Bing Li, 1004191910
Tapasvi Patel, 1004176606

1.0 Performance

$CPI = \sum(\frac{number\ of\ cycles\ per\ instruction_i}{frequency\ of\ instruction_i})$, where i is the executed instruction. For our case, the ideal instruction has CPI=1. An instruction with 2 stalls adds $2 * \frac{number\ of\ times\ instruction\ occurs}{total\ number\ of\ instructions}$ CPI - similarly for 1 stall.

1.1 Question 1 Performance Drop

$Calculated\ CPI = 1.0 + (\frac{hazard\_q1\_1\_stall}{num\_insn}) * 1 + (\frac{hazard\_q1\_2\_stall}{num\_insn}) * 2 = 1.6551$

$\%\ Drop = (\frac{Ideal\ CPI - Calculated\ CPI}{Ideal\ CPI}) * 100 = (\frac{1-1.6551}{1}) * 100 = -65.51\%$ = 65.51% slowdown

1.2 Question 2 Performance Drop

$Calculated\ CPI = 1.0 + (\frac{hazard\_q2\_1\_stall}{num\_insn}) * 1 + (\frac{hazard\_q2\_2\_stall}{num\_insn}) * 2 = 1.4129$

$\%\ Drop = (\frac{Ideal\ CPI - Calculated\ CPI}{Ideal\ CPI}) * 100 = (\frac{1-1.4129}{1}) * 100 = -41.29\%$ = 41.29% slowdown

2.0 Microbenchmark

build command: `ssbig-na-sstrix-gcc mbq1.c -O0 -o mbq1` run command: `./sim-safe mbq1`
The microbenchmark was used to test the correctness of our solution. We used an optimization level of 0. For question 1, we used one counter to count the number of 1 stalls, another counter to count the number of 2 stalls and a final counter to count the number of 1 stalls plus 2 stalls. The benchmark was run two times. Once with the for loop running 10M times, and another with the for loop running 100M times. The results are summarized below.

	10M Loops		100M Loops	
Counter	Expected	Actual	Expected	Actual
sim_num_RAW_hazard_q1_1_stall	10,000,000	10,000,146	100,000,000	100,000,146
sim_num_RAW_hazard_q1_2_stall	70,000,000	70,000,824	700,000,000	700,000,824
sim_num_RAW_hazard_q1	80,000,000	80,000,970	800,000,000	800,000,970

The answers above clearly indicate that the changes made to the simulator are working. When running 10M loops, each iteration of the loop should increment the 1 stall counter and the 2 stall counter by 1. That is precisely what is happening for the 1 stall counter. For the two stall counter, we would also expect a count of 10M. However, further inspection of the assembly code shows us that the additional 60M (70M – 10M) comes from the for loop. There are an additional of 6 instructions that cause a 2 stall delay because of the for loop. 6 times 10M loops is an additional 60M counts for 2 stalls. Add in the 10M from our code and it comes out to 70M. The difference of 146 for the 1 stall counter and 824 for the two stall counter come from the other parts of the code (such as the standard library, etc). Adding up the 1 stall and 2 stall counters also equals the total stall counter. The results for this match the 100M loops test case as well.