体系结构大作业报告

2016ACM 班　范裕达

2018 年 1 月 12 日

1　摘要

本次体系结构的大作业是写一个 RISC-V 框架下的 CPU，我选择了使用上课所学的 Tomasulo 算法，并实现了基本的指令集。我的代码都是在 gnu-tool-chain 和 Icarus Verilog version 10.1 的框架下进行编译运行，输出的波形可以通过生成的 vcd 用 gtkwave 查看。

2　框架简述

我的代码使用的经典的 issue-execution-commit 结构，采用的是乱序执行和顺序写回的策略，对于分支跳转语句采用的是 stall 的方法。

2.1　issue

issue 阶段包括了从内存 fetch 指令, 并且进行解码，从而发射到对应的保留站和 ROB 里，并且获得该条指令在 ROB 里的位置，并且使用这个位置来进行 renaming。ROB 在这个阶段需要把这个指令要占用的寄存器地址发射出去，在 regstatus 里打上标记。

2.2　execution

execution 阶段主要是各保留站在工作，在时钟周期开始的时候，各保留站检查一下站内的指令，如果发现有一条指令 ready 了，就把它进行运算，并把所得的结果和其在 ROB 中的位置一同广播出去。一共有两个 CDB，一个是给 Add 运算使用的，一个是给 Load 运算使用的，对于 Store 和 Bne

运算，则不需要广播，直接传输到 ROB 就行。同时各保留站时刻保持监听 CDB 总线，如果有广播的数据是自己需要的，就把它进行站内的更新。

2.3 commit

commit 阶段基本都是 ROB 在工作，主要负责各指令的写回。ROB 是本框架中唯一一个有权限对寄存器和内存进行修改的元件。对于跳转指令，ROB 则要发射信号给 pcControl，使其更改 pc 值。同时对于有寄存器写入的指令，ROB 将于 regstatus 进行交互，以判断是否将对应的占用的标记去除。

3 具体功能介绍

3.1 regfile regstatus

这两个元件主要负责对寄存器的读写，尤其是标示寄存器的占用情况，对于正在被写入的寄存器，它会被打上 tag 表示正在由哪条指令写入（以 ROBindex 标示），如果未被占用，则会用 invalid 来标志。

3.2 reorderBuffer

ROB 有几大功能：一是应对各个运算元件的数据的 request，因为一个寄存器正在被写入而且对应的指令已经不在保留站里了，那么对应的数据一定在 ROB 里面。二是接收各运算元件的结果，对队列中的指令进行更新。三是写回指令，更重要的是更新 pc，并且决定是否进行 stall 和跳转，具有控制的功能。

4 不足与困难

一是如何保证对内存的操作是有序的，防止 RAW 的 anti-hazard。解决方法有两种，一是对内存操作也开一个 buffer，保证有序，二是保证 ROB 里有 Store 的时候，Load 指令不会上线。

二是如何应对 JALR 操作，因为该指令还涉及到了对寄存器的写入，在我的框架中我把它扔进了加法保留站，从而对分支预测带来了不小的麻烦。

后来经过助教的指点，发现可以直接在 Decoder 阶段就进行跳转，同时将它翻译成一个 Li 指令，而不经过 ROB 的阶段。