# 171491208 魏家富

# 操作系统实验一

### 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

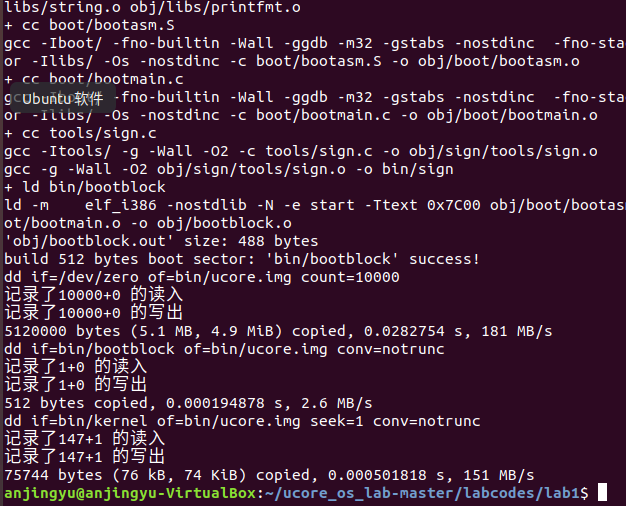
和ucore OS，整个bootloader执行代码小于512个字节，这样才能放到硬盘的主引导扇区中。

实验内容：

lab1中包含一个bootloader和一个OS。这个bootloader可以切换到X86保护模式，能够读磁盘

并加载ELF执行文件格式，并显示字符。而这lab1中的OS只是一个可以处理时钟中断和显示字符的幼儿园级别OS

练习1：理解通过make生成执行文件的过程。

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) 能

显示了想要生成uore.img需要先生成bootlock

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？

如何调试Makefile

当执行make时，一般只会显示输出，不会显示make到底执行了哪些命令。

如想了解make执行了哪些命令，可以执行：

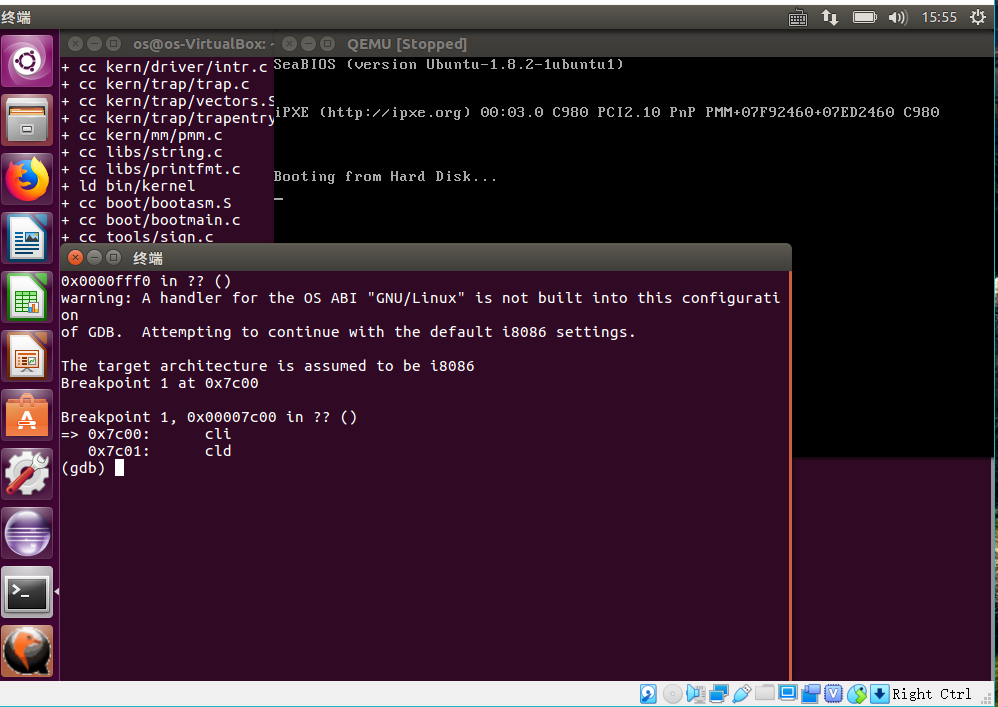
$ make "V

练习2：使用qemu执行并调试lab1中的软件。

查看labcodes\_answer/lab1\_result/tools/lab1int文件

代码：$ cd labcodes\_answer/lab1\_result/

$ make lab1-mon



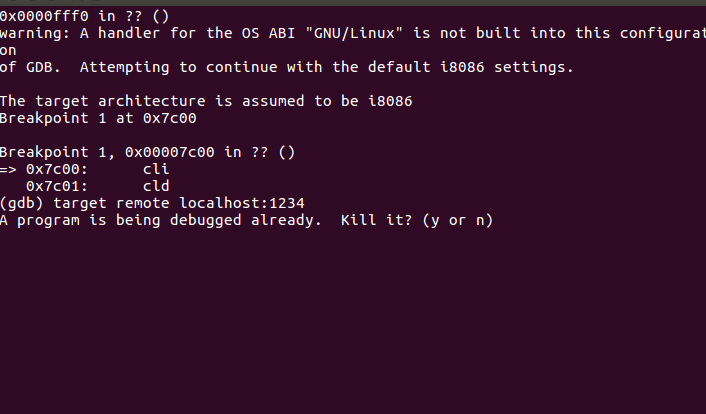
补充材料： 我们主要通过硬件模拟器qemu来进行各种实验。在实验的过程中我们可能会遇上

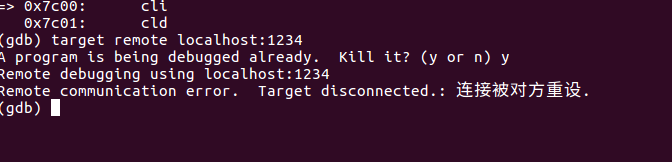
各种各样的问题，调试是必要的。qemu支持使用gdb进行的强大而方便的调试。所以用好

qemu和gdb是完成各种实验的基本要素。

默认的gdb需要进行一些额外的配置才进行qemu的调试任务。qemu和gdb之间使用网络端口

1234进行通讯。在打开qemu进行模拟之后，执行gdb并输入target remote localhost:1234





即可连接qemu，此时qemu会进入停止状态，听从gdb的命令。

另外，我们可能需要qemu在一开始便进入等待模式，则我们不再使用make qemu开始系统的

运行，而使用make debug来完成这项工作。这样qemu便不会在gdb尚未连接的时候擅自运行

了。

gdb的地址断点

在gdb命令行中，使用b \*[地址]便可以在指定内存地址设置断点，当qemu中的cpu执行到指定

地址时，便会将控制权交给gdb。

关于代码的反汇编

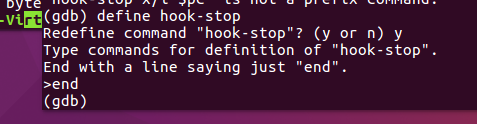
有可能gdb无法正确获取当前qemu执行的汇编指令，通过如下配置可以在每次gdb命令行前强

制反汇编当前的指令，在gdb命令行或配置文件中添加：

define hook-stop

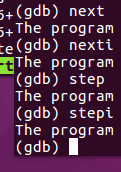
x/i $pc

end



gdb的单步命令

在gdb中，有next, nexti, step, stepi等指令来单步调试程序



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）

·为何开启A20，以及如何开启A20

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

·如何初始化GDT表

当 A20 地址线控制禁止时，则程序就像在 8086 中运行，1MB 以上的地是不可访问的。在保护模式下 A20 地址线控制是要打开的。为了使能所有地址位的寻址能力,必须向键盘控制器 8042 发送一个命令。键盘控制器 8042 将会将它的的某个输出引脚的输出置高电平,作为 A20 地址线控制的输入。一旦设置成功之后,内存将不会再被绕回(memory wrapping),这样我们就可以寻址 intel 80286 CPU 支持的 16M 内存空间，或者是寻址 intel 80386 以上级别 CPU 支持的所有 4G 内存空间了。

·如何使能和进入保护模式

gdt:  
SEG\_NULLASM

SEG\_ASM(STA\_X|STA\_R, 0x0, 0xffffffff)   
SEG\_ASM(STA\_W, 0x0, 0xffffffff)