|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | 臧宇萱 | **学号** | 37 |
| **实验题目** | Ucore实验lab1 | | |
| **实验内容** | 1. 练习一  * 项目组成：boot,kern,driver，Makefile，tools等等 * 查看makefile的组成：     输入make clean，再输入make，得到的实验结果如上图所示    这是输入make V=得到的结果，从中可以看出其中执行的命令，可以看出得到的结果更加详细   * 生成过程：   首先我们要得到bootmain.c和bootasm.s还有sign，生成了bootblock.o之后再生成bootblock，在kernel中需要用GCC编译器将kern目录下所有的.c文件全部编译生成的.o文件的支持，生成kernel模块之后，生成一个有10000块的文件，每个块默认512字节，     * Sign作用：检查   引导扇区的大小为512字节，最后两个字节为标志性结束字节0x55，0xAA，做完这样的检查才能认为是符合规范的磁盘主引导扇区   1. 练习二   修改gdbinit，先make debug，得到gdb的调试界面，从第一条指令开始执行单步跟踪BIOS  1  查看当前的BIOS代码  2  在初始化位置设置实地址断点，测试断点，由于粘贴格式有误，所以采用直接手敲的方式进行，断点正常，跟踪运行，并将反汇编得到的代码与bootasm.S和 bootblock.asm进行比较，得到相同的部分，如下图：  5   1. 练习三  * 关闭中断，将各个段寄存器重置      * 将A20置1，即开启A20，由于保护模式下 A20 地址线控制是要打开的，所以需要通过将键盘控制器上的A20线置于高电位，使得全部32条地址线可用      * 加载GDT表      * 将CR0第0位置1      * 长跳转到32位代码段，重装cs和eip      * 重装DS、ES等段寄存器      1. 练习四     首先由readseg函数对硬盘扇区进行读取，而readseg函数则循环调用了真正读取硬盘扇区的函数readsect来每次读出一个扇区，这是bootloader读取硬盘扇区的阶段    读完磁盘以后，开始加载elf格式的文件，首先要判断是不是elf，头部有描述elf文件应加载到内存什么位置的描述表，读取出来之后存在ph，按照描述载入内存，根据入口信息，找到内核入口开始运行   1. 练习五  * 通过read\_ebp()和read\_eip()函数来获取当前ebp寄存器和eip 寄存器的信息。之后通过ebp+12,ebp+16,ebp+20,ebp+24来输出4个参数的值，最后更新ebp：ebp=ebp[0],更新eip：eip=ebp[1]。直到ebp 对应地址的值为0，实现函数如下：      * 执行make qemu结果后如下：      1. 练习六  * 一个表的结构如下：     中断描述符表（也可简称为保护模式下的中断向量表）中一个表项占 8 个字节,2-3 字节是段选择子，0-1 字节和 6-7 字节拼成位移（offset），两者联合便是中断处理程序的入口地址   * 编程完善trap.c     首先，声明一个vectors数组，通过这个指针可以跳转到该中断处理的地点，setgate这个函数的作用是设置正确的interrupt/trap gate 描述符，然后，填充idt   * 补充剩余代码     得到运行结果：    实验结束   1. Challenge   目的是完成init.c中的switch\_to\_user和switch\_to\_kernel和trap.c中trap\_dispatch()的case T\_SWITCH\_TOU和case T\_SWITCH\_TOK四个函数  首先是init.c中的switch\_to\_user是用来从内核态切换到用户态的。从中断返回时，会多两位，并用这两位的值更新ss、sp，所以要先把栈压两位预留一段 8 字节空间用来给之后 ss 和 esp 的入栈。  调用了T\_SWITCH\_TOU中断，在trap.c中完善该函数  其次是init.c中的switch\_to\_kernel是用来从用户态切换到内核态的  最后是case T\_SWITCH\_TOK，恢复数据 | | |
| **总结** | 1. 初步了解ucore的“项目组成”； 2. 学会使用make编译文件，初步了解Makefile的组成，懂得使用make 语句查看编译信息； 3. 尝试使用了qemu和gdb共同配合对代码进行调试，并能够设置断点进行测试； 4. 了解了实模式转换到保护模式的过程和elf文件加载； 5. 初步了解了函数print\_stackframe的使用； 6. 能够在实际应用中找到中断描述符表并找到处理代码的入口； 7. 学会使用print\_ticks子程序； | | |
| **日期** | 2020.05.31 | **成绩** |  |