RISC-V

指令级模拟器设计与实现

汇报人：刘文利

汇报内容

- 项目描述
- ELF格式解析
- 指令模拟器结构及工作流程
- 基本功能测试
- 系统调用实现
- 系统调用的执行结果与优化方案

1.项目描述

- 项目内容：实现RISC-V指令级模拟器设计，同时该模拟器可支持系统调用。输入是RISC-V工具链生成的可执行文件，输出是该可执行文件的执行结果，并统计输出执行程序的指令数。
- 项目思路：
  - 首先对使用RISC-V工具链生成的ELF文件进行解析，ELF文件的最初52个字节记录了所有的Header Table在文件中的位置，在ELF文件头中放有程序的虚拟地址入口点。
  - 然后进行模拟器初始化即根据读取的ELF文件信息将数据段和代码段载入到虚拟内存中，并初始化SP、GP、PC。
  - 最后对指令进行取址、译码、执行操作，同时通过添加scall指令实现系统调用。

2.ELF格式解析

- ELF文件中的信息可以划分为两种：Header和Section。Header描述Section的信息，如Section的用途，长度，以及如何在整个文件中找到。读取ELF文件的过程，就是根据Header找到Section的过程。ELF文件的最初52个字节记录了所有的Header Table在文件中的位置。

2.ELF格式解析

Elf Header的定义如下：

```
#define EI_NIDENT 16
typedef struct {
    unsigned char  e_ident[EI_NIDENT];
    Elf32_Half     e_type;
    Elf32_Half     e_machine;
    Elf32_Word     e_version;
    Elf32_Addr     e_entry;
    Elf32_Off      e_phoff;
    Elf32_Off      e_shoff;
    Elf32_Word     e_flags;
    Elf32_Half     e_ehsize;
    Elf32_Half     e_phentsize;
    Elf32_Half     e_phnum;
    Elf32_Half     e_shentsize;
    Elf32_Half     e_shnum;
    Elf32_Half     e_shstrndx;
} Elf32_Ehdr;
```

e_entry定义了程序入口地址，根据此信息来设置pc寄存器

e_phoff定义了program header table在整个文件中的位置

e_phentsize定义了program header table中每一项的大小

e_phnum定义了program header table 中有多少项

2.ELF格式解析

Elf Header的定义如下：

```
typedef struct {
    Elf32_Word    p_type;
    Elf32_Off     p_offset;
    Elf32_Addr    p_vaddr;
    Elf32_Addr    p_paddr;
    Elf32_Word    p_filesz;
    Elf32_Word    p_memsz;
    Elf32_Word    p_flags;
    Elf32_Word    p_align;
} Elf32_Phdr;
```

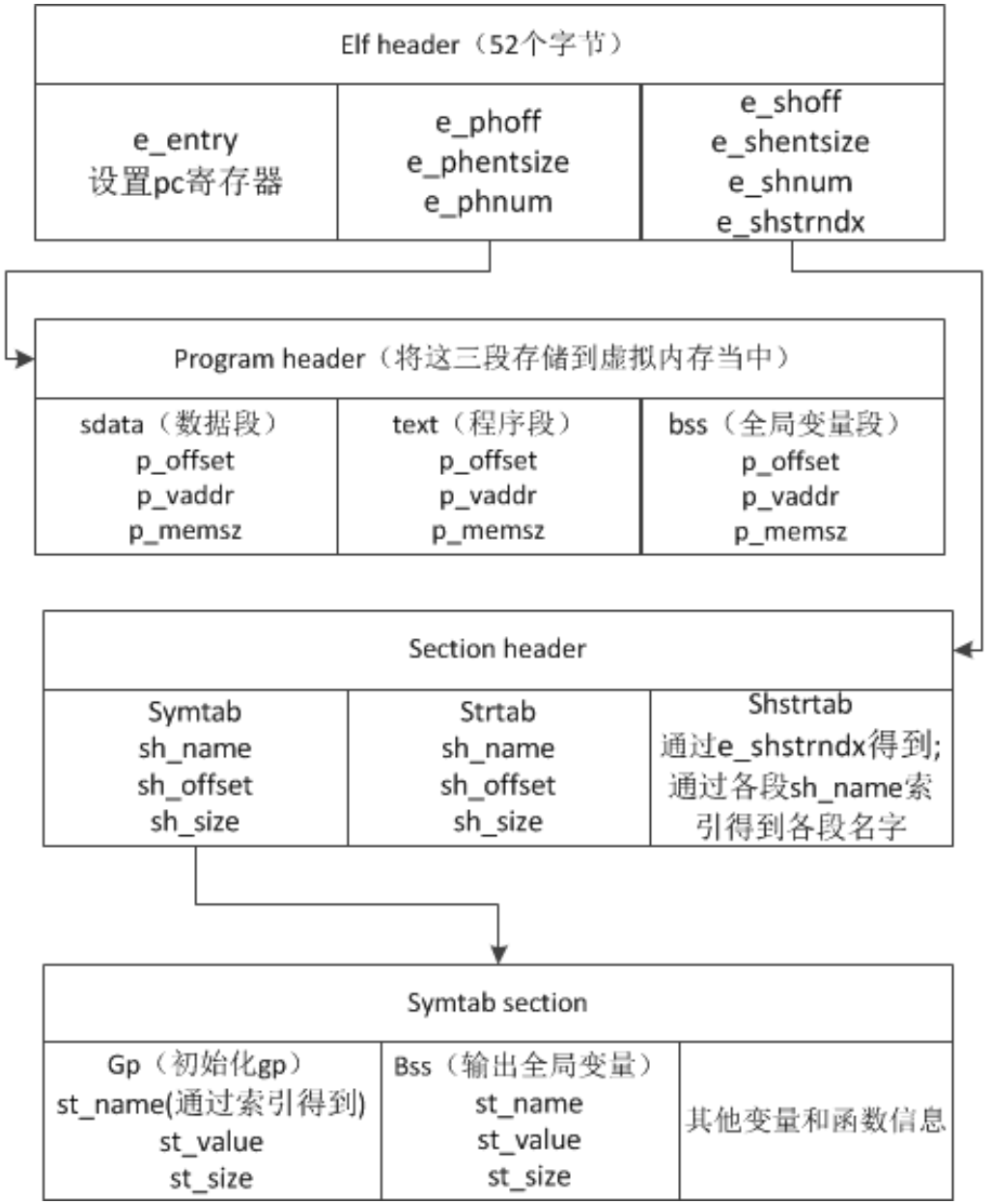
p_offset定义了程序段在整个文件中的位置

p_vaddr定义了程序段放在内存中时，首个字节的虚拟地址

p_memsz定义了程序段在内存中的大小

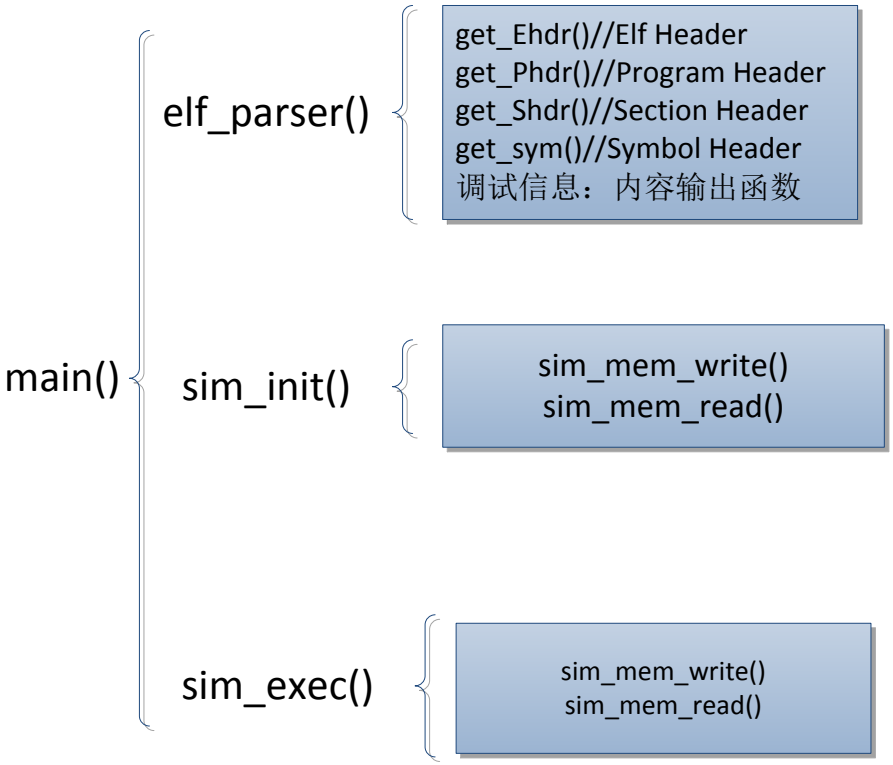
可以根据这三个内容将文件中起始位置为p_offset的程序段载入到起始地址为P_vaddr，大小为p_memsz的虚拟内存块中

2.ELF格式解析


Elf header（52个字节）
e_entry
设置pc寄存器
e_phoff
e_phentsize
e_phnum
e_shoff
e_shentsize
e_shnum
e_shstrndx
Program header（将这三段存储到虚拟内存当中）
sdata（数据段）
p_offset
p_vaddr
p_memsz
text（程序段）
p_offset
p_vaddr
p_memsz
bss（全局变量段）
p_offset
p_vaddr
p_memsz
Section header
Symtab
sh_name
sh_offset
sh_size
Strtab
sh_name
sh_offset
sh_size
Shstrtab
通过e_shstrndx得到;
通过各段sh_name索
引得到各段名字
Symtab section
Gp（初始化gp）
st_name(通过索引得到)
st_value
st_size
Bss（输出全局变量）
st_name
st_value
st_size
其他变量和函数信息

3.指令模拟器结构及工作流程

- 指令模拟器的组织结构用函数结构图来表示：



调用header、section各部分解析函数，将得到的这些信息存储到相应的结构中，然后将这些信息输出，以便调试。

主要实现的功能
①将数据段和代码段存入内存。
②初始化GP、PC、SP等
③插入开始和结束指令。
其中会调用内存读写函数
sim_mem_read()和sim_mem_write()

实现指令的取址、译码、执行操作，并通过实现scall指令实现系统调用。

图3-1 指令模拟器函数结构图

3.指令模拟器结构及工作流程

- 指令模拟器的工作流程如图：

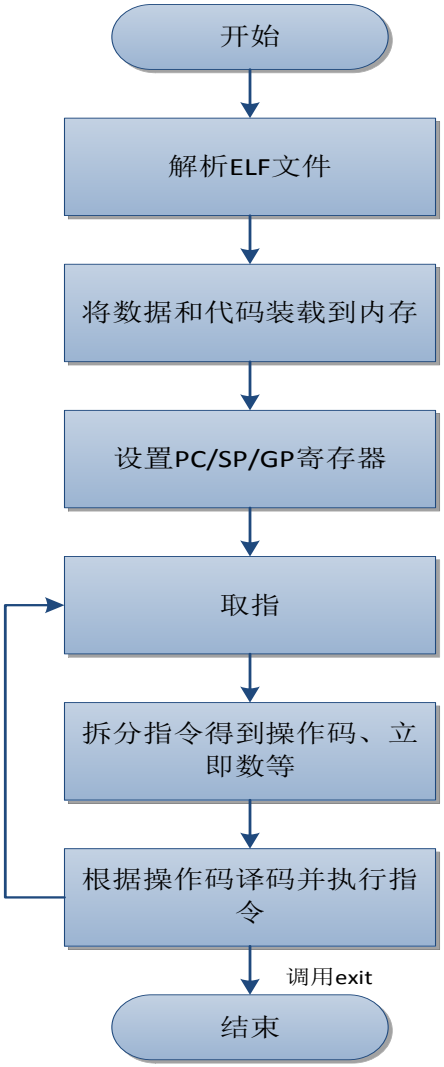


图3-2 指令模拟器工作流程

4.基本功能测试

- 测试程序:
  - (1)HanNo.c
  - (2)quicksort.c
  - (3)mat.c

4.基本功能测试

HanNo.c测试结果

- 编译选项：

  riscv64-unknown-linux-gnu-gcc -o HanNo.out -m32 -static -e main HanNo.c

- 指令数为:1264

```
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final$ ./sim.out test/HanNo/HanNo.out
result:1 2 1 3 2 3 1 2 3 1 3 2 1 2 1 3 2 3 2 1 3 1 2 3 1 2 1 3 2 3 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0
total inst is:1264
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final$
```

4.基本功能测试

quickSort.c测试结果

- 编译选项：

  riscv64-unknown-linux-gnu-gcc -o quickSort.out -m32 -static -e main quickSort.c

- 指令数为:1603

```
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final/test/mat$ ris
cv64-unknown-linux-gnu-gcc -o mat.out -m32 -static -e main mat.c
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final/test/mat$
riscv64-unknown-linux-gnu-gcc -o mat.out -m32 -static -e main mat.c
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final/test/mat$
cd ../..
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final$ ./sim.out
 test/mat/mat.out
result:1 3 5 2 4 6 3 5 7
total inst is:606
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final$ cd test/q
uickSort/
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final/test/quickSort$
v64-unknown-linux-gnu-gcc -o quicksort.out -m32 -static -e main quicksort.
riscv64-unknown-linux-gnu-gcc: error: quicksort.c: No such file or directo
riscv64-unknown-linux-gnu-gcc: fatal error: no input files
compilation terminated.
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final/test/quickSort$
quickSort.c  result.log  Zhang_zhao.log
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final/test/quickSort$
esult.log
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final/test/quickSort$
hang_zhao.log
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final/test/quickSort$
v64-unknown-linux-gnu-gcc -o quickSort.out -m32 -static -e main quickSort.
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final/test/quickSort$
riscv64-unknown-linux-gnu-gcc -o quickSort.out -m32 -static -e main quickS
ort.c
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final/test/quickSort$
cd ../..
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final$ ./sim.out test/
quickSort/quickSort.out

total inst is:1603
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final$
```

4.基本功能测试

mat.c测试结果

- 编译选项：

  riscv64-unknown-linux-gnu-gcc -o mat.out -m32 -static -e main mat.c

- 指令数为:606

```
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final$ ./sim.out
result:1 2 1 3 2 3 1 2 3 1 3 2 1 2 1 3 2 3 2 1 3 1 2 3 1 2 1 3 2 3 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0
total inst is:1264
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final$ ls
dhrystone            dhrystone-scanf-time  riscv-simulator  test
dhrystone_put_clock  dhstone_put_time      sim.out          tool.sh
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final$ cd test
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final/test$ ls
HanNo  mat  quickSort
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final/test$ cd m
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final/test/mat$
mat.c  result.log
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final/test/mat$
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final/test/mat$
cv64-unknown-linux-gnu-gcc -o mat.out -m32 -static -e main mat.c
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final/test/mat$
riscv64-unknown-linux-gnu-gcc -o mat.out -m32 -static -e main mat.c
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final/test/mat$
cd ../..
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final$ ./sim.out
 test/mat/mat.out
result:1 3 5 2 4 6 3 5 7
total inst is:606
liuwenli@liuwenli-Lenovo-G460:~/riscv-elf/simulator-final$
```

5.1系统调用基本原理说明

- 系统调用是用户程序与内核交互的一个接口。
- RISC-V是将系统调用号存储在寄存器a[7]即r[17]中，通过a[7]的值就可判断调用的系统调用。
- 对于参数传递，通过寄存器完成的。最多允许向系统调用传递7个参数，分别依次由a[0]、a[1]、a[2]、a[3]、a[4]、a[5]、a[6] 这个7个寄存器完成。
- 一个系统调用函数可能被多个函数进行调用。

5.2系统调用实现流程

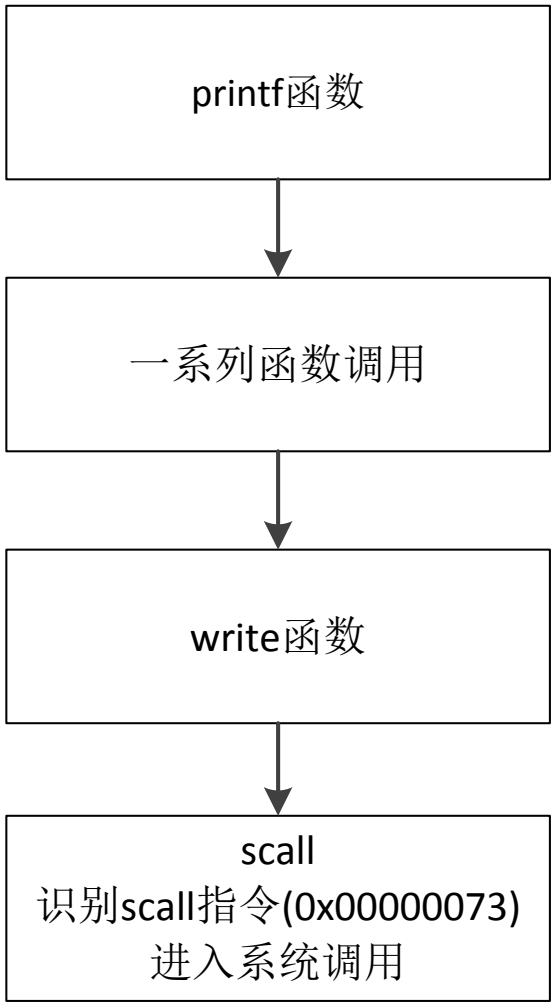


图5.2.1 printf为例系统调用过程

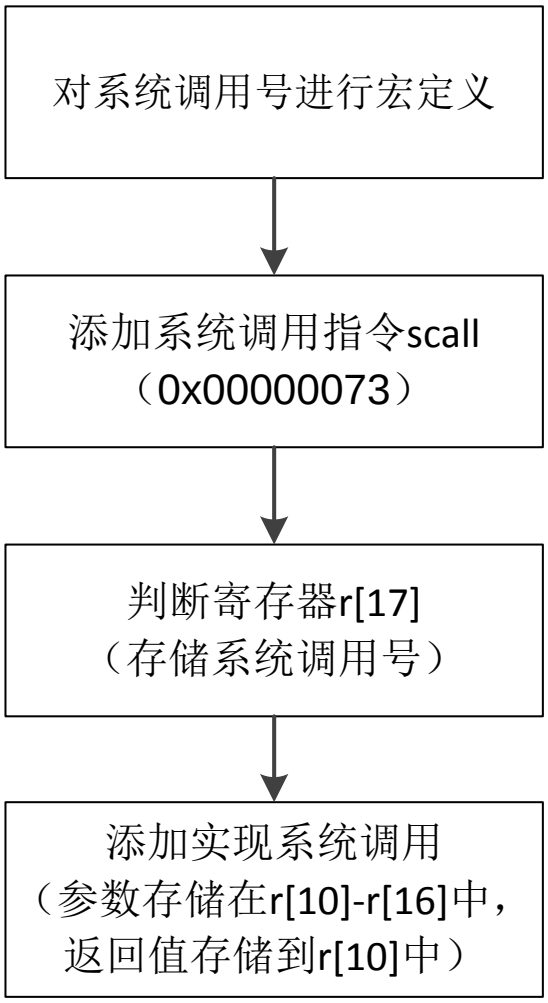


图5.2.2 系统调用实现

6.1系统调用的执行结果

dhrystone执行结果

Dhrystone编译：用Makefile进行编译

编译工具链为：riscv64-unknown-elf-gcc

编译选项为：-m32 -march=RV32I

- **动态执行指令数：**统计的是大循环中的指令数

总指令数为71663702，循环执行次数为100000次，

所以动态执行指令数为：716条

执行时间为2521668μm，程序运行速度为28419166条/秒。

```
User Time: 2521668, Begin Time: 4894, End Time: 2526562
Number of runs: 100000
Microseconds for one run through Dhrystone:    0.0
Dhrystones per Second:                         0.0

total inst is:71663702
```

6.2优化方案

- 指令计数从大循环处开始。
- 尝试不同的编译选项，对结果进行比较。
- 实现64位指令模拟器。

谢谢