Lecture 6: Pipelining

CSCE 2610 Computer Organization

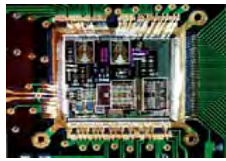
Instructor: Saraju P. Mohanty, Ph. D.

NOTE: The figures, text etc included in slides are borrowed from various books, websites, authors pages, and other sources for academic purpose only. The instructor does not claim any originality.

The Big Picture: Where are We Now?

- We know five classic components of a computer.
- We understand how the instruction set plays a key role in determining the performance, the design complexity of the datapath and controller.
- We designed a processor comprising of datapath and controller for a small set of instructions.
- Datapath:
  - Single-cycle implementation - Too slow
  - Multi-cycle implementation – Large instructions take longer time, small instructions take shorter time
- Controller: Approach I: *FSM Based Approach*
  - Structured approach to derive a circuit implementation from FSM specification
  - Implementation styles: (1) Random-logic (2) PLA (3) ROM

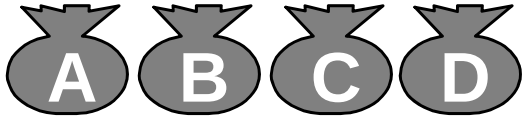
    Approach 2: *Microprogramming*
  - Control written as a program using microinstructions
  - Flexible but slower



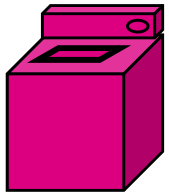
Pipelining is Natural!

Laundry Example

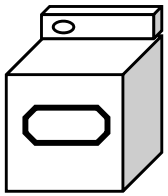
- Ann, Brian, Cathy, Dave each have one load of clothes to wash, dry, and fold.



- Washing takes 30 minutes.

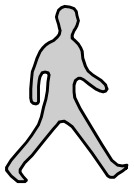


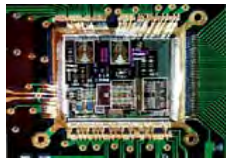
- Drying takes 30 minutes.



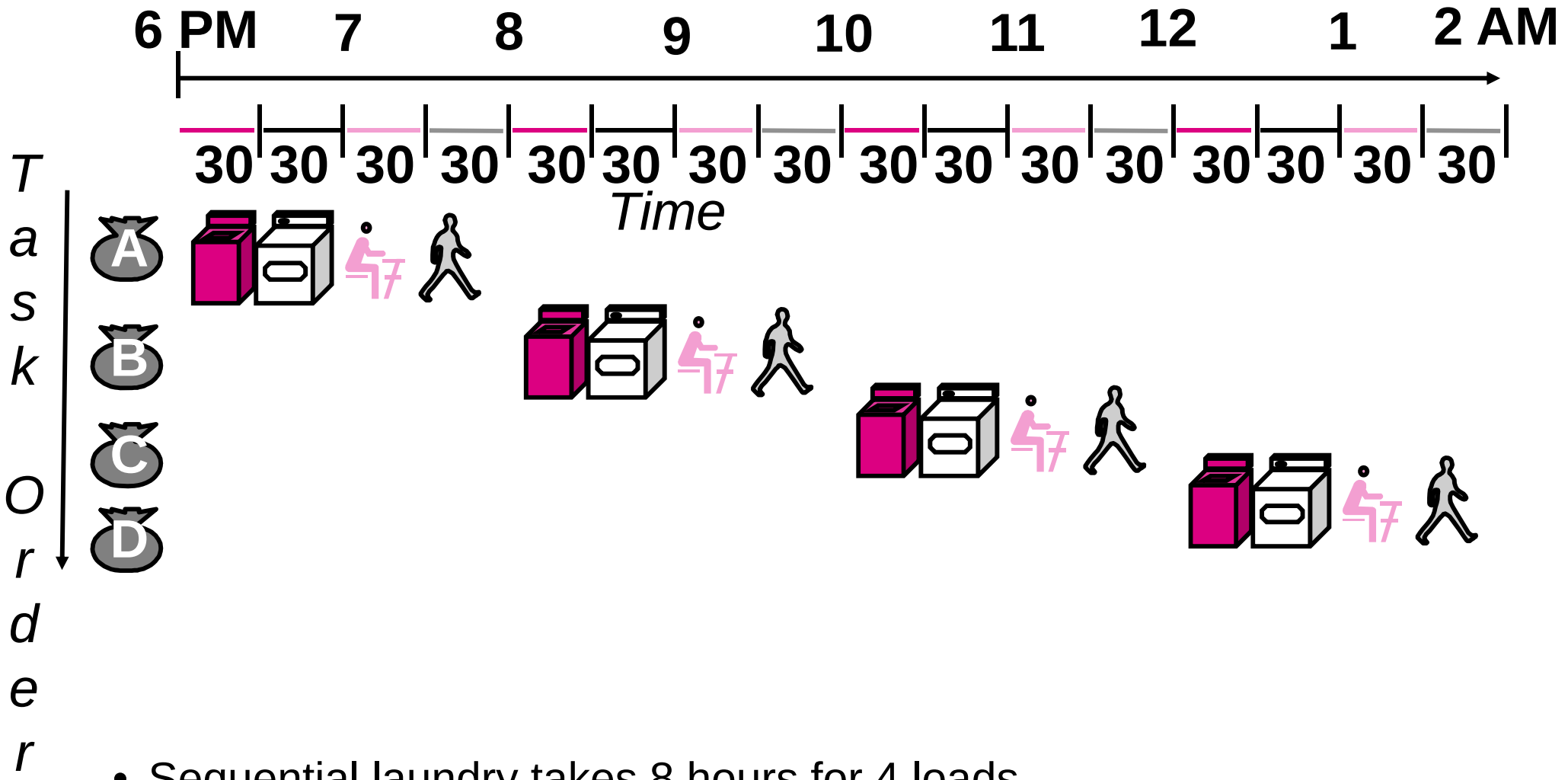
- Folding takes 30 minutes.

- Putting-away takes 30 minutes to put clothes into drawers.

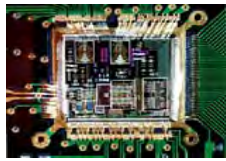




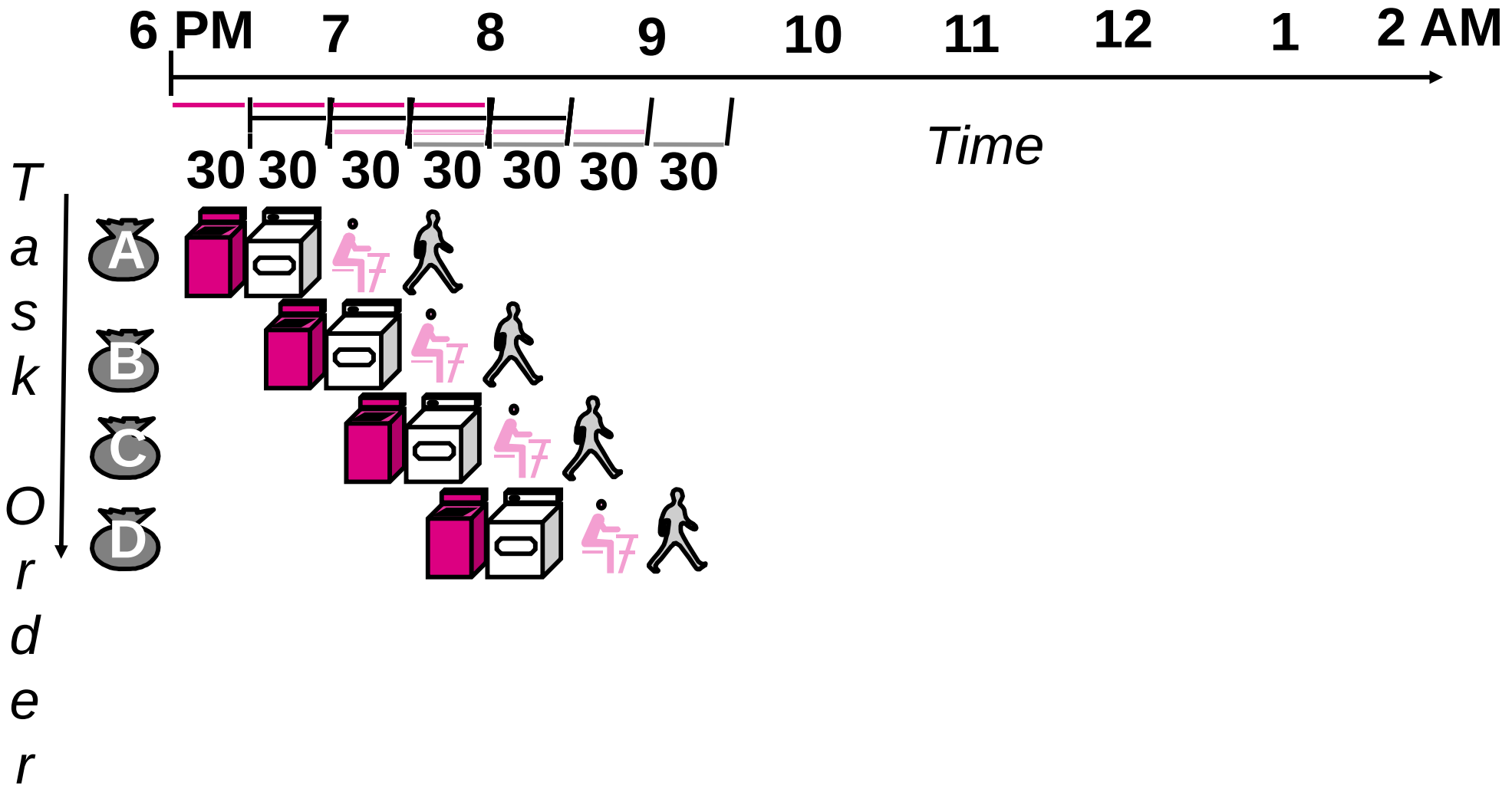
Sequential Laundry



- Sequential laundry takes 8 hours for 4 loads.
- If they learned pipelining, how long would laundry take?

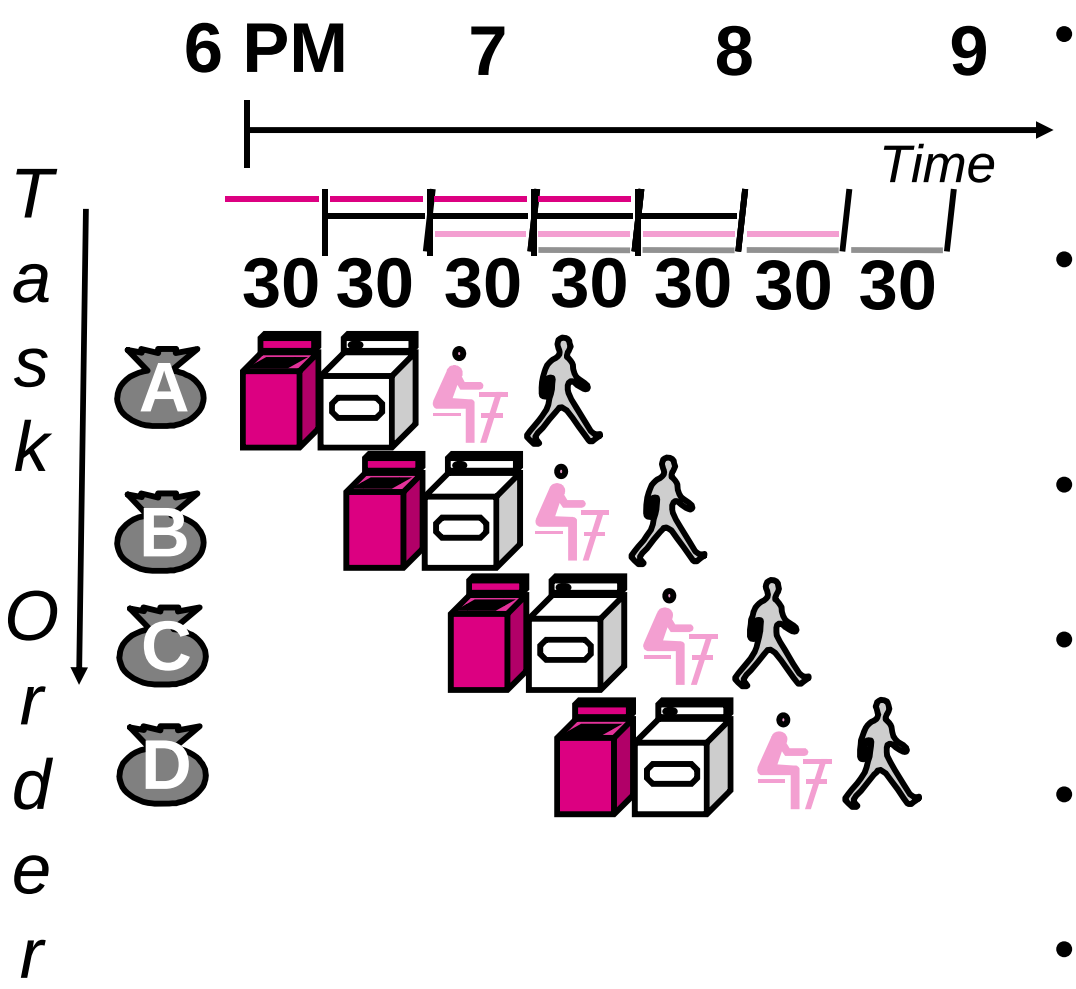


Pipelined Laundry: Start work ASAP



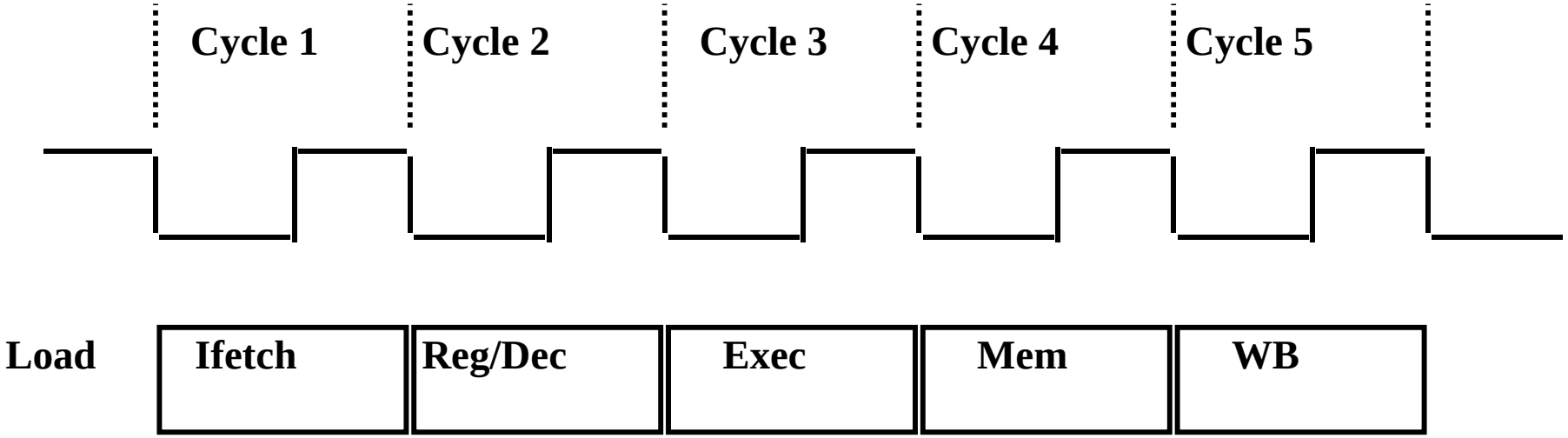
- Pipelined laundry takes 3.5 hours for 4 loads!

Pipelining Lessons



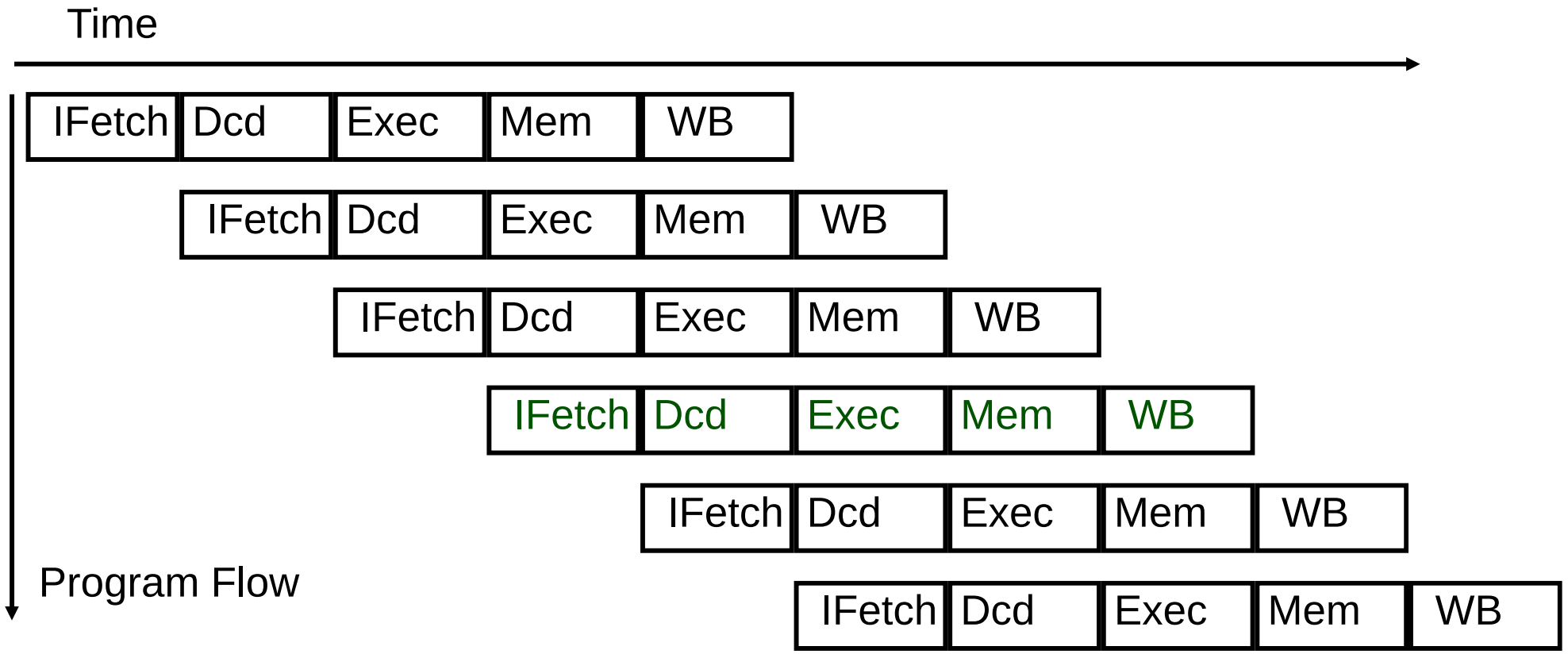
- Pipelining doesn't help latency of single task, it helps throughput of entire workload.
- Multiple tasks operating simultaneously using different resources.
- Potential speedup = Number pipeline stages.
- Pipeline rate limited by slowest pipeline stage.
- Unbalanced lengths of pipe stages reduces speedup.
- Time to "fill" pipeline and time to "drain" it reduces speedup.
- Stall for dependences.

MIPS case: The Five Stages of Load

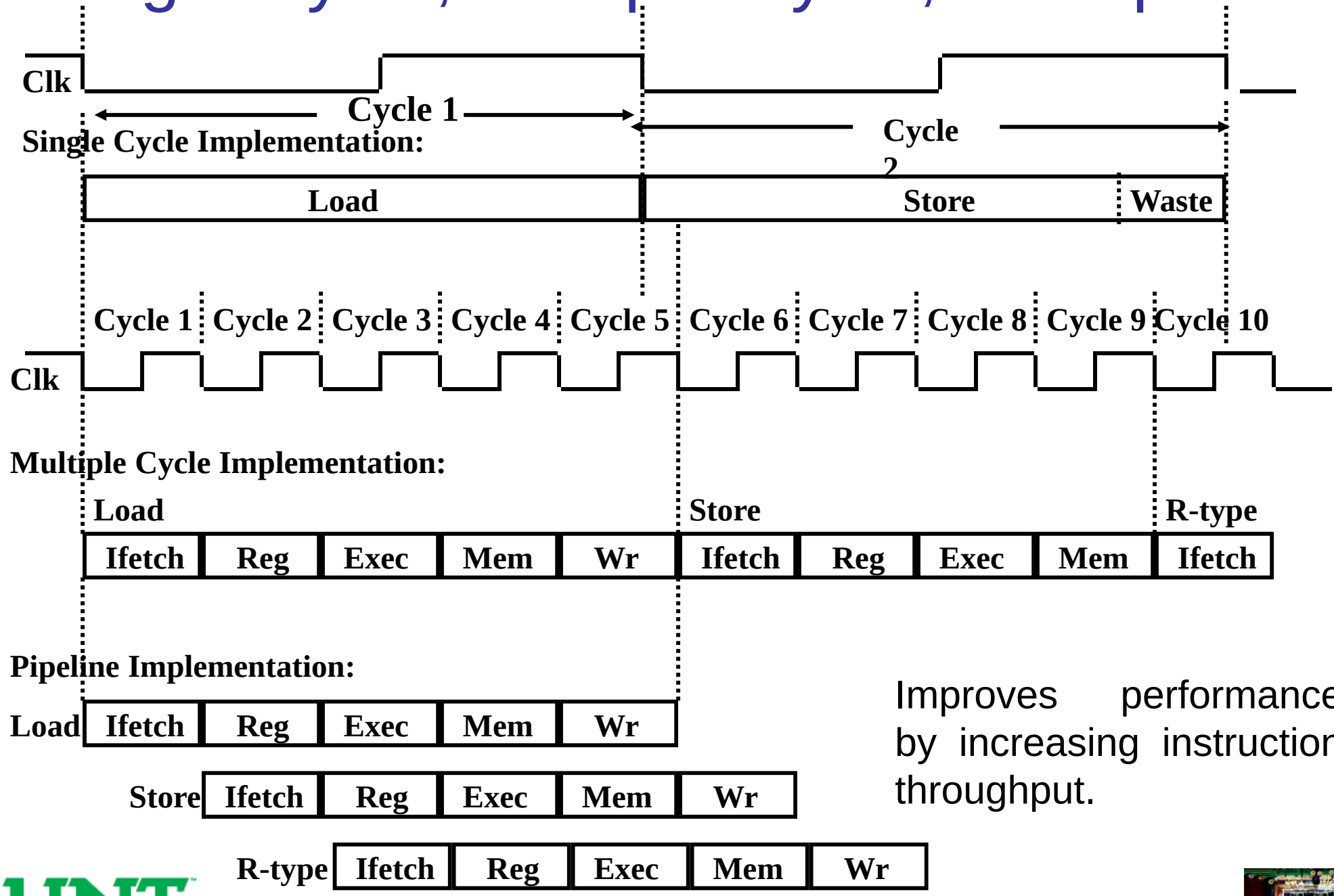


- Ifetch: Instruction Fetch
  - –Fetch the instruction from the Instruction Memory
- Reg/Dec: Registers Fetch and Instruction Decode
- Exec: Calculate the memory address
- Mem: Read the data from the Data Memory
- WB: Write the data back to the register file

Conventional Pipelined Execution Representation

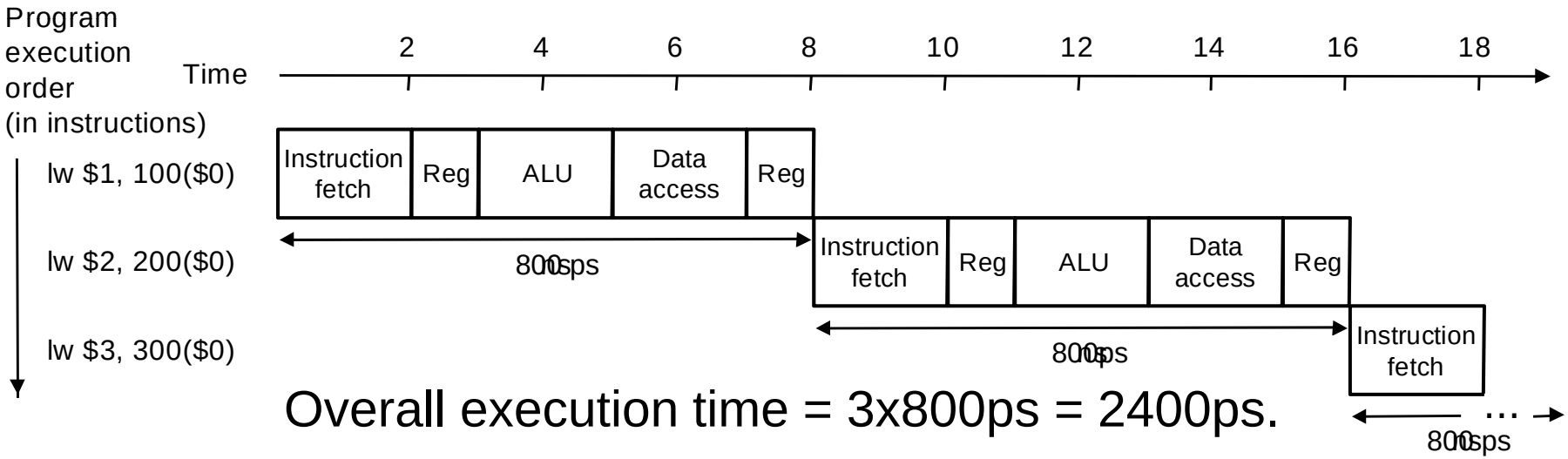


Single Cycle, Multiple Cycle, vs. Pipeline

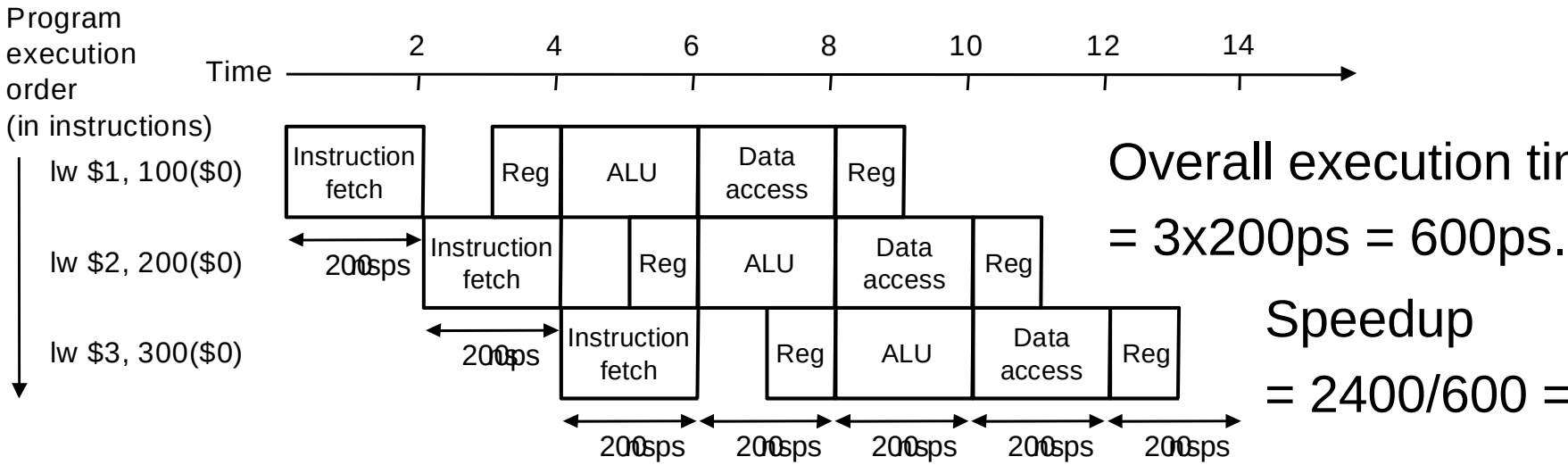


Improves performance by increasing instruction throughput.

Single Cycle Vs Pipelining



Overall execution time = 3x800ps = 2400ps.



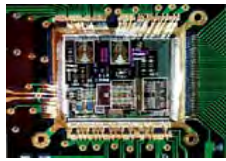
Overall execution time = 3x200ps = 600ps.

Speedup = 2400/600 = 4.

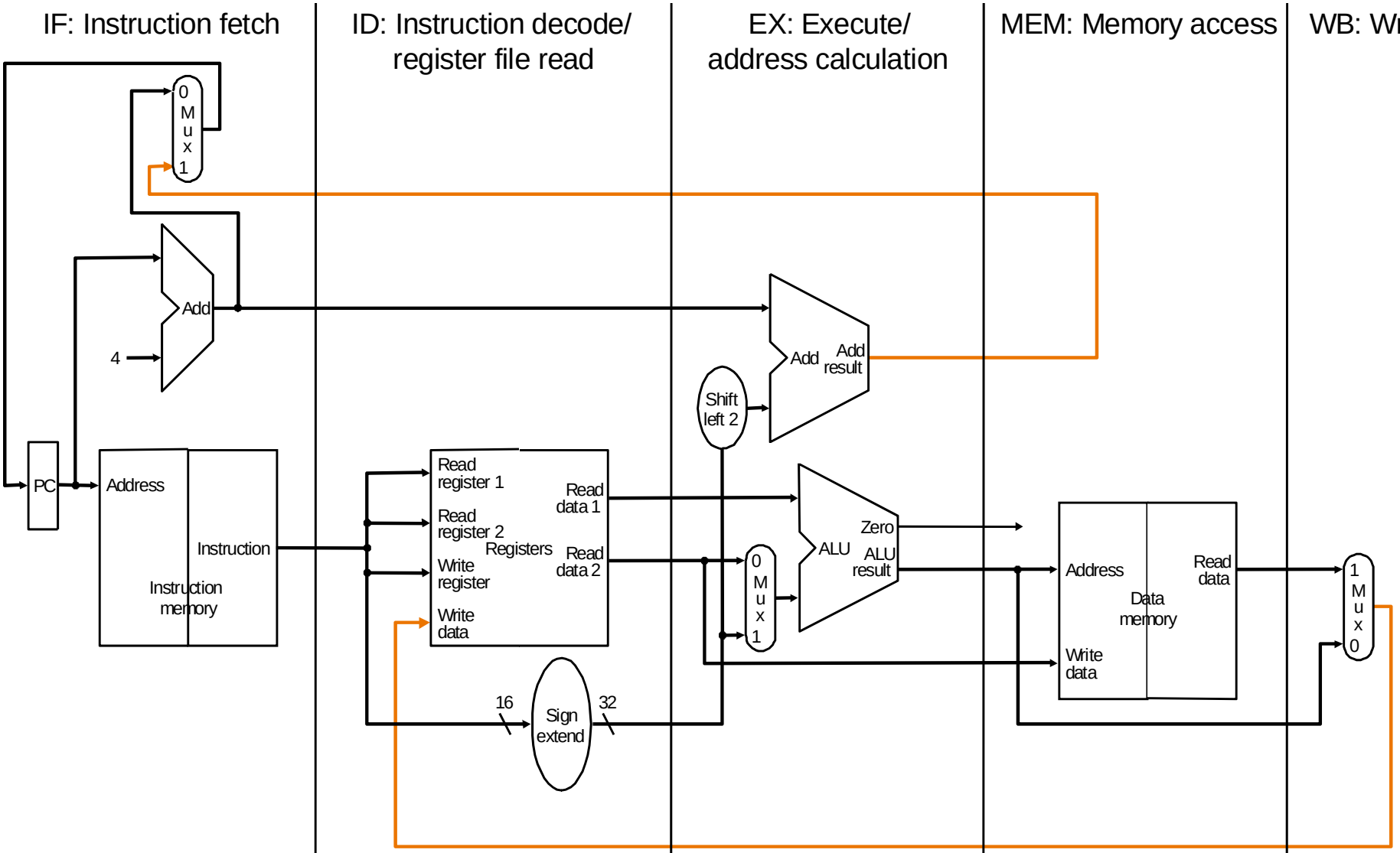
Ideal speedup is number of stages in the pipeline. Do we achieve this?

Pipelining – What makes it easy/hard?

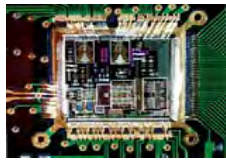
- What makes it easy
  - all instructions are the same length
  - just a few instruction formats
  - memory operands appear only in loads and stores
- What makes it hard?
  - structural hazards: suppose we had only one memory
  - data hazards: an instruction depends on a previous instruction
  - control hazards: need to worry about branch instructions
- We'll build a simple pipeline and look at these issues.



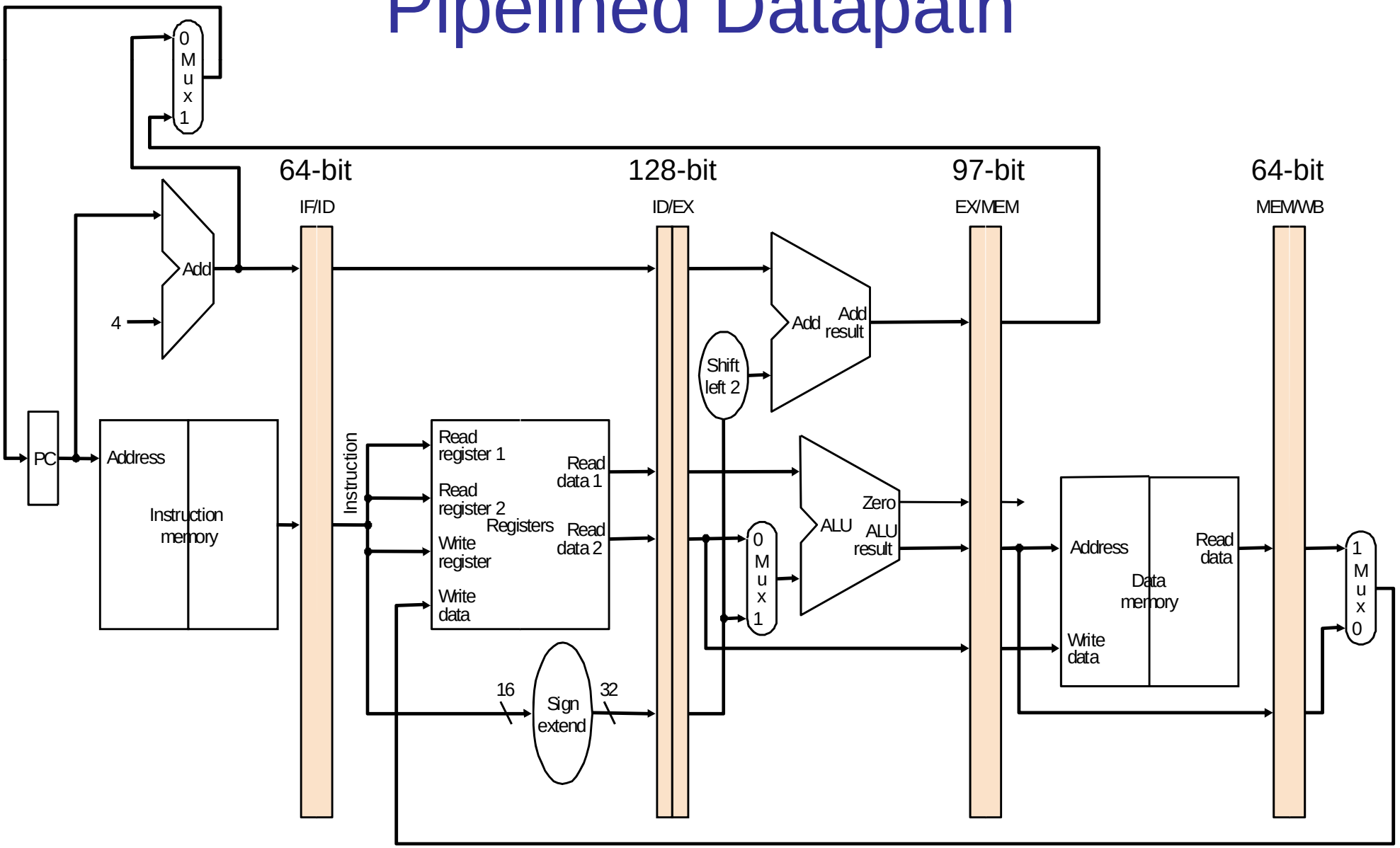
Pipelining the Datapath: Basic Idea



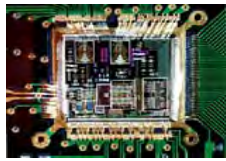
What do we need to add to actually split the datapath into stages?



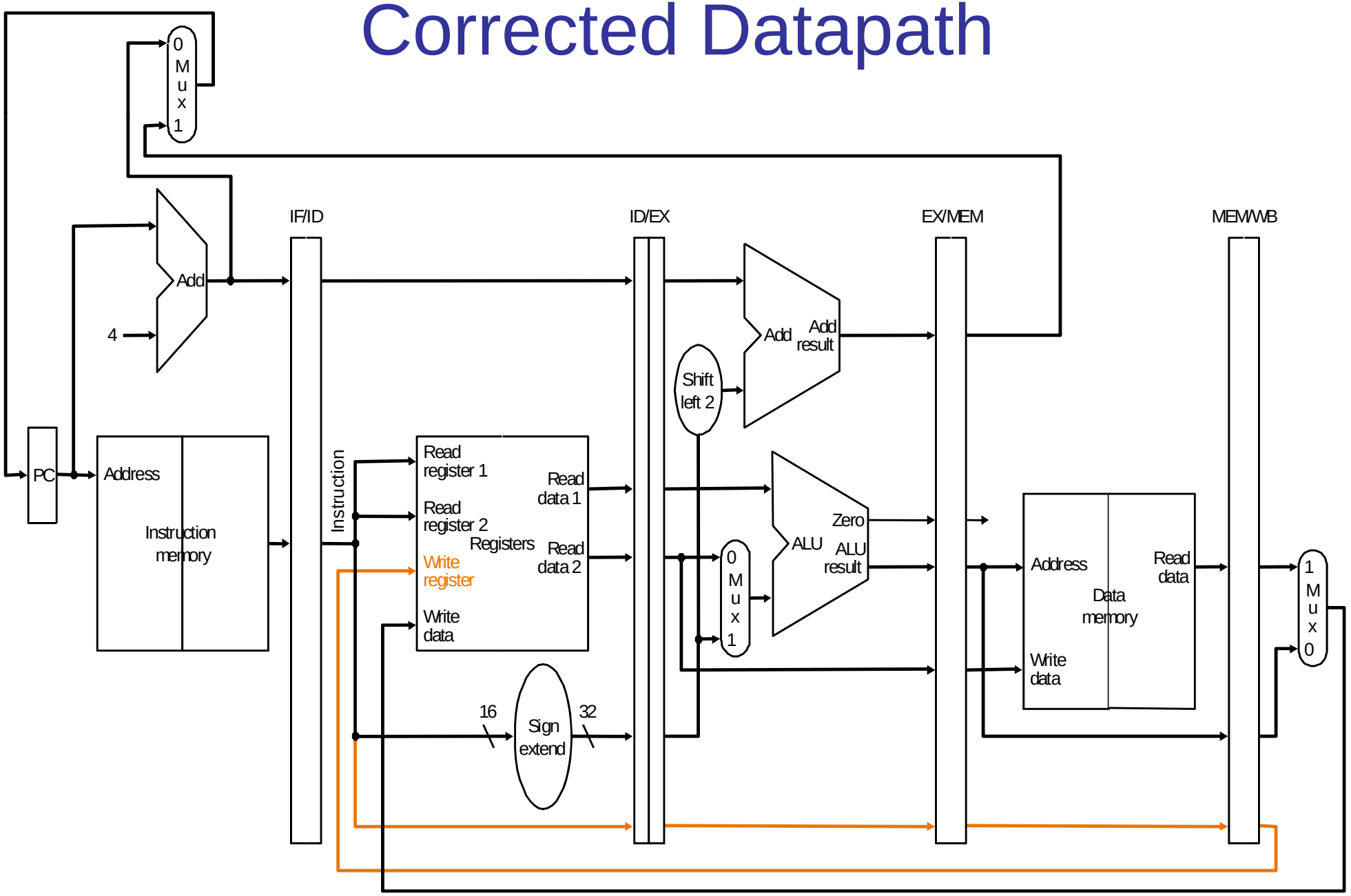
Pipelined Datapath



- Follow Fig. 12 (page-389) to Fig. 14 (page-391) to understand pipelined execution of *lw* instruction. Others instructions will be similar.

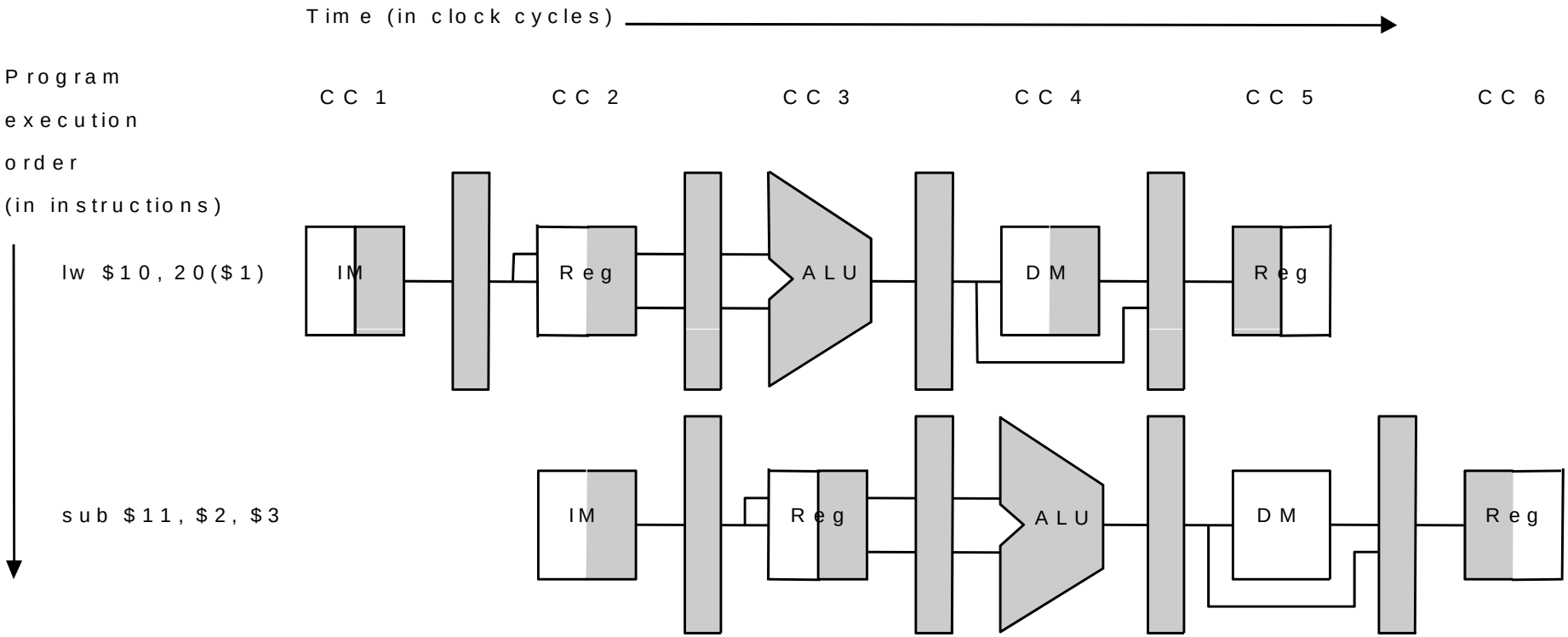


Corrected Datapath



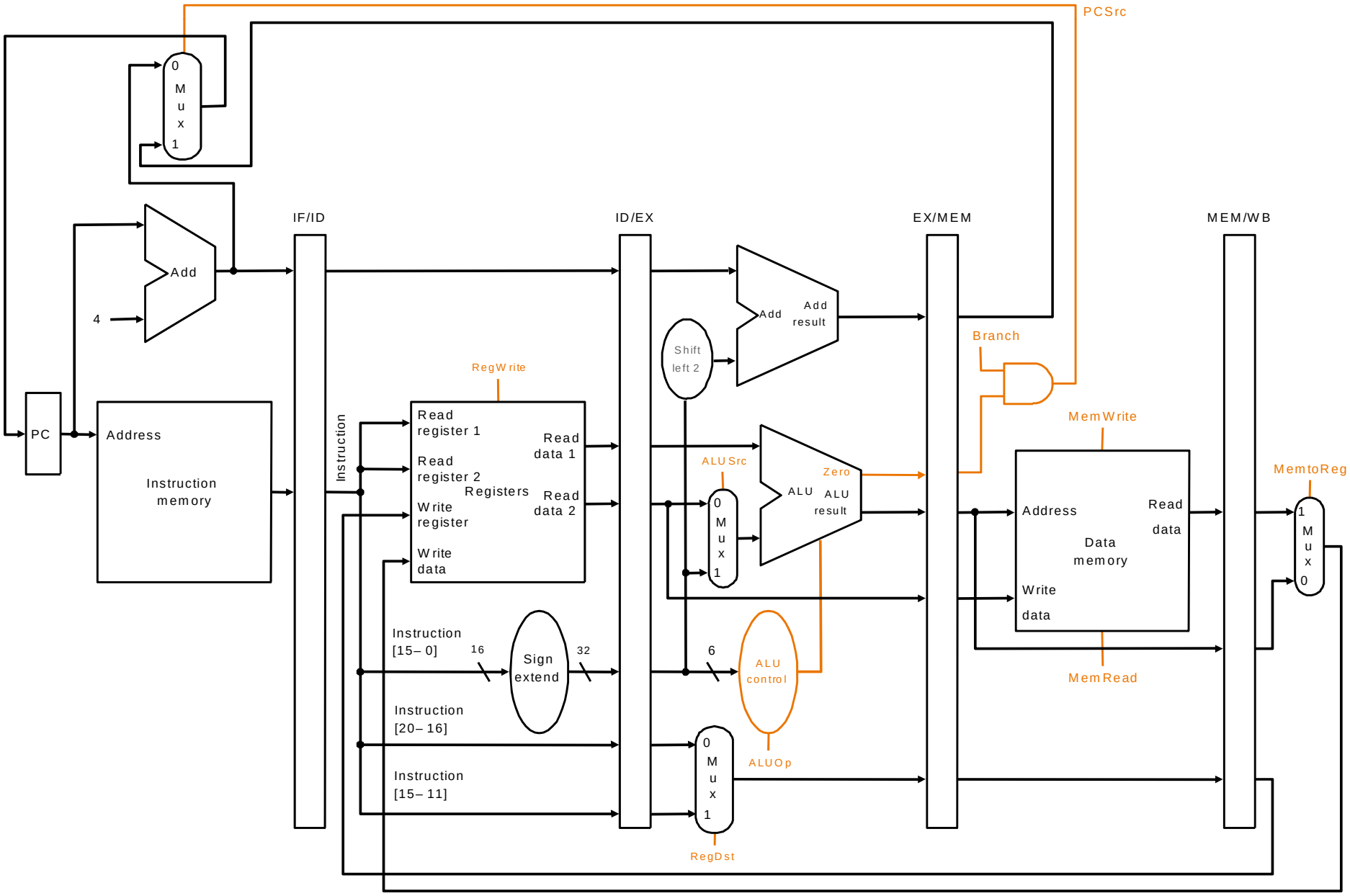
- For load instruction, register number is needed in the last stage, thus same needs to be passed along in order to be preserved.

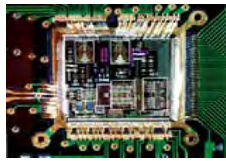
Graphically Representing Pipelines



- Pipeline can be thought of as a series of datapaths shifted in time.
- The above graphics can help in answering questions like:
  - how many cycles does it take to execute this code?
  - what is the ALU doing during cycle 4?
  - use this representation to help understand datapaths

Pipeline Control





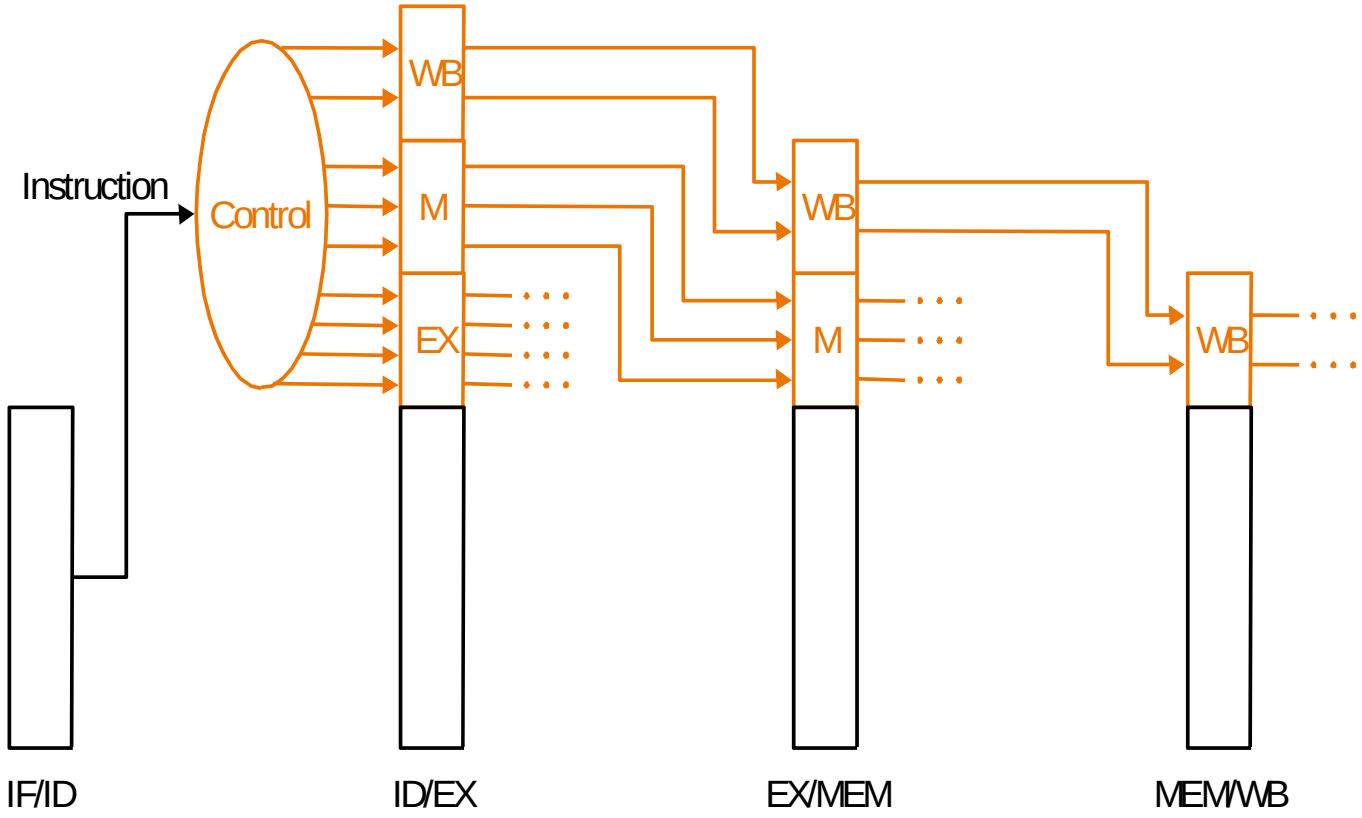
Pipeline Control

- We have 5 stages. What needs to be controlled in each stage?
  - Instruction Fetch and PC Increment
  - Instruction Decode / Register Fetch
  - Execution
  - Memory Stage
  - Write Back

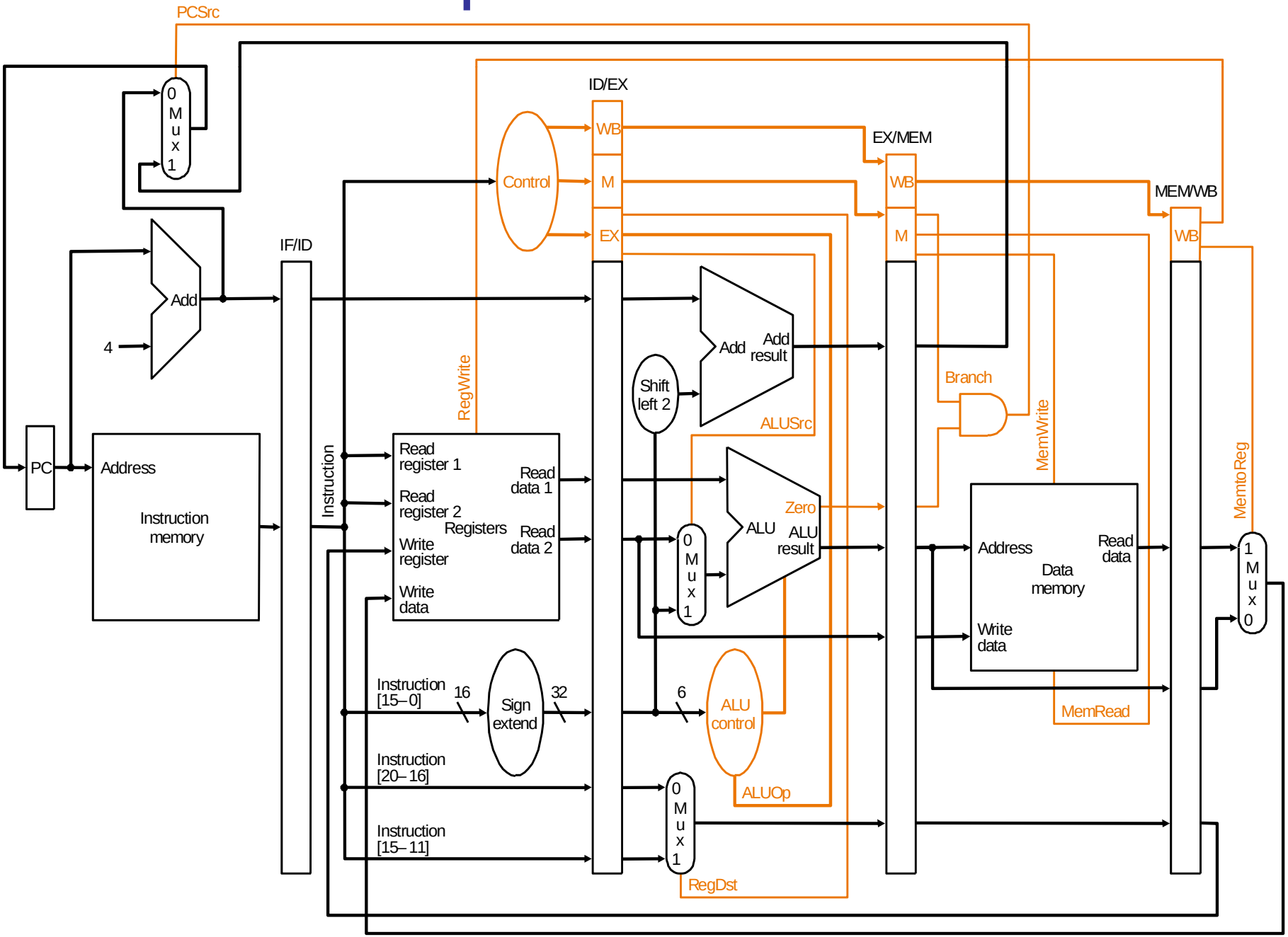
Pipeline Control

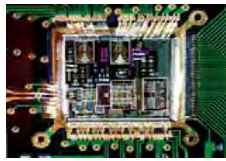
- Pass control signals along just like the data.

Instruction	Execution/Address Calculation stage control lines				Memory access stage control lines			stage control lines	
	Reg Dst	ALU Op1	ALU Op0	ALU Src	Branch	Mem Read	Mem Write	Reg write	Mem to Reg
R-format	1	1	0	0	0	0	0	1	0
`lw`	0	0	0	1	0	1	0	1	1
`sw`	X	0	0	1	0	0	1	0	X
`beq`	X	0	1	0	1	0	0	0	X



Datapath with Control





Major Hurdles of Pipelining

- Pipeline Hazards
  - Dictionary meaning of hazard: “a source of danger”
  - structural hazards: attempt to use the same resource two different ways at the same time
    - e.g., combined washer/dryer would be a structural hazard
  - data hazards: attempt to use item before it is ready
    - Instruction depends on result of prior instruction still in the pipeline
  - control hazards: attempt to make a decision before condition is evaluated
    - Branch instructions
- One Solution: Wait until dependencies are resolved
  - pipeline control must detect the hazard
  - take action (or delay action) to resolve hazards