

221220034

OS Lab1 实验报告

报告人

姓名：王旭

专业：计算机科学与技术系

学号：221220034

邮箱：2069625874@qq.com

实验进度

我完成了所有任务

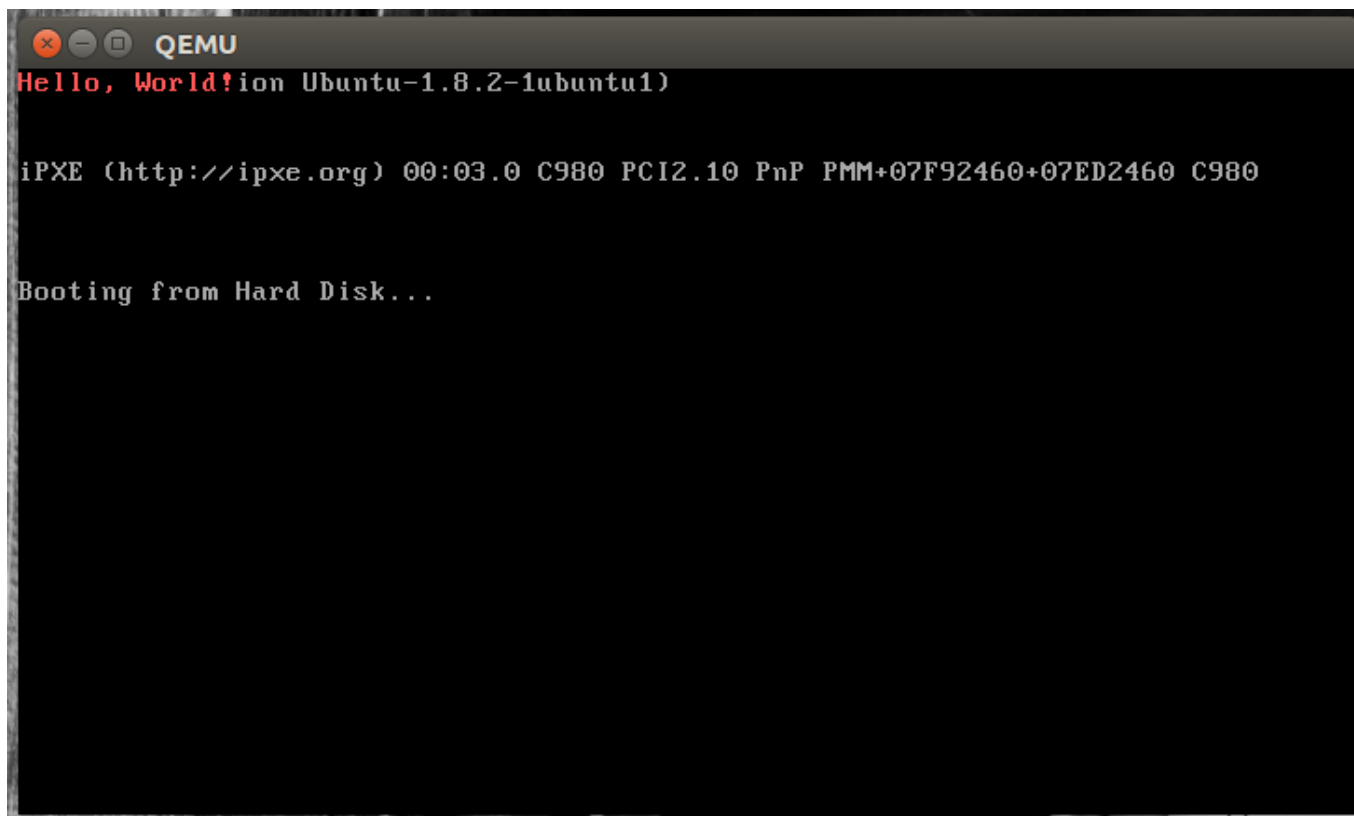
实验过程&结果

lab1.1 实模式下打印“Hello World”

代码实现

在bootloader/start.s文件中的任务一板块，添加了引发中断并提前设置各种参数（如字体的颜色、个数）的函数displayStr，在主函数中跳转执行此函数即可

实验结果



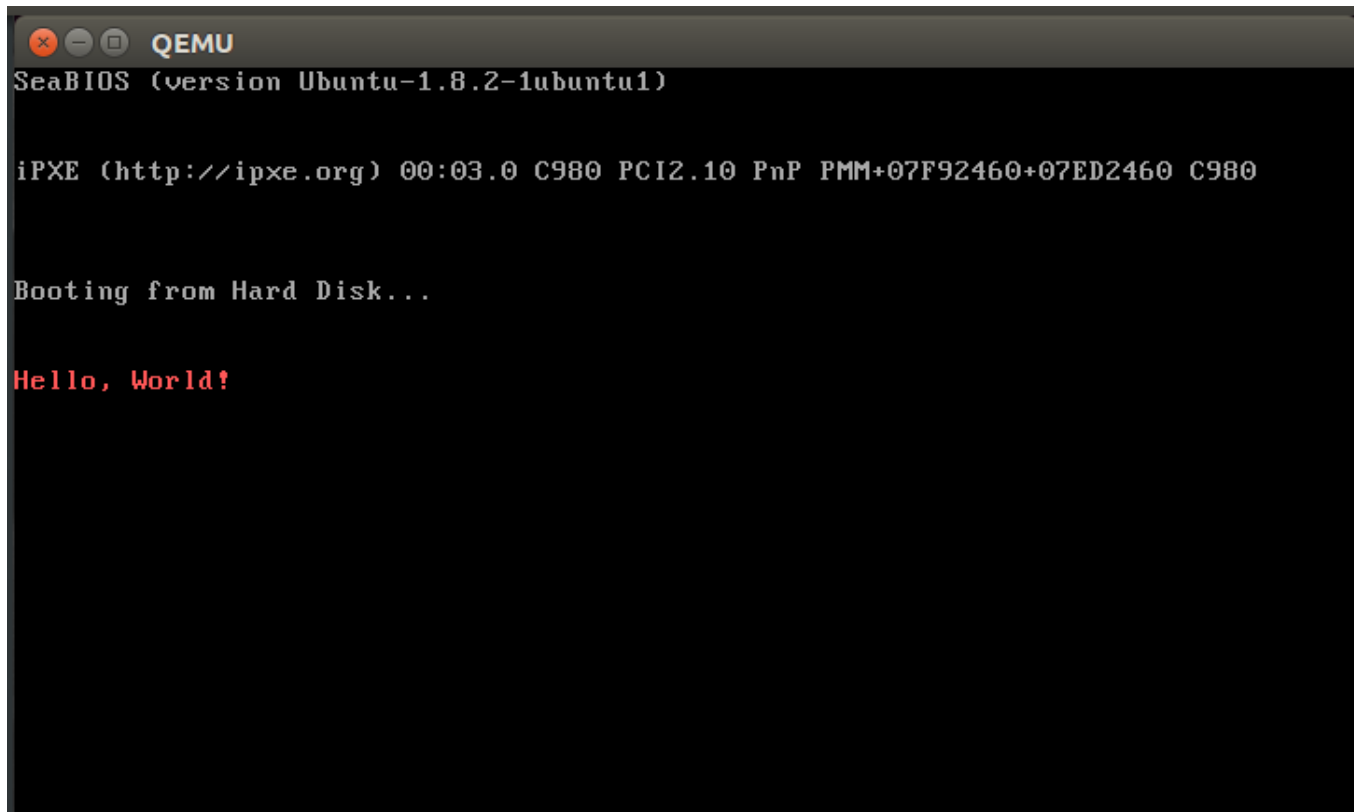
lab1.2 保护模式实现“Hello World” 程序

代码实现

1. 在bootloader/start.s文件中使用“cli”指令关中断，然后利用%eax寄存器和xor指令将cr0的PE 置1；
在设置esp的后面添加打印Hello World 的代码，主要是先将字符串的长度、地址以及字体显现特征赋给寄存器，并且第一个字符设定在VGA的第10行第0列；然后设置一个循环，每次利用ax将下一个字符的地址与设置传给VGA的段中某一地址，然后依次循环打印，直至打印13个字符。
2. 补充GDT表项：根据说明上的描述判断每个段的基地址以及type即可，其中type的判断需要额外到手册上找相关说明。

实验结果

注：我所设置的是 $(80 \times 10) + 0$ ；也就是在第10行第0列打印第一个字符



```
QEMU
SeaBIOS (version Ubuntu-1.8.2-1ubuntu1)

iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+07F92460+07ED2460 C980

Booting from Hard Disk...

Hello, World!
```

lab1.3 保护模式下加载磁盘程序运行

代码实现

1. 实现GDT表项：按照任务而二中表项填充即可；
2. start.s文件中跳转至bootloader/boot.c中的bootmain函数，补充此函数，答题思路就是利用readSect()函数读入RAM后的第一个扇区，在利用某种办法跳转至函数入口，为此有两种做法：
3. 利用函数指针，将0x8c00强制类型转换并赋值给该指针，在读入磁盘，执行此函数，跳转至0x8c00；
4. 直接声名void* 型变量，然后将0x8c00强制类型转换赋值，然后利用行内汇编跳转过去。

实验结果

```
QEMU
SeaBIOS (version Ubuntu-1.8.2-1ubuntu1)

iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+07F92460+07ED2460 C980
Hello, World!
Booting from Hard Disk...
-
```

其他

问题与思考

1. lab1使我遇到的第一个问题在于index文件中提供的样例中%ebx的赋值，0x000c的意义。因为在bios中断手册中并未详细写出%ebx中各个取值的意义，还有后续的lab1.2中应该如何将字符拷至VGA对应内存中以及如何去实现打印的属性以及位置等等实现时的小问题。
2. 通过在网上找相关博客去更多的了解详细信息，以及阅读更多的汇编代码，解决了遇到的问题。
3. 最大的问题在于C 语言中可以轻易实现的用汇编语言则会稍微麻烦一些，同时最基本的汇编语言的跳转，在C 语言中实现是需要一些技巧的，就如lab1.3，可以使用行内汇编，但也可以用函数指针，然后赋值，调用实现跳转。

思考题

1. CPU是中央处理器，负责调取并执行指令，它会从内存中读取数据与指令，并且会把数据存入内存中；

2. Bios是开机引导程序，开机后CPU会先执行Bios程序，从磁盘中读入主引导扇区，并执行其加载程序，然后该程序将操作系统的代码与数据加载到内存，跳转到操作系统的入口，控制权交给操作系统。

实验心得

1. 汇编代码上手比较生疏，刚开始不知干什么，在仔细阅读并分析index中代码逐渐上手，这不由的想到徐峰老师说的，要先会读代码、多读代码，才有可能写出好代码；
2. OSlab与之前PA有很大不同，手册中的提示信息较少，很多时候不知如何下手，不知有哪些没有顾忌到，更需要自己会查找资料，这也很大促进了学习能力，同时对开机时如何加载操作系统并执行有了更深的理解。