OS-lab1实验报告

1.思考题

1.1也许你会发现我们的readelf程序是不能解析之前生成的内核文件(内核文件是可执行文件)的，而我们之后将要介绍的工具readelf则可以解析，这是为什么呢？(提示：尝试使用readelf -h，观察不同),

分别用file命令对testEFL和vmlinux检查，得到如下结果

```
file testEFL
testELF: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically
linked (uses shared libs), for GNU/Linux 2.6.24,
BuildID[sha1]=0xf9101868d73bbb598aa9153dba68e9c547464f47, not stripped

file vmlinux
vmlinux: ELF 32-bit MSB executable, MIPS, MIPS-I version 1 (SYSV), statically
linked, not stripped
```

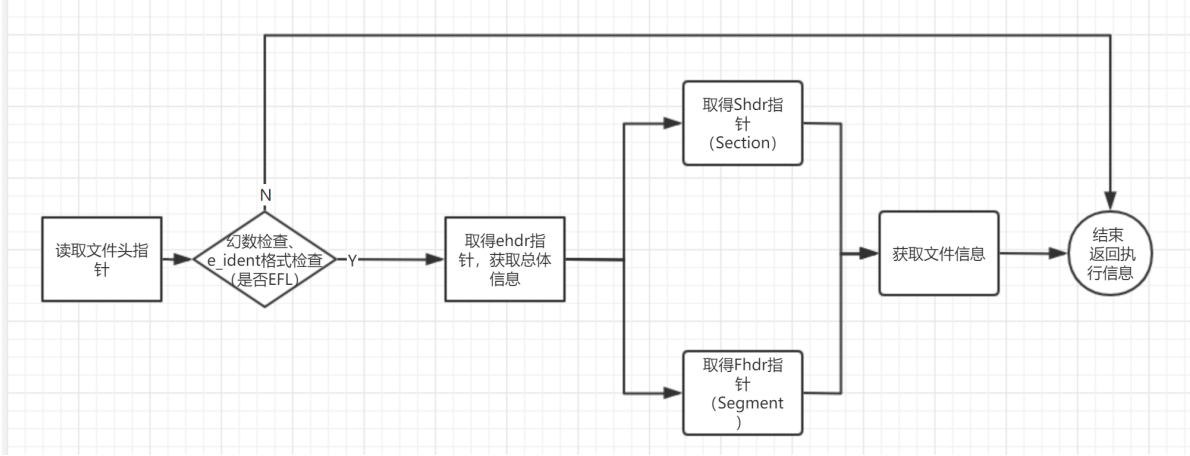
可以发现，vmlinux是MSB(Most significant bit)的，而testEFL是LSB(Least significant bit)的，而我们构造的readelf程序只能检查LSB的elf ，对于MSB的elf需要进行大小端转化后才可以进行解析。

1.2 内核入口在什么地方？main 函数在什么地方？我们是怎么让内核进 入到想要的 main 函数的呢？又是怎么进行跨文件调用函数的呢?

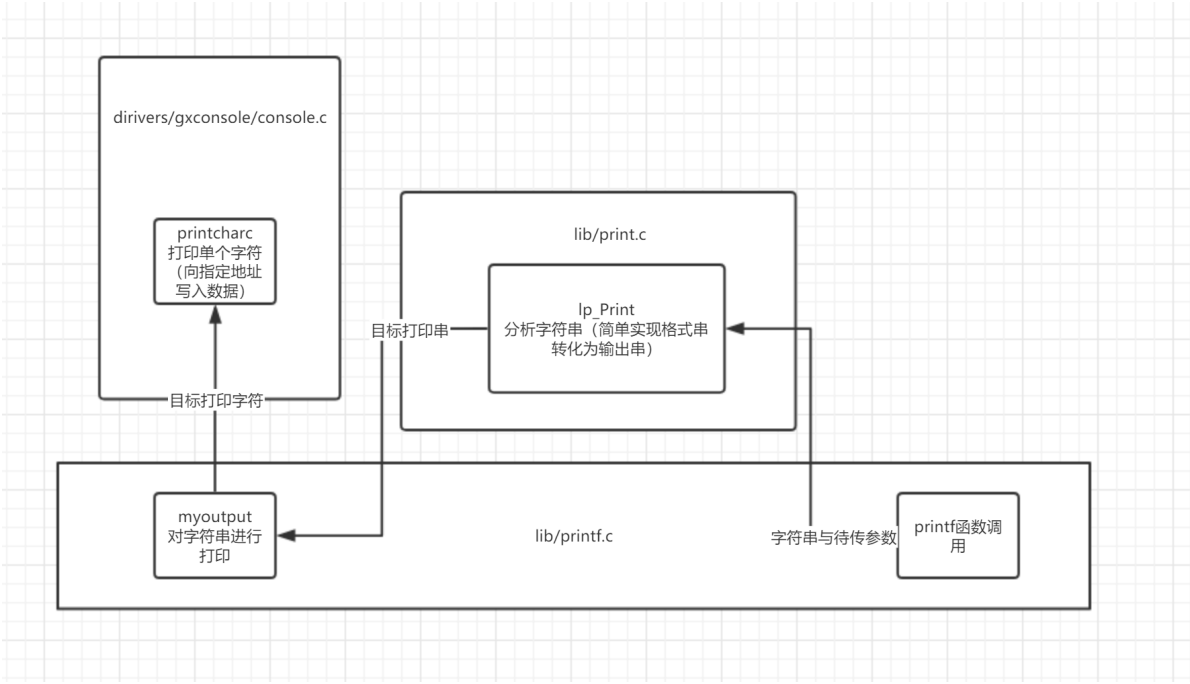
内核入口在start.S；main函数在文件中位于`init/main.c`中；（在装载运行时导入内存分配的位置）通过jal跳转指令进入到main函数；跨文件调用通过交叉编译时通过ELFrelocate文件格式补充相关函数入口地址（c语言程序表现为.o文件的链接）通过跳转至指定地址调用函数。

2.实验难点图示

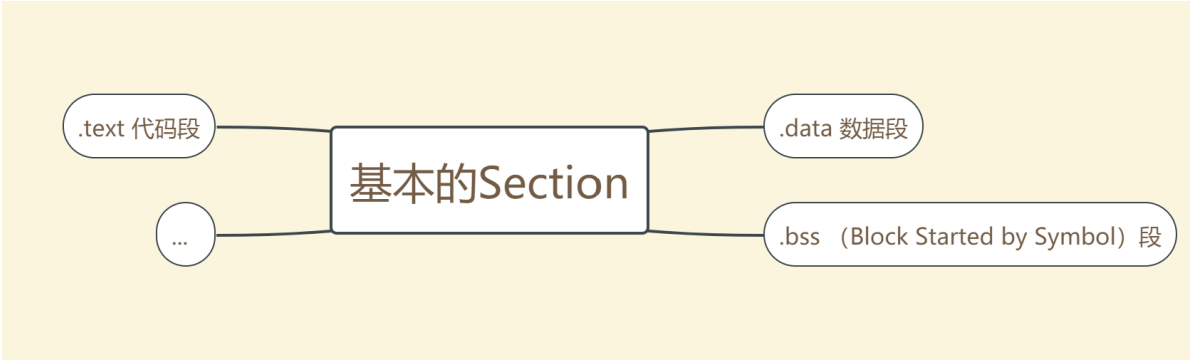
1.readelf程序的基本框架



2.简易系统调用printf的流程



3.基本的Section

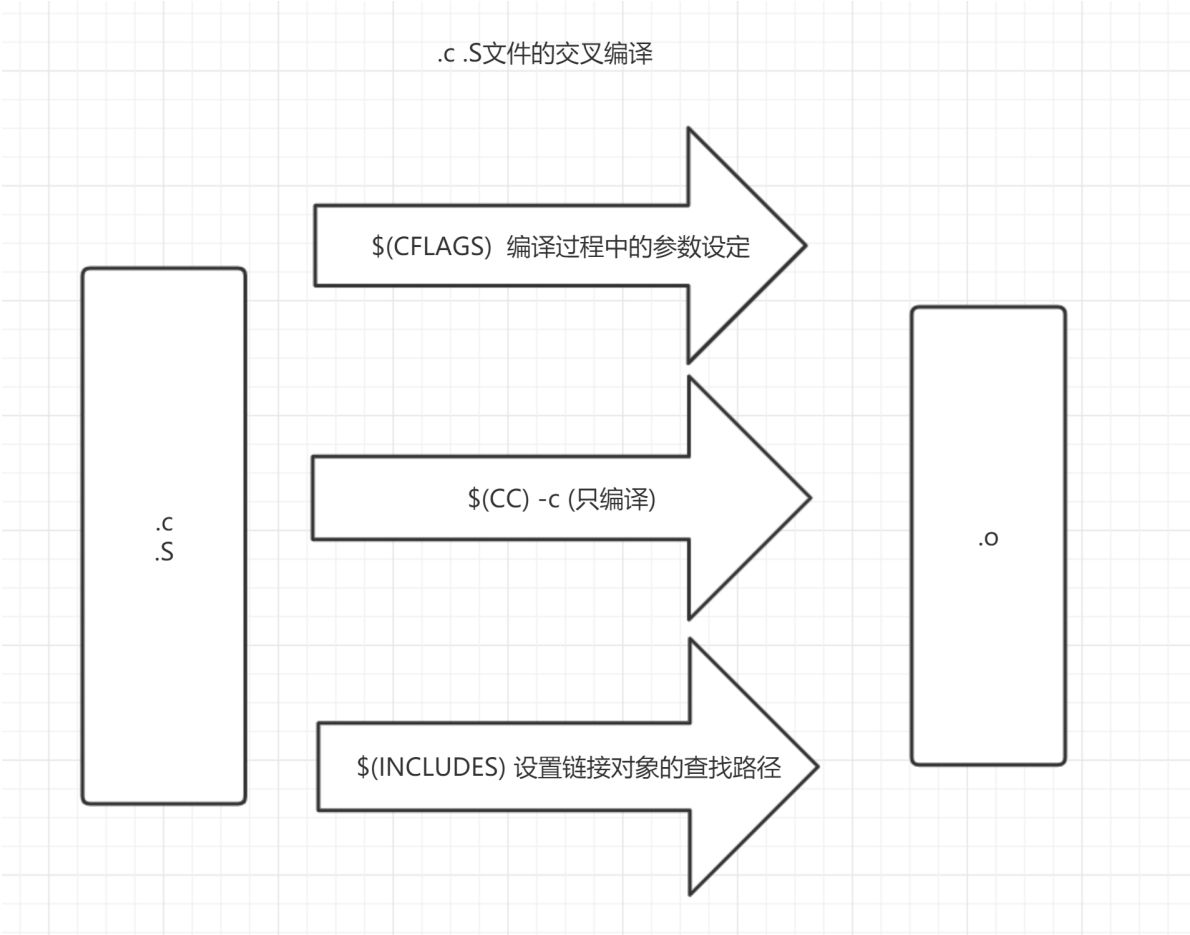


4.Makefile的结构（以编译.S .c文件为主）

首先给出这些Makefile的一些指定的宏

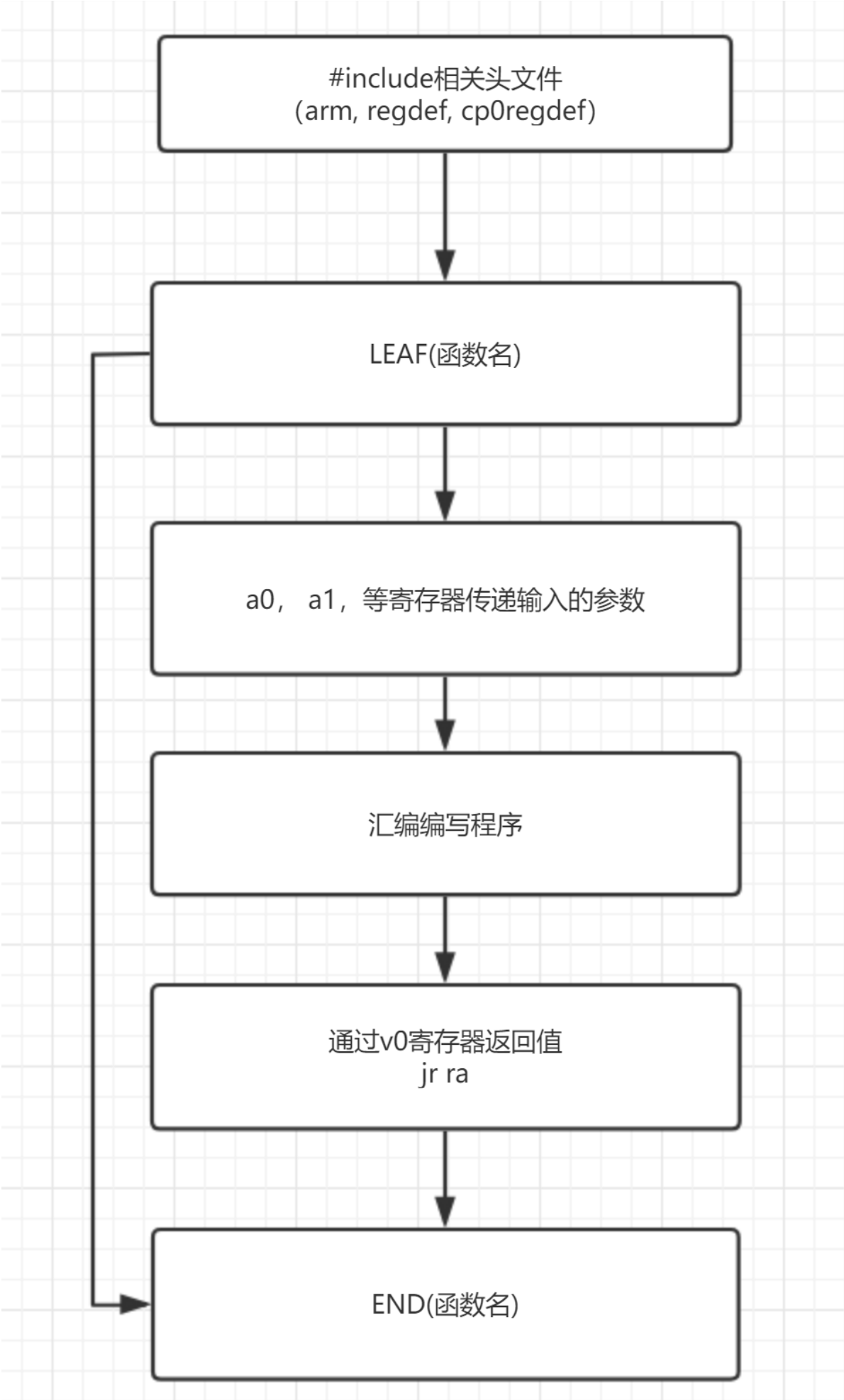
```
CROSS_COMPILE := bin/mips_4KC-
CC  := $(CROSS_COMPILE)gcc
CFLAGS  := -O -G 0 -mno-abicalls -fno-builtin -Wa,-xgot -Wall -fPIC -Werror
LD  := $(CROSS_COMPILE)ld
INCLUDES := -I./ -I../ -I../include/
```

对于.c .S（.c 可以通过 gcc -S 转化为.S 文件）而言：



5.S文件的编写

lab1-2的上机Extra考察了.S文件的编写，下面对此进行一个总结，编写流程如下


#include相关头文件
(arm, regdef, cp0regdef)
LEAF(函数名)
a0， a1，等寄存器传递输入的参数
汇编编写程序
通过v0寄存器返回值
jr ra
END(函数名)

注意的问题

1.这里使用的地址均为逻辑地址

2.如果需要读取从console进行的输入，需要写一个循环直到指定的地址处被输入了数据才跳出循环执行之后的程序

```
loop:
 lb t0, add(off)
 beq t0, 0, loop
 nop
sb t0, add(off)
...
```

3.体会与感想

lab1实验难度总体不是很大，在本次实验中其他方面花费的时间比较少，但在ELF解析的练习中，由于一开始对ELF中内容了解不够透彻，花了很长时间才搞懂应该怎么样读取每个Section的地址。在本次课下实验中，我们实现了简易的printf的系统调用，这使得我们对C语言系统调用的过程有了更为深刻的理解。而读取ELF格式文件的相关信息的训练，是我们简略地了解到ELF文件是如何被读取到详细的信息从而(在未来)被链接和执行的。而课上第一次测试让我们更一步了解了readelf解析的过程，以及Bigendian和Littleendian格式的文件的不同的读取方法；第二次测试考察我们对printf的实现函数的理解情况和灵活运用能力，以及MIPS汇编简单的输入输出函数（构建自己的简单的读写IO的driver）。

4.指导书反馈

1.如讨论区所述，感觉对Section，Segment的翻译不太统一，对entry的翻译有点问题（是翻译成"入口"还是"项目" "元素"，应该根据实际情况进行斟酌）。

2.感觉实验第一题（readelf.c的填补）在之前的指导中还是有点不够明确，要是能够多解释一点（关于ph_size和sh_size的详细情况）可能会更好。

5.残留难点

其实不太清楚我们什么时候是在linux实际的环境下工作的，什么时候实在我们搭建的操作系统的环境下工作的。比如关于段空间分配：我们在练习中修改了Linker_Script，但是仅有基本部分（.bss, .data, .text）被设置了，那么我们调用自己的printf和readelf时，是只需要这些段就够了？还是实际调用了linux系统本身的Linker_Script？其余练习题同理。