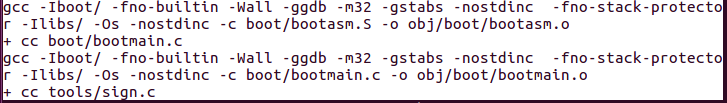
Lab 1实验

练习一：理解通过make生成执行文件的过程

操作系统镜像文件ucore.Img

Make V=得到以下命令



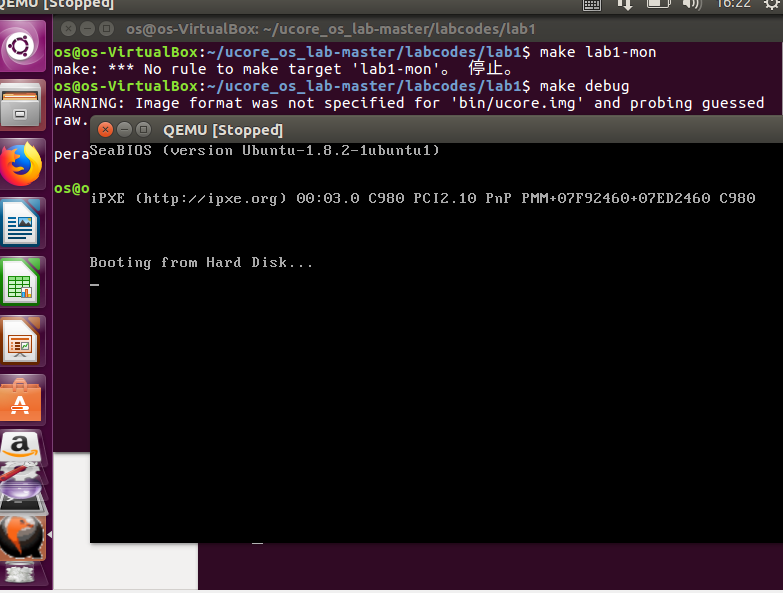
对应生成以下ucore.img代码

8

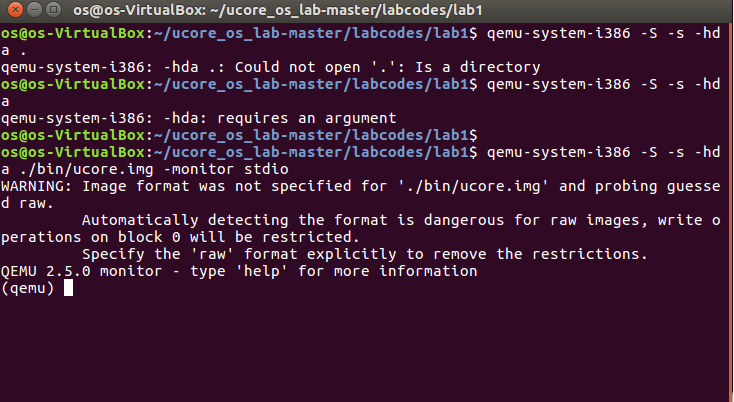
练习2使用qemu执行并调试lab1中的软件

2.1 从CPU加电后执行的第一条指令开始，单步跟踪BIOS的执行。

如果直接输入make debug会导致虚拟机直接死机不得不开机重新启动



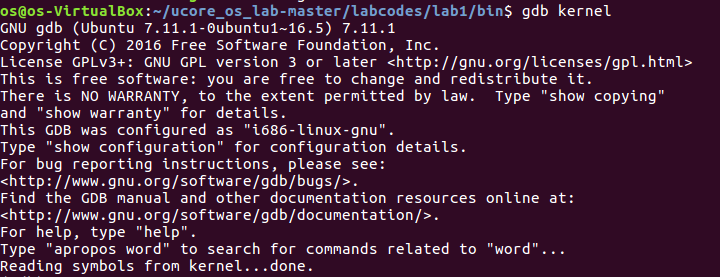
在lab1文件中进入终端，在其终端上输入qemu-system-i386 -S -s -hda ./bin/ucore.img -monitor stdio 来等待操作

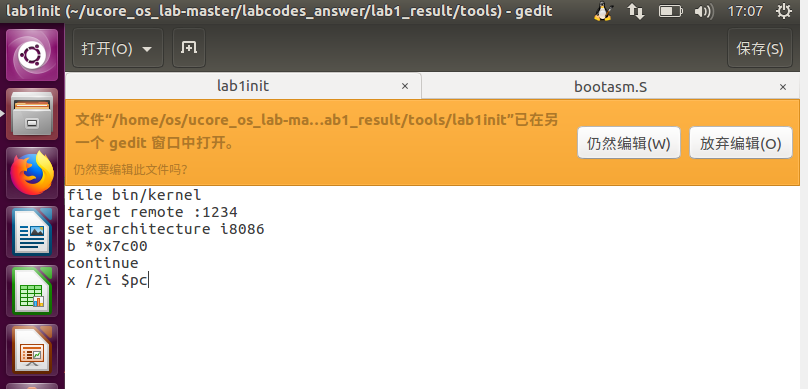


然后，打开另一个终端进入目标文件夹下，输入ls 可查看可操作文件

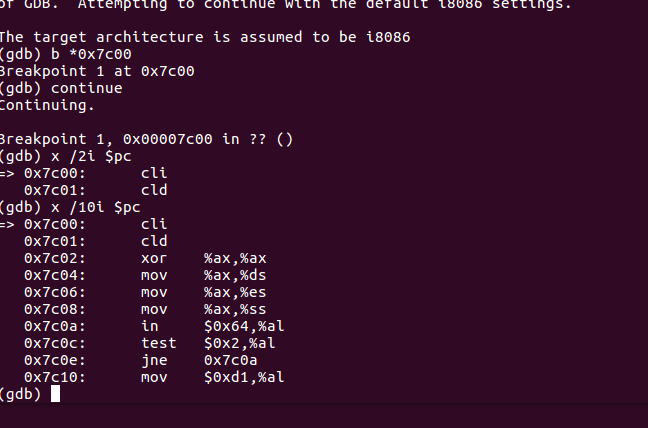
12

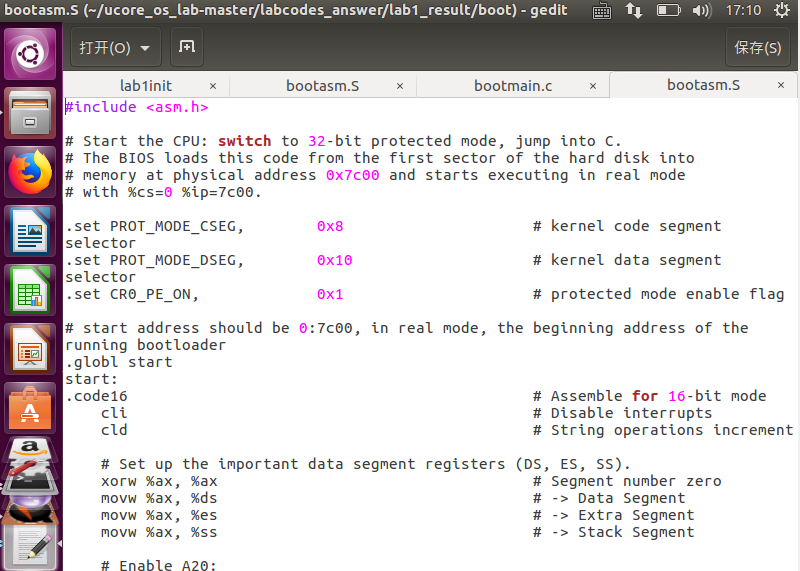
输入gdb kernel进行下一步操作



再打开下面的文件夹目录进行gdb操作

结果为



在以下的文件夹查看结果

为何开启A20

A20地址线并不是打开保护模式的关键，只是在保护模式下，不打开A20地址线，你将无法访问到所有的内存（具体参考下面的第5点）

用于80286与8086兼容

用于80286处于实模式下时，防止用户程序访问到100000h~10FFEFh之间的内存（高端内存）

8086模式，A20关闭的情况下，访问超过1MB内存时，会自动回卷

8086模式下，A20打开的情况下，访问超过1MB内存，就真实的访问

保护模式下，A20关闭（始终为0），则用户的地址只能是：0 - (1MB-1), 2 - (3MB-1), 4 - (5MB-1)，我们可以这样设想，A20为个位数（以1MB为单位），如果它始终为0，你永远不可能让这个数变成奇数。

保护模式下，A20开启，则可以访问全地址，没有奇偶MB的问题。

调用BIOS中断就可以实现A20 Gate的控制功能。

这个BIOS中断为 INT 15h,AX=2401h。被称为Fast A20。

如何开启A20：

三种方法分别是：1.通过键盘控制器。2.通过BIOS提供的中断。3.通过系统的I/O端口。

具体的实现如下：

1.通过键盘控制器：

cli ;Disables interrupts  
push ax ;Saves AX  
mov al, 0xdd ;Look at the command list //如果是0xdf就是关闭A20地址线  
out 0x64, al ;Command Register   
pop ax ;Restore's AX  
sti ;Enables interrupts  
ret

2.通过BIOS：

push ax  
mov ax, 0x2401 //如果是0x2400则关闭A20地址线  
int 0x15   
pop ax

这个方法有返回状态，如下：

CF = clear if success  
AH = 0  
CF = set on error  
AH = status (01=keyboard controller is in secure mode, 0x86=function not supported)

对应的还有一种检查当前状态的命令0x2402

push ax  
push cx  
mov ax, 0x2402   
int 0x15   
pop cx  
pop ax

他的返回状态如下：

CF = clear if success  
AH = status (01: keyboard controller is in secure mode; 0x86: function not supported)  
AL = current state (00: disabled, 01: enabled)  
CX = set to 0xffff is keyboard controller is no ready in 0xc000 read attempts  
CF = set on error

第三种方法：

push ax  
inb  al, 0x92  
orb 0x2, al  
outb 0x92, al

但是这是个八位的值，具体的每一位有不同的作用，如下：

Bit 0 - Setting to 1 causes a fast reset   
Bit 1 - 0: disable A20, 1: enable A20  
Bit 2 - Manufacturer defined  
Bit 3 - power on password bytes. 0: accessible, 1: inaccessible  
Bits 4-5 - Manufacturer defined  
Bits 6-7 - 00: HDD activity LED off, 01 or any value is "on"

所以打开A20的方法就是用将第二位置位即可。

如何初始化GDT表：

  实地址模式：

        286架构以前的采用实地址模式，使用16位段寄存器的内容乘以16得到段基地址，加上16位的偏移地址得到内存地址。这种方式只能访问1MB的内存地址。

保护地址模式：

80286开始出现的保护模式可以寻址1MB以上的内存，采用的是段号加GDT的方式。

        CPU用8个字节的数据表示这些信息，由于段寄存器只有16位，加上CPU设计上的原因，段寄存器的低3位不能使用，能够使用的只有13位。所以只能有2^13=8192个段号，即0~8191。这些段号信息记录在GDT中，需要8x8192个字节即64KB。由于CPU的存储能力有限，所以GDT存放于内存之中，将内存起始地址和有效设定个数存放在CPU里的GDTR 寄存器之中。

# **[如何进入保护模式？](https://www.cnblogs.com/wudibuzaijia/p/8483871.html)**

IA-32为操作系统及其软件的发展提供了大量的支持，它提供了4种操作模式：

 1,Real Mode,汇编书中的实模式，工作方式相当于8086。

 2,Protected Mode,保护模式，提供支持多任务环境的工作形式。

 3,Virtual 8086 Mode,虚拟8086模式，可以从保护模式切换成一种8086工作方式，可以使用户在保护模式下方便的运行一个或者多个8086源程序。

 4,System Management Mode。

在IA-32中，我们需要重点学习的是Protected Mode。

 查阅资料发现稍微老一些的汇编书中，只列出1-3这3种模式及他们的工作特点。

 出于好奇和探究，对第四种模式SMM进行了了解。

 首先只有SMI才会引起进入SMM，处理器保护现场，切换到SMRAM里的一个独立地址空间执行SMM代码，RSM指令会使系统返回原来的正常模式。

 SMM相当于实模式，没有特权级和地址映射，可寻址4GB，可执行所有I/O和可用系统指令。

 当系统处于SMM时，处理器不识别后继的SMI请求，但是第一个SMI请求可以被锁存，并在系统退出SMM后被处理。

 内存在IA-32中被分为2个部分，Segmentation,Paging。

 Segmentation提供了一种机制可以分隔DATA,CODE,STACK以便多个程序在同一个处理器运行时不会互相妨碍。

 Paging提供了一个通常的命令页，将程序执行的虚拟内存系统映射在物理内存当中。它同样也为多任务提供隔离。

 GDT是Protected Mode必须的数据结构，所以进入保护模式之前必须先设定好GDT。

 设定GDT，可以采用Basic Flat Model,因为这个Model要求至少两个段描述符，一个用来引用Data segment,一个用来引用Code segment。

 除此之外，还需要定义一个系统规定的空描述符。

 设置好GDT后，需要通过LGDT指令将设定的gdt入口地址和gdt表大小装入GDTR寄存器。

这时需要设置A20。

 A20我感觉就是系统升级时留下的BUG，8086/8088只有20位地址线，能够表示的最大内存为1M,当需要访问100000h~10FFEFh之间的内存，必须有第21根地址线，这个就是A20。

 现在在实模式下，能访问最大的内存也只有10FFEFh，再访问更大的需要进入保护模式，进入保护模式前，需要先打开A20。现在多数PC都是通过8204键盘控制器来处理A20，但A20和8204其实没什么关系。通过汇编实现A20开启代码如下：

asm push ax

asm in al,92h

asm or al,00000010b

asm out 92h,al

asm pop ax

之后开始切换到保护模式

asm mov eax,cr0

asm or eax,1

asm mov cr0,eax

 实质是将之前说过的PE位置1。

 理论上，这样就进入了保护模式。

如何使能：将cr0寄存器置1.