CSE 140
Computer Architecture

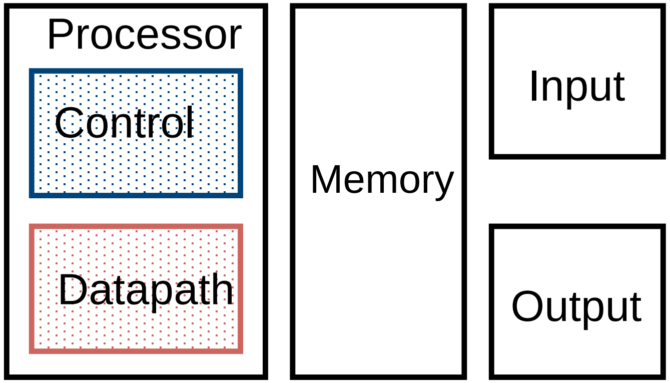
Lecture 4 – Pipelining

Announcement

- Lab #1 starts this week
  - Due in one week
- Reading assignment #2
  - Chapter 2.5, 4.5 – 4.8
    - Do all Participation Activities in each section
    - Access through CatCourses
    - Due Thursday (9/12) at 11:59pm
  - Review CSE 31 materials (available at CatCourses)
    - Assembly language and machine code: Ch. 2.1-2.8, 2.9-2.10, 2.13
  - Quick review:
    - https://classroom.udacity.com/courses/ud219

Review: Single-cycle Processor

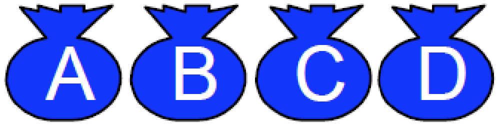
- 5 steps to design a processor
  - Analyze instruction set → datapath requirements
  - Select set of datapath components & establish clock methodology
  - Assemble datapath meeting the requirements
  - Analyze implementation of each instruction to determine setting of control points that effects the register transfer.
  - Assemble the control logic
- Control is the hard part
  - Not easy to design control based on C/C++
- MIPS makes that easier
  - Instructions same size
  - Source registers always in same place
  - Immediates same size, location
  - Operations always on registers/immediates



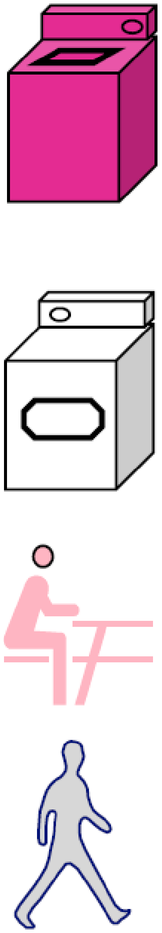
Processor Performance

- Can we ***estimate*** the clock rate (frequency) of our single-cycle processor?
  - 1 cycle per instruction *(Which instruction takes the longest?)*
  - ***lw*** is the most demanding instruction.
  - Assume these delays for major pieces of the datapath:
    - Instr. Mem, ALU, Data Mem : 2ns each, regfile: 1ns
    - Instruction execution requires: 2 + 1 + 2 + 2 + 1 = 8ns
    - ⇒ 125 MHz Around the speed of a Pentium I processor in mid 90's
- What can we do to improve clock rate?
- Will this improve performance as well?
  - We want increases in clock rate to result in programs executing quicker.

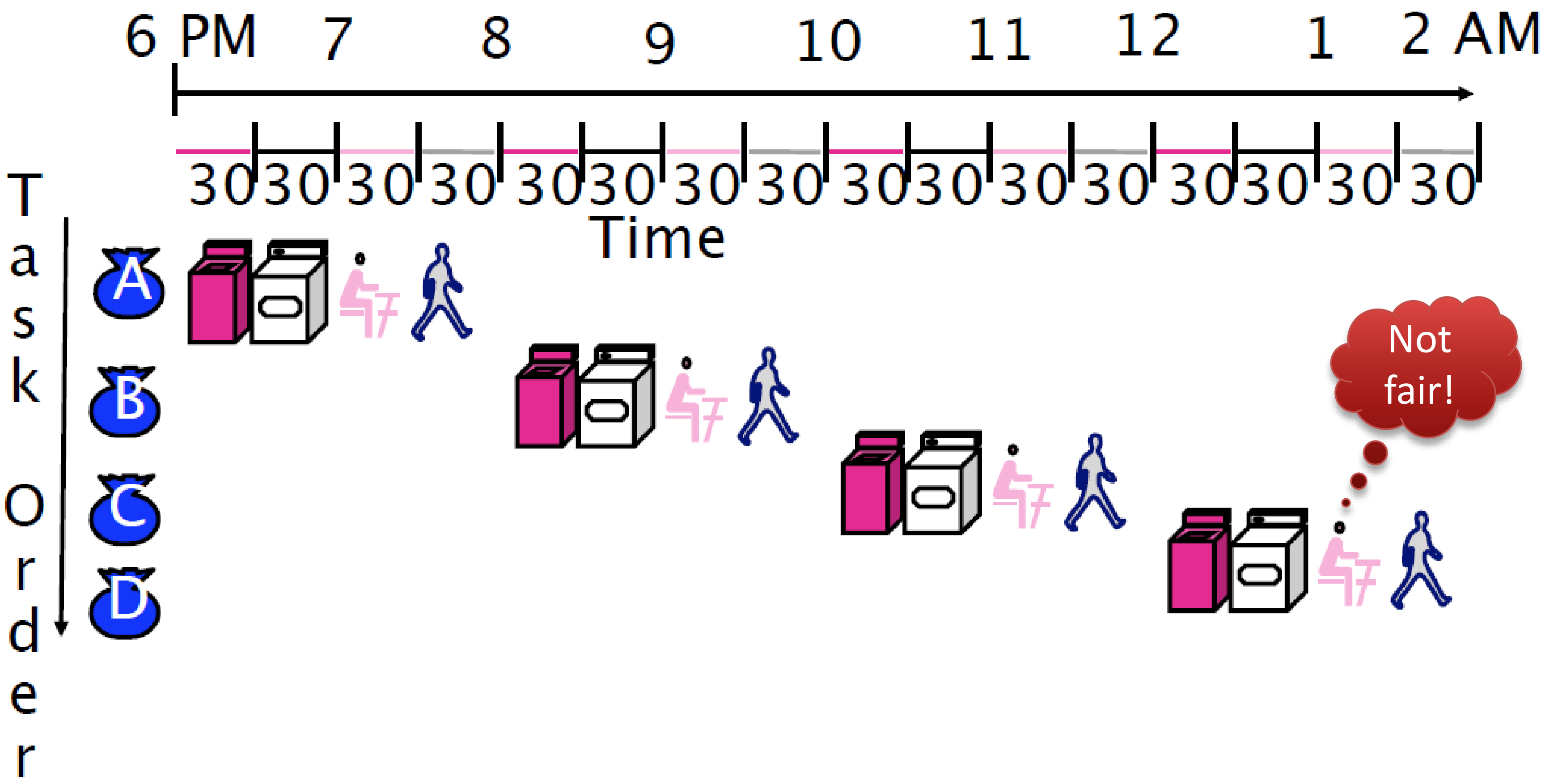
Gotta Do Laundry



- Harry, Ron, Hermione, Draco each have one load of clothes to wash, dry, fold, and put away
  - Washer takes 30 minutes
  - Dryer takes 30 minutes
  - “Folder” takes 30 minutes
  - “Stasher” takes 30 minutes to put clothes into drawers



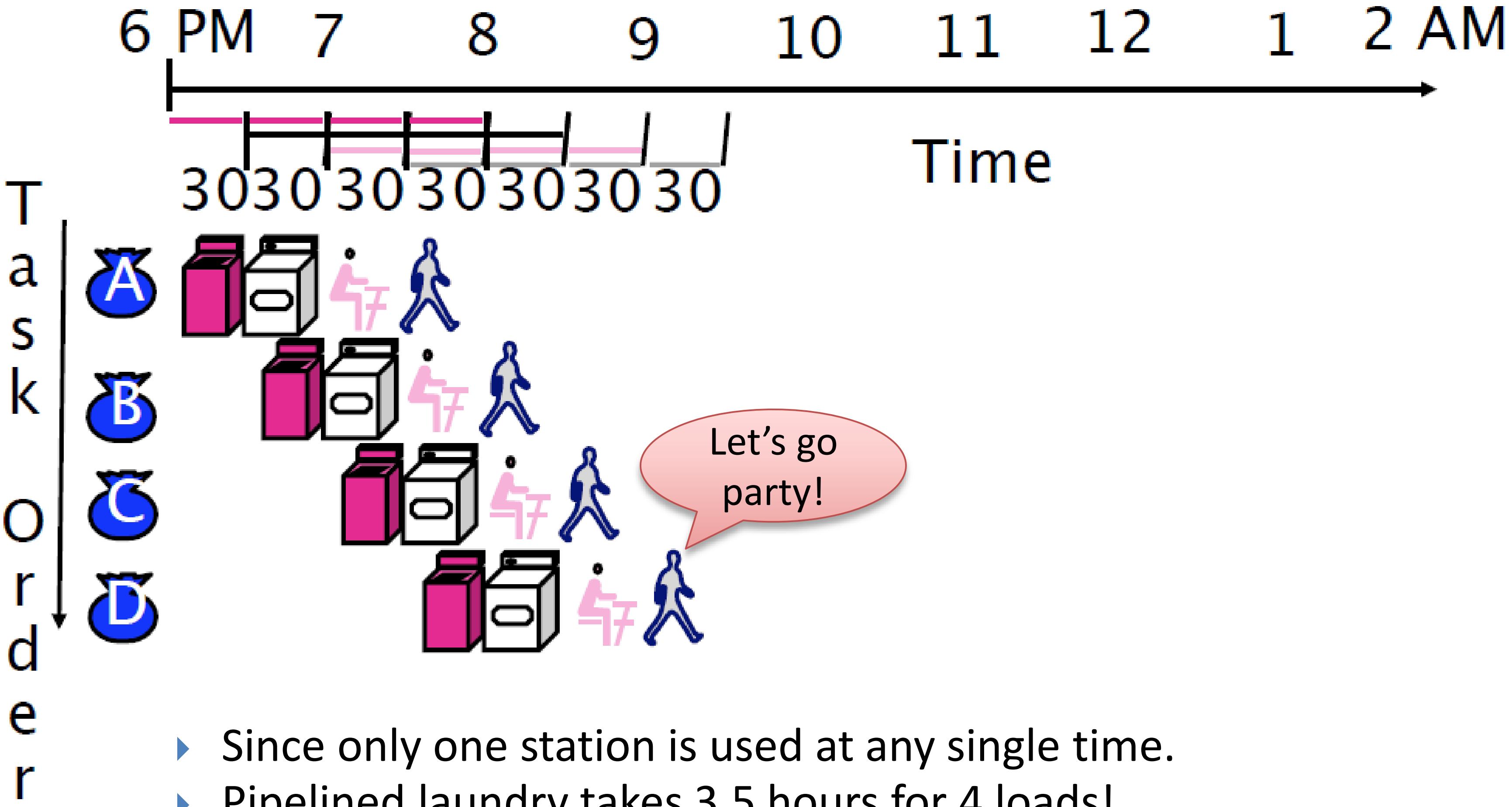
Sequential Laundry



- Sequential laundry takes 8 hours for 4 loads!

Can we speed up the process?

Pipelined Laundry



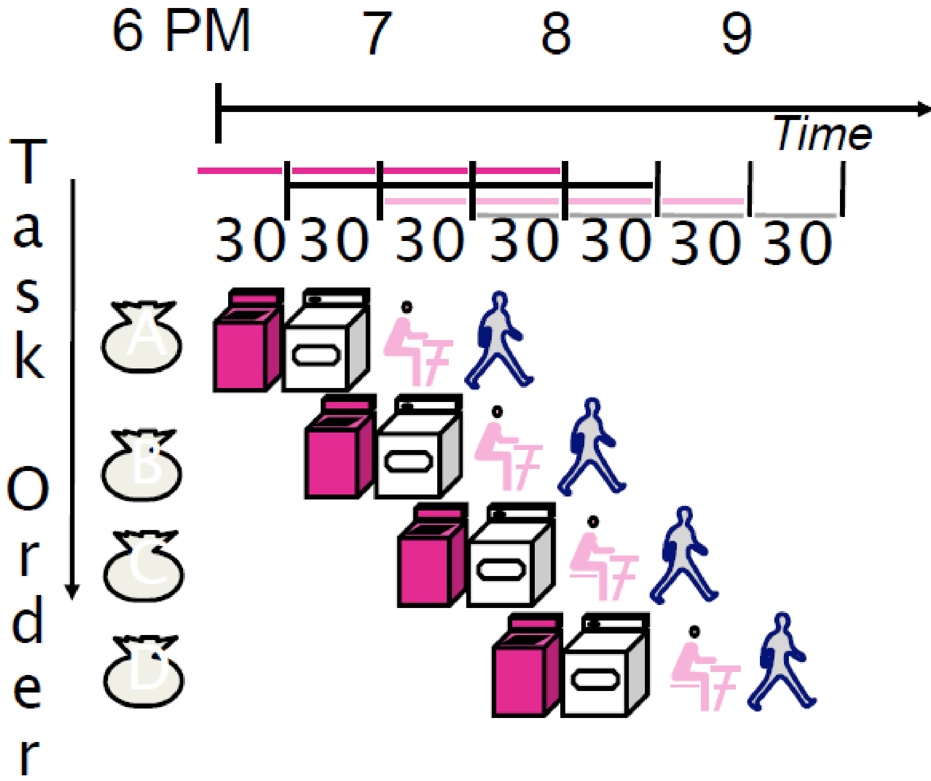
- Since only one station is used at any single time.
- Pipelined laundry takes 3.5 hours for 4 loads!
- We can apply this to processor too!

General Definitions

- Latency: time to completely execute a certain task
  - for example, time to read a sector from disk is disk access time or disk latency
  - Individual stage
- Throughput: amount of work that can be done over a period of time
  - Overall process

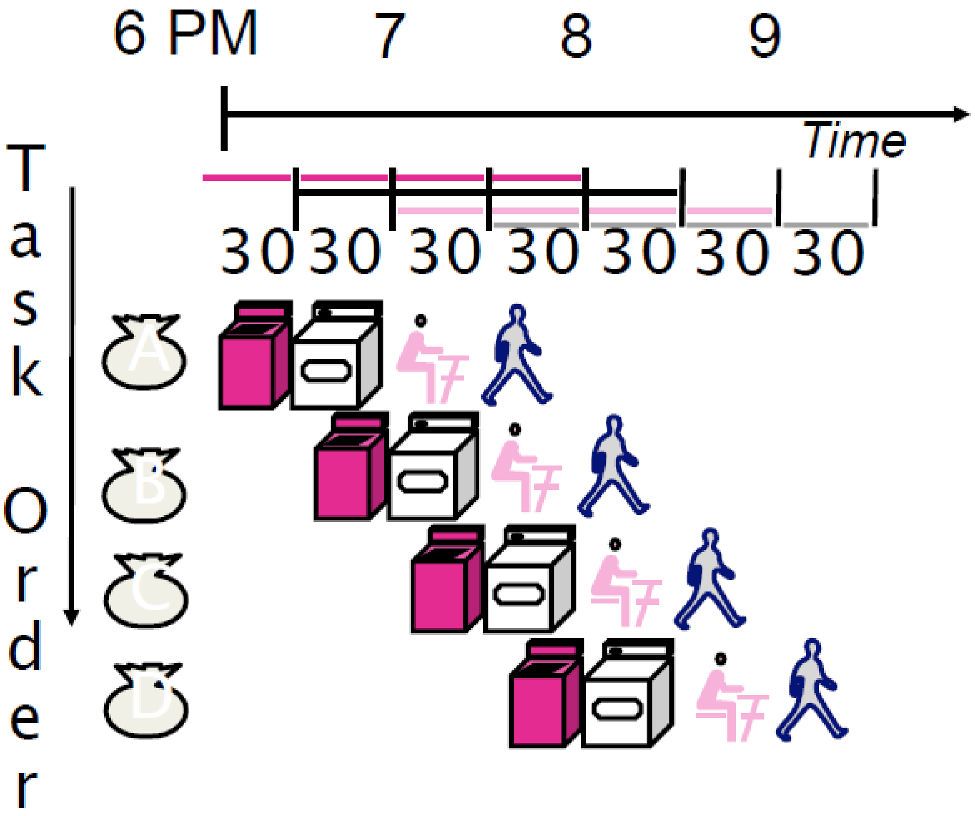
Pipelining Lessons (1/2)

- Pipelining doesn't help latency of single task, it helps throughput of entire workload
- Multiple tasks operating simultaneously using different resources
- Potential speedup = Number pipe stages
- Time to "fill" pipeline and time to "drain" it reduces speedup: 2.3X (4 loads) v. 4X (many loads) in this example



Pipelining Lessons (2/2)

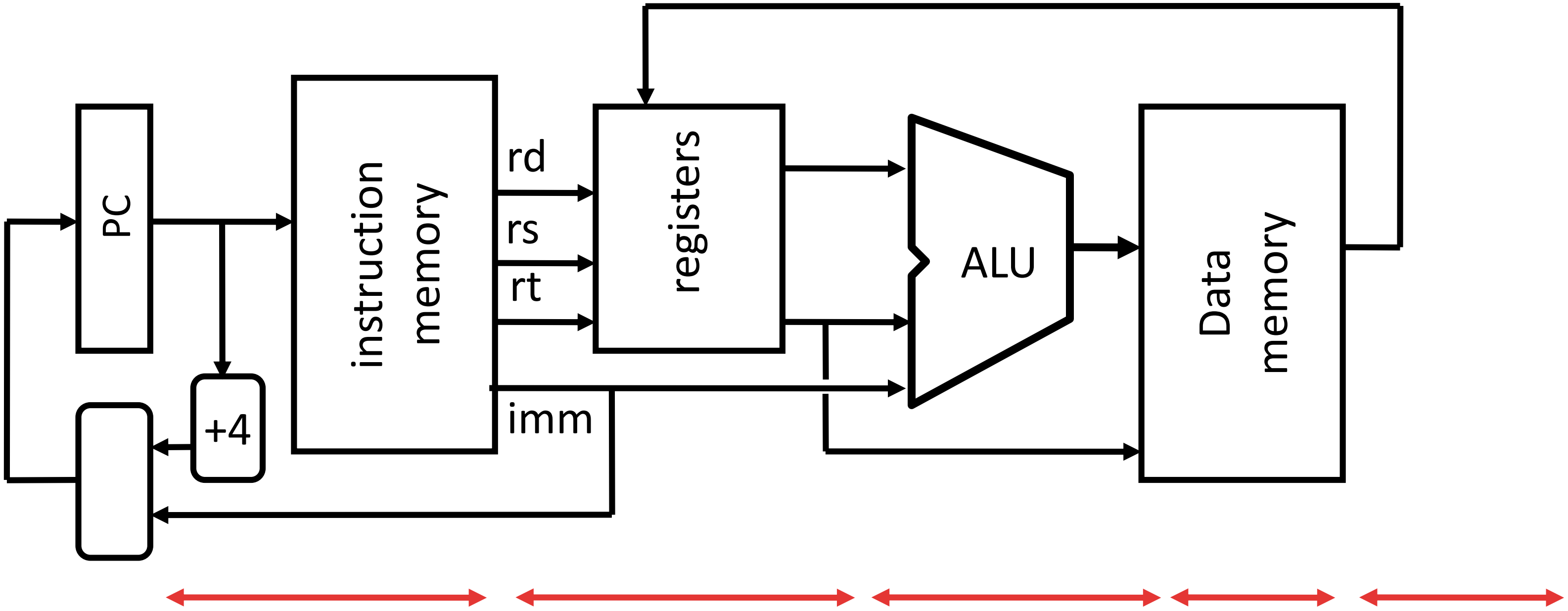
- Suppose new Washer takes 20 minutes, new Stasher takes 20 minutes. How much faster is pipeline?
- Pipeline rate limited by slowest pipeline stage
- Unbalanced lengths of pipe stages reduces speedup



Steps in Executing MIPS

1. IFtch: Instruction Fetch, Increment PC
2. Dcd: Instruction Decode, Read Registers
3. Exec:
    - Mem-ref: Calculate Address
    - Arith-log: Perform Operation
4. Mem:
    - Load: Read Data from Memory
    - Store: Write Data to Memory
5. WB: Write Data Back to Register

Redrawn Single-Cycle Datapath



1. Instruction Fetch

2. Decode/ Register Read

3. Execute

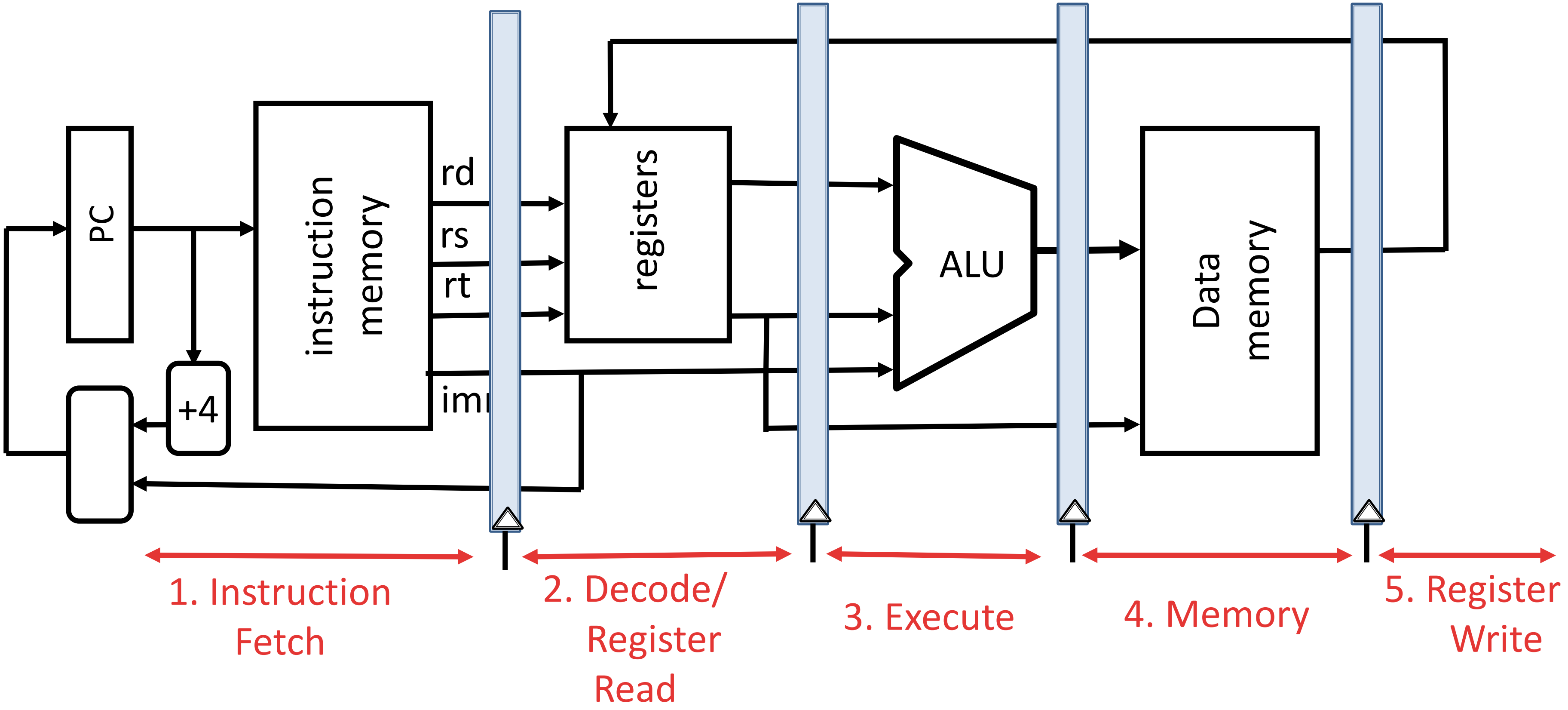
4. Memory

5. Register Write

How can we use the same datapath to process different instructions at the same time?

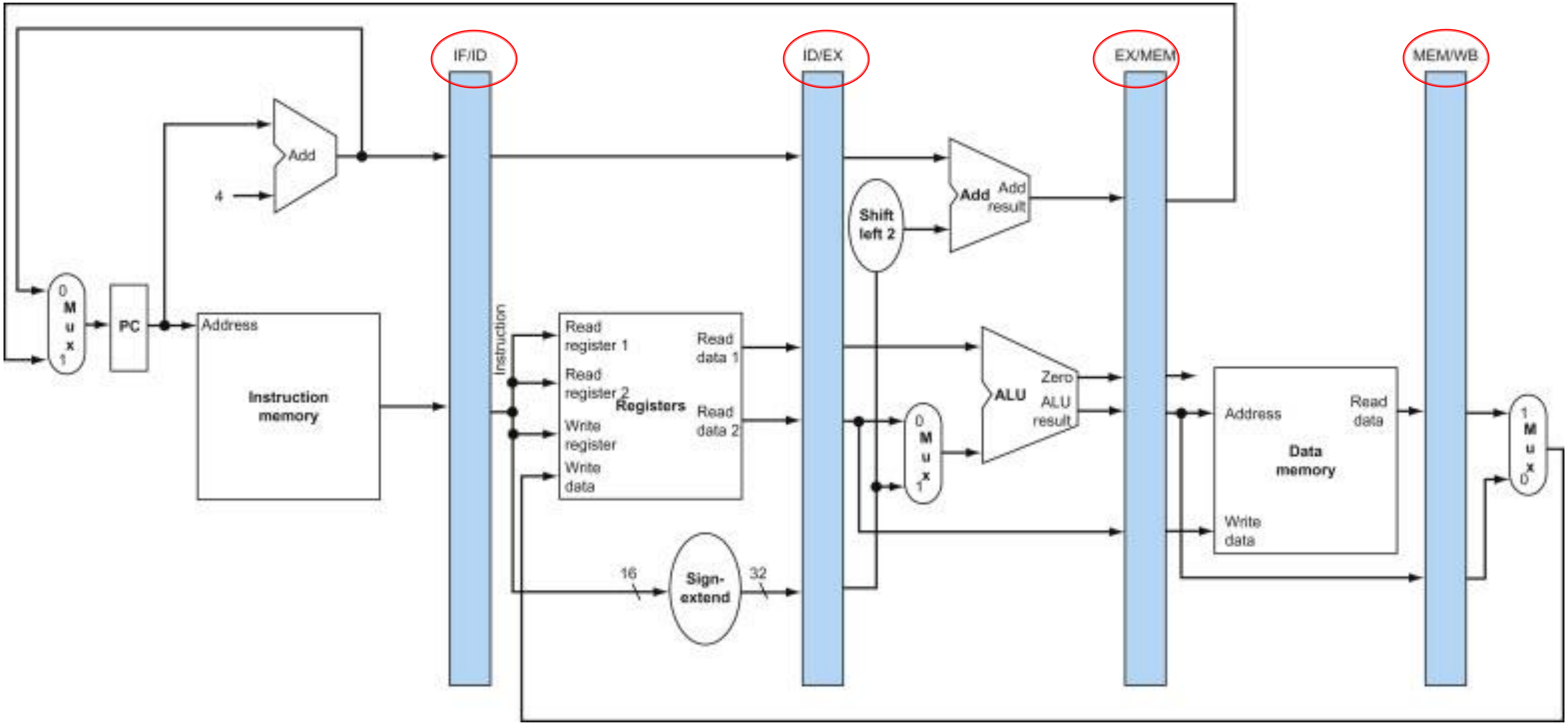
Laundry basket!

Pipelined Datapath



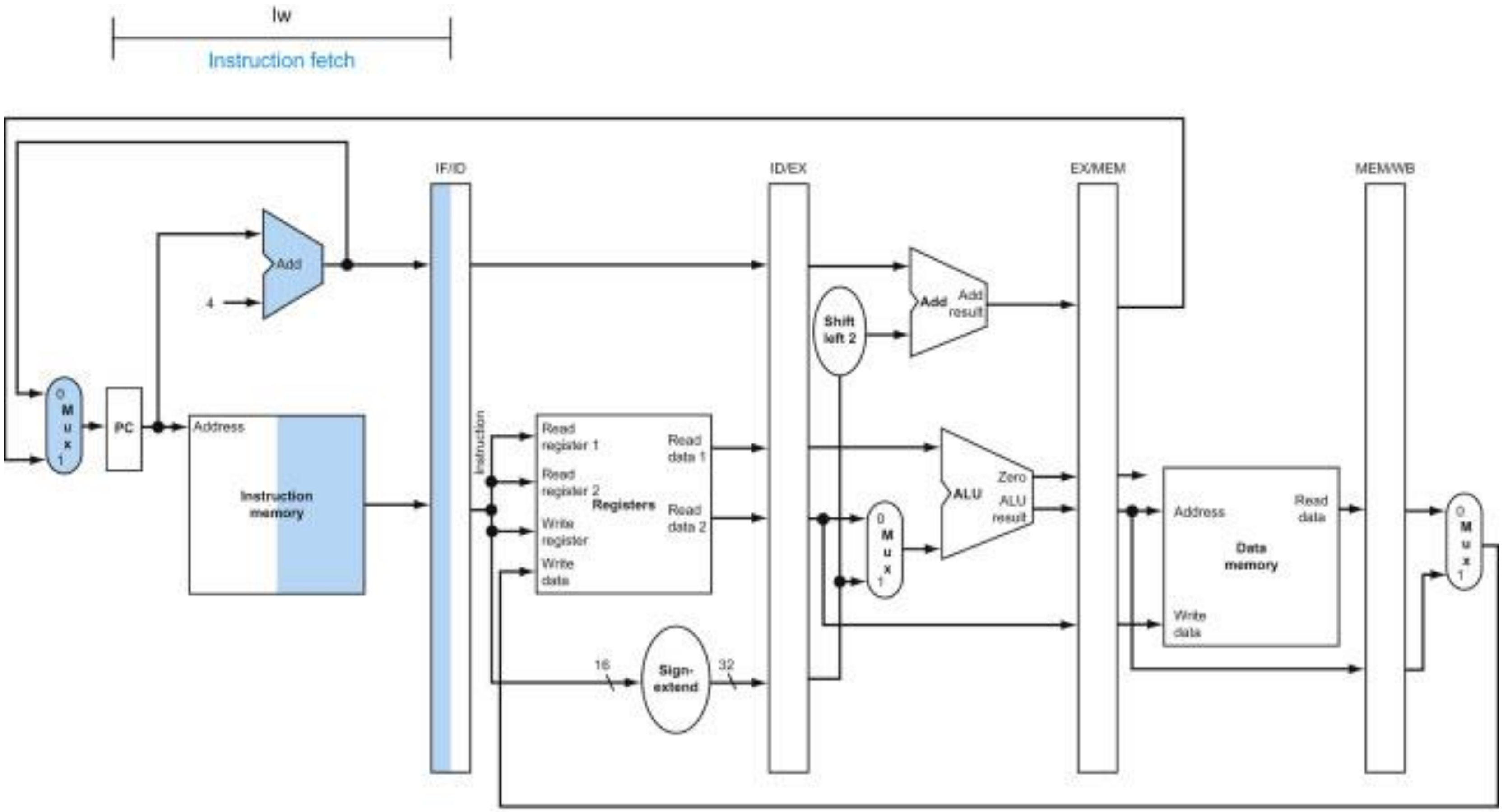
- Add registers between stages
  - Hold information produced in previous cycle
- 5 stage pipeline; clock rate potential 5X faster

More Detailed Pipeline



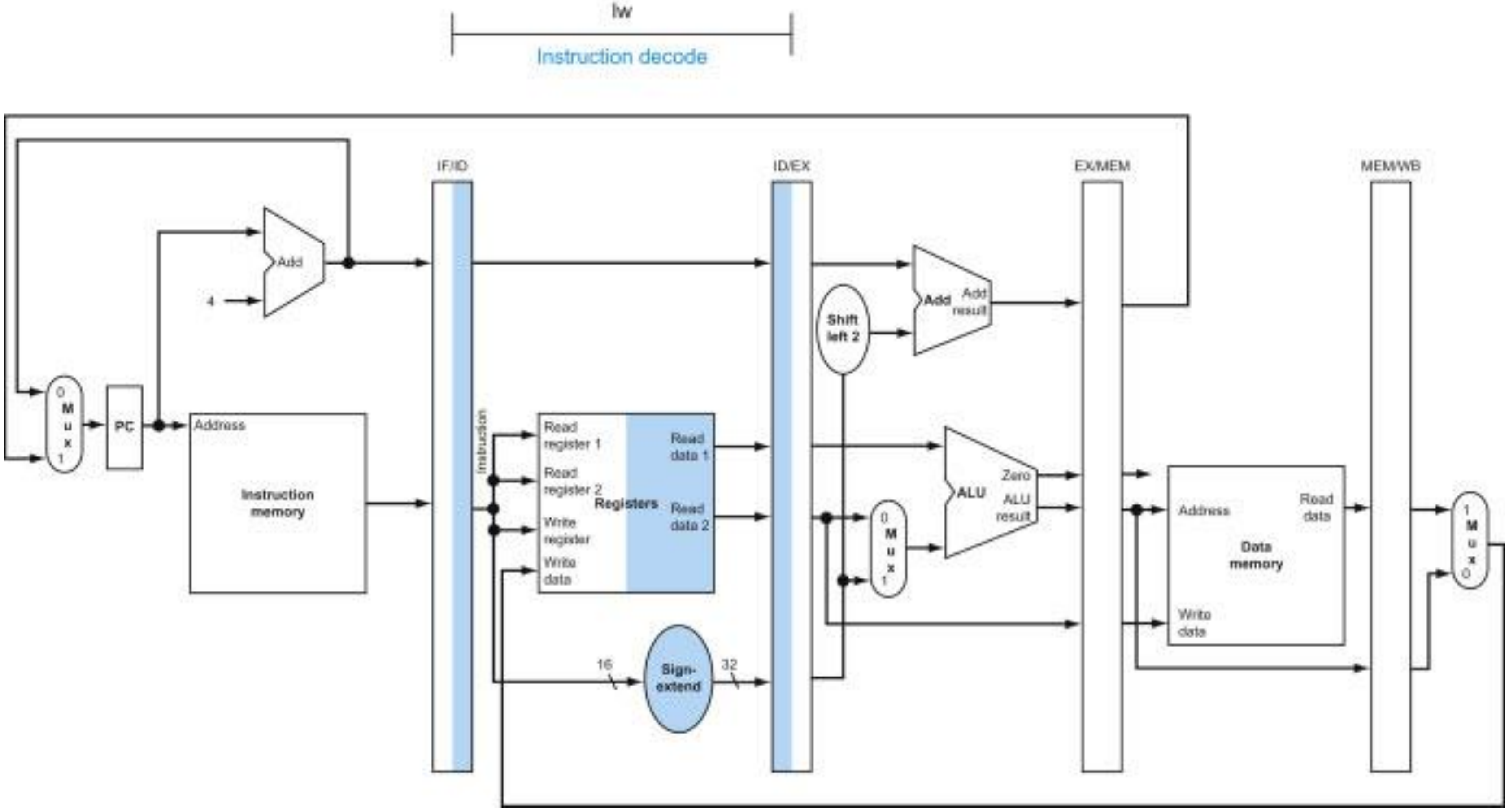
Registers named for adjacent stages, e.g., IF/ID

IF for Load, Store, …

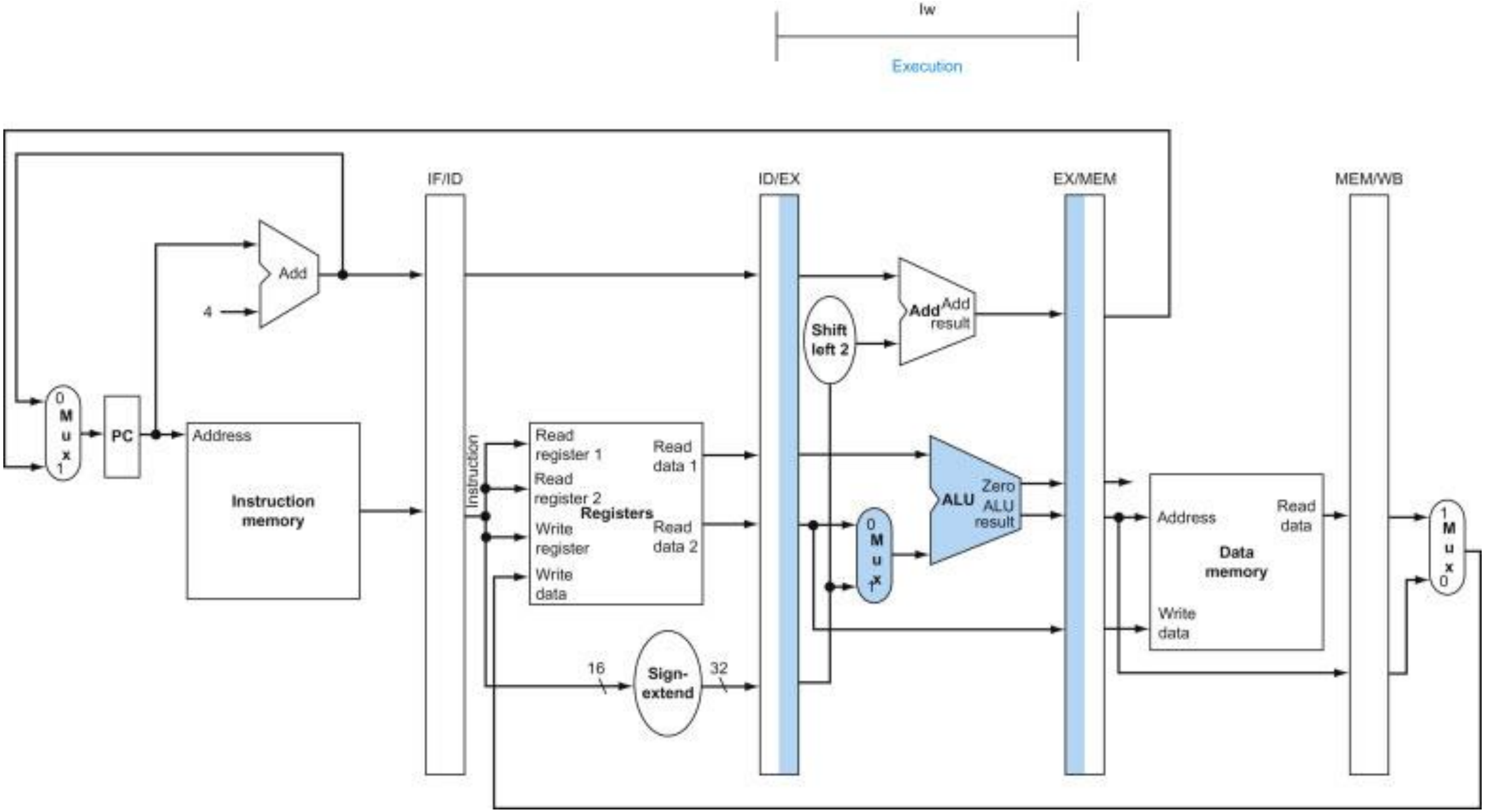


Highlight combinational logic components used + right half of state logic on read, left half on write

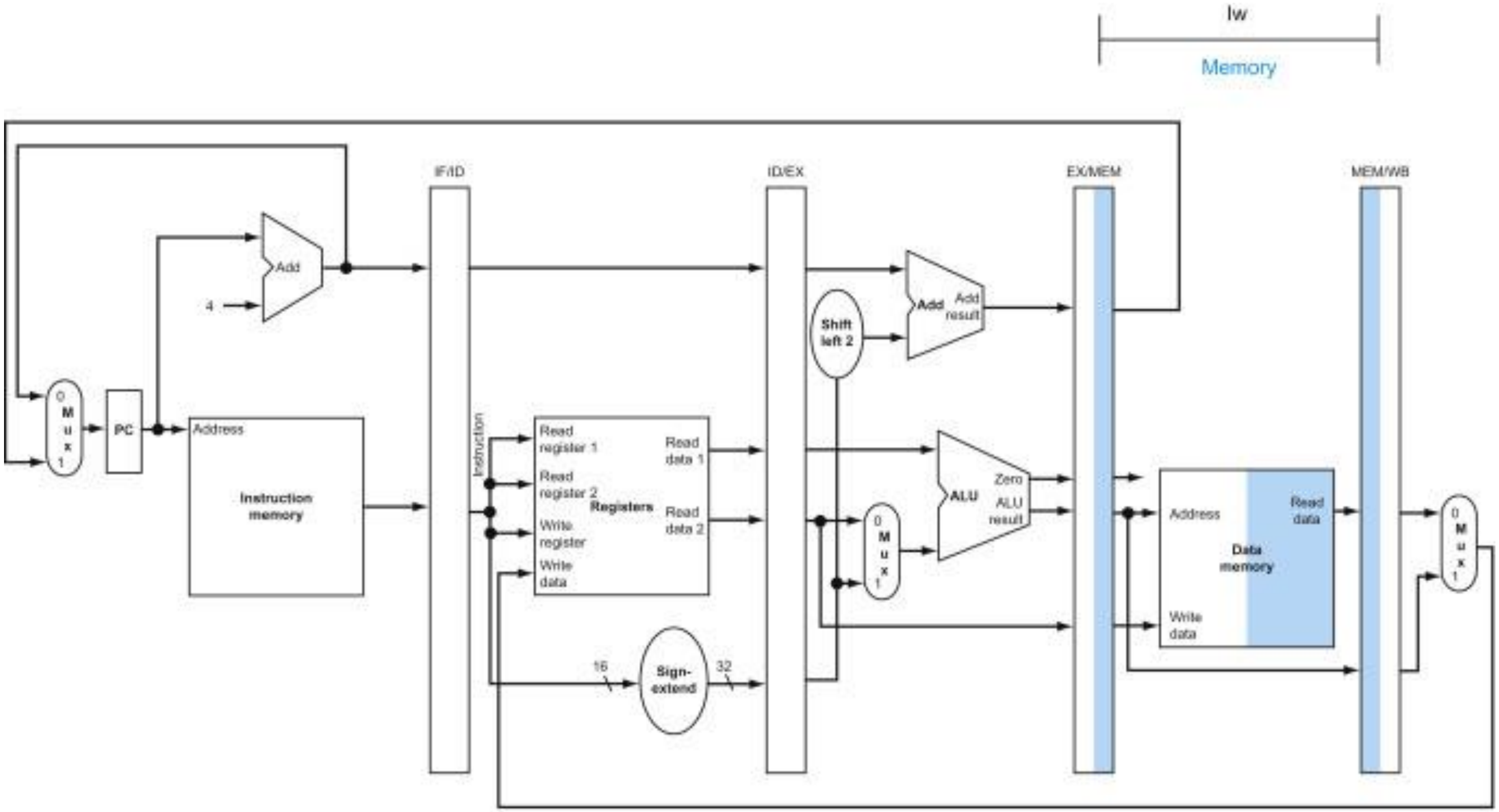
ID for Load, Store, …



EX for Load


lw
Execution
IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
0
M
u
x
1
PC
Address
Instruction memory
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
16
Sign-extend
32
Shift left 2
Add
Add result
0
M
u
x
1
ALU
Zero
ALU result
Address
Read data
Data memory
Write data
1
M
u
x
0

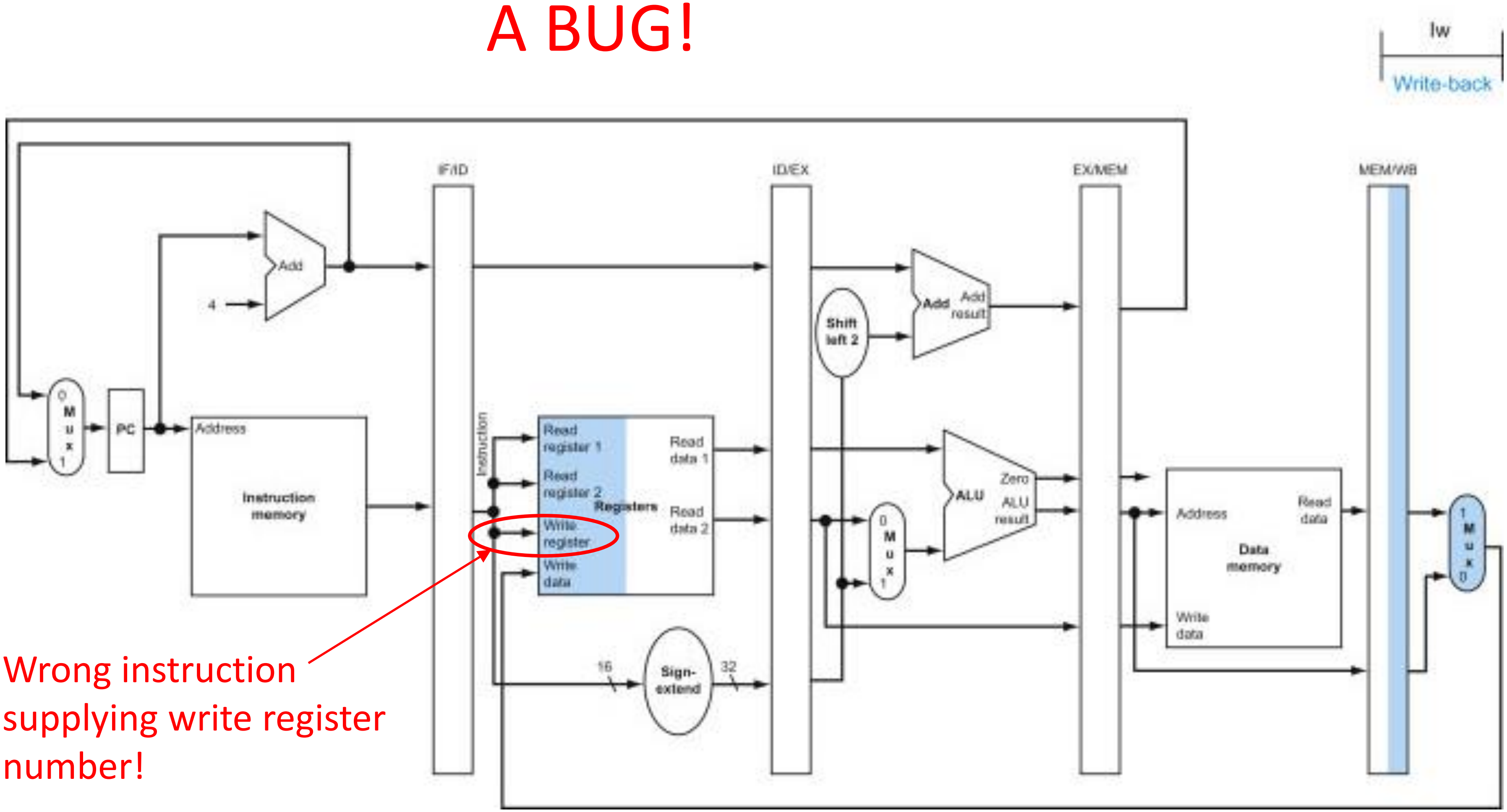
MEM for Load



lw
Memory
IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
Add
Add result
Shift left 2
0
M
u
x
1
PC
Address
Instruction memory
Instruction
Read register 1
Read register 2
Write register
Write data
Registers
Read data 1
Read data 2
16
Sign-extend
32
Zero
ALU
ALU result
Address
Read data
Data memory
Write data

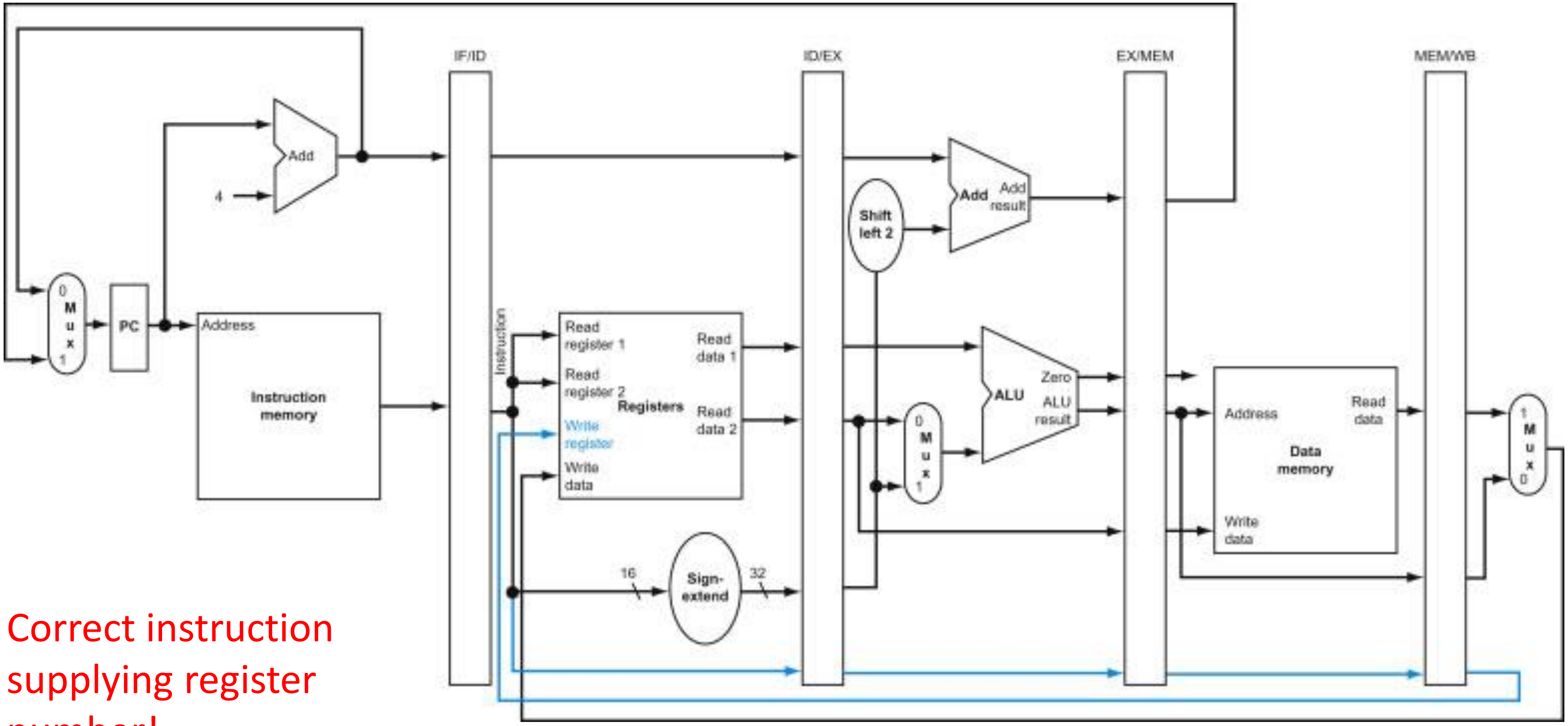
WB for Load

A BUG!



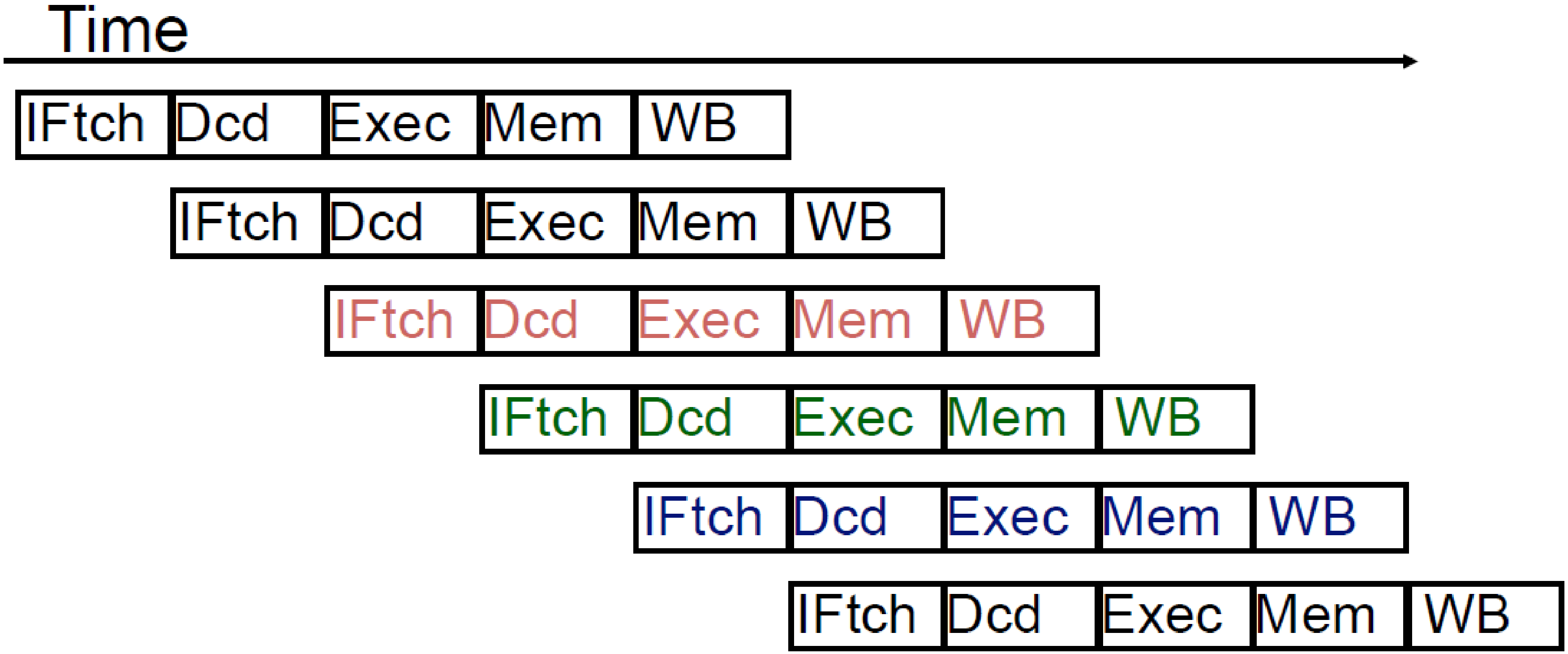
Wrong instruction supplying write register number!

Corrected Datapath for Load



Correct instruction supplying register number!

Pipelined Execution Representation



Time

IFtch Dcd Exec Mem WB

IFtch Dcd Exec Mem WB

IFtch Dcd Exec Mem WB

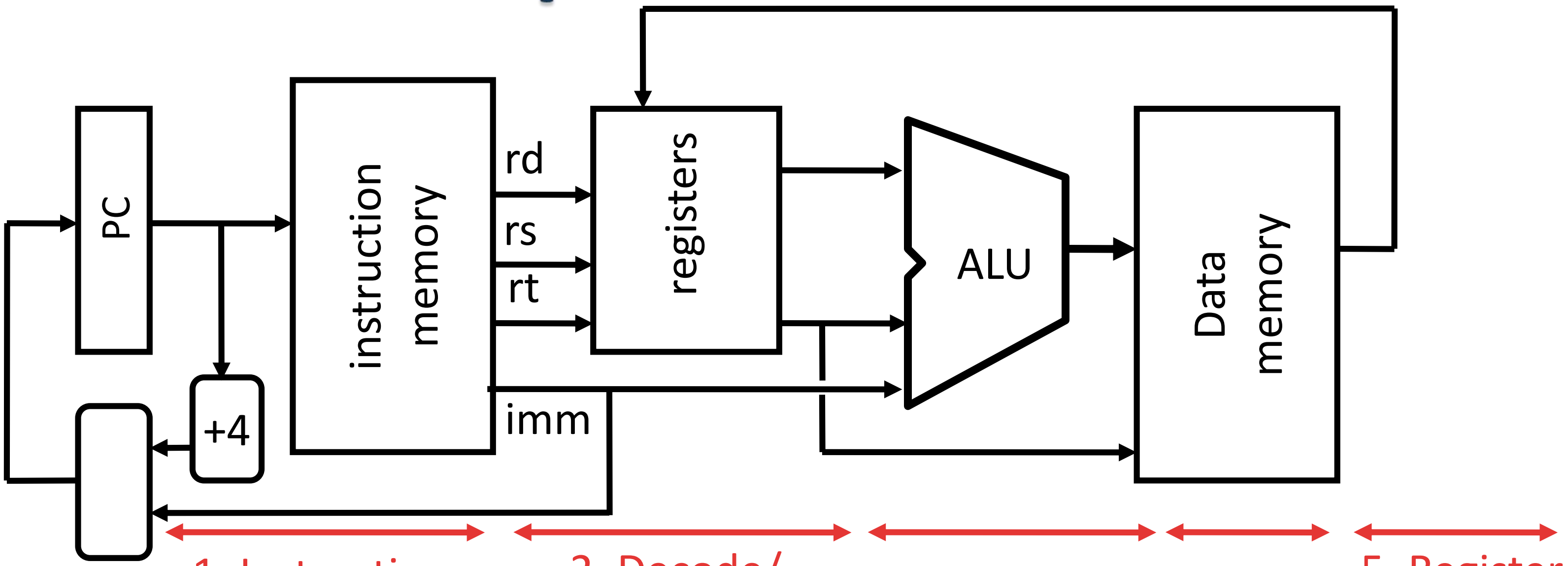
IFtch Dcd Exec Mem WB

IFtch Dcd Exec Mem WB

IFtch Dcd Exec Mem WB

- Every instruction must take same number of steps, also called pipeline “stages”, so some will go idle sometimes

Review: Datapath for MIPS



1. Instruction Fetch

2. Decode/ Register Read

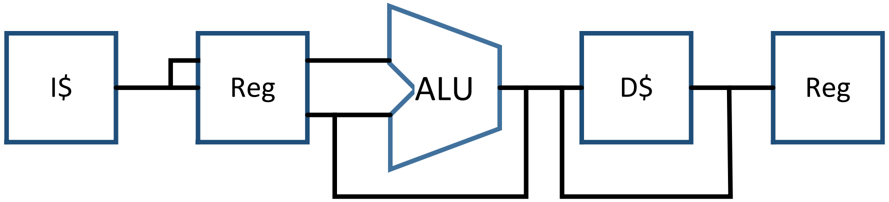
3. Execute

4. Memory

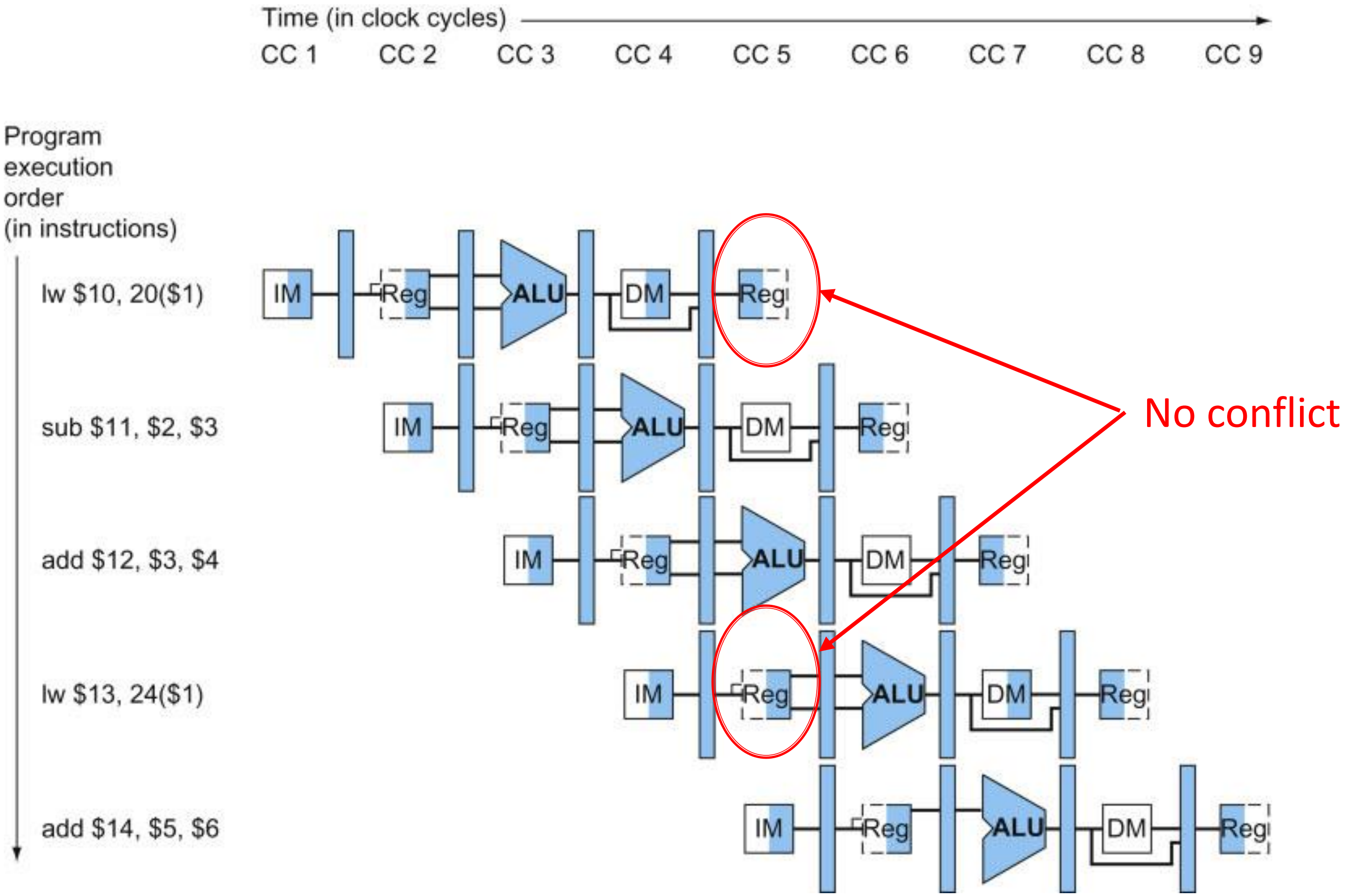
5. Register Write

Use datapath figure to represent pipeline

IFtch	Dcd	Exec	Mem	WB



Graphical Pipeline Representation



- In Reg, right half highlight read, left half write

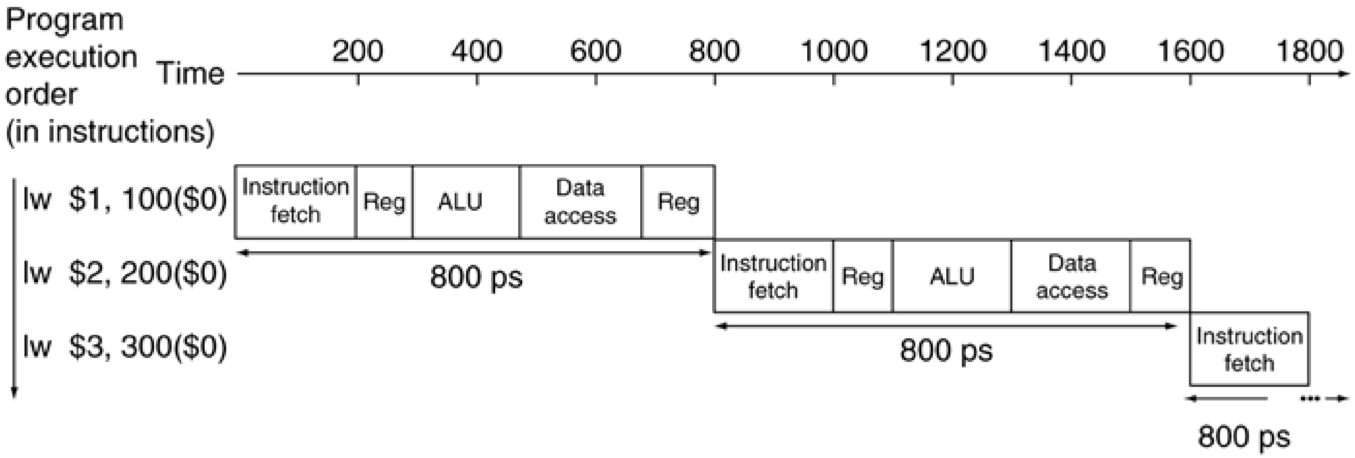
Pipeline Performance

- Assume time for stages is
  - 100 ps for register read or write
  - 200 ps for other stages
- What is pipelined clock rate?
  - Pipelined datapath vs single-cycle datapath

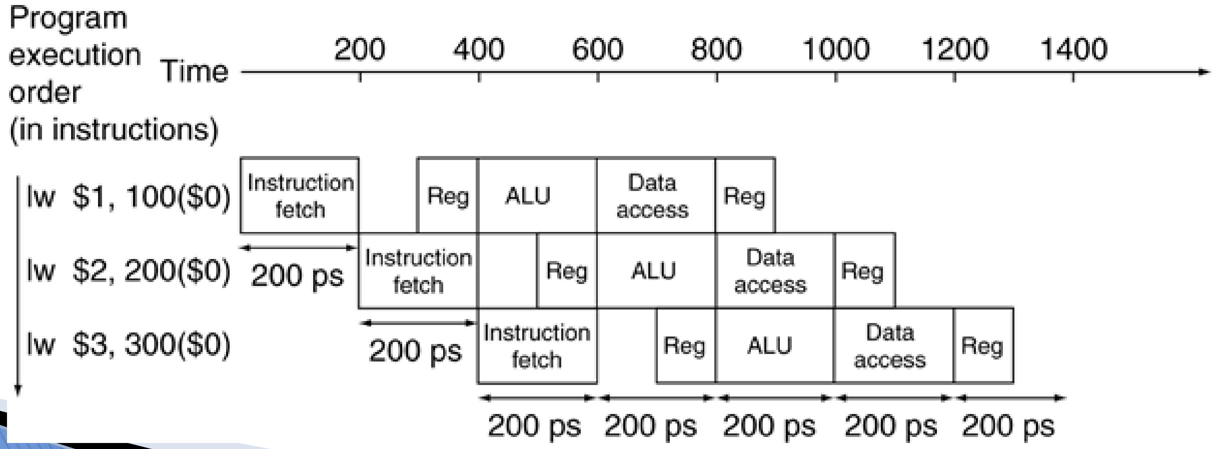
Instr	Instr fetch	Register read	ALU op	Memory access	Register write	Total time
lw	200 ps	100 ps	200 ps	200 ps	100 ps	800 ps
sw	200 ps	100 ps	200 ps	200 ps		700 ps
R-format	200 ps	100 ps	200 ps		100 ps	600 ps
beq	200 ps	100 ps	200 ps			500 ps

Pipeline Performance

Single-cycle (T_c= 800ps or f_c = 1.25GHz)



Pipelined (T_c= 200ps or f_c = 5GHz)


Program execution order (in instructions)
Time
200
400
600
800
1000
1200
1400
lw $1, 100($0)
Instruction fetch
Reg
ALU
Data access
Reg
lw $2, 200($0)
200 ps
Instruction fetch
Reg
ALU
Data access
Reg
lw $3, 300($0)
200 ps
Instruction fetch
Reg
ALU
Data access
Reg
200 ps
200 ps
200 ps
200 ps
200 ps