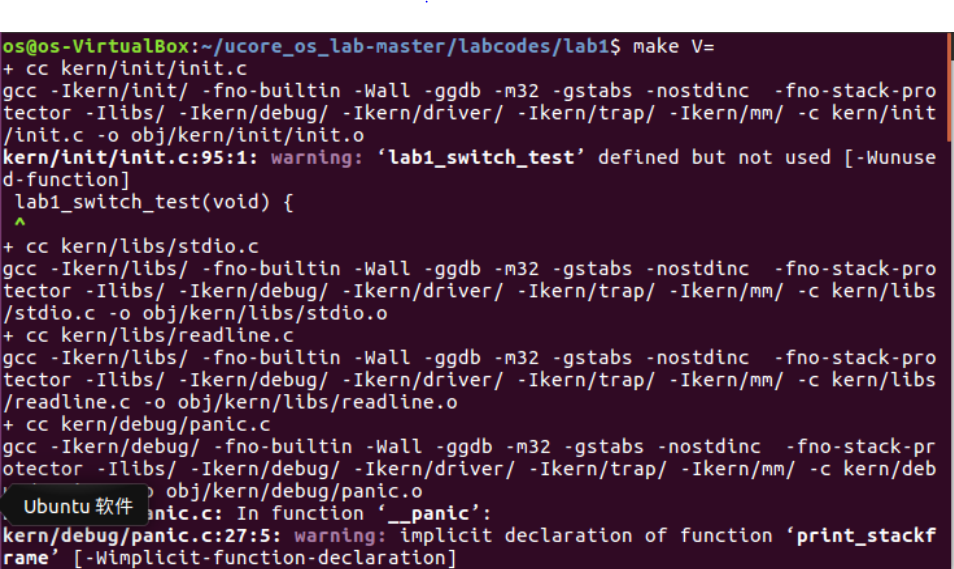
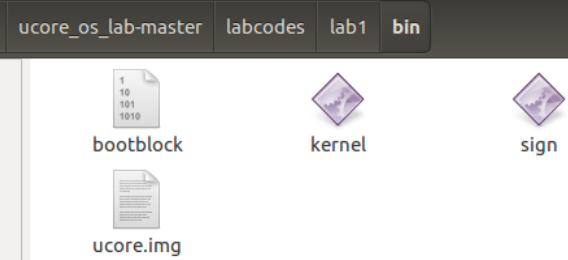
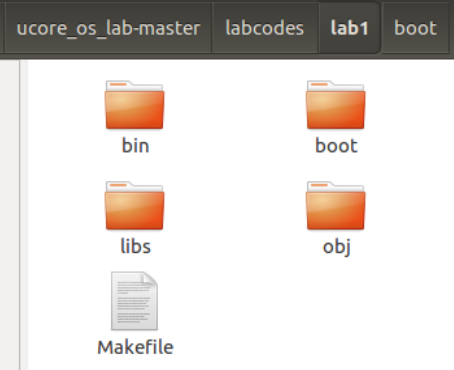
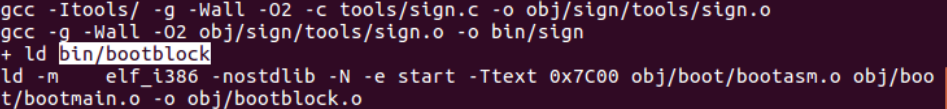
练习1、

截图:

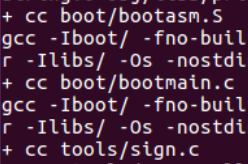


操作系统镜像文件ucore.img是如何一步一步生成？

可知需要生成ucore.img首先得生成bootblock文件

C:\Users\Administrator\AppData\Roaming\Tencent\Users\1366212870\QQ\WinTemp\RichOle\~PU@WKW550YJJWG1T0%V98J.png

生成block之前需要先生成bootasm.o和bootmain.o两个文件，以及一个sign.c文件



这些文件由这些命令生成

C:\Users\Administrator\AppData\Roaming\Tencent\Users\1366212870\QQ\WinTemp\RichOle\M}L2V3$)@46KECZAOCY5INQ.png

生成bootblock，elf\_i386表示生成elf头，0x7c00为程序的入口

C:\Users\Administrator\AppData\Roaming\Tencent\Users\1366212870\QQ\WinTemp\RichOle\1$SBFTWC[OY$7)23FCDXF(S.png

说明bootblock文件的大小为488byte

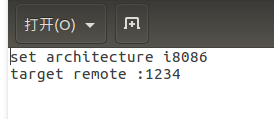
C:\Users\Administrator\AppData\Roaming\Tencent\Users\1366212870\QQ\WinTemp\RichOle\W[319WOCMUC}GW$RRDN$IA5.png

可知需要512byte大小才能生成成功，需要增加文件大小直到512byte

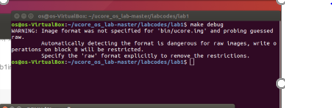
C:\Users\Administrator\AppData\Roaming\Tencent\Users\1366212870\QQ\WinTemp\RichOle\1QM]MMZD]Y7N%`HMO76JJLE.png

可知确定已创建的ucore.img文件是否在bin目录下并符合kernel of的要求

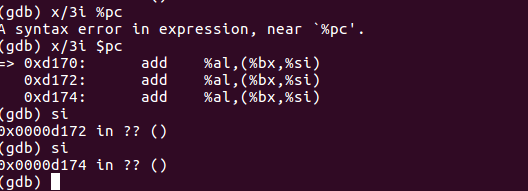
练习2、



修改tools/gbdbinit文件



在文件目录下打开终端，输入make dubug



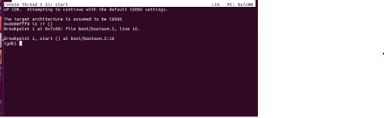
单步跟踪执行代码，得出反汇编的汇编语言

图片包含 屏幕截图

描述已自动生成

修改tools/gbdbinit文件

继续在lab1文件中打开终端，输入make debug运行指令



单步执行，输入si查看bios执行过程中cs和pc地址变化，x/ni$pc反汇编得到汇编语言

图片包含 屏幕截图

描述已自动生成

可以知道0x7c00的程序地址代码与Bootasm.s.一样，反汇编代码也一样

练习3、

a为何开启A20,以及如何开启A20

在i8086时代，CPU的数据总线是16bit，地址总线是20bit，寄存器是16bit，因此CPU只能访问1MB以内的空间。因为数据总线和寄存器只有16bit，如果需要获取20bit的数据, 我们需要做一些额外的操作，8086采用偏移4位的方法，使之能访问20bit的数据。理论上，20bit的地址可以访问1MB的内存空间(0x00000 - (2^20 - 1 = 0xFFFFF))。但在实模式下, 这20bit的地址理论上能访问从0x00000 - (0xFFFF0 + 0xFFFF = 0x10FFEF)的内存空间。也就是说，理论上我们可以访问超过1MB的内存空间，但越过0xFFFFF后，地址又会回到0x00000。上面这个特征在i8086中是没有任何问题的(因为它最多只能访问1MB的内存空间)，但到了i80286/i80386后，CPU有了更宽的地址总线，数据总线和寄存器后，这就会出现一个问题： 在实模式下, 我们可以访问超过1MB的空间，但我们只希望访问1MB以内的内存空间。为了解决这个问题， CPU中添加了一个可控制A20地址线的模块，通过这个模块，我们在实模式下将第20bit的地址线限制为0，这样CPU就不能访问超过1MB的空间了。进入保护模式后，我们再通过这个模块解除对A20地址线的限制，这样我们就能访问超过1MB的内存空间了。默认情况下，A20地址线是关闭的(20bit以上的地址线限制为0)，因此在进入保护模式(需要访问超过1MB的内存空间)前，我们需要开启A20地址线(20bit以上的地址线可为0或者1)。

如何初始化GDT表

从bootasm.s可以看出这里所有GDT表项初始化为全段,除了空段

如何使能和进入保护模式

1. 等待8086 Input buffer为空;

2. 发送Write 8086 Output Port (P2)命令到8086 Input buffer;

3. 等待8086 Input buffer为空;

4. 将8086 Output Port(P2)得到字节的第2位置1,然后写入8086Input buffer;