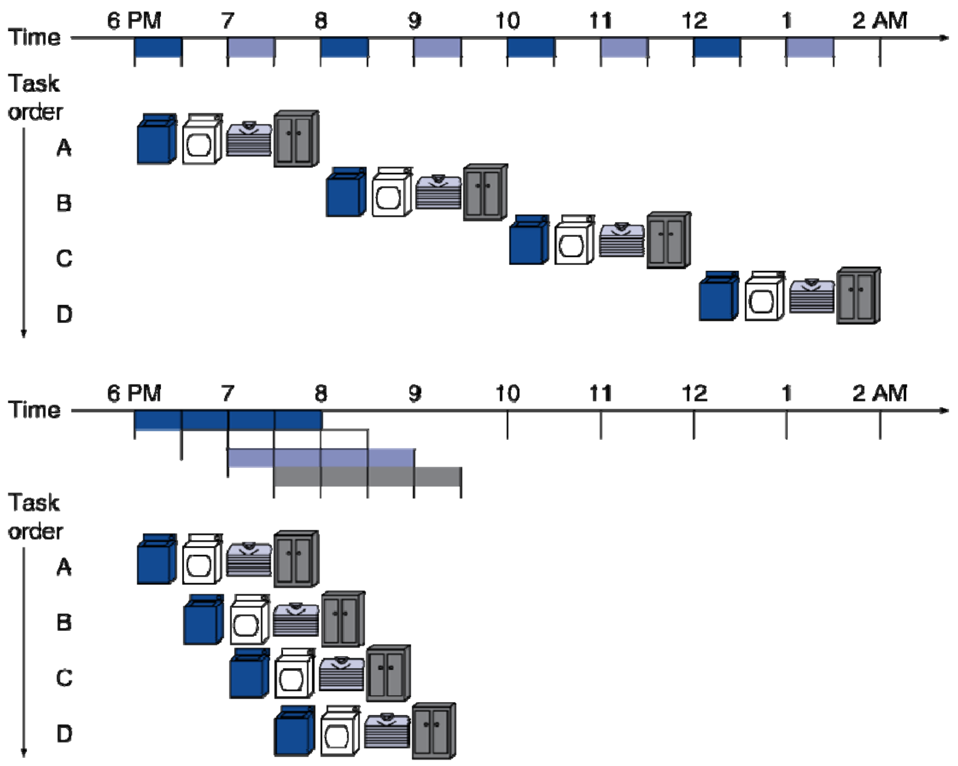
PIPELINING

An implementation technique in which multiple instructions are overlapped in execution.

Pipelining Analogy

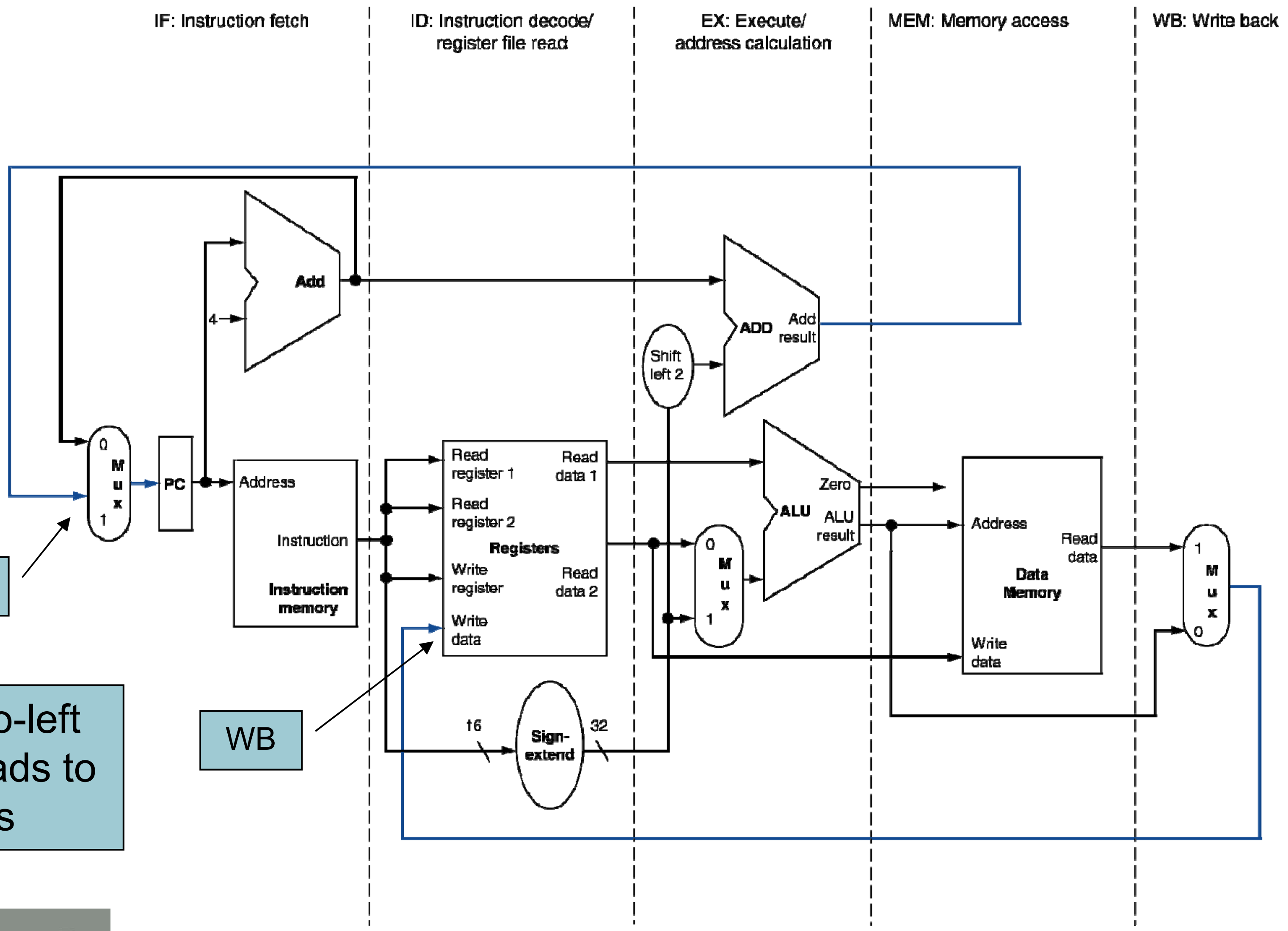
- Pipelined laundry: overlapping execution
  - Parallelism improves performance



MIPS Pipeline

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

MIPS Pipelined Datapath



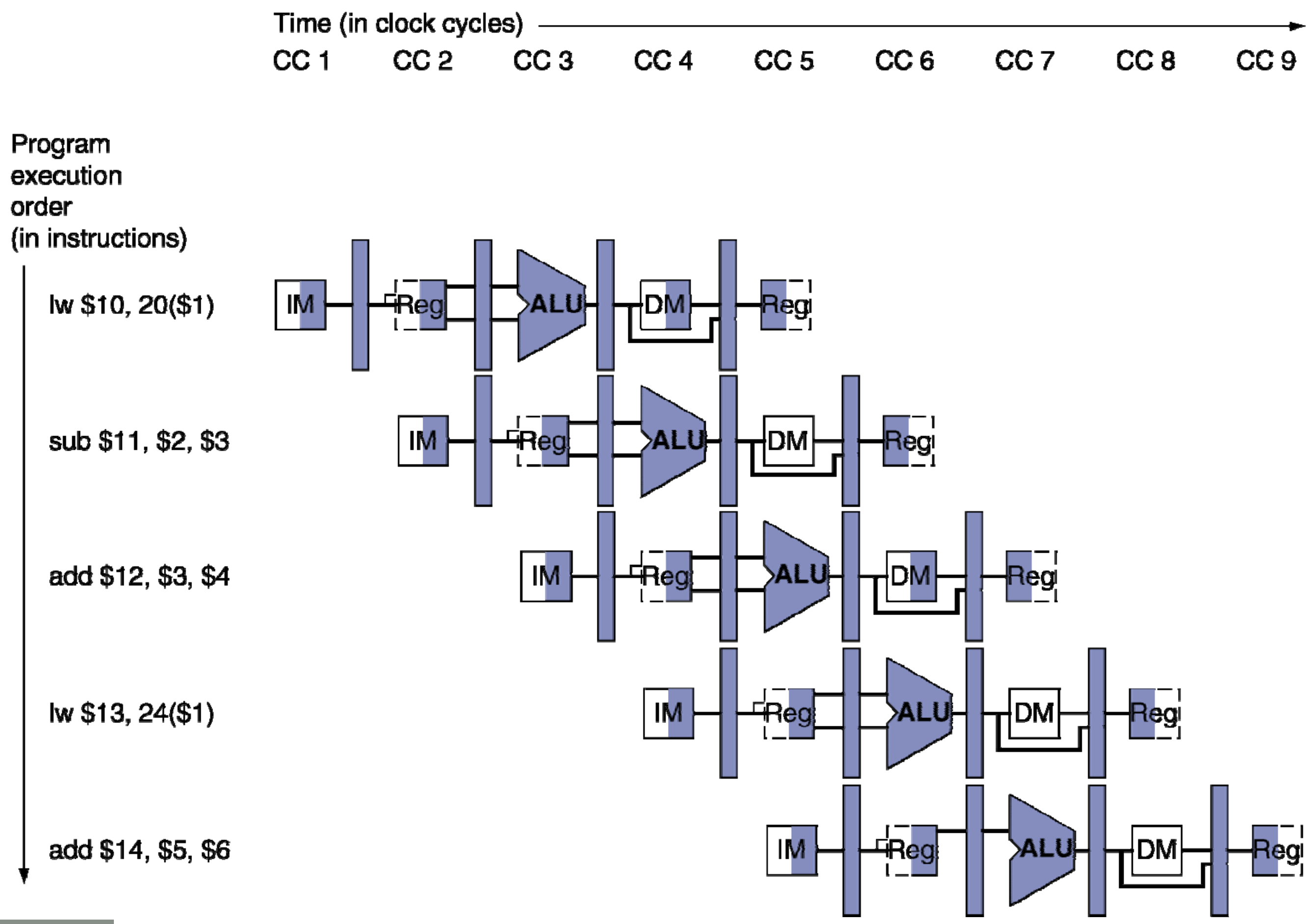
MEM

WB

Right-to-left flow leads to hazards

Multi-Cycle Pipeline Diagram

- Form showing resource usage



Example

A RISC processor has a five-stage pipeline, and a program consists of 10 instructions. Each instruction requires 5 clock cycles for execution, with each clock cycle lasting 5 microseconds. Determine the total execution time (in microseconds) needed to complete the program in both cases: with and without pipelining.

- **Case 1: No Pipelining Used**

In a **non-pipelined processor**, each instruction must complete all 5 stages before the next instruction starts. So, the total execution time is calculated as:
Total time=No. of instructions*Clock cycles per instruction*Time per clock cycle
In more generally

$$T_{non-pipeline} = N * C * T_c$$

Where:

$T_{non-pipeline}$ = Total execution time (in microseconds)
N: Total number of instructions in the program = 10
C = Number of clock cycles required per instruction=5
T_c: Clock Cycle time = 5 microseconds

Total execution time=$10\times5\times5$=250 microseconds

Example

A RISC processor has a five-stage pipeline, and a program consists of 10 instructions. Each instruction requires 5 clock cycles for execution, with each clock cycle lasting 5 microseconds. Determine the total execution time (in microseconds) needed to complete the program in both cases: with and without pipelining.

- **Case 2: With 5-Stage Pipelining**
  - A **5-stage pipeline** means that after the first instruction completes the first stage, the next instruction starts immediately in the first stage (like an assembly line).
  - The first instruction still takes **5 clock cycles** to complete, but each subsequent instruction completes **one per clock cycle** thereafter.

Total execution time = (Number of instructions + Pipeline stages - 1)*Time per clock cycle
In more generally

$$T_{pipeline} = (N + K - 1) * T_c$$

Where:

N: Total number of instructions in the program = 10
K: Number of Pipeline stages = 5
T_c: Clock Cycle time = 5 microseconds

=(10+5−1)*5= 70 microseconds

Example

A RISC processor has a five-stage pipeline, and a program consists of 10 instructions. Each instruction requires 5 clock cycles for execution, with each clock cycle lasting 5 microseconds. Determine the total execution time (in microseconds) needed to complete the program in both cases: with and without pipelining.

- **Case 3: 5-Stage Pipelining with stalls**
  - If you also consider stalls due to hazards (e.g., data hazards, resources hazards), you can modify the equation to:

$$T_{pipeline} = (N + K - 1 + \text{Stalls}) * T_c$$

Where:

N: Total number of instructions in the program

K: Number of Pipeline stages

T_c: Clock Cycle time =

Stalls: Number of stall cycles