lab1实验报告

思考题

Thinking 1.1

反汇编结果

使用MIPS交叉编译工具链得到hello.o的反汇编如下：

如图，和即将要跳转到的t9地址有关的立即数都是0

而使用MIPS交叉编译工具链得到hello的反汇编如下：

此时，原本的0都换成了真实的地址，可以看到链接器的工作是将正确的地址填入。

objdump参数含义

```
objdump -DS hello > mipsout3
```

中-D选项的意思是强制返汇编所有可执行段

不使用-D反汇编出来的内容明显少于使用-D

-S选项的意思是使用源代码和汇编代码交叉显示，如下图。

但是显示的前提是编译的时候用了-g选项加入调试信息。

如果不加-g选项，-S几乎没有效果

```
#include <stdio.h>
int main()
{
    1149:       f3 0f 1e fa             endbr64
    114d:       55                      push   %rbp
    114e:       48 89 e5                mov    %rsp,%rbp
        printf("Hello World!\n");
    1151:       48 8d 05 ac 0e 00 00    lea    0xeac(%rip),%rax
_stdin_used+0x4>
    1158:       48 89 c7                mov    %rax,%rdi
    115b:       e8 f0 fe ff ff          call   1050 <puts@plt>
        return 0;
    1160:       b8 00 00 00 00          mov    $0x0,%eax
}
    1165:       5d                      pop    %rbp
    1166:       c3                      ret
```

Thinking 1.2

解析ELF文件

```
git@23373175:~/23373175/tools/readelf (lab1)$ ./readelf ../../target/mos
0:0x0
1:0x80020000
2:0x80022070
3:0x80022088
4:0x800220a0
5:0x0
6:0x0
7:0x0
8:0x0
9:0x0
10:0x0
11:0x0
12:0x0
13:0x0
14:0x0
15:0x0
16:0x0
17:0x0
```

readelf的不同

使用readelf -h可以看出readelf文件类别是ELF64，而hello的类别是ELF32。

观察makefile也可以看出，编译出hello的命令选项多了一个-m32

```
git@23373175:~/23373175/tools/readelf (lab1)$ cat Makefile
%.o: %.c
	$(CC) -c $<

.PHONY: clean

readelf: main.o readelf.o
	$(CC) $^ -o $@

hello: hello.c
	$(CC) $^ -o $@ -m32 -static -g

clean:
	rm -f *.o readelf hello
```

Thinking 1.3

上电启动的地址是bootloader的地址，bootloader完成初始化功能之后，在硬盘中找到内核，将内核加载入内存，然后跳转到内核入口。所以内核入口没有放在上电启动地址。

难点分析

- 操作系统的启动过程

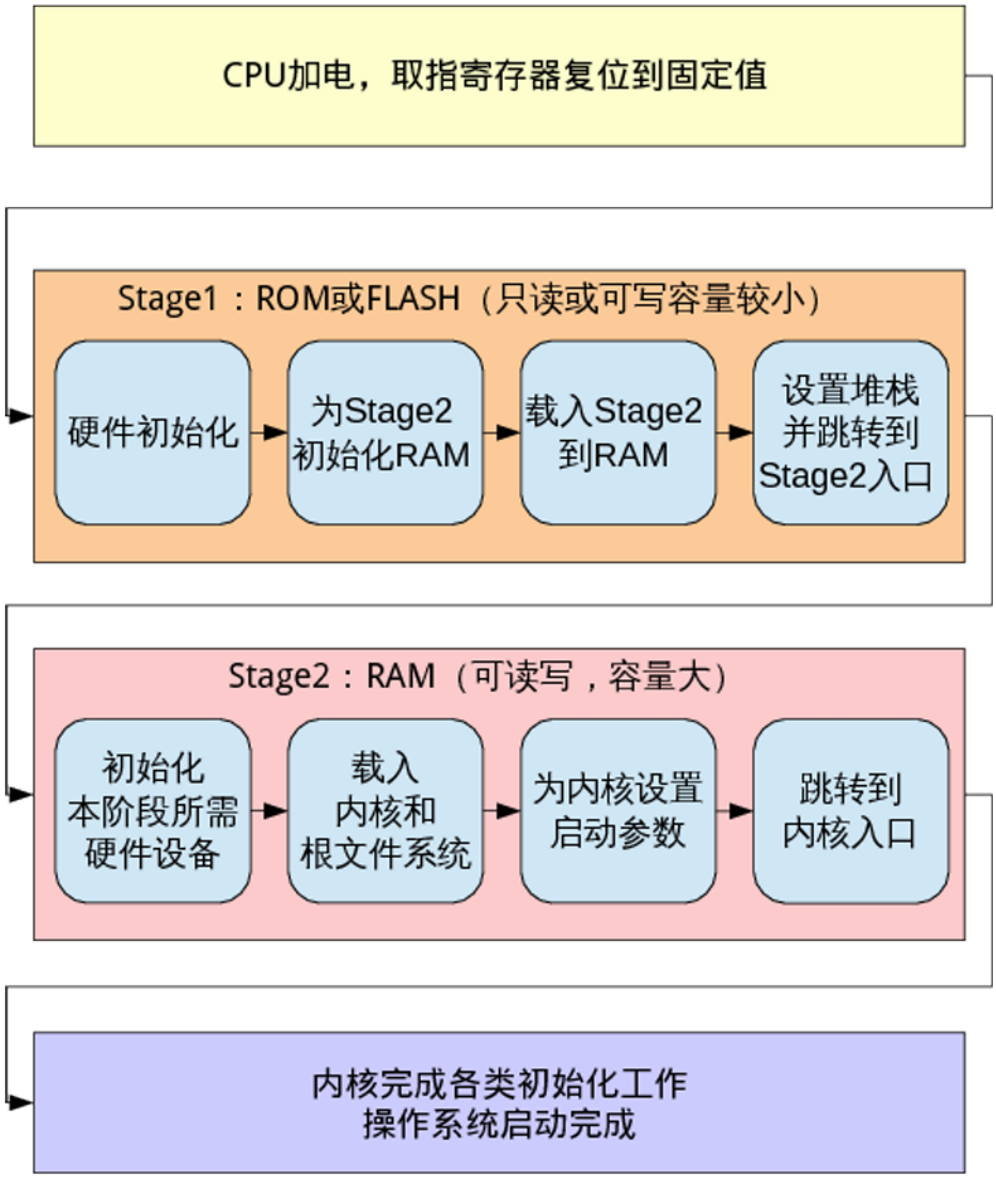

图 A.1: 启动的基本步骤

- 内核文件的本质（ELF文件格式）

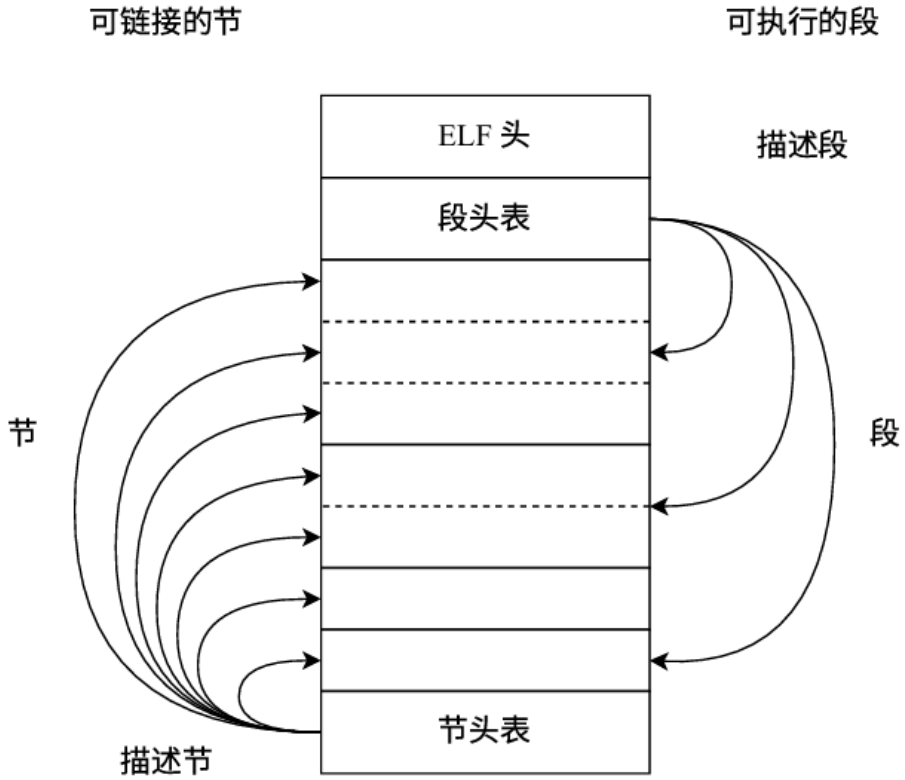

图 1.1: ELF 文件结构

可以用结构体描述，需要意识到C语言结构体变成二进制文件之后是什么样，需要意识到段头表和节头表的表项就是对应节的地址，在C语言中直接取指或者用箭头使用就可以了。

- MIPS的内存布局

  不要想太多为什么这么安排，这样布局自有道理，按照设计好的布局使用内存就好了

- 控制内核加载地址

  内核加载地址最终在内核ELF文件中，这里的控制其实就是编写链接器

- 理解printk函数
  - 理解往控制台输出字符就是读写某一个特殊的内存地址
  - 理解可变长参数
  - 根据函数的规格编写vprintfmt

实验体会

lab1实验介绍了操作系统的启动，内核的本质，ELF文件格式，MIPS内存布局，如何控制内核加载地址，知道了这些之后，我们可以比较清楚操作系统是如何开始工作的。在这个学习的过程中，我们需要清楚现在是在“模拟”制作一个设计成熟的操作系统，而不是从零开始做出人类历史上第一个操作系统。我们的实验环境是成熟的linux操作系统，有成熟的编译工具和调试工具来帮助我们编写，编译，调试，最终生成一份ELF文件。这个过程非常方便，因为我们借助了成熟的工具，这些工具都是建立在成熟的操作系统之上的。不应该去考虑”如果没有这些工具应该如何制作出这个内核“这样的问题。如果没有这些工具，制作的难度会大大提高，但是也不是不可能，只不过需要我们手动编写，编译，调试，”手撕“一份二进制文件出来。不用去考虑先有鸡还是先有蛋的问题，努力弄清楚现有的系统是如何工作的就好了。