Lab1

梁峻滔　PB19051175

2021 年 4 月 24 日

一、初始内存盘

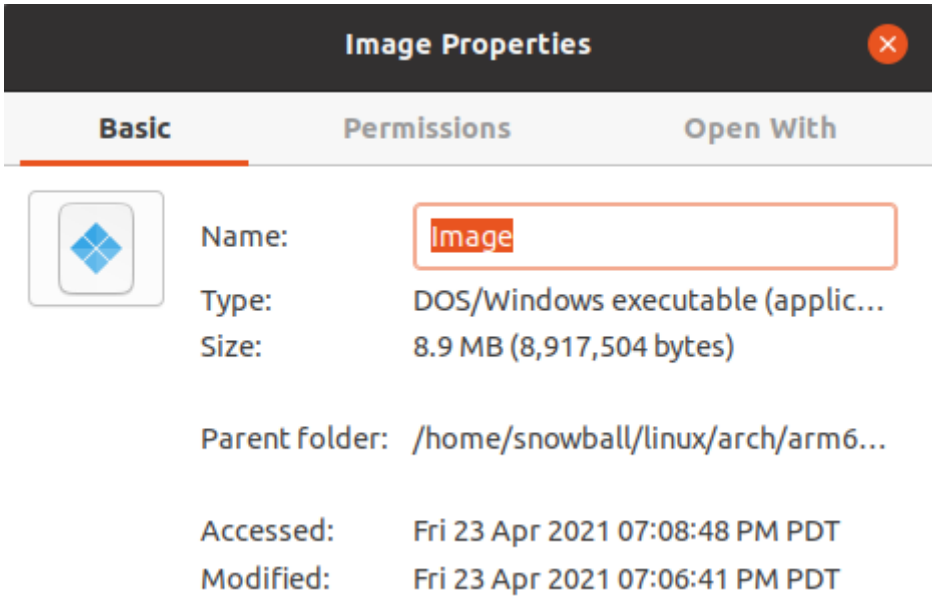
- 构建 `initrd.cpio.gz` 文件，见github。
- init.c

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<unistd.h>
#include<sys/sysmacros.h>
#include<sys/wait.h>

int main()
{
    //create device files
    if(mknod("./null", S_IFCHR | S_IRUSR | S_IWUSR, makedev(1, 3)) == -1)
    {
        perror("mknod() failed");
    }
    if(mknod("/dev/ttyS0", S_IFCHR | S_IRUSR | S_IWUSR, makedev(4, 64)) ==
-1)
    {
        perror("mknod() failed");
    }
    if(mknod("/dev/ttyAMA0", S_IFCHR | S_IRUSR | S_IWUSR, makedev(204, 64))
== -1)
    {
        perror("mknod() failed");
    }
    if(mknod("/dev/fb0", S_IFCHR | S_IRUSR | S_IWUSR, makedev(29, 0)) == -1)
    {
        perror("mknod() failed");
    }
    printf("here\n\n");
    //call 3 test procedures
    if(fork() == 0)
    {
        if((execl("/tools/binary/1","1","execl",NULL)) == -1)
        {
            perror("execl");
            exit(1);
        }
    }
```

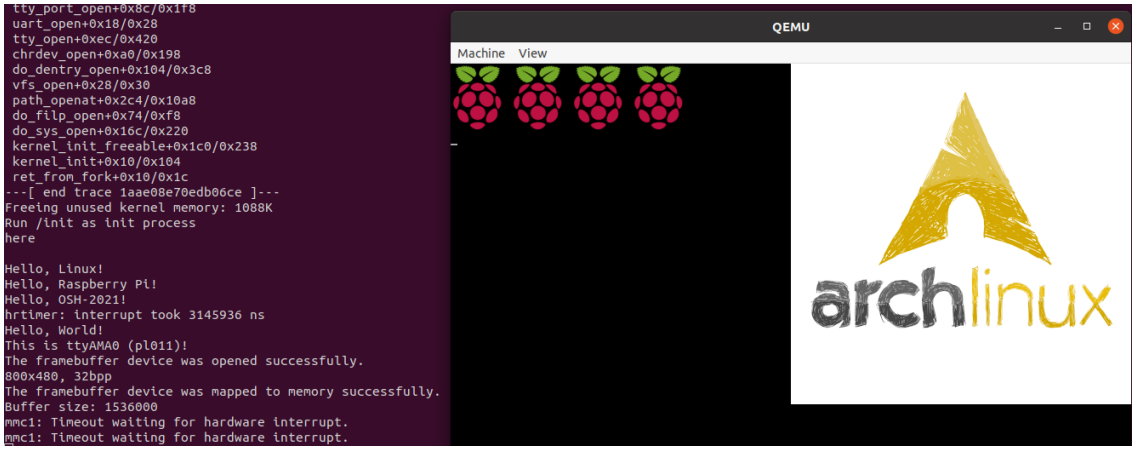
```
    sleep(3);
    if(fork() == 0)
    {
        if((execl("/tools/binary/2","2","execl",NULL)) == -1)
        {
            perror("execl");
            exit(1);
        }
    }
    sleep(3);
    if(fork() == 0)
    {
        if((execl("/tools/binary/3","3","execl",NULL)) == -1)
        {
            perror("execl");
            exit(1);
        }
    }
    while(1);
    return 0;
}
```

二、Linux 内核和执行测试程序

- 编译适用于树莓派的内核和裁剪内核

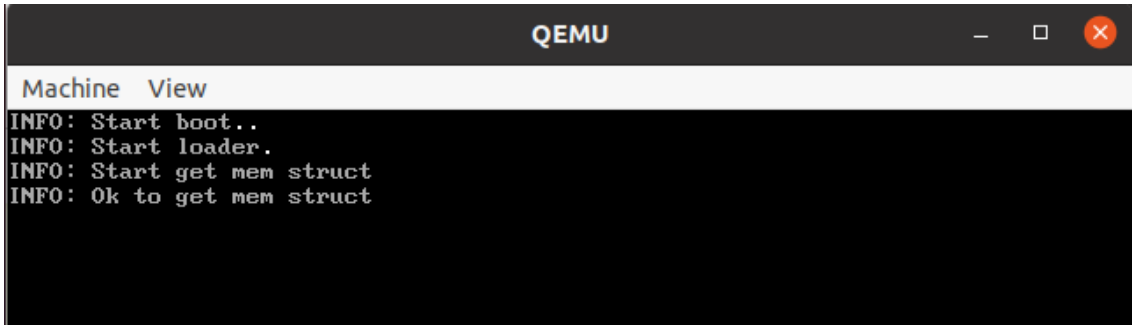


- 执行测试程序



三、初识Boot

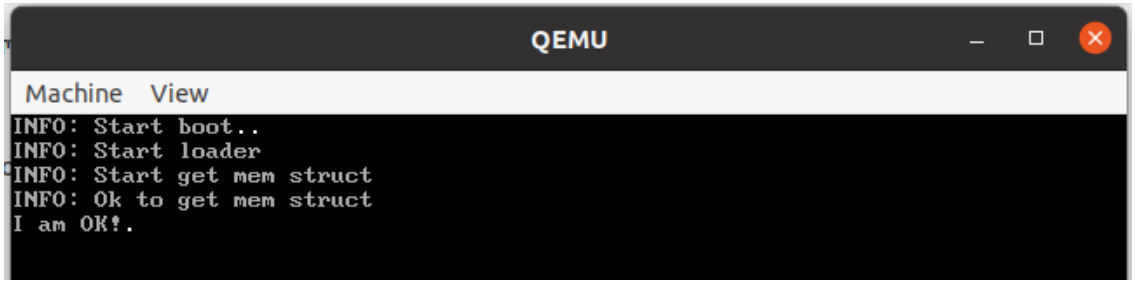
- 构建 `bootloader.img` 文件，见github。
- `xor ax, ax` 将 `ax` 清零，这样的清零操作在执行过程中只需读取一个寄存器放到ALU中运算，不需要占据额外的存储空间，而且执行速度快。
- `$` 表示当前指令的地址，`jmp $` 就是不断跳转到当前地址。
- 初始输出：



在loader.asm中新增代码：

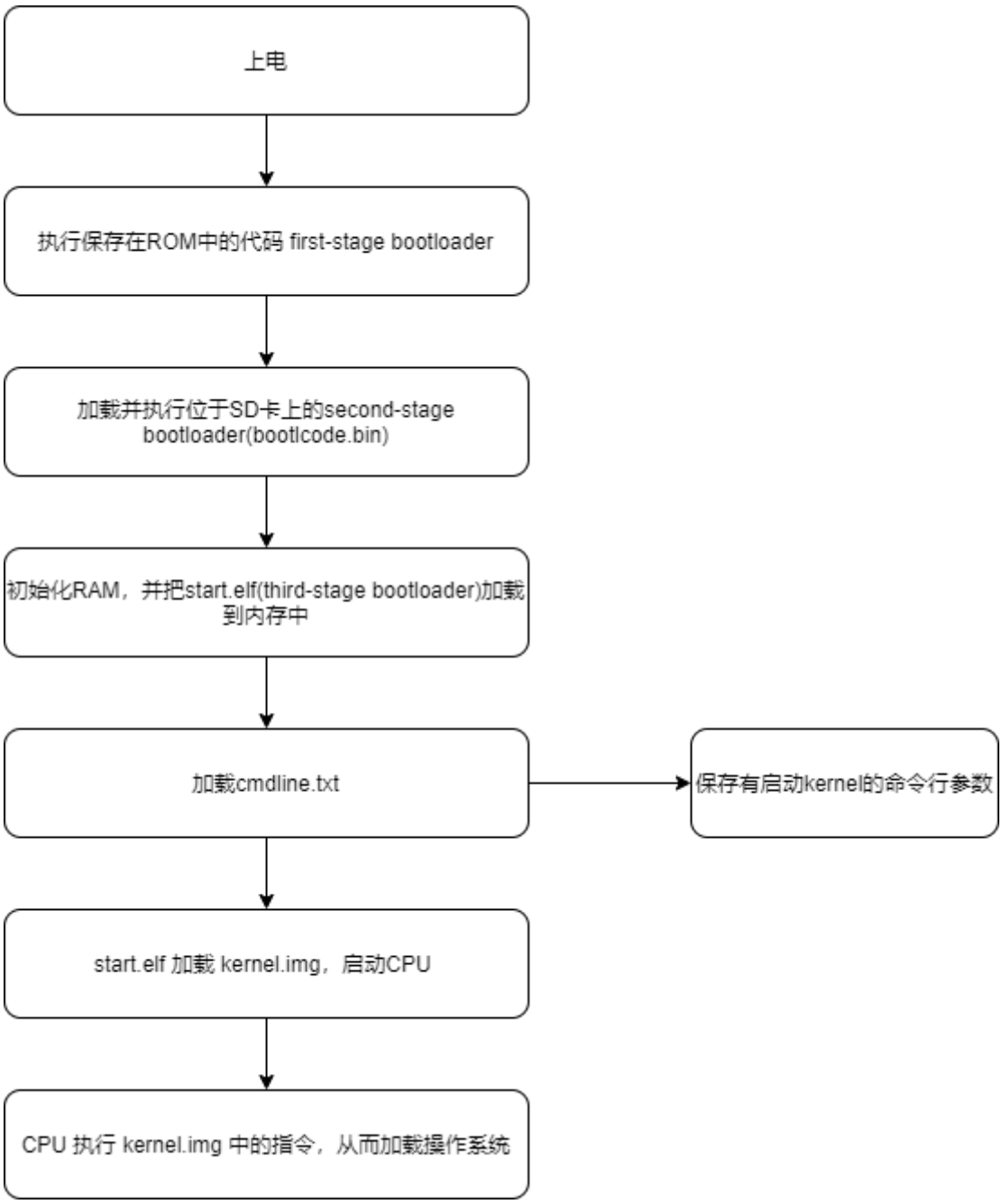
```
; Print "I am OK"
log_info AddPrint, 8, 4

AddPrint: db 'I am OK!'
```

新增输出：



四、思考题

1. `Linux` 是一种操作系统，而 `Ubuntu`、`Debian`、`ArchLinux`、`Fedora` 是不同的 `Linux` 发行版，包含 `Linux` 内核和支撑内核的实用程序和库，通常还带有大量可以满足各类需求的应用程序。
2. 本实验不需要把内核装到SD卡上，因为本实验所运行的三个测试程序虽然需要通过内核来运行，但并不需要在树莓派上运行，把内核装到SD卡上是为了能把内核装载到树莓派上。这样，本实验在进行内核裁剪时就可以把与SD卡相关的驱动等移除或者改为模块，而模块是可以按需随时装入和卸下的，有助于减小内核大小。
3. 树莓派启动过程：



4. pass

5.
   - qemu 在 `user mode` 配置方式下，可以运行跟当前平台指令集不同的平台可执行程序，跨指令集是 qemu 模拟器本身的一个特点(优势)，即可以用 `qemu-user` 在 x86 上运行 ARM 的可执行程序，只要两个平台是同一种操作系统，在这里就是 Linux。将一个 ARM 程序传入 qemu 模拟器中，qemu 把程序中的 ARM 指令翻译成 x86 指令(需要交叉编译环境)，然后在 x86 的CPU中执行。
   - `user mode` 和 `system mode` 是 qemu 的两种配置方式。qemu 在 `system mode` 配置下模拟出整个计算机，可以在 qemu 上运行一个操作系统；qemu 在 `user mode` 配置方式下，可以运行跟当前平台指令集不同的平台可执行程序。`qemu-system` 用于模拟运行操作系统，`qemu-user` 则用于运行可执行用户程序。