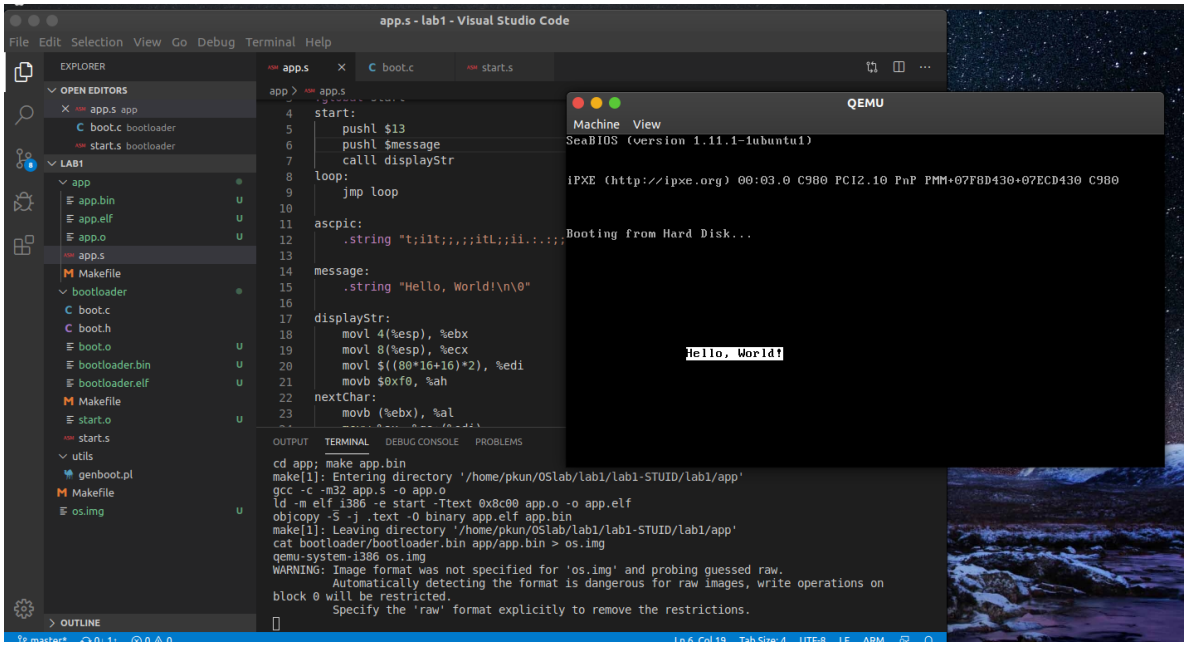
Lab 1 实验报告

> 181250068 李淳
>
> 邮箱：181250068@smail.nju.edu.cn
>
> QQ：936637810

实验进度

我完成了所有内容，在保护模式下加载磁盘中的Hello World程序并且运行

实验结果



修改了`%edi`，换了个位置打印

实验修改的代码

`boot.c`中添加了两条内联汇编（14，15行），在读完硬盘后跳到`0x8c00`的位置运行app

`start.s`中注释掉了中断打印的代码，在第26-33行进行了段寄存器和栈的初始化

实验中遇到的问题

- 不知道该如何初始化段寄存器（解决）
  - 最开始的时候是仿照`index.pdf`中的代码直接将`%ax`的内容movw到段寄存器中，结果发现根据没法运行，也不知道该怎么办，就会去重读`lab1.pdf`，后来意识到自己不知道如何初始化本质上应该还是对保护模式下寻址的全过程了解的不够细致。
    整个寻址的过程是先从段选择子（也就是段寄存器）中取出能在gdt中找到对应段描述符的索引，那就是说段寄存器的初始化的内容应该指向gdt中的索引，这才明白该如何初始化段寄存器
- 不知道如何初始化栈

- 在解决了上面那个问题后，只剩初始化栈了，我最开始将栈设置在了`0x8c00`的位置，因为这是Hello World程序开始的位置，运行后发现成功打印了，但是我又想，如果在这个地方设置一个栈，会不会把原来写进内存的程序给覆盖掉？于是我又尝试了几个不同的栈初始化的地址，发现均可以成功运行程序，但是将栈初始化到`0x7d00`就不能运行了，后来问了一下实验助教后知道只需要留够一定的地址就可以了

0.不打开A20总线可以寻32位的地址吗？有什么缺陷

经过查找资料后，不打开A20总线是不可以寻32位地址的，A20总线是系统升级时为了向下兼容而被创造出来的。因为在实模式下，能够表示出的最大内存为`0xFFFF<<4+0xFFFF=0x10FFEF=1MB+64KB-16B`，但是8086只有20根地址线，如果要访问超过1MB的地址就需要21根地址线。这第21根地址线就是A20总线

> 所以当程序员给出超过1M（100000H-10FFEFH）的地址时，系统并不认为其访问越界而产生异常，而是自动从重新0开始计算，也就是说系统计算实际地址的时候是按照对1M求模的方式进行的，这种技术被称为wrap-around。（From CSDN, https://blog.csdn.net/ruyanhai/article/details/7181842）

因此，在A20不被打开的情况下，如果程序访问1MB以上的地址，系统将仍然按照实模式来寻址。对于80386来说有32位地址总线，**如果A20被关闭，那么表示地址的32bit中第20-bit是无效的，永远被当成0，可以被访问的内存也有限，无法寻32位地址**

1.对GDT的分析

代码段描述符：

```
.word 0xffff,0
.byte 0,0x9a,0xcf,0
```

段基址为：`0xff00009a`

段限长为：`0xfcf00`

段权限为：`0b00`，特权级最高

数据段描述符：

```
.word 0xffff,0
.byte 0,0x92,0xcf,0
```

段基址为：`0xff000092`

段限长为：`0xfcf00`

段权限为：`0b00`，特权级最高

视频段描述符：

```
.word 0xffff,0x8000
.byte 0x0b,0x92,0xcf,0
```

段基址为：`0xff000b92`

段限长为：`0xfcf00`

段权限为：`0b00`，特权级最高

2.CPU、内存、BIOS、磁盘、主引导扇区、加载程序、操作系统的含义与相互关系

含义

- CPU：处理机器指令，操作数据
- 内存：所有程序都是放在内存中的，CPU取数据是和内存进行交互
- BIOS：内存ROM条上的固件，是开机后加载的第一个程序，能够将磁盘上的操作系统加载到内存中
- 主引导扇区：0号柱面，0号磁头，0号扇区对应的扇区，存放了加载程序
- 加载程序：将操作系统的代码和数据从磁盘加载到内存中
- 操作系统：向软件提供硬件接口并且管理硬件资源

关系

内存是存放**加载程序**，**BIOS固件**，**操作系统**的地方，**BIOS**的目的是为了将**磁盘**中的**操作系统**加载进**内存**，其过程主要靠**主引导扇区**里面的**加载程序**来加载，所有的指令都由**CPU**完成