Project1 Bootloader设计文档

中国科学院大学

葛忠鑫

2020/9/27

# Bootblock设计

（1）Bootblock主要完成的功能

将操作系统代码搬运到内存。

（2）Bootblock如何调用SD卡读取函数

$a0~$a7寄存器是用于存放函数调用参数的，在调用read\_sd\_card函数时，将它的三个参数移动目的地址dest，SD卡读取kernel位置的偏移量offset和读取数据的大小size分别存入$a0，$a1和$a2。未实现重定位时，要将内核加载到Boot Loader后面，所以dest为0xffffffffa0800200（即kernel的值）；Boot Loader有512个字节，存放在SD的第一个扇区，其后kernel相对SD卡的偏移量offset即为0x200（=512）；在仅实现小核（一个扇区足够存放）加载的时候，只需读取一个扇区大小的size=0x200即可。

read\_sd\_card所用参数准备好后，利用jal指令跳转至其所在地址执行。



（3）Bootblock如何跳转至kernel入口

将kernel的起始地址0xffffffffa0800200（kernel\_main）写入临时寄存器，然后执行跳转指令。



（4）在设计、开发和调试bootblock时遇到的问题和解决方法

对MIPS64指令集不熟悉，不知道说明命令还可以用，什么命令需要扩展位宽。后面老师发的MIPS64指令手册解决了一些困惑，不知道的时候查指令集手册看看具有所需功能指令的具体用法。

# Createimage设计

（1）Bootblock编译后的二进制文件、Kernel编译后的二进制文件，以及写入SD卡的image文件这三者之间的关系

写入SD卡的image文件由两部分组成，第一部分是Bootblock编译后的二进制文件中的程序段部分，需要存放在SD卡中的第一扇区，即image的前512比特；第二部分是 Kernel编译后的二进制中的程序段部分，在SD卡中的第一扇区后面的一个或多个扇区，即image和512比特位置以后。

（2）如何获得Bootblock和Kernel二进制文件中可执行代码的位置和大小，你实际开发中从kernel的可执行代码中拷贝了几个segment？

Bootblock和Kernel二进制文件开头都是ELF头和程序头，ELF头结构体中的e\_phoff变量表示程序头里ELF头的偏移量，可以找到程序头的位置，程序头结构体中的p\_offset和p\_filesz分别表示可执行代码段的偏移量和大小。

（3）如何让Bootblock获取到Kernel的大小，以便进行读取

将kernel的大小（双字节，以扇区为单位）写到第一个扇区的末尾的一个固定地址中，供Boot Loader执行时使用。

（4）在设计、开发和调试createimage时遇到的问题和解决方法

对文件读写操作不熟悉，需要反复查阅函数使用方法。

# A-Core/C-Core设计

你设计的bootloader是如何实现重定位的？如果bootloader在加载kernel后还有其他工作要完成，你设计的机制是否还能正常工作？

函数调用后的返回地址存放在$ra寄存器中，可以通过修改read\_sd\_card函数的返回地址以实现重定位，即在跳转（jr）至read\_sd\_card前给$ra寄存器写入正确的kernel的起始地址。具体代码如下：



其中kernel和kernel\_main地址改为覆盖Boot Loader后的起始地址0xffffffffa0800000，kernel\_size为存放在Boot Loader末端的内核大小地址0xffffffffa08001fc。

如果加载kernel后还有其他工作，在上述方法下就不能正常工作了。如果有其他工作需要完成，可以先将Boot Loader内容拷贝至内存中kernel的后面（因为在这个可以加载大核的设计中kernel大小可以提前读出），然后根据机器指令找到拷贝后未执行的Boot Loader代码段的起始位置继续执行（也可以根据机器指令只拷贝未执行的代码部分），在执行完Boot Loader后将该块内存清空也可实现重定位以避免资源浪费。

重定位要注意修改Makefile中main的-Ttext参数为0xffffffffa0800000。

# 关键函数功能

1. write\_os\_size函数



kernel\_size两个字节以小端序存放在0xffffffffa08001fc起始的两个比特内，方便Boot Loader使用lw指令直接取出。这里要注意根据image总的写入字节数nbytes计算kernel大小时需要减去Boot Loader的那一个扇区。

1. write\_segment函数



注意存储写入的扇区数first的初始值为0且向上取整，同时，存储写入扇区的字节数nbytes计算时要以SECTOR\_SIZE（0x200）为单位向上取整。

为初始化bss以及填0补齐扇区，直接将filesz到扇区结束全部赋0。

1. kernel部分代码



结合C语言的指针调用函数并查看串口状态，以完成打印与回显任务。

参考文献

1. The MIPS64 Instruction Set Reference Manual, Revision 6.06

▄