ECE552 Lab 1 Report

Shawn Zhai 1006979389
Victor Wu 1007069039

1. % Performance Drop

It is calculated using the formula given in the lab handout:

$$drop\% = 100(1 - speedup)$$

Where $speedup = \frac{IC \times CPI_{old} \times TC}{IC \times CPI_{new} \times TC} = \frac{CPI_{old}}{CPI_{new}}$

Where $CPI_{old} = 1,\ CPI_{new} = 1 + \frac{(1 \times num\_inst\_1\_stall + 2 \times num\_inst\_2\_stall)}{num\_total\_inst}$

Q1 drop% = $100(1 - \frac{CPI_{old}}{CPI_{new}}) = 100(1 - \frac{1}{1.6642}) = 39.91\%$

Q2 drop% = $100(1 - \frac{CPI_{old}}{CPI_{new}}) = 100(1 - \frac{1}{1.3903}) = 28.07\%$

2. Microbenchmark

The microbenchmark is written as a 1-million iteration loop that contains code triggering both the 1 stall and 2 stalls RAW hazards. Therefore, in total sim-safe should observe ~2 million hazards, ~1 million is 1 stall, ~1 million is 2 stalls. This is exactly what was observed through running sim-safe, which confirms beyond a doubt that our sim-safe code was working correctly.

The benchmark creates RAW hazards by simply assigning values to a variable (write) and then having another variable read the previously assigned variable (read). For 1-cycle stall, the RAW hazard is created in the following C code, 2-cycle removes the instruction in the middle.

```
a = 1;
// extra inst to make it a 1 cycle stall, remove the line to create 2-cycle stalls
random_inst = 0;
b = a;
```

This produces the following assembly code, causing a 1-cycle stall as R3 is written, followed by non-dependent instruction, then R3 read. For 2-cycle stalls, it is similar but no middle line, only the write and read.

```
li      $3,0x00000001       # 1
move    $8,$0
move    $4,$3
```

Compiler flags used were -o0 and -S (inspect assembly). -o0 kept optimization to minimum to make sure no extra compiler's implicit behavior effect on the generated RAW assembly code.