RISCV Simulator 作业

2023.6.19

刘祎禹

项目目标

- 了解计算机架构、指令集架构
- 学习调度算法
- 了解硬件设计

Your Task

- 模拟一个 RISCV CPU
- 通过所有下发数据

执行流程

- 从标准输入读入机器指令
- 从内存 0x0000 处开始取指令执行，每次连续取 4 个 2 位十六进制数，组成一条指令（如取到“37 17 00 00”，拼成32位指令“00001737”）
- 执行到指令 0x0ff00513 (li a0,255) 时，向标准输出 输出程序的返回值（一个 0-255 的非负整数），结束模拟。注意：
  - 程序的返回值存在 a0 寄存器里，但是寄存器是 32 位的，返回值是 8 位的，所以你应该输出 a0 的后八位。例如，你的 a0 寄存器是 int 数组 reg 中的 reg[10]，你应该输出的是 ((unsigned int)reg[10]) & 255u。

要求

- 尽可能遵守硬件要求（具体参见 ISA 的 slides）
  - 每个 module 要求可以乱序执行
  - 建议每个储存信息的时候维护新状态和旧状态
- 实现五级流水或 Tomasulo 逻辑
- 数据内存读写和代码内存读写不冲突
- 数据内存访问需要三个周期
- 代码内存读写一个周期可读出一条指令，且不需要等待3个周期
- PC 指针计算和预测可以是组合逻辑

评分标准 – 五级流水

- 支持打乱 stage 执行（模拟出硬件并行）
  - 60% 测试点得分 + 10% 理解得分
  - 仅支持顺序执行：30%
- forwarding：12%
- 二位饱和预测：8%
- Bonus：10%
- Code Review：10%

评分标准 – Tomasulo

- 实现分支预测的 Tomasulo 算法：75% 测试点得分 +15%理解得分
  - 若不支持分支预测：60% 测试点得分 +10% 理解得分
- Bonus：10%
- Cr：10%

Bonus

- 高级分支预测
- 模拟 Cache
- 合理的多级流水
- 多发射

下发数据

- .c: 源代码
- .dump: 内存对应的汇编数据
- .data: 内存数据

.dump

```
Disassembly of section .rom:
00000000 <.rom>:
   0:  00020137        lui   sp,0x20
   4:  040010ef        jal   ra,1044 <main>
   8:  0ff00513        li    a0,255
   c:  000306b7        lui   a3,0x30
  10:  00a68223        sb    a0,4(a3) # 30004 <__heap_start+0x2e004>
  14:  ff9ff06f        j     c <printInt-0xff4>
```

address

machine code

opcode

operand(s)

一些建议

- 了解 RISCV-32I 指令集（除了 FENCE 指令）
- 了解硬件特性
- 写一个不带五级流水的简单模拟器（用于调试）

谢谢