|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | 邹\*波 | **学号** | 44 |
| **实验题目** | Ucore实验Lab1 | | |
| **实验内容** | 1.（练习1）了解ucore的“项目组成”。  2.（练习1）会使用make编译文件，了解Makefile中的主要组成，实验报告中要有简单介绍。查看使用make与make “V=”所生成信息的不同。 根据make编译信息，简要说明gcc编译器是如何一步步生成ucore可执行文件的。sign.c的作用是什么。  Makefile就相当于是一个批处理命令，将一系列的.c和.h文件编译成可执行代码，是gcc命令的一个说明，即告诉gcc应该怎么去编译，编译方式等内容。  Makefile：    而make命令正是按照makefile文件说明而进行执行的命令，可以编译出许多可执行文件出来，本身make命令并不会把编译过程显示出来，但是加上“V=”后即可显示出编译的具体过程。  Make:    Make V=:    Gcc编译生成ucore可执行文件的过程：  先生成kernel的.o汇编文件然后使用ld进行链接生成kernel可执行文件，然后再用同样的方法再进行bootblock的编译链接生成bootblock.o可执行文件，再填充ucore.img至512字节，最后将kernel和bootblock放到ucore.img中，这样就生成了ucore可执行文件。  Sign.c具有标记功能，作用是做一个引导扇区的特征标志。  3.（练习2）学会使用qemu与gdb协作进行调试ucore代码；从CPU加电后执行的第一条指令开始，单步跟踪BIOS的执行。在初始化位置0x7c00设置实地址断点,测试断点正常。从0x7c00开始跟踪代码运行,将单步跟踪反汇编得到的代码与bootasm.S和 bootblock.asm进行比较。自己找一个bootloader或内核中的代码位置，设置断点并进行测试。（截图并简要说明实验过程）  加电执行第一条指令：    在0x7c00设置实地址断点：          输入C继续执行：    输入C再继续执行：      重新在0x8c00设置实地址断点：、    输入命令make debug启动执行：    输入C继续执行：    再次输入C继续执行：    4.（练习3）了解如何在bootloader中完成由实模式到保护模式的转换的。（要有过程或对应代码的截图）   1. 首先清理环境：包括将flag置0和将段寄存器置0 2. 开启A20：通过将键盘控制器上的A20线置于高电位，全部32条地址线可用，可以访问4G的内存空间。      1. 初始化GDT表：一个简单的GDT表和其描述符已经静态储存在引导区中，载入即可      1. 进入保护模式：通过将cr0寄存器PE位置1便开启了保护模式，通过长跳转更新cs的基地址，设置段寄存器，并建立堆栈，最后转到保护模式完成，进入boot主方法，开始执行bootmain.c里的命令。     5.（练习4）根据代码，通过阅读bootmain.c，了解bootloader如何加载ELF文件。（需要截图，并简要说明）    6.（练习5）要求完成函数kern/debug/kdebug.c::print\_stackframe的实现.完成kdebug.c中函数print\_stackframe的实现。（需要make qemu 后的结果等截图，简要说明打印信息，说明调用关系。）      从bootmain开始一步步调用函数，bootloader设置的堆栈从0x7c00开始，使用call bootmain转入bootmain函数。  7.（练习6）从代码找出中断描述符表（也可简称为保护模式下的中断向量表）的定义，并简要说明中断描述符表中一个表项占多少字节？各分别表示什么？其中哪几位代表中断处理代码的入口？（截图并简要说明）    具体的数据结构gatedesc定义在mmu.h中：    一个表项占8字节，2-3字节是段选择子，0-1字节和6-7字节拼成偏移量，通过段选择子去GDT中找到对应的基地址，再加上偏移量即可找到中断处理程序的地址所在。  8.（练习6）请编程完善kern/trap/trap.c中对中断向量表进行初始化的函数idt\_init。在idt\_init函数中，依次对所有中断入口进行初始化。使用mmu.h中的SETGATE宏，填充idt数组内容。每个中断的入口由tools/vectors.c生成，使用trap.c中声明的vectors数组即可。（截图并简要说明）    9.（练习6）请编程完善trap.c中的中断处理函数trap，在对时钟中断进行处理的部分填写trap函数中处理时钟中断的部分，使操作系统每遇到100次时钟中断后，调用print\_ticks子程序，向屏幕上打印一行文字”100 ticks”。 （截图并简要说明）  完善trap函数：    运行后的打印结果：    10.参考答案labcodes\_answer/lab1,并在labcodes/lab1中完成challenge1内容，并简要说明实现的过程（设计到哪些函数，分别在哪个文件中，执行中断的过程中，先后使用了哪些函数。）。      使用了lab1\_print\_cur\_status()、lab1\_switch\_to\_user()、lab1\_switch\_to\_kernel、lab1\_switch\_test函数，这些函数在kern/init.c文件中，发生中断时，打印当前状态，先进入用户态后打印当前状态，在用户态找到内核堆栈，然后再转成内核态打印当前状态，内核响应，来进行中断处理，最后执行中断完成后再转回到用户态继续执行。 | | |
| **总结** | 1. 基本了解了ucore项目组织的形式 2. 学会了使用gdb对汇编指令执行进行调试 3. 学会看makefile文件，利用make命令以及加上各种选项进行编译或调试 4. 了解了ucore镜像文件的整个加载过程以及组成 5. 了解了执行中断的过程以及在过程中用到的一些函数 6. 理解了os内核的一个启动流程 | | |
| **日期** | 2020.6.1 | **成绩** |  |