|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | 常昊 | **学号** | 4062017002 |
| **实验题目** | Lab2 | | |
| **实验内容** | 练习1   1. 了解项目组成，包括boot,kern,driver，Makefile，tools等等。 2. 输入make clean,然后输入make，可以看到如下代码：  输入make V=，显示出更详细的信息，注意，在输入之前要再次执行make clean。 3. 生成ucore.img的过程：首先先创建一个大小为10000字节的块儿，然后再将bootblock拷贝过去。 生成ucore.img需要先生成kernel和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；在bootblock中首先需要生成bootasm.o、bootmain.o、sign。   练习2   1. 补全lab1-mon的代码。  执行命令：make lab1-mon，   练习3   1. 将各个寄存器置0，关闭中断 2. 开启A20，使得全部32条地址线可用。 3. 加载GDT表。 4. 将CR0的第0位变成1. 5. 长中转到32位代码段，重装CS和EIP。      1. 重装DS、ES等段寄存器。 2. 转到保护模式完成，进入boot主方法。   练习4   1. Bootloader读取硬盘扇区 2. bootloader加载 ELF格式的 OS   练习5   1. 补全代码。 2. 在这个文件夹中执行make qemu，可以得到输出结果，结果中包括了edp,eip，它们的参数与函数名。其调用关系中，函数调用栈：上一层函数调用时的ebp值”,而在每一层函数调用中,都能通过当时的ebp值“向上(栈底方向)”能获取返回地址、参数值,“向下(栈顶方向)”能获取函数局部变量值。如此形成递归,直至到达栈底。   练习6  1. 中断描述符表一个表项占8字节。其中0~15位和48~63位分别为offset的低16位和高16位。16~31位为段选择子。通过段选择子获得段基址，加上段内偏移量即可得到中断处理代码的入口。  2.补全代码。 补全代码，执行make qemu。可以得到如下界面： | | |
| **总结** | （通过练习掌握到的知识，按条目列出。）   1. 初步了解make命令，以及gcc 是如何生成ucore的大致步骤。 2. 初步掌握了如何用qemu与gdb对ucore代码进行调试。 3. 了解了bootloader 加载ELF文件的大体步骤。 | | |
| **日期** | 2020.06.01 | **成绩** |  |