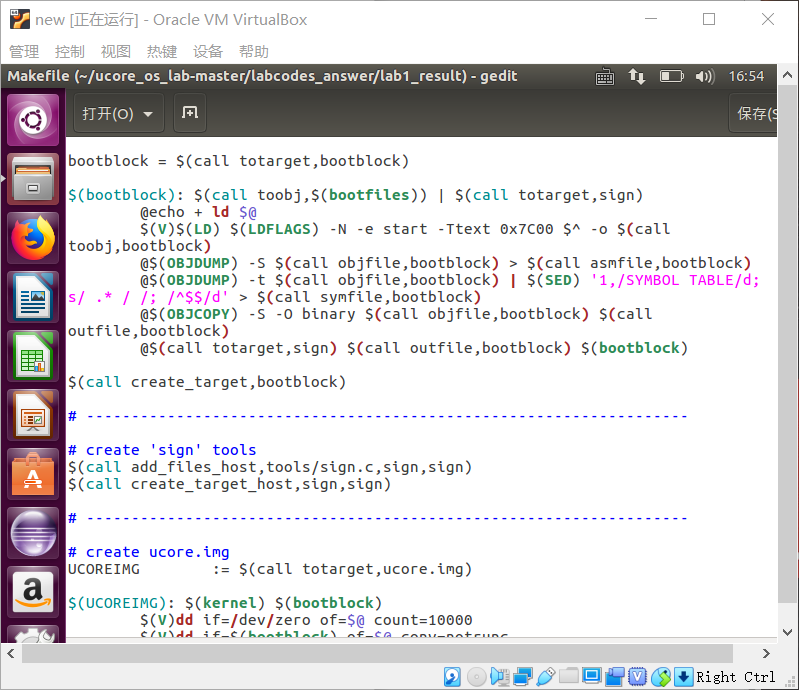
171491106 安南

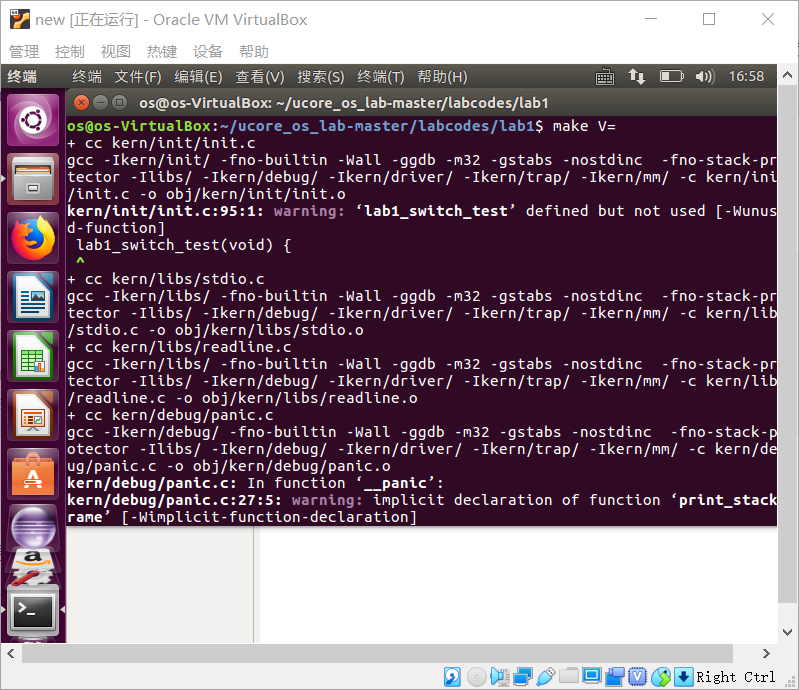
练习1：理解通过make生成执行文件的过程。（要求在报告中写 出对下述问题的回答）

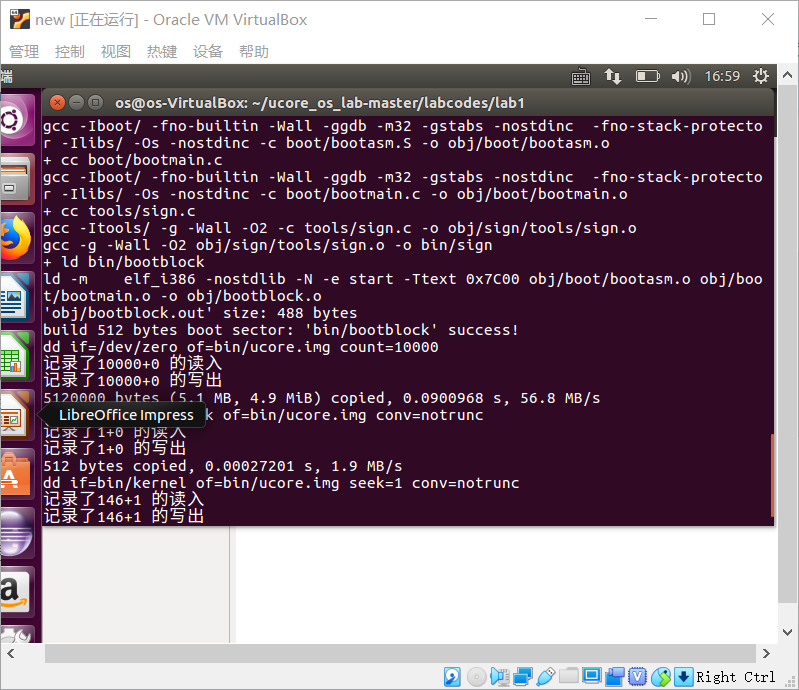
1. 操作系统镜像文件ucore.img是如何一步一步生成的？(需要比较详细地解释Makefile中每 一条相关命令和命令参数的含义，以及说明命令导致的结果)











实现编译的主题：UCOREIMG :=$(call totarget,ucore.img)

$表示着依赖，该主体依赖着kernel，bootlock等其他文件

dd表示着连接文件，dev/zero表示着空白

$(V)dd if=/dev/zero of=$@ count=10000 表示着申请大小为1000的空间，为ucore.img做出准备，对应着dd if=/dev/zero of=bin/ucore.img count=1000

$(V)dd if=$(bootblock) of=$@ conv=notrunc 表示将bootblock放入ucore.img中,对应着dd if=bin/bootblock of=bin/ucore.img conv=notrumc

$(V)dd if=$(kernel) of=$@ seek=1 conv=notrunc 表示将kernel放入ucore.img中,对应着 dd if=bin/kernel of=bin/ucore.img seek=1 conv=notrumc

@$(OBJDUMP) -S $(call objfile,bootblock) > $(call asmfile,bootblock)对应着编译汇编语言文件

I386-elf-gcc -Iboot/ -fno-builtin -Wall -ggdb -m32 -gstabs -nostdinc -fno-stack-protector -Ilibs/ -Os -nostdinc -c boot/bootasm.S -o obj/boot/bootasm.o

@$(OBJDUMP) -t $(call objfile,bootblock) | $(SED) ‘1,SYMBDL TABLE/D; s/ .\*/ /;/^$$d’>$对映着编译C语言文件

I386-elf-gcc -Iboot/ -fno-builtin -Wall -ggdb -m32 -gstabs -nostdinc -fno-stack-proctector -Ilibs/ -Os -nostdinc -c boot/bootmain.c -o obj/boot/bootmain.o

build 512 bytes boot sector:’bin/bootblock’ success!表示着UCOREIMG所依赖的boot block模块编译成功

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）

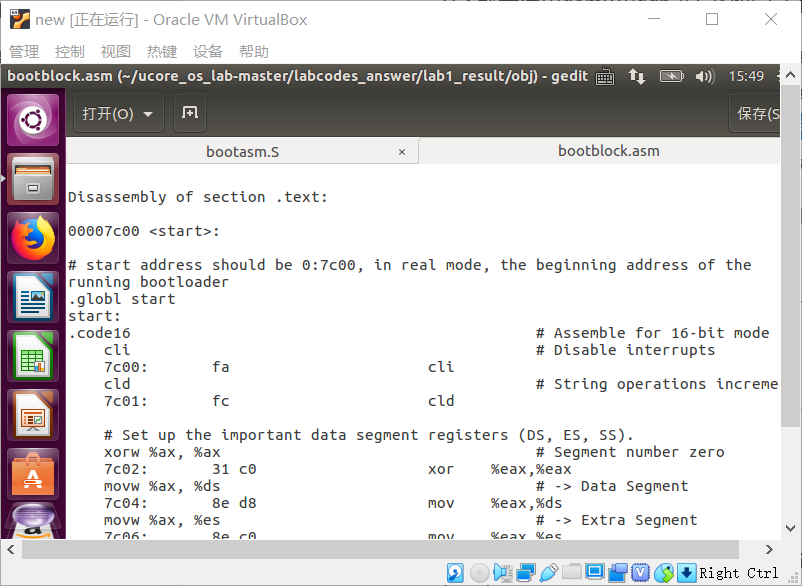
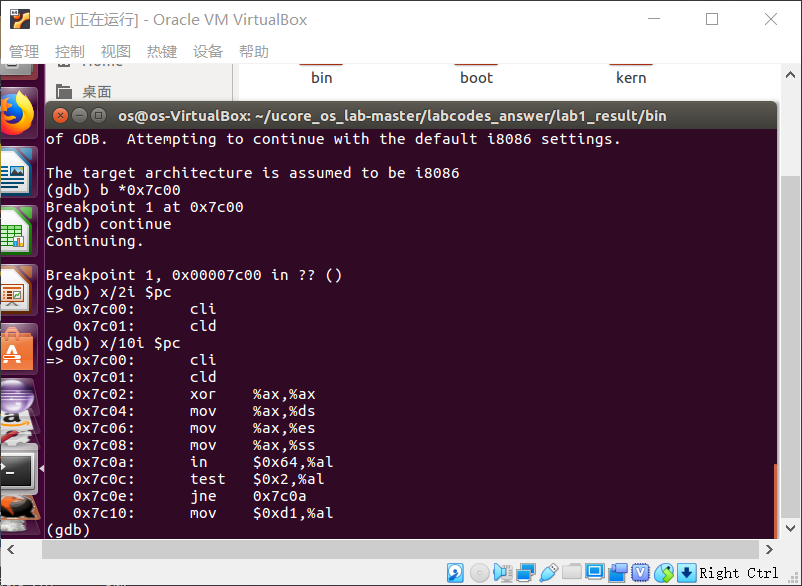
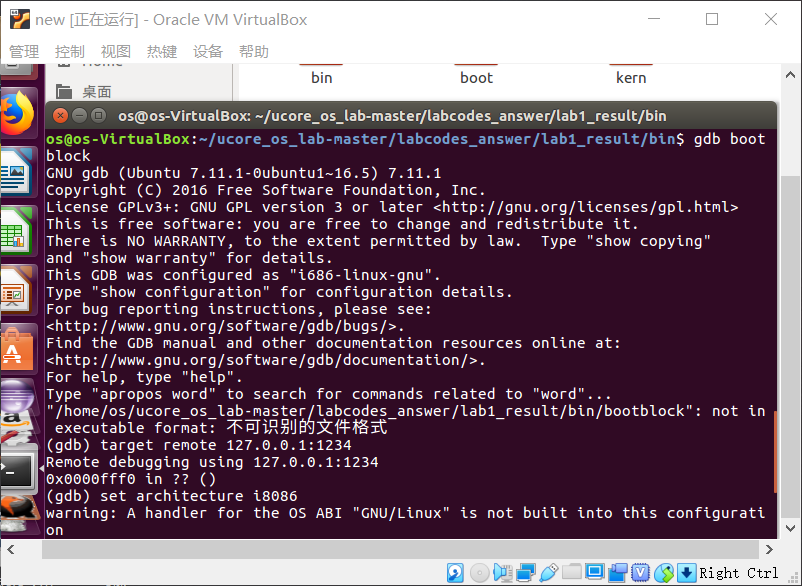
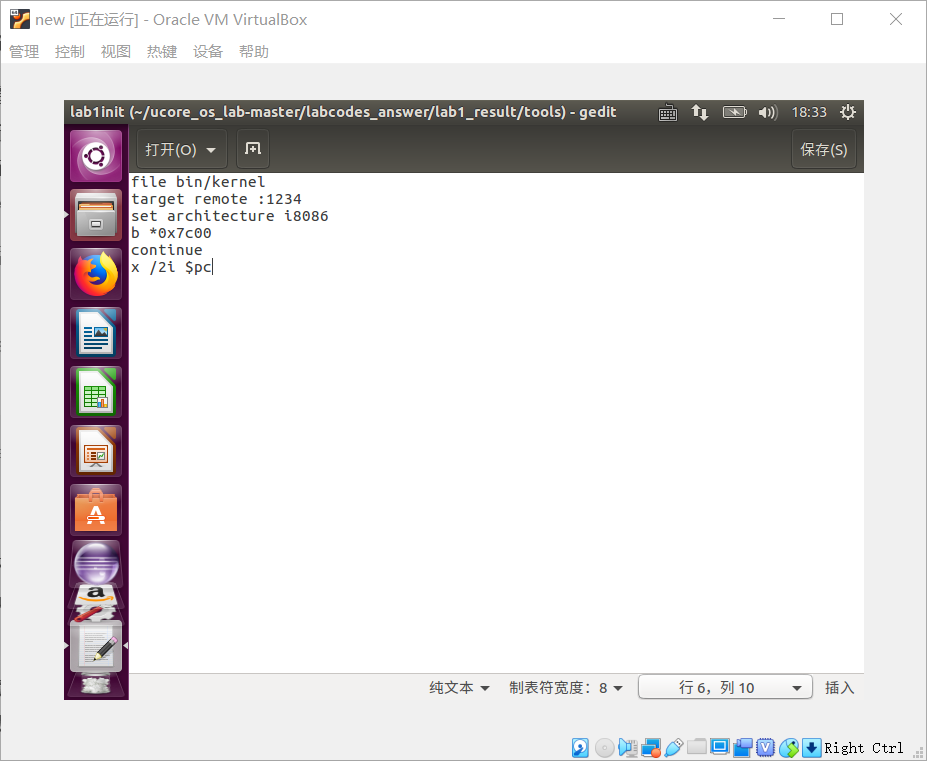
为了熟悉使用qemu和gdb进行的调试工作，我们进行如下的小练习：

1. 从CPU加电后执行的第一条指令开始，单步跟踪BIOS的执行。

2. 在初始化位置0x7c00设置实地址断点,测试断点正常。

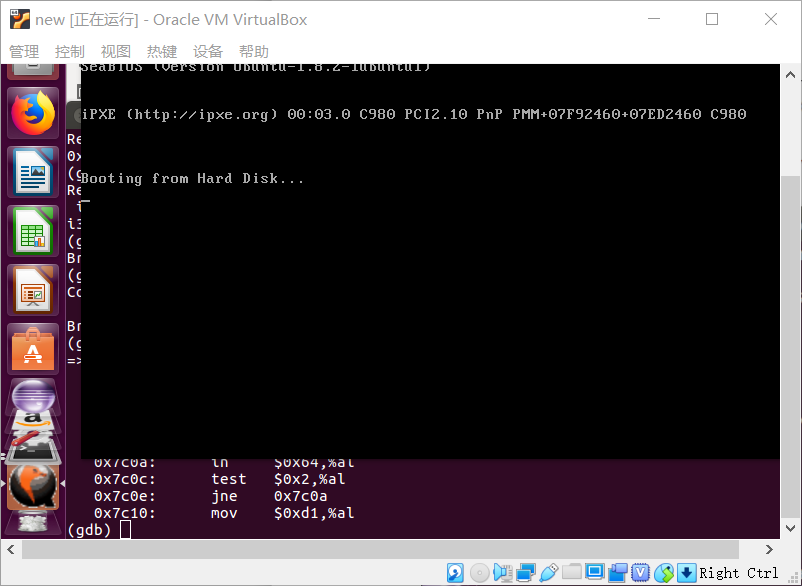
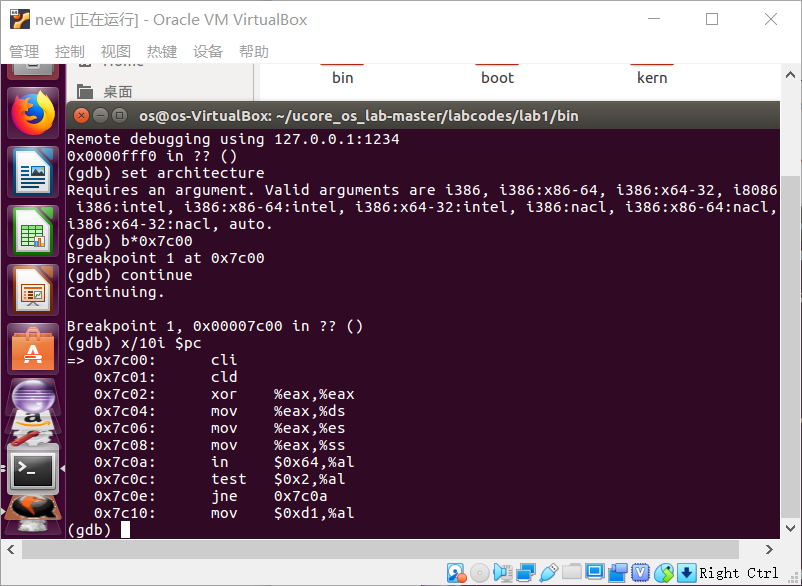
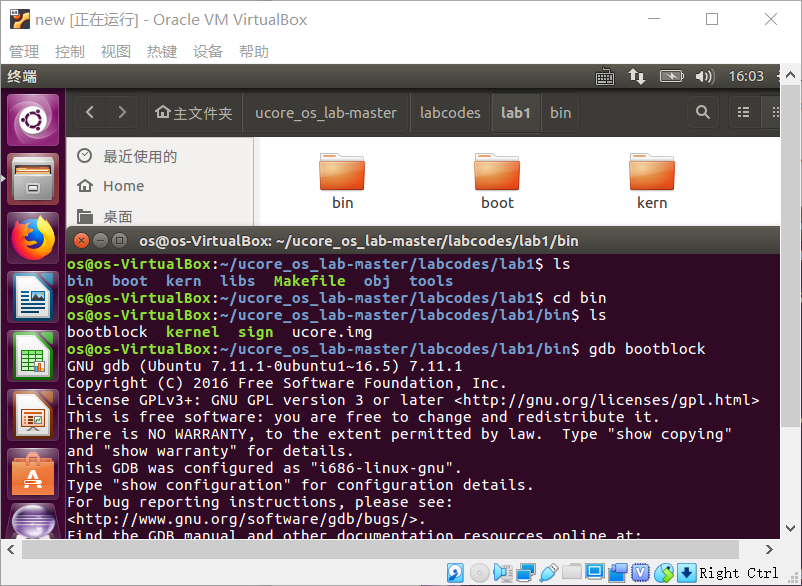
3. 从0x7c00开始跟踪代码运行,将单步跟踪反汇编得到的代码与bootasm.S和 bootblock.asm进行比较。

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



bootasm.S中的指令与单步跟踪反汇编得到的代码一一对应。

bootblock.asm中的指令与单步跟踪反汇编得到的代码隔行队形，是汇编语言源程序的扩展，看起来更接近机器语言



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。

提示：需要阅读小节“保护模式和分段机制”和lab1/boot/bootasm.S源码，了解如何从实模式 切换到保护模式，需要了解：

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

