OS Lab1 实验报告

姓名：孙文博

学号：201830210

邮箱：201830210@smail.edu.nju.cn

---

1. 实验进度

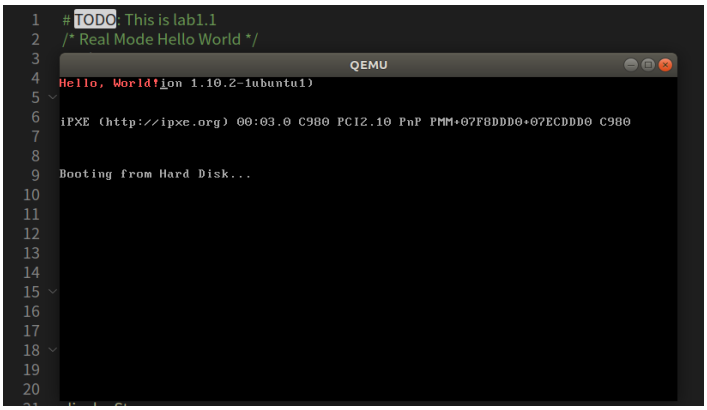
完成了实验 `lab1` 的全部内容，回答了部分思考题。

---

2. 实验结果及修改代码

2.1 实模式 lab 1.1:

按照手册要求配置好实验环境，将`start.s`中第一部分代码注释去掉即可。



2.2 实模式 -> 保护模式 lab 1.2

这部分我们需要将操作系统从实模式切换至保护模式，框架代码中已经实现了部分步骤如关中断等，我们需要补充的地方有四处：

```
    movl %cr0, %eax                 # 设置CR0的PE位（第0位）为1
    orb $0x01, %al
    movl %eax, %cr0
```

```
    movw $0x10, %ax                 # 初始化DS ES FS GS SS
    movw %ax, %ds
    movw %ax, %es
    movw %ax, %ss
    movw %ax, %fs
```

```
    movw $0x18, %ax
    movw %ax, %gs
    movl $0x1c00, %eax              # 初始化栈顶指针ESP
    movl %eax,%esp
```

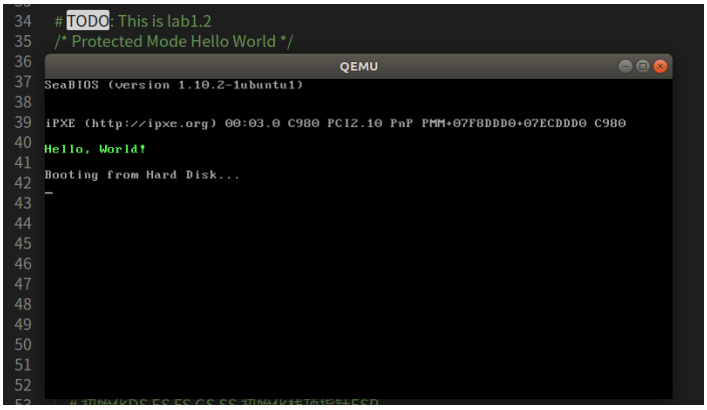
```
gdt:                                # 填写GDT表
    .word 0,0                       # GDT第一个表项必须为空
    .byte 0,0,0,0

    .word 0xffff,0                  # 代码段描述符
    .byte 0,0x9a,0xcf,0

    .word 0xffff,0                  # 数据段描述符
    .byte 0,0x92,0xcf,0

    .word 0xffff,0x8000             # 视频段描述符
    .byte 0x0b,0x92,0xcf,0
```

其中esp的初始值不唯一，不超过0x7c00就ok。修改之后成功显示如下：
（修改了displayStr部分中控制输出颜色的语句）



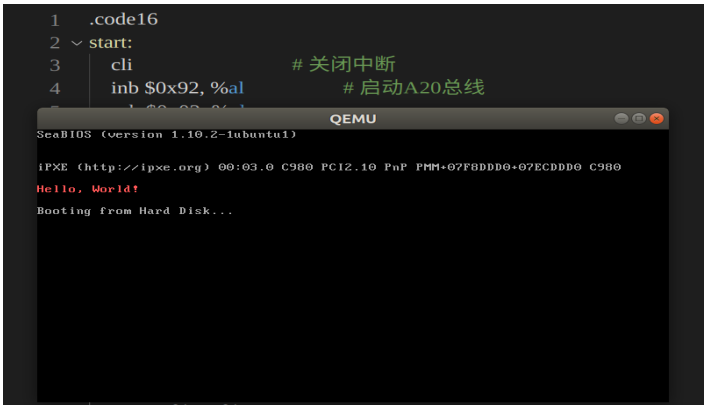
2.3 保护模式 lab 1.3

这部分中需要使用代码框架中的readSect(void *dst, int offset)接口，实现对磁盘扇区的读取，实现的代码如下：

```
void bootMain(void)
{
    int i = 0;
    void (*elf)(void);
    elf = (void(*)(void))0x8c00;
    for (i = 0; i < 200; i++)
    {
        readSect((void*)(elf + i*512), 1+i);
    }
    readSect((void*)elf, 1); // loading sector 1 to 0x8c00
```

```
    // jumping to the loaded program
    elf();
}
```

其中HelloWorld程序已经写入1号扇区，将它加载到内存运行，结果如下：



---

3. 实验思考题

Exercise1

答：CPU是计算机中央存理器，内存是计算机主存储器，BIOS是基本输入输出系统，磁盘是计算机的存储介质，加载程序是主引导扇区中负责将操作系统的代码和数据从磁盘加载到内存中的一组程序，操作系统是管理控制计算机操作、运行各种软硬件及组织用户交互等的系统软件程序。

Exercise2

答：中断向量表是系统内存中最低端 1K 字节空间，它的作用是按照中断类型号从小到大的顺序存储对应的中断向量，总共存储 256 个中断向量.

当响应中断的时候, cpu 可以从获取的中断类型号计算中断向量在中断向量表中的位置, 然后获取中断服务程序的地址.

Exercise3

答：因为段内偏移量为 0x0000~0xFFFF , 换算成二进制就是 16 位, 2^16B = 64KB.

Exercise4

答：当文件大于510字节后, genboot.pl 的处理代码如下：

```
if($n > 510){
print STDERR "ERROR: boot block too large: $n bytes (max 510)\n";
exit 1;
}
```

大概可以推断出, 此时 genboot.pl 会报错 ERROR: boot block too large: $n bytes (max 510)\n , 然后结束程序的运行. 这样做的原因是当文件超过510字节时, 再添加两个字节 0x55, 0xaa 后, BootLoader 的大小就会超过512字节, 就会超过了 MBR 的大小.

Exercise5

答：电脑开机后, 首先开启 `BIOS`。

`BIOS` 先做一些硬件检查, 然后将我们处理好的`bootloader.bin` , 也就是我们自己做的 `MBR` , 导入到地址为 `0x7c00` 的位置。

然后控制权被转交给 `bootloader.bin` , `bootloader.bin` 加载了 `GDT` ; 将 `cr0` 最低为设为1, 开启保护模式 ; 设置了段寄存器 ; 最后将我们写好的 `app.bin` 读取到 `0x8c00` 的位置, 并跳转到 `0x8c00` 的位置执行, `app.bin` 在屏幕上打印了 `Hello, World!`

---

4. 实验心得

本次实验作为 `OS_lab` 的开端，设置的并不是很困难，主要目的在于帮我们设置环境以及熟悉背景知识等。整体完成后确实会加深我们对课内知识的理解，具体的理解了一些概念如 `GDT` 表等。

遇到的最大困难可能是虚拟机给的存储空间不够（😨），导致几次写完代码之后没有保存成功，解决方法是重装了一次虚拟机并清理了一些内存。

感谢老师和助教提供的悉心指导的实验手册，感谢群友们的答疑解惑，希望之后的实验也可以一帆风顺！ 🤞