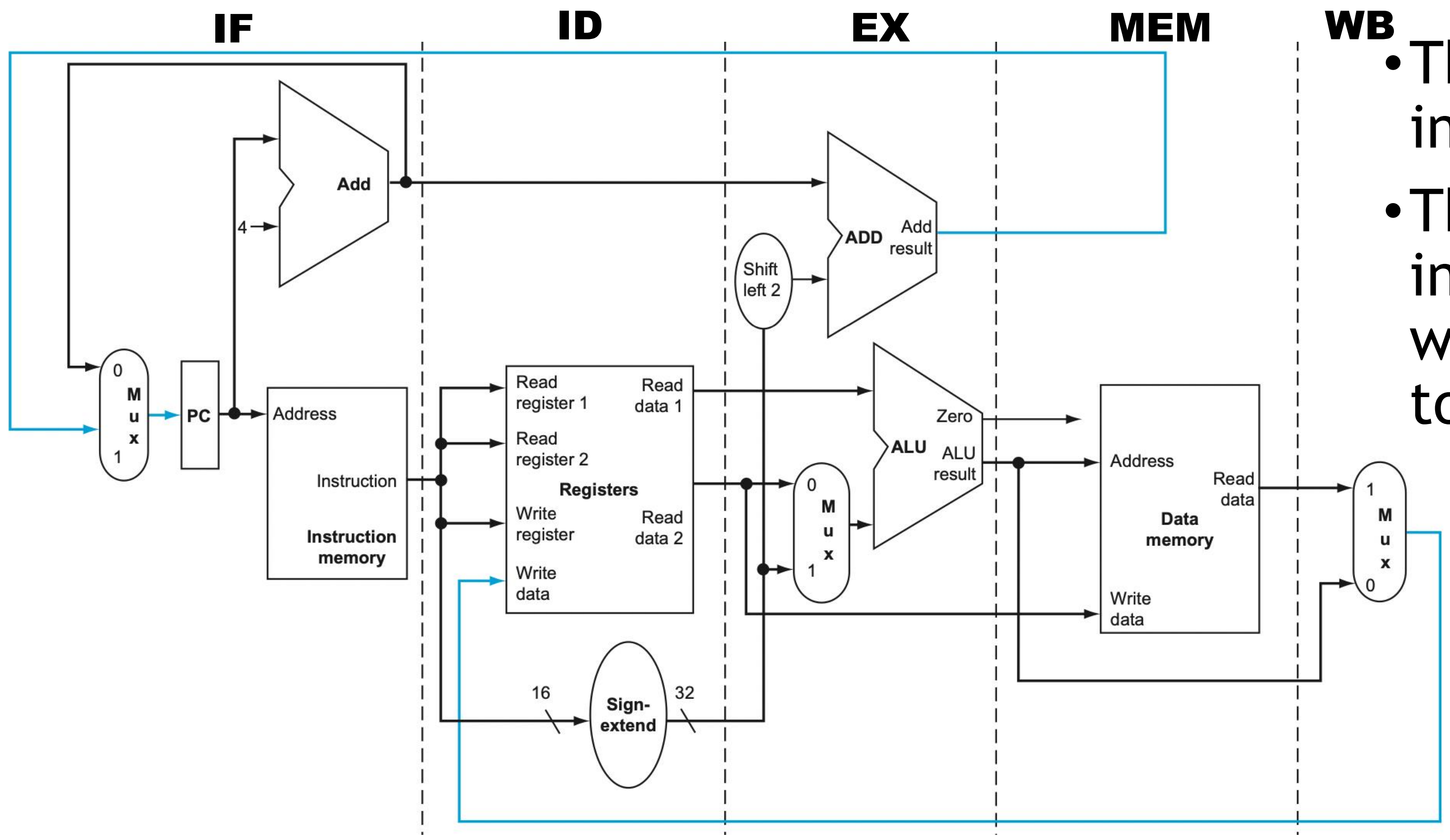
Computer Architecture
(ENE1004)

Lec – 15: The Processor (Chapter 4) - 6

Upcoming Schedule

- May 15 – Recorded video lecture
- May 21 – Assignment #1 deadline
- May 22 – Recorded video lecture

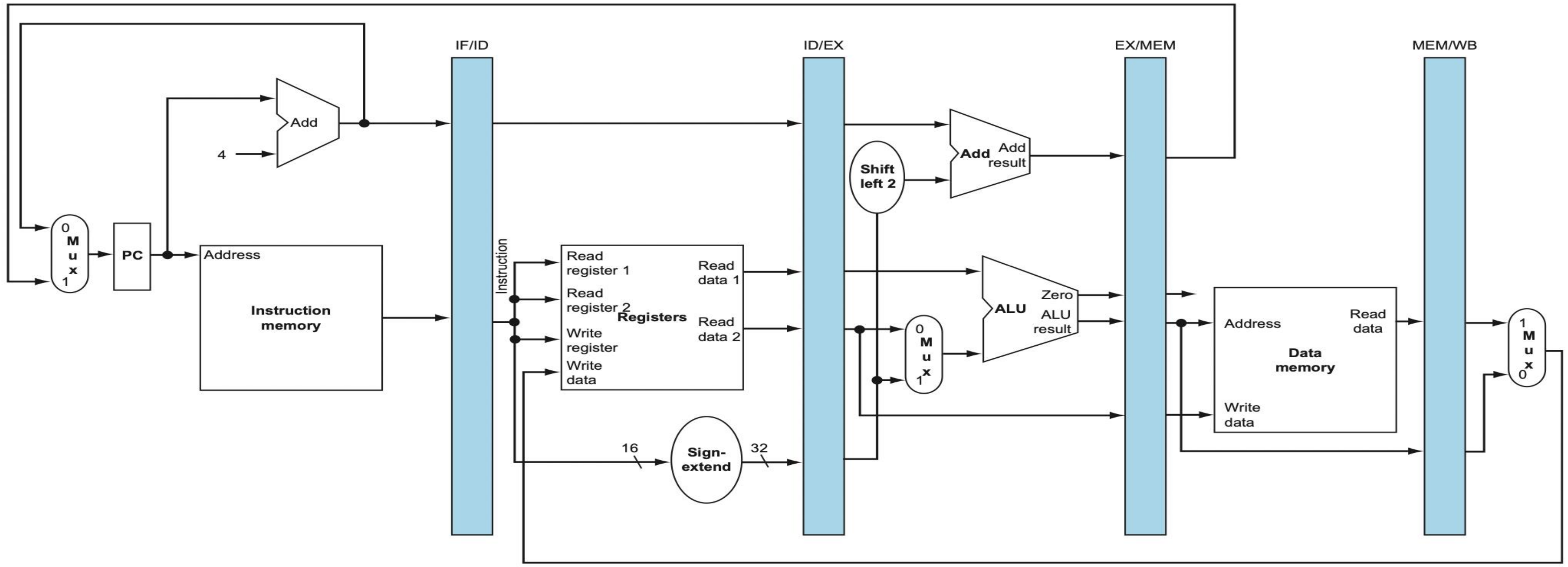
Need of Keeping Information of Instructions



- There is another serious problem in the non-pipelined datapath
- The information of an instruction in a stage is lost when the next instruction enters to the stage
  - @t, **beq** is fetched in IF
  - @t+1, **beq** moves to ID
  - @t+1, **lw** is fetched in IF
    - Here, (PC+4)+4; but, **beq** needs PC+4
  - @t+2, **beq & lw** move to EX & ID
    - Here, a sign-extended value of 10 can be given to the adder; but, **beq** needs a sign-extended value of 100
- Similar problematic situations can happen to every stage

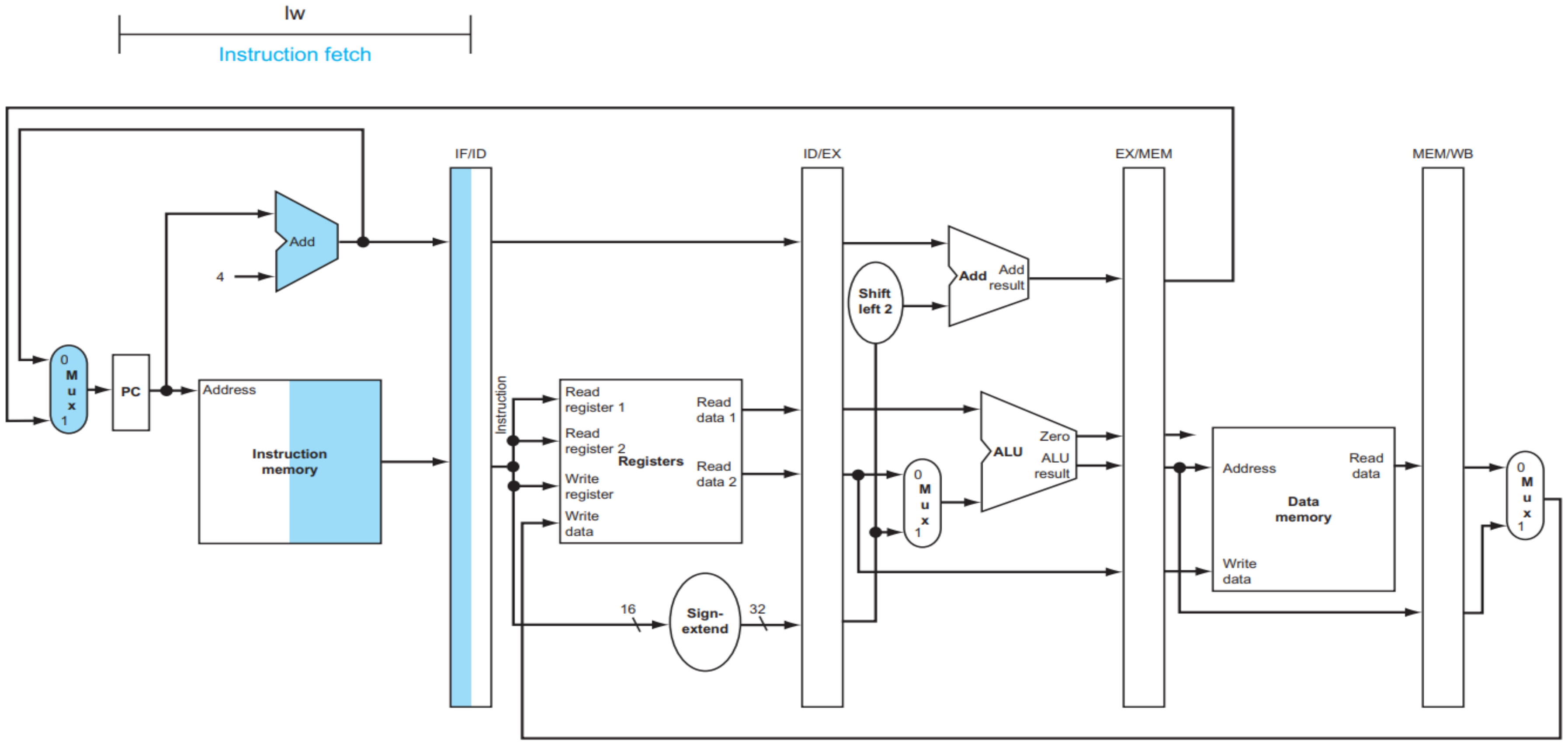
@t	**beq \$t1,\$t2,100**		
@t+1	**lw \$s1,10(\$s2)**	**beq \$t1,\$t2,100**	
@t+2	3rd instruction	**lw \$s1,10(\$s2)**	**beq \$t1,\$t2,100**

Pipelined Datapath with Pipeline Registers



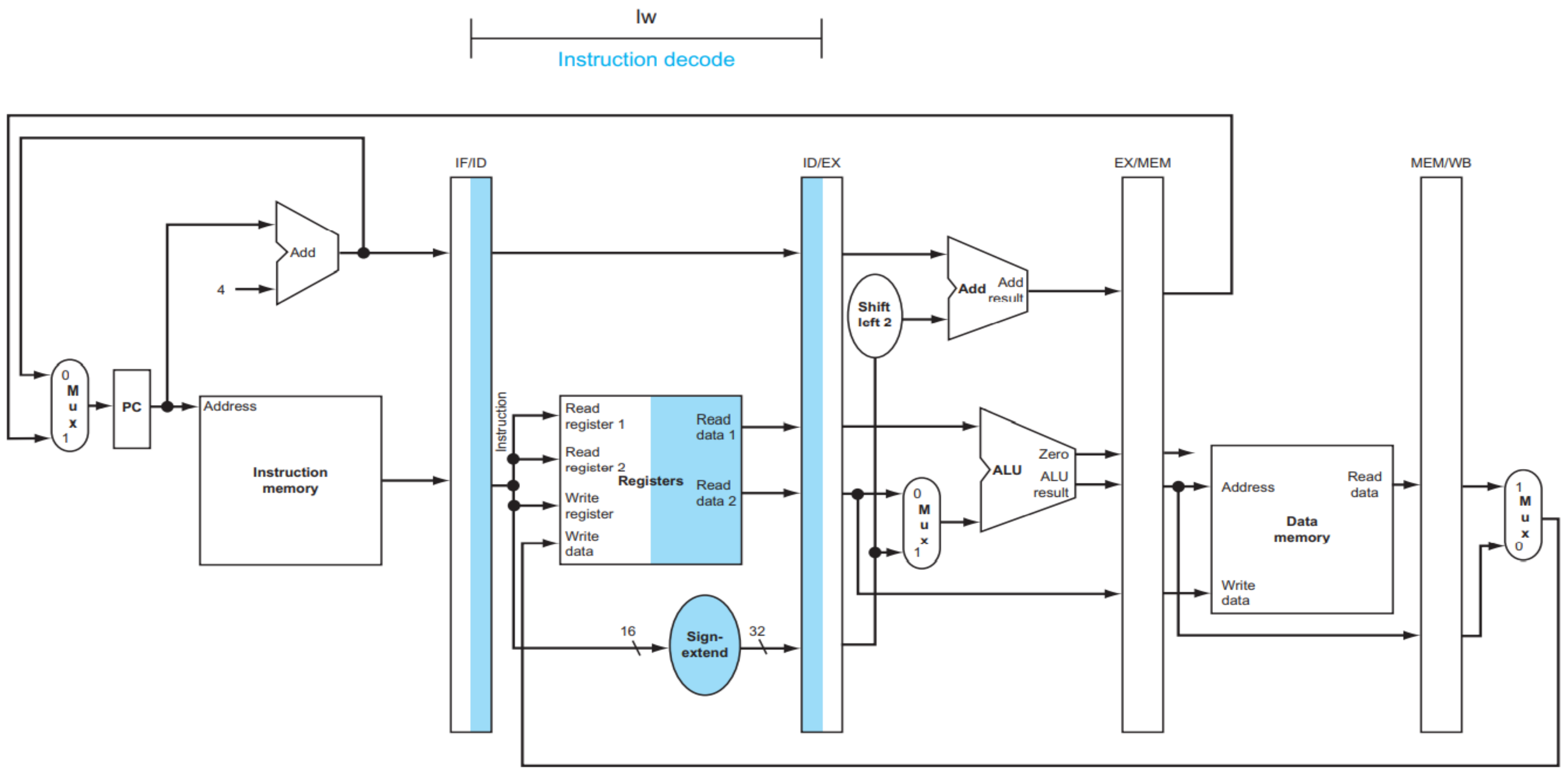
- We place (pipeline) registers between any two stages
  - In the laundry analogy, we might have a basket between any two steps to hold the clothes
  - The registers are named for the two stages separated by that register (e.g., ID/EX btw ID and EX)
- All instructions advance from one pipeline register to the next at a time
  - As an instruction advances, its required information is also copied to the next pipeline registers

lw Instruction in Pipelined Datapath – **IF (1)**



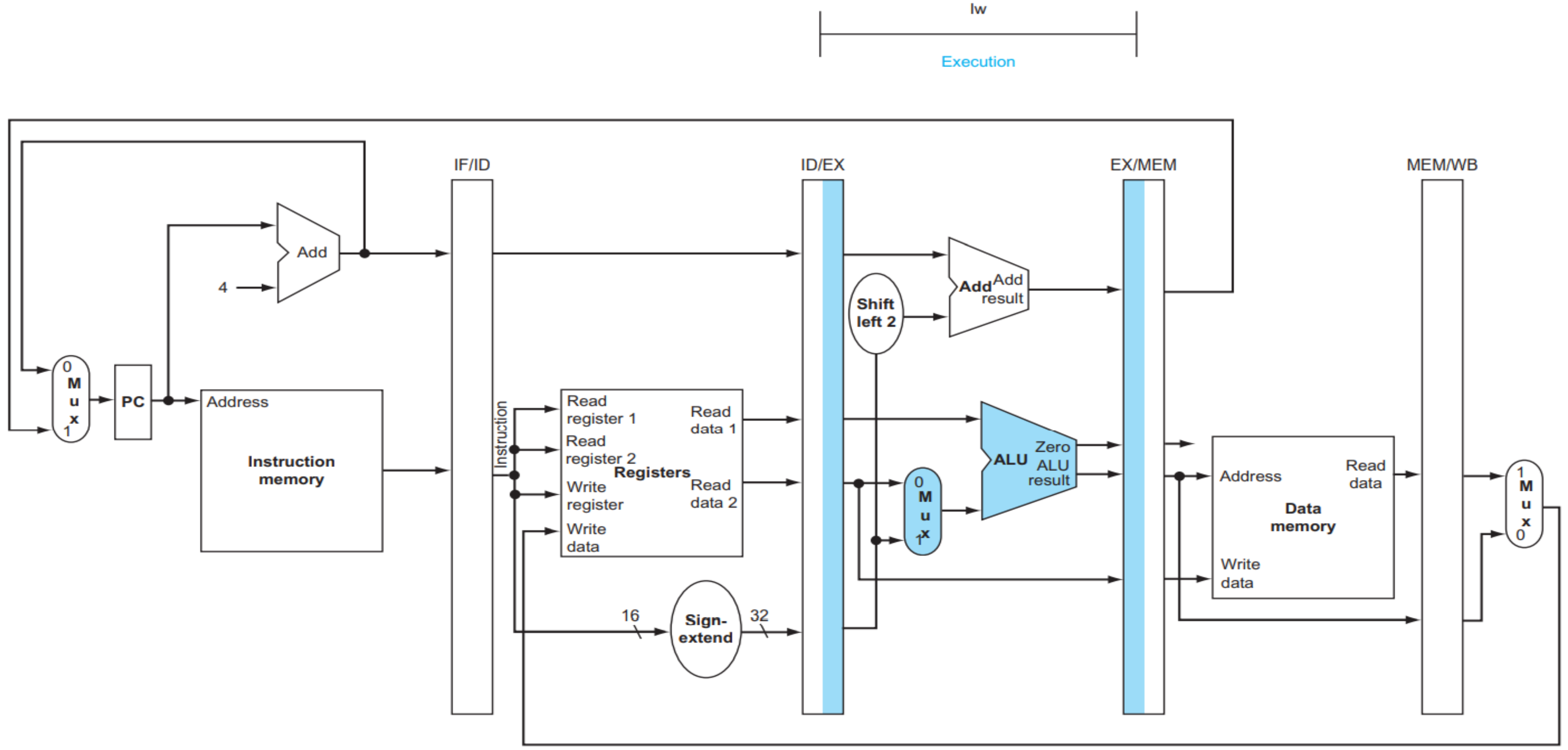
- The instruction is fetched from the instruction memory and stored in IF/ID register
- PC is incremented by 4, which is written back into the PC for the next instruction
- The incremented PC is also saved in IF/ID register in case it is needed later for (**beq**)
  - The processor cannot know what instruction is being fetched; so, it must prepare for any case

lw Instruction in Pipelined Datapath – **ID (2)**



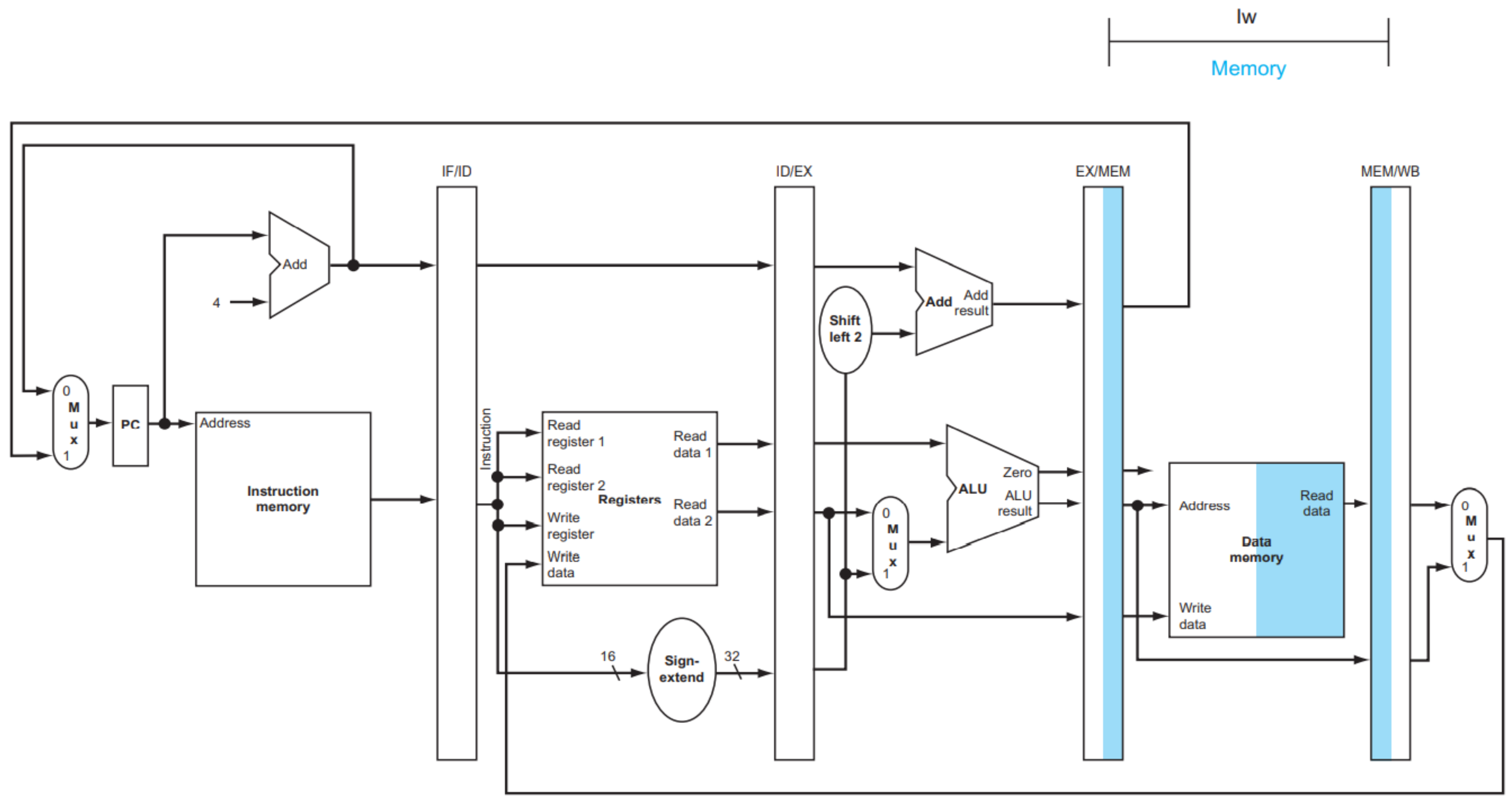
- IF/ID register supplies the instruction (now, the datapath knows that it is **lw**)
- A set of data generated in ID stage are stored in ID/EX register
  - Two 32-bit data read from the two registers
  - A sign-extended 32-bit immediate value
  - The incremented PC in IF/ID

lw Instruction in Pipelined Datapath – **EX (3)**



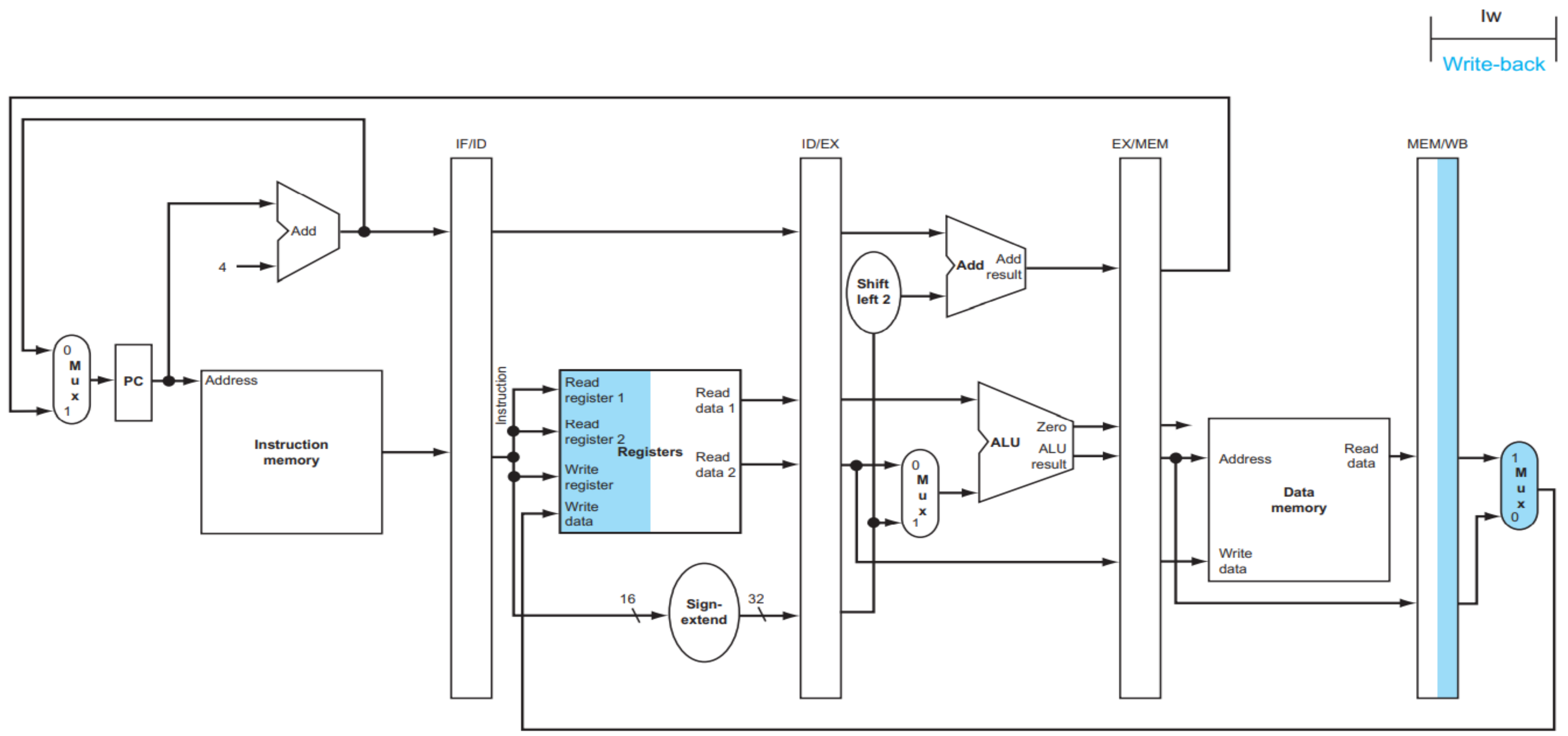
- (i) The value of register 1 and (ii) the sign-extended immediate value are read from ID/EX register
- (i) and (ii) are added using the ALU to compute the target data memory address
- The computed target address is stored in EX/MEM register

lw Instruction in Pipelined Datapath – **MEM (4)**



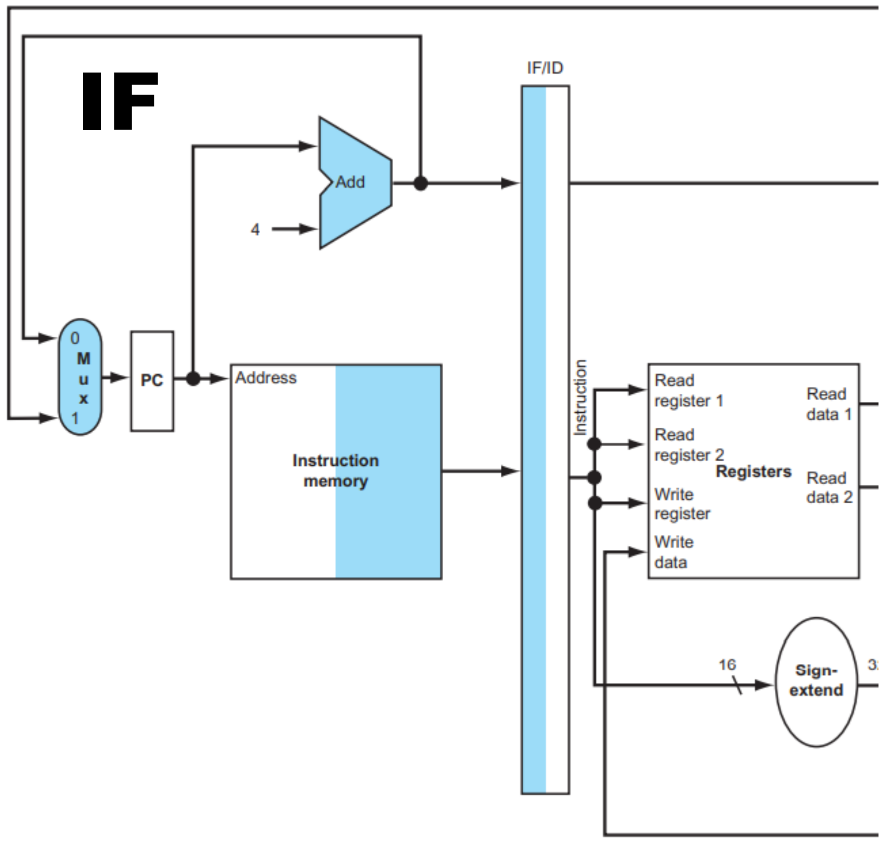
- Using the address from EX/MEM register, data is read from the data memory
- The read data is stored in MEM/WB register

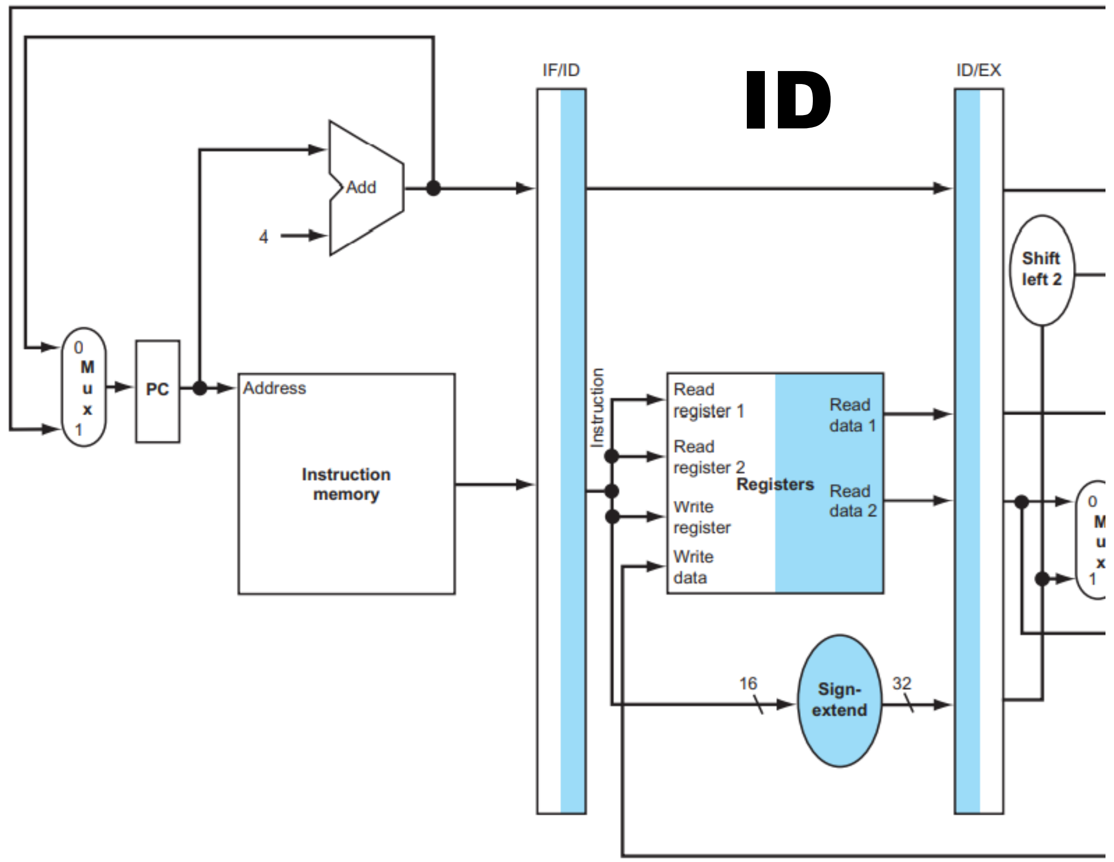
lw Instruction in Pipelined Datapath – **WB (5)**



- The data is read from MEM/WB register
- The data is written into the register file

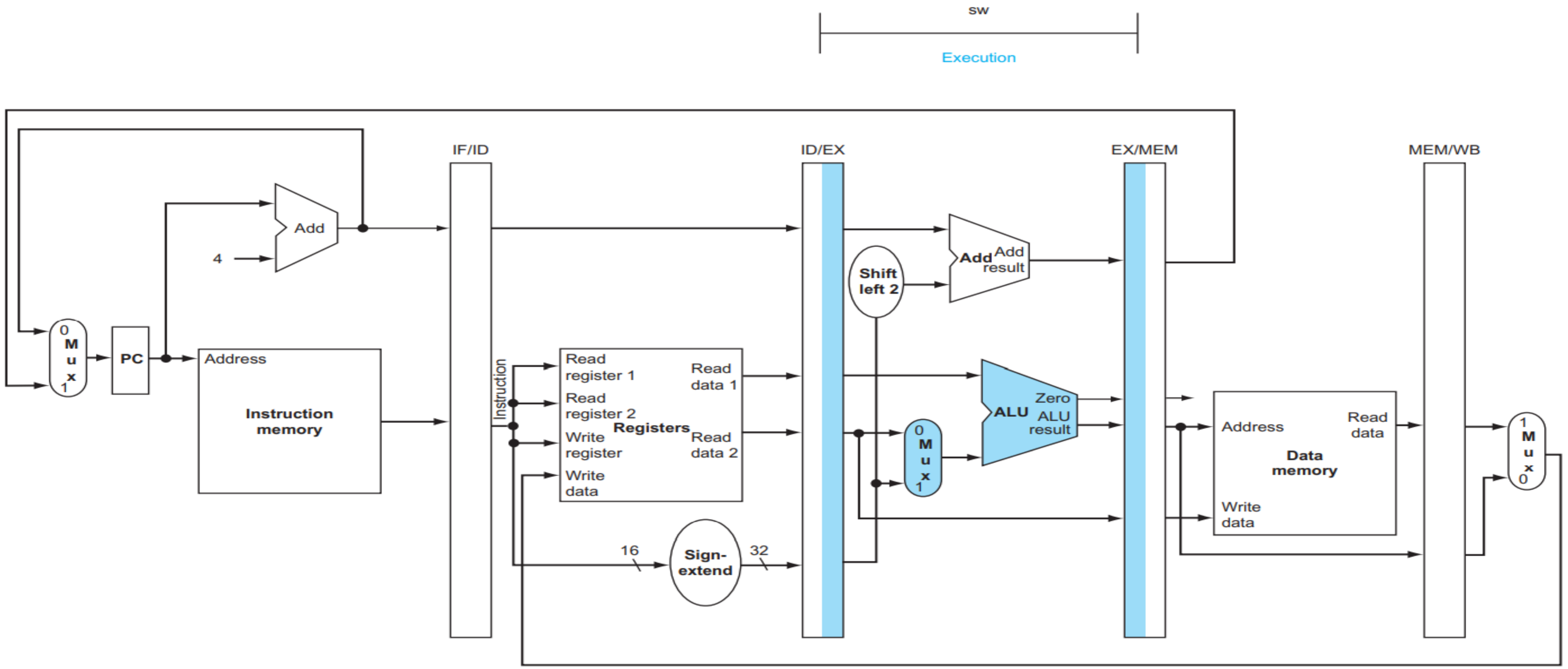
sw Instruction in Pipelined Datapath – **IF/ID (1/2)**





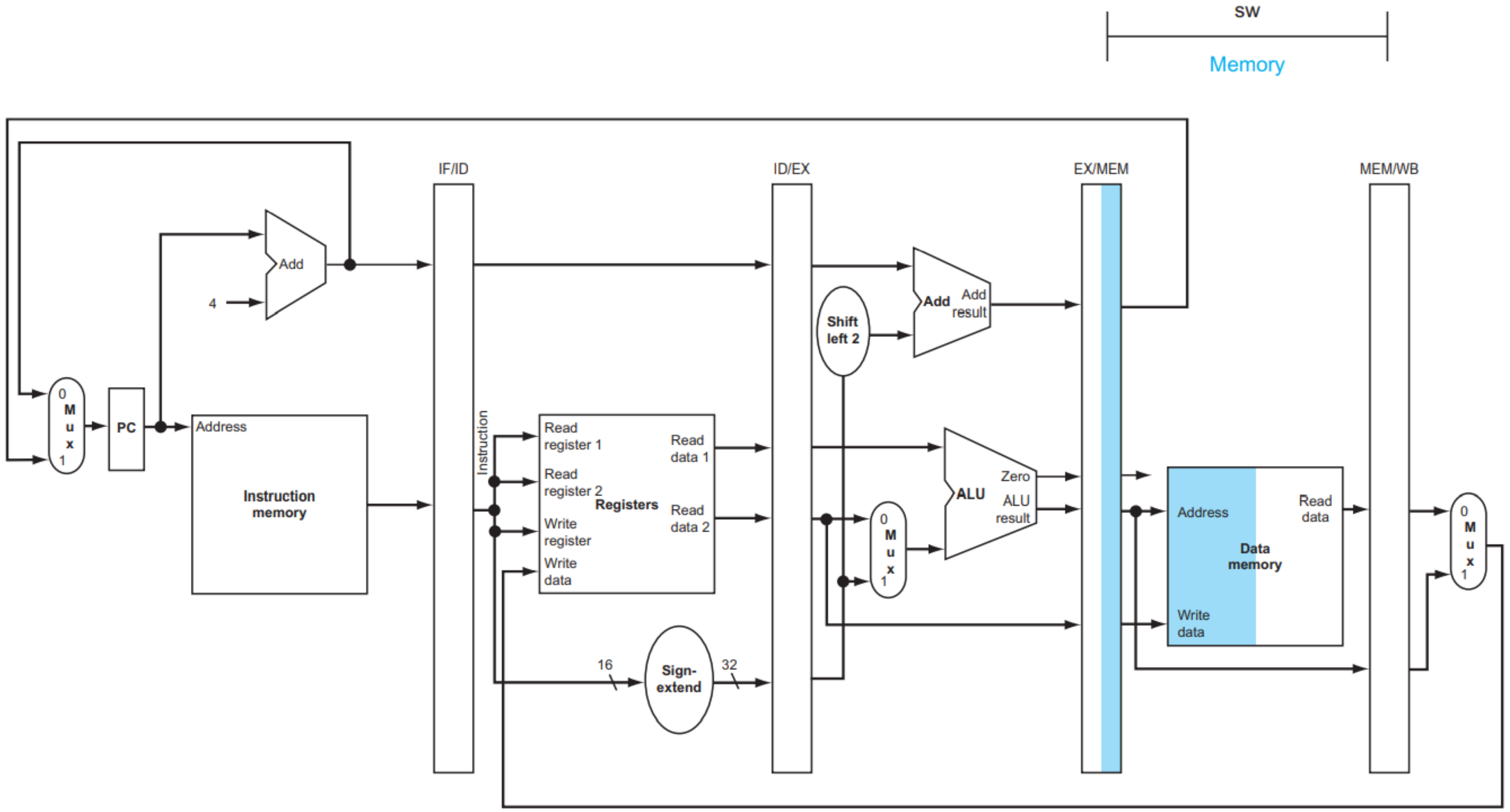
- The first two (IF/ID) stages are common in all the instructions
  - In ID stage, the datapath knows the fetched instruction
- IF stage: the fetched instruction & the incremented PC are stored in IF/ID register
- ID stage: two register values & sign-extended value are stored in IF/ID register

sw Instruction in Pipelined Datapath – **EX (3)**



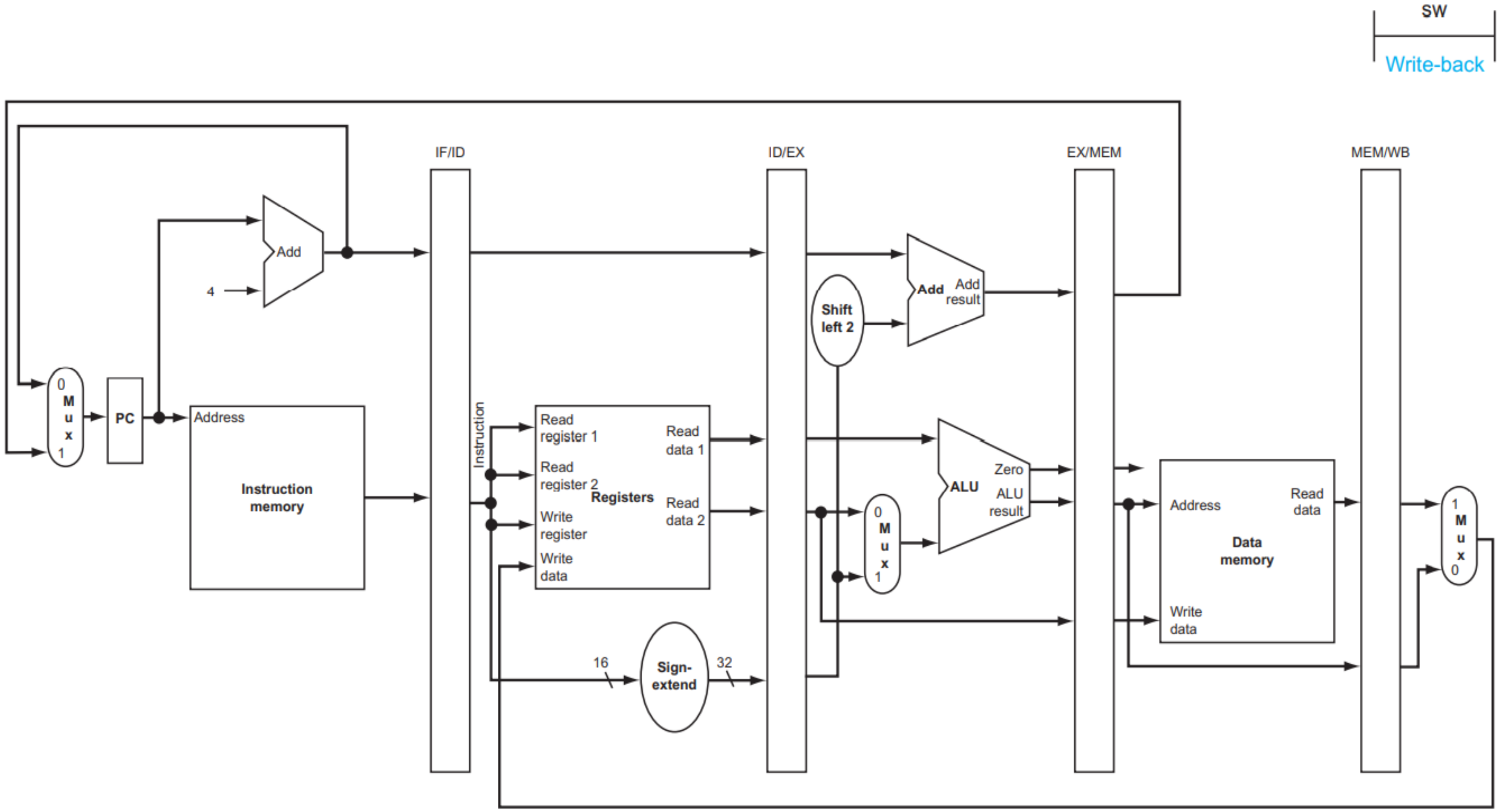
- (i) The register 1 value and (ii) the sign-extended value are read from ID/EX register, which are added using the ALU for the memory address
- The computed memory address is stored to EX/MEM register
- The register 2 value, the data to be written into the data memory, is stored to EX/MEM register

sw Instruction in Pipelined Datapath – **MEM (4)**



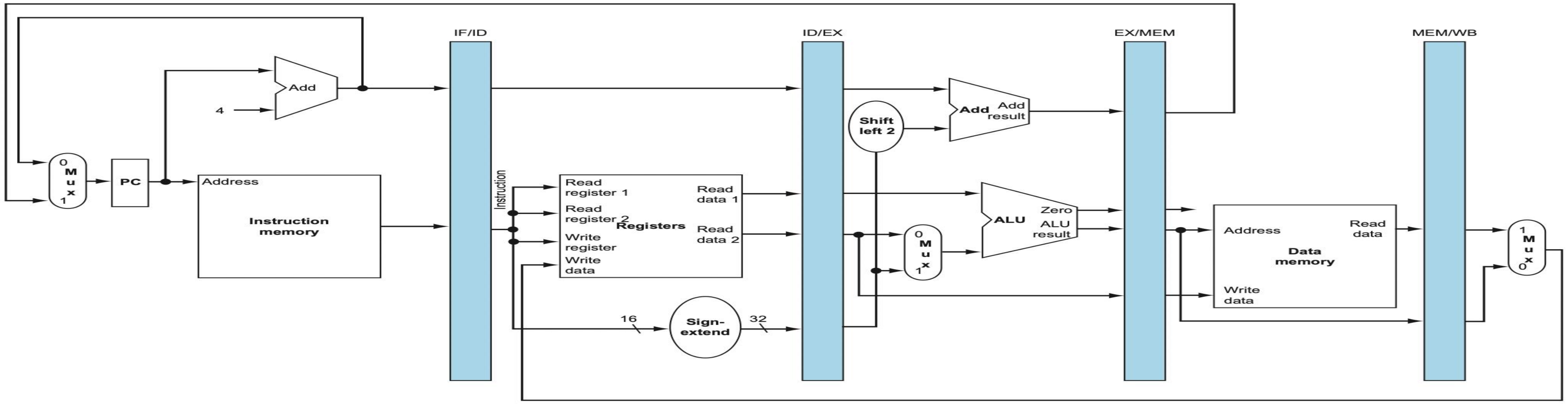
- The address and the data (register 2 value) are read from EX/MEM register
- The data is written into the data memory at the address

sw Instruction in Pipelined Datapath – **WB (5)**



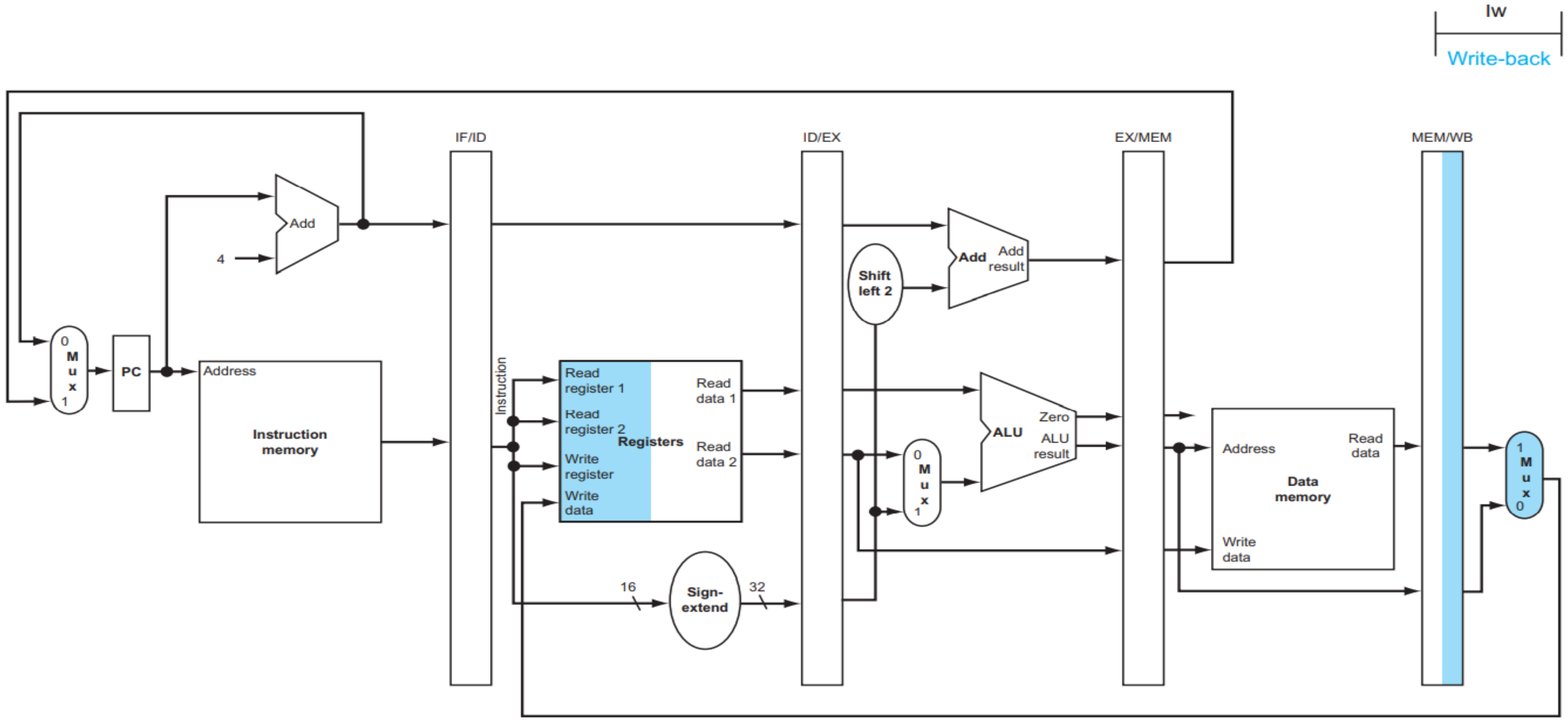
- There is nothing left to complete the store instruction; so, nothing happens in WB stage
  - In MEM stage of **sw** instruction, the data is written into the data memory
- An instruction passes through a stage even if there is nothing to do in it, because all the following instructions are already in progress at the maximum rate

Two Key Points in Pipelined Datapath



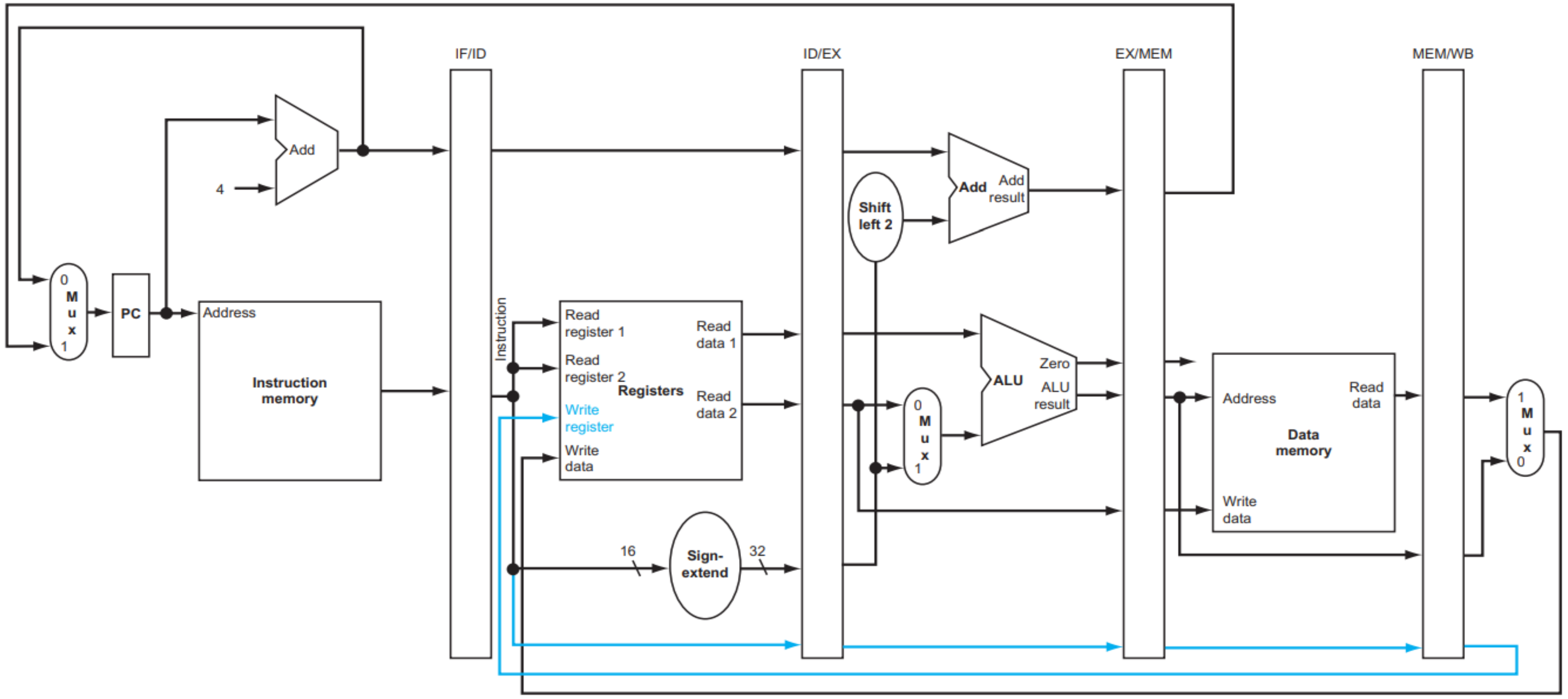
- (1) To pass something from a stage to another, it must be placed in a pipeline register
  - Otherwise, the data or the information is lost when the next instruction enters that stage
  - For **sw**, the register 2 value obtained from ID is passed to EX (through ID/EX register) and again is passed to MEM (through EX/MEM register)
- (2) Each component - instruction memory, register read ports, ALU, data memory, register write port - can be used only within a single stage
  - Hardware cannot be shared by multiple instructions simultaneously
  - Register file has separate ports for read/write; so, it can be considered as different components

A Corrected Pipelined Datapath: Bug



- In WB stage of **lw** instruction, the read data should be written into the write register
- Then, at the moment, who supplies the write register number?
  - It is the instruction now in the ID stage, which is a later instruction of the load instruction
- We need to preserve the write register number of the load instruction till it reaches WB stage

A Corrected Pipelined Datapath: Fix



- To fix the bug, we need to pass the write register number from ID stage to WB stage
  - The write register number is stored in ID/EX register in the ID stage
  - It is copied to EX/MEM register in the EX stage
  - It is copied to MEM/WB register in the MEM stage
- Then, it can be read from MEM/WB register and used in WB stage