

---

# EECE 321: Computer Organization

Mohammad M. Mansour

*Dept. of Electrical and Compute Engineering*

*American University of Beirut*

Lecture 22: Pipelining

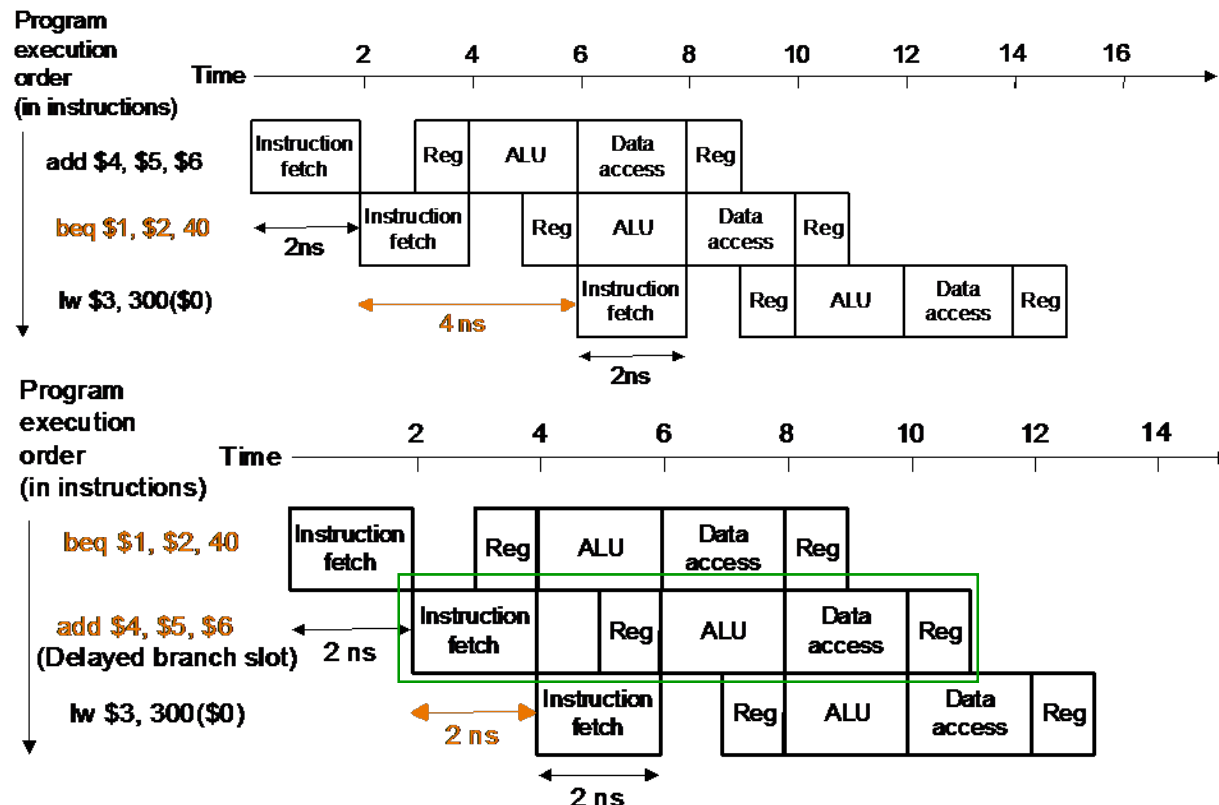
---

## 2. Control Hazards

### ■ Solution 3 – Delayed branches:

- This solution is actually used in MIPS.
- Place an instruction that is not affected by the branch (e.g. an instruction appearing before the branch) immediately after it.
- Delay taking the branch 1 clock cycle (i.e., delay loading of PC one more clock cycle)

- Example: `add $4,$5,$6` doesn't affect the branch, so it can be moved into the delayed branch slot.



# Summary of Control Hazard Solutions

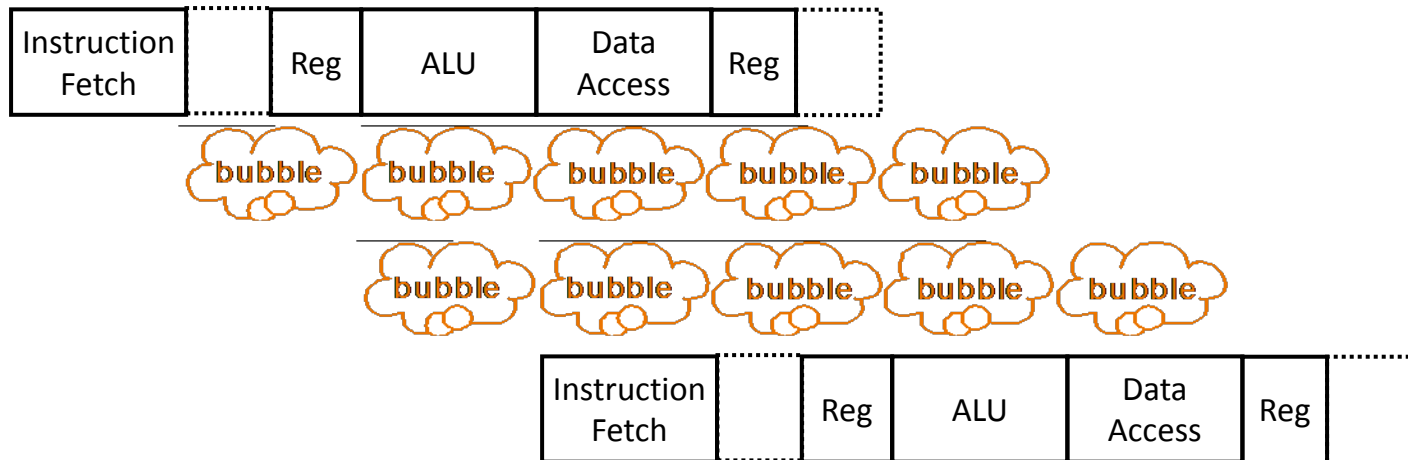
---

- Stall the pipeline
- Do branch prediction
- Use delayed branches

### 3. Data Hazards

- This occurs when the next instruction depends on the result generated by the current instruction:  

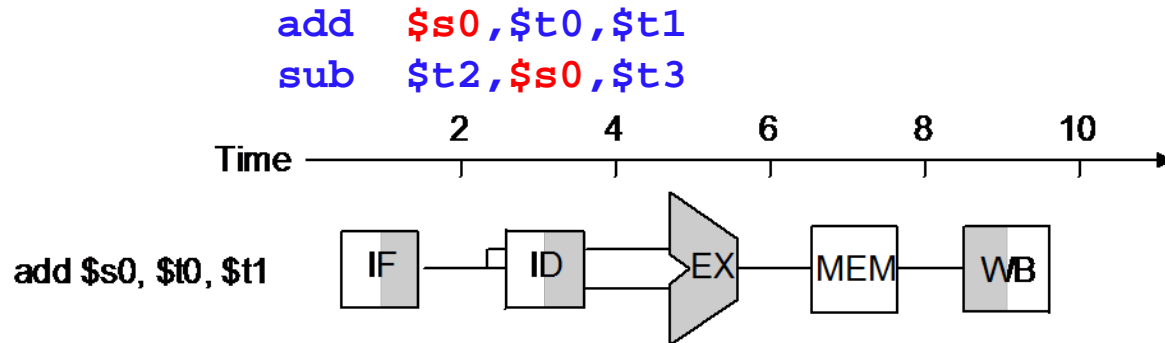
<code>add</code>	<code>\$s0,\$t0,\$t1</code>	<code>#producer of \$s0</code>
<code>sub</code>	<code>\$t2,\$s0,\$t3</code>	<code>#consumer of \$s0</code>
- The add instruction doesn't write the result until the 5th stage, so we need to add two bubbles to the pipeline (2 NOPs in MIPS).



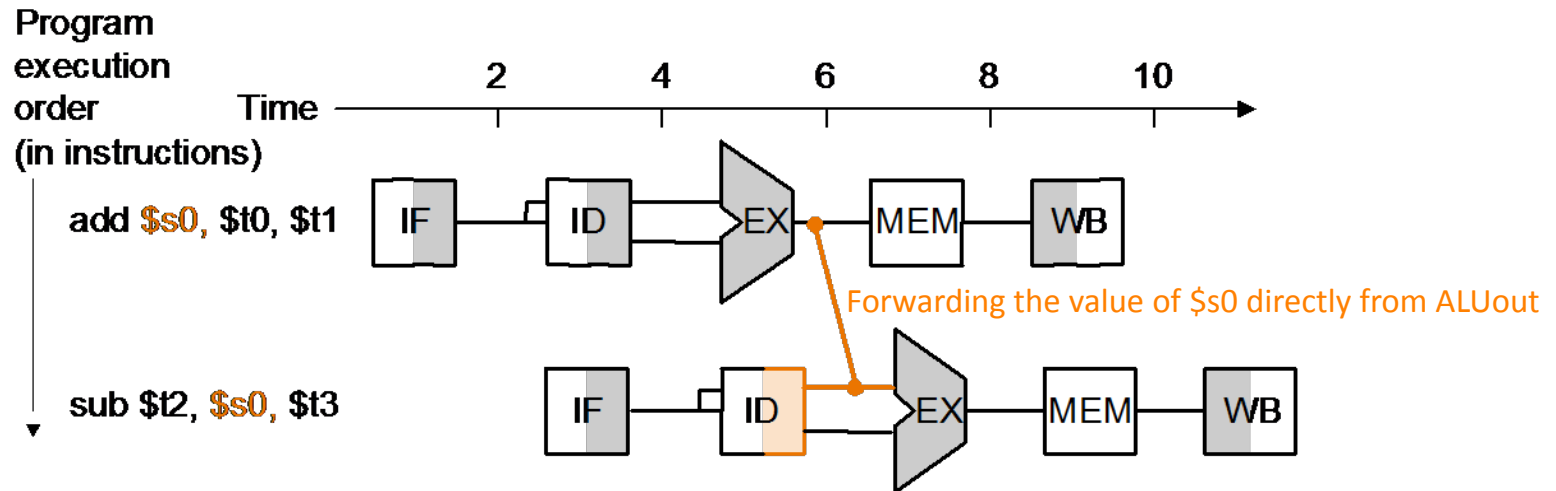
- Solution:** Observe that we don't have to wait for the first add instruction to complete to resolve the data hazard.
  - As soon as the first add finishes its third stage, the sum is ready and can be **forwarded** to sub.
- Getting the missing item early from the internal resources is called **register forwarding** or **bypassing**.

### 3. Data Hazards – Example1

- For the two instructions below, show what pipeline stages can be connected by forwarding. Use the figure below to represent the datapath during the 5 stages.



- Solution:

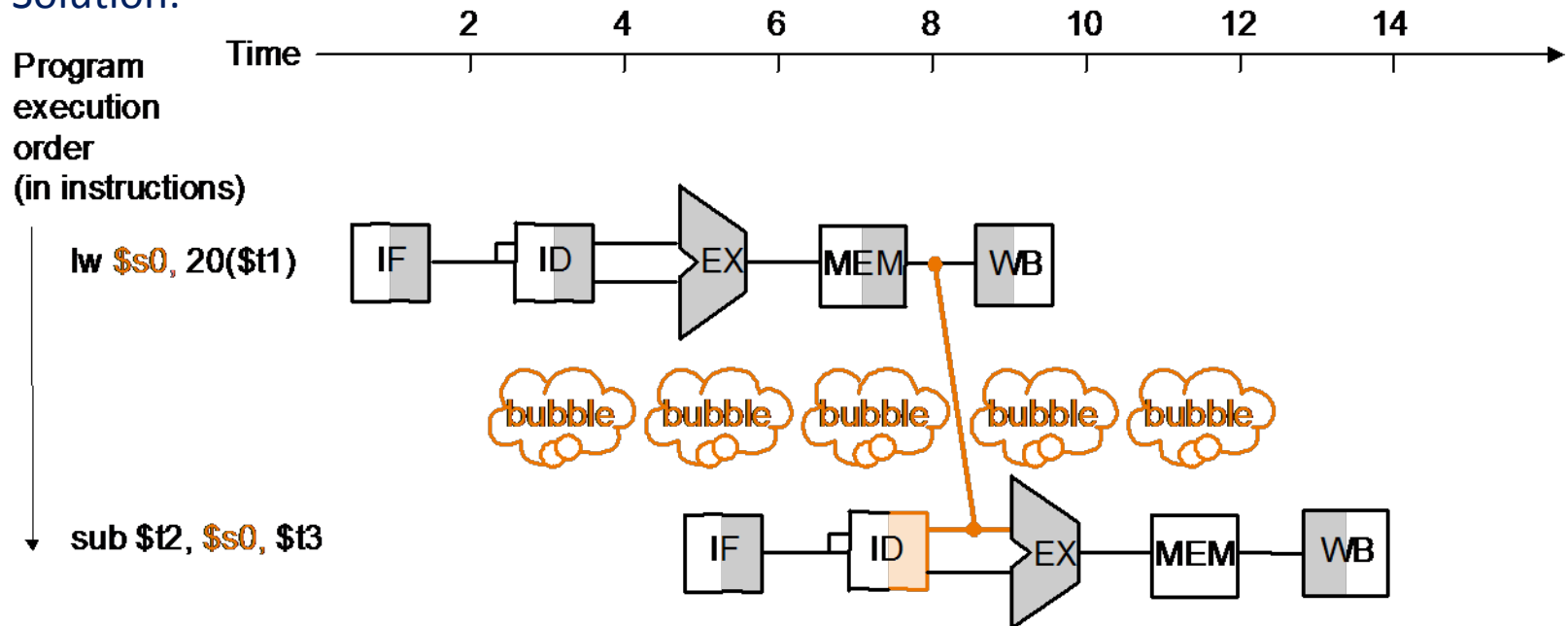


### 3. Data Hazards – Example2

- Repeat for the following pair of instructions. Does forwarding remove all stalls?

```
lw    $s0, 20($t1)
sub    $t2, $s0, $t3
```

- Solution:



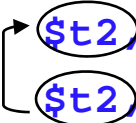
- Here since result of load is available only after the 4th stage in MDR, and sub needs it in 3rd stage, still need to insert a pipeline bubble.

### 3. Data Hazards – Reordering Code to Avoid Pipeline Stalls

---


- Can the assembler or compiler rearrange the code to eliminate stalls?

```
lw    $t0, 0($t1)
lw    $t2, 4($t1)
sw    $t2, 0($t1)
sw    $t0, 4($t1)
```

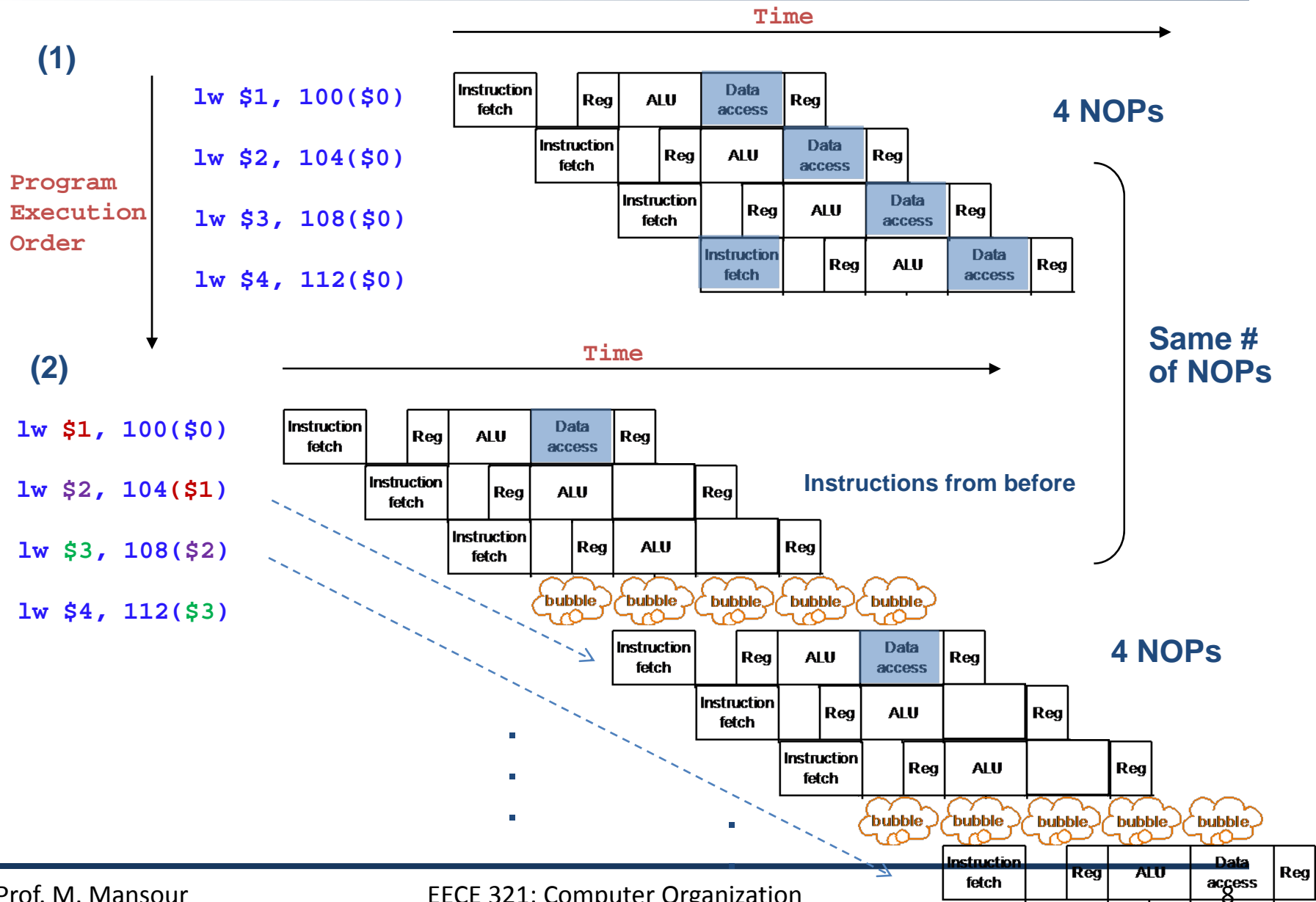


- Solution:

```
lw    $t0, 0($t1)
lw    $t2, 4($t1)
sw    $t0, 4($t1)
sw    $t2, 0($t1)
```

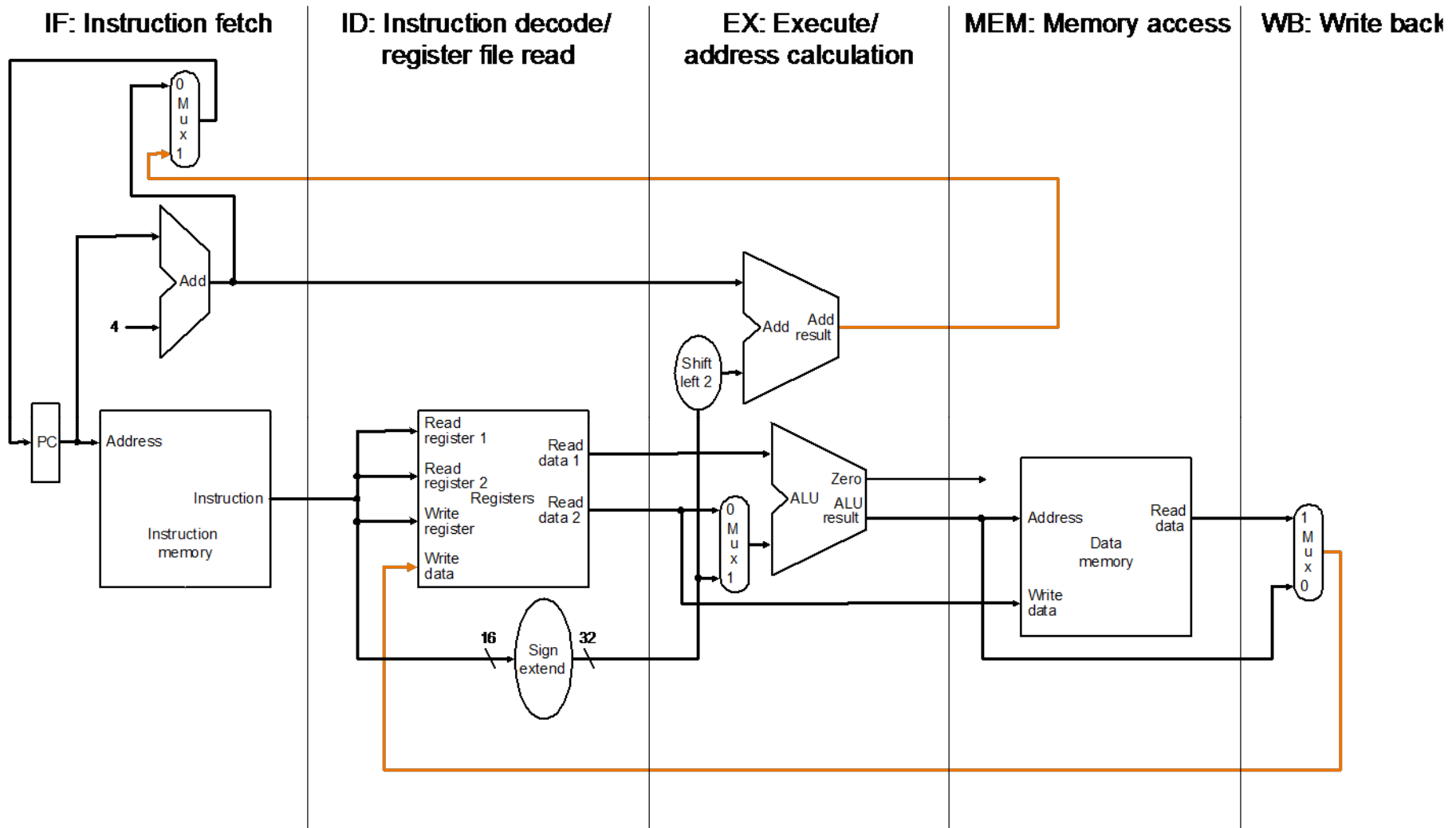


# Reordering Code to Support Structural & Data Hazards

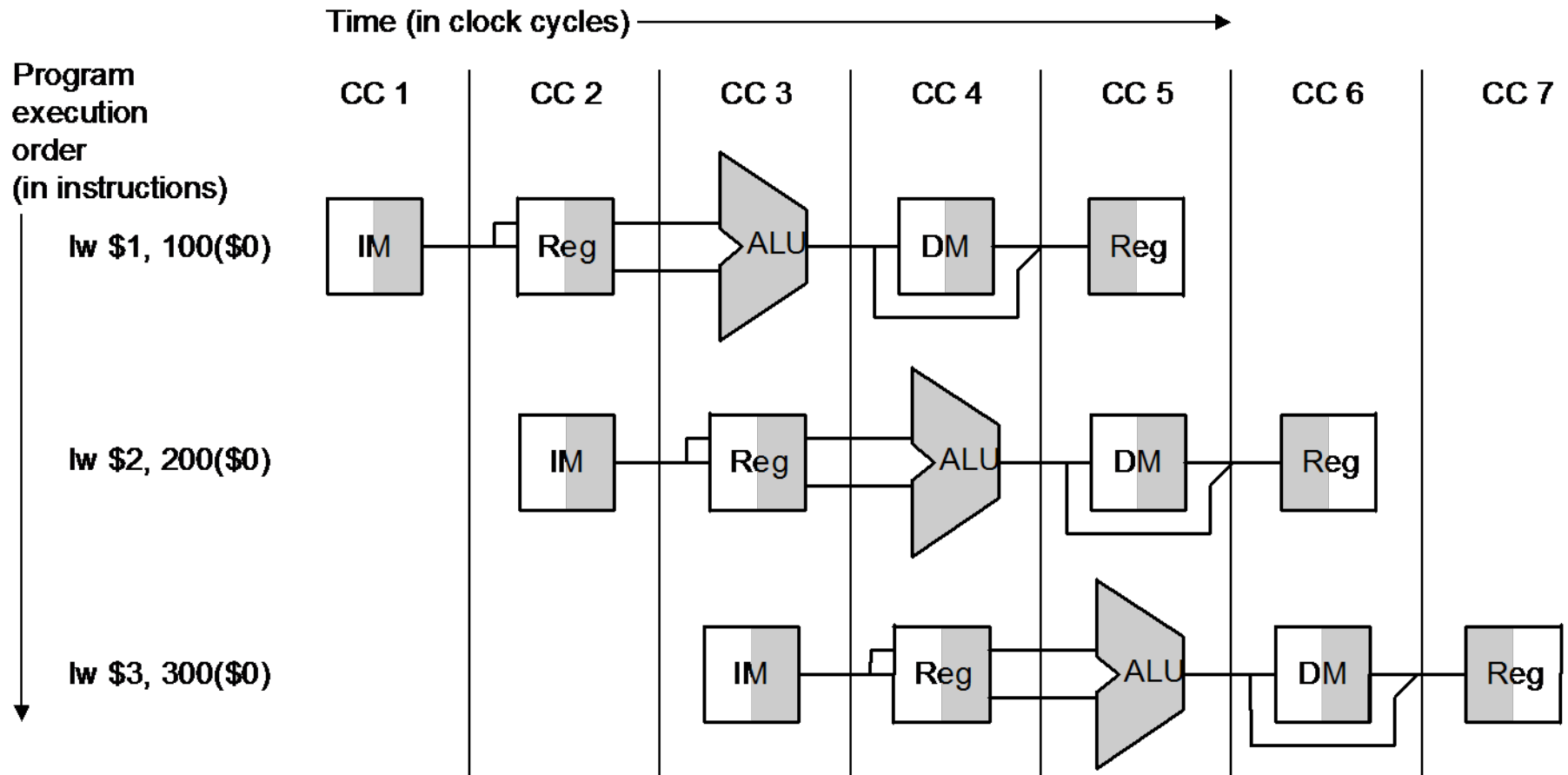




# Reorganized Single-Cycle Datapath

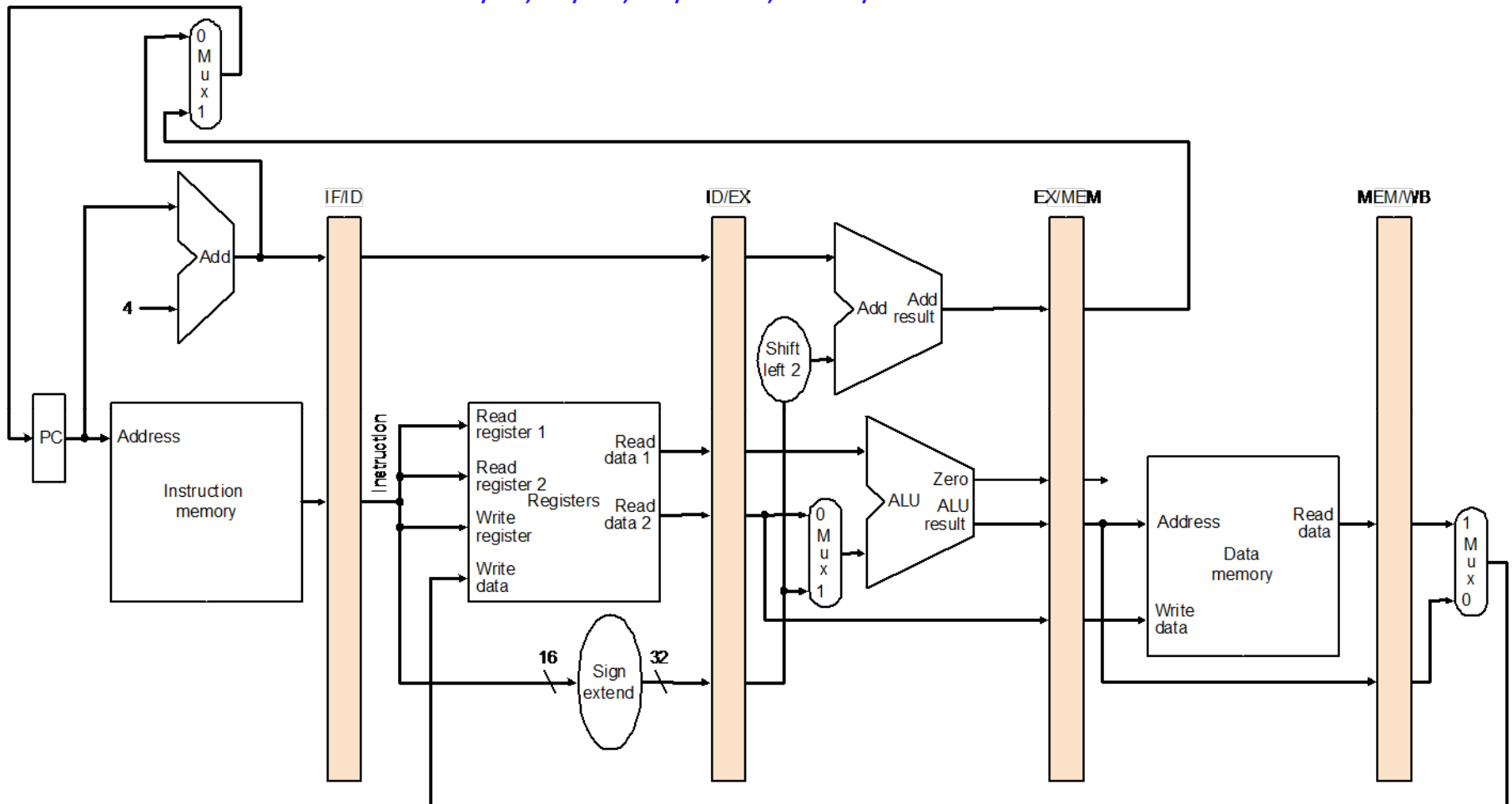


# Pipelined Execution

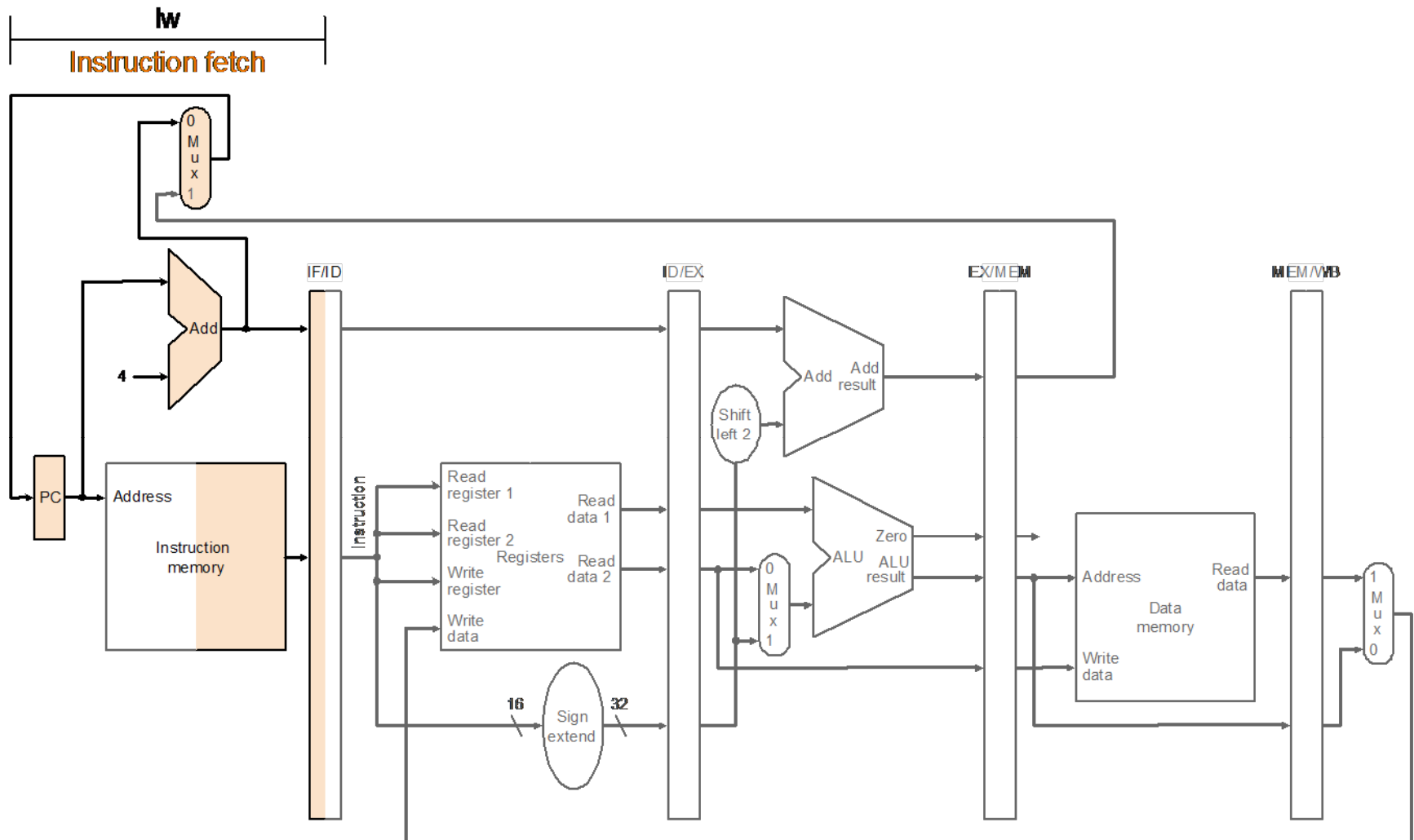


# Pipelined Datapath: Adding Pipeline Registers

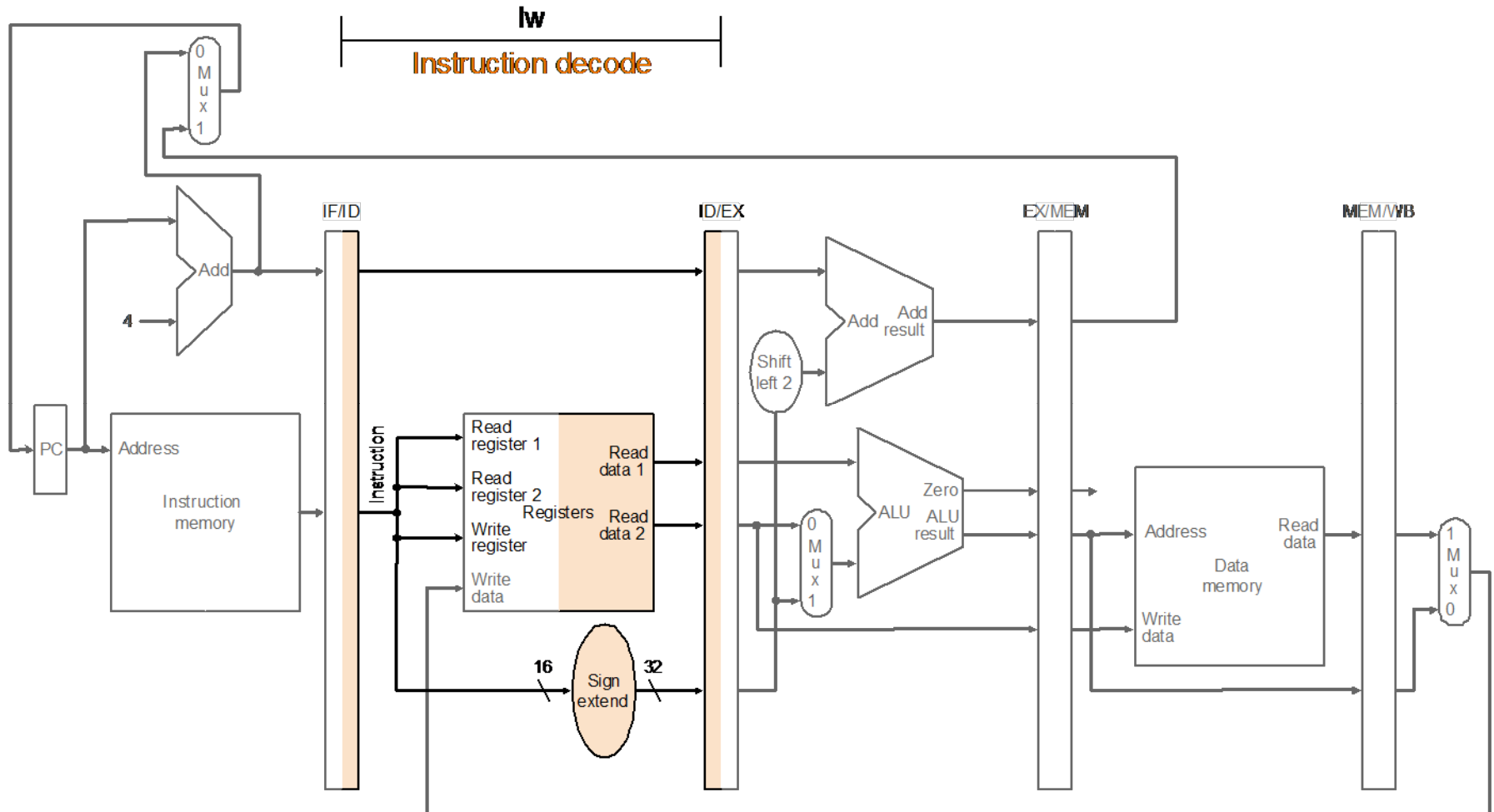
- Forward necessary information used in later execution stages using pipeline registers
  - Give them names: IF/ID, ID/EX, EX/MEM, MEM/WB



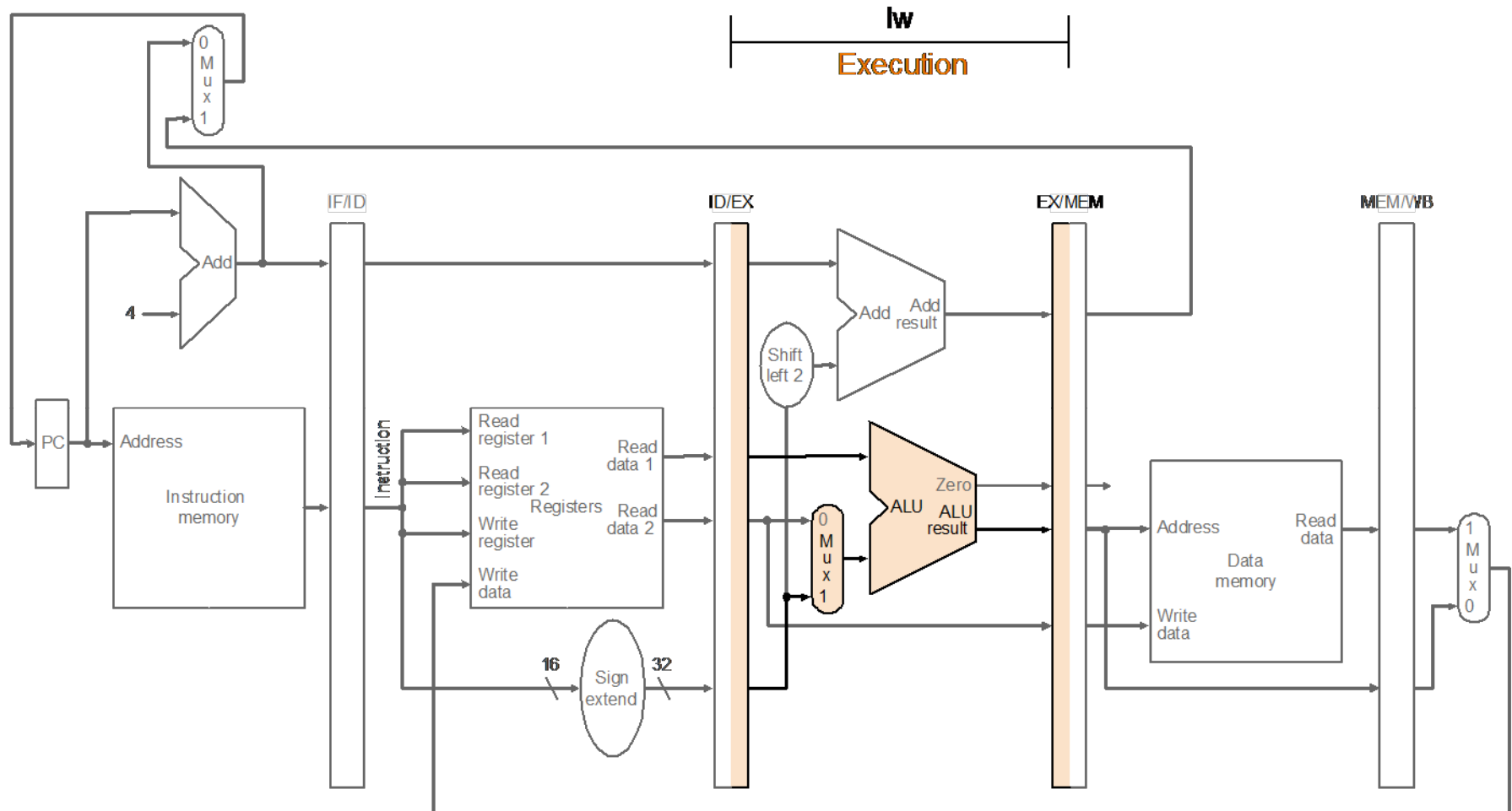
# Execution of `lw` on Pipelined Datapath: IF (1/5)



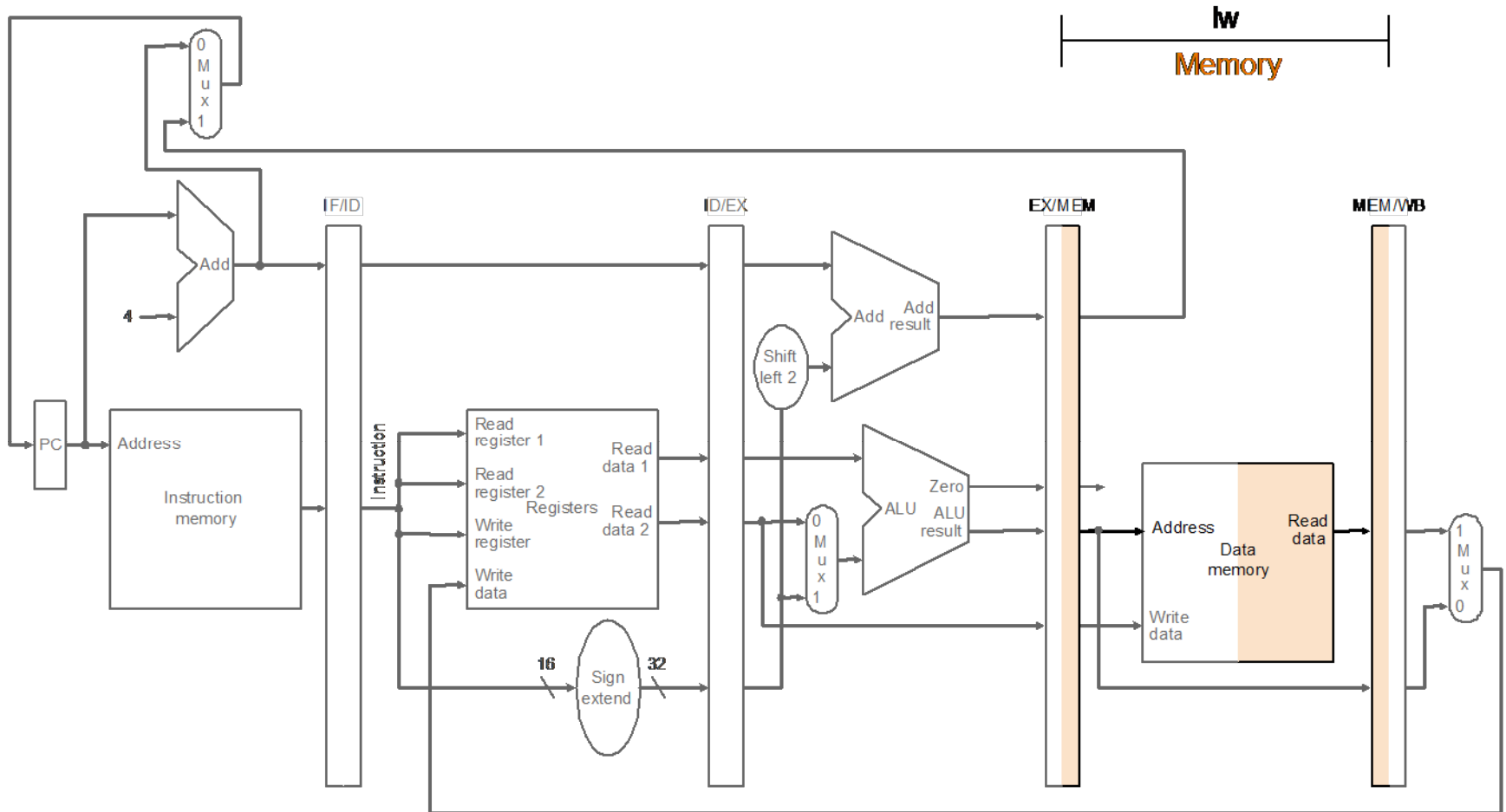
## Execution of **lw** on Pipelined Datapath: ID (2/5)



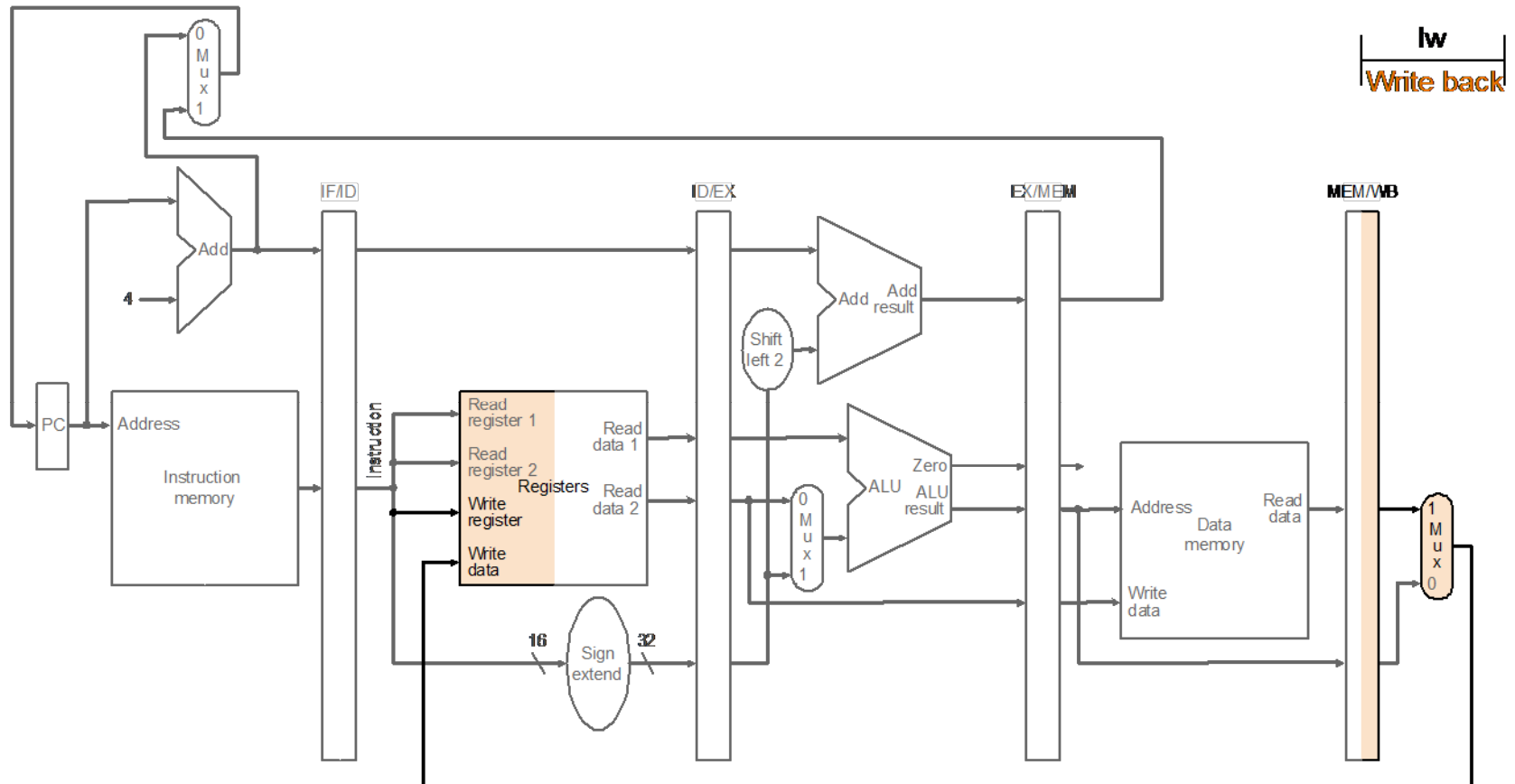
## Execution of **lw** on Pipelined Datapath: EX (3/5)



# Execution of **lw** on Pipelined Datapath: MEM (4/5)



## Execution of **lw** on Pipelined Datapath: WB (5/5)



- Where is the loaded value written? Above datapath has a problem.



# Corrected Pipelined Datapath to Properly Handle 1w

