# Project1 Bootloader 设计文档

中国科学院大学

刘杰

2018.9.26

## 1. Bootblock 设计流程

（1）Bootblock 主要完成的功能是将操作系统的代码搬运到内存。
（2）Bootblock 被载入内存后通过调用 BIOS 函数 read_sd_card 读取 SD 卡上操作系统的内核，并放置到内存的制定位置 0xa0800200，读取完内存后，Bootblock 最后跳转到内核代码入口处开始执行。
（3）在 bootblock.s 汇编文件中加入 read_sd_card 函数调用相关指令实现读取 SD 卡上内核到内存指定位置，具体实现是先用三个 LI 指令把 read_sd_card 函数的三个参数顺序存入 a0-a2 这三个寄存器中，再使用 JAL 指令跳转到 read_sd_card 函数的入口地址。
（4）在 bootblock.s 汇编文件加入调用 read_sd_card 函数相关指令后，再使用 JAL 指令跳转到内核镜像的入口地址，是在内核文件的最前面，放到内存后地址为 0xa0800200。
（5）在 PMON 界面，输入 loadboot 后重复打印 HELLO OS，因为一开始我使用的是 JR 指令跳转到内核镜像的地址，此时 31 寄存器保存的是执行 JR 前一条 JAL 指令的返回地址，所以从内核镜像返回后又回到 JR 指令执行，陷入死循环。把 JR 指令改为 JAL 指令即可。

## 2. Createimage 设计流程

（1）Bootblock 编译后的二进制文件、Kernel 编译后的二进制文件都是 ELF 文件，在最终生成的 image 镜像中不包含这两个 ELF 文件中用于分析代码和数据位置的 header，只保留 segment 部分。
（2）打开 ELF 文件后，先读入 ELF header，再通过 ehdr 结构体中 e_phoff 找到程序头位置，读入 programs header，在 phdr 结构体中 p_offset 找到可执行代码位置，p_filesz 得到可执行代码大小。
（3）通过调用函数计算 kernel 的扇区数，并调用函数写入 bootblock 的 os_size 位置来通知 bootblock 内核的大小，方便读取 kernel。
（4）写入 bootblock 时应该在 image 偏移量为 0x1fe 依次写入 0x55，0xAA 确保写入的 bootblock 块为 512B 大小。

## 3. 关键函数功能

1. bootblock.s 代码片段

```
main:
        # 1) task1 call BIOS print string "It's bootblock!"
la      $a0, msg            //写入 printstr 参数
jal     0x8007b980          //调用 printstr 函数

        # 2) task2 call BIOS read kernel in SD card and jump to kernel start
li      $a0, 0xa0800200
li      $a1, 0x00000200
li      $a2, 0x00000090  //依次写入 read_sd_card 三个参数
jal     0x8007b1cc          //调用 read_sd_card 函数
jal     0xa0800200          //跳转至内核入口地址
# while(1) --> stop here
```

2. createimage 代码片段

```
Elf32_Phdr *read_exec_file(FILE *opfile)
{
   uint8_t numread, num_phdr;

   Elf32_Ehdr ehdr;
   Elf32_Phdr *phdr = NULL;

      //read in elf header
   fseek(opfile, 0, SEEK_SET);
   numread=fread(&ehdr, 1,sizeof(Elf32_Ehdr), opfile);
   assert(numread == sizeof(Elf32_Ehdr));
   num_phdr = ehdr.e_phnum;

      //read in program headers
   assert(ehdr.e_phentsize == sizeof(Elf32_Phdr));
   fseek(opfile, ehdr.e_phoff, SEEK_SET);              //set read position to phdr
   phdr = calloc(sizeof(Elf32_Phdr), num_phdr);        //allocate memmory to store phdr

   numread = fread(phdr,    sizeof(Elf32_Phdr),num_phdr, opfile);
   assert(numread == num_phdr);

   return phdr;
}
```

```
void write_bootblock(FILE *image, FILE *file, Elf32_Phdr *phdr)
{

    size_t numread;
    char buffer[512];

    /* set file offsets */
    fseek(file, (*phdr).p_offset, SEEK_SET);
    numread=fread(buffer, 1, (*phdr).p_filesz, file);
    assert(numread == (*phdr).p_filesz);

    // write bytes
    fseek(image, 0, SEEK_SET);
    numread = fwrite(buffer, 1, (*phdr).p_filesz, image);
    assert(numread == (*phdr).p_filesz);
    //write terminator
    fseek(image, 0x1fe, SEEK_SET);
    fputc(0x55, image);
    fputc(0xAA, image);
}
```