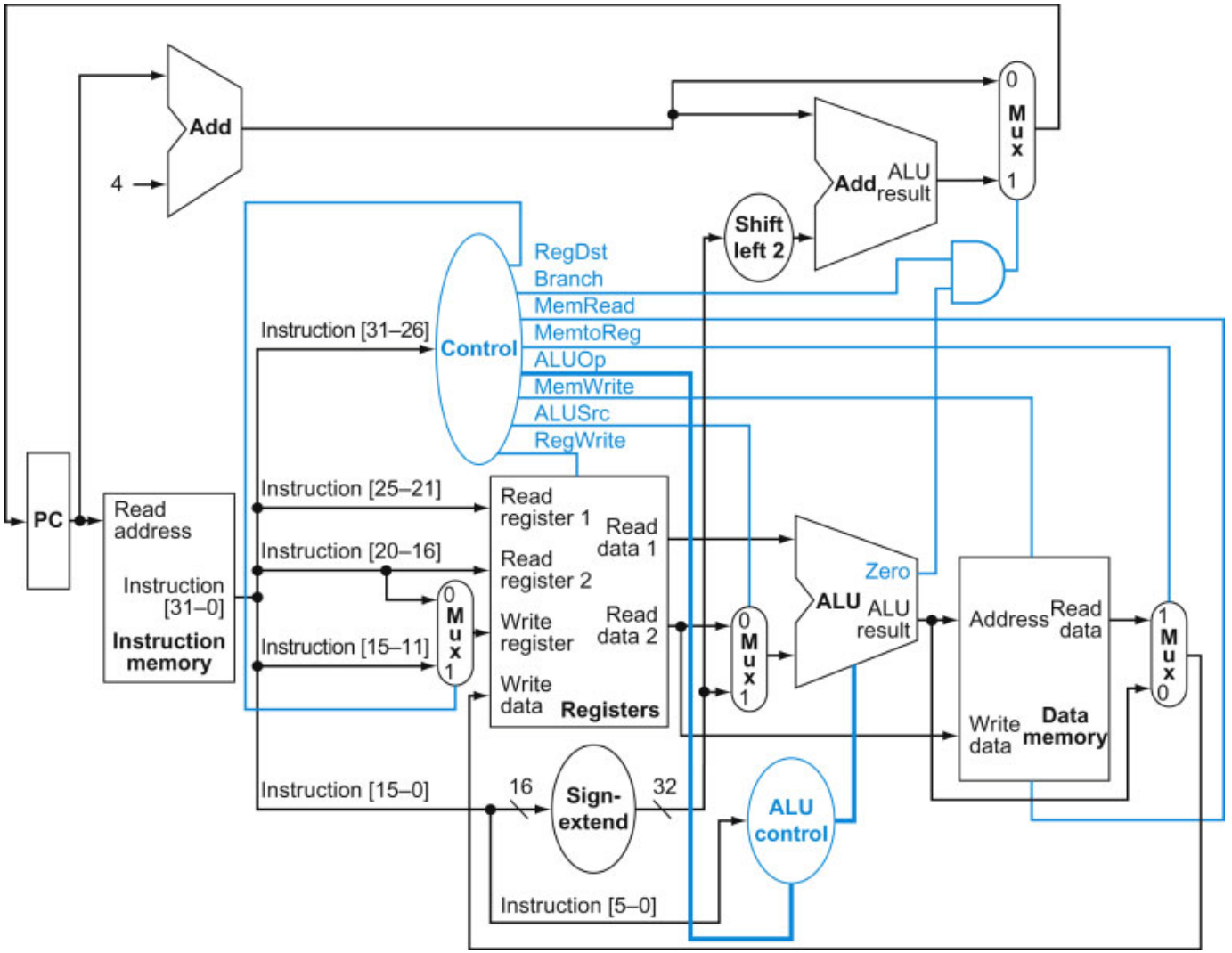
# Computer Architecture

**Dr. Haroon Mahmood**

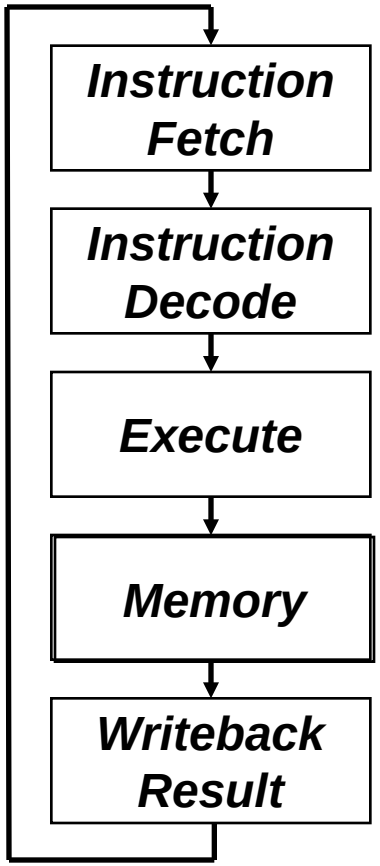
**Assistant Professor**

**NUCES Lahore**

# Top View of MIPS architecture



# Instruction Execution Cycle




Obtain instruction from program storage

Determine required actions and obtain operand data

Locate and obtain operand data

Load or Store values in memory

Deposit results in register for later use

# 1st and 2nd Instruction cycles

- **Instruction fetch (IF)**

  IR ← Mem[PC];

  NPC ← PC + 4

- **Instruction decode & register fetch (ID)**

  A ← Regs[$IR_{21..25}$];

  B ← Regs[$IR_{16..20}$];

  Imm ← (($IR_{15}$)$^{16}$ # # $IR_{0..15}$)

# 3rd Instruction cycle

- **Execution & effective address (EX)**
  - **Memory reference**
    - **ALUOutput ← A + Imm**
  - **Register - Register ALU instruction**
    - **ALUOutput ← A *func* B**
  - **Register - Immediate ALU instruction**
    - **ALUOutput ← A *op* Imm**
  - **Branch**
    - **AddOutput ← NPC + Imm; Cond ← (A *op* B)**

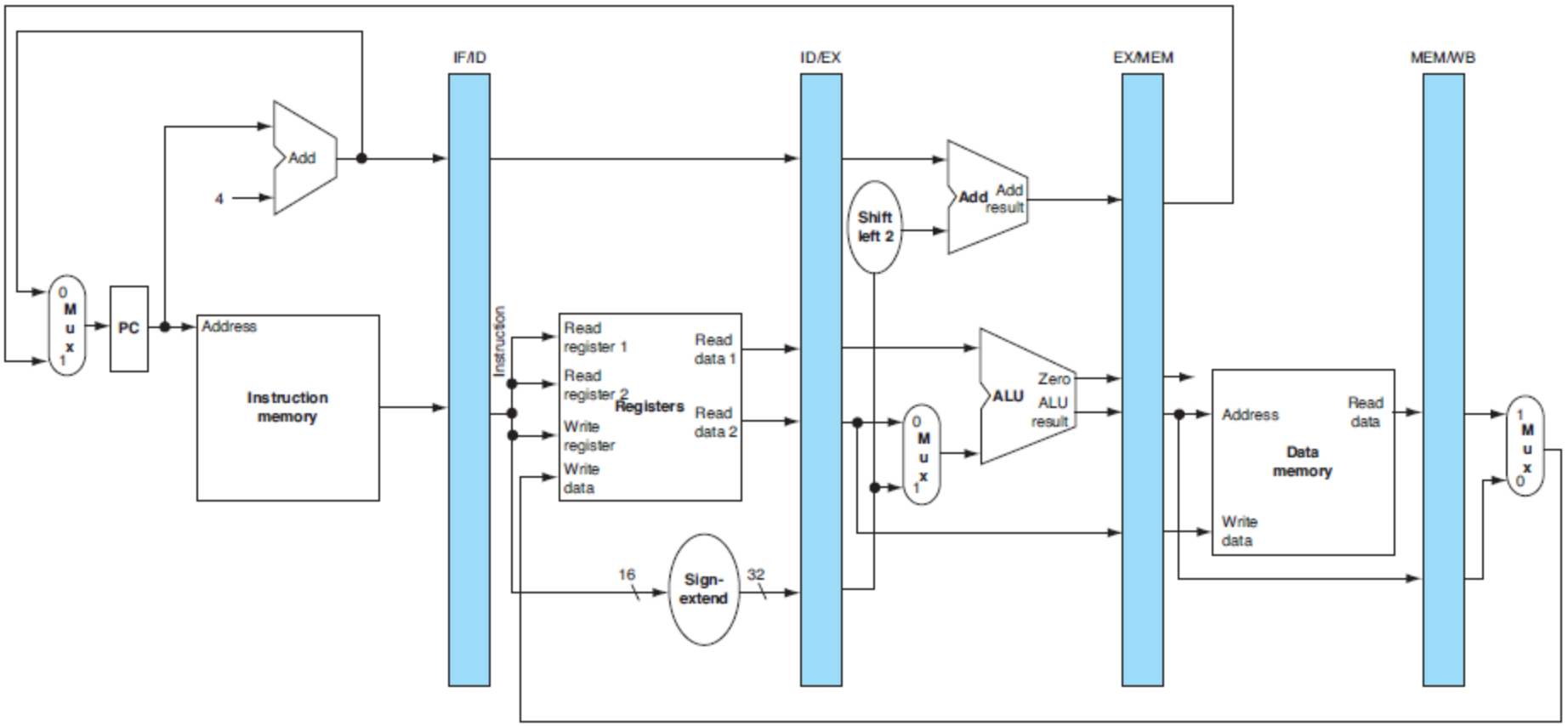
# 4th Instruction cycle

- **Memory access & branch completion (MEM)**
  - **Memory reference**
    - **PC ← NPC**
    - **LMD ← Mem[ALUOutput]  (load)**
    - **Mem[ALUOutput] ← B  (store)**
  - **Branch**
    - **if (cond) PC ← AddOutput; else PC ← NPC**

# 5th Instruction cycle

- **Write-back (WB)**
  - **Register - register ALU instruction**
    - **Regs[$IR_{11..15}$] ⟵ ALUOutput**
  - **Register - immediate ALU instruction**
    - **Regs[$IR_{16..20}$] ⟵ ALUOutput**
  - **Load instruction**
    - **Regs[$IR_{16..20}$] ⟵ LMD**

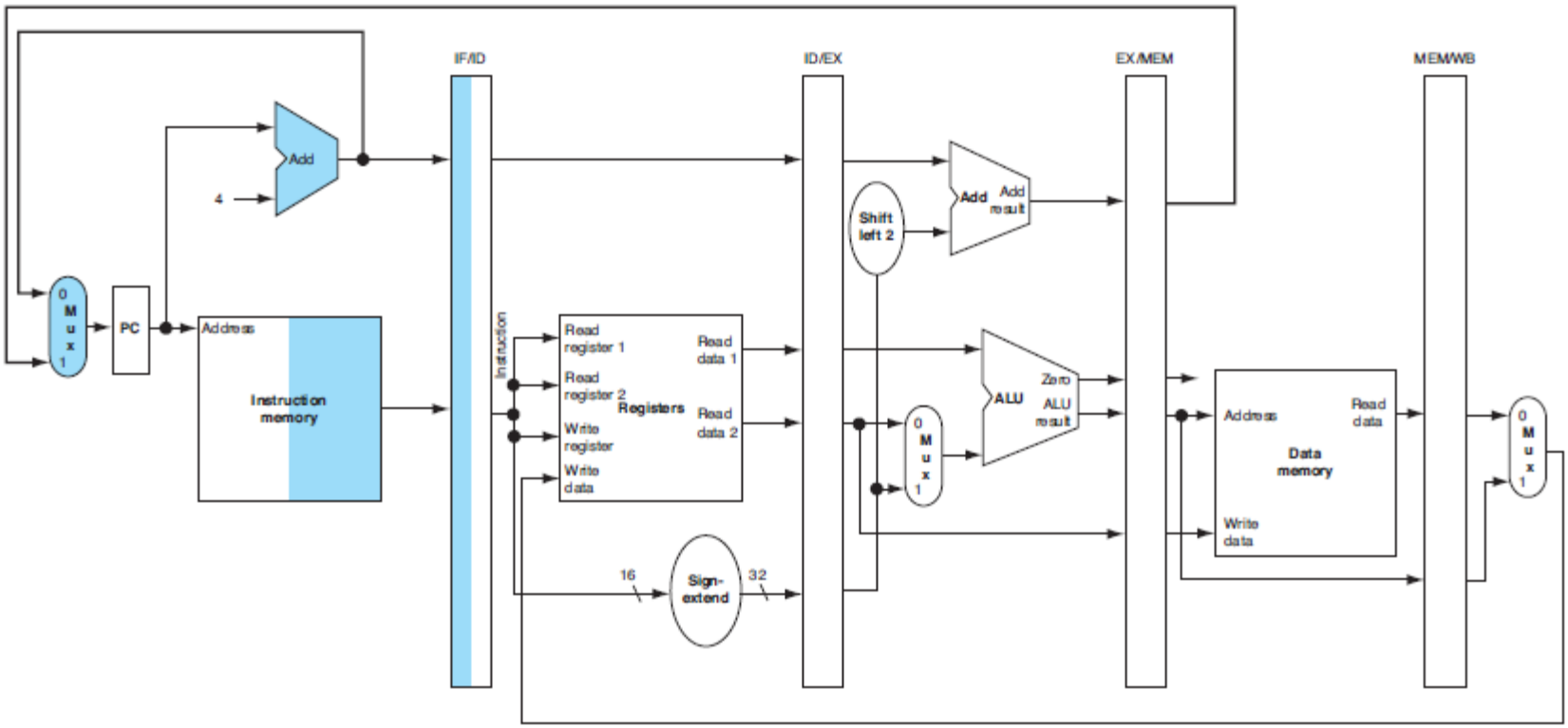
# Adding registers to the pipeline



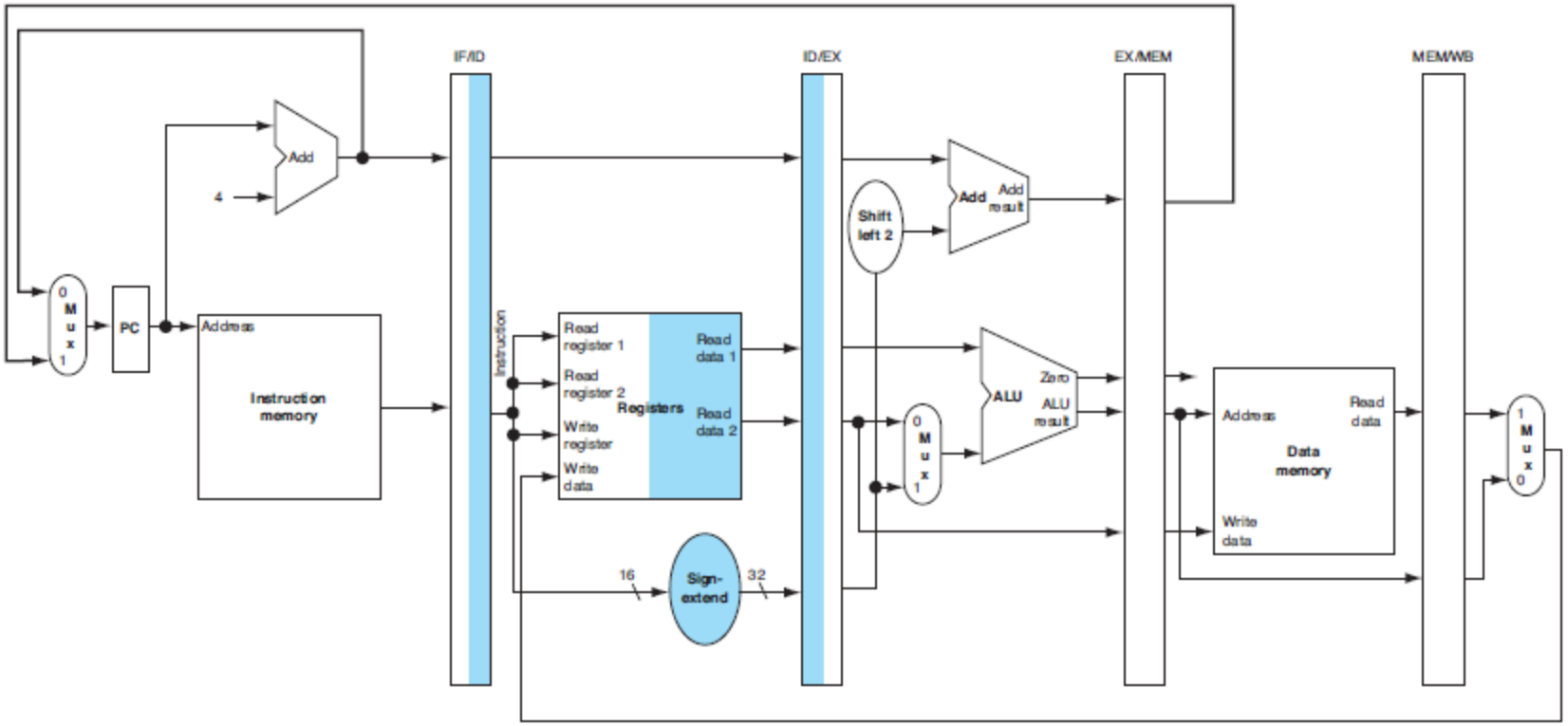
# Adding registers to the pipeline

- **No register at the write back stage**
- **PC as a pipeline register**
- **Registers or memory are read in second half of the cycle and written in the first half of the cycle**
- **In the figures below, we highlight the *right half of registers or memory when* they are being *read and highlight the left half when they are being written.***

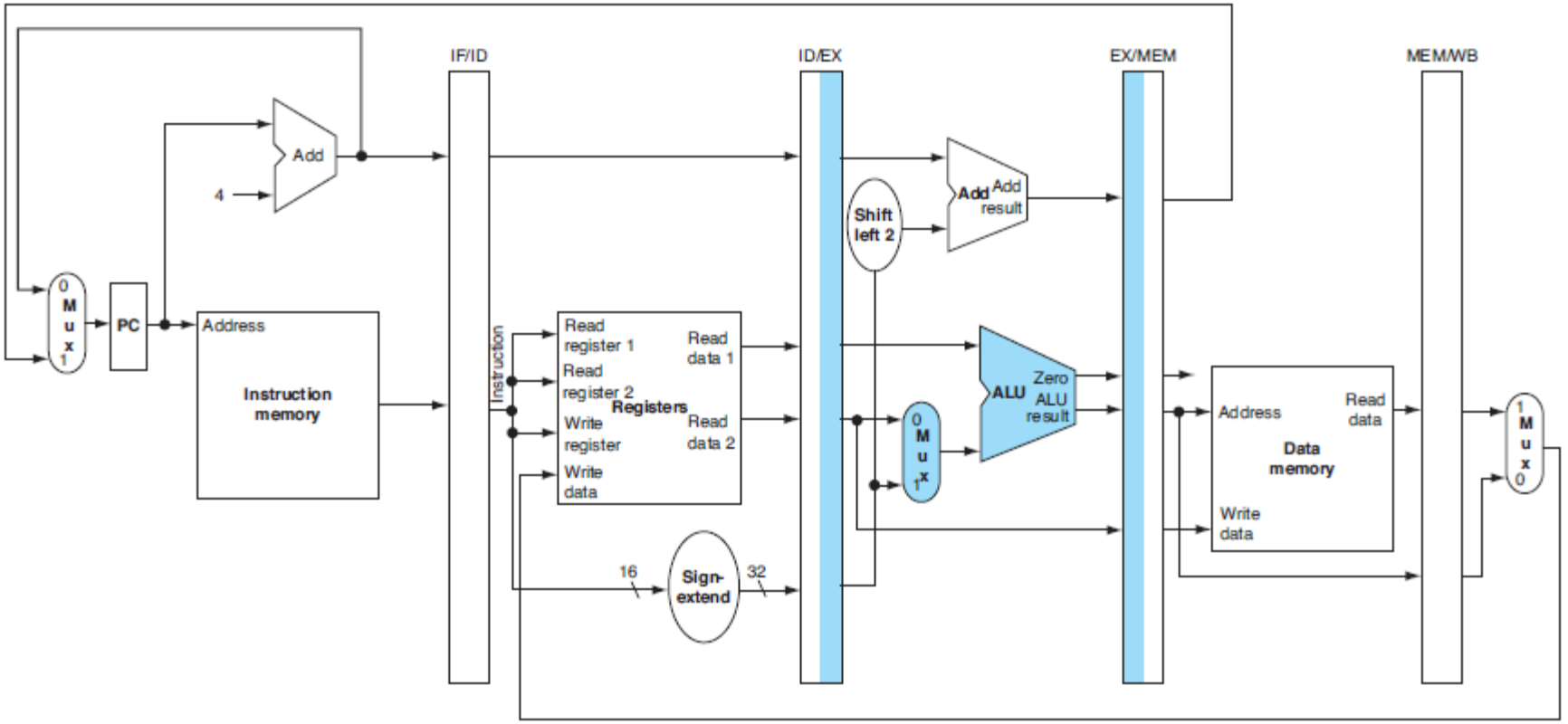
# LW instruction in pipelining



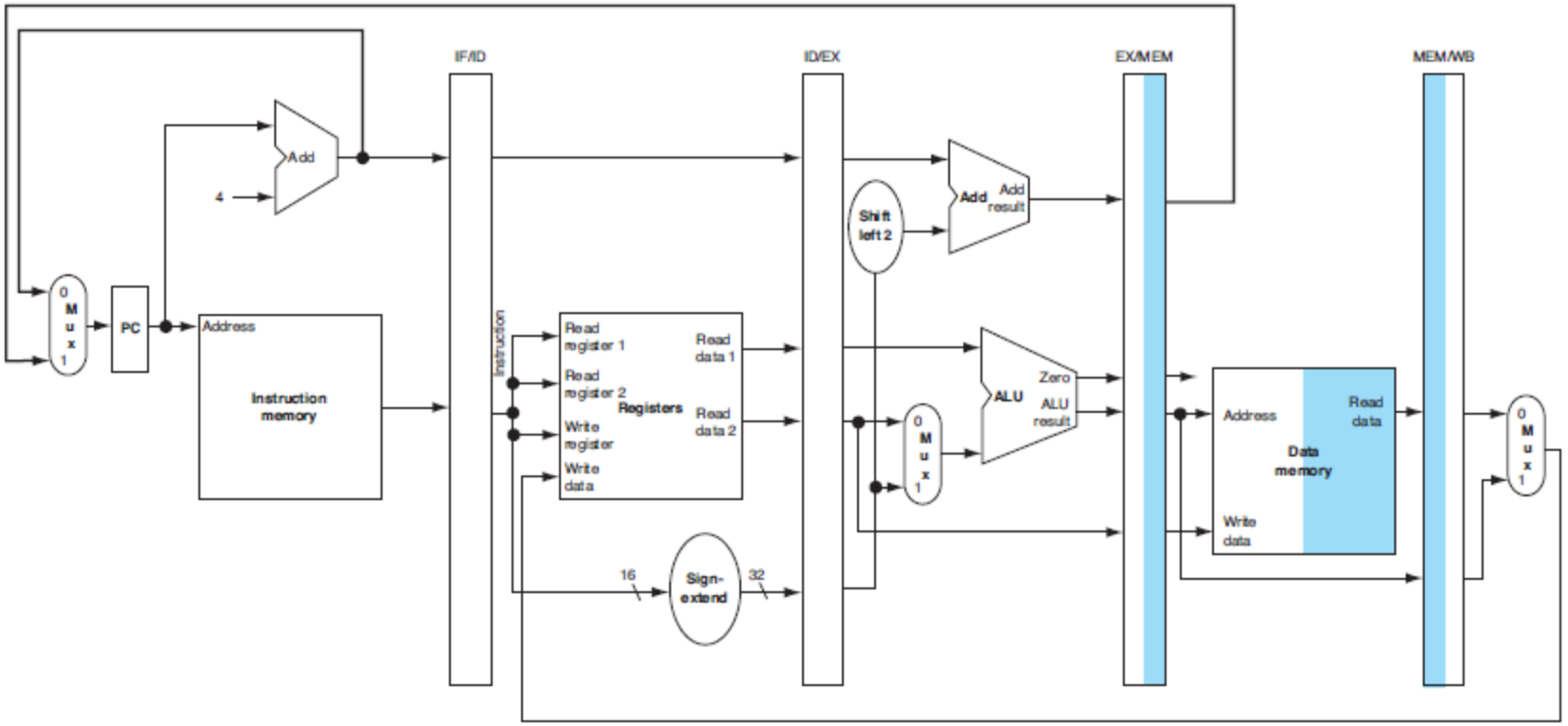
# LW instruction in pipelining



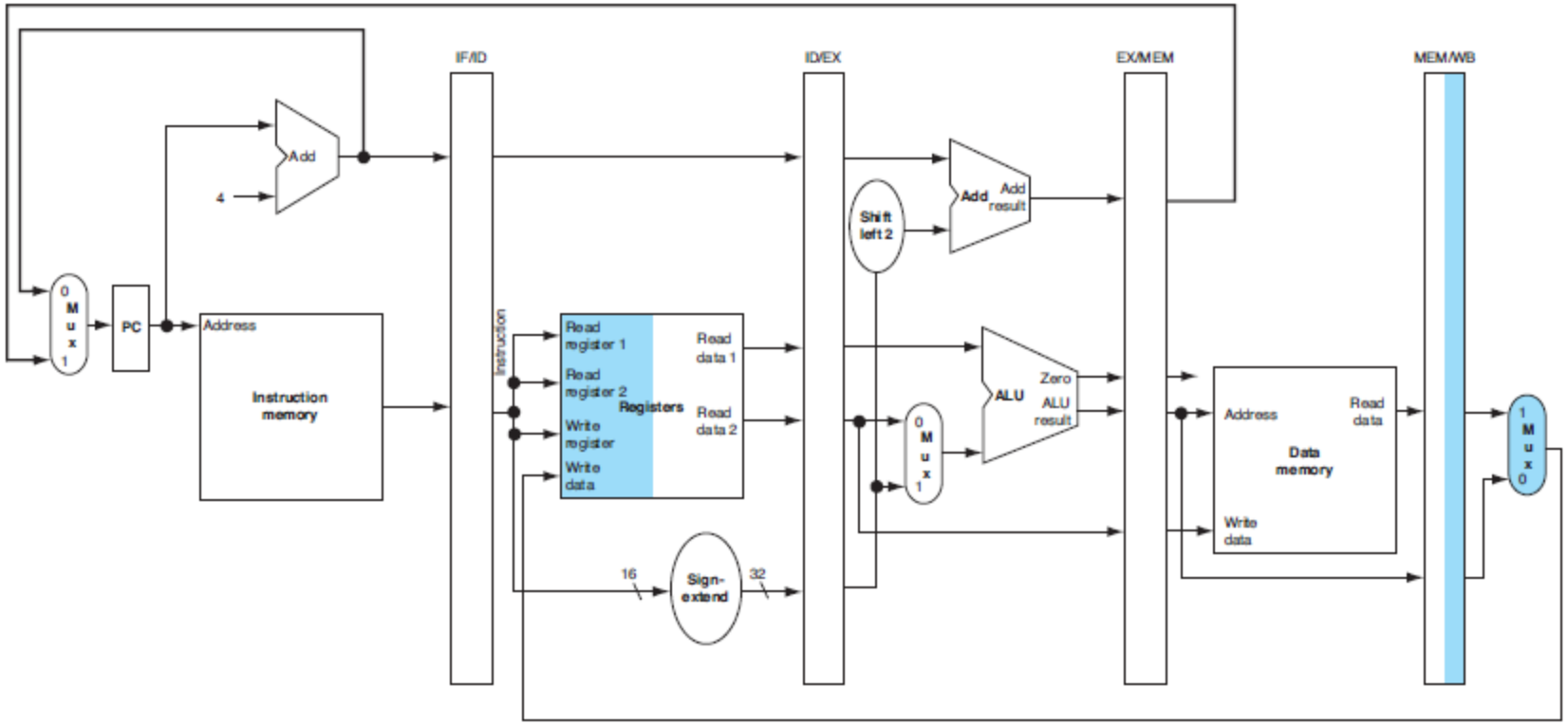
# LW instruction in pipelining



# LW instruction in pipelining

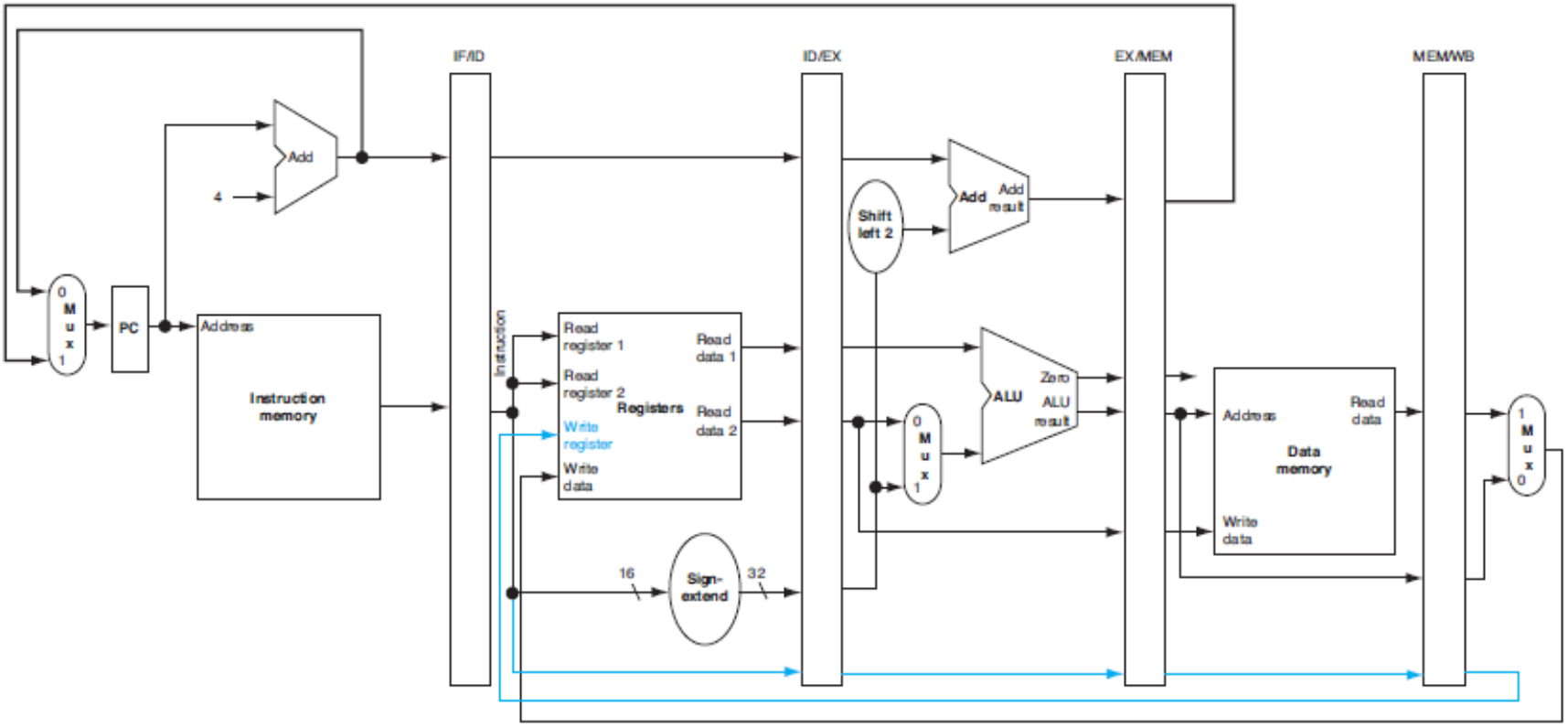


# LW instruction in pipelining

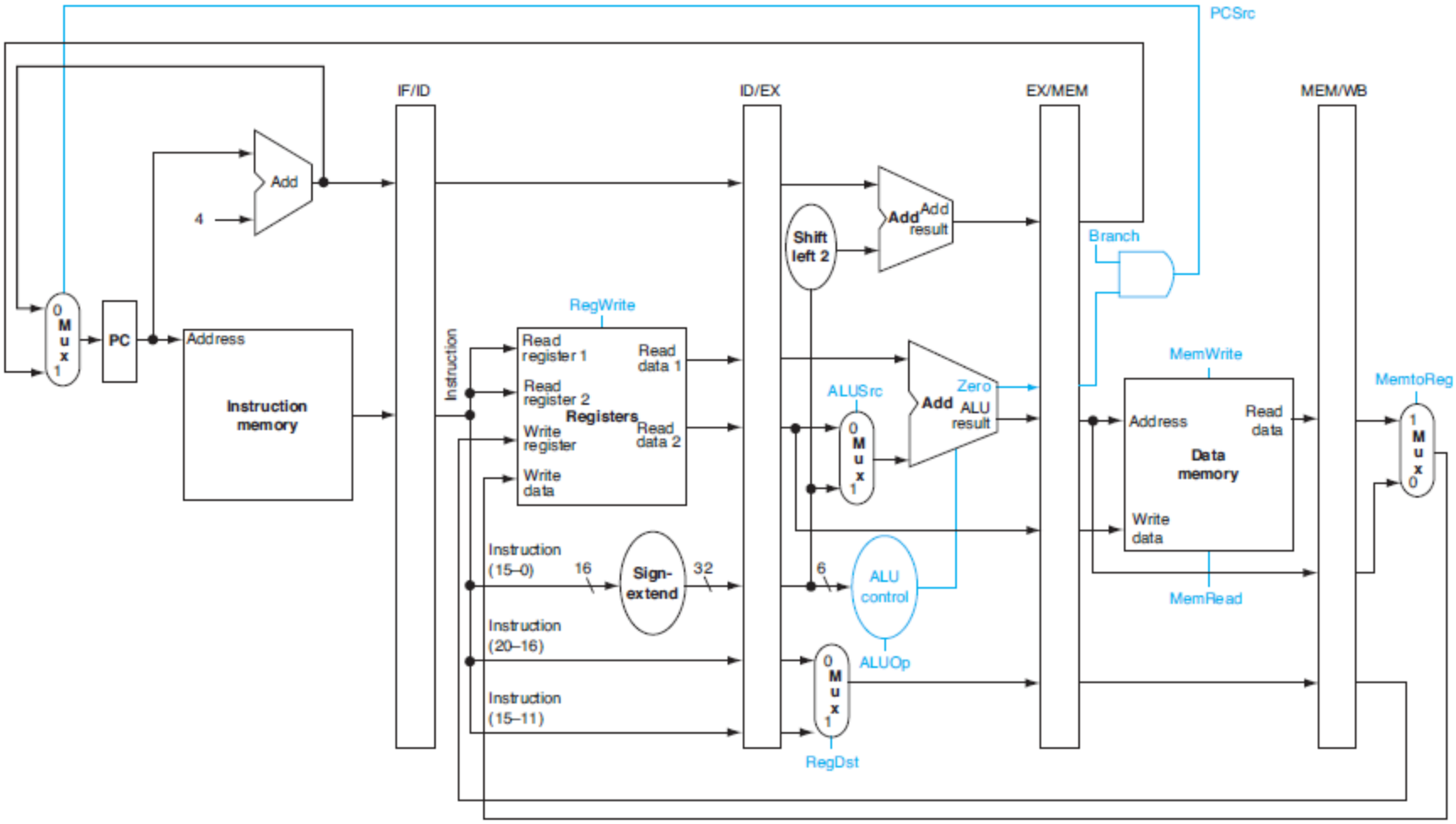


# Registers – Values Saved

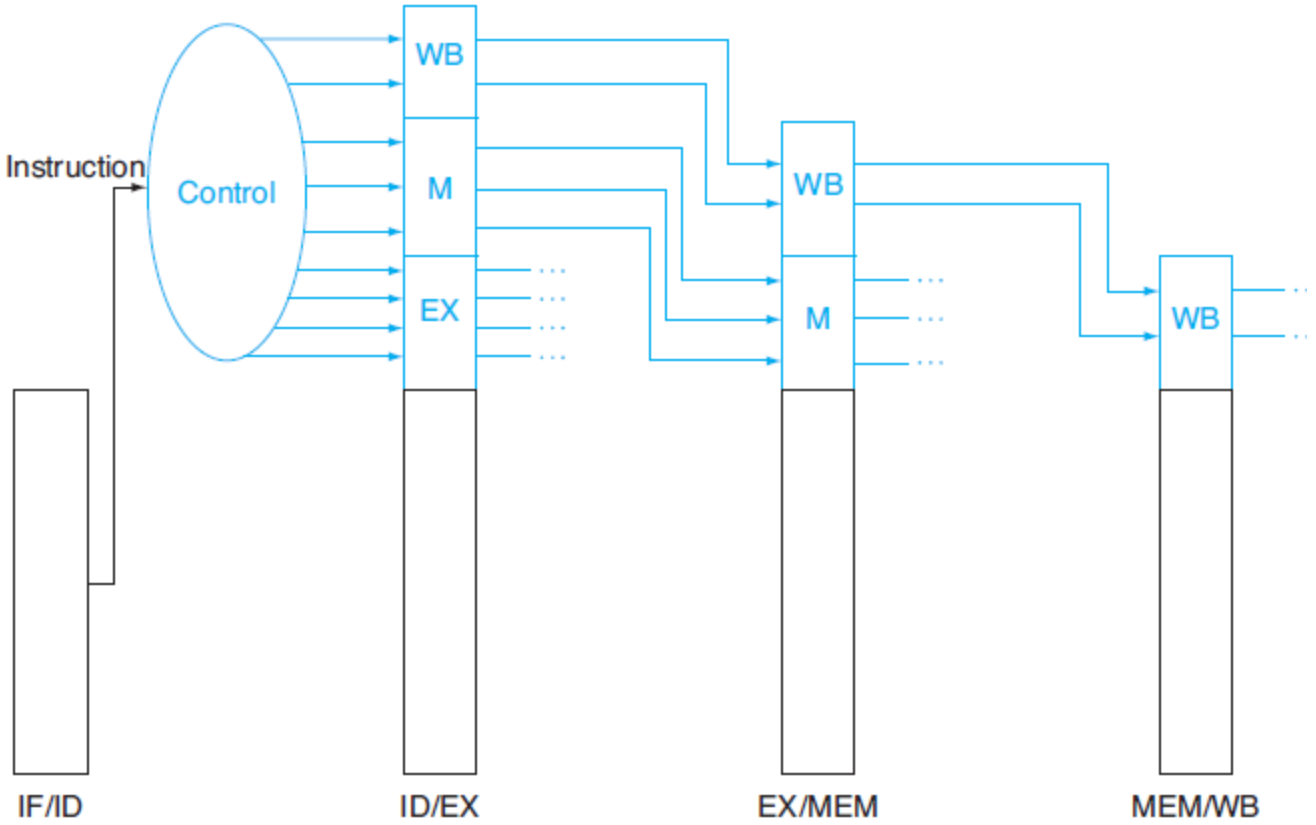
- To share the pipelined datapath, we need to preserve the instruction read during the IF stage, so each pipeline register contains a portion of the instruction needed for that stage and later stages
- Load Instruction needs the register write number in the WB stage



# Designing Control Signals for Pipelining



# Transfer of Control Information



# Transfer of Control Information