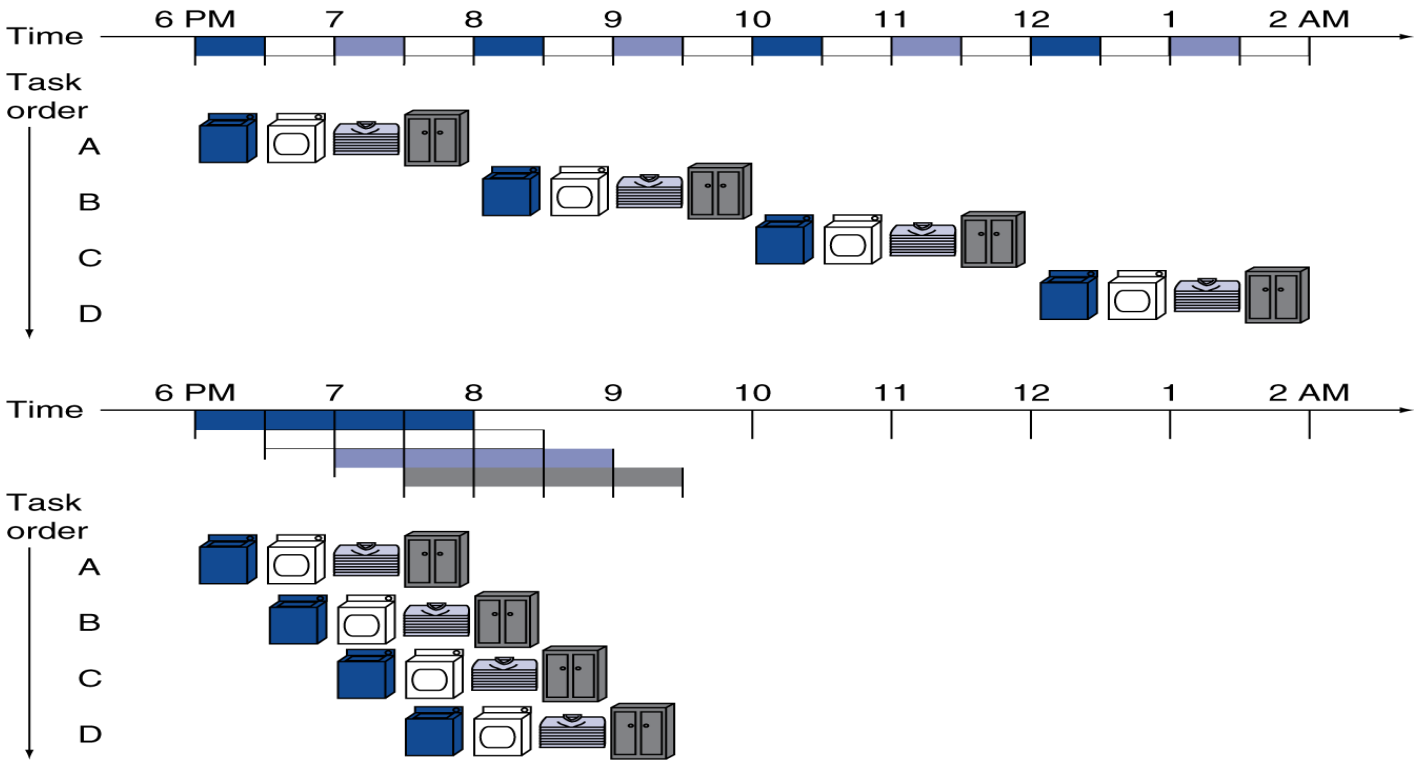
Advanced Computer Architecture

Pipelining

Computer Science

Pipelining Analogy

- Pipelined laundry: overlapping execution
  - Parallelism improves performance



Pipelining: Improving Performance

	Latency	Max. Throughput
Non-Pipelined	2 hours	0.5
Pipelined	2 hours	2

Length of time for each load does not change.

Assuming all stages of pipeline are busy at all times.

Latency = time from start of one load to the end of same load.

Maximum Throughput = # of loads completed per hour.

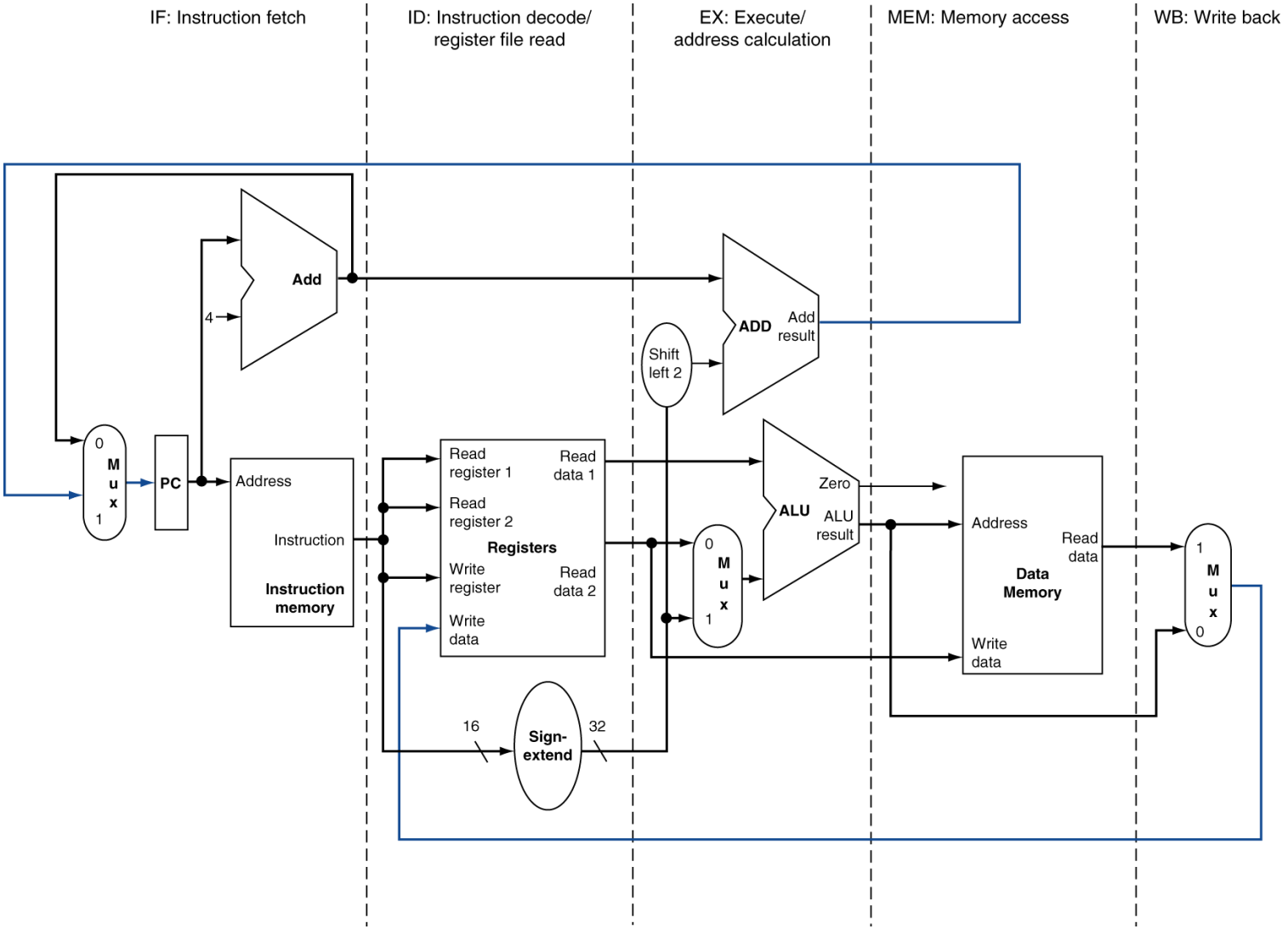
Pipelining: Improving Performance

- Objective: Keep all stages of the pipeline busy at all times
- Pipelining improves performance by increasing instruction throughput, rather than decreasing execution time of an individual instruction.

MIPS Pipeline

- Five stages, one step per stage
  - IF : Instruction fetch from memory
  - ID : Instruction decode & register read
  - EX : Execute operation or calculate address
  - MEM : Access memory operand
  - WB : Write result back to register

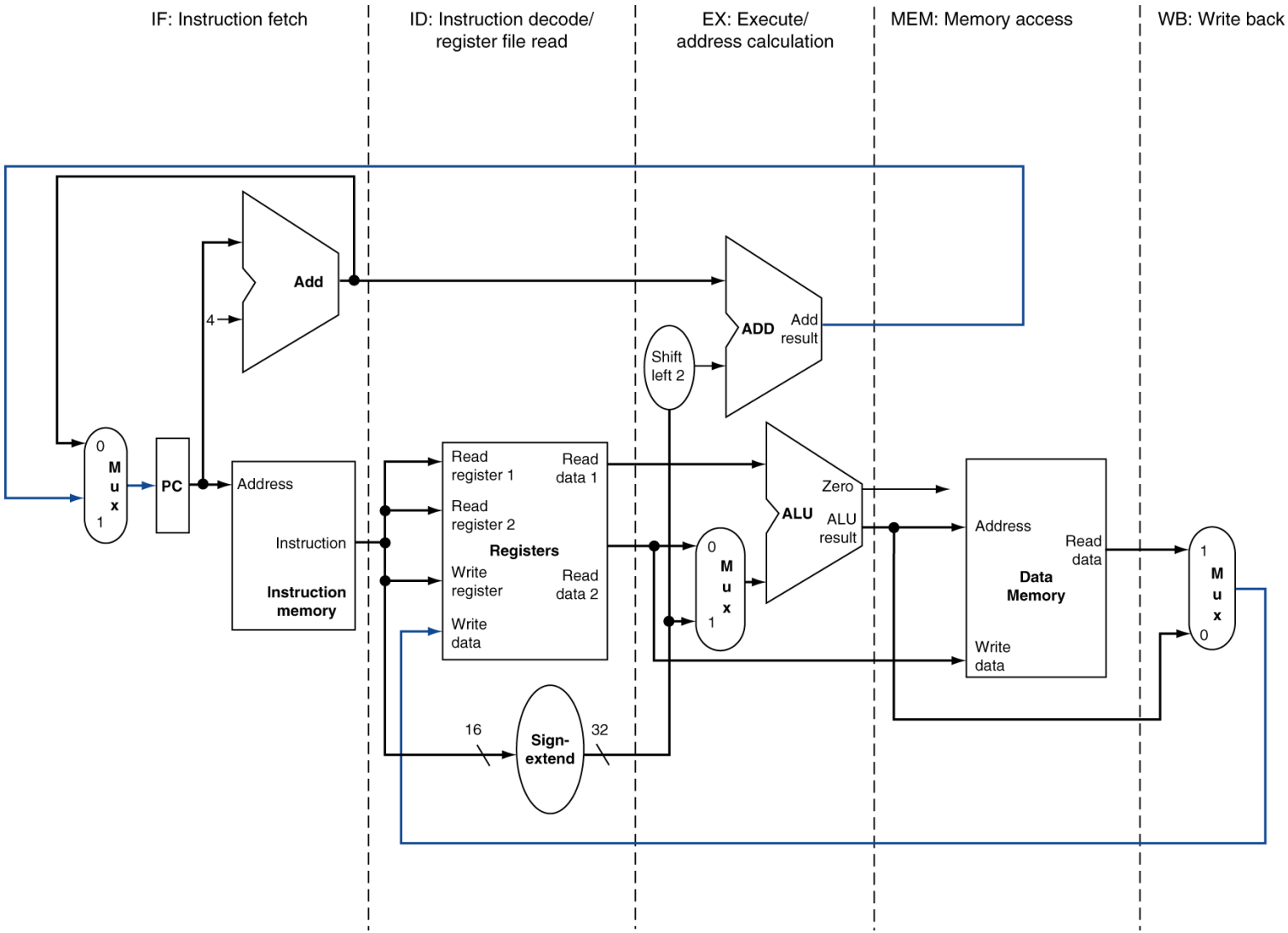
MIPS Pipeline



Pipelining and ISA Design

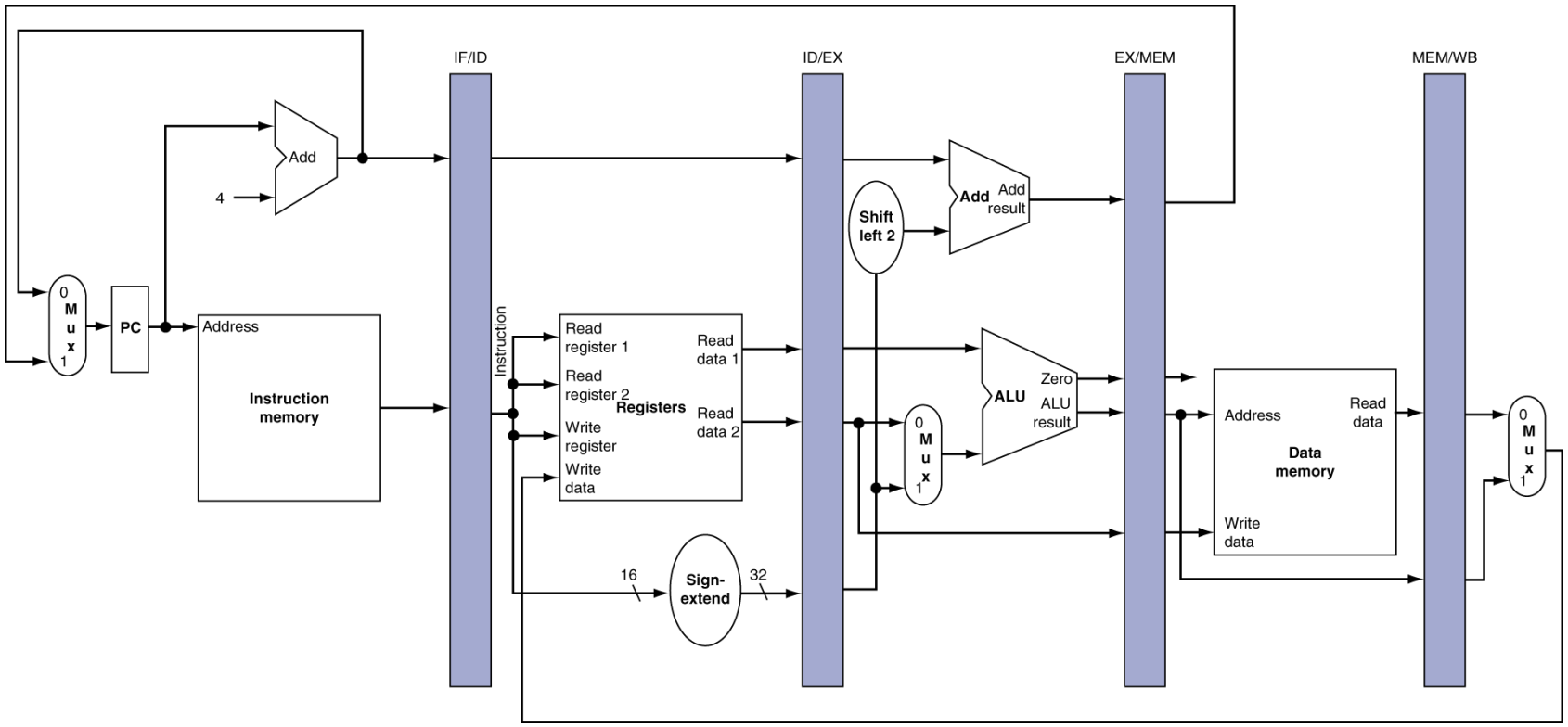
- MIPS ISA designed for pipelining
  - All instructions are 32-bits
    - Easier to fetch and decode in one cycle
    - x86: 1- to 17-byte instructions
  - Few and regular instruction formats
    - Can decode and read registers in one step
  - Alignment of memory operands
    - i.e. on word boundaries
    - Memory access takes only one cycle

MIPS Pipelined Datapath



Pipeline registers

- Need registers between stages
  - To hold information produced in previous cycle

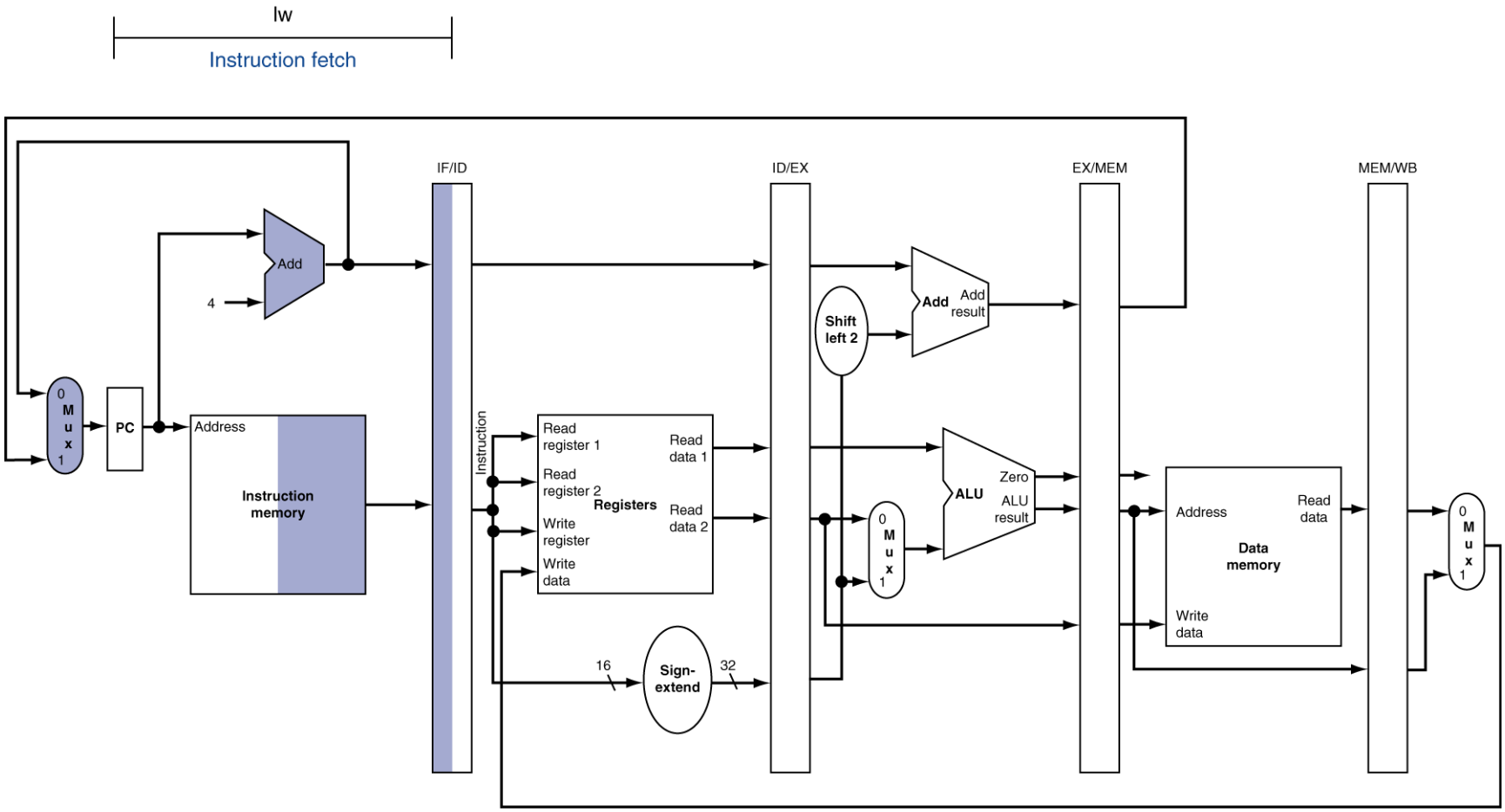


Pipeline Operation

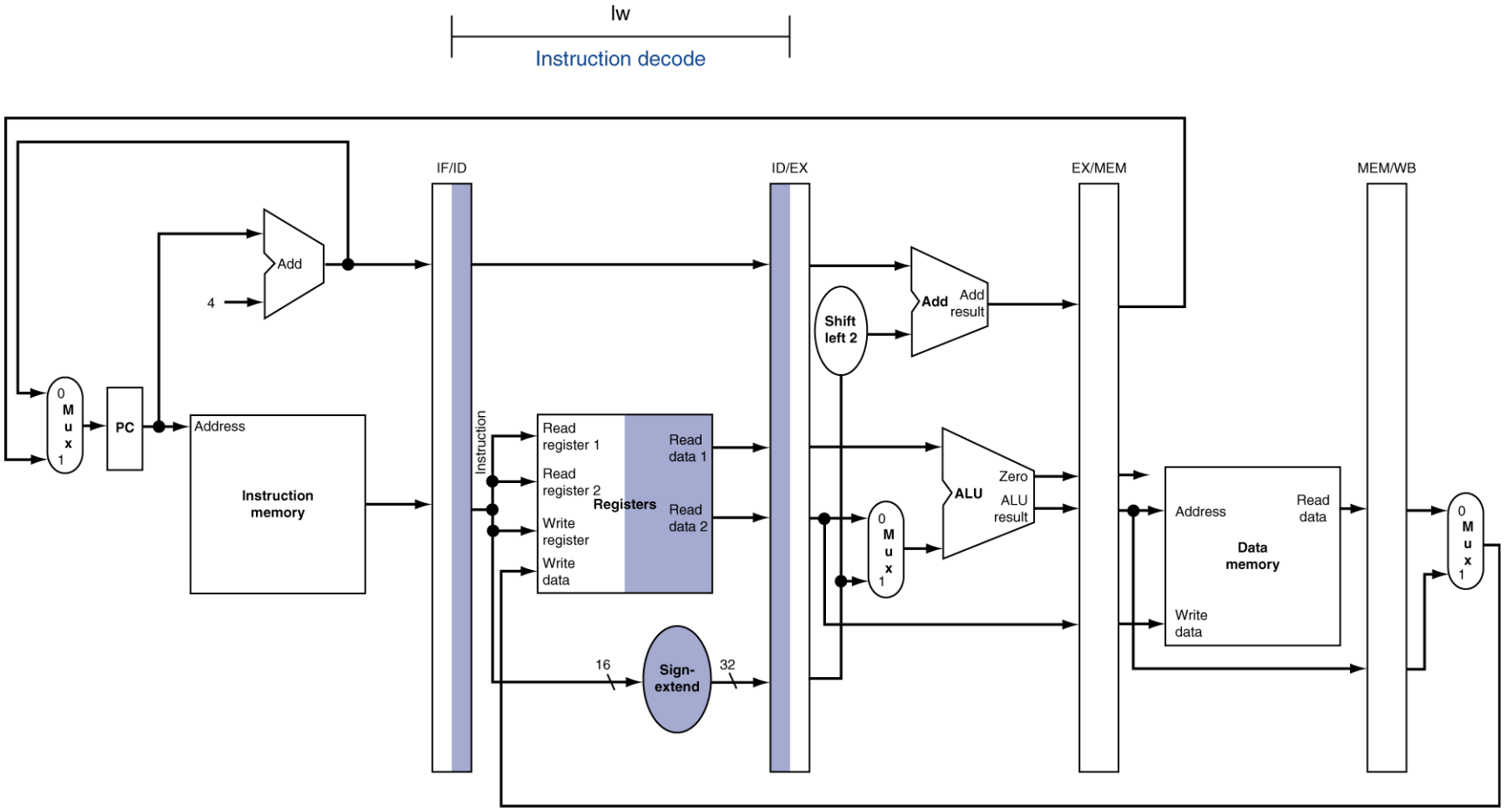
- Cycle-by-cycle flow of instructions through the pipelined datapath
  - "Single-clock-cycle" pipeline diagram
    - Shows pipeline usage in a single cycle
    - Highlight resources used
  - "Multi-clock-cycle" diagram
    - Graph of operation over time

- We'll look at "single-clock-cycle" diagrams for load word and store word.

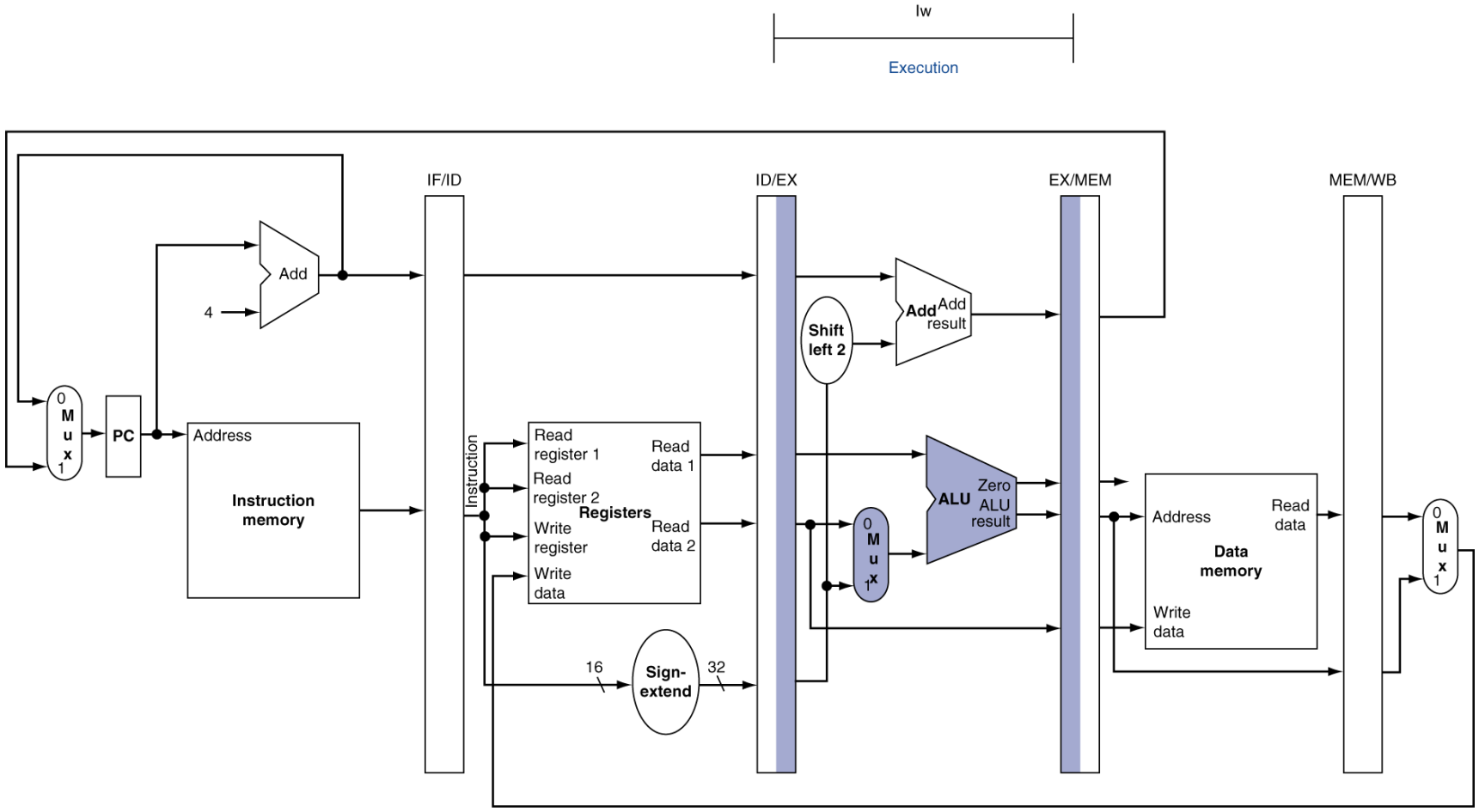
IF for Load, Store, …



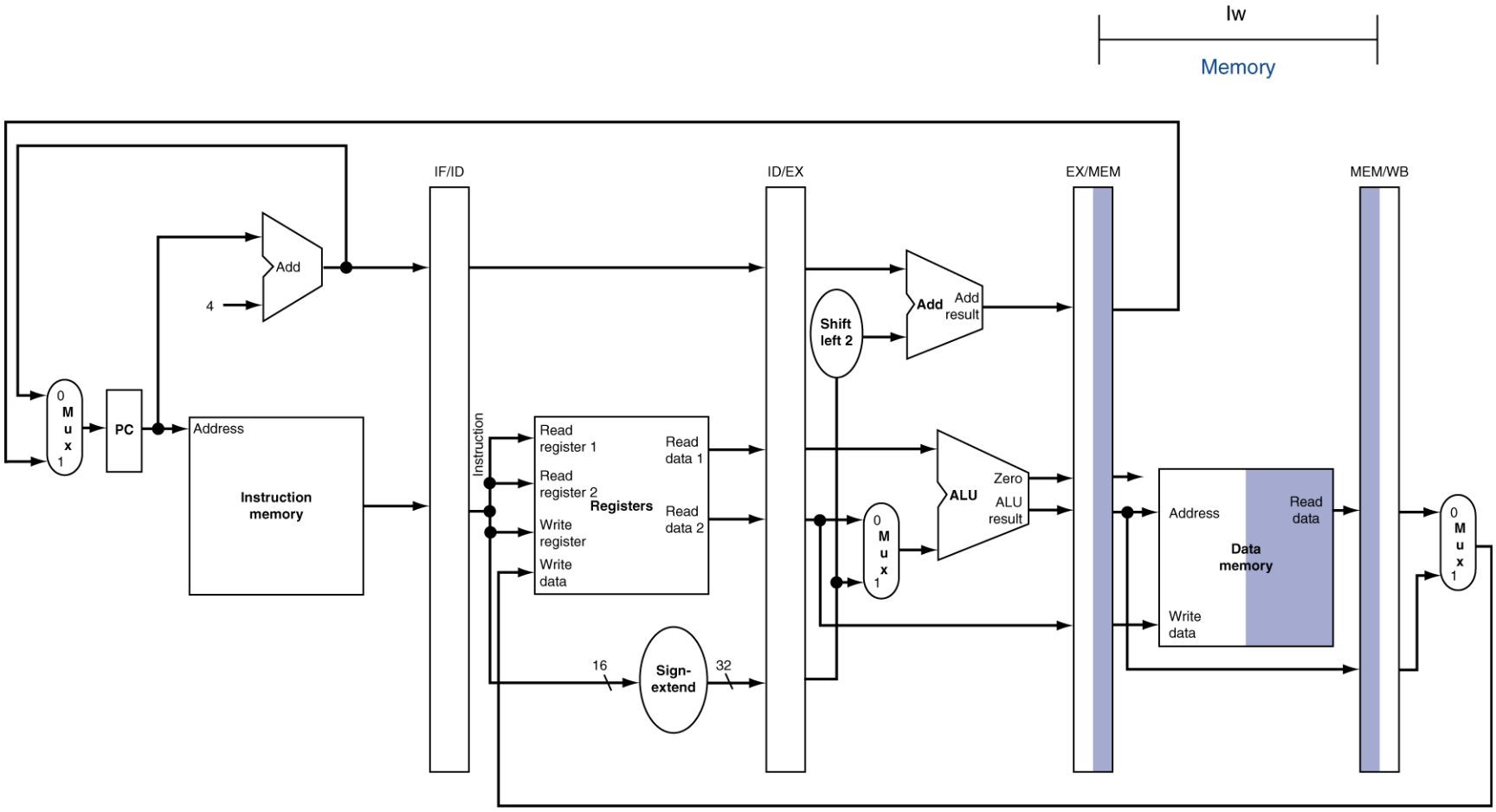
ID for Load, Store, …



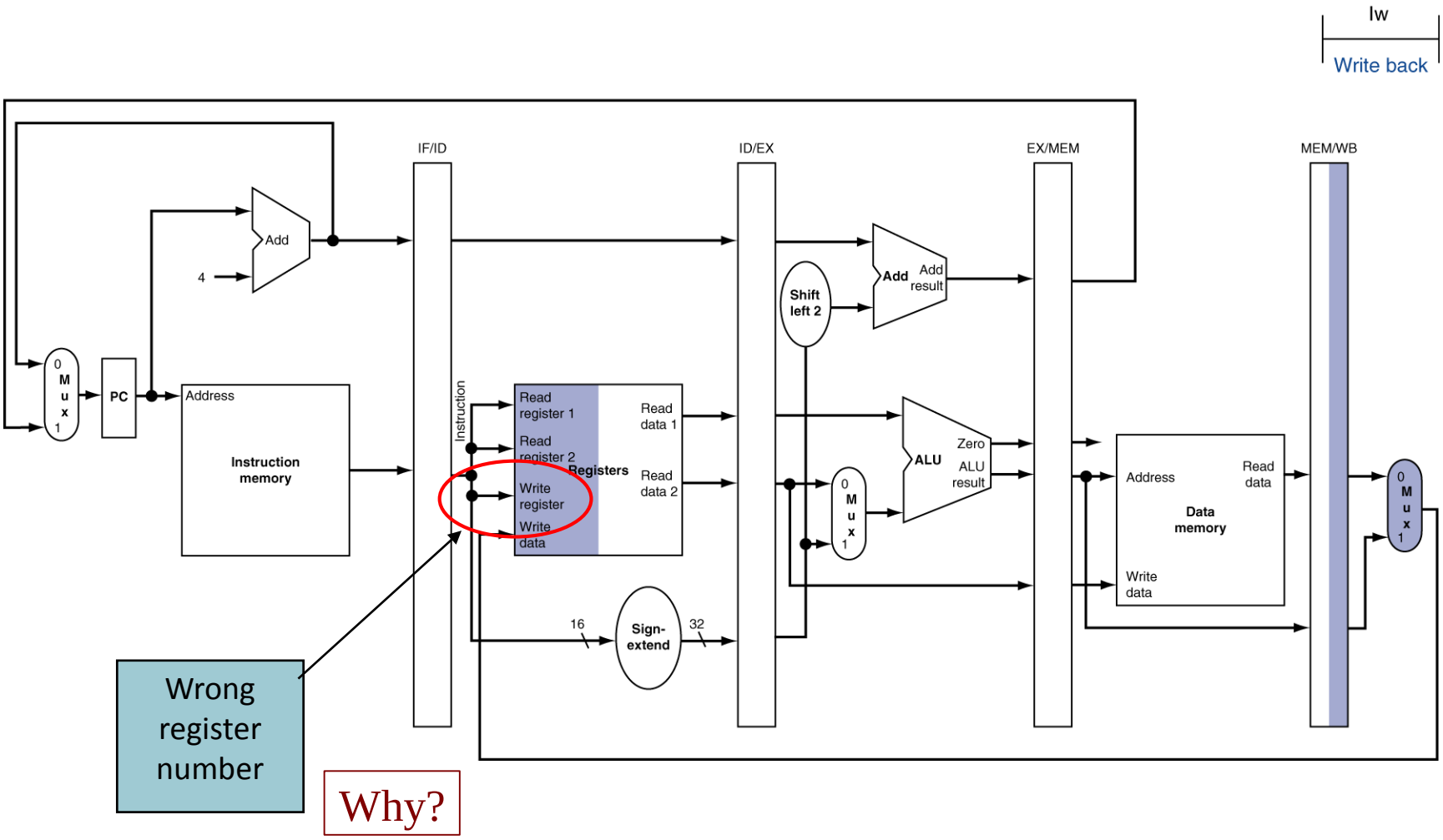
EX for Load


lw
Execution
IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
M u x
0
1
PC
Address
Instruction memory
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Shift left 2
Add
Add result
ALU
Zero
ALU result
16
Sign-extend
32
Address
Read data
Data memory
Write data

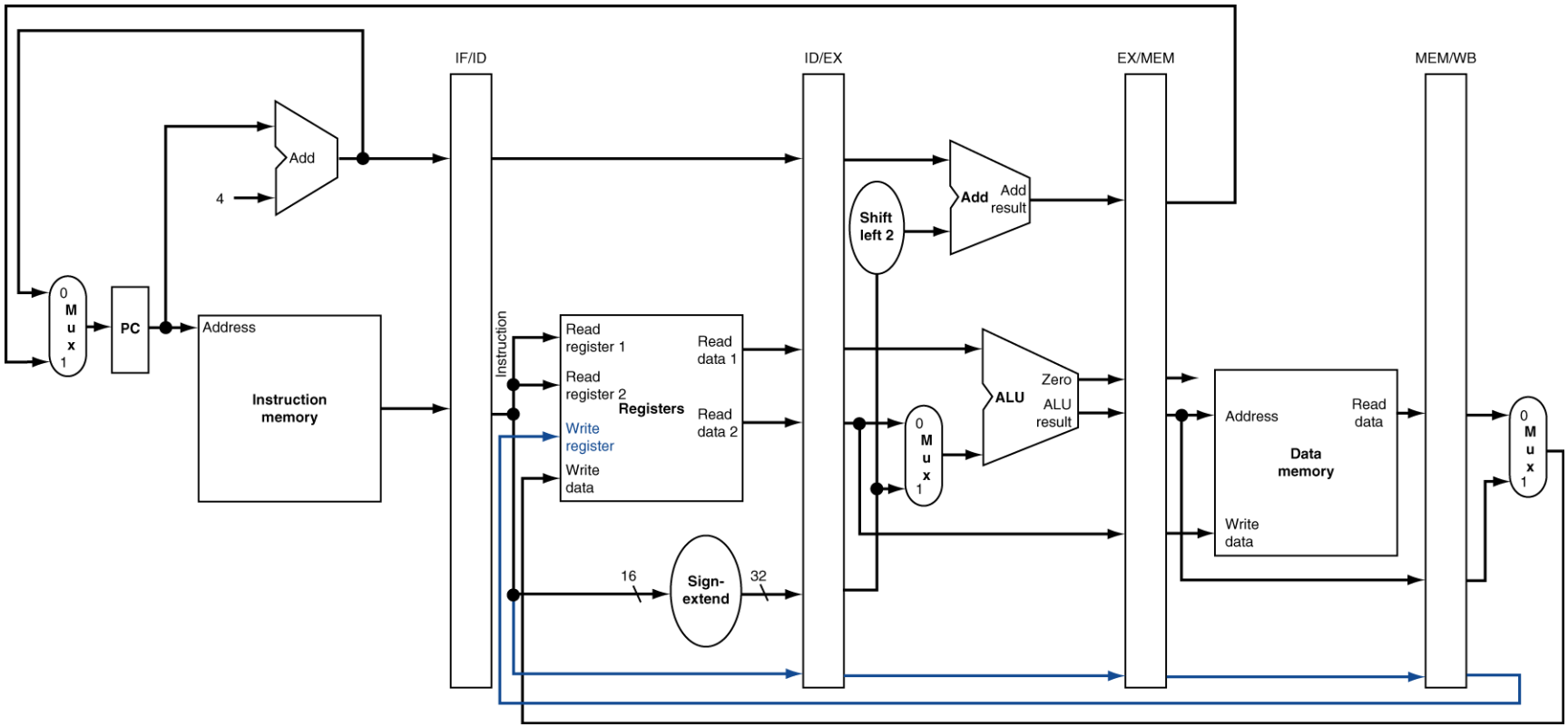
MEM for Load


lw
Memory
IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
M u x
0
1
PC
Address
Instruction memory
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Shift left 2
Add
Add result
ALU
Zero
ALU result
16
Sign-extend
32
Address
Data memory
Read data
Write data

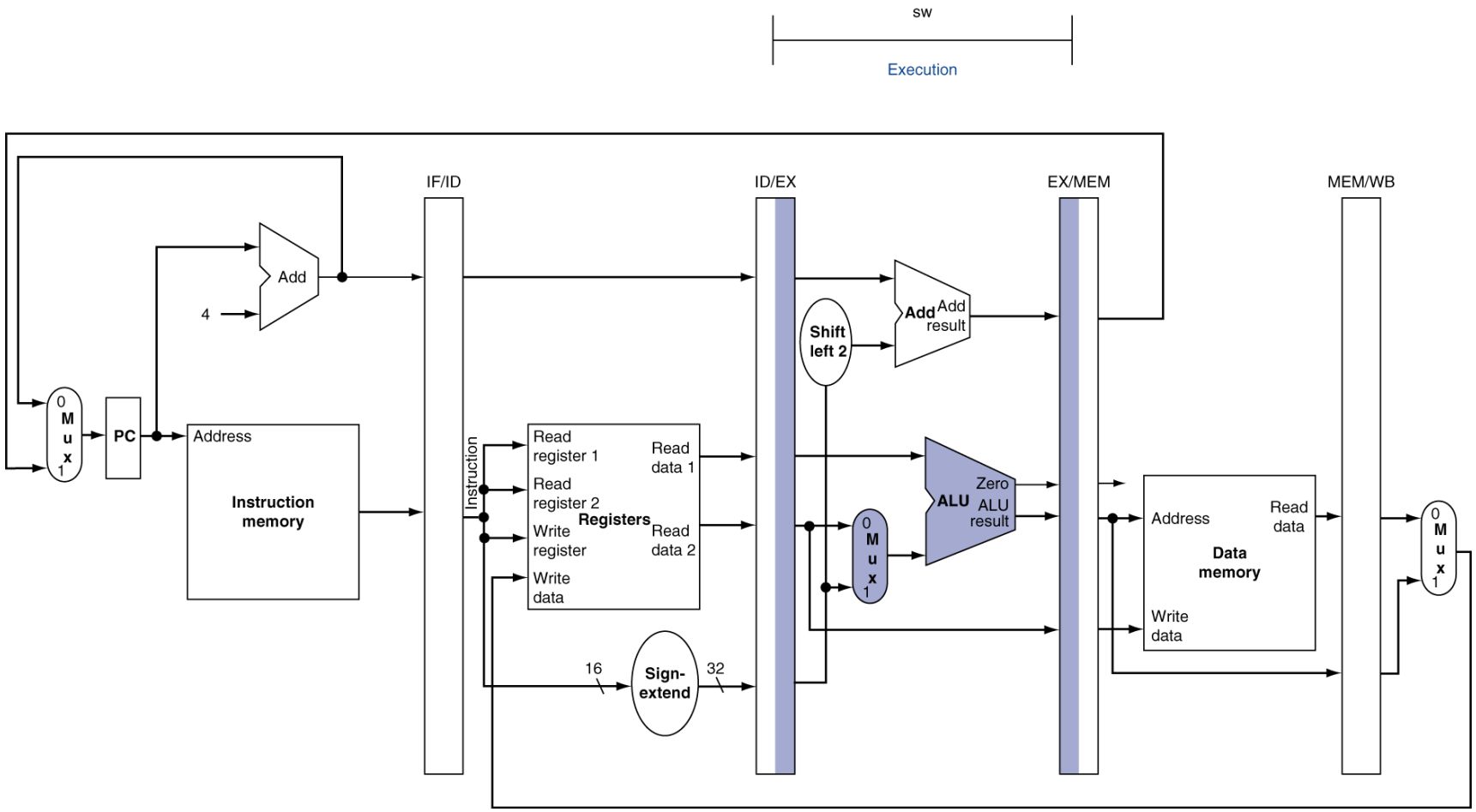
WB for Load


lw
Write back
IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
M u x
0
1
PC
Address
Instruction memory
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Shift left 2
Add
Add result
ALU
Zero
ALU result
16
Sign-extend
32
Address
Read data
Data memory
Write data
Wrong register number
Why?

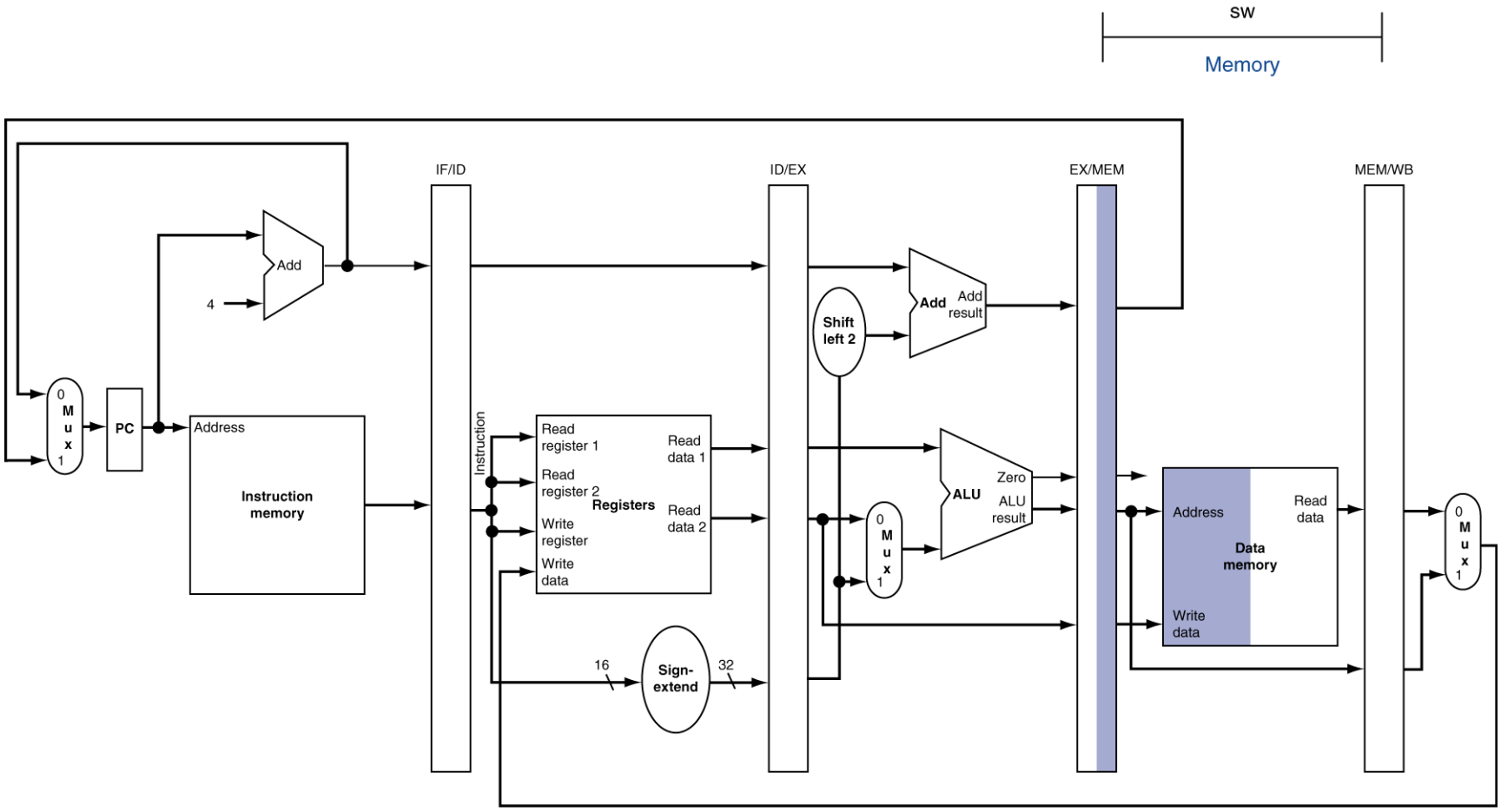
Corrected Datapath for Load



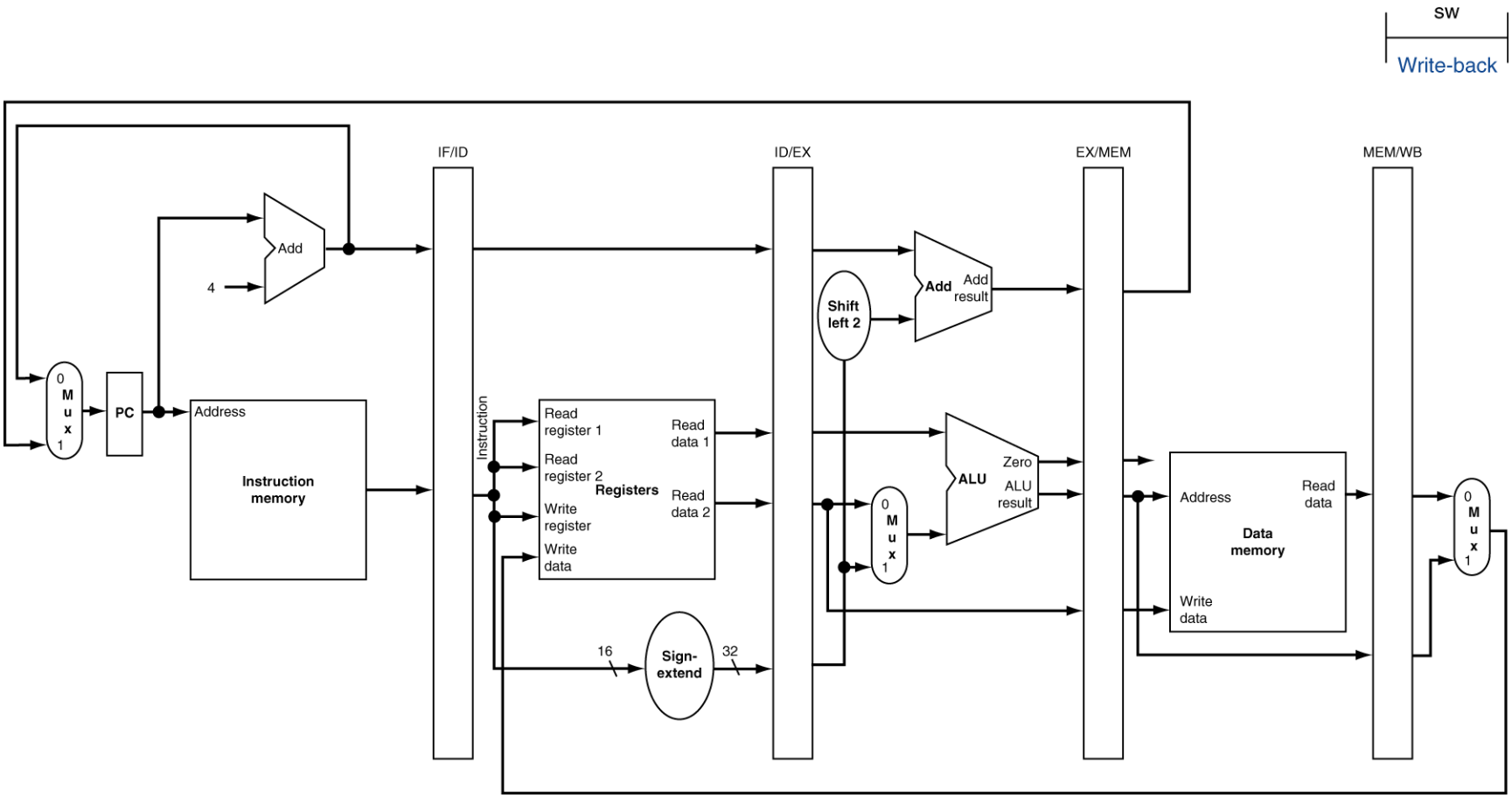
EX for Store


sw
Execution
IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
M u x
0
1
PC
Address
Instruction memory
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Shift left 2
Add
Add result
ALU
Zero
ALU result
16
Sign-extend
32
Address
Read data
Data memory
Write data

MEM for Store

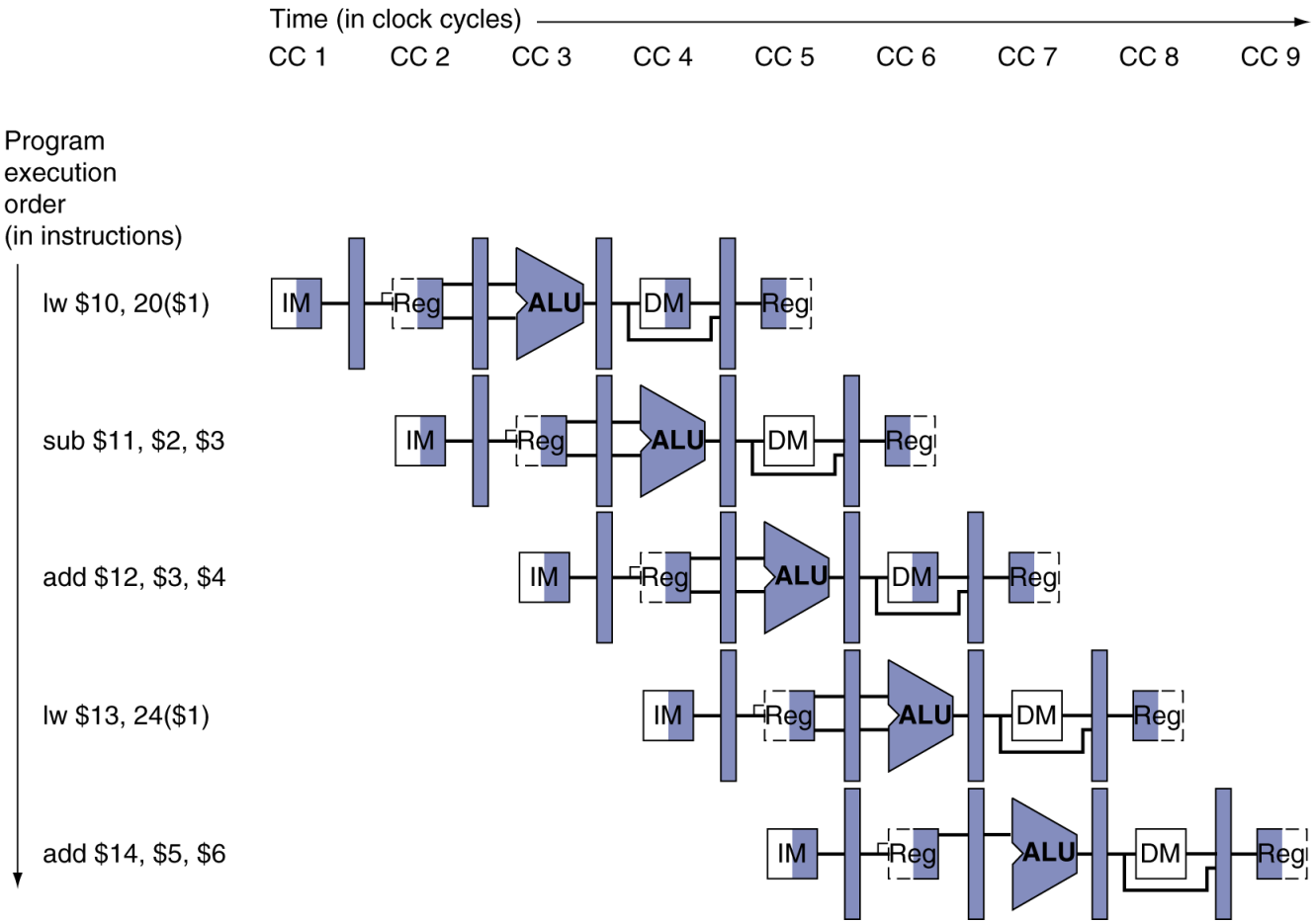


WB for Store


SW
Write-back
IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
M u x
0
1
PC
Address
Instruction memory
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Shift left 2
Add
Add result
ALU
Zero
ALU result
16
Sign-extend
32
Address
Read data
Data memory
Write data

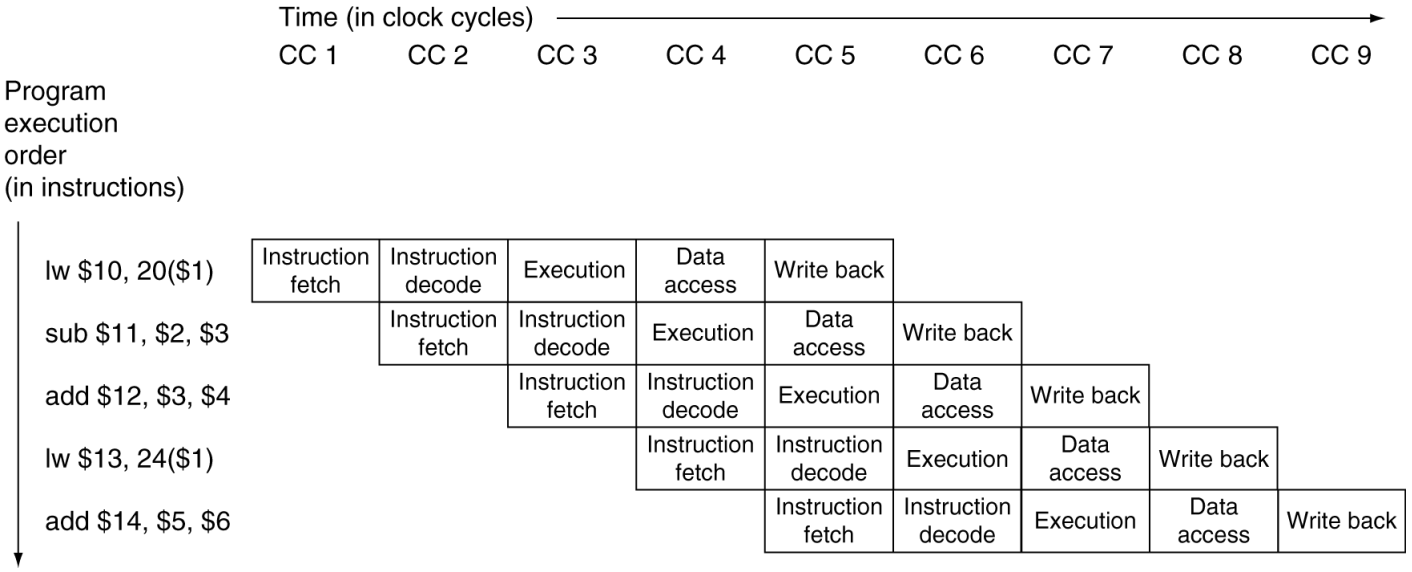
Multi-Cycle Pipeline Diagram

- Form showing resource usage



Multi-Cycle Pipeline Diagram

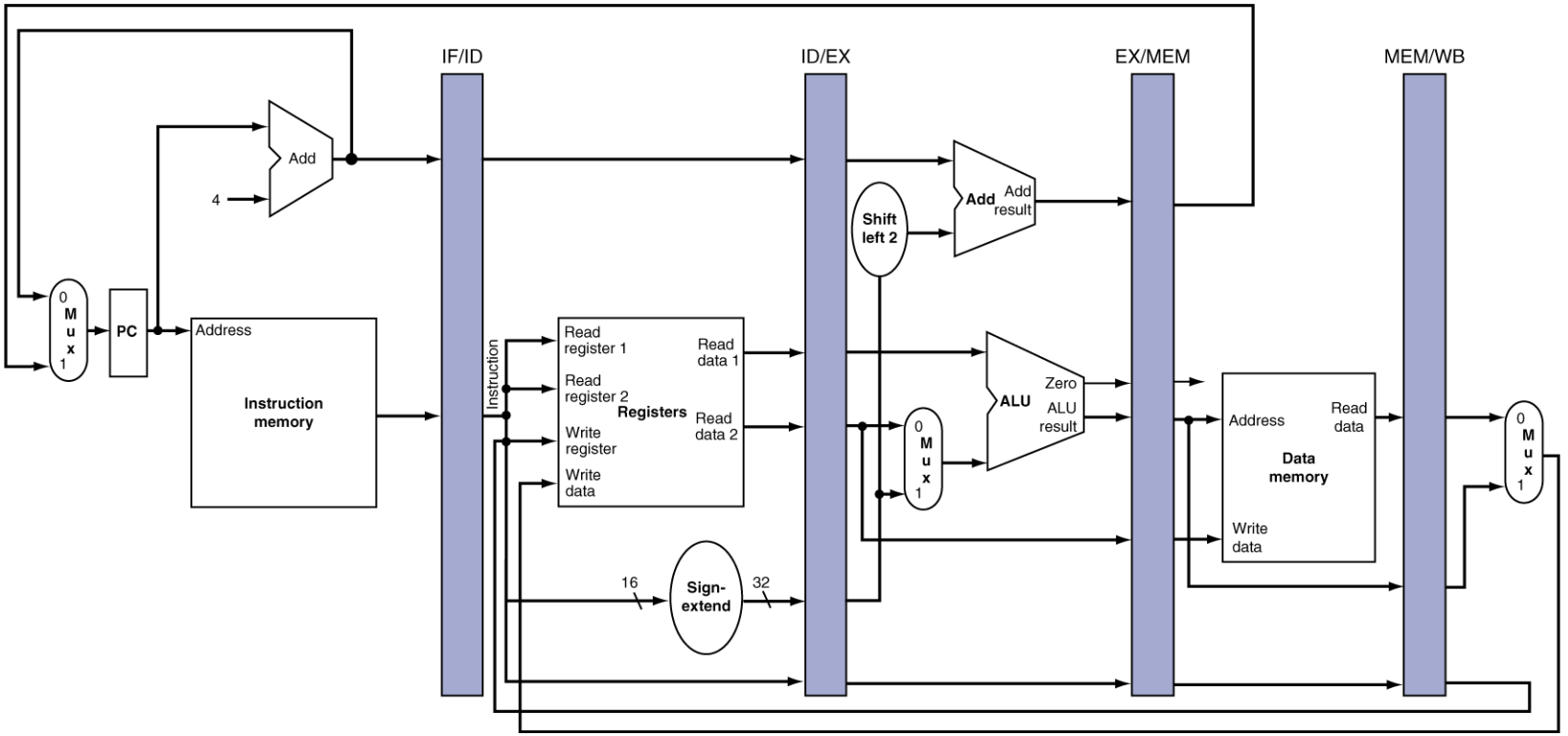
- Traditional form



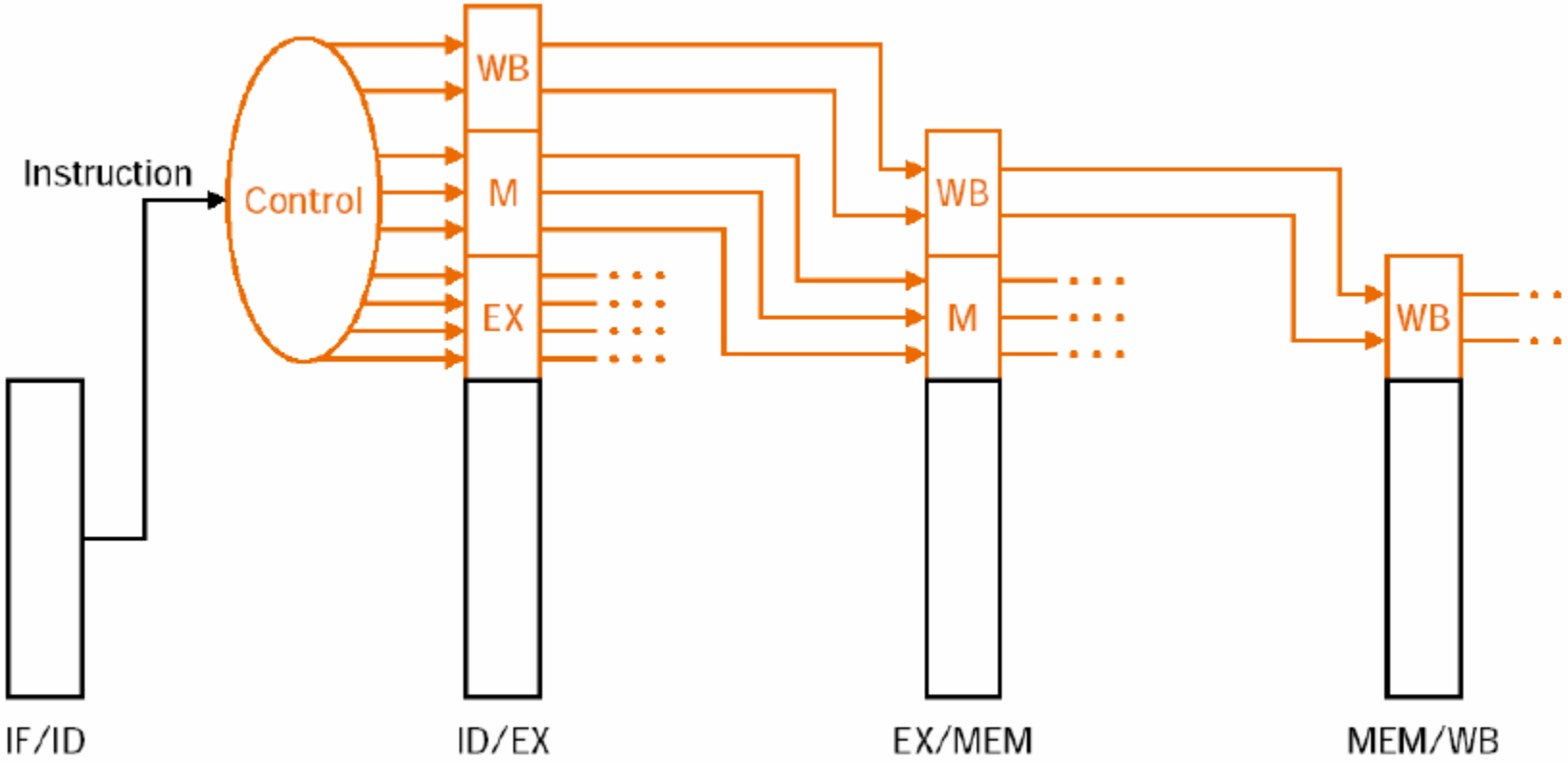
Single-Cycle Pipeline Diagram

- State of pipeline in a given cycle

add $14, $5, $6	lw $13, 24 ($1)	add $12, $3, $4	sub $11, $2, $3	lw $10, 20($1)
Instruction fetch	Instruction decode	Execution	Memory	Write-back



Pipelining Control



Pipelining Control