CS152 Lab 2
Profiling a Pipelined RV32I Implementation

Version 0.3. 2022-02-09

1. 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. It will be evaluated using a vanilla rv32i implementation, **without the Mul instruction**.
2. A report addressing the questions listed in the following 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.

2. Questions for the report

2.1. Performance measurement of the original code

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.

Question 2.1.1: What is the performance of the original non-pipelined processor on the sudoku benchmark? Ignore the fact that execution fails before reaching the end of the program due to the missing Mul instruction implementation.

2.2. Performance impact of the Mul instruction

Question 2.2.1: What is the performance of the new processor with the Mul instruction implementation?

Question 2.2.2: What do you think is the reason for this performance change?

2.3. Performance impact of stalling-based pipelining

Question 2.3.1: What is the performance of the new processor with stalling-based pipelining?

Question 2.3.2: What is the performance of the new processor with stalling-based pipelining, **but without the Mul instruction?**

Question 2.3.3: Was this change worth it? Why or why not?

2.4. Performance impact of forwarding

Question 2.4.1: What is the performance of the new processor with forwarding?

Question 2.4.2: What is the performance of the new processor with forwarding, **but without the Mul instruction?**

Question 2.4.3: Do you think the addition of forwarding is worth it? Why or why not?

2.5. Profiling performance

Question 2.5.1: What is the access latency of instruction and data memory? You can look at system.log and eyeball how fast a particular instruction (you can tell apart different instructions by their PC values), or add some logging code to Processor.bsv using "$write".

Question 2.5.2: Are the pipeline FIFO sizes and Scoreboard depth sufficient to accommodate this memory latency? Are they too large or two small?

Question 2.5.3: What is the percentage of stalled cycles in the decode stage, due to the scoreboard? How much of that is RAW stalls, and how many are Load-Use stalls?

Question 2.5.4: What is the percentage of mis-predicted instructions?

Question 2.5.5: Give the above profiling results, what would you say is the most pressing issue to solve, to achieve higher performance?