

本科生实验报告

实验课程:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_操作系统\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

实验名称:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_保护模式\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

专业名称:\_\_\_\_\_\_\_\_\_\_\_计算机科学与技术\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

学生姓名:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_凌国明\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

学生学号:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_21307077\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

实验地点:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_教室\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

实验成绩:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

报告时间:\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023.04.20\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **实验要求**

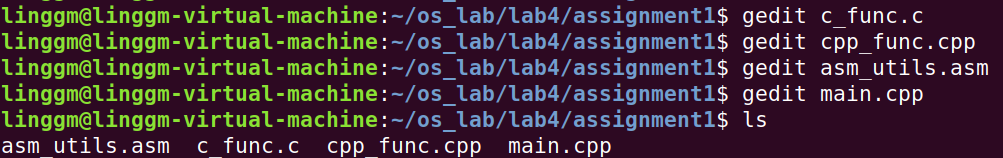
* 学习C代码是如何通过预编译、编译、汇编和链接生成最终的可执行文件
* 学习一种C/C++项目管理方案
* 学习C和汇编混合编程方法，即如何在C代码中调用汇编代码编写的函数和如何在汇编代码中调用使用C编写的函数
* 学习保护模式下的中断处理机制和可编程中断部件8259A芯片
* 通过编写实时钟中断处理函数来将本章的所有内容串联起来。

**实验任务**

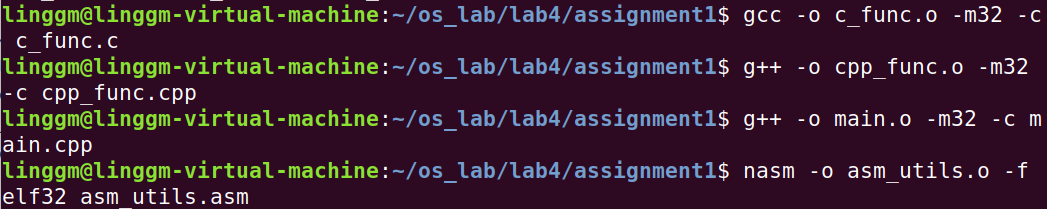
* 复现Example 1，结合具体的代码说明C代码调用汇编函数的语法和汇编代码调用C函数的语法。结合代码说明global、extern关键字的作用，结合代码说明global、extern关键字的作用
* 复现Example 2，在进入setup\_kernel函数后，将输出 Hello World 改为输出你的学号，结果截图并说说你是怎么做的
* 复现Example 3，你可以更改Example中默认的中断处理函数为你编写的函数，然后触发之，结果截图并说说你是怎么做的
* 复现Example 4，仿照Example中使用C语言来实现时钟中断的例子，利用C/C++、 InterruptManager、STDIO和你自己封装的类来实现你的时钟中断处理过程，结果截图并说说你是怎么做的。注意，不可以使用纯汇编的方式来实现。(例如，通过时钟中断，你可以在屏幕的第一行实 现一个跑马灯。跑马灯显示自己学号和英文名，即类似于LED屏幕显示的效果。)

1. **实验过程**
2. **复现example1**

* 第一步：编写各个文件



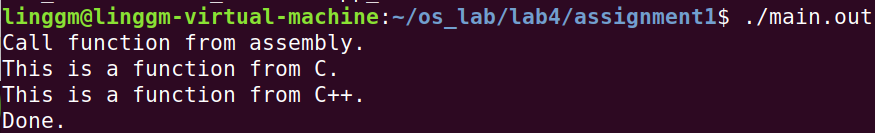
* 第二步：编译与汇编，c文件用gcc，cpp文件用g++，asm文件用nasm



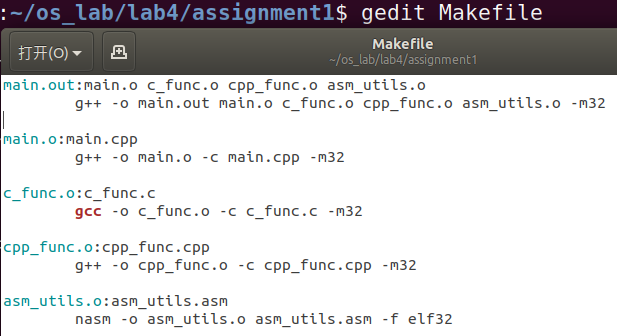
* 第三步，链接



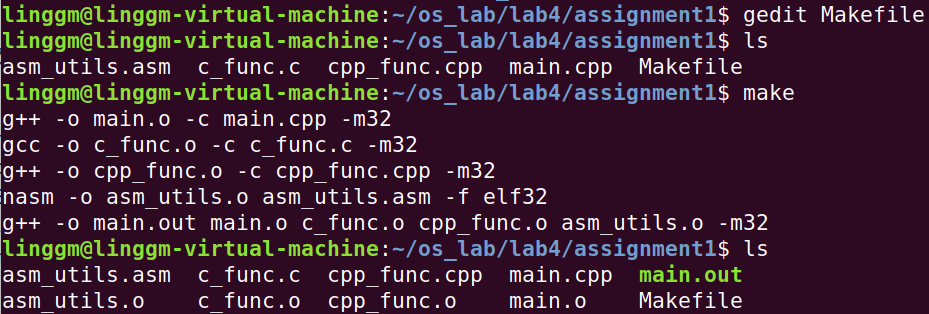
* 第四步：执行



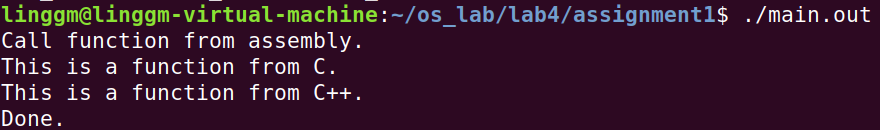
* 第一步：编写Makefile



* 第二步：make



* 第三步：运行

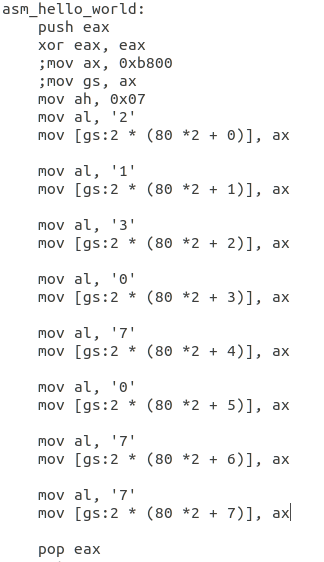


关键字：汇编代码中“global”关键字的作用是“使汇编符号对链接器可见”，这意味着和这份汇编代码链接的所有代码都“认识”这个符号，也就是可以“call”这个符号。C代码中的“extern”关键字表示这里引用了一个文件外部的变量，使用“extern”关键字可以使变量变为全局变量，可以跨文件访问。

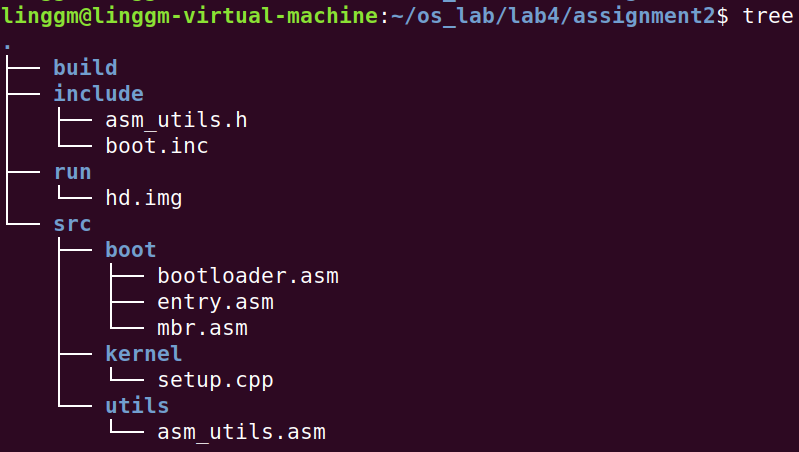
Make：编写makefile时，注意C文件用gcc，C++文件用g++，asm汇编代码文件用nasm。注意gcc、g++中的-m32与nasm中的-f elf32

1. **复现example2**

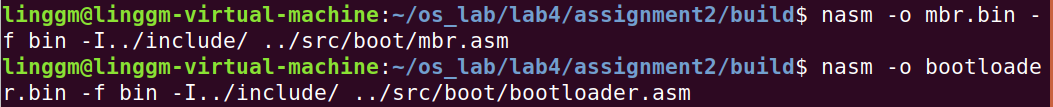
* 第一步：编写各项文件



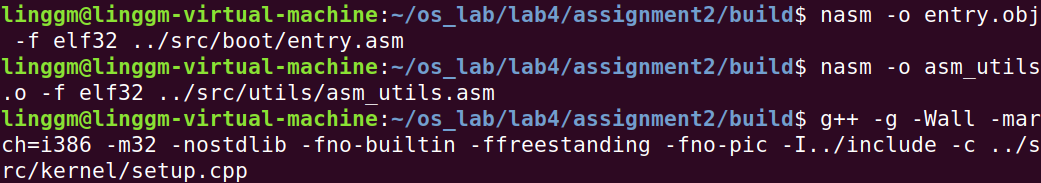
* 初始项目结构如下



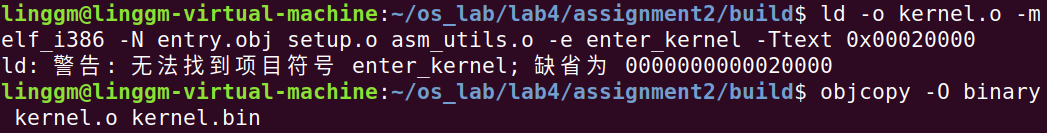
* 第二步：在build文件夹下编译mbr，bootloader



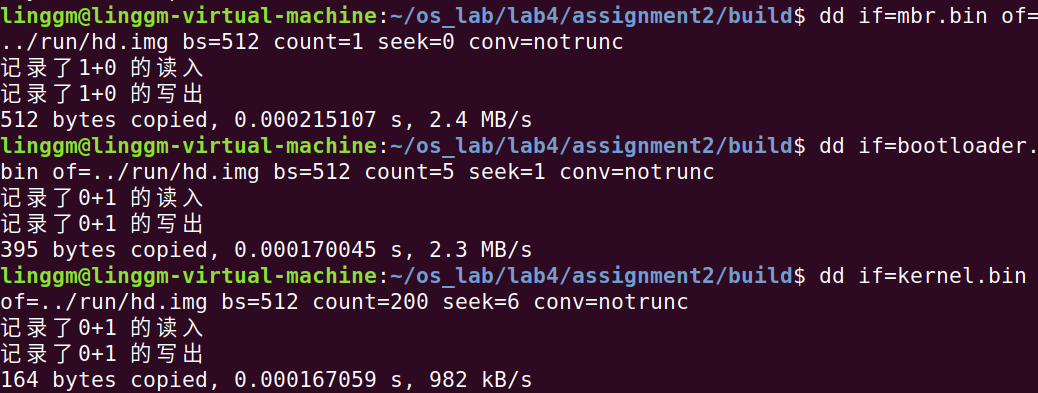
* 第三步：编译内核代码



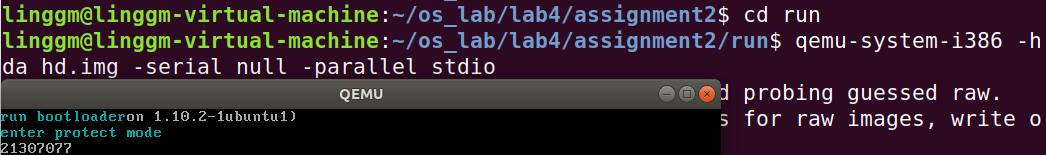
* 第四步：链接生成两个可重定位文件，kernel.bin只包含代码，kernel.o是可执行文件

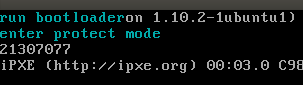


* 第五步：将mbr，bootloader，kernel写入磁盘



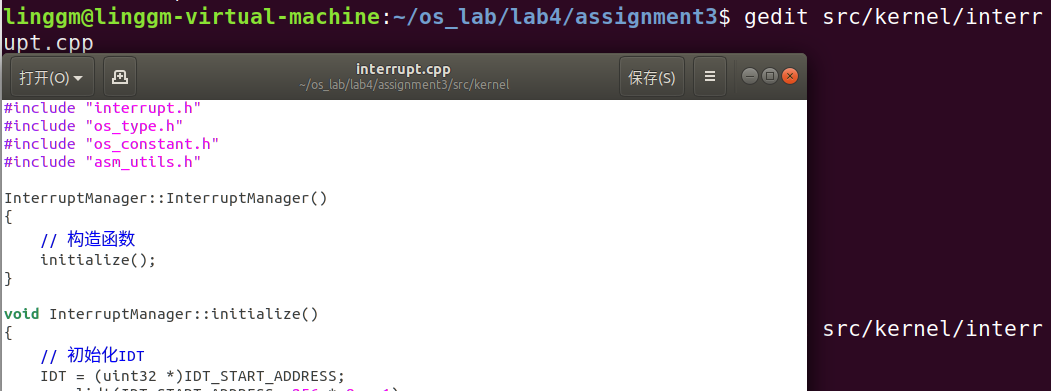
* 第六步：启动qemu测试



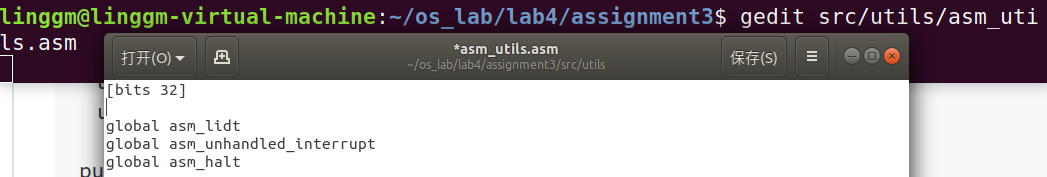


1. **复现example3**

* 第一步：编写各个代码文件



构造函数，初始化函数，加载idt函数



asm\_unhandled\_interrupt函数输出字符串

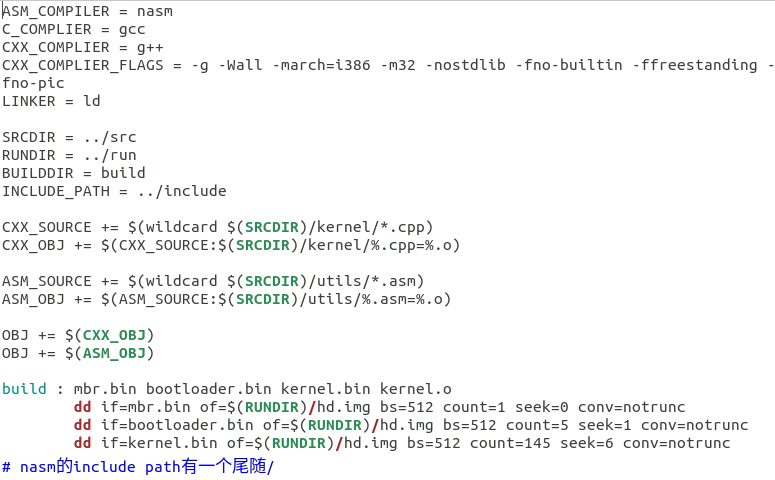
asm\_halt函数死循环，asm\_lidt函数加载idt



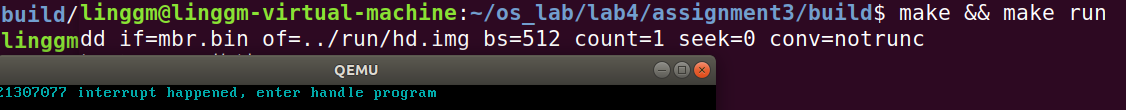
Bootloader最后要读200个扇区，最后跳转到kernel

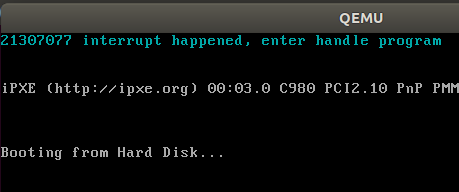
**编写文件过程中要特别注意各个include关系**

* 第二步：复制教程中的makefile



* 因为ex2中已经实现了用nasm，gcc，g++编译汇编链接各个文件的过程，这里直接用makefile。make run效果如下



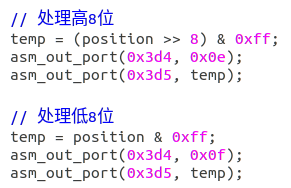


1. **复现example4**

* 第一步，编写stdio类

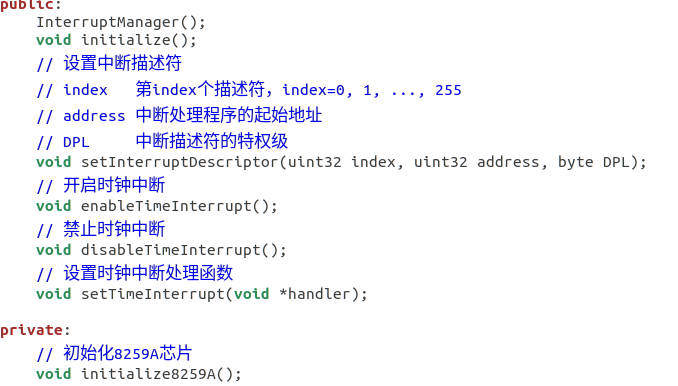


类的声明如上



注意：在对光标位置进行读写时，temp是八位的，pos是十六位的，所以temp=pos>>8表示temp赋值为pos的高八位；而temp=pos进行了位截断，表示pos的低八位；然后对两个端口进行读写即可

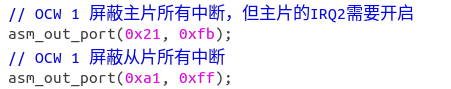
* 第二步：编写interrupt类



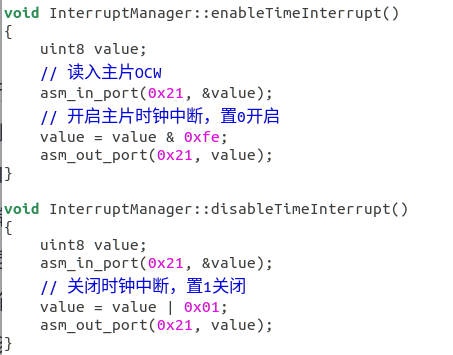
类的声明如上



相比assignment3，interrupt类的init函数新增了“初始化中断次数”和“初始化8259A”芯片



“初始化8259A芯片“的步骤较为固定，不赘述，但要注意屏蔽所有主从片中断（因为没有建立8259A芯片的中断处理函数）



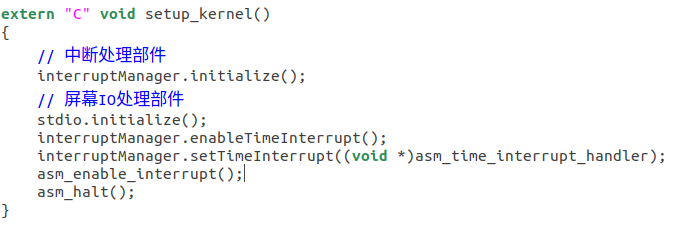
开关时钟中断部分：0x21端口的第0位表示IRQ0的屏蔽，即时钟中断的屏蔽，将第0位置0表示屏蔽时钟中断

* 第三步：在interrupt类中编写自己的中断处理函数



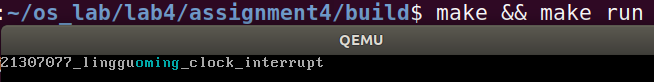
实现了利用时钟中断，进行跑马灯输出学号和姓名的功能

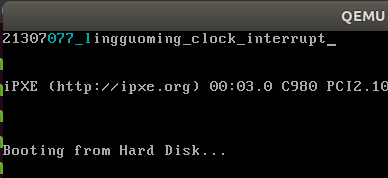
* 第四步：编写setup.cpp

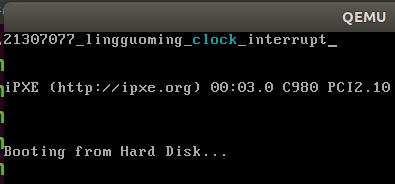


进行asm\_halt()后，不断执行jmp $，每隔一段时间触发时钟中断，从而转向中断处理程序进行跑马灯输出

* 第五步：启动qemu运行，结果如下





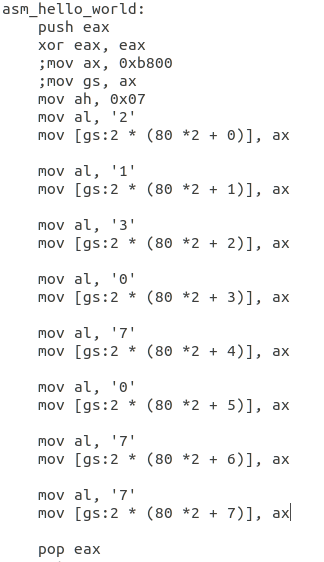


**关键代码**

1. Assignment1

所有代码与教程一致

1. Assignment2（debug信息见过程部分）



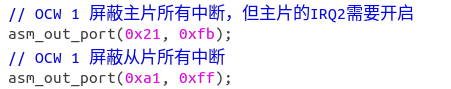
进入setup\_kernel函数后，调用asm\_hello\_world，将我的学号写入显存，输出到屏幕上。只有asm\_hello\_world代码与教程不同，所有仅展示这部分代码，其他省略。

1. Assignment3

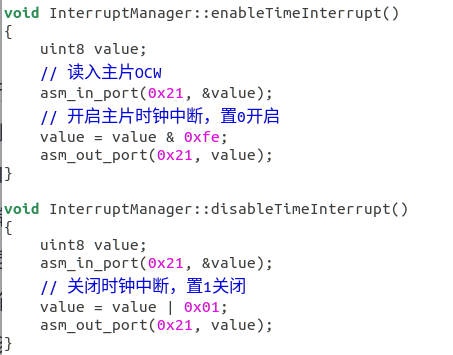


Bootloader最后要读200个扇区，最后跳转到kernel

1. Assignment4



“初始化8259A芯片“的步骤较为固定，不赘述，但要注意屏蔽所有主从片中断（因为没有建立8259A芯片的中断处理函数）



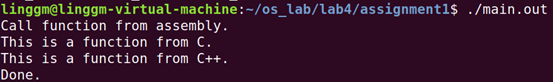
开关时钟中断部分：0x21端口的第0位表示IRQ0的屏蔽，即时钟中断的屏蔽，将第0位置0表示屏蔽时钟中断



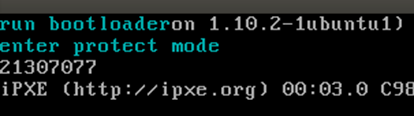
定义字符串str为我的学号和姓名，然后for遍历字符串进行输出

字符串的长度为36，当中断次数times % 36 和 字符串索引i距离较近时，对这部分字符进行着色（青色0x03），其他字符则是默认颜色（白色0x07）

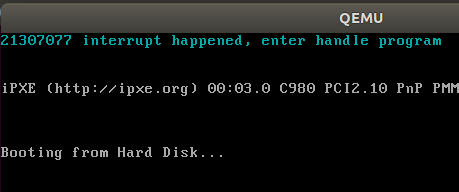
1. **实验结果**
2. Assignment1

****

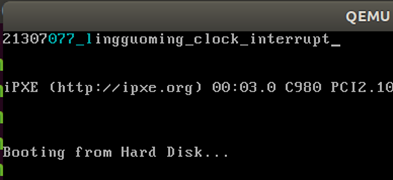
1. Assignment2

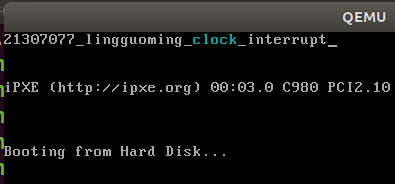


1. Assignment3



1. Assignment4

****

****