Lab Report

Performance Drop

To compute the new CPI, we first obtain the count of instructions that experience 1 stall and 2 stalls due to RAW hazards. Subsequently, we employ the following formula:

$$\text{new CPI} = 1 + \frac{\text{total \# of 1 stall instructions}}{\text{total \# of instructions}} + \frac{2 * \text{total \# of 2 stall instructions}}{\text{total \# of instructions}}$$

To determine the percentage decrease in performance, we utilize the following formula:

$$\text{performance drop} = 100\% \times \left(1 - \frac{\text{old CPI}}{\text{new CPI}}\right)$$

With gcc.eio as our reference benchmark, we determined for P1,

new CPI = **1.6642**. performance drop = **39.9%**

By utilizing the same CPI calculation methodology. We have calculated for P2,

new CPI: **1.3903** performance drop: **28.1%**

Microbenchmark

For the microbenchmark, we use the -o0 compilation flag to ensure that the assembly code is generated as expected. Each testcase is composed of a while loop with a number of iterations. The validation method we use is as follows: first, we take a reasonably large number of iterations and calculate its stalls count. Once calculated, we increment the iteration count by 1 and recalculate the stalls count, then compare it with the previous observation to see if number increases as expected.

In mbq1.c we have 2 basic testcases and 5 advanced testcases supporting our validation of code. Basic testcases(switch 1,2) simulates,

1. two cycle stalls
2. one cycle stall

While for advanced testcases(switch 3~6) simulates,

3. two cycle stalls + two cycle stalls(propagated)
4. one cycle stall + two cycle stalls

5. one cycle stall + one cycle stall(propagated)
6. Priority question on taking whether one stall or two stalls
7. two cycle stalls + one cycle stall(propagated)