OSLab Project1:Bootloader

严贝 2019K8009937001

1. call 指令的使用

在 S_core 的 bookblock.S 编写中，跳转大多使用了 call 指令，功能正常。但在 A_core 的 bootblock.S 中出现了 bug。在 debug 的过程中，才发现了自己对 call 指令的认知存在一些误区。call 指令是根据标号在段中的偏移地址进行跳转的，而 A_core 中跳转到 kernel 前需要再次拷贝 bootblock，在新的地址重新启动 bootblock，所以在此处应使用 j、jr 等指令。以后的实验中，对 call 指令的使用要更加注意。

2. fence.i 指令的使用

fence.i 指令用于刷新指令 cache，每次通过 SD_READ 拷贝的时候是通过数据 cache 进行的，可能指令 cache 和数据 cache 未同步出现错误，所以需要通过 fence.i 刷新指令 cache，使其重新拉取数据。在 A_core 和 C_core 的编写中，在拷贝 bootblock 后，都进行了刷新。在测试中，发现虽然通过了 qemu 测试，但上板测试时似乎一直在重复读取 SD 卡。后来经过助教的提醒才发现拷贝 kernel 后未再次刷新，而 qemu 上由于没有模拟 cache，因此不会出现这样的错误。

3. gdb 的使用

在前期调试代码的过程中，因为对 gdb 不太熟悉使用的比较少，所以 debug 的效率也比较低。后期因为代码结构更加复杂，通过反汇编和运行结果大致定位错误比较困难，对 gdb 的使用增加，但对大部分的功能还是比较陌生，希望之后课程中有空的时候可以多增加一些 gdb 调试方法的介绍。

4. Makefile 的编写

这次实验 A_core 和 C_core 的 Makefile 需要自己修改，在编写过程中对 Makefile 脚本文件的理解更加深入，了解掌握了一些相关知识。