ECE552 Lab 1 Report

Let CPI_{HAZARD} be the CPI of a program with RAW hazards. It is calculated as the following:

$$CPI_{HAZARD} = 1 + (num\_1cycle\_stall + 2*num\_2cycle\_stall)/sim\_num\_insn$$

where num_1cycle_stall = number of 1 cycle stall
num_2cycle_stall = number of 2 cycle stalls
sim_num_insn = number of total instructions

Question 1:

The % performance drop is calculated as:

$$\% \text{ Slowdown} = (CPI_{HAZARD\_Q1} - CPI_{IDEAL}) / CPI_{IDEAL}$$
$$= (1.5 - 1)/1$$
$$= 50\%$$

Our microbenchmark creates per-iteration RAW hazards in this ratio: two 2-cycle stalls (back-to-back ALU→ALU and LOAD→use) and one 1-cycle stall (ALU, one unrelated ALU, then dependent ALU).

The instructions were one XOR command, followed by an ADD. XOR was used because the compiler kept optimizing two back to back ADDs or Subtracts so that we wouldn't see the hazard reflected in the dump as expected. The second set of instructions was an add followed by an intermediate instruction then another add dependent on the first add. The last set of instructions was a load, via a pointer, followed by immediately using that loaded value. The ratio was 2:1 for 2 cycle vs 1 cycle stalls.

The run statistics match this design:

- num_2cycle_stall_q1 = 2,003,037 ≈ 2× num_1cycle_stall_q1 = 1,000,953
- sim_num_RAW_hazard_q1 = 3,003,990 ≈ (1×1-cycle + 2×2-cycle) per iteration aggregated

We compiled with -O1 because after comparing the mbq1.dump file to the c code, we confirmed the three patterns we expected. We ran the microbenchmark in a loop with 1 million iterations to drown out any hazards during initialization.

Question 2:

The % performance drop is calculated as:

$$\% \text{ Slowdown} = (CPI_{HAZARD\_Q2} - CPI_{IDEAL}) / CPI_{IDEAL}$$
$$= (1.4 - 1)/1$$
$$= 40\%$$