操作系统实验报告

Lab2

姓名：刘博

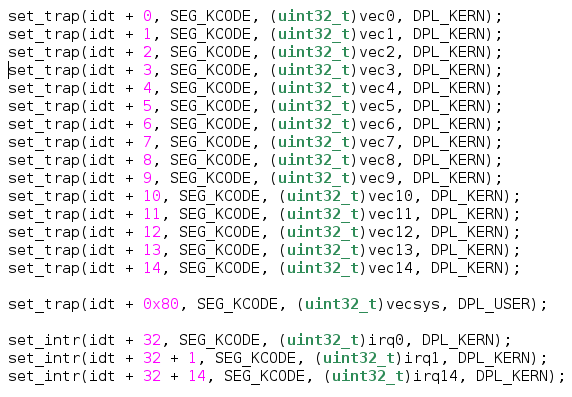
学号：141220065

计算机科学与技术系

2016.4.4

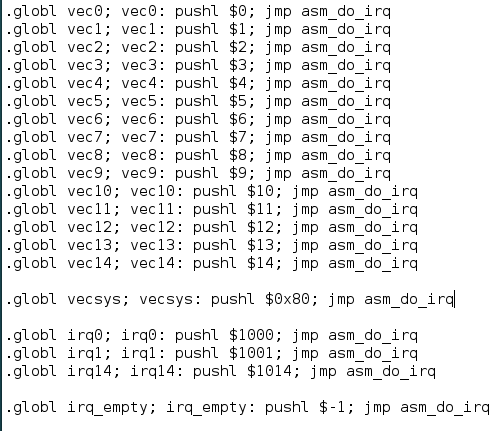
邮箱：1610266604@qq.com

1. **实验目的：**
2. 从实模式进入保护模式
3. 加载内核到内存某地址并跳转运行
4. 初始化中断向量表
5. 初始化 GDT 表
6. 配置 TSS 段
7. 从磁盘加载用户程序到内存相应地址, 并修改用户程序的各个 GDT 表项
8. 进入用户空间前的相关配置
9. 正式进入用户空间
10. 调用库函数 printf（这里我是在显存进行输出）
11. **实验过程：**
12. 首先将原lab1中的bootloader内的内容拷贝到lab2的bootloader文件夹下。完成对kernel文件的加载；
13. 然后对中断向量表进行初始化：



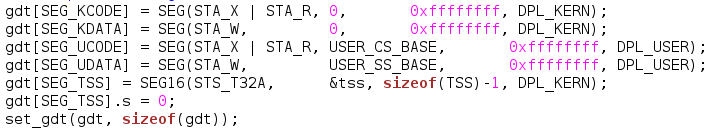
这里对14个系统陷阱进行初始化，并且加入系统调用；

然后再do\_irq.s进行函数定义：



每个中断处理函数都如上图定义。

初始化中断向量表后，继续对全局描述符表进行初始化：

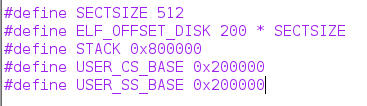


这里分成了两类，一类是kernel的全局描述符，一类是user的全局描述符，分别用dpl区分。

初始化全局描述符表后，要对tss段进行初始化，由于tss是kernel才能访问的段，所以将其特权级设为dpl\_kern；



这里stack是一个宏定义，定义如下：

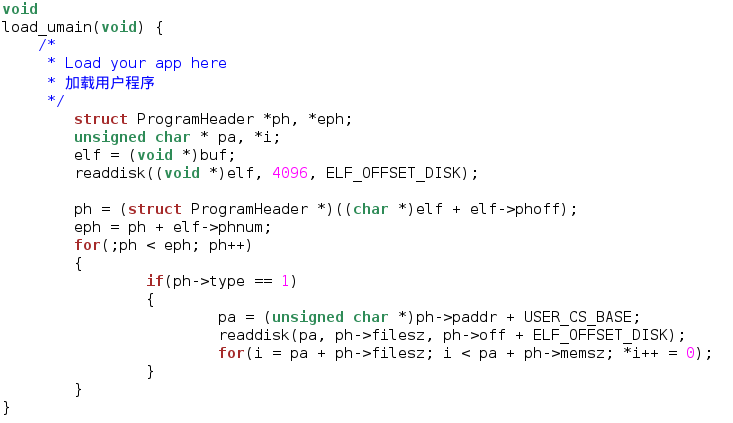


由于i386进行特权级切换时需要进行栈帧切换，而且i386进行栈帧切换时不考虑esp，只用ss段进行切换，所以这里esp可以初始化成0x800000，ss段为kernel的ss段（基址为0）

，这样kernel内核栈的起始位置就是ss0：esp0 = 0x800000；

初始化tss段寄存器后需要用ltr指令将其地址加载到tr寄存器中；

初始化段寄存器后，就可以加载用户的程序代码了：

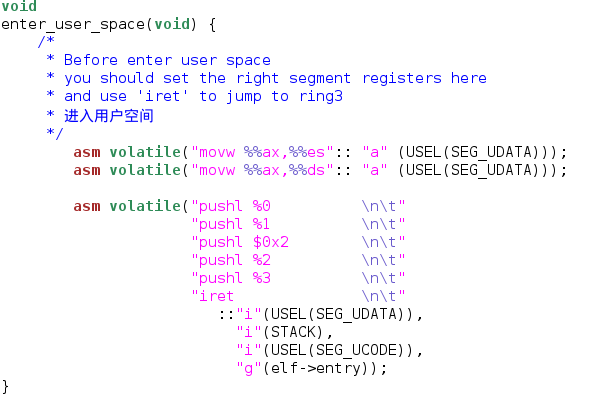


其中加载用户程序的代码如上图，这里需要注意两点：

1. 由于用户程序的存放位置是磁盘中的第202个扇区，所以加载时需要在磁盘偏移量为512\*200的位置处开始进行加载，这里将它定义为一个ELF\_OFFSET\_DISK宏；
2. 由于ph->off是相对于elf在磁盘中的偏移量，所以加载时需要在ph->off的基础上加上对应的elf的偏移量才能对正确的程序代码进行加载；

由于cpu读取代码时需要加上cs的基地址才能读取指令，所以要在加载程序代码之前对所有代码加载的基地址进行偏移（加0x200000）

正确加载用户代码后，就可以进行向用户空间的跳转了：



这里进行进入用户空间之前的特权级切换，首先需要将段寄存器更新成用户的段寄存器值，然后将用户的所有栈帧进行更新：

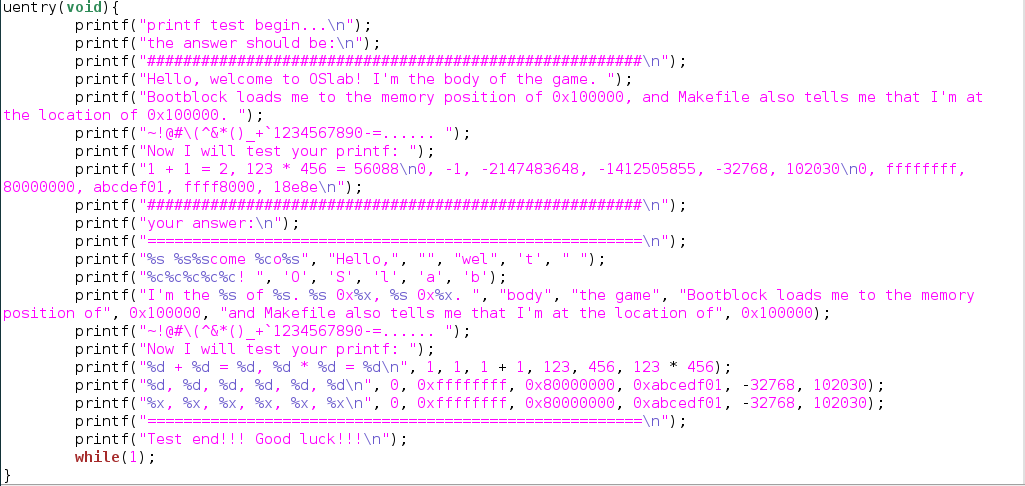
Esp = esp0；（便于栈帧切换）

Ss = usr\_ss；

Eflags = 0x2；（eflags的初始值）

Eip = elf->entry；（将用户用户代码的入口存放在eip中，这样iret后，程序代码就会自动运行）

跳转进入用户空间后，就可以进行printf输出了：



这里使用了网站上的测试用例；

Printf函数声明如下：



这里format是要输出的字符串，我们需要将其中的所有控制符进行区分，然后每遇到一个控制符便将后面的一个对应的参数输出；

提取参数的方法如下：

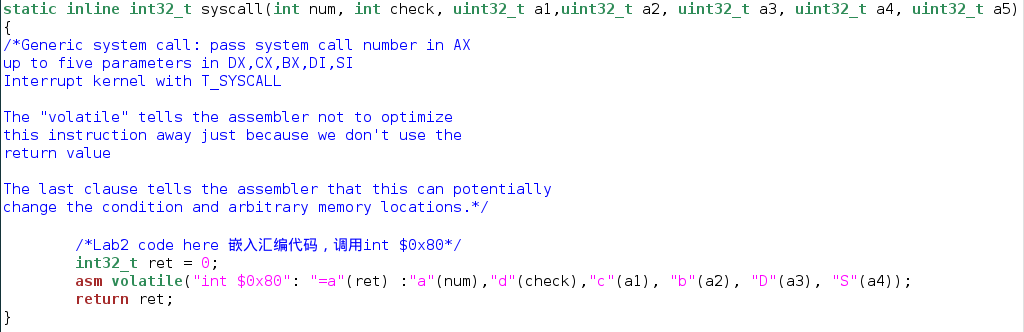


这里利用了二级指针，将format的地址提取出来，则其上一个地址就是需要输出的字符串；

即args便是指向参数列表的二级指针；

参数记为args[0],args[1],args[2]…；

Printf对所有的控制符进行区分，然后对每个控制符进行转化和整理，最后整理成一个可输出的字符串，并将其传给syscall函数：



Syscall函数将所用到的所有参数分别赋值给相应的寄存器，然后使用int指令进入中断：

参数说明：eax：系统调用号（返回值）

Edx：文件描述符

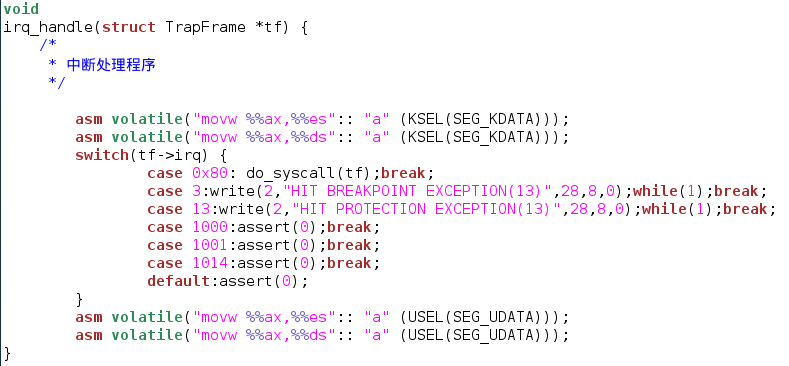
Ecx：字符串的地址

Ebx：字符串输出的长度

Edi：输出的位置（行）

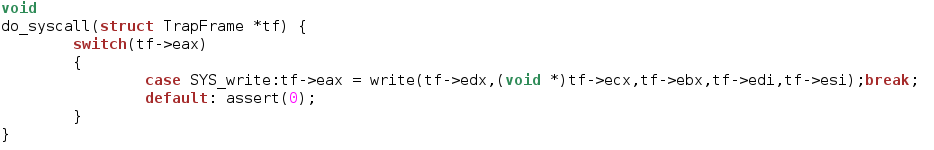
Esi：输出的位置（列）

调用int指令后，系统会自动保存当前所有寄存器的值，然后陷入内核中断：

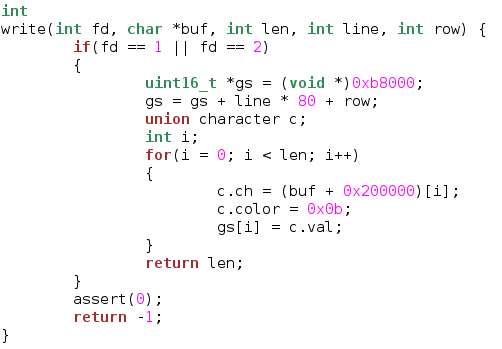


这里需要注意先将ds，es段寄存器更新（手动），因为int指令只会切换栈帧和cs，如果不进行手动切换，则系统判断eax的值会出错（变为0）；

进入中断后，根据eax的值判断属于哪种中断，根据0x80判断为系统调用，于是进入do\_syscall函数中进行系统调用；



Do\_syscall函数根据eax寄存器值判断是系统调用中的哪种函数，根据eax = 4判断属于write函数的系统调用，于是调用write函数，并且将文件描述符，字符串起始地址，字符串长度，字符串打印位置传给write函数；

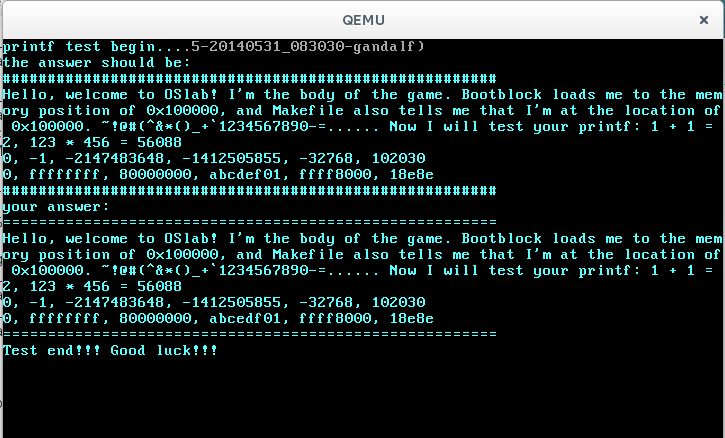


Write函数进行输出，这里可以有两种输出方式，一种是使用定义好的putchar函数进行串口的输出，另外一种是直接写显存进行屏幕输出，这里选择的是显存的输出（串口输出更为简单）

由于显存的起始地址是0xb8000，所以我们用gs指向该地址，然后将buf中的字符值输出到gs中，操作方法类似于boot；

注意这里buf的地址并不是实际上字符串的地址，而是需要在其基础地址上加上cs的偏移量才可以，否则无法读出正确的字符串；

然后就可以进行输出测试了，输出的结果如图：



至此lab2的所有操作完成；

1. **总结与感想：**

本次实验让我了解了操作系统内部真正的特权转换和段切换，从而更好的为后面的实验打下基础，不过还是希望助教能够更加清楚地进行讲解，我们也会积极提问；