Mohammed Orhun SAKE
26754

Homework #3

Assigned: 18/04/2023
Due: 30/04/2023

1. Consider a five-stage pipelined datapath for MIPS architecture with the following latencies:

IF	ID	EX	MEM	WB	Pipeline registers
235 ps	115 ps	165 ps	235 ps	100 ps	15 ps

Note that pipeline registers also have some latency, namely 15 ps (last column in the table). (**20 pts**)

→ 3.4 times faster

a. Assuming there are no stalls, what is the speedup achieved by pipelining a single-cycle datapath? (**10 pts**)

1 cycle => 235 + 115 + 165 + 235 + 100 = 850 ps

Longest => 235 + 15 = 250 ps

speed-up => 850/250 = 3.4 (times faster)

b. Assume that instructions of a benchmark executed by the pipelined processor are broken down as follows:

→ 5.03 clock cycle

Load	Store	Branch	Jump	R-type
30%	15%	10%	5%	40%

Also assume that 40% of loads are immediately followed by an instruction that uses the result of load; and branch penalty on misprediction is three clock cycles and 20% of branches are mispredicted. Jumps take two clock cycles to execute. Compute CPI of the pipelined processor (**10 pts**).

0.1 + 1.62 + 0.56 + 0.75 + 2 = 5.03 cc

Jump = 2 * 0.05 = 0.1 cc

Load = 0.3 * ((0.4 * 6) + (0.6 * 5)) = 1.62 cc
(0.4 * 6): immediate, (0.6 * 5): wait

Branch = 0.1 * ((0.2 * 8) + (0.8 * 5)) = 0.56 cc
(0.2 * 8): miss, (0.8 * 5): hit

store = 0.15 * 5 = 0.75 cc

R-type = 0.4 * 5 = 2 cc

2. Consider the following 10-bit floating-point number system in which we use 1 bit for the sign, 4 bits for the exponent, and 5 bits for the significand (mantissa). The exponent bias is equal to 7, and the largest and smallest biased exponents are reserved (similar to IEEE 754 Standard). The significand uses a hidden (implicit) 1. The value of a floating-point number can be calculated using the following expression:

$$value = (-1)^{sign} \times (1.0 + significand) \times 2^{exponent-bias}$$ (**20 pts**)

a. Find floating-point representation of the following real numbers: (**10 pts**)

x = 45.0

y = 0.75

z = -44.0

b. Compute `t = x+y` using precise floating-point arithmetic with guard (G), Round (R), and Sticky (S) bits. Use round-to-nearest-even scheme if you need to. (**5 pts**)

							G	R	S
x	1.	0	1	1	0	1	0	0	0
y									
t									
t									

c. Compute `v = t+z` the same way. Is what you found correct? (**5 pts**)

			G	R	S
t	1.				
z	-				
v					

3. Assume that your benchmark has the following instruction frequencies:

	R-type	branch	the rest
Benchmark	40%	20%	40%

Also assume the following branch predictor accuracies:

	Always-taken	Always-not-taken	Dynamic predictor
Benchmark	45%	55%	60%

Assume also that CPI = 1 without branch mispredictions. (**20 pts**)

a. Stall cycles due to mispredicted branches increase CPI. Assume that the branch outcomes are determined in **`MEM`** stage, that there are no data hazards, and that no delay slots are used. Calculate the CPI after the stalls due to mispredicted branches with "Always-taken", "Always-not-taken", and "Dynamic" predictors? (Hint: When a branch is mispredicted, the instructions in **`IF`**, **`ID`**, and **`EX`** stages must be flushed out of the pipeline) (**10 pts**)

If + ID + EX => 3 cc

Always-taken → (0.8 * 1) + 0.2 * (0.45 * 1 + 0.55 * 4) = 1.33

Always-not taken → (0.8 * 1) + 0.2 * (0.55 * 1 + 0.45 * 1) = 1.27

Dynamic → (0.8 * 1) + 0.2 * (0.60 * 1 + 0.4 * 4) = 1.24

b. With the "dynamic" predictor, what speedup would be achieved if we eliminated half of the branches by turning each branch into two R-type instructions? Assume that the performance of the predictor improves to 90% after this modification. (**Hint**: `clock count = IC × CPI`) (**10 pts**)

Start = 100 * 1.24 = 124

0.4 R + 0.4 T + 0.2 B => 0.6 R + 0.4 T + 0.1 B

=> CPI → 100/110 + (10/110 * (0.9 + 0.4)) = 1.027

=> END = 110 * 1.027 = 112.97

(124)/(112.97) = 1.0976

4. Consider a program consisting of five conditional branches. Below are the outcomes of each branch for one execution of the program (T for taken, N for not taken).

   Branch 1: T-T-T
   Branch 2: N-N-N-N
   Branch 3: T-N-T-N-T-N
   Branch 4: T-T-T-N-T
   Branch 5: T-T-N-T-T-N-T

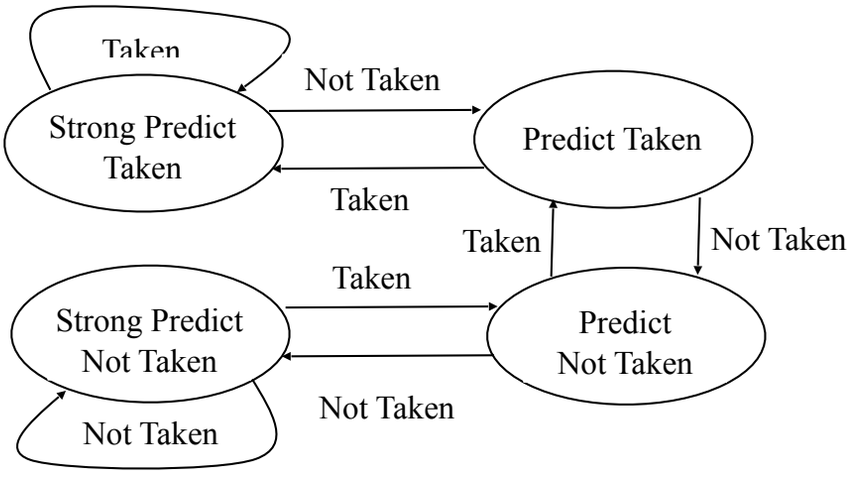
   For dynamic schemes, assume each branch has its own prediction buffer. List the predictions for the following branch prediction schemes for the execution of the code above. What are the total prediction accuracies? (**20 pts**)

   a. There is one 1-bit dynamic predictor for each branch, and each is initialized to "Predict Not Taken". (**5 pts**)

Branch 1: N - T-T → 2/3
Branch 2: N-N-N-N → 4/4
Branch 3: N-T-N-T-N-T → 0/6
Branch 4: N - T-T-T-N → 2/5
Branch 5: N-T-T-N-T-T-N → 2/7

   b. You use 2-bit dynamic predictor and that each branch has its own prediction buffer which is initialized to "Predict Not Taken". Compute the prediction accuracy for each branch and overall branch prediction accuracy. (**15 pts**)

Branch 1: N-T-T → 2/3
Branch 2: N-N-N-N → 4/4
Branch 3: N-T-N-T-N-T → 0/6
Branch 4: N-T-T-T-T → 3/5
Branch 5: N-T-T-T-T-T-T → 4/7



5. (**25 pts**) Consider the following piece of code that will be executed in a five-stage pipelined datapath of MIPS processor:

```
1  lw  $t1, -16($t5)
2  add $t6, $t2, $t2
3  sw  $t6, 50($t1)
```

Assume that a value written to a register can be read in the following clock cycle.

a. Indicate dependencies. (**5 pts**)

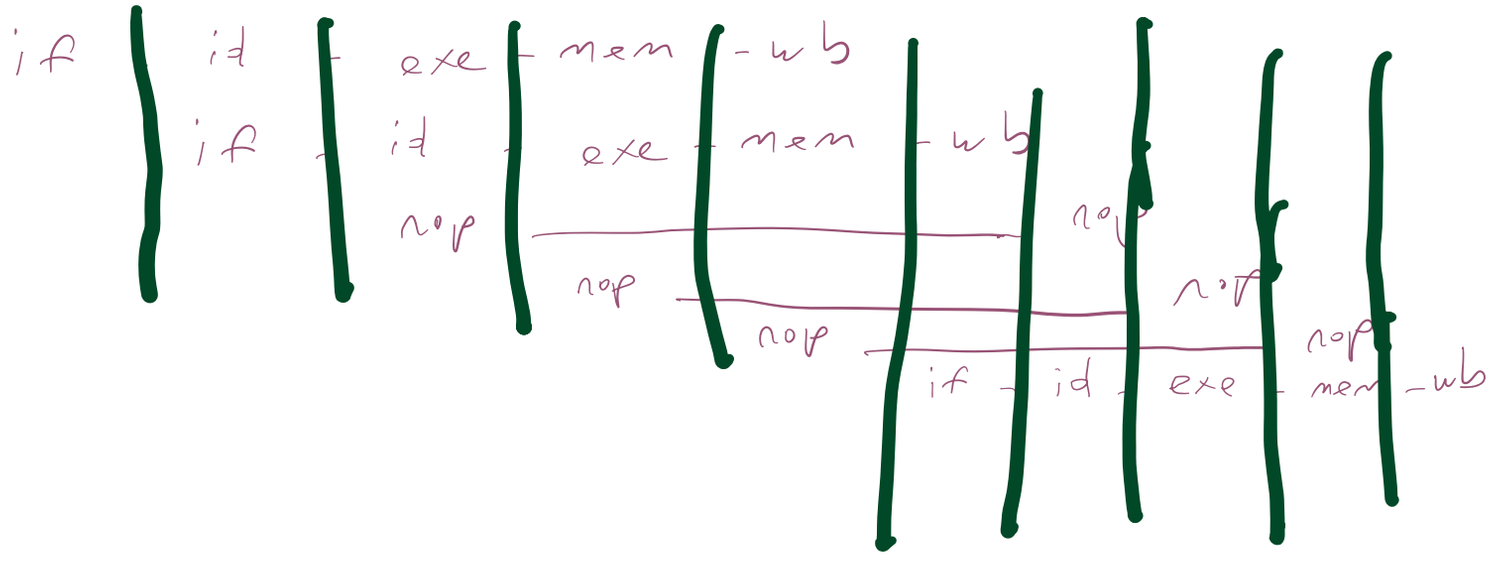
1 → No dependency t_1 on "mem"

2 → No dependency t_6 on "exe"

3 → Depends on 1-2 → t_1 and t_6 dependency

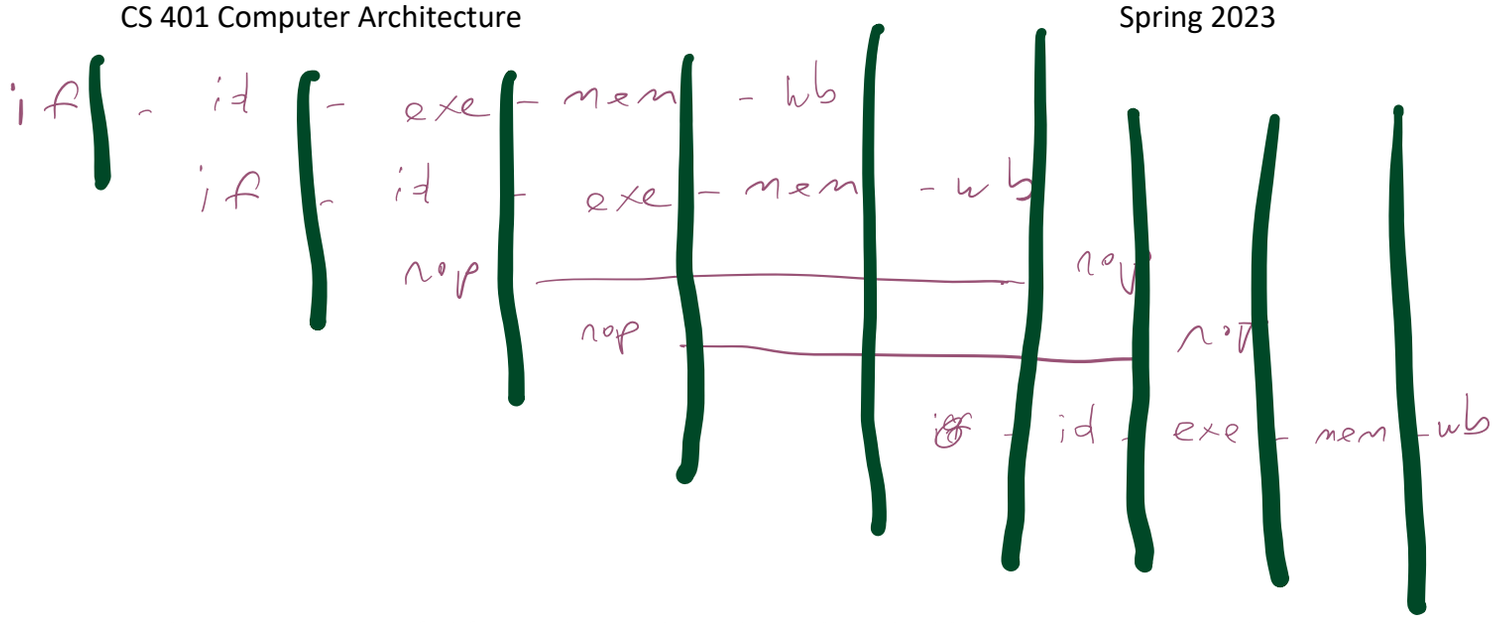
b. Assume that there is no forwarding in this pipelined processor. Indicate hazards and add **nop** instructions to eliminate them. (**5 pts**)

t_1 and t_6 must be available for 3rd instruction



c. Assume there is forwarding only from the ALU. Indicate hazards and add nop instructions to eliminate them. (**5 pts**)

ALU forwarding → 3rd instruction must wait for 1 clock "wb" stage



d. Assume there is full forwarding. Indicate hazards and add `nop` instructions to eliminate them. (**5 pts**)

forwarding removes dependencies

→ No need for "nop" instructions

Use the following clock cycle times for the remainder of this exercise. Notice that the forwarding technique complicates the data path, which can result in slower clock frequency.

Without forwarding	With ALU forwarding	With full forwarding
300 ps	360 ps	400 ps

e. What is the total execution time of this instruction sequence without forwarding, with ALU-forwarding and with full forwarding? (**5 pts**)

without => 300 * 10 = 3000 ps

ALU => 360 * 10 = 3600 ps

Full => 400 * 7 = 2800 ps