Homework #2

Assigned: 7/04/2021
Due: 14/04/2021

Note: Only hardcopy submissions are acceptable

1. (**50 pts**) Consider the following piece of code that will be executed in a five-stage pipelined datapath of MIPS processor:

```
lw  $t2, 4($t5)
add $t7, $t3, $t3
sw  $t7, -24($t2)
```

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

a. Indicate dependencies. (**10 pts**)

There is no dependency for t5 in the first lw instruction. t2 will be ready after MEM stage because it is a load instruction.
There is no dependency for t3 in the second instruction. t7 will be ready after EXE stage because it is an arithmetic operation.
For third instruction, t2 depends on first instruction and t7 depends on second instruction. So third instruction depends on first 2 instructions, that's why third instruction should either wait for first 2 instructions should be completed or we should use forwarding to try to eliminate these hazards.

b. Assume that there is no forwarding in this pipelined processor. Indicate hazards and add **nop** instructions to eliminate them. (**10 pts**)

There are 2 data hazards because of t2 an t7. t2 and t7 should be ready for ID stage of the 3rd instruction. Since there is no forwarding, we eliminate these hazards by making sure that both first and second instruction's WB stage is completed before third instruction's ID stage so we add 3 cycles of nop instructions. (There is no WB stage for sw instruction)

IF	ID	EXE	MEM	WB				
	IF	ID	EXE	MEM	WB			
		NOP	NOP	NOP	NOP	NOP		
			NOP	NOP	NOP	NOP	NOP	
				NOP	NOP	NOP	NOP	NOP
					IF	ID	EXE	MEM

c. Assume there is forwarding only from the ALU. Indicate hazards and add `nop` instructions to eliminate them. (**10 pts**)

If there is forwarding from the ALU, second instruction can forward t7 to third instruction after EXE stage. But third instruction still must wait for first instruction to complete WB stage since there is no forwarding there.

IF	ID	EXE	MEM	WB			
	NOP	NOP	NOP	NOP	NOP		
		NOP	NOP	NOP	NOP	NOP	
			IF	ID	EXE	MEM	WB
				IF	ID	EXE	MEM

d. Assume there is full forwarding. Indicate hazards and add `nop` instructions to eliminate them. (**10 pts**)

With full forwarding, t2 will be ready after MEM and can be forwarded to EXE of 3rd instruction. t7 will be ready after EXE and can be forwarded to EXE of 3rd instruction. No need to add NOP instructions.

IF	ID	EXE	MEM	WB	
	IF	ID	EXE	MEM	WB
		IF	ID	EXE	MEM

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? (**10 pts**)

Without forwarding (b): 9 clock cycles → 9 * 300 = 2700 ps
ALU-forwarding (c): 8 clock cycles → 8 * 360 = 2880 ps
Full forwarding (d): 6 clock cycles → 6 * 400 = 2400 ps

2. (**25 pts**) Consider the following code segment executing on a five-stage MIPS processor:

```
      ...
Loop: lw   $t0,  0($s1)
      sub  $t0,  $t0,  $s2
      sw   $t0,  0($s1)
      addi $s1,  $s1,  -4
      bne  $s1,  $zero, Loop
      ...
```

a. During the seventh clock cycle of the first iteration of the loop, which registers are being read and which registers are being written? Assume that data forwarding is used. Do not change the order of the instructions. (**5 pts**)

(**Hint:** You can fill the following table for the schedule of the code)

instruction	CC 1	CC 2	CC 3	CC 4	CC5	CC 6	CC 7	CC 8	CC 9
lw	IF	ID	EX	MEM	WB				
sub		IF	ID	ID	EX	MEM	WB		
sw			IF	IF	ID	EX	MEM		
addi					IF	ID	EX	MEM	WB
bne						IF	ID	EX	MEM

$t0 is being written by sub instruction, $s1 and zero register is being read by bne (but $s1's value is not in the register yet it will be forwarded to bne).

b. Reorder the instructions so that all data and control hazards are eliminated. Assume that **delayed-branch** technique is used. (**Hint:** the delayed branch technique places an independent instruction into the slot following a branch instruction, where this instruction is always executed independent of the branch outcome. Note that the branch outcome is known in the ID stage of the pipeline.) (**10 pts**)

```
      ...
Loop: lw   $t0,  0($s1)
      addi $s1,  $s1,  -4
      sub  $t0,  $t0,  $s2
      bne  $s1,  $zero, Loop
      sw   $t0,  4($s1)
      ...
```

c. Unroll the loop twice and remove all hazards. Assume that **$s1** contains an even number before the loop. (**10 pts**)

```
      ...
Loop: addi $s1,  $s1,  -8
      lw $t0, 4($s1)
      lw $t1, 8($s1)
      sub  $t0,  $t0,  $s2
      sub $t1, $t1, $s2
      sw $t0, 4($s1)
      sw   $t1,  8($s1)
      bne  $s1,  $zero, Loop
      ... (Assuming there is forwarding from MEM)
```

3. (**25 pts**) Consider a five-stage pipelined datapath for MIPS architecture with the following latencies:

IF	ID	EX	MEM	WB	Pipeline registers
335 ps	315 ps	365 ps	335 ps	300 ps	25 ps

Note that pipeline registers also have some latency, namely 25 ps (last column in the table).

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

When pipelined, longest stage is EX (critical path), which lasts for 365 ps + 25 ps = 390 ps by taking pipeline register's latency into consideration.
A single cycle duration is 335 ps + 315 ps + 365 ps + 335 ps + 300 ps = 1650 ps without any pipeline.
1650 ps/ 390 ps = 4.23 approximately, pipelining is 4.23 times faster than single cycle (gained speedup).

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

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 (**15 pts**).

Loads immediately followed by an instruction that uses the result of load causes 1 more cycle delay, because it must wait for MEM stage to be completed. Load, store (taking WB stage into consideration and assuming sw takes 5 cycles as well), correctly predicted branching, R-type instructions take 5 cycles.
Assuming there are X instructions to calculate cycles per instruction.
Load: X*0.3 * (0.6 * 5 + 0.4 * (5+1)) = 1.62X cycles
Store: X*0.15 * 5 = 0.75X cycles
Branch: X*0.1 * (0.8 * 5 + 0.2 * (5+3)) = 0.56X cycles
Jump: X*0.05 * 2 = 0.1X cycles
R-type: X*0.4 * 5 = 2X cycles
Total = 1.62+0.75+0.56+0.1+2 = 5.03*X cycles per X instructions
So CPI is 5.03.