CS152 Lab 1
Pipelined RV32I Implementation

Version 0.4. 2023-04-25

1. Overview

In this lab, you will take a non-pipelined implementation of a baseline RISC-V processor, and pipeline it. You will observe the performance implications of individual modifications to understand their impact. Performance impact will come from two sources: changing the number of cycles, and changing the speed of the clock.

Due: 2023-05-19 (Fri)

2. What to turn in

You will need to submit two files, collected into a single tar, tgz, or zip file:

1. **Processor.bsv** with your best effort pipeline implementation. It should achieve the highest possible **wall-clock speed**, taking the clock speed into account.
2. A report addressing the questions listed in the "Questions for the report" section. Please read the questions before implementing everything, as some information needs to be collected from partial implementations.
   Please submit in any major document format including .txt, .rtf, .doc, .docx, .odt, .abw, .wpd, or .pdf files.

3. Tools and Code

The tools and code involved in this lab will be introduced during class, and slides uploaded.

All tools will be pre-installed in a Virtualbox VM. You can SSH into the VM after it boots on your local machine, using your favorite SSH client (PuTTY, MobaXTerm, etc). If you have trouble setting things up, we can provide you an account in one of our lab servers.

The VM image can be downloaded from here:
https://drive.google.com/file/d/1plT9o1QIeDkci0l_jB4Si9BTwmqmeIgF/view?usp=share_link

Apple M1 users have run into problems with the provided VM. We can provide you with server accounts in this case.

4. Tasks

4.1. Performance measurement of original architecture

Let's define performance as (IPC × Clock speed), where IPC is instructions per cycle, and clock speed is in MHz.

Remember, IPC can be obtained from the debug output from the simulation (You will need to do some arithmetic to get IPC), and clock speed can be known from the build log for hardware synthesis. System simulation logs are in system.log, and hardware synthesis logs can be generated via "make | tee build.log". Inside build.log, look for "Max frequency for clock [...omitted....] : XXX. MHz (PASS at 25.00 MHz)". XXX is what we are interested in. There is likely two lines in this format. The second instance is what we are interested in.

4.2. Implementation and measurement of the Mul Instruction

The originally provided processor does not implement the M extension of RISC-V, and does not support the multiplication instruction. The provided benchmark, "minisudoku" does use multiplication instructions. As a result, the processor halts when it encounters the multiplication instruction and complains.

This task asks you to implement the multiplication instruction by modifying the Decode and Execution stages, and measure if it has any impact on performance.

4.3. Implementation and measurement of stalling-based pipelining

This task asks you to implement pipelining based on stalling, as described in the accompanied presentation slides, and measure its performance impact. (Remember, performance = IPC times clock speed)

This may involve implementing a branch predictor. (PC+4, or something more complex) You could decide to stall always until the branch target is discovered, or to forward between execute and fetch without a branch predictor. These approaches will all have their pros and cons.

4.4. Implementation and measurement of forwarding

This task asks you to implement forwarding for RAW hazards, and measure the performance impact. (Remember, performance = IPC times clock speed)

Also remember, forwarding can be done between the execution stage and decode stage, as well as between memory/writeback and decode stage as well. These approaches may have their pros and cons.

5. Questions for the report

1. What is the performance of the original processor design?
2. Did the implementation of Mul impact performance? Why or why not?
3. What is the performance of your best pipeline?
4. Do you think forwarding is worth it? Why or why not?

6. Grading

Correct implementation of Mul: 20%
Processor performance: 50%
Report: 30%

Processor performance will be given full marks if it is better than a reference implementation of a stalling-based pipeline with PC+4 branch predictor.