ECE552 PRA0102 - Station 8
Ting-Hao (Tim) Cheng (997289090)
Isaiah Tan (999004937)

LAB 1 Report

% performance drop vs ideal pipeline with (CPI 1.0) for the two questions.

Please see ***Appendix A*** for sim-safe outputs stat-counters.
Assumptions: clock cycle unchanged for all cases.

Question 1: 5-stage pipeline, no forwarding or bypassing
% of 2 stall hazards = sim_num_RAW_hazard_q1_2stall/sim_num_insn = 28.98499854%
% of 1 stall hazards = sim_num_RAW_hazard_q1_1stall/sim_num_insn = 6.28285967513 %
new CPI = 1.0 + 0.2898*2 stalls + 0.06283*1 stall = 1.643
slowdown = CPI/idea CPI = 1.655X => 64.3 % performance drop

Question 2: 6-stage pipeline, 2 EX stages. Assume an ideal memory system and ignore control hazards.
% of 2 stall hazards = sim_num_RAW_hazard_q2_2stall/sim_num_insn = 7.23642996762%
% of 1 stall hazards = sim_num_RAW_hazard_q2_1stall/sim_num_insn = 26.1243578196%
new CPI = 1.0 + 0.0723*2 stalls +0.2612*1 stall = 1.406
slowdown = CPI/idea CPI = 1.406X => 40.6 % performance drop

Briefly explain how your microbenchmark collected statistics validate the correctness of your code for the first problem statement. Feel free to refer to comments within the mbq1.c file, as needed. Specify which compilation flags you used.
We tried to create a controlled number of RAW hazards with our microbenchmark code by first assigning a new, calculated value to a variable and then subsequently use that variable to compute something else. Please see ***Appendix C*** for the while loop that simulates RAW hazards 100000 times.

We then examined the resulting assembly code to make sure that the program would have the desired behavior. We compiled the microbenchmark with -O2 flag, which resulted in the assembly code as seen in ***Appendix D***.

Note that we used no-op to help us better identify the while loop in assembly. We can see that there are 2 different RAW hazards - one that would cause 1 stall and another that would cause 2 stalls. Our loop intentionally created the one-stall hazard (it would've been 2 stalls), and the program has a two-stall hazard that's unintentional. This means that if we run the loop 100,000 times, we would count approximately 2x the number of one-stall hazards and 1x the number of two-stall hazards.

The microbenchmark result validates the correctness of our code. (***Appendix B***)
sim_num_RAW_hazard_q1_1stall 200789 # total number of (1 stall) RAW hazards (q1)
sim_num_RAW_hazard_q1_2stall 102368 # total number of (2 stall) RAW hazards (q1)

APPENDIX

A: sim: ** simulation statistics ** (gcc.eio)

```
sim_num_insn            279373007 # total number of instructions executed
sim_num_RAW_hazard_q1_1stall    17552614 # total number of RAW hazards (q1)
sim_num_RAW_hazard_q1_2stall    80976262 # total number of RAW hazards (q1)
sim_num_RAW_hazard_q2_1stall    72984404 # total number of RAW hazards (q2)
sim_num_RAW_hazard_q2_2stall    20216632 # total number of RAW hazards (q2)
sim_num_RAW_hazard_q1     98528876 # total number of RAW hazards (q1)
sim_num_RAW_hazard_q2     93201036 # total number of RAW hazards (q2)
CPI_from_RAW_hazard_q1      1.6425 # CPI from RAW hazard (q1)
CPI_from_RAW_hazard_q2      1.4060 # CPI from RAW hazard (q2)
```

B: sim: ** simulation statistics ** (mbq1)

```
sim_num_insn              511754 # total number of instructions executed
sim_num_RAW_hazard_q1_1stall      200690 # total number of RAW hazards (q1)
sim_num_RAW_hazard_q1_2stall      102379 # total number of RAW hazards (q1)
sim_num_RAW_hazard_q1       303069 # total number of RAW hazards (q1)
CPI_from_RAW_hazard_q1      1.7923 # CPI from RAW hazard (q1)
```

C: mbq1 code - simulating RAW hazards

```
while(i < 100000){
        temp = temp + 5; // Create a RAW hazard.
        sum = temp + 5;
        i = i + 1;
}
```

D: mbq1 while loop assembly code

```
#NO_APP
        li      $4,0x00010000           # 65536
        ori     $4,$4,0x869f
$L16:
        addu    $5,$5,5         # temp = temp + 5
        addu    $3,$3,1         # i = i + 1 being run here
        addu    $16,$5,5        # sum = temp + 5 (RAW hazard with 1 stall)
        slt     $2,$4,$3        # RAW hazard (1 stall)
        beq     $2,$0,$L16      # RAW hazard (2 stalls) that is unintentional
#APP
        nop
#NO_APP
```