RISCV\_32I Simulator

本次RISCV32I模拟器经过大大小小的几次改动，主要版本如下：

1. 串行版本

串行版本的实现方式是将所有指令均设成类，IF,ID,EX,MA,WB五个阶段设为成员函数，寄存器、程序计数器等设为了全局变量。首先有一个Operations的大类，然后根据指令类型设定R、I、S、B、U、J六个子类，最后根据指令类型将每条指令设为这六个类的子类。

此版本实现时的主要问题是指令实现出现错误，其间移位操作立即数的位数定义没弄清楚，然后符号右移与逻辑右移弄反，检查这些小错误花费了很长时间。

1. 并行版本（暂停）

为实现并行进行的改动是增加了五个缓冲区Dec,Exe,Mem,Write,rg,分别用于存储每条指令在IF,ID,EX,MA,WB阶段的运行结果。其中rg寄存器设立的目的是为了模拟并行，五个阶段按WB,MA,EX,ID,IF顺序执行，为了防止WB对当前周期的寄存器取值造成影响，先将值存入rg中，一个周期结束后再将rg中的值存入寄存器Rg[32]。

暂停处理数据冲突即控制冲突的实现方式是为每个阶段都设定一个status，0表示当前周期正常执行；1表示当前周期不执行；其中ID阶段还有status=2,表示遇到数据冲突，需要暂停。此外还对32个程序寄存器即PC寄存器设置了locked参数，表示当前是否能从中取值。

此版本实现时调试了很长时间，最后发现问题在于将!=写为了=！导致判断变成了赋值，此次教训惨重，以后一定要注意书写，最好要习惯将常量放在左边。

1. 并行版本（forwarding）

实现此版本进行的改动是取消了32个程序寄存器的locked参数。实现数据转发的方式是直接在ID读取rs1,rs2寄存器的值时判断EX,MA,WB阶段执行的指令是否会向rs1,rs2中写值，如果有，就直接在结果中取。

唯一一个特殊情况是load指令后下一个指令要取值，此时将下一个指令暂停一周期，等load指令进入到MA阶段是就可以进行数据转发。

1. 并行版本（forwarding+分支预测）

实现分支预测进行的改动也比较小，只针对条件分支指令。最开始分支预测方式是假设所有指令都跳转。预测错误的处理方式是将ID,IF阶段的指令都取消，条件分支指令执行到MA阶段时再取新指令。

之后进行的小改动是将预测方式改为反向预测，即向imm<0时全假设跳转,imm>0时全假设不跳转，改动过后预测成功率略有提高。

最后实现的预测方式是在网上查的2bit动态预测，预测成功率最高有95%，最低在60%左右，平均在80%左右。

经过此次工作，我体会到了有条理的代码的重要性，今后一定要努力争取编写清晰易读的代码。同时还要耐心调试，目前的水平写出来的代码都是bug,要有调试的耐心，同时还要不断完善自己的思维，尽量减少写代码的bug。