SimpleScalar Simple Analysis

1. SimpleScalar introduction

Simplescalar工具集起源于 80 年代中后期由Manoj Franklin开发的模拟工具。1995 年夏，Steve Bennett开发了SimpleScalar x86 工具。SimpleScalar模拟器由Intel公司微型计算机研究室的Todd M. Austin在 1996 年最终编写而成，美国麦迪逊威斯康星大学计算机学院的Doug Burger参与开发和文档整理工作，二人共同拥有该模拟器的版权。Austin与Bennett两人都是威斯康星大学计算机学院的助手，由Guri Sohi教授指导研究工作。

SimpleScalar模拟器模拟的是一个超标量，5 级流水的RISC体系结构的CPU模型，提供了从最简单到超标量乱序发射的不同的模拟程序。sim-outorder 是一个具有完整功能的模拟程序。在sim-outorder中使用了几乎所有的模拟资源，在阅读代码之前对模拟的体系结构和模拟资源充分的了解，能够大大提高下一步工作的效率。

Simplescalar 模拟器在功能级上实现了执行驱动、解释执行，在行为级上实现了流水线模拟。该工具集提供了一个以 GCC 为主的编译器以及相关组件，能够产生基于 Simplescalar 体系结构的目标代码，然后在 Simplescalar 模拟器上运行。

运行模拟器时，主程序 main( )做所有的初始化工作，并将二进制目标代码载入内存，然后调用 sim_main()，sim_main()在每个模拟器中单独说明，预先译码整个正文段，加快模拟，然后开始目标程序的模拟。

2. SimpleScalar中的虚拟的资源部件

SimpleScalar 模拟器采用的是执行驱动，所以在模拟器中要对大部分传统的RISC-CPU部件进行说明，诸如寄存器文件，存储系统（cache+mem），TLB，功能单元，保留站，再定序缓冲，分支预测部件，和 5 级流水线等。同时针对模拟的需要，在几乎所有硬件资源的软描述中都添加或简化了许多功能。

SimpleScalar是使用C语言编写的。下面以sim-outorder为例说明SimpleScalar的执行流程。主流程从main.c文件中的主函数main()开始。阅读main()可以看到它主要在作模拟前的准备工作，而真正的模拟执行跳转到 sim-outorder.c文件中模拟主函数sim_main()。函数sim_main()主干是一个死循环，它的流程完全模拟流水线的处理过程，不同的是由于模拟的需要将流水的顺序逆转，以使逻辑上的流水站不要提前处理要下一拍才能处理的指令。模拟结束通过信号量返回主函数main()，退出。

3. sim_main()流程

首先是一些准备工作，然后开始进入 5 级流水：
ruu_commit()→ruu_writeback()→ruu_issue()→ruu_dispatch()→ruu_fetch()，然后在每个流水周期结束后，cycle需要加"1"。

SimpleScalar是采用的执行方式驱动，所以在模拟器中要对目标处理器＋存储器的几乎全部单元进行说明，而被模拟程序就在这些虚拟的部件上"运行"，从中得到统计分析所需的数据。要理解SimpleScalar的代码，首先要熟悉它所定义的虚拟部件。模拟器中的部分资源是处理器的映像，部分资源是模拟所需引入的特有资源，不好直接映射到硬件资源。下面分别说明这些资源在模拟器中的描述方式：

- ✧ 模拟器运行的可执行文件描述：在模拟器中运行的是由SimpleScalar附带的编译器sslittle-na-sstrix-gcc或者ssbig-na-sstrix-gcc编译生成的。这种可执行文件

并不是通常意义下的可执行文件，在现有的操作系统平台下这种文件是无法用shell直接执行的。它的可执行性是相对模拟器程序而言的，除了附带在./tests/目录下预先生成的文件外，其他的模拟程序都需要将c源代码使用上述的编译器得到可以在模拟器中运行的"可执行"文件。

- ✧ 主存mem描述：主要的文件是memory.c和memory.h文件。SimpleScalar模拟的主存结构实际上是一个段页式结构。但是这种段页式结构与现代操作系统中的段页式结构有明显的区别。首先它仅有一个虚拟进程（被模拟执行的可执行文件），其次每个段的基地址都是固定的，在mem.h中使用了几个宏定义来规定所有段的起始地址。
- ✧ cache描述：在SimpleScalar中cache是可选的。一级指令cache、二级指令cache、一级数据cache、二级数据cache和指令TLB与数据TLB，都是使用相同的cache结构来描述。与实际中的cache结构不同的是，模拟器中的cache与TLB在大多数情况下都不包含实际的指令或数据，它们仅仅记录当前结构中对应cache/TLB记录的标记，用于模拟cache算法。
- ✧ 分支预测部件bpred：描述在现代处理器中分支预测是必不可少的，也是设计中比较复杂的部分。在SimpleScalar的乱序执行模拟sim-outorder中，控制指令的Next_PC在流水线的ruu_dispatch阶段计算出来，分支预测则是在取指阶段完成，预测的下一指令PC值信息跟随当前指令流入流水线。分支预测的模式有四种：
  - ➢ BPred2Level：两级分支预测模式
  - ➢ BPred2bit:简单的两位直接映射预测模式，（分支目标缓存BTB）
  - ➢ BPredTaken:分支发生静态预测模拟
  - ➢ BPredNotTaken：分支不发生静态预测模拟

  另外可以使用两级和位预测联合的预测模式(BPredComb)
- ✧ 寄存器reg描述：SimpleScalar模拟器映射的是MIPS芯片的寄存器文件结构，包括的寄存器资源有：
  - ➢ 整数寄存器文件：regs_R[0:31];
  - ➢ 浮点寄存器文件：regs_F[0:31];
  - ➢ 记录乘/除法结果的寄存器：高位regs_HI,低位regs_LO;
  - ➢ 浮点条件码寄存器：regs_FCC;
  - ➢ 程序计数器：regs_PC;
- ✧ 功能单元池res_pool描述：对于处理器中的算术逻辑部件ALU、乘除法部件和浮点部件SimpleScalar使用资源池来描述。
- ✧ 保留站RUU_station描述：保留站是乱序执行的核心，在SimpleScalar中保留站叫做寄存器更新单元RUU(Register Update Unit)，它的作用比一般意义上的保留站要强，它和保留站链接机构RS_link(Reserve station link)一起可以形成指令提交队列，完成重定序缓冲的功能。