Lab 1 Report

思考题

Thinking 1.1

为了观察.o和.out文件中的具体指令细节，我们使用objdump工具对其进行反汇编：`objdump -DS hello.o > instr.txt`（以.o为例），内容重定向至instr.txt。

- -D--disassemble-all
- -S--source:
  Display source code intermixed with disassembly, if possible.

Thinking 1.2

使用自创readelf解析target/mos:

```
git@21371295:~/21371295/tools/readelf (lab1)$ ./readelf
~/21371295/target/mos
0:0x0
1:0x80400000
2:0x80401a80
3:0x80401a98
4:0x80401ab0
5:0x0
6:0x0
7:0x0
8:0x0
9:0x0
10:0x0
11:0x0
12:0x0
13:0x0
14:0x0
15:0x0
16:0x0
```

使用自创readelf无法解析自身。

使用系统readelf解析target/mos:

```
git@21371295:~/21371295/tools/readelf (lab1)$ readelf -h
~/21371295/target/mos
ELF 头：
  Magic：   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  类别:                              ELF32
  数据:                              2 补码，小端序 (little endian)
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI 版本:                          0
  类型:                              EXEC (可执行文件)
  系统架构:                          MIPS R3000
  版本:                              0x1
  入口点地址：               0x80401650
  程序头起点：          52 (bytes into file)
  Start of section headers:          19328 (bytes into file)
  标志：             0x1001, noreorder, o32, mips1
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         4
  Size of section headers:           40 (bytes)
  Number of section headers:         17
  Section header string table index: 16
```

使用系统readelf解析自创readelf:

```
git@21371295:~/21371295/tools/readelf (lab1)$ readelf -h ./readelf
ELF 头：
  Magic：   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  类别:                              ELF64
  数据:                              2 补码，小端序 (little endian)
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI 版本:                          0
  类型:                              DYN (Position-Independent
Executable file)
  系统架构:                          Advanced Micro Devices X86-64
  版本:                              0x1
  入口点地址：               0x1180
  程序头起点：          64 (bytes into file)
  Start of section headers:          14488 (bytes into file)
  标志：             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         13
  Size of section headers:           64 (bytes)
  Number of section headers:         31
```

```
  Section header string table index: 30
```

Makefile中两个target的差别：

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

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

注意到readelf在编译时缺少参数`-m32`，它的类别是ELF64，hello和target/mos的类别均是ELF32。自创readelf只能解析32位的文件，因此无法解析自身。

Thinking 1.3

启动过程可以分为两个阶段：第一阶段是加载bootloader，第二阶段是由bootloader加载操作系统内核。

MIPS启动入口地址为0xBFC00000,位于kseg1,MMU将虚拟地址的高三位清零就得到物理地址用于访存。

在 GXemul 仿真器上运行的操作系统可以直接加载 ELF 格式的内核，省去了第一阶段，只需将内核加载到内存后通过Linker Script跳转到内核入口处。

难点分析

1.1 readelf()

使用c语言指针在遍历节头时的地址操作：

`(Elf32_Shdr *)p + 1`即为`(void *)p + sizeof(Elf32_Shdr)`

1.2 kernel.lds

include/mmu.h中包含内存布局图

1.3 start.S

由于mips_init函数没有返回值，使用`j`不使用`jal`

栈空间从高地址向低地址增长

1.4 vprintfmt()

对不同文件的结合阅读，需要找到函数在头文件中的原型，逐层寻找函数之间的调用，同时关注函数参数的来源与使用。特别注意**回调函数** `outputk` 的用法。

使用c语言指针对字符串进行解析，提取 `printk` 函数的各参数，形式化定义如下：

`%[flags][width][length]<specifier>`

实验体会

本次实验中接收并尝试理解了大量概念，涉及了地址、编译等知识，初步形成了操作系统启动的印象。以下是我尝试梳理的内容。

首先使用Makefile构建内核，其中包含编译和链接两过程。链接使用Linker Script调整各节的地址，其中包含代码与变量地址，程序入口地址，并由它链接出可执行文件。最终产生的内核（可执行文件）是ELF格式的，由ELF头，段头表，节头表等构成，其中节头表在编译及链接时需要使用，形成了闭环，对此我还需加深理解。

学习后感觉操作系统是一些互相联系的文件的集合，层次十分复杂，处理这样一个多文件综合系统是一个有趣的挑战，像在建房子。