《操作系统》实验报告

lab1

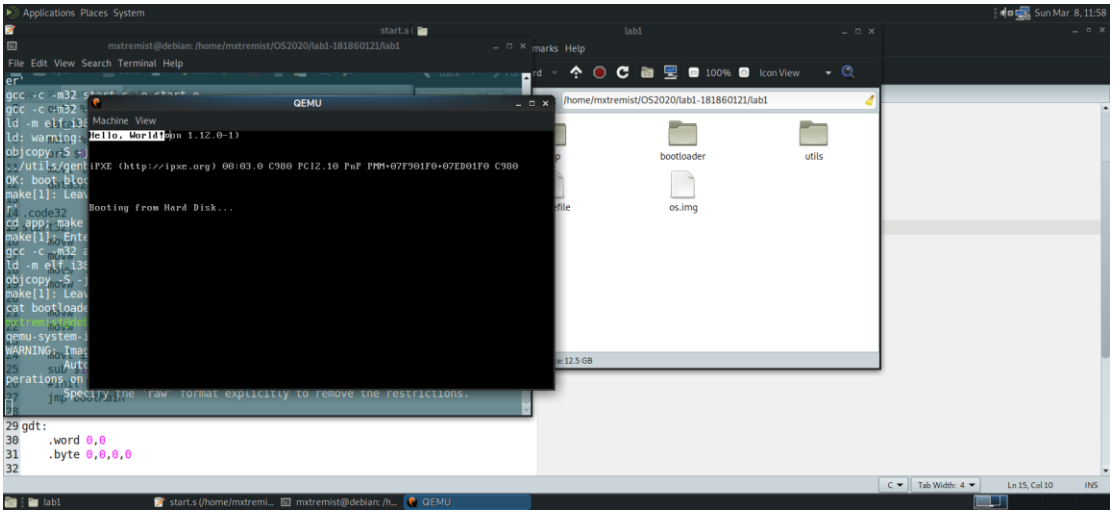
2020-3-7

薛飞阳 181860121
502126785@qq.com

1. 实验进度

我完成在实模式下在终端中打印 Hello, World!；从实模式切换至保护模式，并在保护模式下在终端中打印 Hello, World!；从实模式切换至保护模式。在保护模式下读取磁盘中的程序至相应位置，跳转并执行该程序，在终端中打印 Hello, World!

2. 实验结果



如图，终端中打印出 Hello, World!

3. 实验修改的代码位置

1) start.s:

为了从实模式切换至保护模式，必须对原有的用于实模式下打印的 start.s 文件进行修改。

首先，复制手册中提供的 gdt 表、启动并跳转至保护模式等内容；然后，补全 start32 中对段寄存器和 ESP 的初始化。DS、ES、SS 指向第二个 gdt 表项，故赋值$(2 << 3)，gs 指向第三个表项，故赋值$(3 << 3)，fs 未使用。ESP 赋值(128 << 20) – 16。

2) boot.c:

跳转到 bootMain 之后，需要在读取磁盘结束后跳转到磁盘一号区的程序，bootMain 中提供了无类型指针 void *elf 指向磁盘的第一个扇区。故将其当作函数指针使用，添加语句 elf()，就可以实现跳转到 1 号区的功能。

4. 问题的思考

```
gdt:
    .word 0,0
    .byte 0,0,0,0

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

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

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

gdt 表的第 0 项为空表项，仅起标志作用。

第 1 项用于初始化 cs 寄存器，0xffff 和 0xcf 中的 f 构成 20 位的限界，一个.word 和第一个.byte 的 0、第 4 个 byte 的 0 构成 32 位的基地址 0，之后的 0x9a = 1001 1010，即 P = 1（在主存中），DPL = 0， S = 1（代码段描述符），Type = 0xa，意为未被访问过的、可读的非一致代码段。0xcf 中的 c = 1100，即 G = 1，D = 1，表示以页为基本单位，地址和数据为 32 位宽。

第 2 项用于初始化 ds、es、ss 寄存器，0xffff 和 0xcf 中的 f 构成 20 位的限界，一个.word 和第一个.byte 的 0、第 4 个 byte 的 0 构成 32 位的基地址 0，之后的 0x92 = 1001 0010，即 P = 1（在主存中），DPL = 0， S = 1（代码段描述符），Type = 0x2，意为未被访问过的、可读写的普通数据段。0xcf 中的 c = 1100，即 G = 1，D = 1，表示以页为基本单位，地址和数据为 32 位宽。

第 3 项用于初始化 gs 寄存器，除了基地址为 0x000b8000 以外其他都与第 2 项相同。