计算机系统第二次作业报告

实验内容

在gem5语言下的x86 ISA中补入名为FSUBR的指令，并在程序中调用此指令测试效果。

实验结果

读入为10.0 1.0，在TimingSimpleCPU和DeriveO3CPU上跑程序，输出均为-0.900000

其中前者模拟时间为0.000390秒，后者为0.000161秒（具体实验数据详见两个stats.txt文件）

实验心得

关于指令FSUBR

FSUBR与FSUB的区别就是操作数的顺序相反，例如取堆栈寄存器中的两个数做减法，FSUB计算的是$st(0) - st(i)$，而FSUBR计算的是$st(i) - st(0)$。

关于在程序中添加参数

第一次实验时实际上已经通过在脚本程序文件中添加命令行参数来运行不同设置下的模拟。

```
parser = OptionParser()
parser.add_option("--clk",help="CPU clk. Default: 1GHz")
parser.add_option("--cpu",help="CPU Model. Default:TimingSimpleCPU")
parser.add_option("--memory_tech",help="Memory technology. Default:DDR3_1600_x64")
parser.add_option("--cache_block",help="CPU cache block. Default: 64")
parser.add_option("--l1i_size",help="L1 instruction cache size. Default: 16kB.")
parser.add_option("--l1d_size",help="L1 data cache size. Default: Default: 64kB.")
parser.add_option("--l2_size",help="L2 cache size. Default: 256kB.")
```

所以这次添加参数时的操作是一样的，还是使用add_option工具。

关于采用不同的ISA的感受

这次的实验所采用的ISA较上一次区别在于添加了一条指令FSUBR，在写了一个C++程序之后确实体验到减法操作数交换的效果。但是新的ISA仅仅是添加了新的指令，原来的指令（例如FSUB）还在，这体现了一种改进ISA的策略——在保留旧指令的功能的基础上添加新的指令，让ISA日趋完善。这种策略避免了因为一条旧指令的变动而"牵一发动全身"对整个ISA造成巨大影响的问题。这也体现了ISA架构的灵活性——可以在ISA上"添砖加瓦"，更新ISA的同时却仍能兼容旧ISA的功能。