|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | 金海波 | **学号** | 4062017012 |
| **实验题目** | 实验lab1 | | |
| **实验内容** | 练习1   1. 了解项目组成，包括boot，kernel等等。      1. 输入make clean将之前编译的内容删除,然后输入make，结果如图： 2. 首先把C的源代码编译为目标文件，再用ld命令将这些目标文件变成可执行文件，最后用dd命令把bootloader放到ucore.img.count的虚拟硬盘中。sign.c文件起检查作用，从sign.c的代码中能看出，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，这样才能认为是符合规范的磁盘主引导扇区。   练习2   1. 将lab1-mon代码补充完整  在终端用cd命令进入labcodes/lab1目录，输入指令：make lab1-mon，结果如图。   测试断点正常  练习3   1. 把寄存器置0，关中断 2. 开启A20，使32条地址线都可用。 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。 6. 重装DS、ES等段寄存器。 7. 转到保护模式，进入bootmain。   练习4   1. 用Bootloader读取硬盘扇区 2. 用bootloader加载 ELF格式的 OS   练习5   1. 补全代码。 2. 在该文件夹中执行make qemu，得到结果，结果中包括了edp,eip。ebp和eip的调用关系是首先将ebp寄存器入栈，然后将栈顶指针eip赋值给ebp。“mov ebp eip”这条指令表面上看是用eip覆盖ebp原来的值，其实不然。因为给ebp赋值之前，原ebp值已经被压栈（位于栈顶），而新的ebp又恰恰指向栈顶。此时ebp寄存器就已经处于一个非常重要的地位，该寄存器中存储着栈中的一个地址（原ebp入栈后的栈顶），从该地址为基准，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，而该地址处又存储着上一层函数调用时的ebp值。   练习6  1. 中断描述符表一个表项8字节。0-15位和48-63位为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，再加上段内偏移量即可得到中断处理代码的入口。如图：  2.补全代码。  首先声明\_\_vertors[],其中存放着中断服务程序的入口地址。  接着填充中断描述符表IDT。   1. 补全代码，执行make qemu指令。结果如下：   成功 | | |
| **总结** | （通过练习掌握到的知识，按条目列出。）   1. 了解了ucore的项目组成。 2. 理解了make命令，以及gcc 是如何生成ucore的大致步骤。 3. 掌握了如何用qemu与gdb对ucore代码进行调试。 4. 了解了bootloader 加载ELF文件的步骤。 5. 了解ucore中断机制的实现流程。 | | |
| **日期** | 2020.06.01 | **成绩** |  |