16.482 / 16.561: Computer Architecture and Design

Summer 2014

Lecture 3: Key Questions

May 27, 2014

1. What are the steps required to execute an instruction?

2. Generally speaking, what are processor datapath elements? What are control signals?

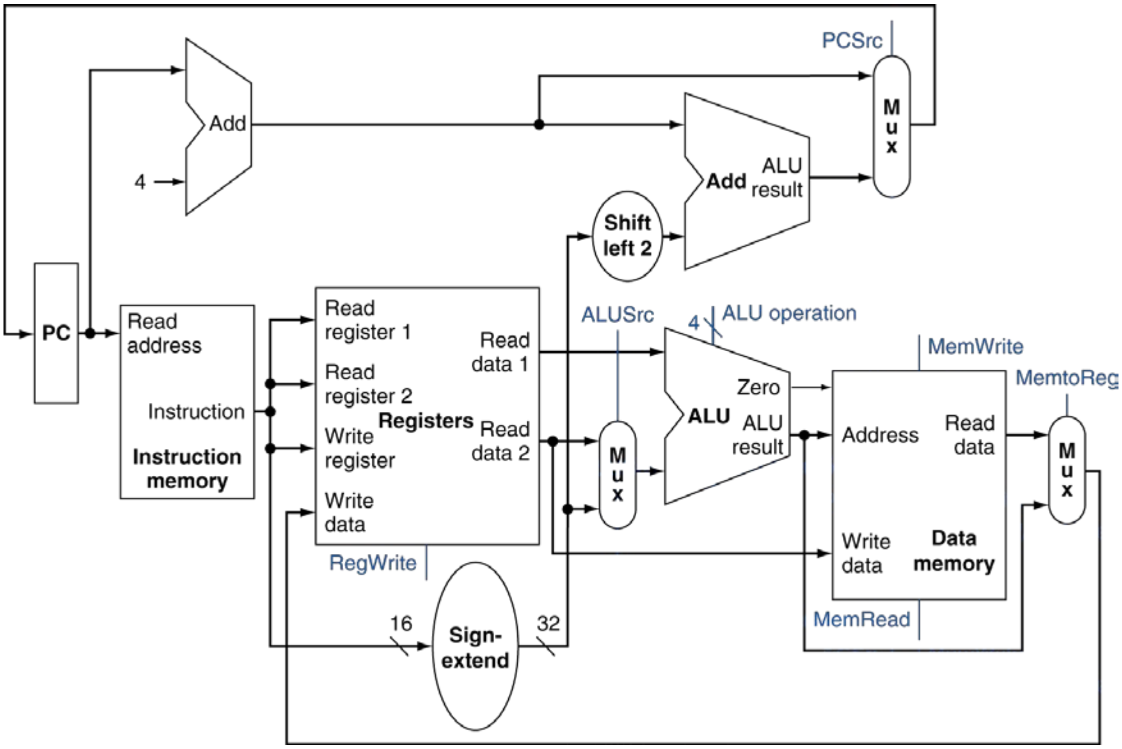
3. What steps do all instructions have in common?

4. What logic is required for instruction fetch?

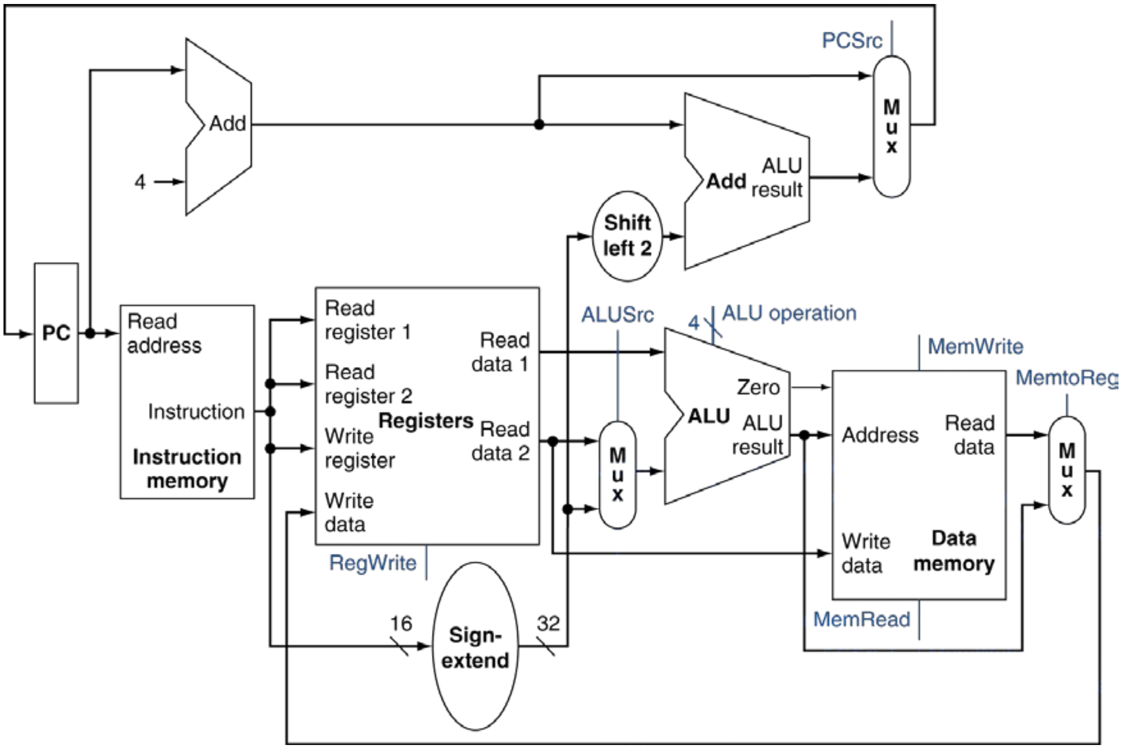
5. Describe generally how instruction decoding logic functions.

6. Show the overall datapath operation for:

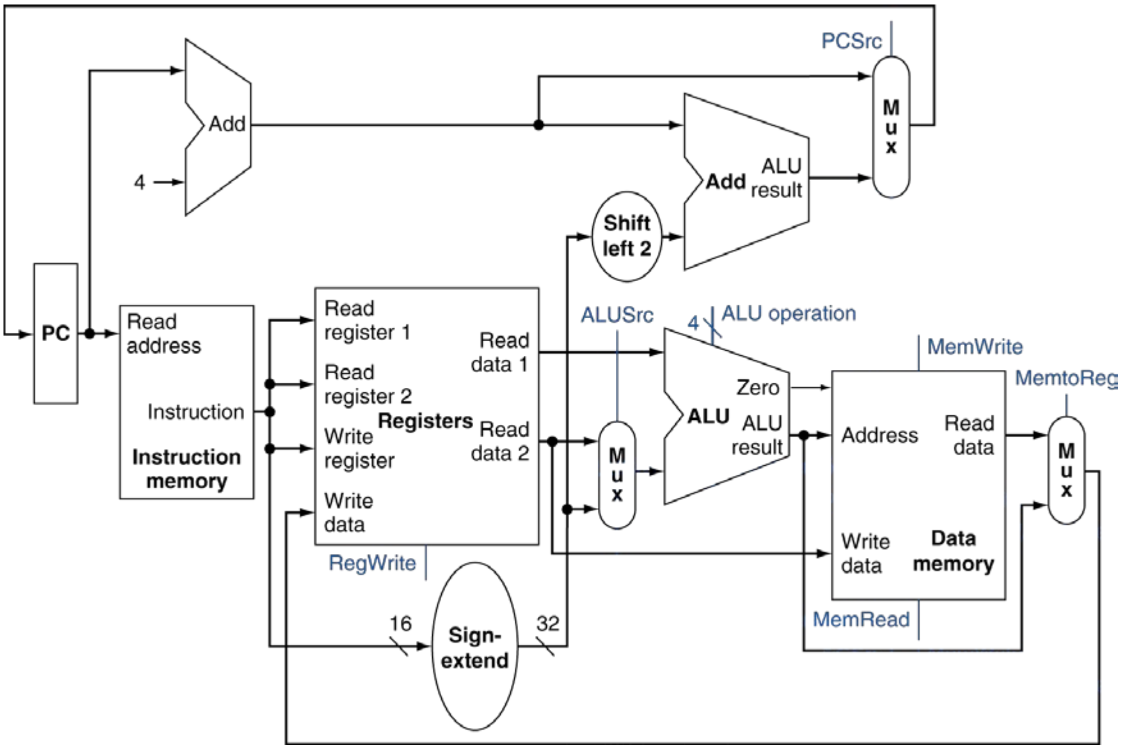
a. R-type instructions



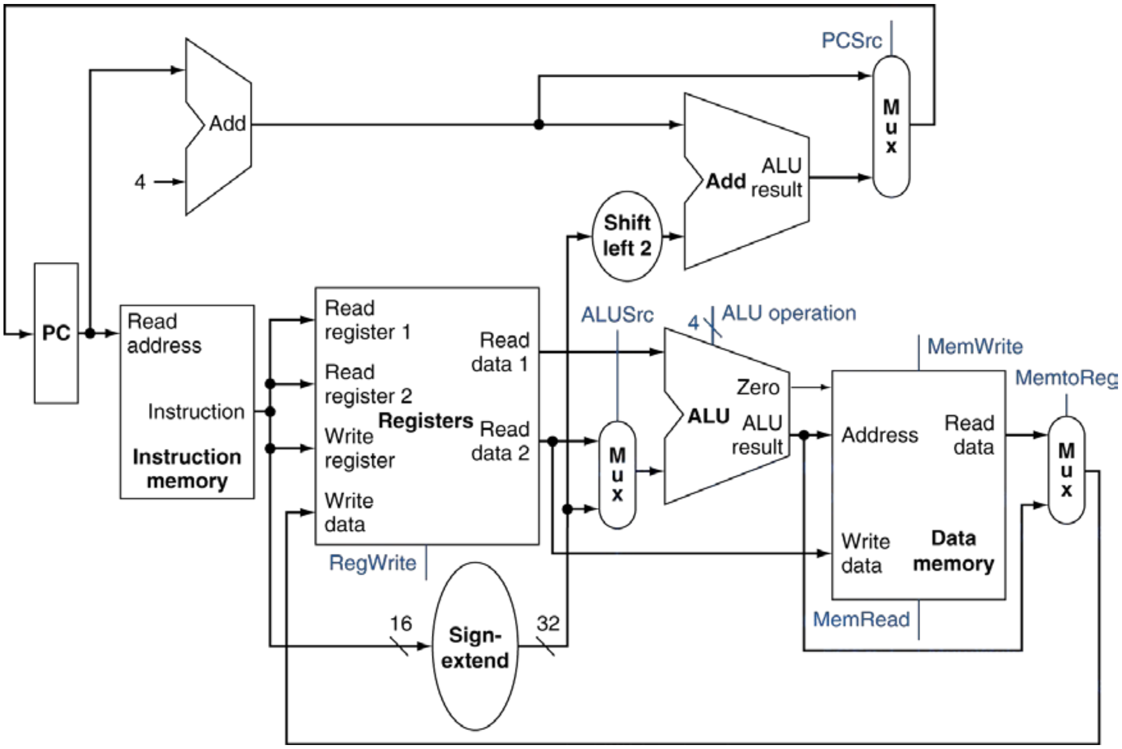
b. I-type ALU instructions



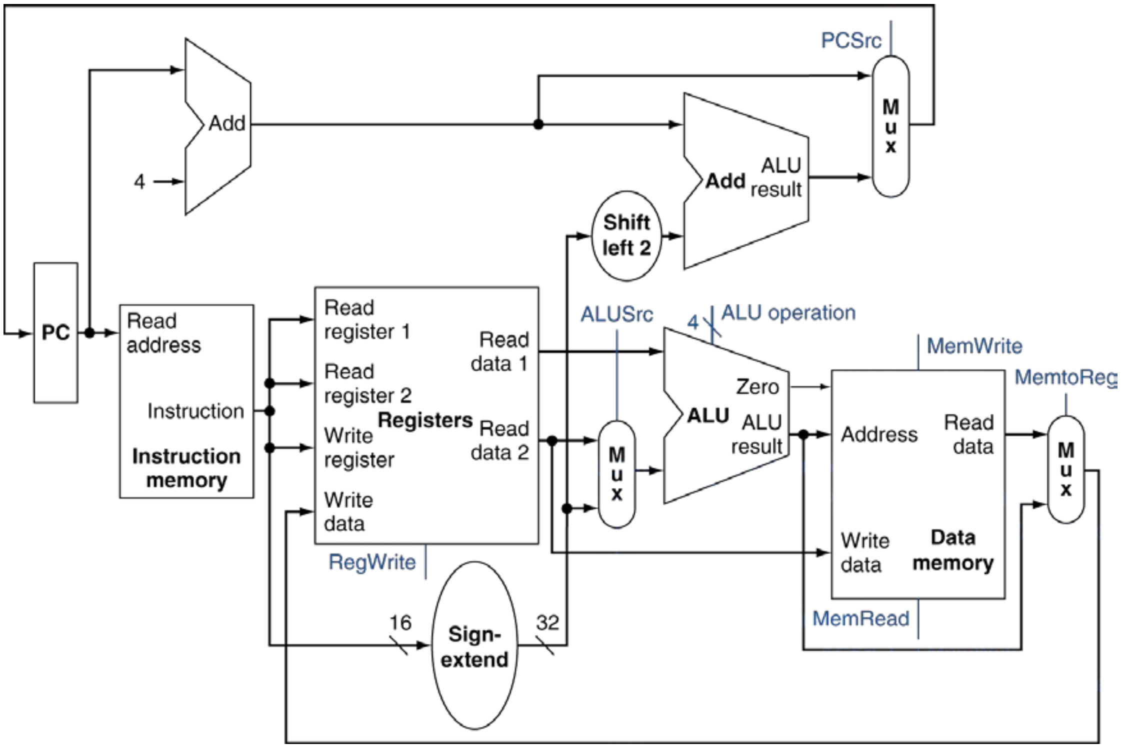
c. beq instructions



d. Loads



e. Stores



7. Describe the additional logic required to implement jump instructions.

8. Describe the basic operation of a pipelined datapath.

9. Does pipelining improve latency or throughput?

10. What is the maximum potential speedup of pipelining?

11. If one pipeline stage can run faster than the others, how does that affect the speedup?

12. Draw a basic pipeline diagram and describe the 5 stages.

13. Every pipelined instruction takes the same number of cycles. Why?

14. Example: Say we have the following code:

```
loop:        add $t1, $t2, $t3
             lw   $t4, 0($t1)
             beq $t4, $t3, end
             sw   $t3, 4($t1)
             add $t2, $t2, 8
             j    loop
end:         ...
```

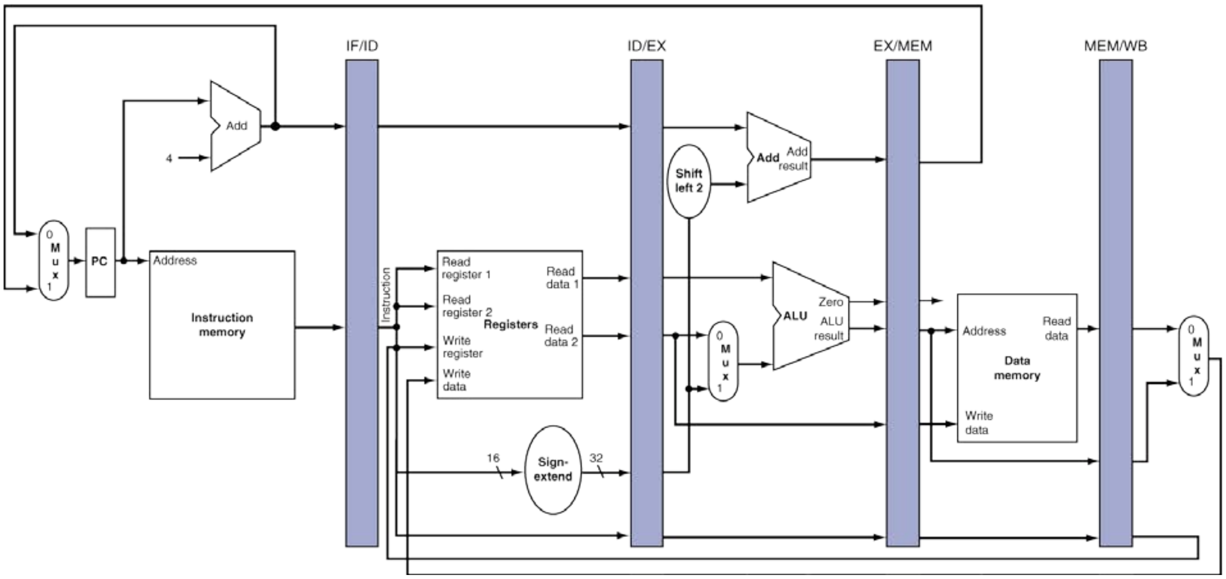
Assume each pipeline stage takes 4 ns.

a. How much time will one loop iteration take on an ideal pipelined datapath (i.e., no delays between instructions)?

b. How much time will one loop iteration take on a single-cycle datapath?

c. How much faster is the pipelined datapath than the single-cycle datapath? Express your answer as a ratio, not an absolute difference.

15. Describe how a pipelined datapath is divided into stages and how data is transferred between stages. (Refer to the figure below.)



16. Describe the three different types of pipeline hazards.

17. Example: What are no-ops? Given the following code, where are no-ops needed?

```
add $t2, $t3, $t4
sub $t5, $t1, $t2
or  $t6, $t2, $t7
slt $t8, $t9, $t5
```

18. Explain how forwarding works in a pipelined datapath.

19. Describe a case in which forwarding alone is not enough to resolve a data hazard.

20. Example: Let's revisit the following code:

```
loop:        add $t1, $t2, $t3
             lw   $t4, 0($t1)
             beq $t4, $t3, end
             sw   $t3, 4($t1)
             add $t2, $t2, 8
             j    loop
end:         ...
```

Again, assume each pipeline stage takes 4 ns.

a. How many cycles would one loop iteration take in a pipelined datapath without forwarding?

b. How many cycles would one loop iteration take in a pipelined datapath with forwarding?

21. Describe the reason processors may experience branch delays.