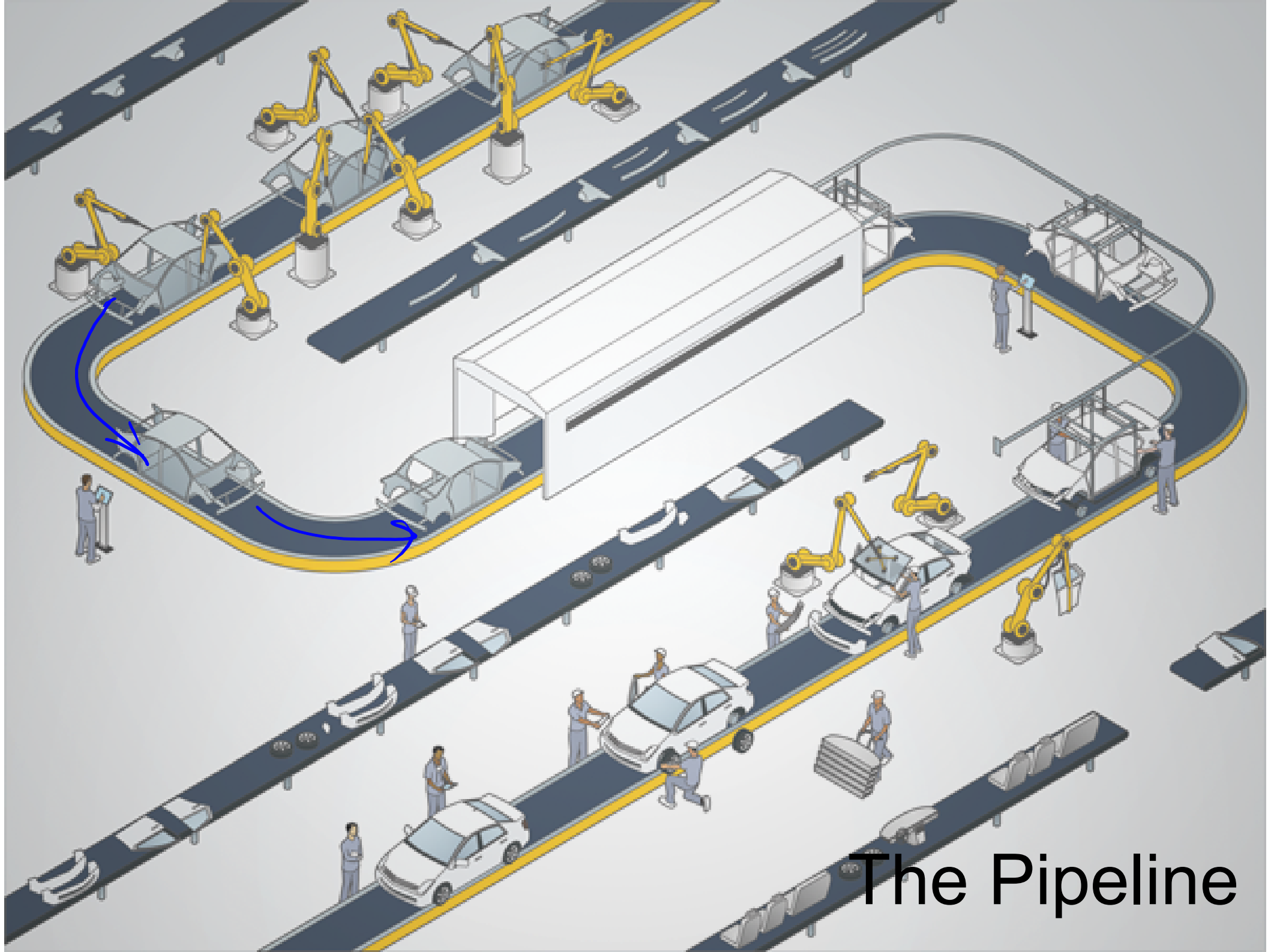
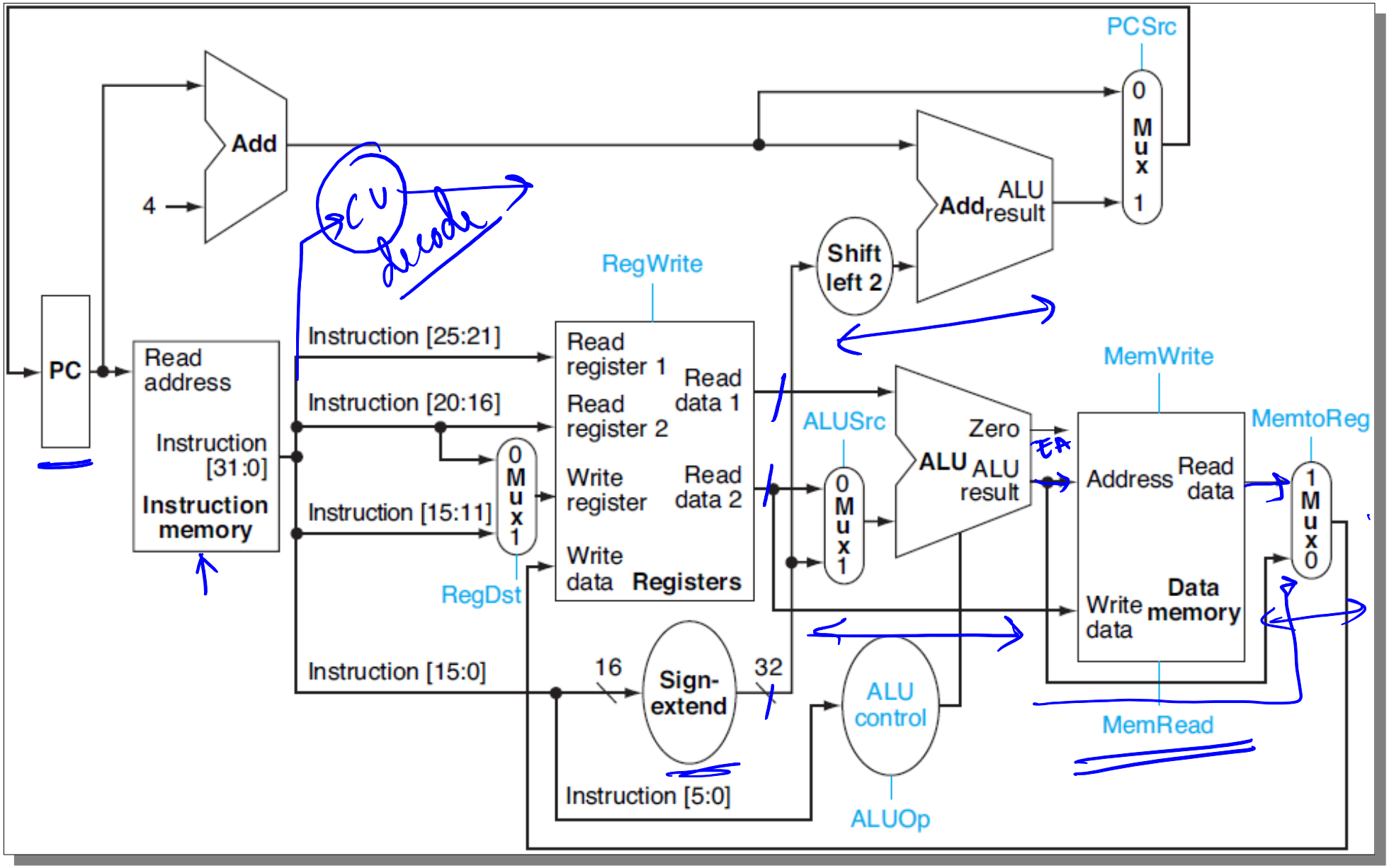

The Pipeline

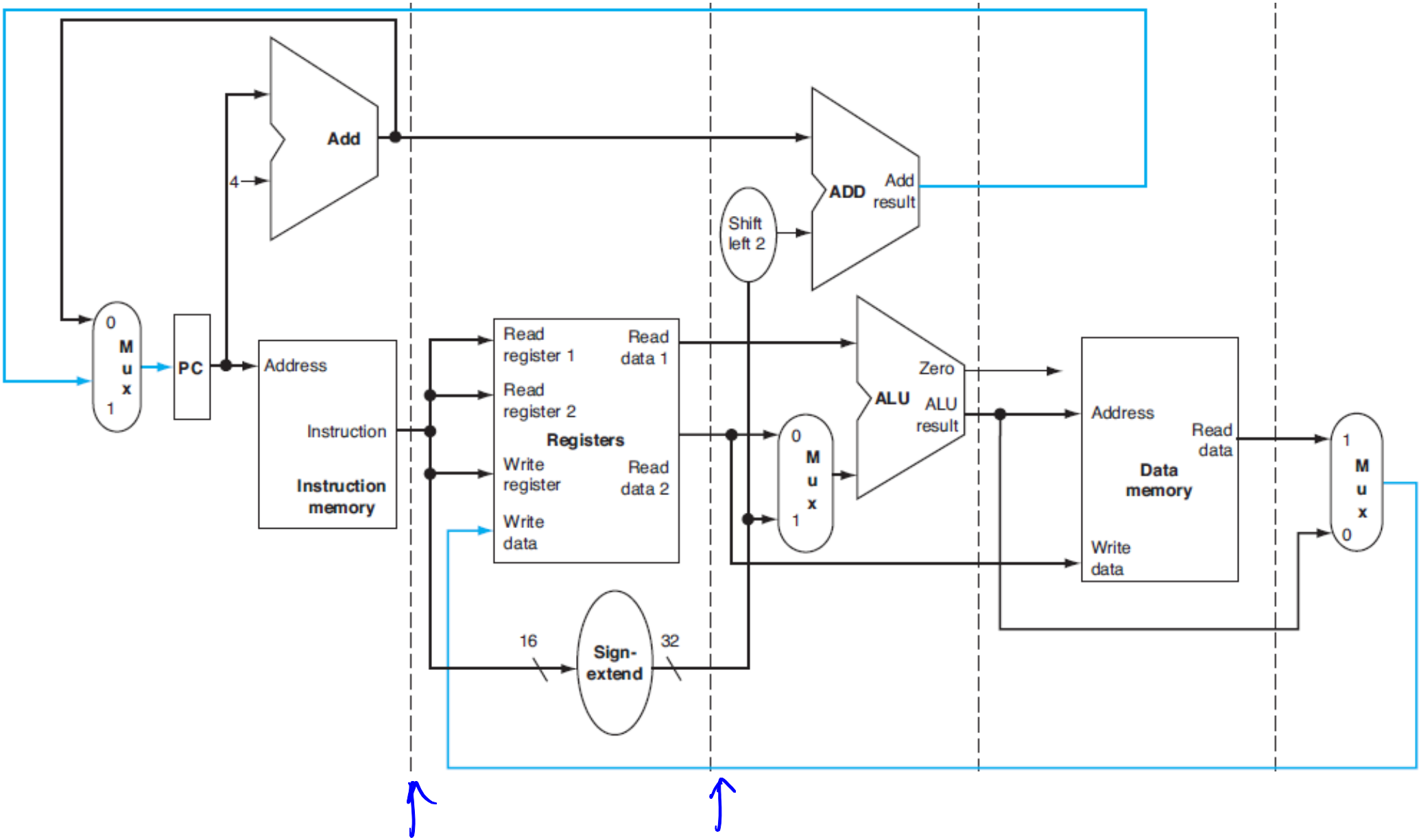
Module Outline

- Why Pipeline?
  - How to pipeline?
- Speedup of the pipeline
- Pipelined datapath
  - Execution of instructions
  - Pipeline Timing diagram
- Dependences, Hazards
  - Structural, Data, Control
  - Stalling, Forwarding
- Branch prediction

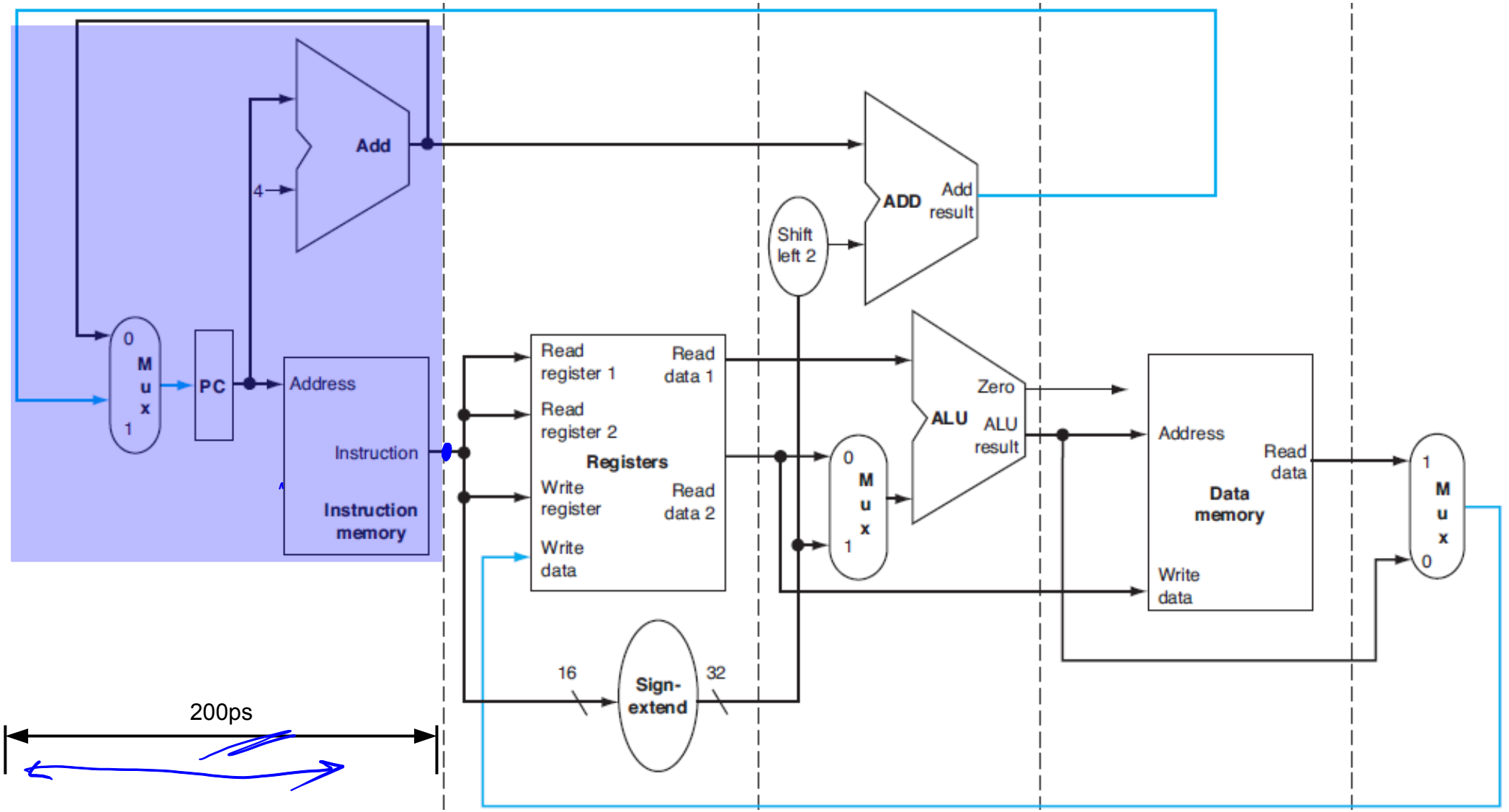
The Processor Datapath



Execution and Timing – Loads

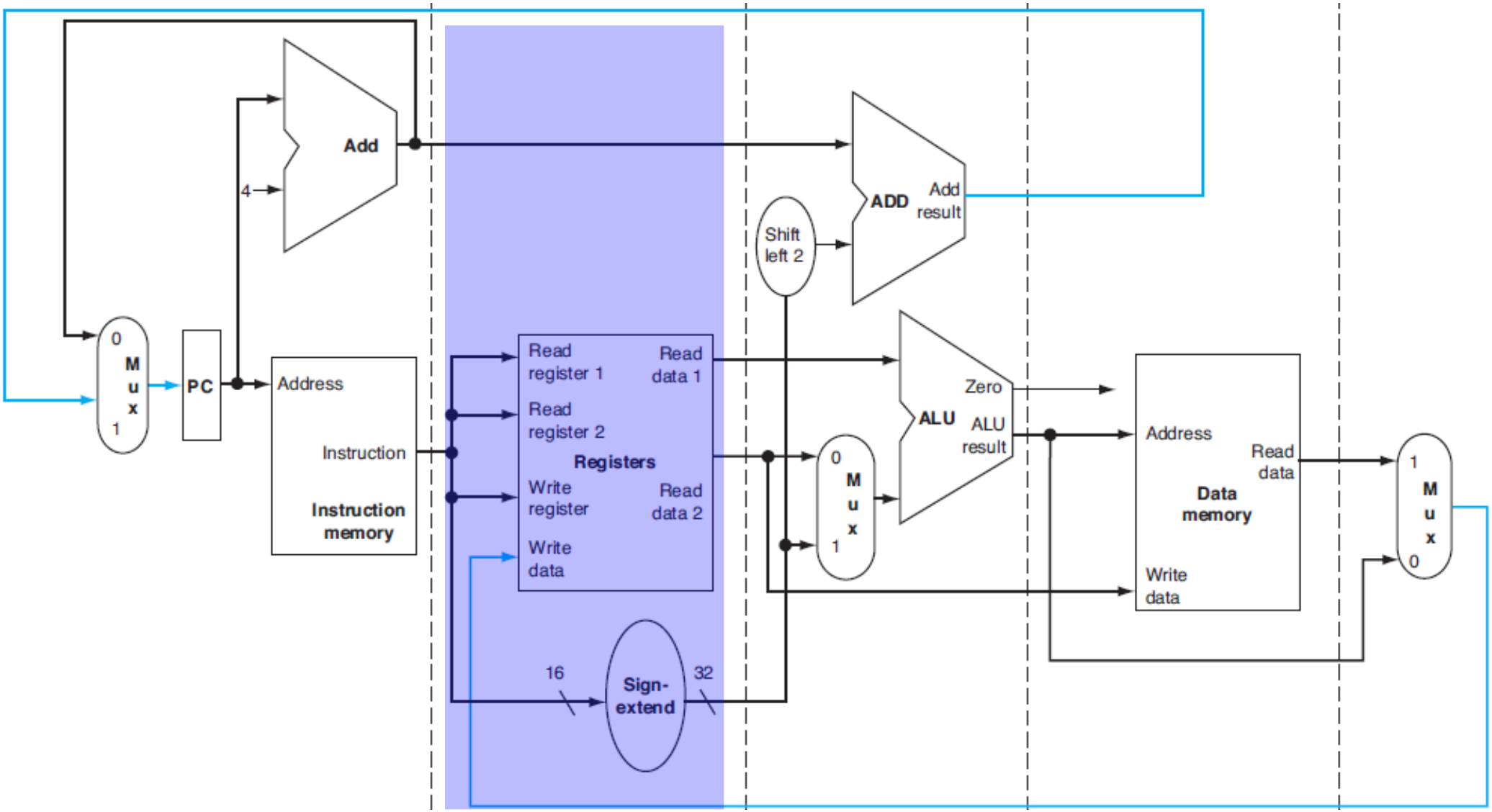


Execution and Timing – Loads



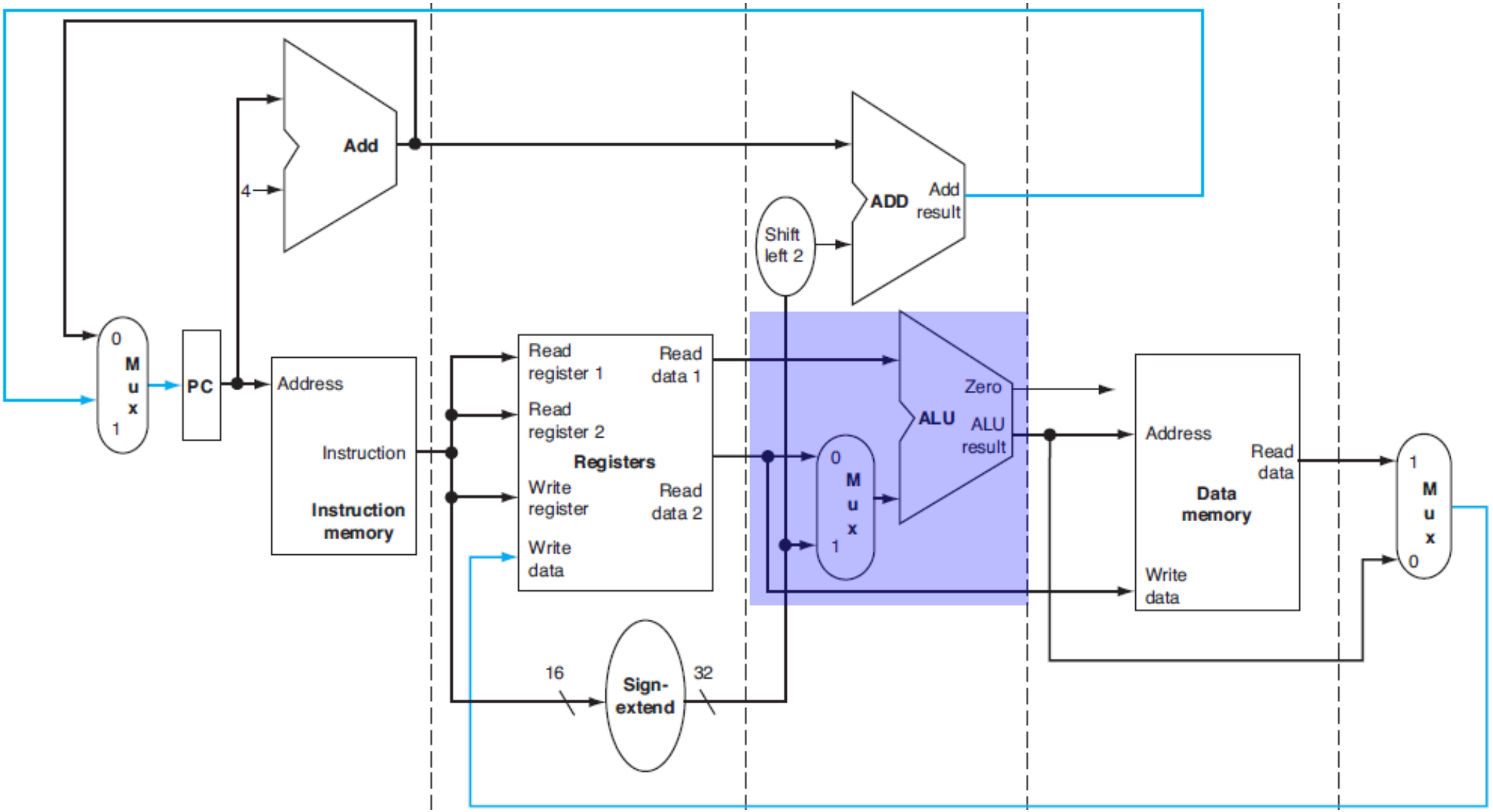
Instruction Fetch (IF)

Execution and Timing – Loads



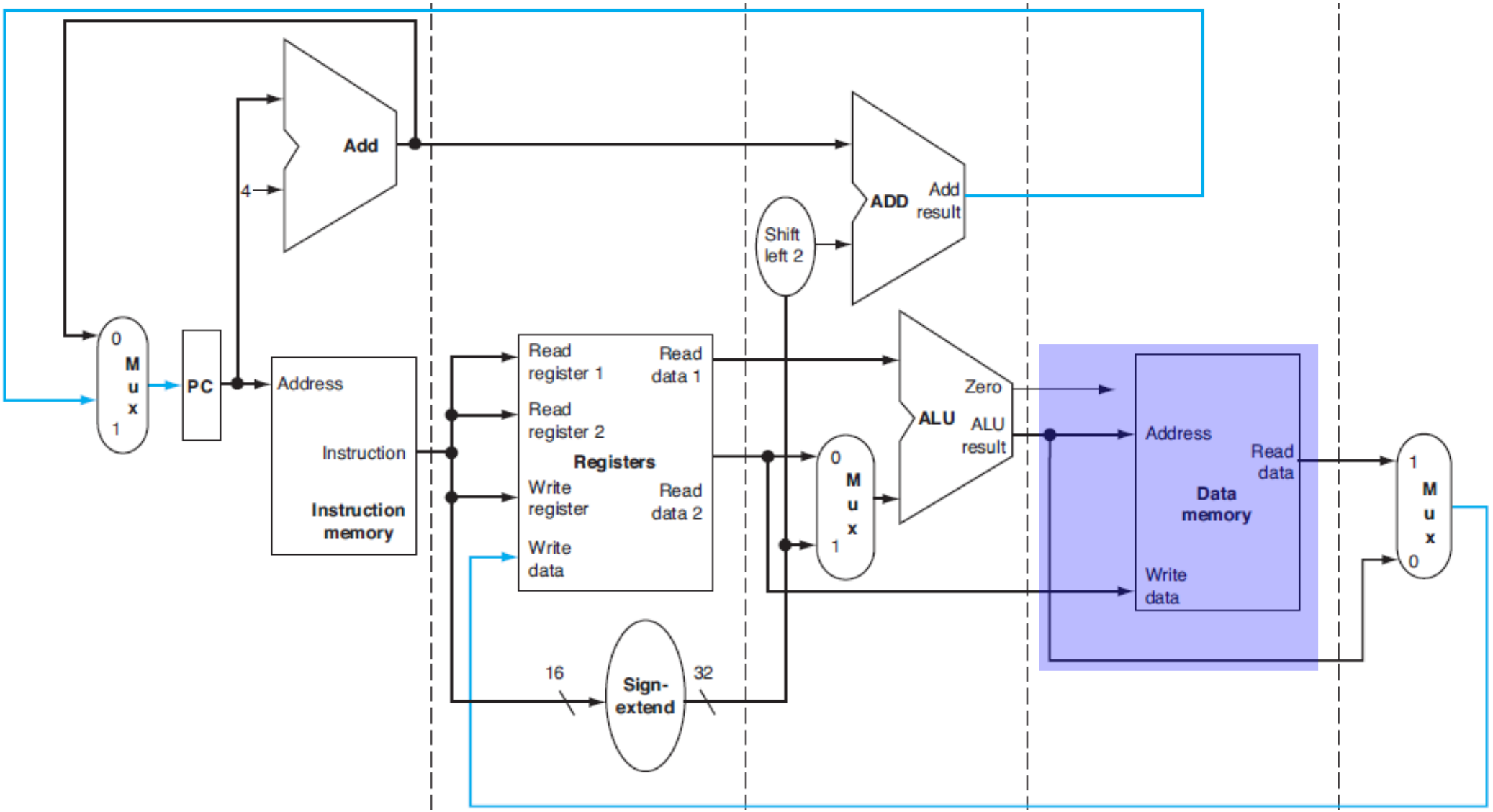
ID: Instruction decode/ Register file read

Execution and Timing – Loads



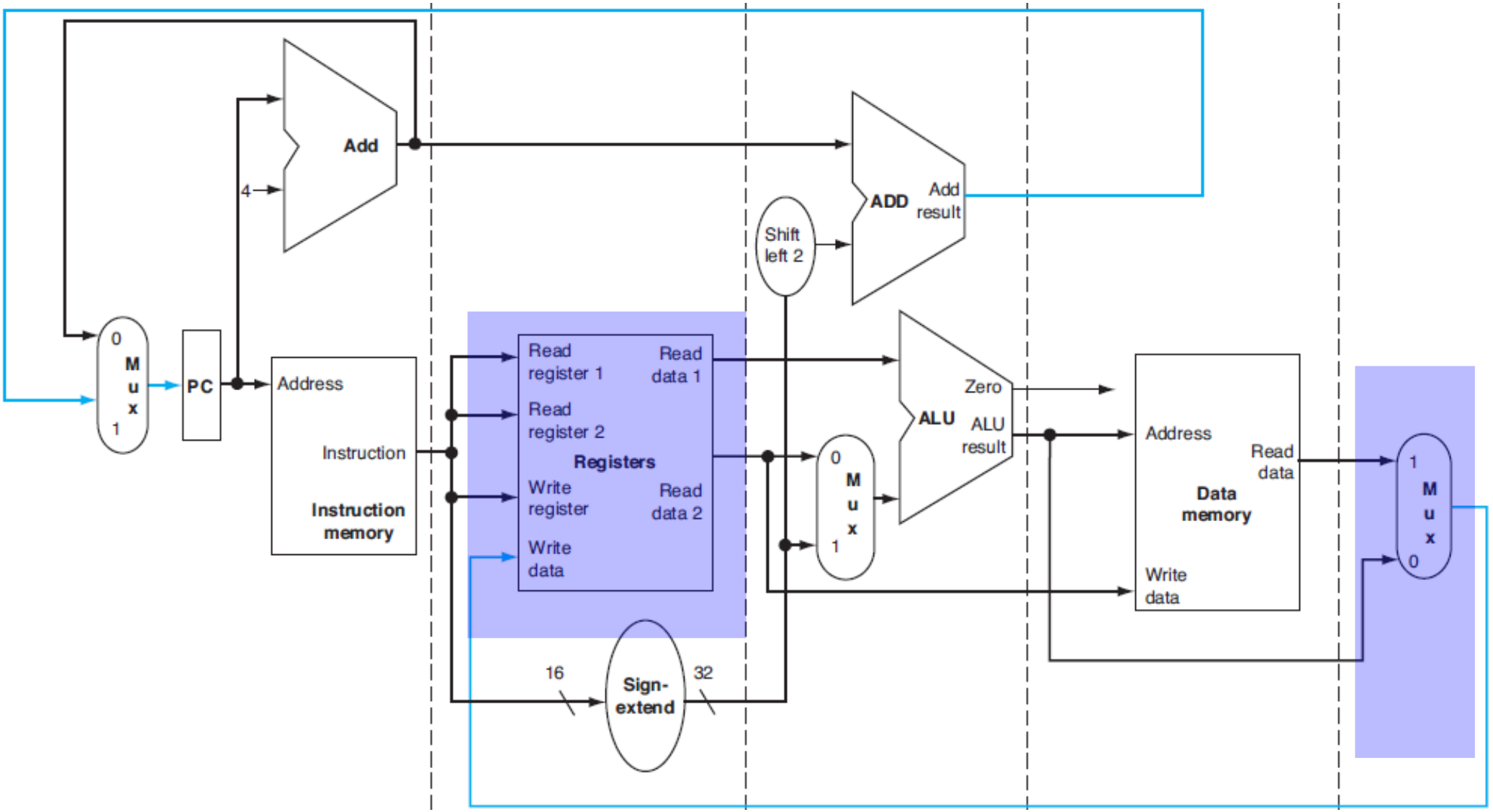
EX: Execution/ Address Calculation

Execution and Timing – Loads



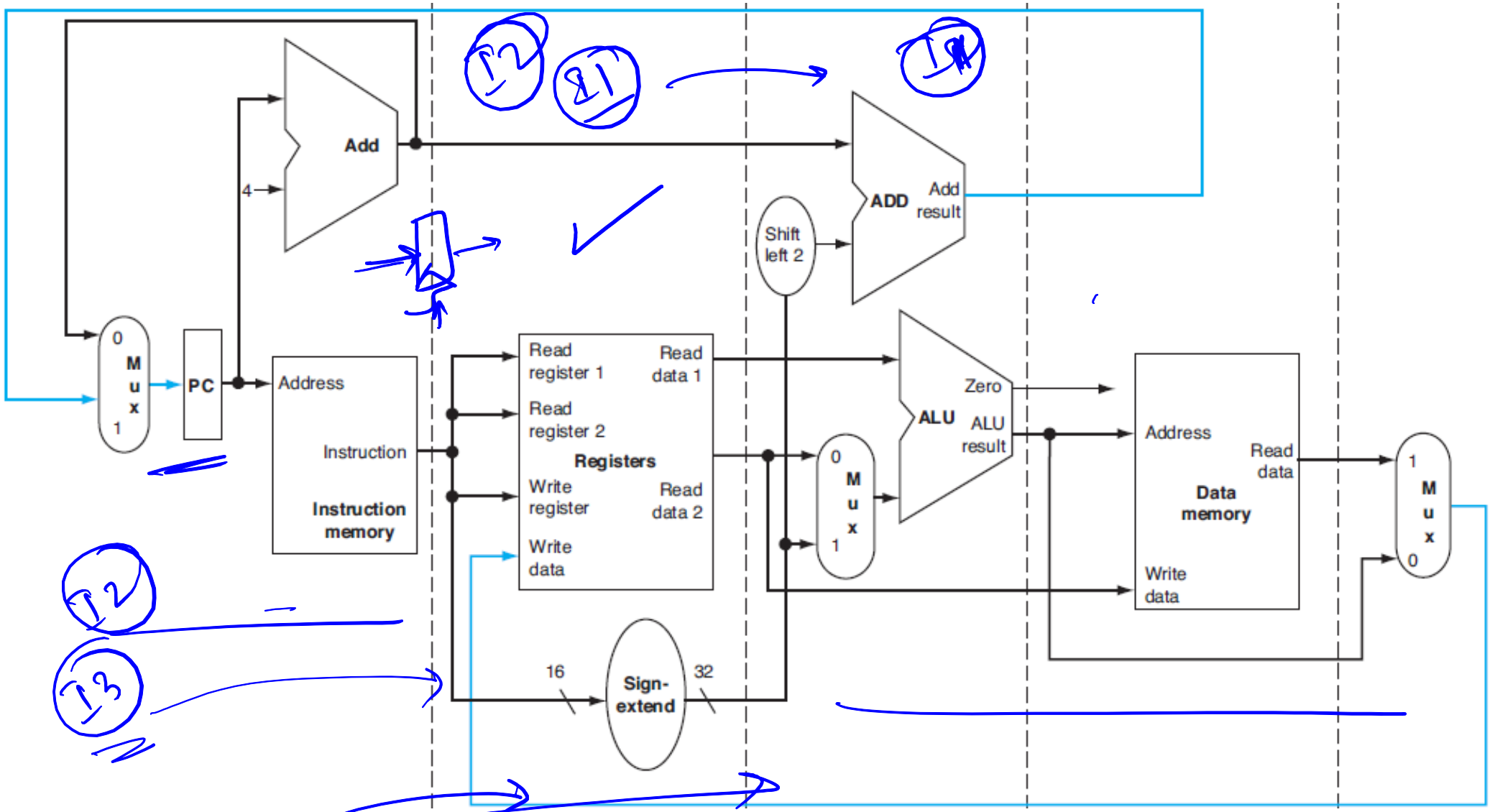
MEM: Memory Access

Execution and Timing – Loads



WB: Write Back

Pipeline Stages



Instruction Fetch (IF) | ID: Instruction decode/ Register file read | EX: Execution/ Address Calculation | MEM: Memory Access | WB: Write Back

Datapath – Observations

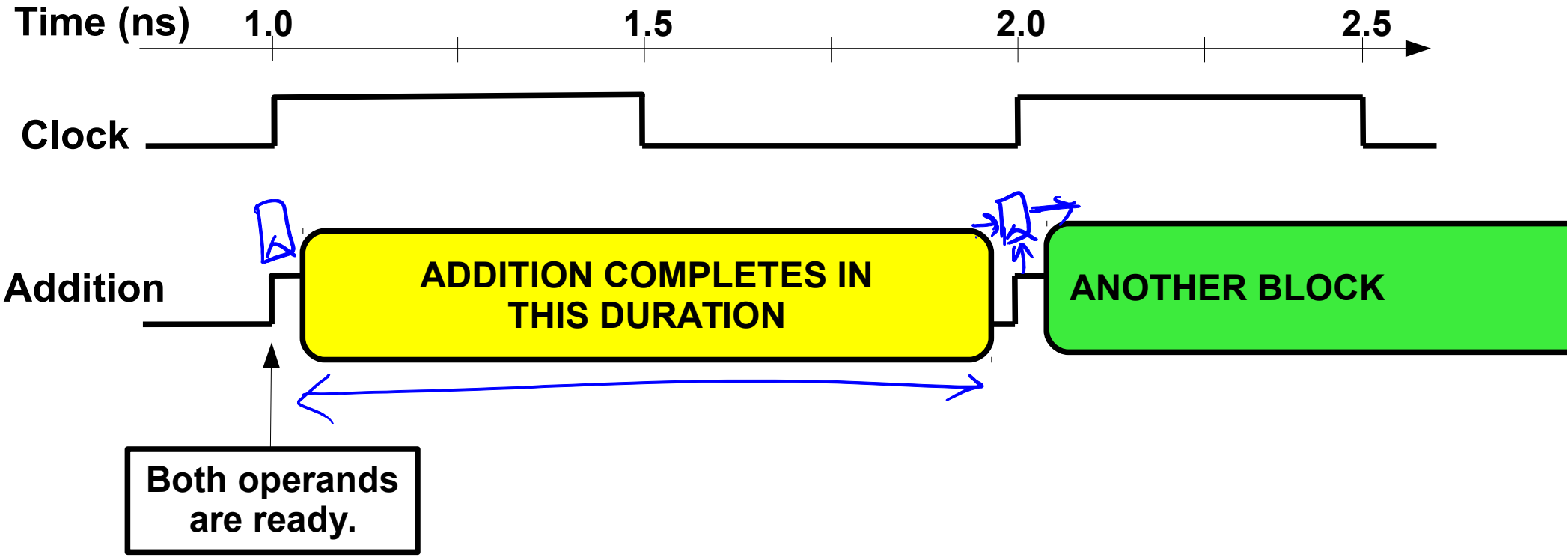
- During IF, other hardware are not utilized
  - During one stage, other stages lie idle

Datapath – Observations

- During IF, other hardware are not utilized
  - During one stage, other stages lie idle
- Improve hardware utilization
  - The entire datapath should be busy

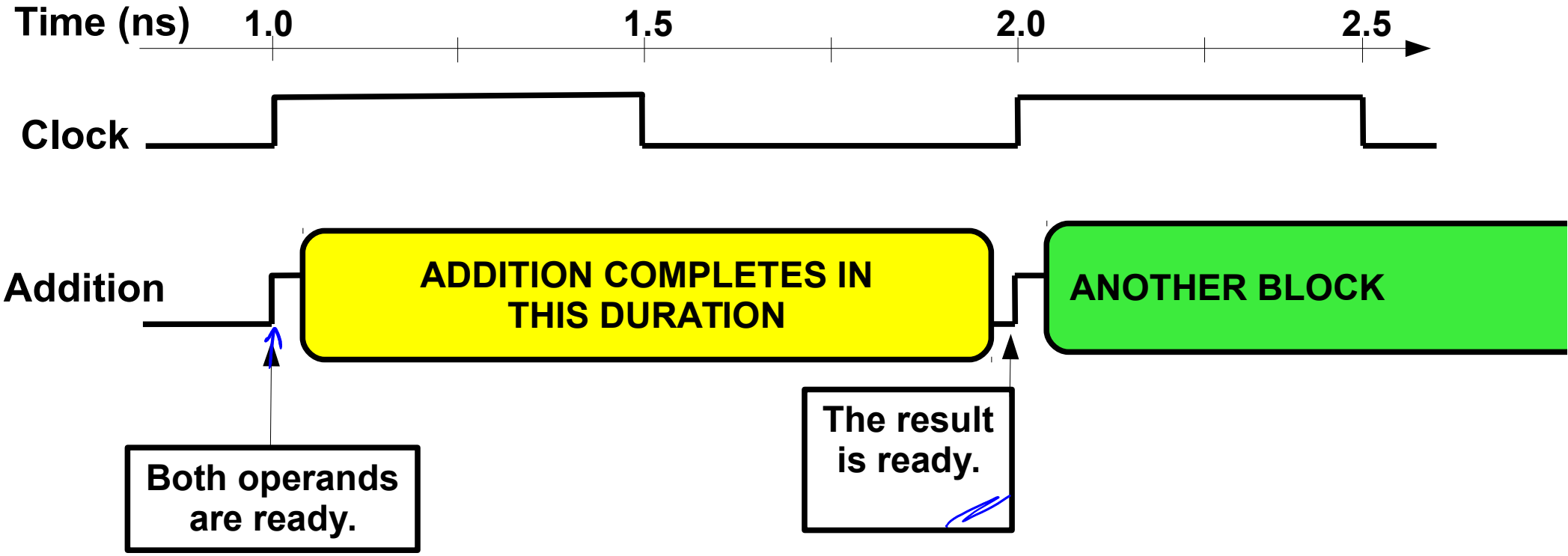
Clock Cycle

- Clock is a special signal to hardware
- A well defined indication for event start and complete.



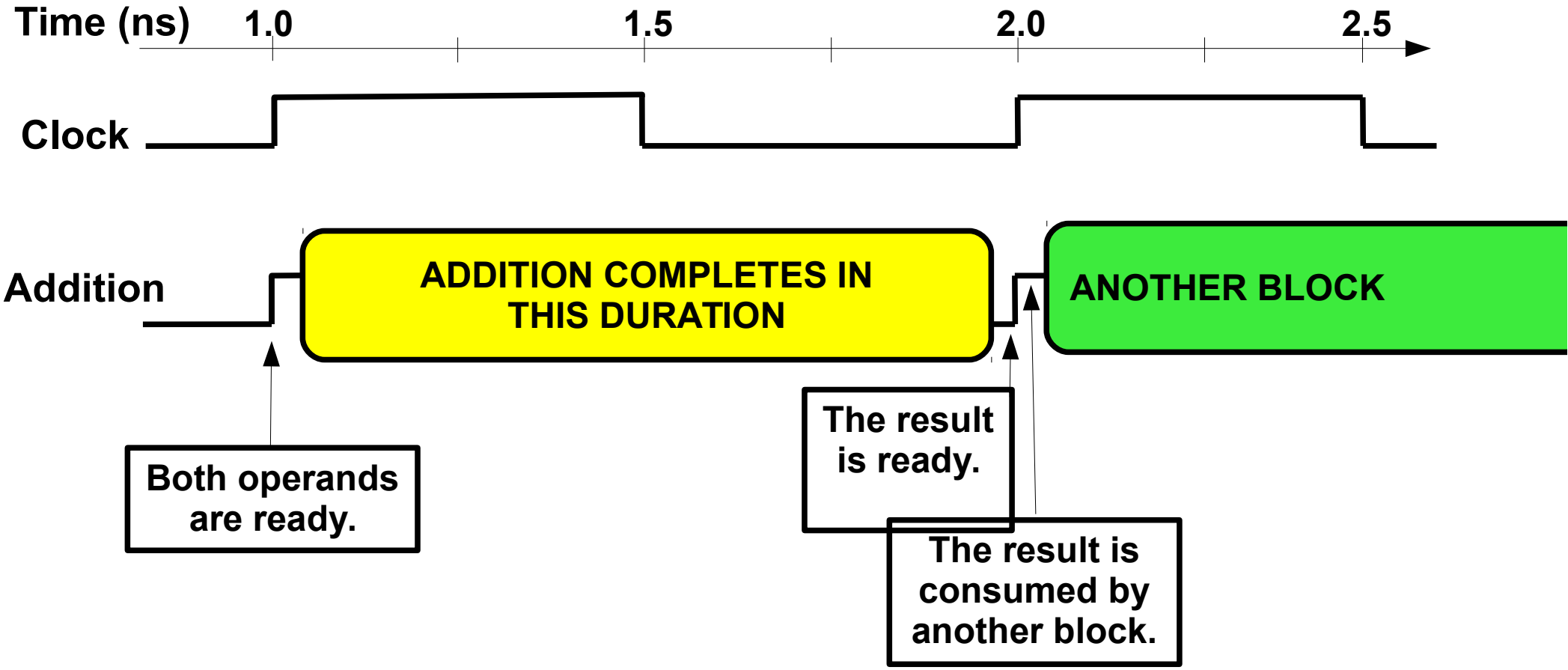
Clock Cycle

- Clock is a special signal to hardware
- A well defined indication for event start and complete.



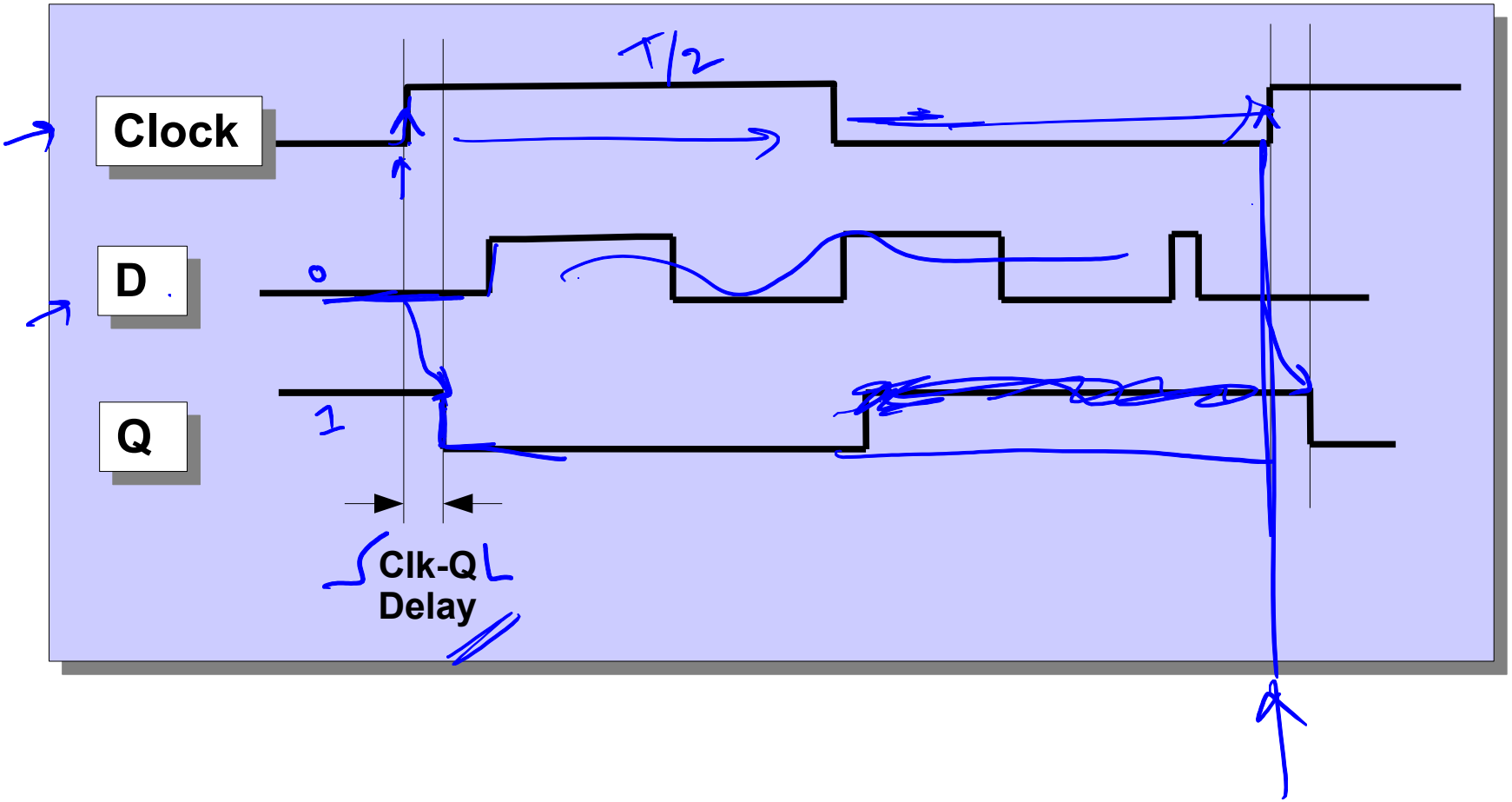
Clock Cycle

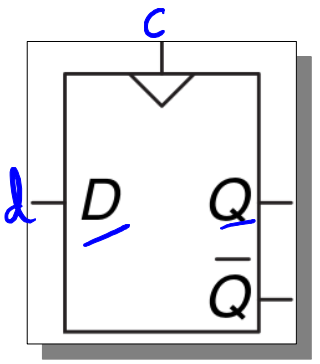
- Clock is a special signal to hardware
- A well defined indication for event start and complete.



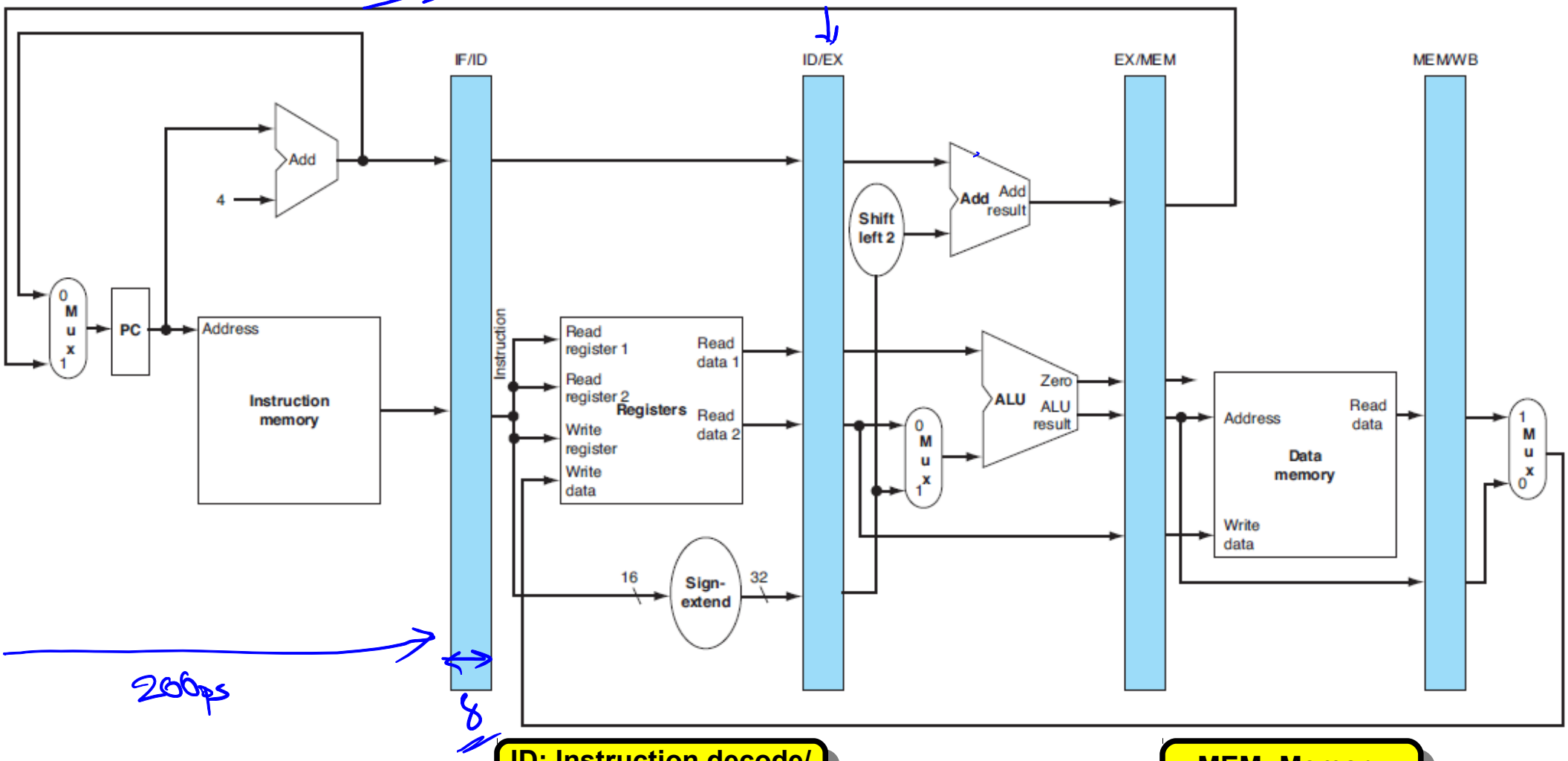
liquid crystal oscillators

Flip Flop Waveforms





Pipelined Datapath



Instruction Fetch (IF)

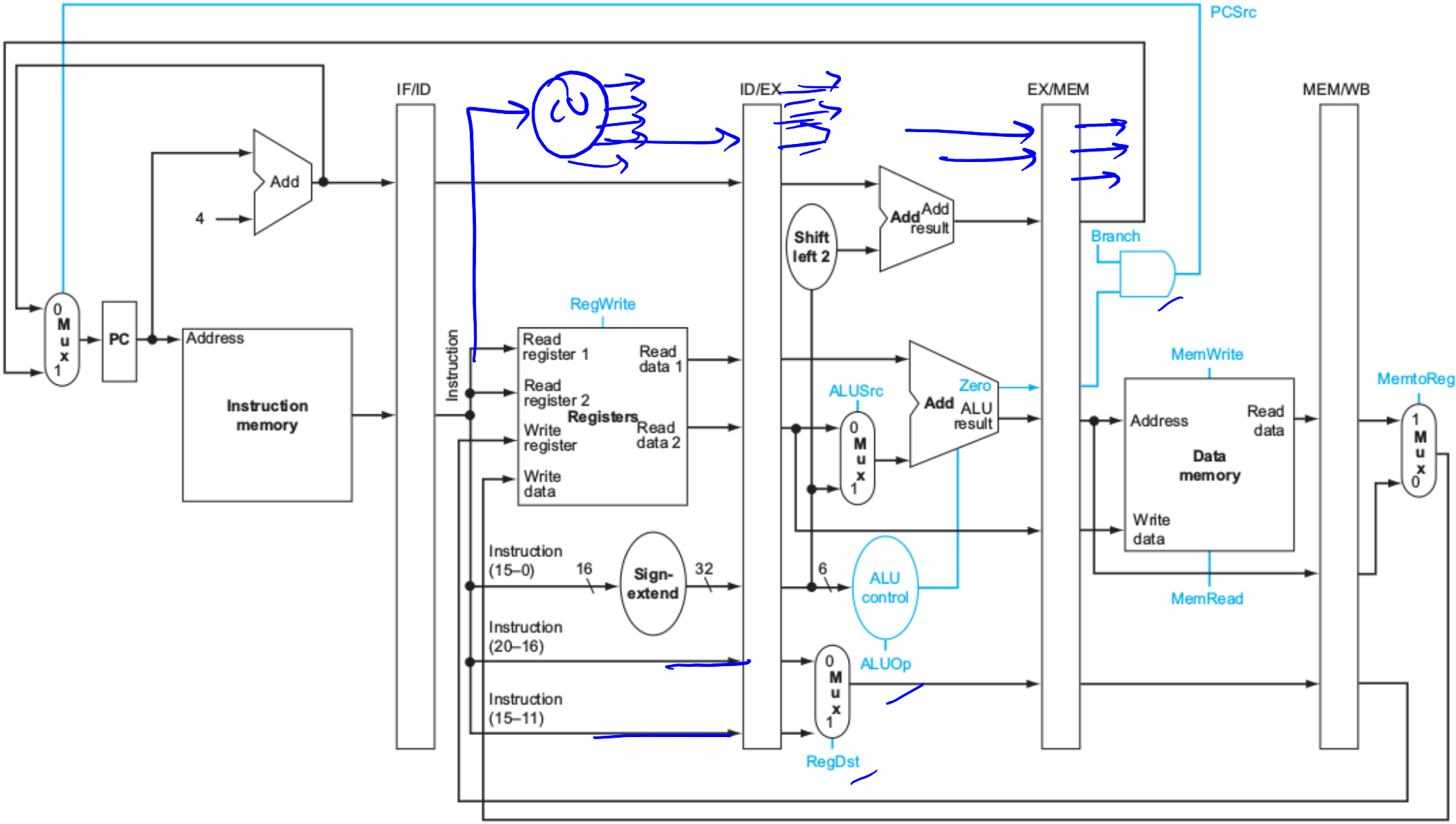
ID: Instruction decode/ Register file read

EX: Execution/ Address Calculation

MEM: Memory Access

WB: Write Back

Pipelined Datapath

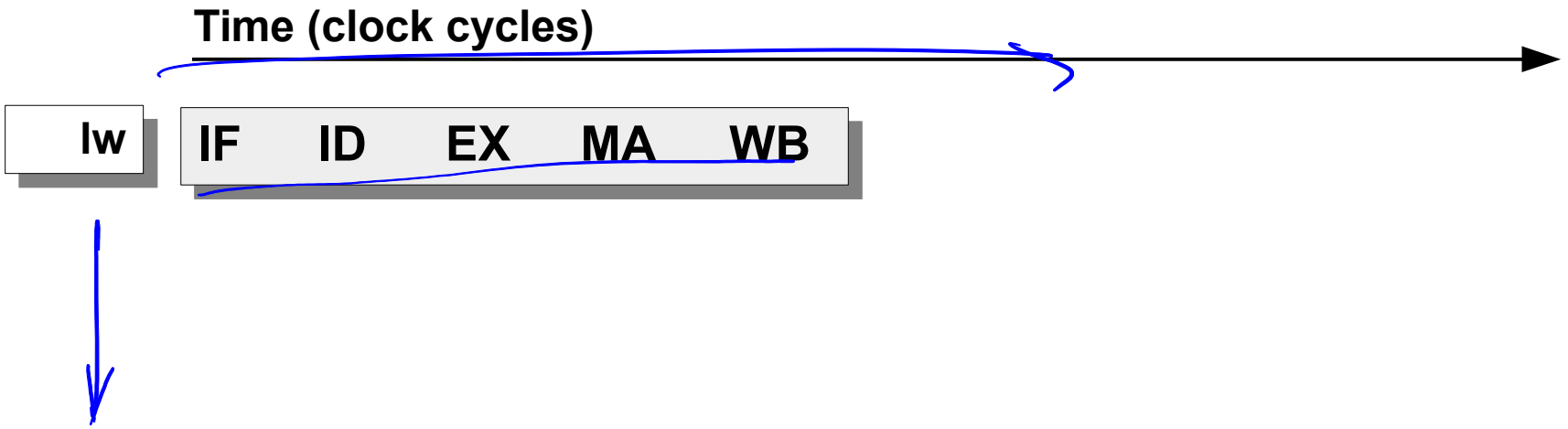


Execution Sequence

```
lw      $10, 20($1)
sub     $11, $2, $3
add     $12, $3, $4
lw      $13, 24($1)
add     $14, $5, $6
```

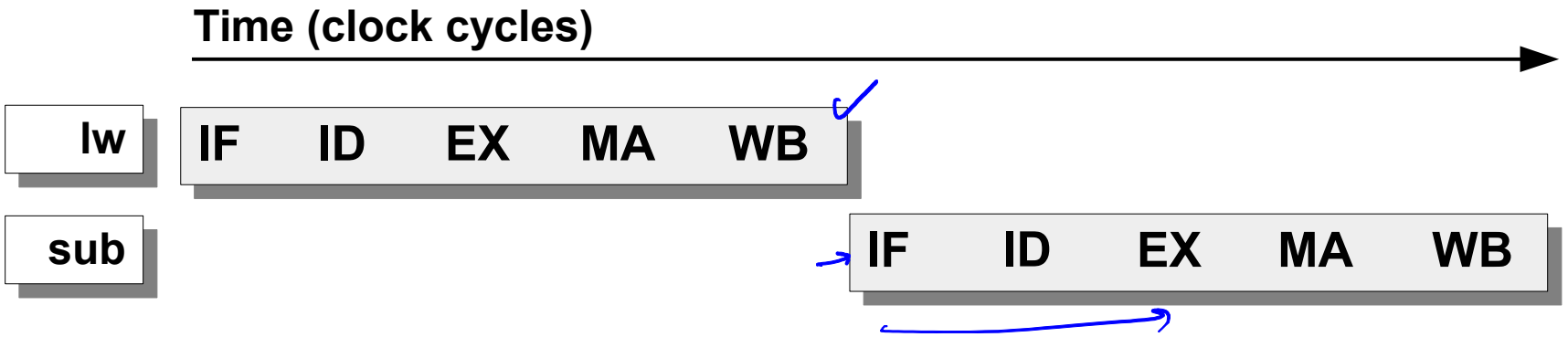
Execution Sequence – Non pipelined

```
lw      $10, 20($1)
sub     $11, $2, $3
add     $12, $3, $4
lw      $13, 24($1)
add     $14, $5, $6
```



Execution Sequence – Non pipelined

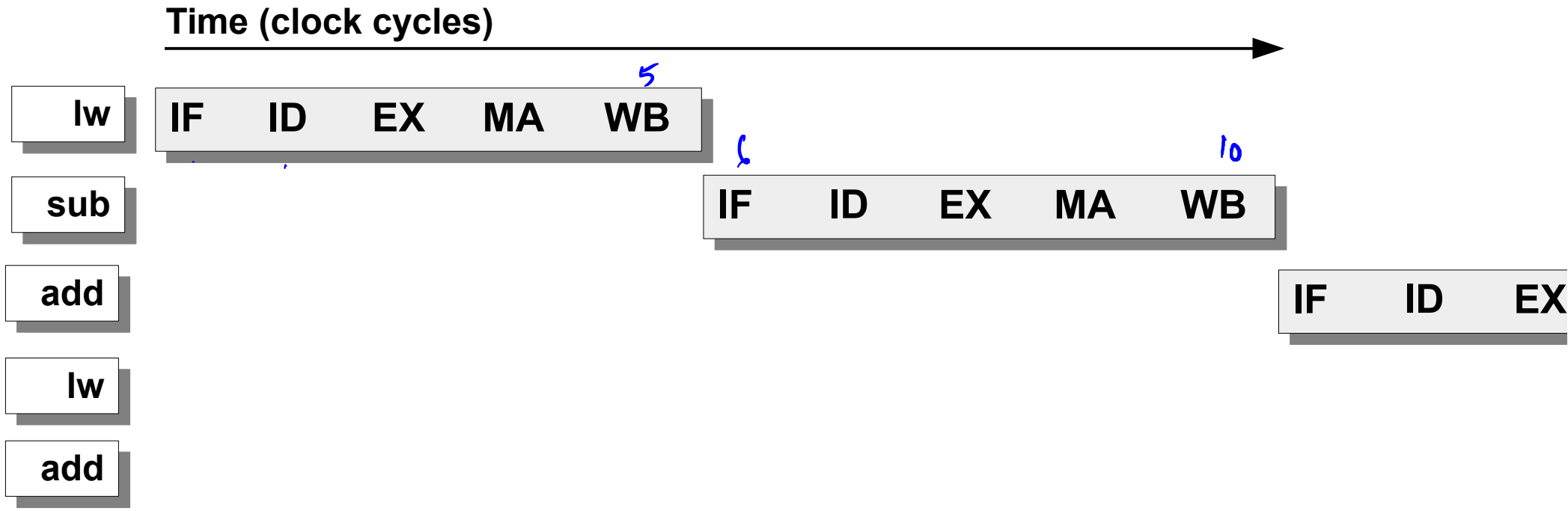
```
lw      $10, 20($1)
sub     $11, $2, $3
add     $12, $3, $4
lw      $13, 24($1)
add     $14, $5, $6
```



Execution Sequence – Non pipelined

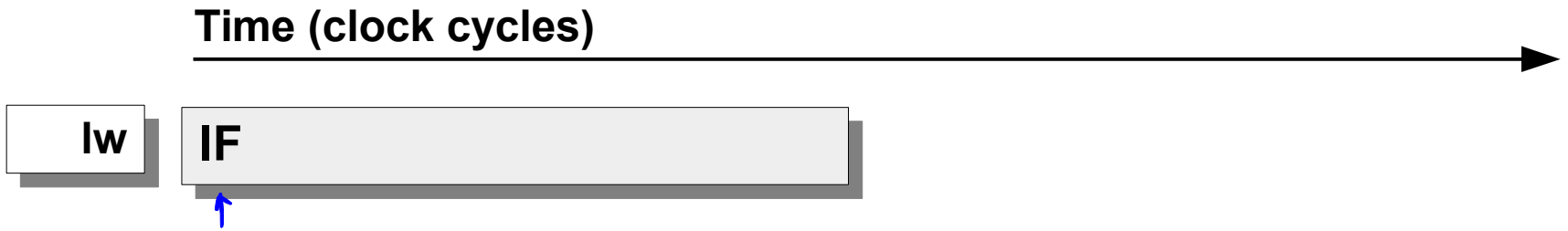
```
lw      $10, 20($1)
sub     $11, $2, $3
add     $12, $3, $4
lw      $13, 24($1)
add     $14, $5, $6
```

$T_{np} = 5 * IC$



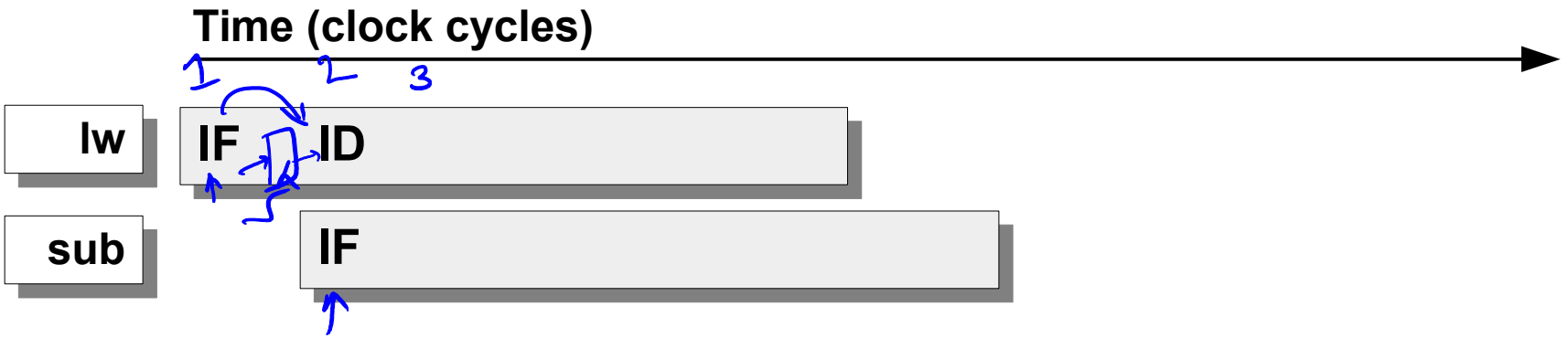
Execution Sequence – Pipelined

```
lw      $10, 20($1)
sub     $11, $2, $3
add     $12, $3, $4
lw      $13, 24($1)
add     $14, $5, $6
```



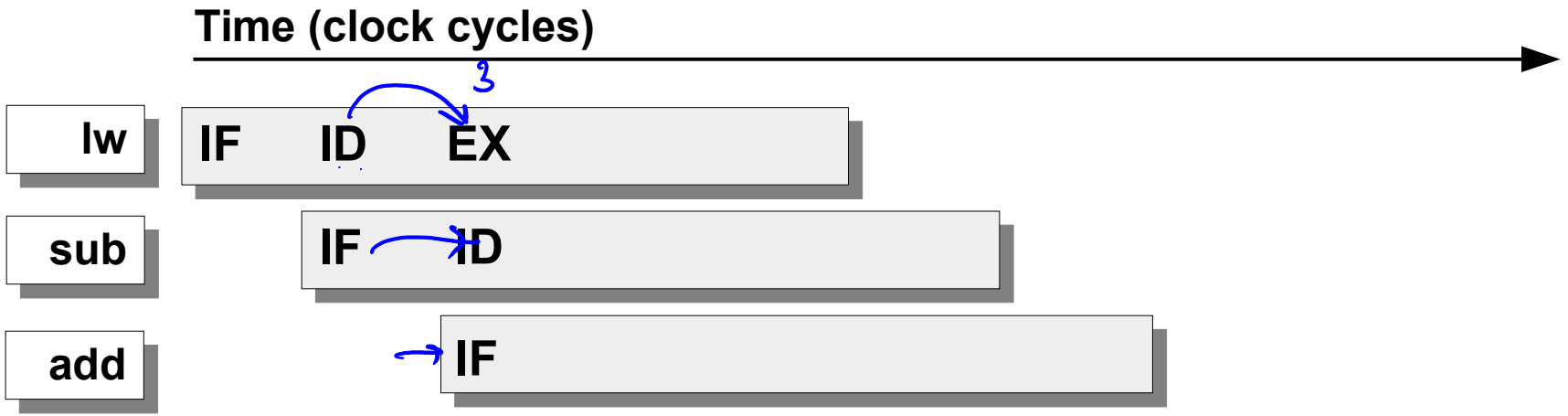
Execution Sequence – Pipelined

```
lw      $10, 20($1)
sub     $11, $2, $3
add     $12, $3, $4
lw      $13, 24($1)
add     $14, $5, $6
```



Execution Sequence – Pipelined

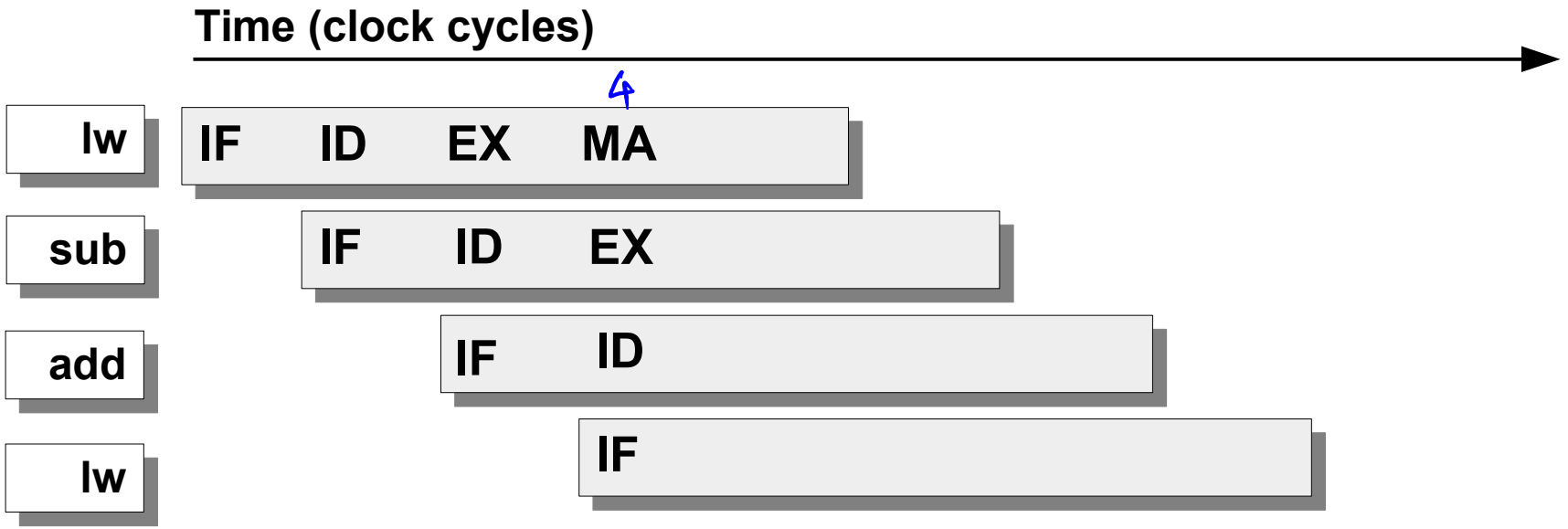
```
lw      $10, 20($1)
sub     $11, $2, $3
add     $12, $3, $4
lw      $13, 24($1)
add     $14, $5, $6
```



Execution Sequence – Pipelined

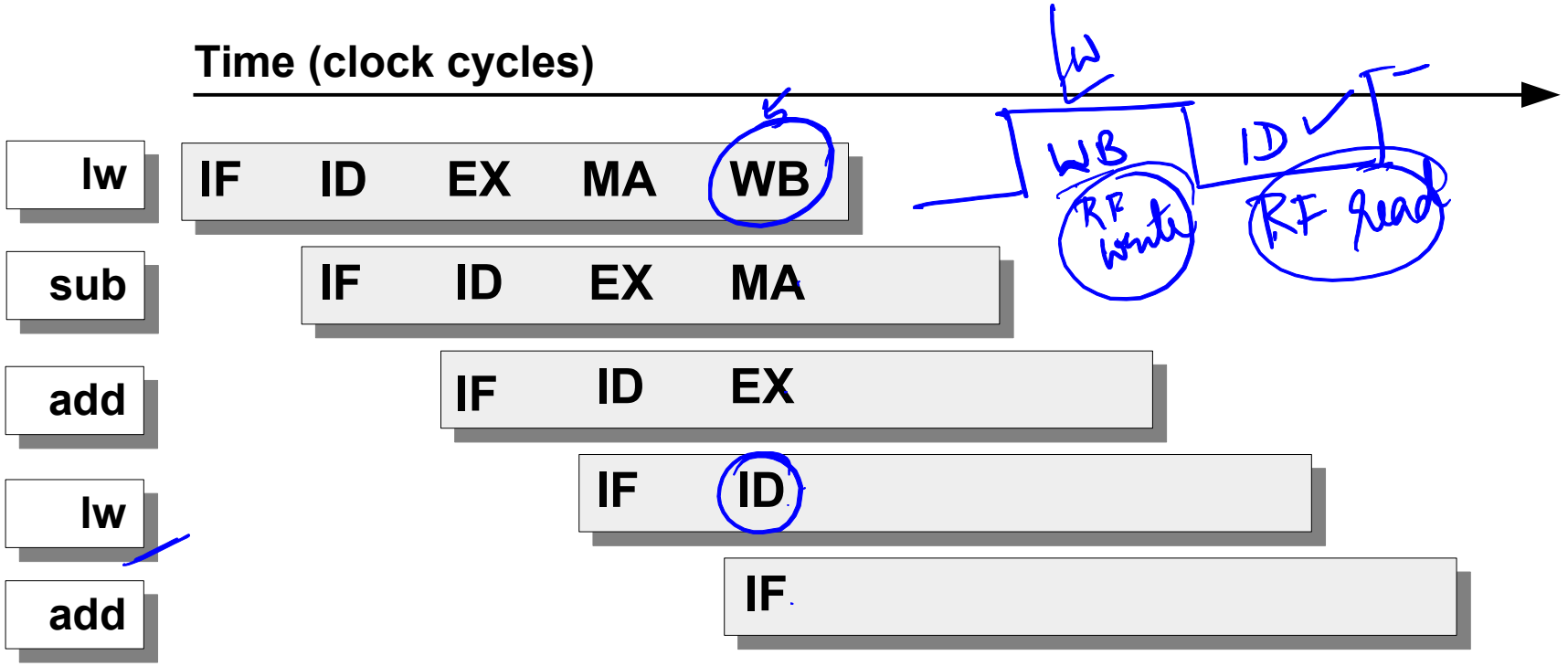
```
lw      $10, 20($1)
sub     $11, $2, $3
add     $12, $3, $4
lw      $13, 24($1)
add     $14, $5, $6
```

lockstep



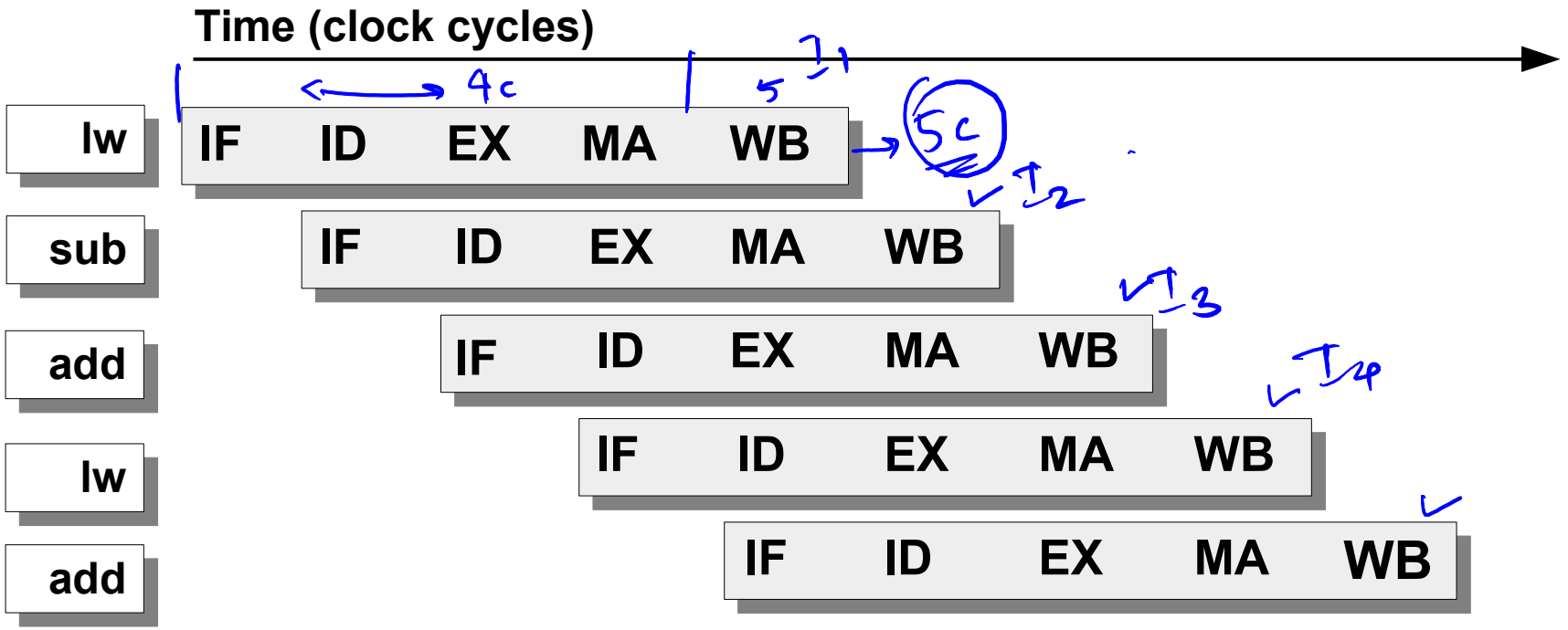
Execution Sequence – Pipelined

```
lw      $10, 20($1)
sub     $11, $2, $3
add     $12, $3, $4
lw      $13, 24($1)
add     $14, $5, $6
```

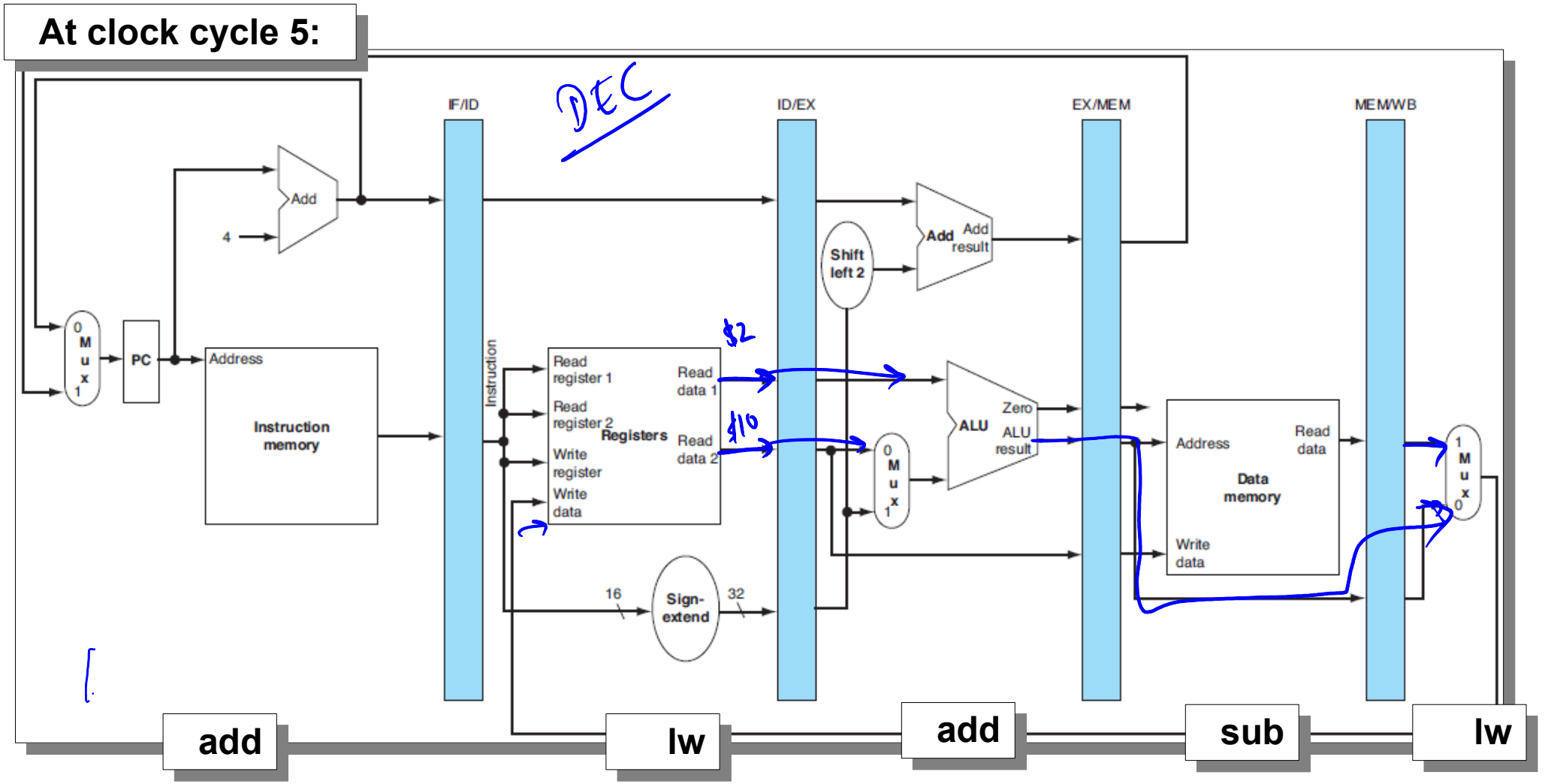


Execution Sequence – Pipelined

```
lw      $10, 20($1)
sub     $11, $2, $3
add     $12, $3, $4
lw      $13, 24($1)
add     $14, $5, $6
```



Execution Sequence – Pipelined



Pipelined vs. Nonpipelined Implementation

- Ratio of execution times between the two?
  - For 10^6 instructions?

Performance $= \frac{1}{T_{exec}}$

$$\text{Speedup} = \frac{T_{np}}{T_p} = \frac{5 \times 10^6}{(10^6 + 4)}$$

$$10^6 + (5-1)$$

$$= 5$$

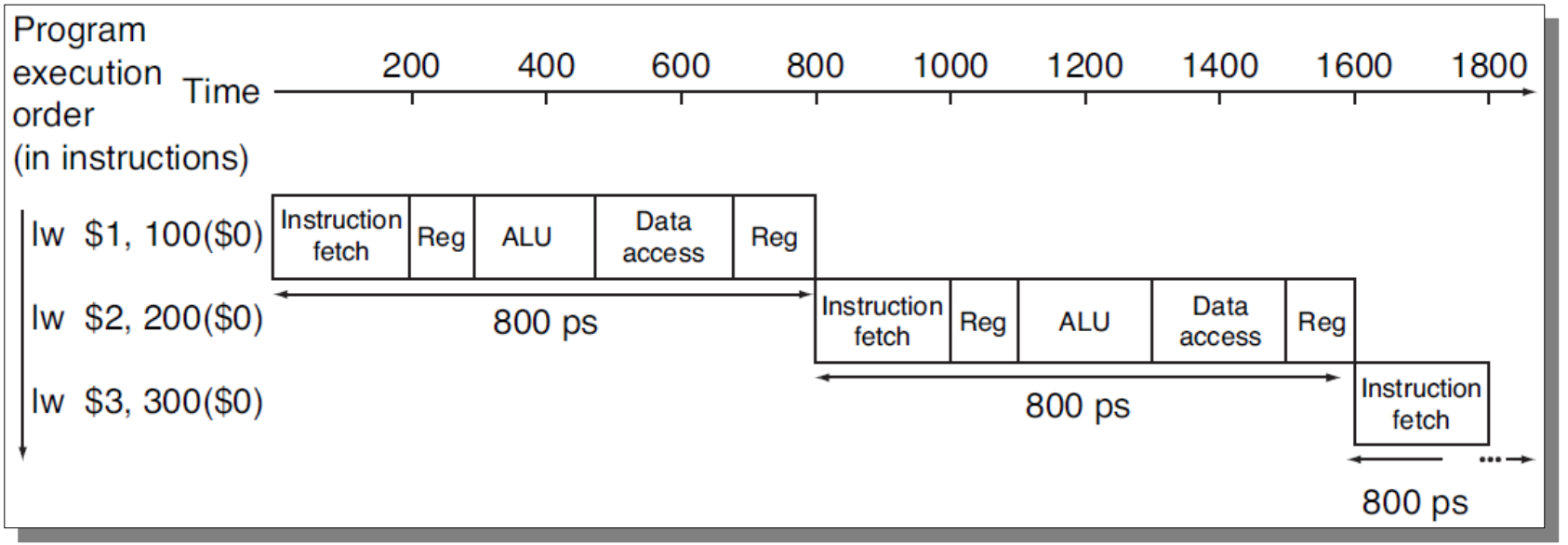
Pipelined vs. Nonpipelined Implementation

- Ratio of execution times between the two?
  - For 10^6 instructions?
- Pipelining increases the instruction throughput opposed to individual instruction execution time.

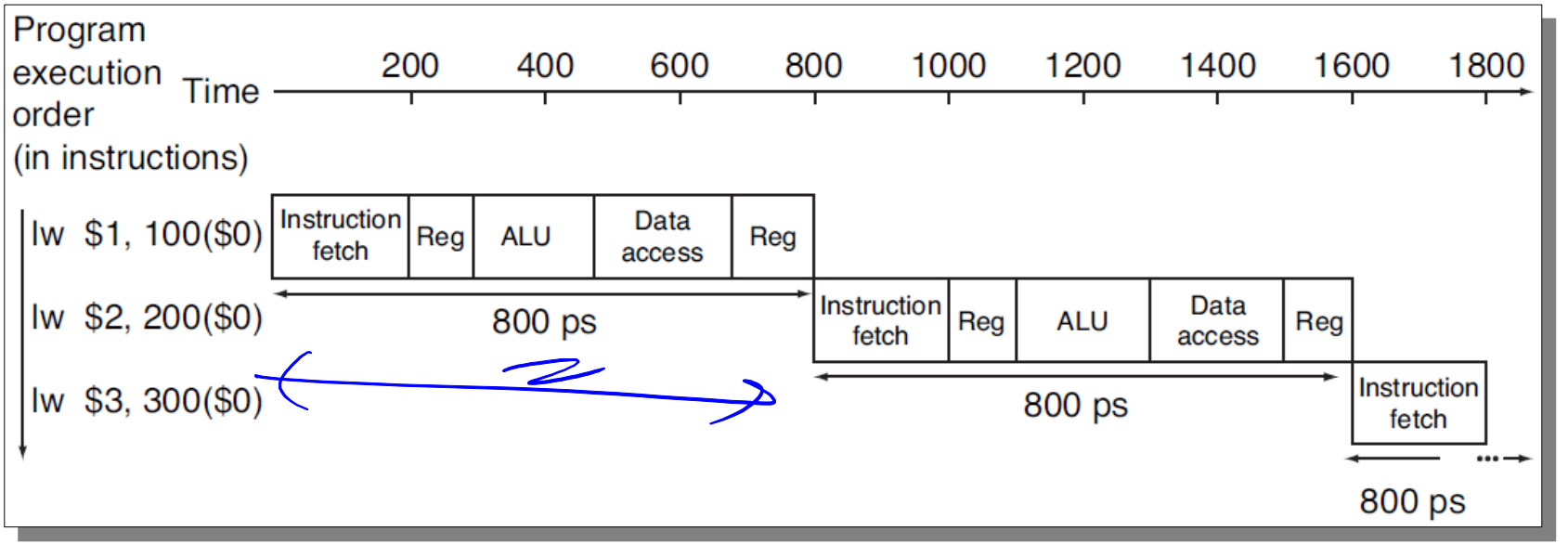
Pipelined vs. Nonpipelined Implementation

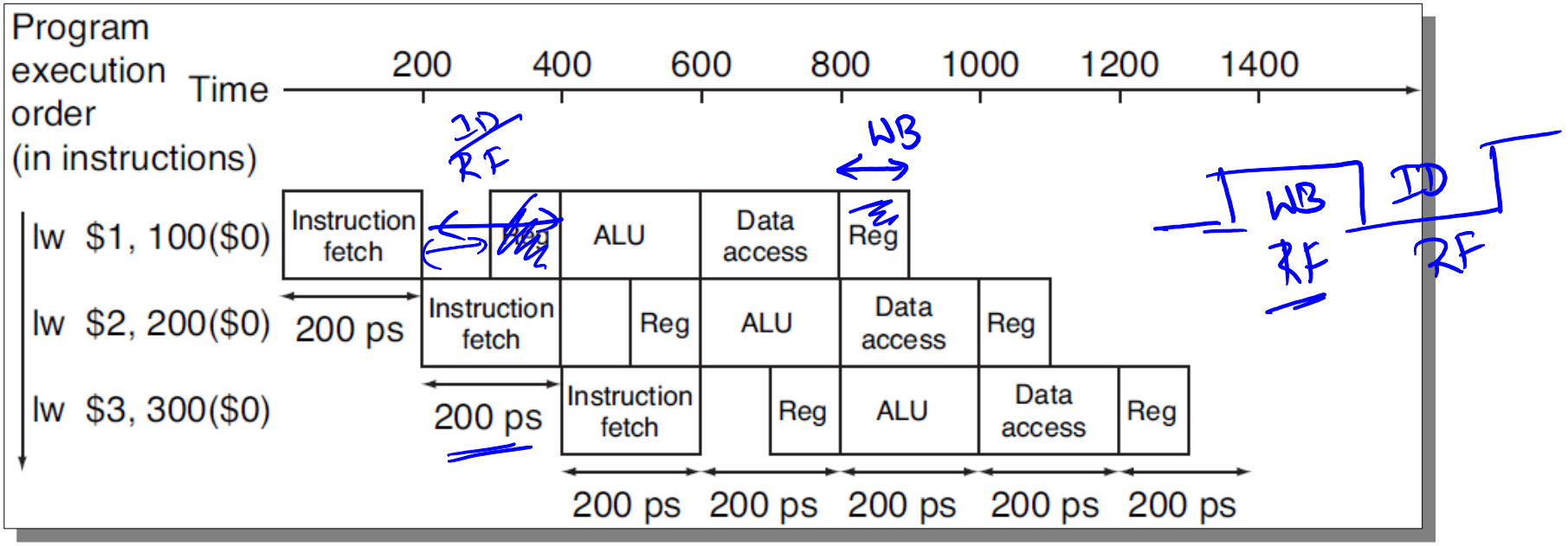
- Ratio of execution times between the two?
  - For 10^6 instructions?
- Pipelining increases the instruction throughput opposed to individual instruction execution time.

Pipelined vs. Nonpipelined Implementation



Pipelined vs. Nonpipelined Implementation





Speedup of the Pipeline

- The speedup of a k stage pipelined processor over an unpipelined processor

$$S_k = \frac{T_{unpipelined}}{T_{pipelined}} = \frac{n \cdot k}{k + (n-1)}$$

- n: number of instructions in the program.
- k: number of pipeline stages

Module Outline

- Why Pipeline?
  - How to pipeline?
- Speedup of the pipeline
- Pipelined datapath
  - Execution of instructions
  - Pipeline Timing diagram
- Dependences, Hazards
  - Structural, Data, Control
  - Stalling, Forwarding
- Branch prediction