Lab 2.1 实验报告

薛犇 1500012752

1. 实验环境

ubuntu 16.04LTS

riscv-toolchain

g++ version:5.5.6

gcc version:5.5.6

2. 实验过程

1. 对riscv官方网站提供的toolchain进行交叉编译。编译后生成riscv-unknow-elf-gcc, riscv-unknow-elf-objdump, riscv-unknow-elf-readelf等后续实验会用到的工具，利用这些工具可以对文件进行基于riscv rv64指令集的编译，并进行相关的分析。

2. 编写自己的readelf工具。读取步骤为：

   （1）读取elf header 。获得一些文件的基本信息，例如大小端法、指令集等等；更重要的是获得后续信息在文件中的定位，例如section header的偏移量sh_offset和programme header的偏移量ph_offset。

   （2）读取elf section。获得每个section的信息，以.data段为例，可以获得本section在文件中的位置，在内存中的虚拟地址、大小等等。

   （3）读取elf symbol。首先根据上个步骤中已经读取出来的.symtab和.symstr中的信息，读取每个符号的地址和名称。

   （4）读取elf programme。根据第（1）步中读出来的programme header偏移量，定位到programme的位置，读取本segment的偏移量、大小、虚拟地址等等。

3. 根据读出来的关键信息，将代码段和数据段拷贝到模拟器构建的模拟内存的memory数组中，拷贝时需要注意将原来的char型转化成memory数组的unsigned int类型。

4. 译码，根据提供的greencard和补充的手册，解析instruction之后判断该指令的功能，并进入相应的代码块实现，需要注明的是，本次实验只进行了最简单的实现，并没有利用信号的传递，尚未实现流水线化，这是后续实验可以改进的地方。

5. 利用riscv-unknow-elf-gcc编译需要执行的代码，然后利用编写的模拟器执行之前生成的目标文件，根据需要打印出debug信息，例如寄存器内的值、内存中的值等等。

3. 实验结果

选择了Lab 1.1中的矩阵相乘程序作为待模拟的程序。做了一些修改，把矩阵全部作为全局变量，并不从文件中读取矩阵数据，而是在main函数中初始化一些特定的值。所有的运算集中在main函数中。

考虑到模拟器的模拟规模有限，实验选择了小规模的矩阵，设置了两个3x3的矩阵a和b作为乘数矩阵，相乘的结果会放到c矩阵中。 a的初始值为：

```
a[3][3] = {{0, 1, 2},
    {1, 2, 3},
    {2, 3, 4};
```

b的初始值为：

```
b[3][3] = {{0, -1, -2},
    {1, 0, -1},
    {2, 1, 0}};
```

利用模拟器执行程序之后，分别打印a、b、c地址之后的9个数，也即a、b、c矩阵内的数值，结果如下：

```
final information:
0. 0  1. 0  2. 2faf080  3. 11f58  4. 0  5. 0  6. 0  7. 0  8. 0  9. 0  10. 0  11. 2  12. 4  13. fffffffffffffff9  14. 3  15. 0  16. 0  17. 0  18
. 0  19. 0  20. 0  21. 0  22. 0  23. 0  24. 0  25. 0  26. 0  27. 0  28. 0  29. 0  30. 0  31. 0
a:
Mem11810 0   Mem11814 1   Mem11818 2   Mem1181c 1   Mem11820 2   Mem11824 3   Mem11828 2   Mem1182c 3   Mem11830 4
b:
Mem117c0 0   Mem117c4 -1   Mem117c8 -2   Mem117cc 1   Mem117d0 0   Mem117d4 -1   Mem117d8 2   Mem117dc 1   Mem117e0 0
c:
Mem117e8 5   Mem117ec 2   Mem117f0 -1   Mem117f4 8   Mem117f8 2   Mem117fc -4   Mem11800 11   Mem11804 2   Mem11808 -7
simulate over!
```

可以读出c矩阵中的值为

```
c[3][3] = {{5, 2, -1},
    {8, 2, -4},
    {11, 2, -7}};
```

符合运算结果，说明程序执行正确。

附加

本模拟器支持单步debug模式，可以单步调试并且打印寄存器和内存信息。 使用方式见README.md文件。