Lab 1 Report

Performance drop compared to ideal pipeline

For the microbenchmark we used, we focused on testing data hazards. We ran a loop 100,000 times which made the set up instructions negligible in terms of instruction count. There were 20 assembly instructions in our loop.

For Q1, every type of hazard causes a stall of 2 cycles in the pipeline. There were 9 instructions dependent on the immediate preceding instruction and 1 instruct dependence on the instruction 2 instructions ago. This means that we can calculate the new CPI. [1.95].

$$CPI = \frac{\#\ of\ cycles}{\#\ of\ instructions} = \frac{9 * 2\ extra\ cycles + 1 * 1 extra\ cycle + 20\ instructions}{20\ instructions} = 1.95$$

A value of 1 million data hazards in q1 agrees with our anticipated results.

For Q2, there were 2 fewer hazards than in q1 and many of the remaining hazards only require a single stall. This leads to a CPI of 1.5 for the architecture in q2.

$$CPI = \frac{\#\ of\ cycles}{\#\ of\ instructions} = \frac{2 * 2\ extra\ cycles + 6 * 1 extra\ cycle + 20\ instructions}{20\ instructions} = 1.50$$

Microbenchmark Statistics

We created our microbenchmark to test every unique stall case. We divided instructions into two categories 1) where the result is available after ALU stage(s) and 2) where the result is available after the memory access stage. For Q1, this distinguishment was not necessary since we could only access the results at the write back stage. We used an instruction what was depend on an instruction two instructions ago to create a single cycle stall. For Q2, we divided up the different cases for when the next instructions needed the result from the previous instruction. We tested ALU after memory operation, ALU after ALU operation, store (address) after ALU. We have labeled our microbenchmark with the corresponding assembly instructions for further reference. We used the O0 flag to compile our microbenchmark to avoid unanticipated optimizations.