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**)

For the third instruction, both $t7 and $t2 are required. $t7 should be computed by the second instruction, because we store the information in $t7 to the memory, hence there is a dependency between the second and third instruction. Moreover, in the first instruction the required value should be loaded from memory to the register $t2, because we need that base address in the third instruction hence there is a dependency between the first and third instruction.

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

Since we have 2 data dependencies which is described in the first question, we have two data hazards in this case. Both $t2 and $t7 should be written back to the register before the third instruction reaches its instruction decode (ID) stage of the pipeline.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
lw $t2, 4($t5)	IF	ID	EXE	MEM	WB					
add $t7, $t3, $t3		IF	ID	EXE	MEM	WB				
			NOP	NOP	NOP	NOP	NOP			
				NOP	NOP	NOP	NOP	NOP		
					NOP	NOP	NOP	NOP	NOP	
sw $t7, -24($t2)						IF	ID	EXE	MEM	WB

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

Since we have forwarding from ALU, we do not have to wait second instruction to be written back to the register, instead we can directly send the value of $t7, after the ALU stage. However, there is still data hazard between the first and third instruction, the value in 4($t5) should be written from the memory to the register $t2. Hence, we still have to wait for the first instruction to complete its write back operation.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
lw $t2, 4($t5)	IF	ID	EXE	MEM	WB					
add $t7, $t3, $t3		IF	ID	EXE	MEM	WB				
			NOP	NOP	NOP	NOP	NOP			
				NOP	NOP	NOP	NOP	NOP		
sw $t7, -24($t2)					IF	ID	EXE	MEM	WB	

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

Since we have full forwarding, we can directly send the value of $t2 to the third instruction after the MEM stage of the first instruction is completed. We can also send the value of $t7 to the third instruction after the EXE stage of the second instruction is completed.

	C1	C2	C3	C4	C5	C6	C7
lw $t2, 4($t5)	IF	ID	EXE	MEM	WB		
add $t7, $t3, $t3		IF	ID	EXE	MEM	WB	
sw $t7, -24($t2)			IF	ID	EXE	MEM	WB

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:

Since, all instructions are completed after 10 cycles without forwarding, total execution time is: 300 * 10 = **3000 ps**

ALU forwarding:

Since, all instructions are completed after 9 cycles with ALU forwarding, total execution time is: 360 * 9 = **3240 ps**

Full forwarding:

Since all instructions are completed after 7 cycles with full forwarding, total execution time is: 400 * 7 = **2800 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	IF	ID	EXE	MEM	WB		
sw				IF	ID	EXE	MEM	WB	
addi					IF	ID	EXE	MEM	WB
bne						IF	ID	EXE	MEM

During the seventh clock cycle of the first iteration of the loop, $t0 is being written and $s1 is being read.

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 $t0, 8($s1)
         lw $t1, 4($s1)
         sub $t0, $t0, $s2
        sub $t1, $t1, $s2
        sw $t0, 8($s1)
        sw $t1, 4($s1)

        bne $s1, $zero, Loop
```

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**)

Single Cycle Path:
Total execution time = 335 + 315 + 365 + 335 + 300 = 1650 ps

Critical Path for pipeline:
Since MEM operation has the longest execution time in the pipeline, critical path is = 365ps + 25ps (pipeline registers) = 390ps

Speedup = 1650 / 390 = 4.23

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**).

CPI = Number of clock cycles / instruction count

Suppose we have **x** instructions:

Cycle count for Load:

40% of the load operations will cause additional 1 cycle because for the consecutive load operations, since the following one uses the result of the previous load operation, the value of the previously loaded one will be available after its MEM stage which causes additional 1 cycle bubble.

Cycle count = (3*x / 10) * (40/100) * 6 + (3*x / 10)*(60/100)*5

Cycle count for Store:
Cycle count = (15*x / 100) * 5

Cycle count for Branch:
Mispredicted branch operation will cause additional 3 clock cycles.
Cycle count = (8 *x / 100) * 5 + (2*x / 100) * 8

Cycle count for Jump:
Cycle count = (5*x / 100) * 2

Cycle count for R-type:
Cycle count = (40*x / 100) * 5

Total cycle counts = 5.03 * x
Total instructions = x
CPI = 5.03 * x / x = **5.03**

Efe Şencan