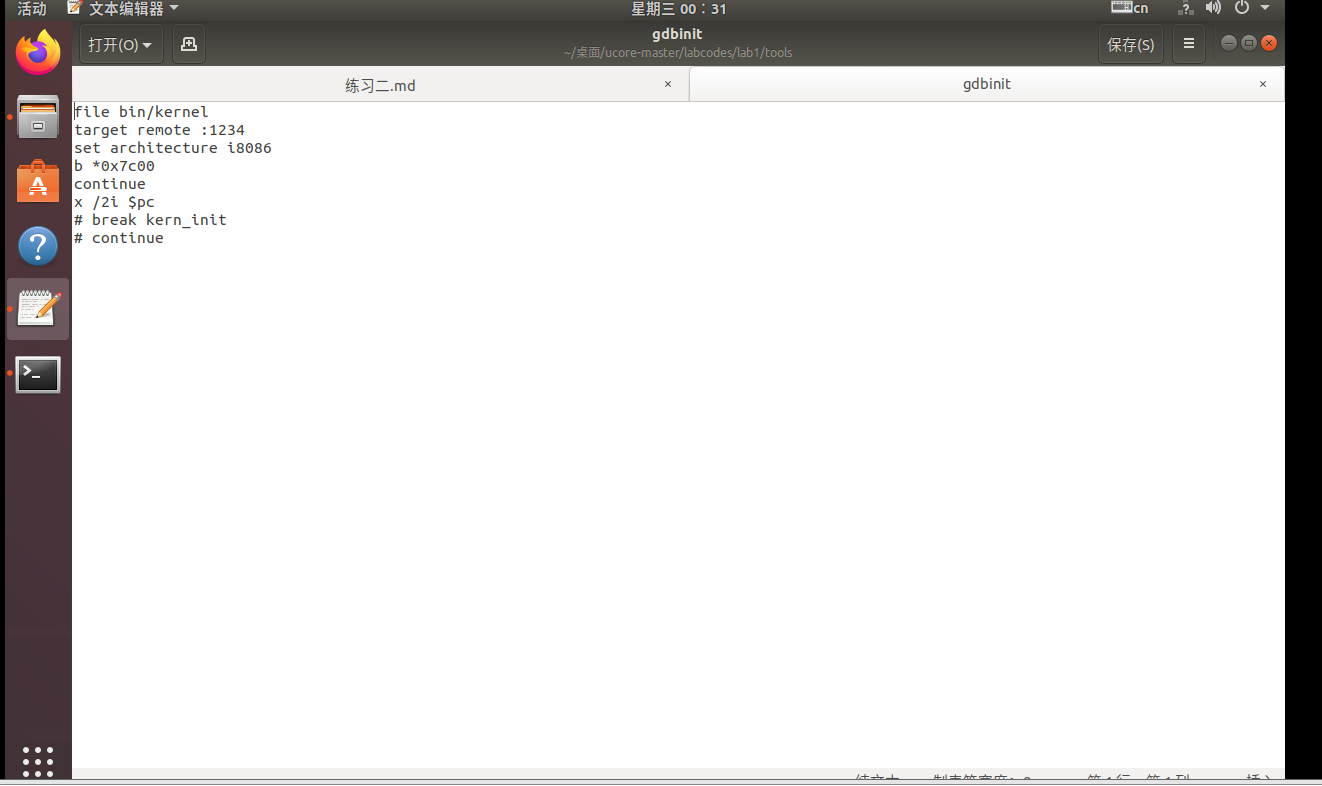
**练习二**

该实验主要是关于断点的调试。

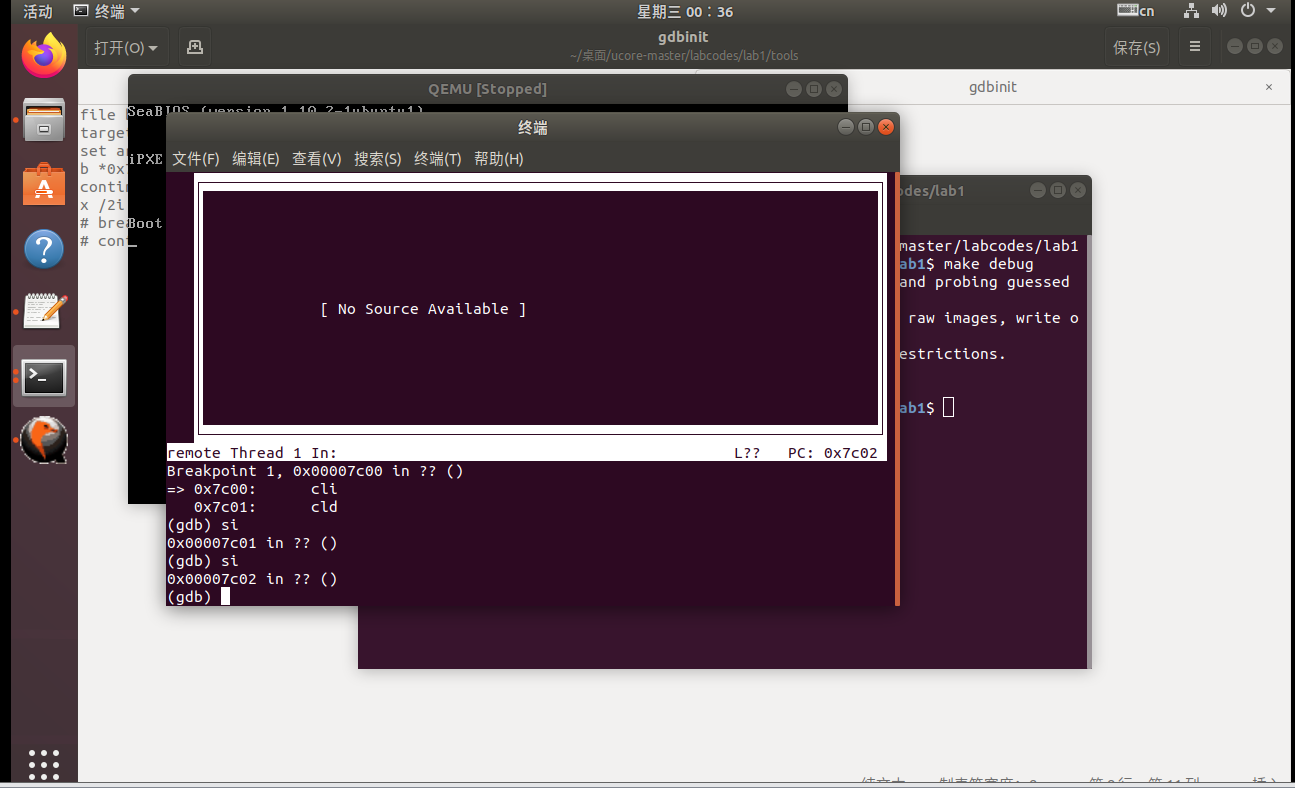
实验步骤：1. 从CPU加电后执行的第一条指令开始，单步跟踪BIOS的执行。

首先，将默认的gdb进行修改：



将断点设置在内核代码的入口地址，即kern\_init函数

然后，在lab 1目录下执行make debug：

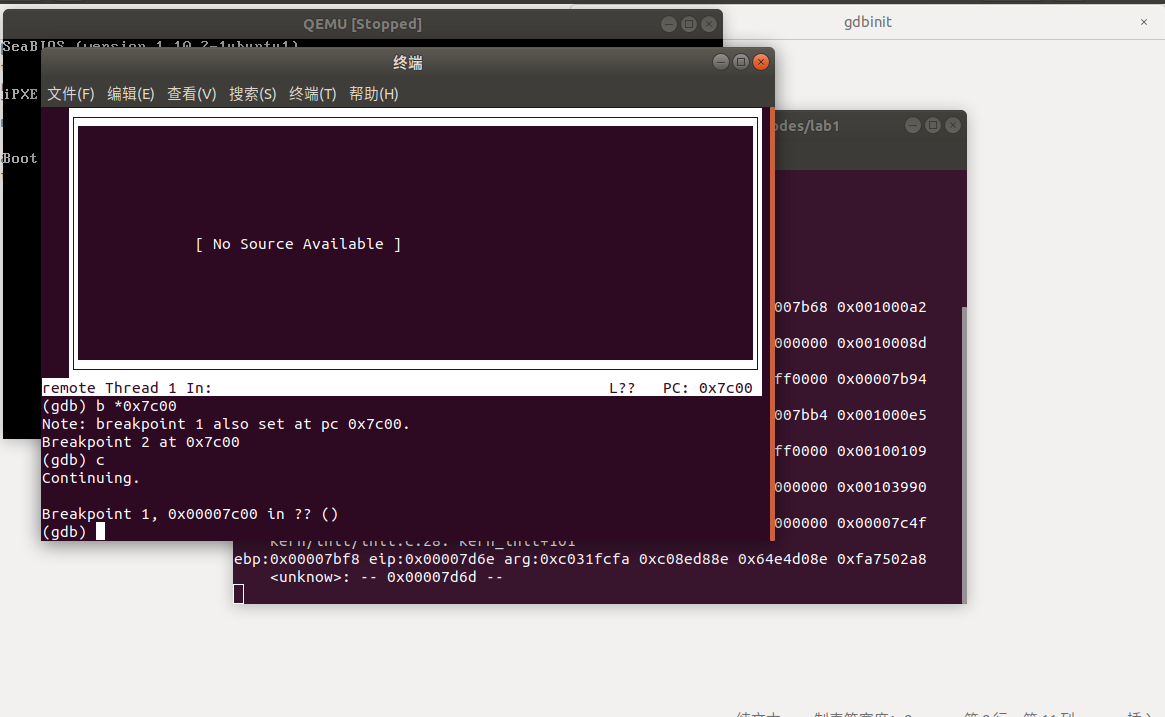


在 gdb中输入si指令，可以实现指令的单条运行

2.在初始化位置0x7c00设置实地址断点，测试断点正常：

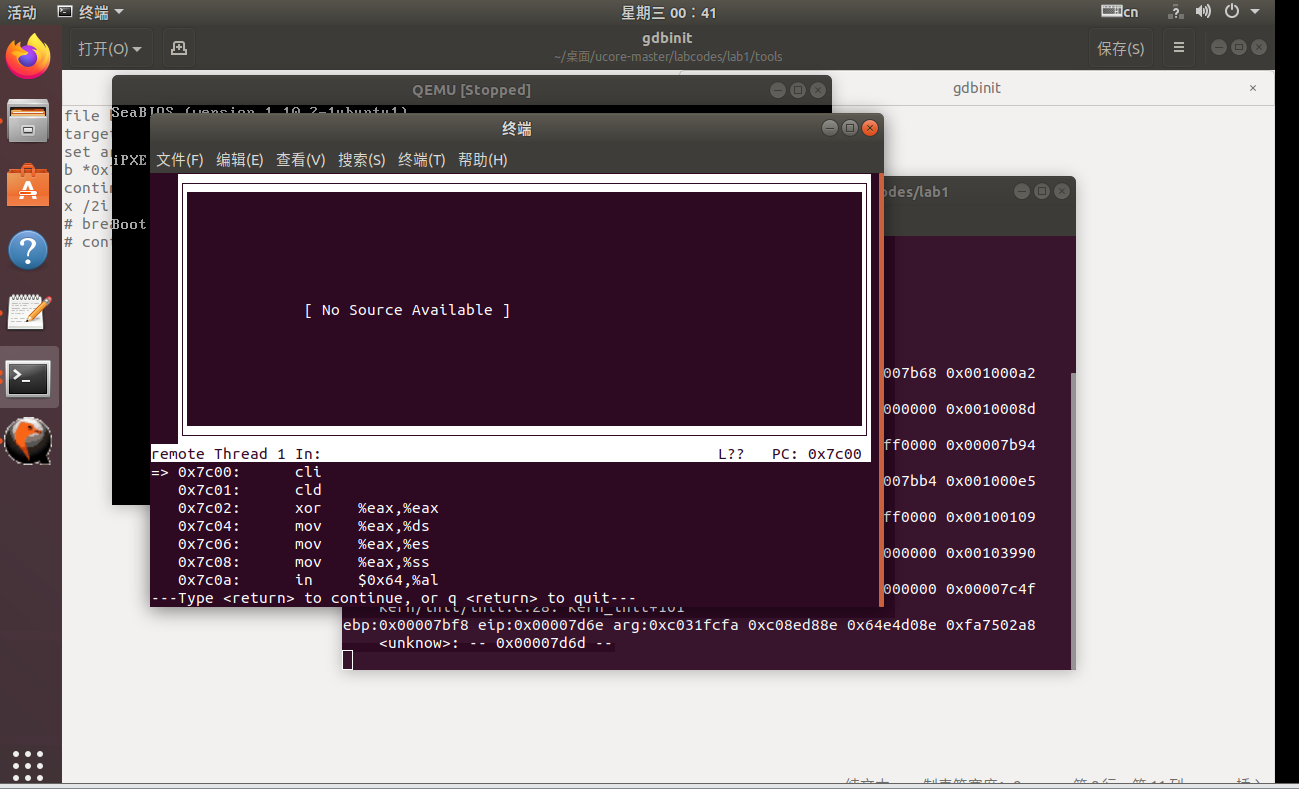
在gdb中输入：‘b \*0x7c00’       //设置断点   
   　　　　　　’c‘                //继续正在调试的程序

执行结果如下图所示：



设置断点为0x00007c00，则下次就在那里中断

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：

在断点0x7c00处多次执行’x /10i $pc’并查看后面10条汇编指令，执行结果如下所示：

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：

将gdbinit改成原来的命令，即对内核代码进行调试，并且将断点设置在内核代码的入口地址，即kern\_init函数;   
执行’make debug‘;    //程序进入init.c中的kern\_init()   
‘b cons\_init’          //设置断点，这是初始化console的函数   
’c‘                   //继续执行正在调试的程序   
4.4遇到断点，在console.c程序的cons\_init()函数处终止：