EE30342 – Digital Design with HDL (II)

Lecture 12

Dr. Ming Xu

Dept of Electrical & Electronic Engineering

XJTLU

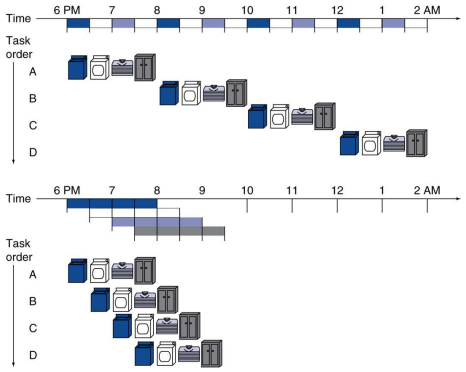
In This Session

- Enhance Performance with Pipelining

Pipelining Analogy

§ 4.5 An Overview of Pipelining

- Pipelined laundry: overlapping execution
  - Parallelism improves performance



- Four loads:
  - Speedup = 8/3.5 = 2.3
- Non-stop:
  - Speedup ≈ number of stages = 4

MIPS Pipeline

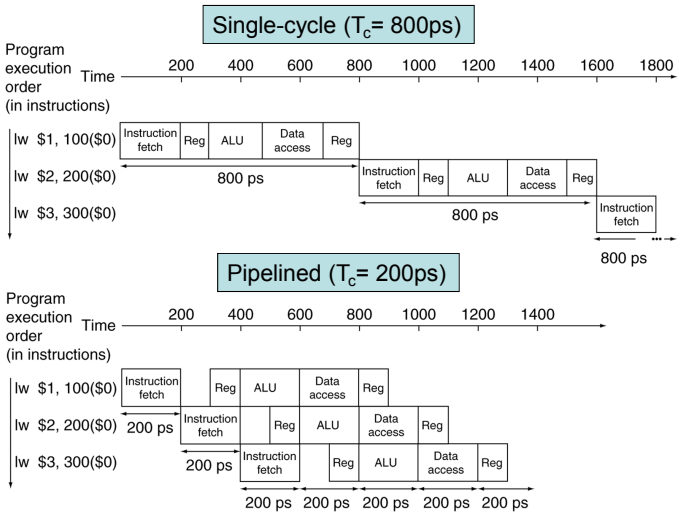
- Five stages
  1. IF: Instruction fetch from memory
  2. ID: Instruction decode & register read
  3. EX: Execute operation or calculate address
  4. MEM: Access memory operand
  5. WB: Write result back to register

Pipeline Performance

- Assume time for stages is
  - 100ps for register read or write
  - 200ps for other stages
- Compare pipelined datapath with single-cycle datapath

Instr	Instr fetch	Register read	ALU op	Memory access	Register write	Total time
lw	200ps	100 ps	200ps	200ps	100 ps	800ps
sw	200ps	100 ps	200ps	200ps		700ps
R-format	200ps	100 ps	200ps		100 ps	600ps
beq	200ps	100 ps	200ps			500ps

Pipeline Performance



Pipeline Speedup

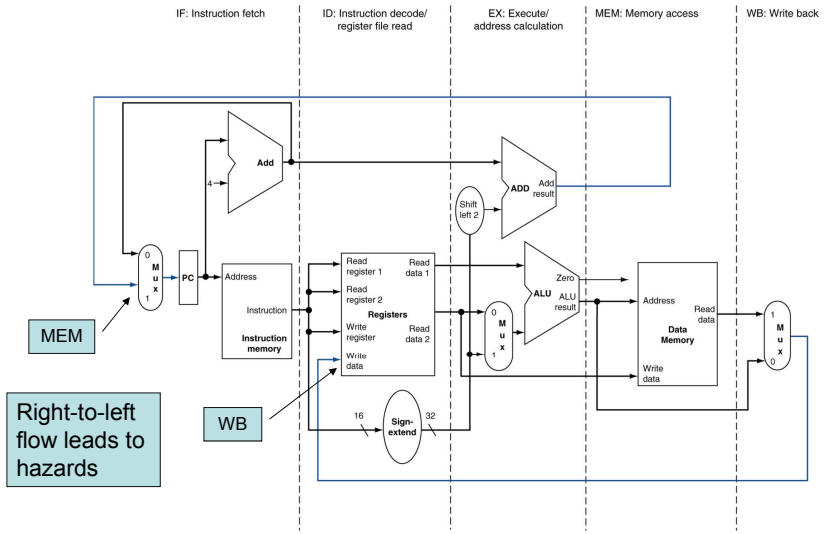
- If all stages are balanced
  - i.e., all take the same time
  - Time between instructions$_{pipelined}$ = $\frac{\text{Time between instructions}_{nonpipelined}}{\text{Number of stages}}$
- If not balanced, speedup is less
- Speedup due to increased throughput
  - Latency (time for each instruction) does not decrease

Pipelining and ISA Design

- MIPS ISA designed for pipelining
  - All instructions are 32-bits
    - Easier to fetch and decode in one cycle
    - c.f. x86: 1- to 17-byte instructions
  - Few and regular instruction formats
    - Can decode and read registers in one step
  - Memory operands in Load/store only
    - Can calculate address in 3rd stage, access memory in 4th stage
  - Alignment of memory operands
    - Memory access takes only one cycle

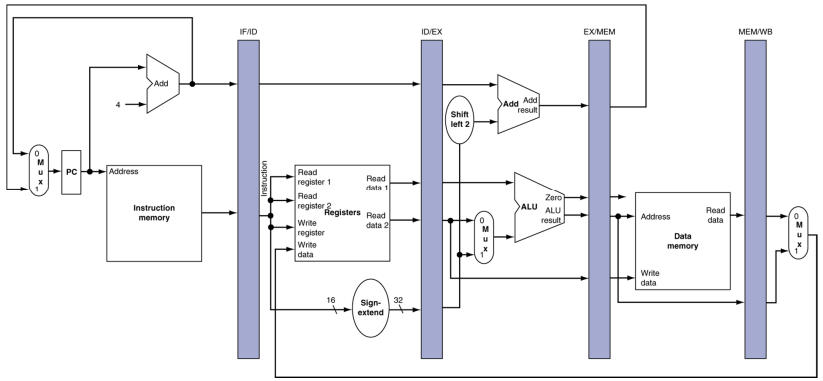
MIPS Pipelined Datapath

§ 4.6 Pipelined Datapath and Control



Pipeline registers

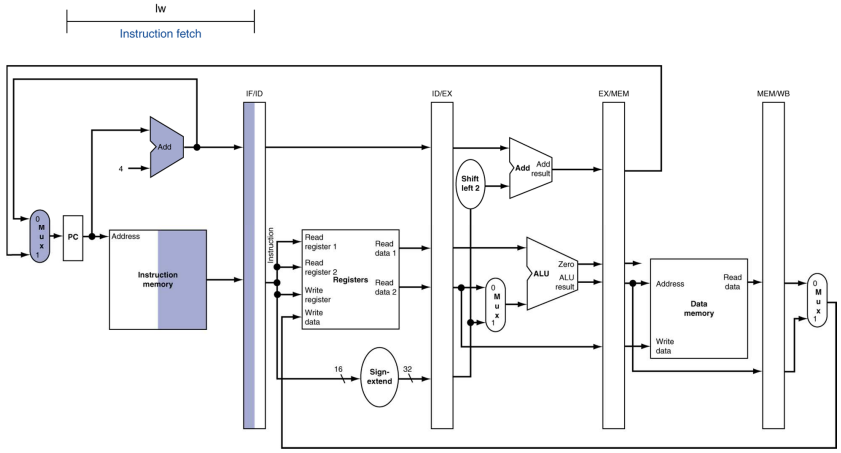
- Need registers between stages
  - To hold information produced in previous cycle



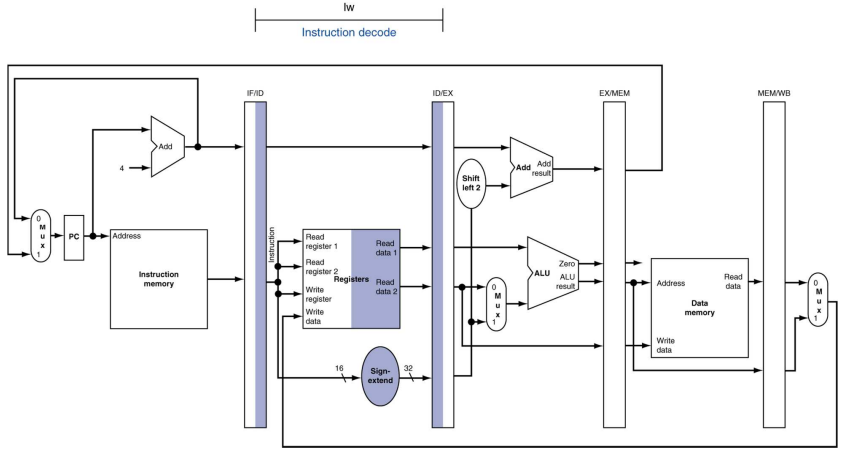
Pipeline Operation

- 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
  - c.f. "multi-clock-cycle" diagram
    - Graph of operation over time
- We'll look at "single-clock-cycle" diagrams for load & store

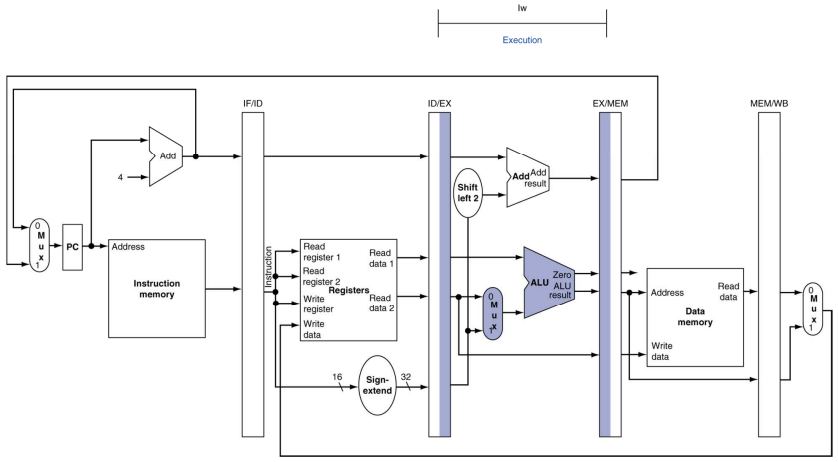
IF for Load, Store, …



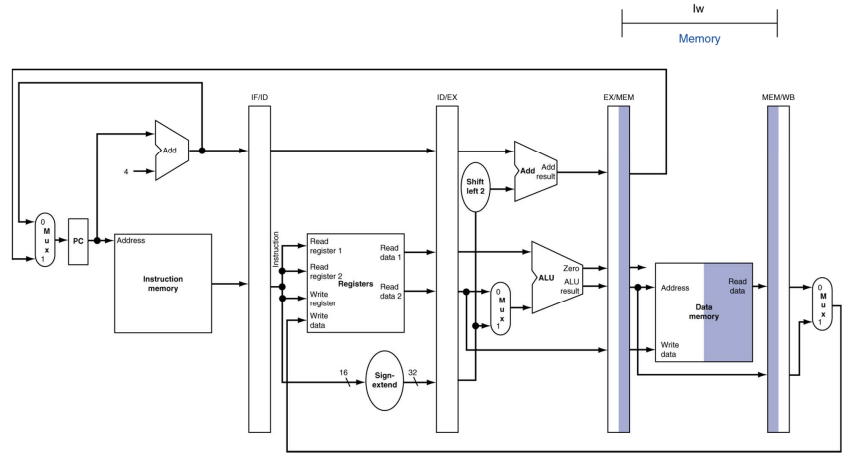
ID for Load, Store, …



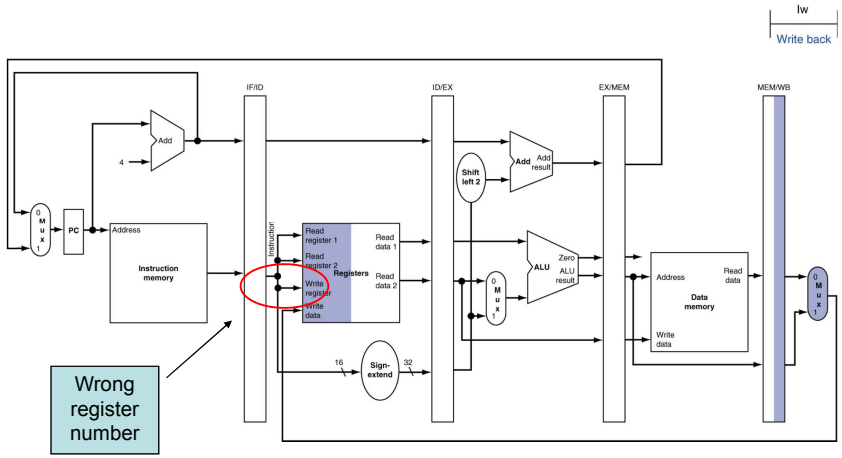
EX for Load



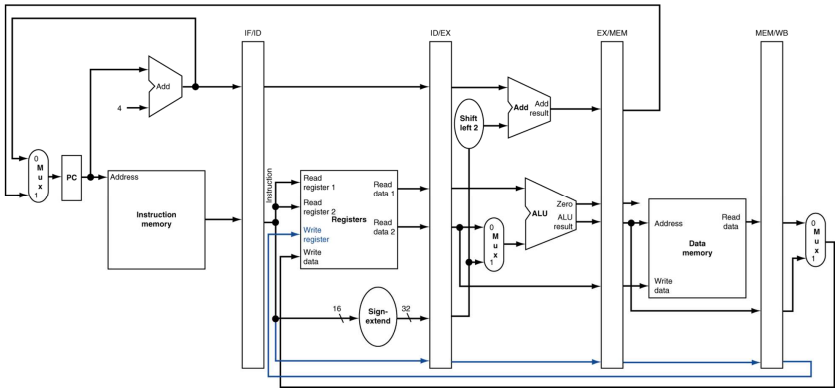
MEM for Load



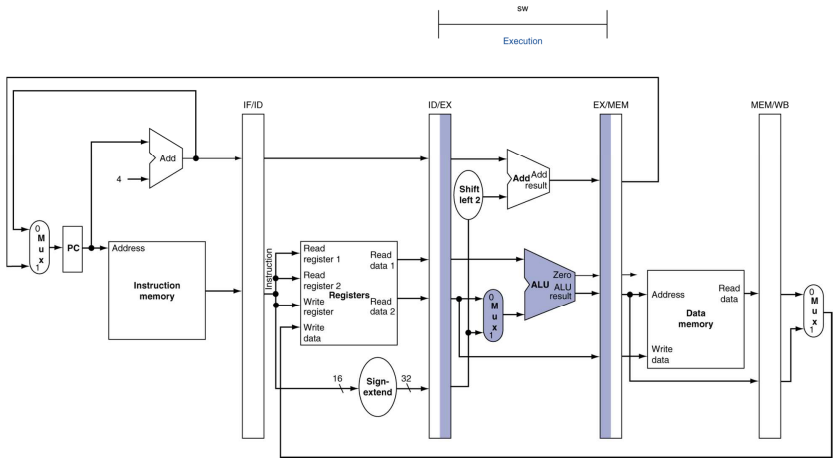
WB for Load



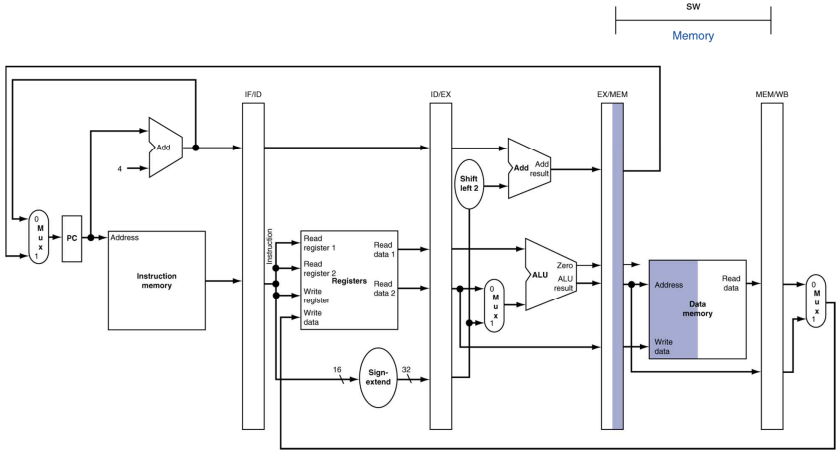
Corrected Datapath for Load



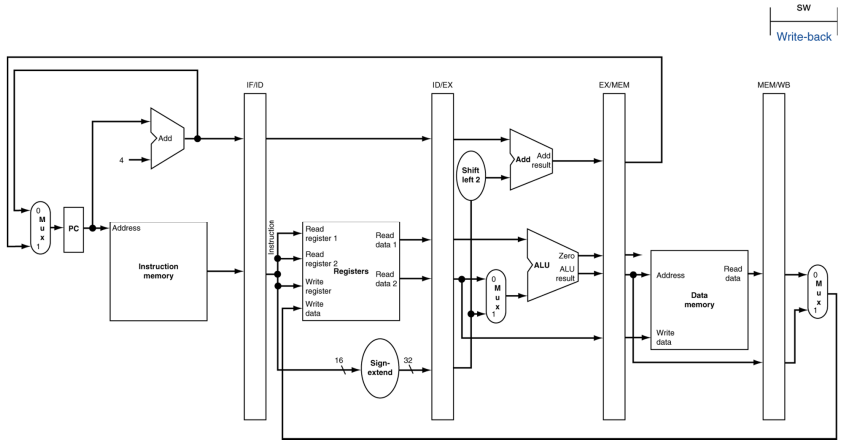
EX for Store



MEM for Store

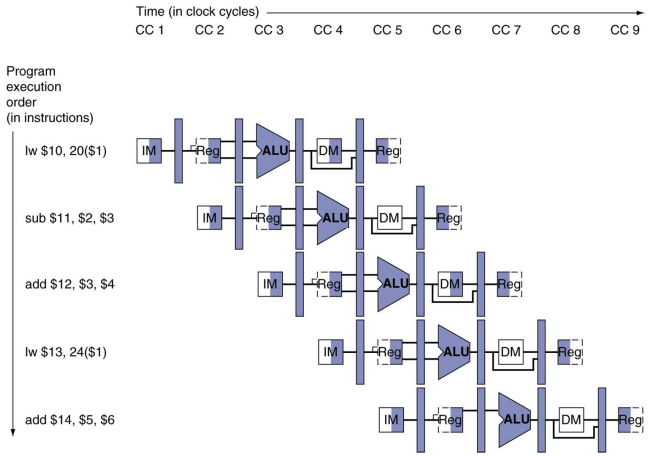


WB for Store



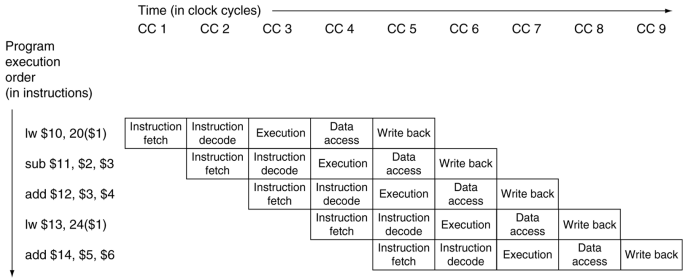
Multi-Cycle Pipeline Diagram

- Form showing resource usage



Multi-Cycle Pipeline Diagram

- Traditional form



Single-Cycle Pipeline Diagram

- State of pipeline in a given cycle