2024-1 Computer Architecture Homework #3

Due: 6/7 (Fri) 11:59 p.m.

1. [25] When processor designers consider a possible improvement to the processor datapath, the decision usually depends on the cost/performance trade-off. In the following three problems, assume that we are beginning with the datapath from Figure 4.21, the latencies from Exercise 4.7, and the following costs:

I-Mem	Register File	Mux	ALU	Adder	D-Mem	Single Register	Sign extend	Sign gate	Control
1000	200	10	100	30	2000	5	100	1	500

Latencies from Exercise 4.7

I-Mem / D-Mem	Register File	Mux	ALU	Adder	Singlegate	Register Read	Register Setup	Sign extend	Control
250ps	150ps	25ps	200ps	150ps	5ps	30ps	20ps	50ps	50ps

Instruction mix from Exercise 4.8

R-type/I-type (non-lw)	lw	sw	beq
52%	25%	11%	12%

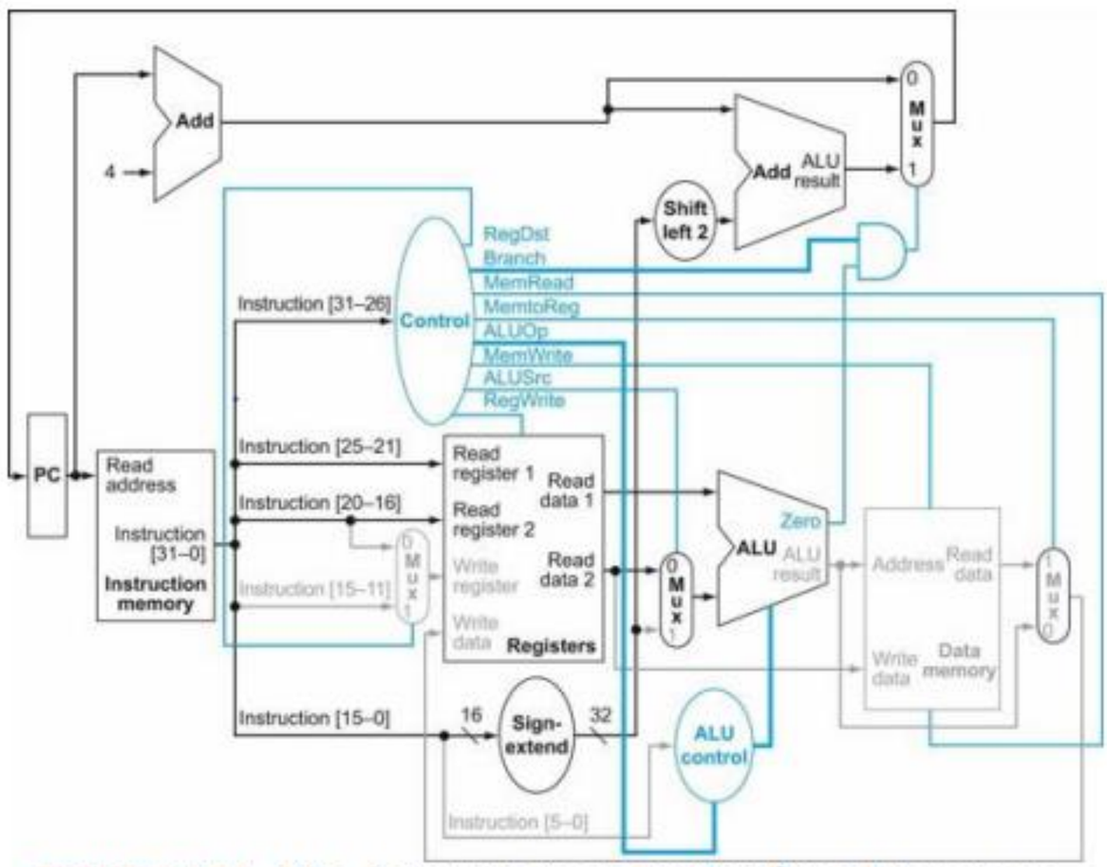

FIGURE 4.21 **The datapath in operation for a branch-on-equal instruction.**
The control lines, datapath units, and connections that are active are highlighted. After using the register file and ALU to perform the compare, the Zero output is used to select the next program counter from between the two candidates.

Suppose doubling the number of general purpose registers from 32 to 64 would reduce the number of lw and sw instruction executed by 12%, but increase the latency of the register file from 150 ps to 160 ps and double the cost from 200 to 400. (Use the instruction mix from Exercise 4.8 and ignore the other effects on the ISA discussed in Exercise 2.18.)

a. [10] What is the speedup achieved by adding this improvement?

b. [10] Compare the change in performance to the change in cost.

c. [5] Given the cost/performance ratios you just calculated, describe a situation where it makes sense to add more registers and describe a situation where it doesn't make sense to add more registers.

2. [20] Consider the fragment of MIPS assembly below:

```
sd $s5, 12($s3)

Id $s5, 8($s3)

sub $s4, $s2, $s1

beqz $s4, label

add $s2, $s0, $s1

sub $s2, $s6, $s1
```

Suppose we modify the pipeline so that it has only one memory (that handles both instructions and data). In this case, there will be a structural hazard every time a program needs to fetch an instruction during the same cycle in which another instruction accesses data.

a. [5] Draw a pipeline diagram to show were the code above will stall.

b. [5] In general, is it possible to reduce the number of stalls/NOPs resulting from this structural hazard by reordering code?

c. [5] Must this structural hazard be handled in hardware? We have seen that data hazards can be eliminated by adding NOPs to the code. Can you do the same with this structural hazard? If so, explain how. If not, explain why not.

d. [5] Approximately how many stalls would you expect this structural hazard to generate in a typical program? (Use the instruction mix from Exercise 4.8.)

3. [20] Consider the following loop.

```
LOOP: ld $s0, 0($s3)
      ld $s1, 8($s3)
      add $s2, $s0, $s1
      addi $s3, $s3, -16
      bnez $s2, LOOP
```

Assume that perfect branch prediction is used (no stalls due to control hazards), that there are no delay slots, that the pipeline has full forwarding support, and that branches are resolved in the EX (as opposed to the ID) stage.

a. [10] Show a pipeline execution diagram for the first two iterations of this loop.

b. [10] Mark pipeline stages that do not perform useful work. How often while the pipeline is full do we have a cycle in which all five pipeline stages are doing useful work? (Begin with the cycle during which the addi is in the IF stage. End with the cycle during which the bnez is in the IF stage.)

4. [35] Problems in this exercise refer to the following sequence of instructions, and assume that it is executed on a five-stage pipelined datapath:

```
add $s3, $s1, $s0
lw $s2, 4($s3)
lw $s1, 0($s4)
or $s2, $s3, $s2
sw $s2, 0($s3)
```

a. [5] If there is no forwarding or hazard detection, insert NOPs to ensure correct execution.

b. [5] Now, change and/or rearrange the code to minimize the number of NOPs needed. You can assume register $t0 can be used to hold temporary values in your modified code.

c. [5] If the processor has forwarding, but we forgot to implement the hazard detection unit, what happens when the original code executes?

d. [10] If there is forwarding, for the first seven cycles during the execution of this code, specify which signals are asserted in each cycle by hazard detection and forwarding units in Figure 4.59.

e. [5] If there is no forwarding, what new input and output signals do we need for the hazard detection unit in Figure 4.59? Using this instruction sequence as an example, explain why each signal is needed.

f. [5] For the new hazard detection unit from 4-e(above question), specify which output signals it asserts in each of the first five cycles during the execution of this code.

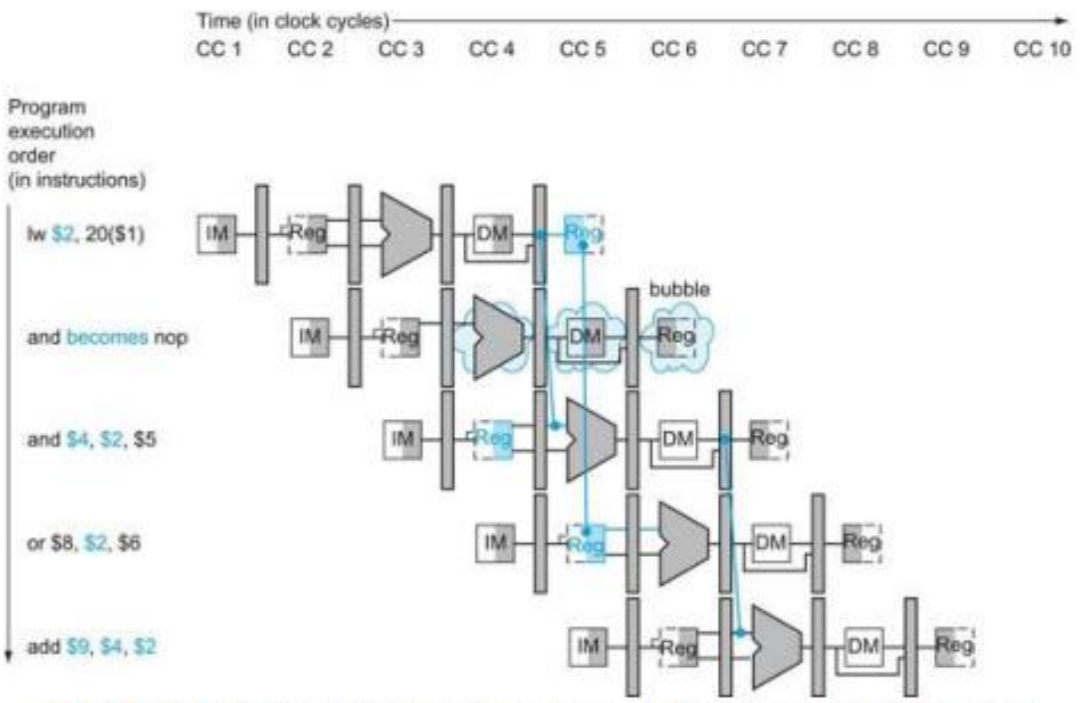

FIGURE 4.59 The way stalls are really inserted into the pipeline.

A bubble is inserted beginning in clock cycle 4, by changing the `and` instruction to a nop. Note that the `and` instruction is really fetched and decoded in clock cycles 2 and 3, but its EX stage is delayed until clock cycle 5 (versus the unstalled position in clock cycle 4). Likewise the `OR` instruction is fetched in clock cycle 3, but its ID stage is delayed until clock cycle 5 (versus the unstalled clock cycle 4 position). After insertion of the bubble, all the dependences go forward in time and no further hazards occur.