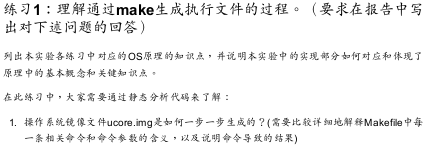
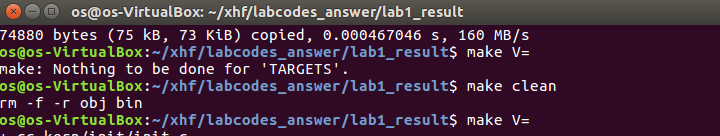
**实验一**

**练习一：**

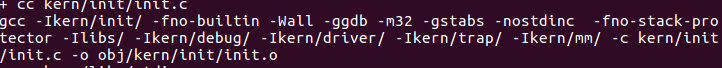


1. 操作系统镜像文件 minios.img 是如何一步一步生成的?(需要比较详细地解释 Makefile中每一条相关命令和命令参数的含义,以及说明命令导致的结果)
2. 进入指定路径，用make v= 指令查看程序具体运行情况（如下）

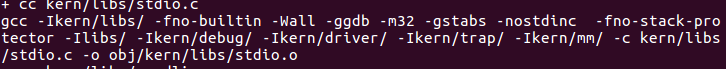


1: 生成bin/kernel

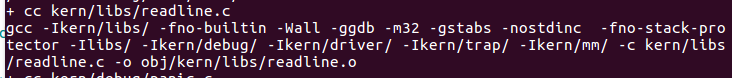
生成init.o



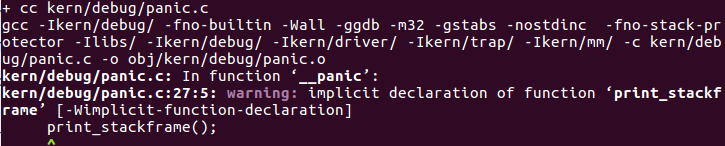
生成stdio.o



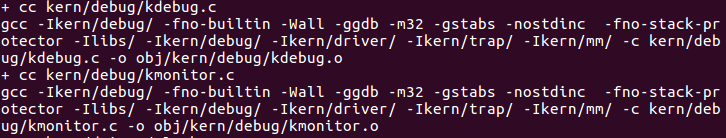
生成readline.o



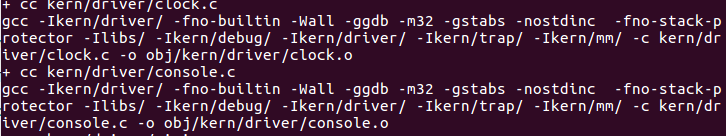
生成panic.o



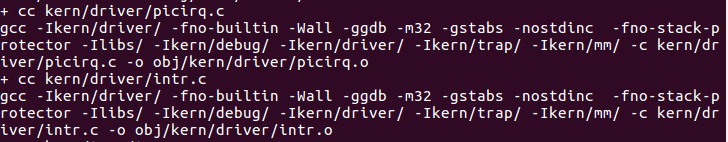
生成kdebug.o及 生成kmonitor.o



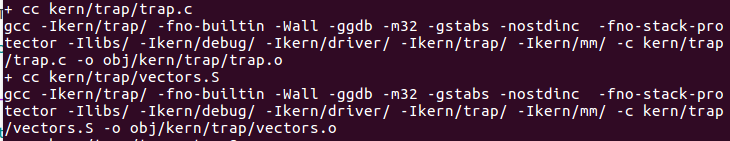
生成clock.o及生成console.o



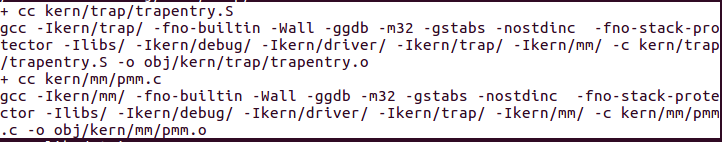
生成picirq.o及 生成intr.o



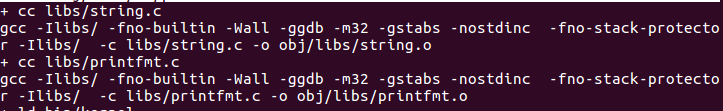
生成trap.o及 生成vector.o



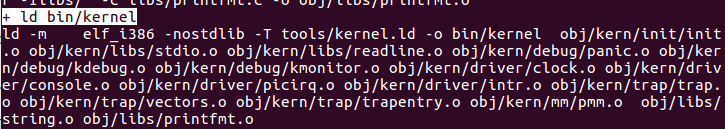
生成trapentry.o及 生成pmm.o



生成string.o及 生成printfmt.o

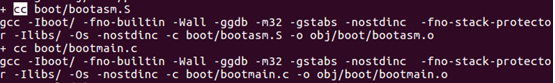


连接.o文件生成bin/kernel



2: 生成bin/bootblock

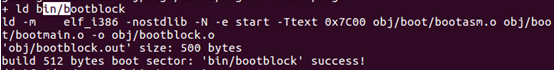
生成bootasm.o 及生成bootmain.o



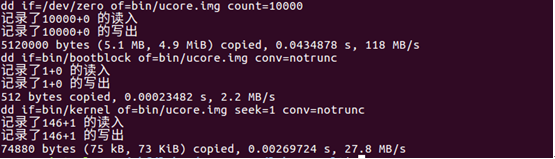
3: 生成bin/sign



连接,生成bootblock



4: 生成ucore.img



其中：

 -g  是为了gdb能够对程序进行调试

 -Wall  生成警告信息

-ggdb 让gcc 为gdb生成比较丰富的调试信息

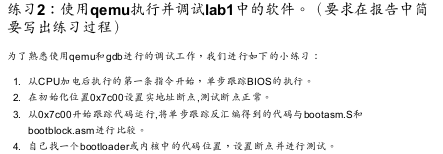
-m32  编译32位程序

 -gstabs  此选项以stabs格式声称调试信息,但是不包括gdb调试信息

-nostdinc  不在标准系统目录中搜索头文件,只在-I指定的目录中搜索

**练习二：**

使用qemu执行并调试lab1中的软件



1. 进行如下操作

set architecture i8086

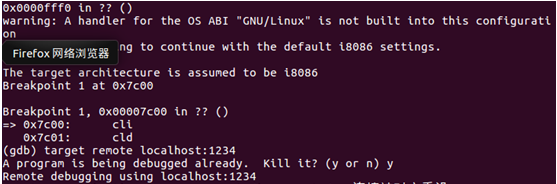
target remote :1234

然后输入makefile labi-mon 指令



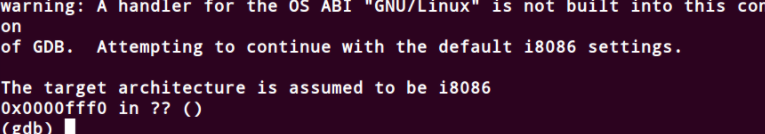
让qume把它执行的指令记录下来放到q.log

和GDB结合调试正在执行的Bootloader



1. 进行debug

Make debug



1. 在gdbinit中添加

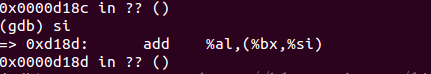
define hook-stop

x/i $pc

end

强制反汇编当前指令。

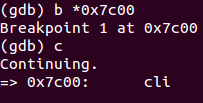
接下来使用si命令追踪bios即可



单步追踪之后，将gdb的结果与bootblock.asm比较，直到0x7ccf,除一些表达上的差异，没有很明显的区别。

1. 设置断点0x7c00并且跟踪代码运行

用指令b \*【断点处】

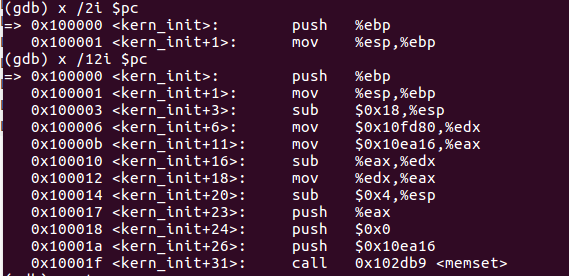


输入c继续运行到断点处然后停止

通过指令：



显示当前的2条汇编指令或多条指令



练习三：

bootloader 进入保护模式的过程。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。

答：bootbloader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，初始化GDT表，切换到保护模式, 跳转更新CS基地址， bootloader重新设置保护模式下的段寄存器，然后设置栈顶指针，然后进行跳转。