16.482 / 16.561: Computer Architecture and Design

Summer 2015

Homework #3 Solution

Consider the following sequence of instructions:

```
loop:     add  $t0, $t1, $t2
          lw   $t3, 10($t0)
          lw   $t4, 14($t0)
          sub  $t5, $t4, $t3
          sw   $t5, 18($t0)
          addi $t2, $t2, 4
          slti $t6, $t2, 200
          bne  $t6, $zero, loop
```

Assume each datapath stage requires the following amount of time to complete:

- *Instruction fetch (IF): 30 ns*
- *Instruction decode (ID): 20 ns*
- *Execute / address calculation (EX): 25 ns*
- *Memory access (MEM): 30 ns*
- *Register write back (WB): 20 ns*

1. *(20 points) How long will a single iteration of this loop take in a single-cycle datapath?*

Solution: For the single cycle datapath, we must determine the total time for the longest instruction, which uses all 5 stages: 30 + 20 + 25 + 30 + 20 = 125 ns. Since the sequence has 8 instructions, the total time is: 8 * 125 = **1000 ns**

2. *(20 points) If we assume ideal pipelining (i.e., no hazards and therefore no stalls), how long will one loop iteration take in a pipelined datapath?*

Solution: Note first that the cycle time is based on the longest individual stage: 30 ns. There are two ways to solve this problem. The first is the equation given in class: given M instructions, and N pipeline stages, the total number of cycles is N + (M-1). Here, N = 5 and M = 8, so the total number of cycles is 5 + (8-1) = 12, and the total time is therefore **360 ns**.

We can also draw a pipeline diagram that shows the same result:

Cycle

	1	2	3	4	5	6	7	8	9	10	11	12
`add`	IF	ID	EX	M	WB							
`lw`		IF	ID	EX	M	WB						
`lw`			IF	ID	EX	M	WB					
`sub`				IF	ID	EX	M	WB				
`sw`					IF	ID	EX	M	WB			
`addi`						IF	ID	EX	M	WB		
`slti`							IF	ID	EX	M	WB	
`bne`								IF	ID	EX	M	WB

3. *(20 points) If we now assume a more realistic pipelined datapath* ***without*** *forwarding, how long will one iteration take? Show a revised code sequence that includes all necessary no-ops to support your answer.*

Solution: This code sequence contains several data hazards:

- The two lw instructions need the result of the add
- The sub needs the result of both loads
- The sw needs the result of the sub
- The slti needs the result of the addi
- The bne needs the result of the slti

Recall that in a datapath without forwarding, the ID stage of the dependent instruction must be in the same cycle (or later) as the WB cycle of the producing instruction to allow the dependent instruction to read the correct value. Put more simply, in a 5-stage pipeline, there must be two cycles between any two dependent instructions; after adding no-ops to satisfy this rule, we have the following code sequence:

```
loop:     add  $t0, $t1, $t2
          nop
          nop
          lw   $t3, 10($t0)
          lw   $t4, 14($t0)
          nop
          nop
          sub  $t5, $t4, $t3
          nop
          nop
          sw   $t5, 18($t0)
          addi $t2, $t2, 4
          nop
          nop
          slti $t6, $t2, 200
          nop
          nop
          bne  $t6, $zero, loop
```

This sequence has 18 instructions; using the equation above, the total time is 5+(18-1) = 22 cycles, or 22 * 30 = **660 ns**. Note that we could also use a pipeline diagram to solve this problem.

4. *(20 points) If we now assume a pipelined datapath* ***with*** *forwarding, how long will one iteration take?*

<u>Solution:</u> The only hazards that cannot be completely resolved using forwarding are hazards between a load instruction and a dependent instruction that immediately follows it. The only hazard in this code that qualified is the second lw$\rightarrow$sub dependence; all others can be resolved using forwarding alone.

Therefore, on a pipelined datapath with forwarding, this code sequence requires only one more instruction (a single no-op between the lw and dependent sub instructions) than it would on an ideal pipelined datapath. It would therefore take one more cycle, or 30 ns longer, making the total execution time 360 + 30 = **390 ns**.

5. *(20 points) Calculate the speedup for each of the pipelined datapaths over the single-cycle case. In other words, determine how much faster each of the pipelined cases are than the single-cycle case.*

<u>Solution:</u> Speedup is the ratio between the slower and faster times—in other words, it tells you how many times faster the better solution is. Therefore:

- $\text{Speedup}_{\text{ideal pipelining}} = (1000 \text{ ns}) / (360 \text{ ns}) \approx \mathbf{2.78}$
- $\text{Speedup}_{\text{no forwarding}} = (1000 \text{ ns}) / (660 \text{ ns}) \approx \mathbf{1.52}$
- $\text{Speedup}_{\text{forwarding}} = (1000 \text{ ns}) / (390 \text{ ns}) \approx \mathbf{2.56}$