Oslab1系统引导实验报告

**一．实验目的**

1.学习在linux环境下编写、调试程序，初步掌握Shell、Vim、GCC、Binutils、Make、QEMU、GDB的使用

2.学习AT&T汇编程序的特点

3.理解系统引导程序的含义，理解系统引导启动的过程

1. **实验设计**

1.程序的流程是：

（1）关中断、开A20地址线、加载GDTR寄存器

（2）设置CR0寄存器PE位为1（表示进入保护模式）

（3）长跳转进入保护模式代码

（4）初始化DS、SS、ES、FS、GS这些段寄存器和ESP

（5）跳转到bootMain中的代码

（6）加载sector 1 to 0x8c00程序然后跳转执行

（7）执行hello，world

2.已经完成的代码：

（1）80386处理器从实模式切换到32位的保护模式；

（2）各个段寄存器以及栈顶指针ESP的初始化；

（3）加载存储在MBR之后的磁盘扇区中的程序到内存特定位置并跳转执行；

（4）生成MBR的genboot.pl；

（5）整个框架的makefile

3.需要完成的任务：

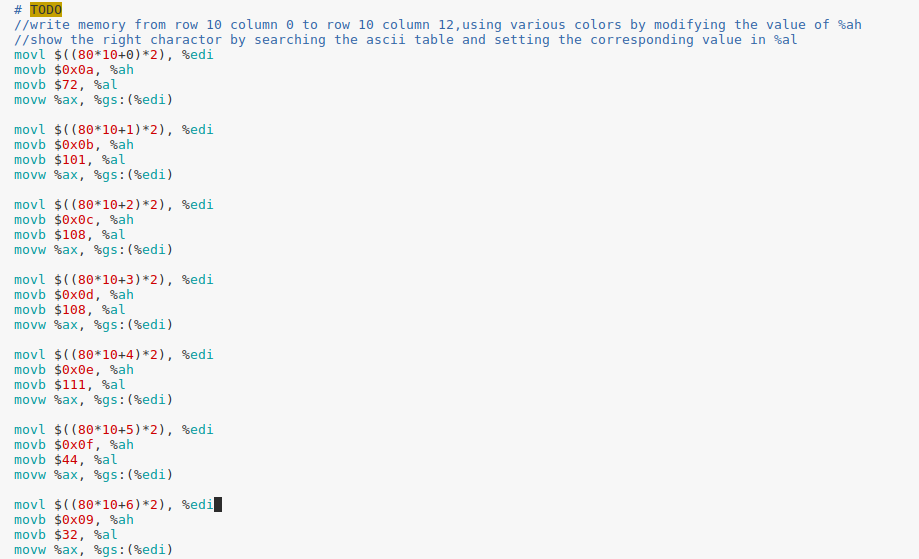
（1）app.s中具体的用户程序hello，world的实现；

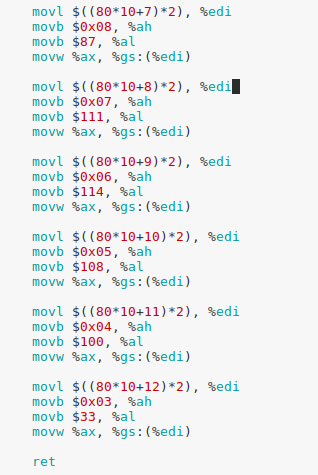
（2）bootloader中start.s里面被注释的跳转指令

4.代码思路

（1）app.s实现

在课程网站上提供了两种实现app.s的参考方案，一个是通过陷入屏幕中断调用BIOS打印字符串“Hello， World！”；另一个是通过写显存的方式打印字符。个人在本次实验选择了后者方案，具体实现截图如下





对于每个字符，设置字符的位置、颜色、ascii码

再用movw %ax，%gs:(%edi)写显存

每个字符即四行代码，一共13个字符“Hello， World！”

Ascii分别为72（H）、101(e)、108（l）、108(l)、111(o)、44(,)、32(space)、87(W)、111(o)、114(r)、108(l)、100(d)、33(!)

最后ret即可

（2）jmp跳转指令补充

只要在start.s中注释的位置jmp 到对应得函数即可，如下代码

C:\Users\吴紫航\AppData\Roaming\Tencent\Users\401986905\QQ\WinTemp\RichOle\V3$]E()HEG_`VA(Y0L~1RIE.png

1. 实验结果

首先chmod 777打开genboot.pl的权限，然后就可以make编译

用make play运行得到以下运行结果截图



1. 实验心得
2. 本次实验相对简单，但是也需要仔细的去阅读manual和网站上的guide，了解实验基本原理，和实现要求
3. 实验过程中遇到的第一个问题是无法make编译，找到原因是permission denied，当我不知道如何处理的时候，通过留心群信息得知用chmod更改文件的访问权限即可。
4. 遇到的第二个问题是没有添加jmp指令，通过仔细阅读框架代码，找到了jmp一条注释的信息，添加注释后成功得到结果！
5. 实验建议

暂无。