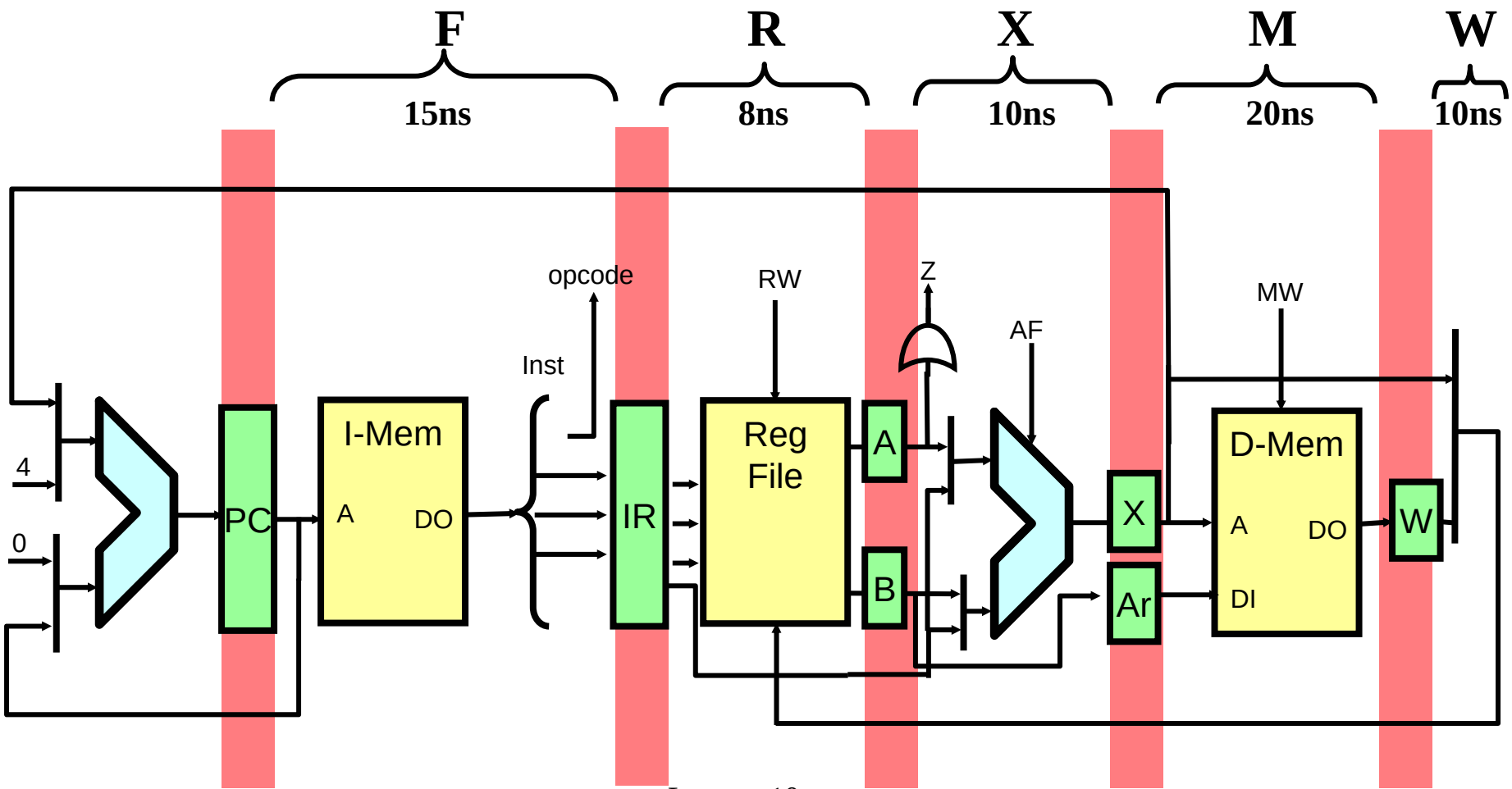
Lecture 10: Pipelining

- Last Time
  - Datapath organization
  - Introduction to pipelining
- Today
  - Pipelining overview, hazards

Can Re-insert Datapath Flip-Flops

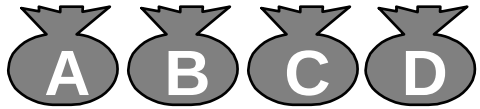
"Multicycle Execution"

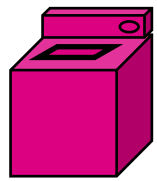
Instruction execution time = ? Clock frequency = ?

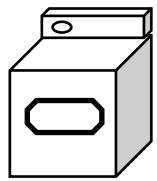


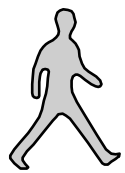
Pipelining is Natural!

- Laundry Example
- Ann, Brian, Cathy, Dave each have one load of clothes to wash, dry, and fold
- Washer takes 30 minutes
- Dryer takes 30 minutes
- "Folder" takes 30 minutes
- "Stasher" takes 30 minutes to put clothes into drawers

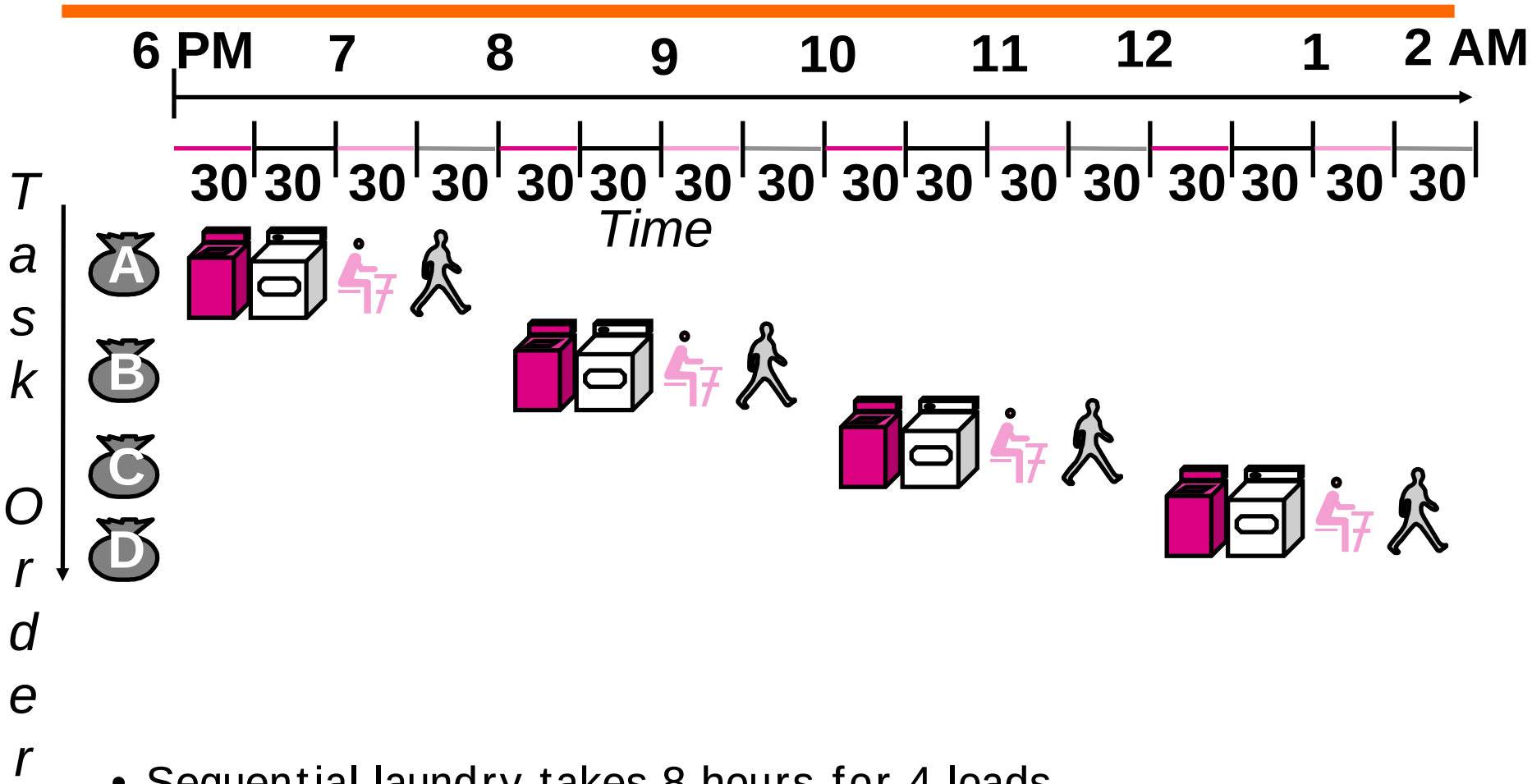






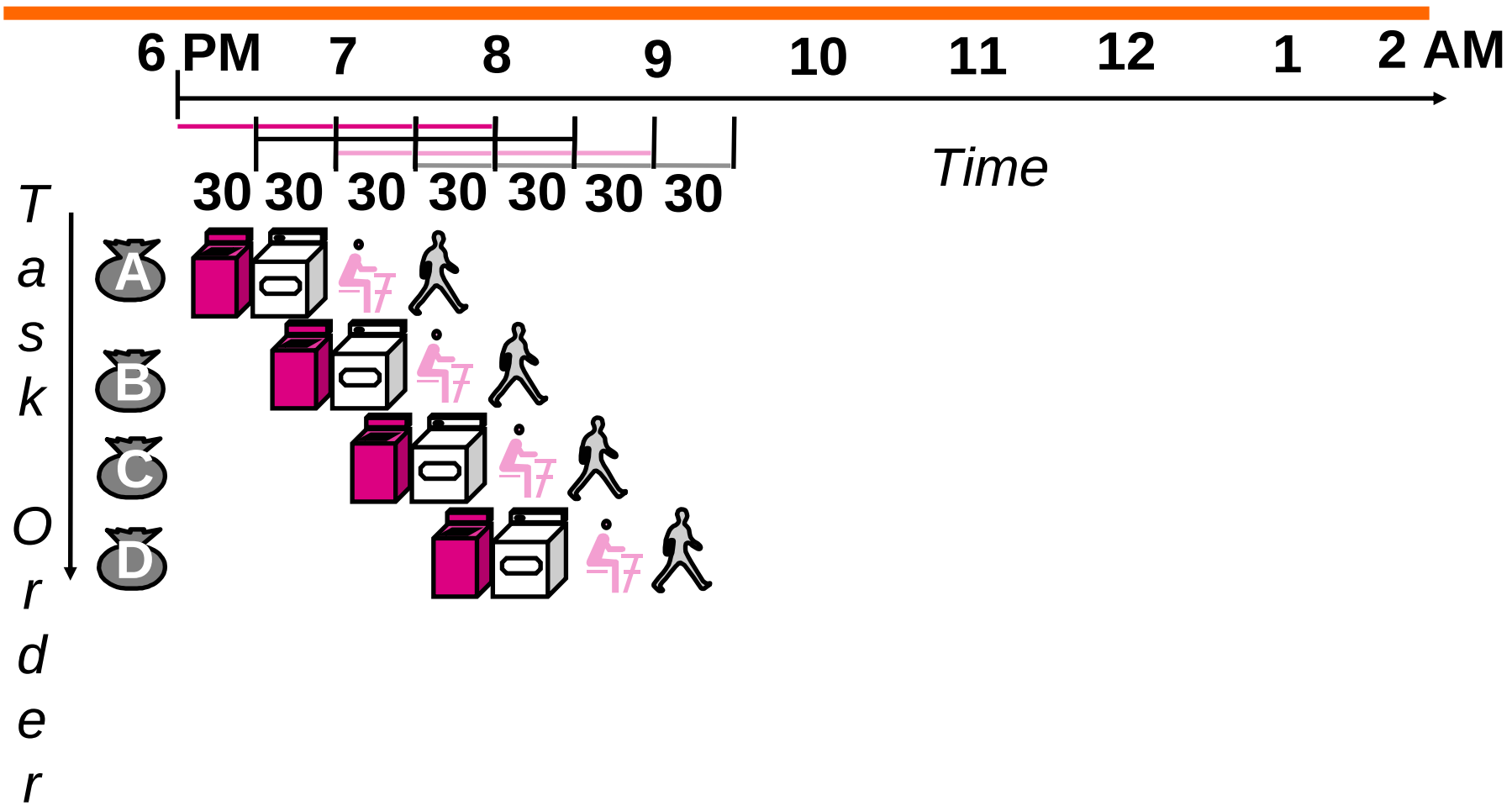


Sequential Laundry



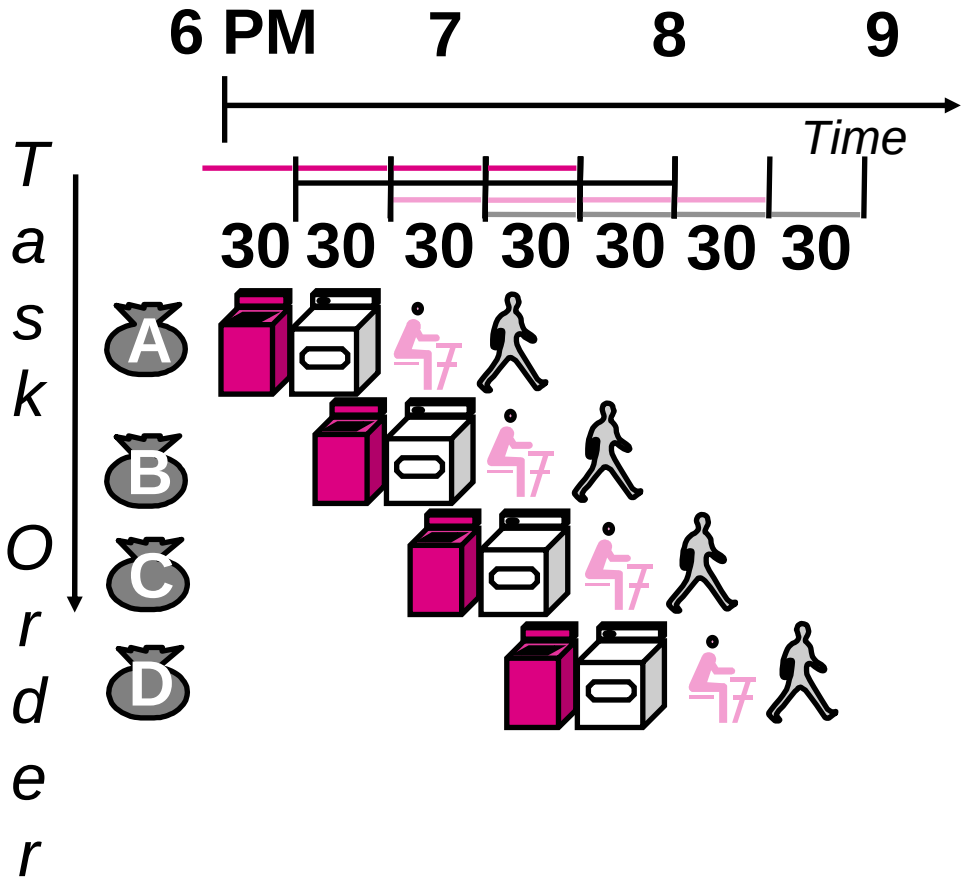
- Sequential laundry takes 8 hours for 4 loads
- If they learned pipelining, how long would laundry take?

Pipelined Laundry: Start work ASAP



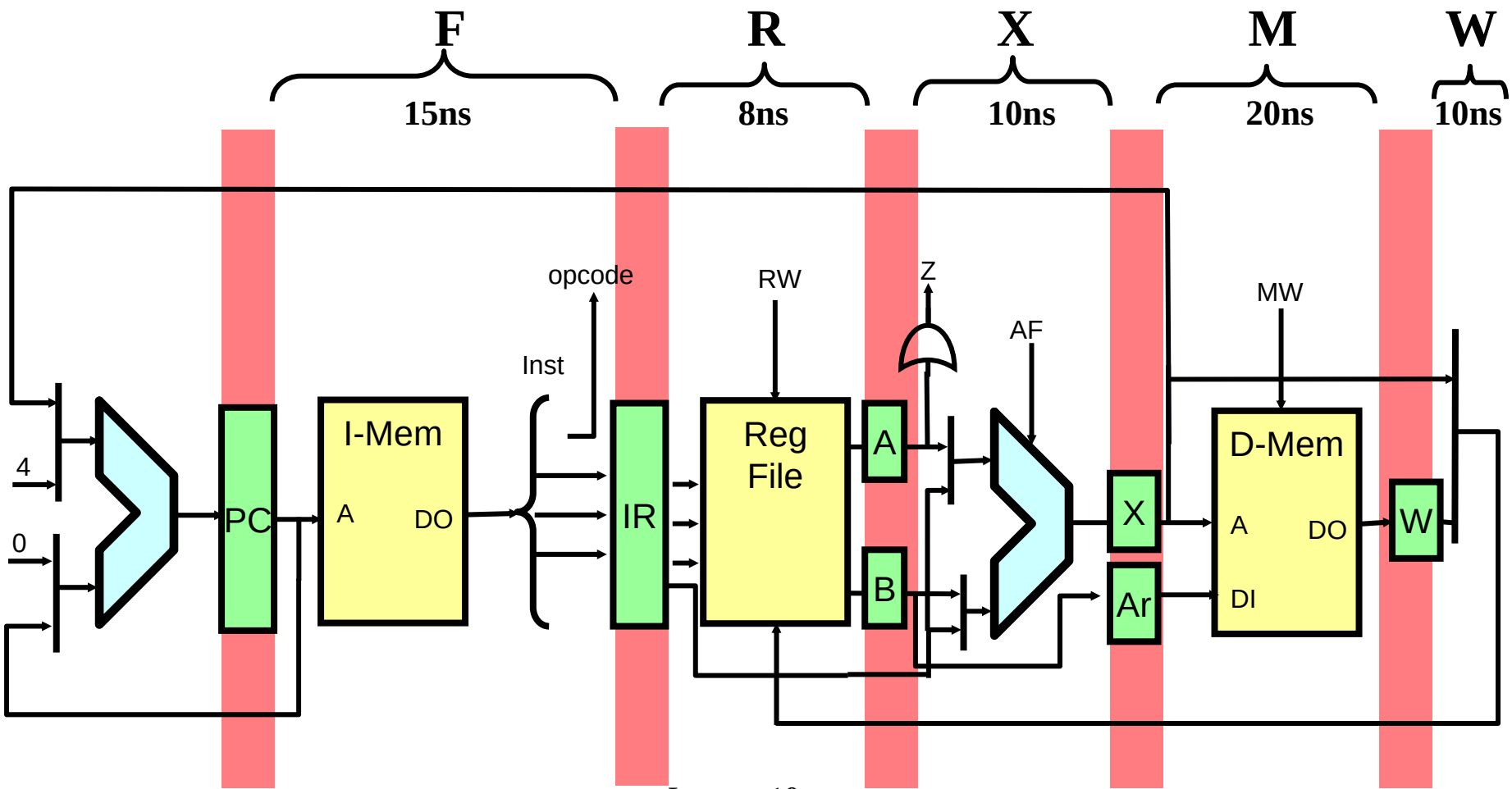
- Pipelined laundry takes 3.5 hours for 4 loads!

Pipelining Lessons

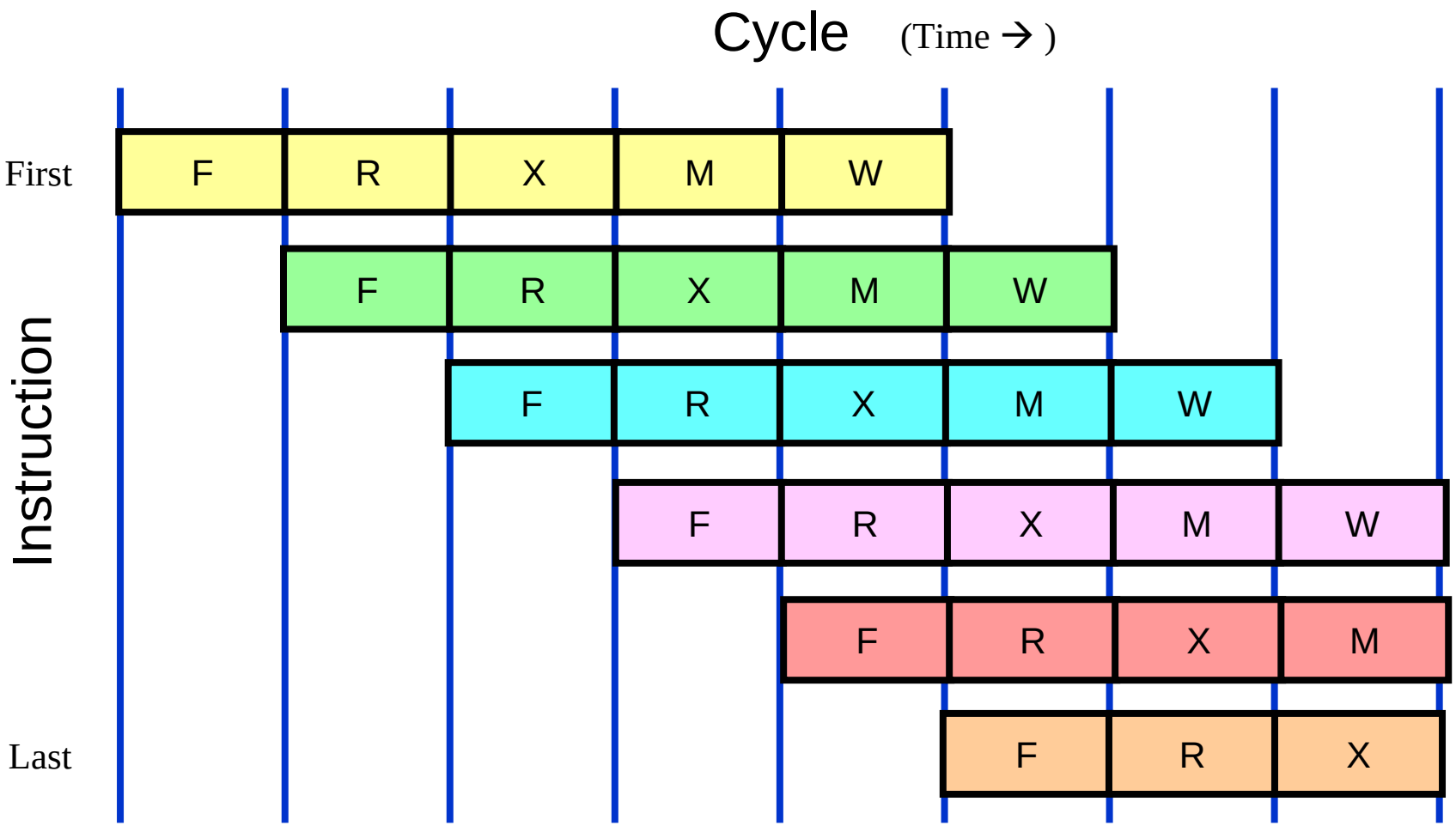


- Pipelining doesn't help latency of single task, it helps throughput of entire workload
- Multiple tasks operating simultaneously using different resources
- Potential speedup = Number pipe stages
- Pipeline rate limited by slowest pipeline stage
- Unbalanced lengths of pipe stages reduces speedup
-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

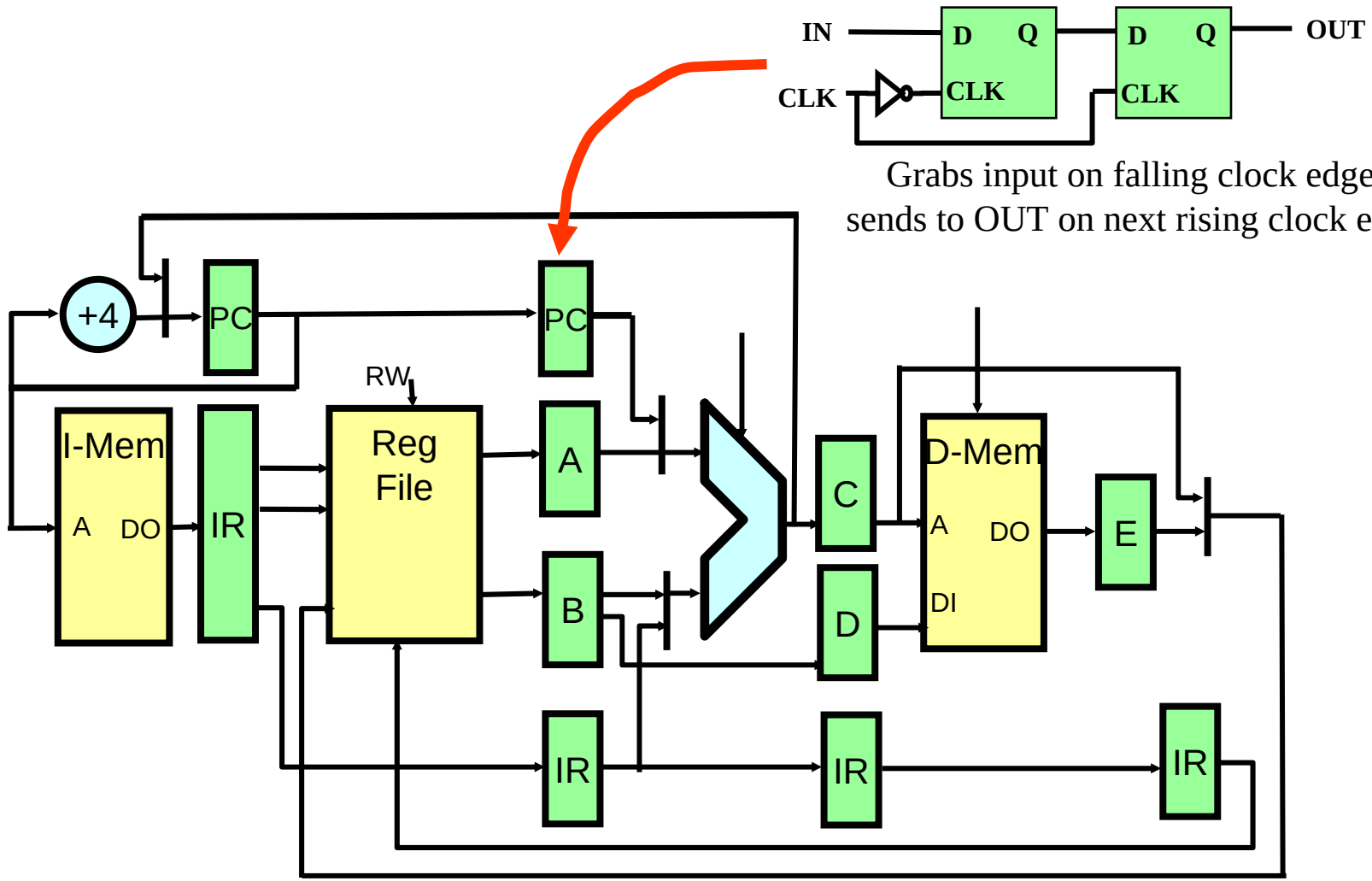
The 5-stage pipeline



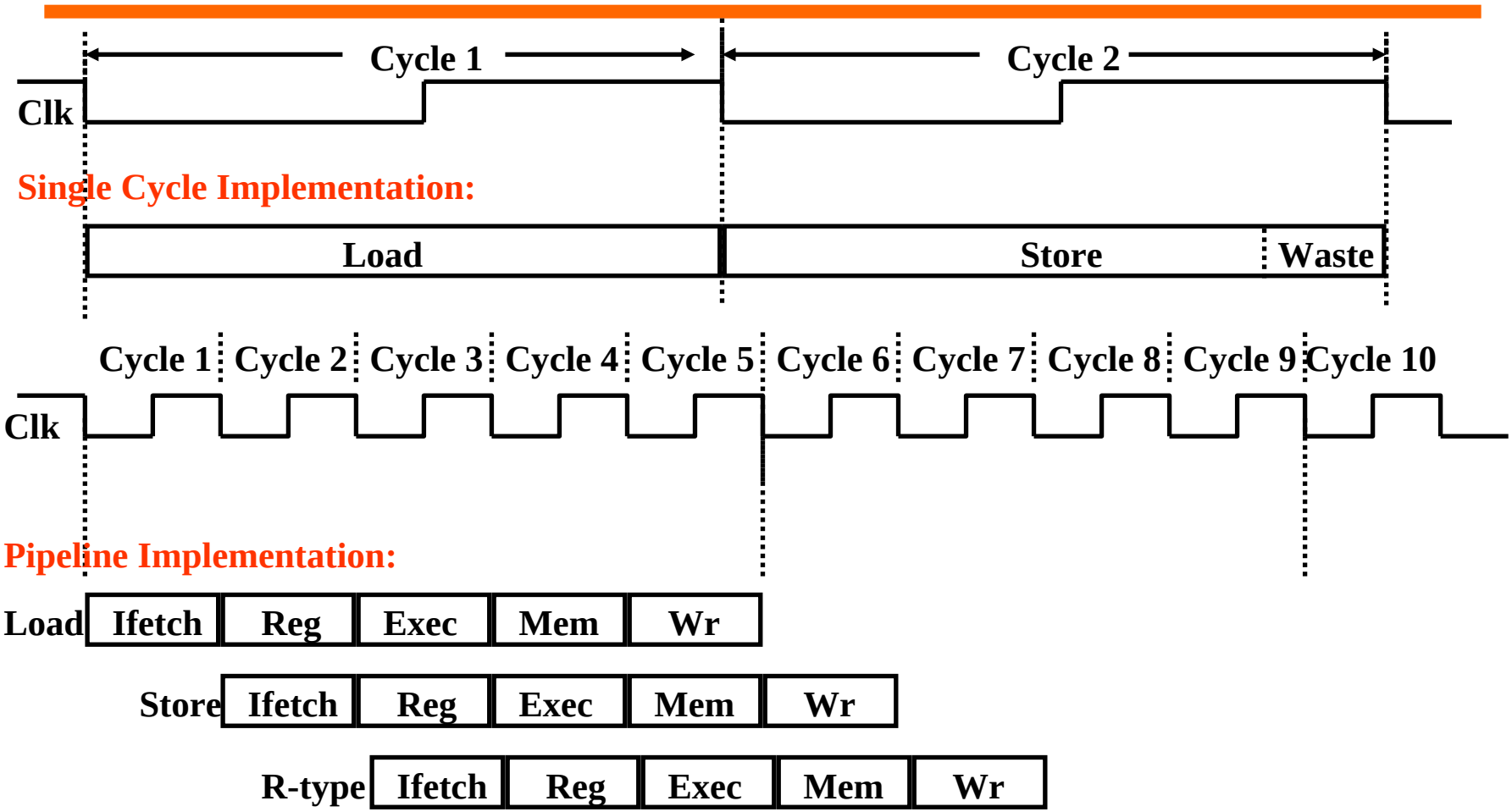
5-stage pipeline for several instructions



Pipelined Implementation – add register every "t" ns

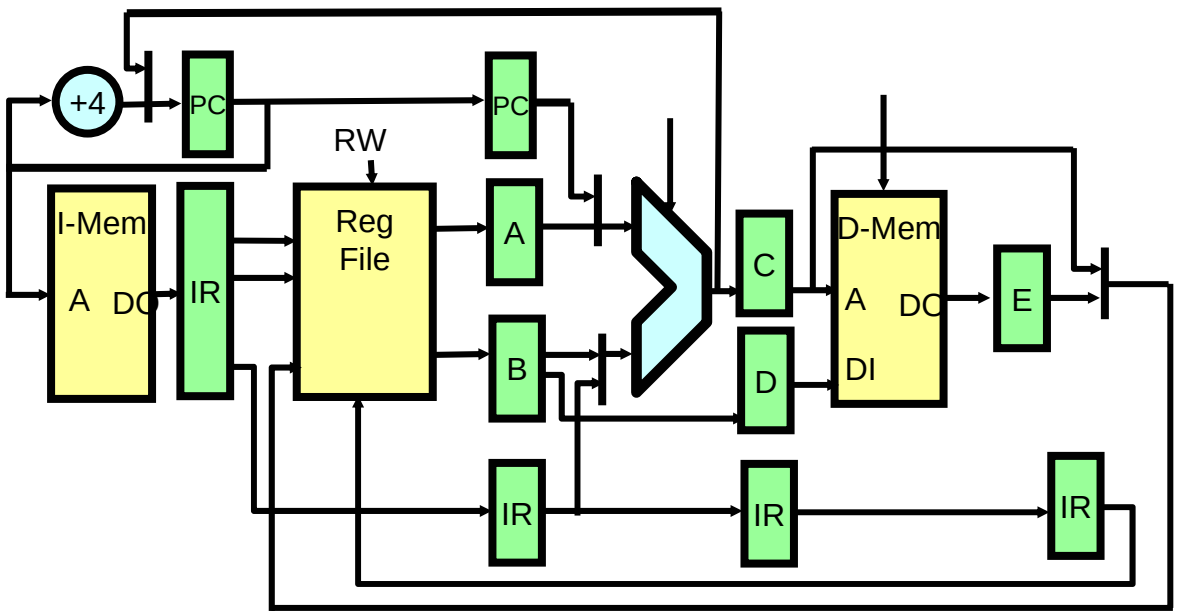


Single Cycle vs. Pipeline



Reservation Table for a Simple Pipeline

	CYCLE				
	1	2	3	4	5
I-MEM	X				
REGS-R		X			
ALU			X		
D-MEM				X	
REGS-W					X



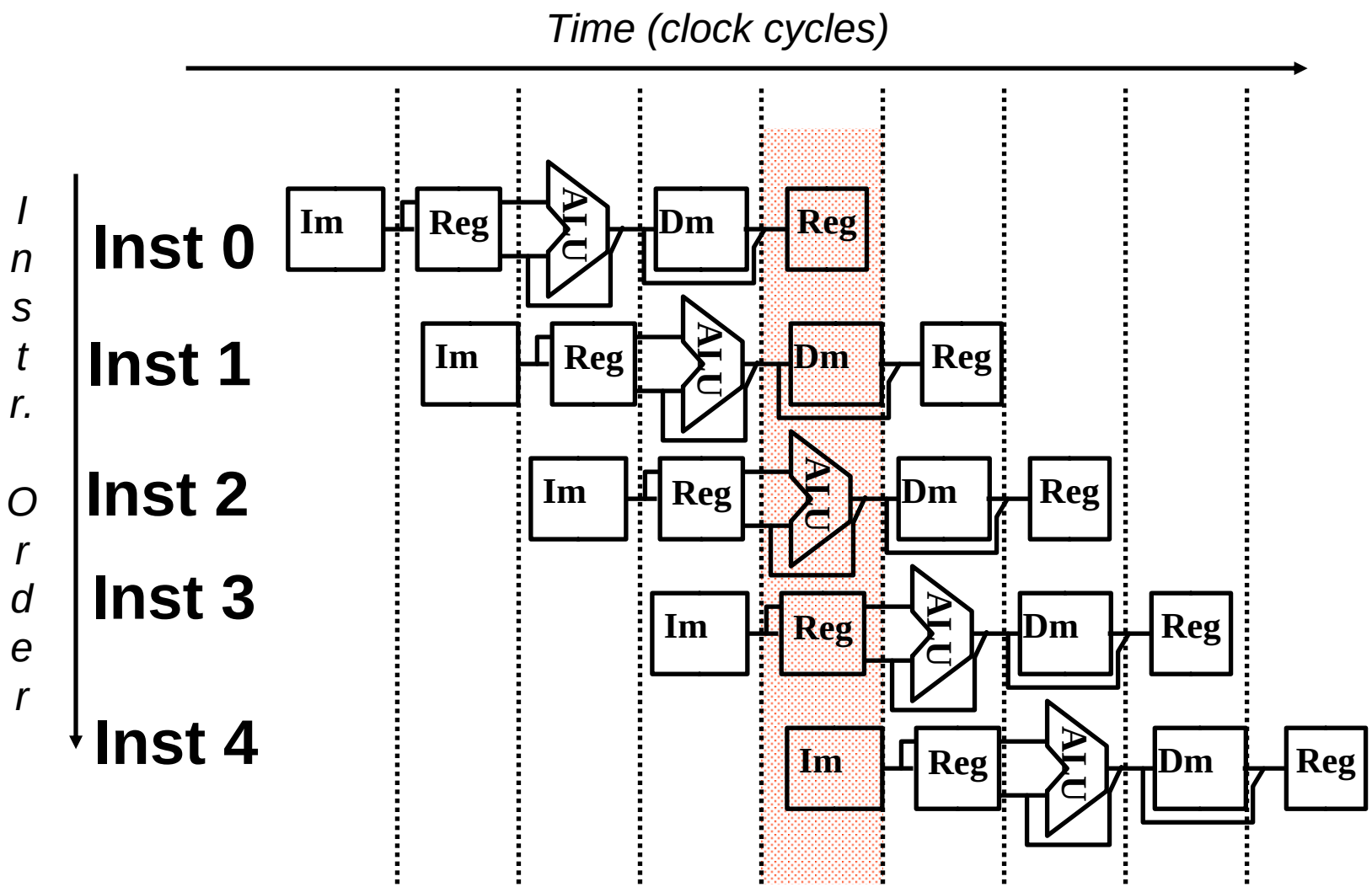
Control signals must be piplined too

(Figure 6.30, pg. 470 from textbook)

Why Pipeline?

- Suppose we execute 100 instructions
- Single Cycle Machine
  - 45 ns/cycle x 1 CPI x 100 inst = 4500 ns
- Ideal pipelined machine
  - 10 ns/cycle x (1 CPI x 100 inst + 4 cycle drain) = 1040 ns

Why Pipeline? Because the resources are there!



Benefits of Pipelining

- Before pipelining:
  - Throughput: 1 instruction per cycle

$$t_{clk} = t_F + t_R + t_X + t_M + t_W$$

  - (or lower cycle time and CPI=5)

- After pipelining (multiple instructions in pipe at one time)
  - Throughput: 1 instruction per cycle

$$t_{clk} = \max(t_F, t_R, t_X, t_M, t_W) + t_{latch}$$

Pipelining Rules

- Forward travelling signals at each stage are latched
- Only perform logic on signals in the same stage
  - signal labeling useful to prevent errors,
    - e.g., IR_R, IR_A, IR_M, IR_W
- Backward travelling signals at each stage represent *hazards*

Pipeline Hazards

- Data hazards
  - an instruction uses the result of a previous instruction (RAW)

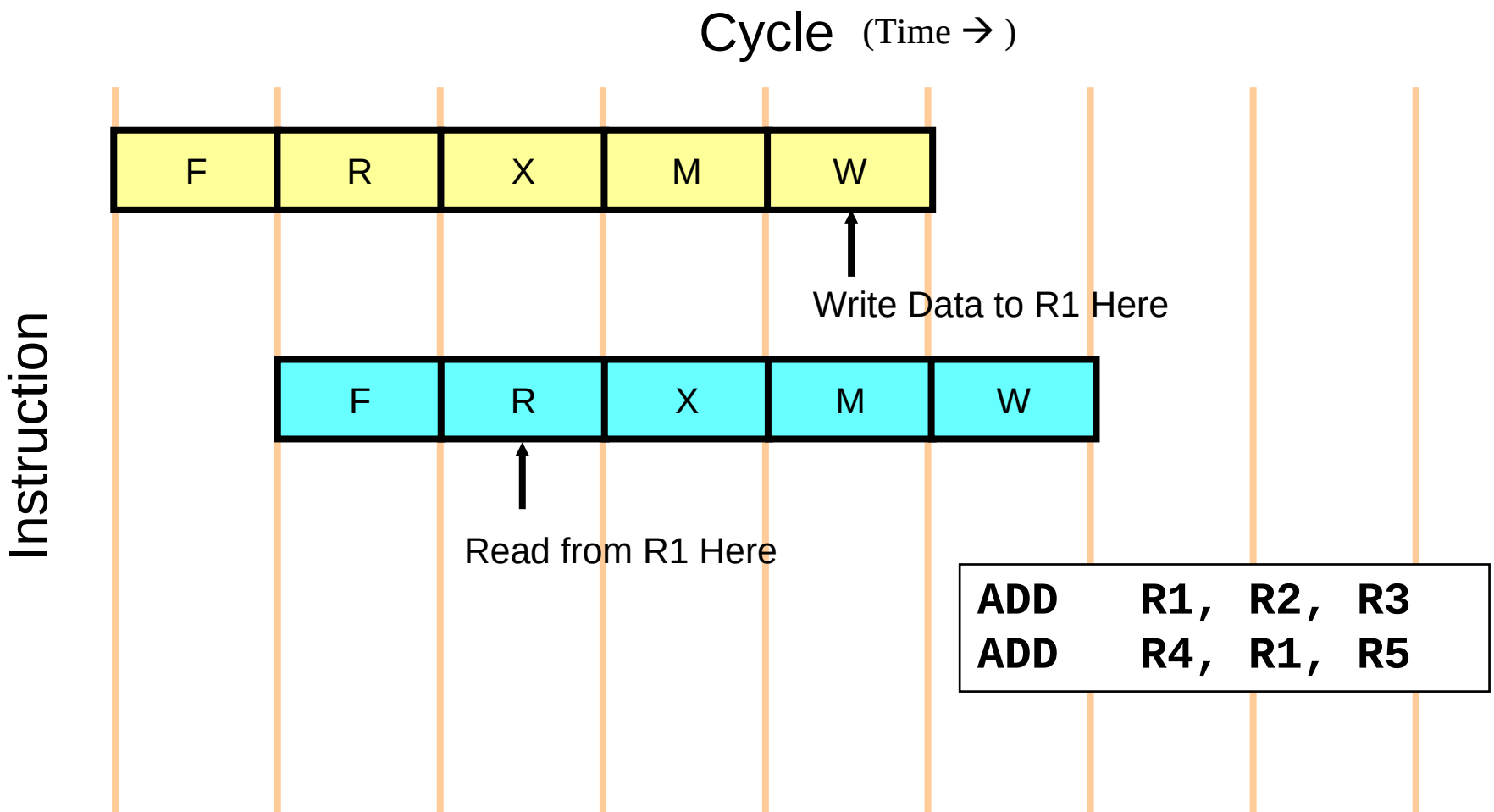
```
ADD   R1, R2, R3        or      SW    R1, 3(R2)
ADD   R4, R1, R5                LW    R3, 3(R2)
```

- Control hazards
  - the location of an instruction depends on a previous instruction

```
        JMP   LOOP
        ...
LOOP:   ADD   R1, R2, R3
```

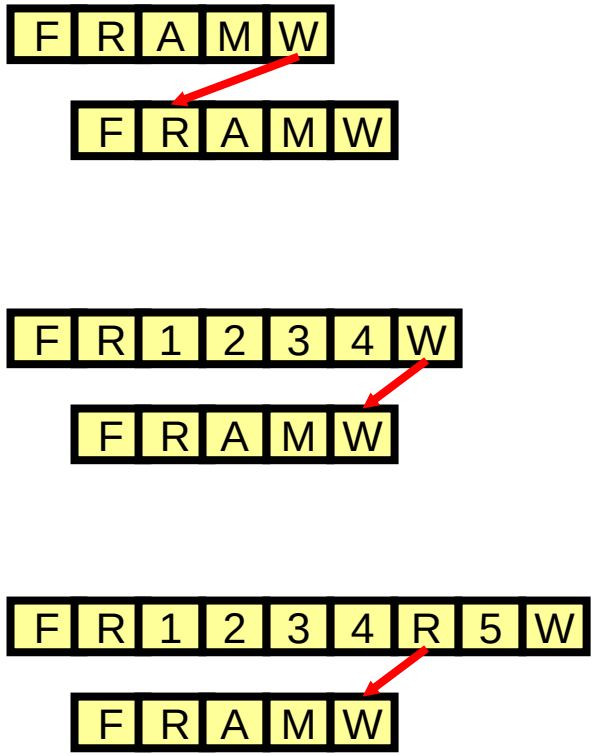
- Structural hazards
  - two instructions need access to the same resource
    - e.g., single memory shared for instruction fetch and load/store
    - collision in reservation table

Data Hazards (RAW)

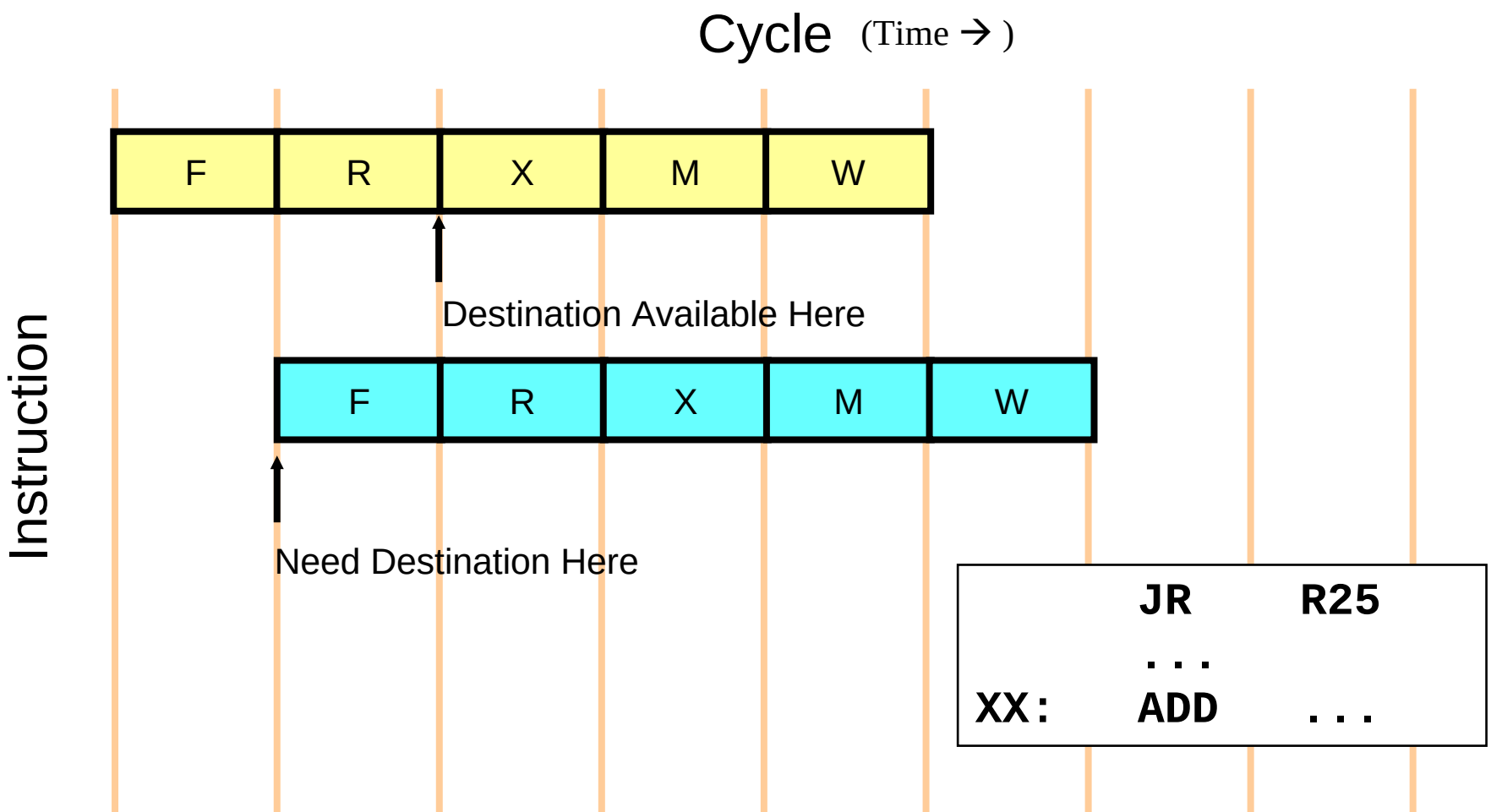


Types of Data Hazards

- RAW (read after write)
  - only hazard for 'fixed' pipelines
  - later instruction must *read* after earlier instruction *writes*
- WAW (write after write)
  - variable-length pipeline
  - later instruction must *write* after earlier instruction *writes*
- WAR (write after read)
  - pipelines with late read
  - later instruction must *write* after earlier instruction *reads*



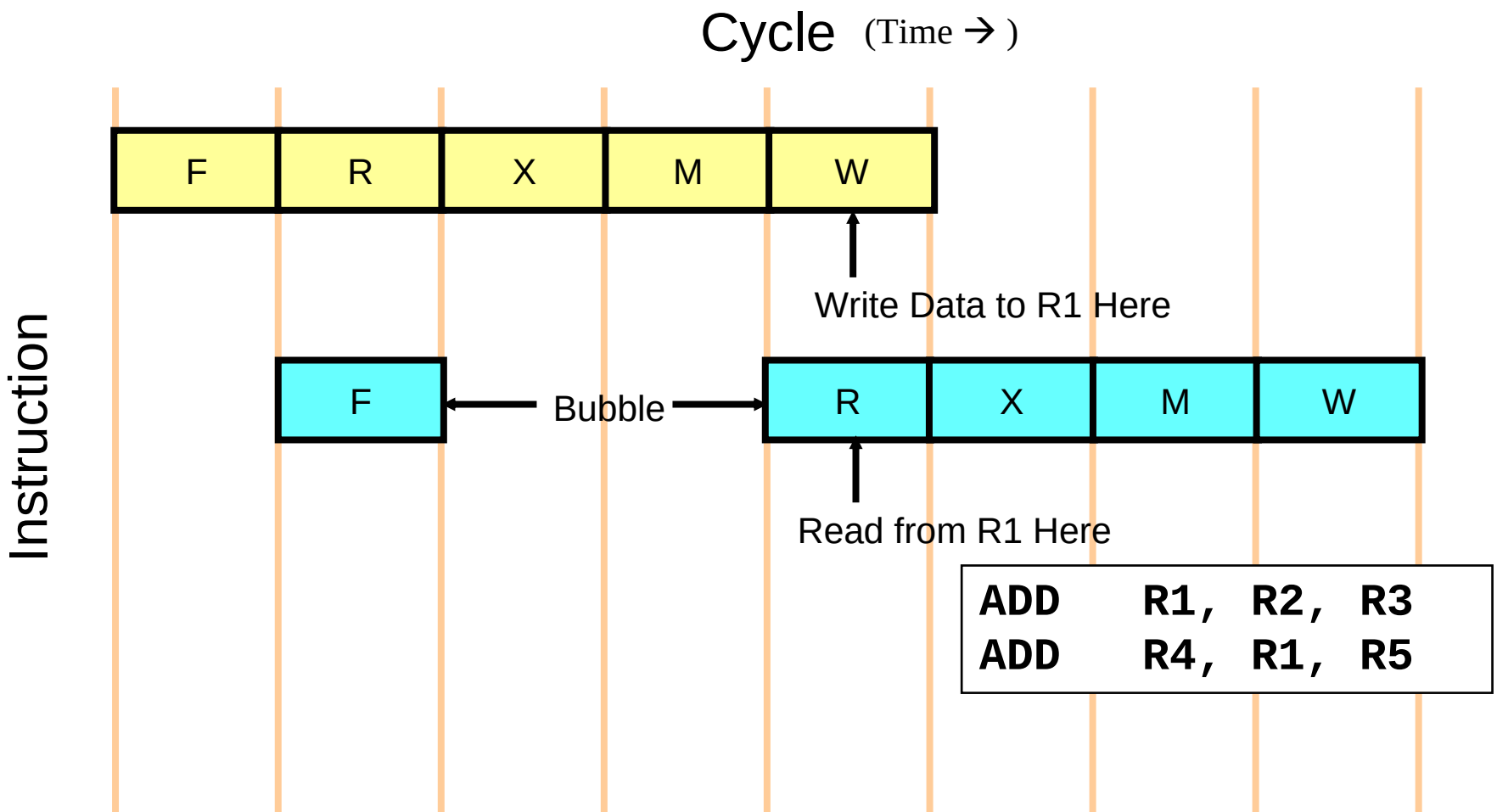
Control Hazards



Resolving Hazards: Pipeline Stalls

- Can resolve any type of hazard
  - data, control, or structural
- Detect the hazard
- Freeze the pipeline up to the dependent stage until the hazard is resolved

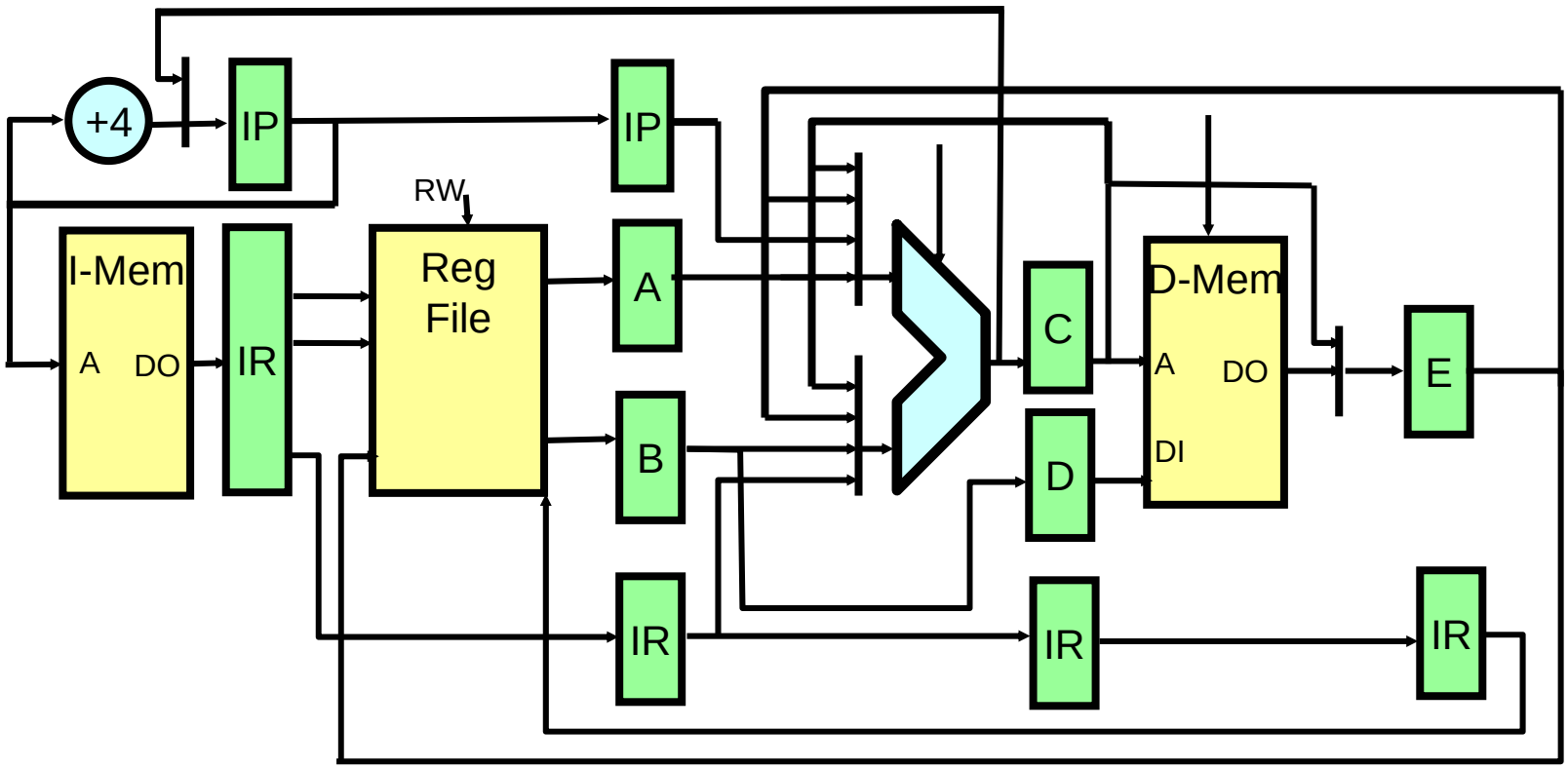
Example Pipeline Stall (Diagram)



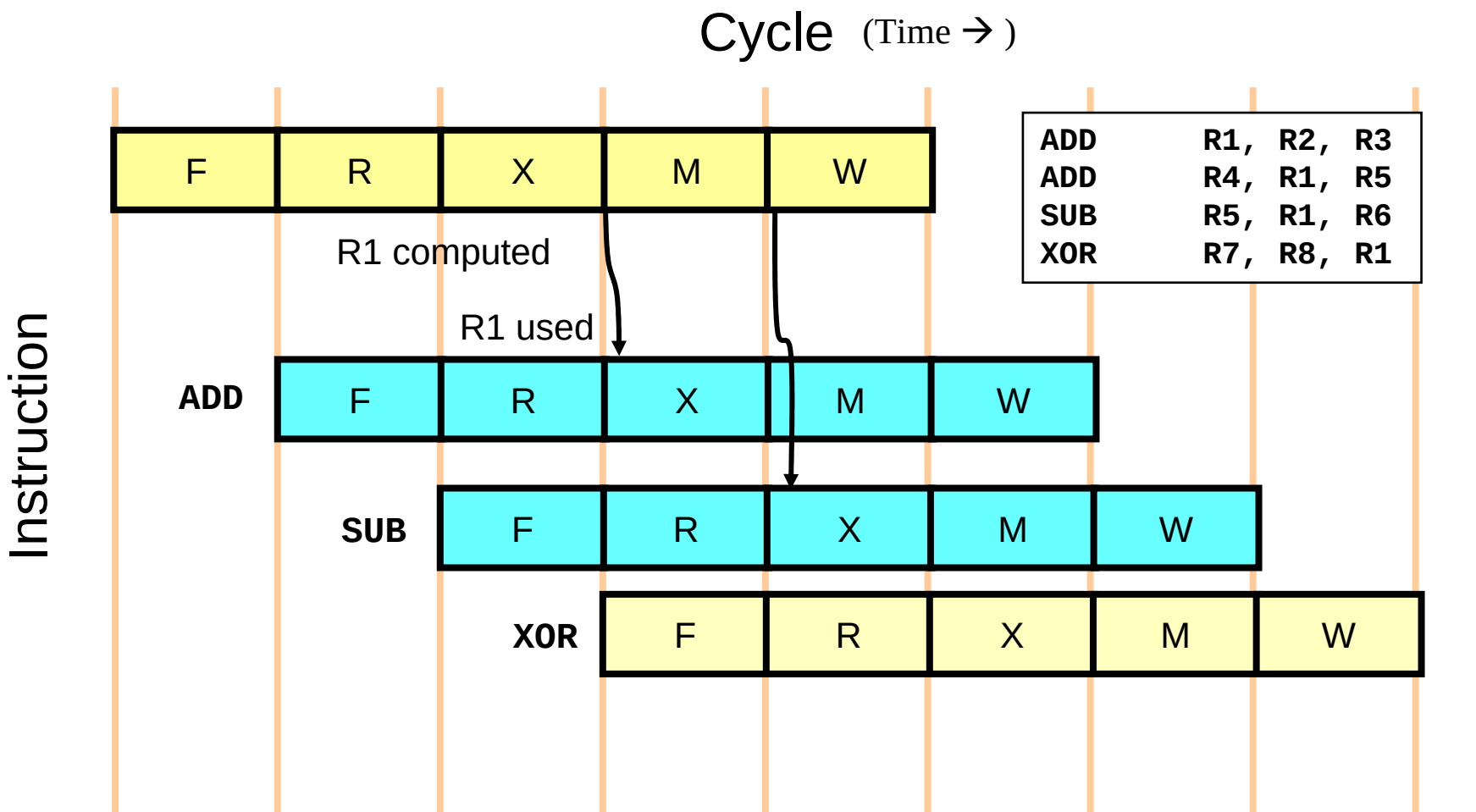
Resolving Hazards: Bypass (Forwarding)

- If data is available elsewhere in the pipeline, there is no need to stall
- Detect condition
- Bypass (or forward) data directly to the consuming pipeline stage
- Bypass eliminates stalls for *single-cycle* operations
  - reduces longest stall to N-1 cycles for N-cycle operations

Simple Pipeline with Bypass Multiplexers



Data Hazards With Bypassing



Summary

- Pipelining principles
- Hazards
- Hazard prevention

- Next Time – more about hazards
  - More bypassing
  - Branch and load delay slots
  - Simple speculation
  - Etc.