Computer Organization and Architecture

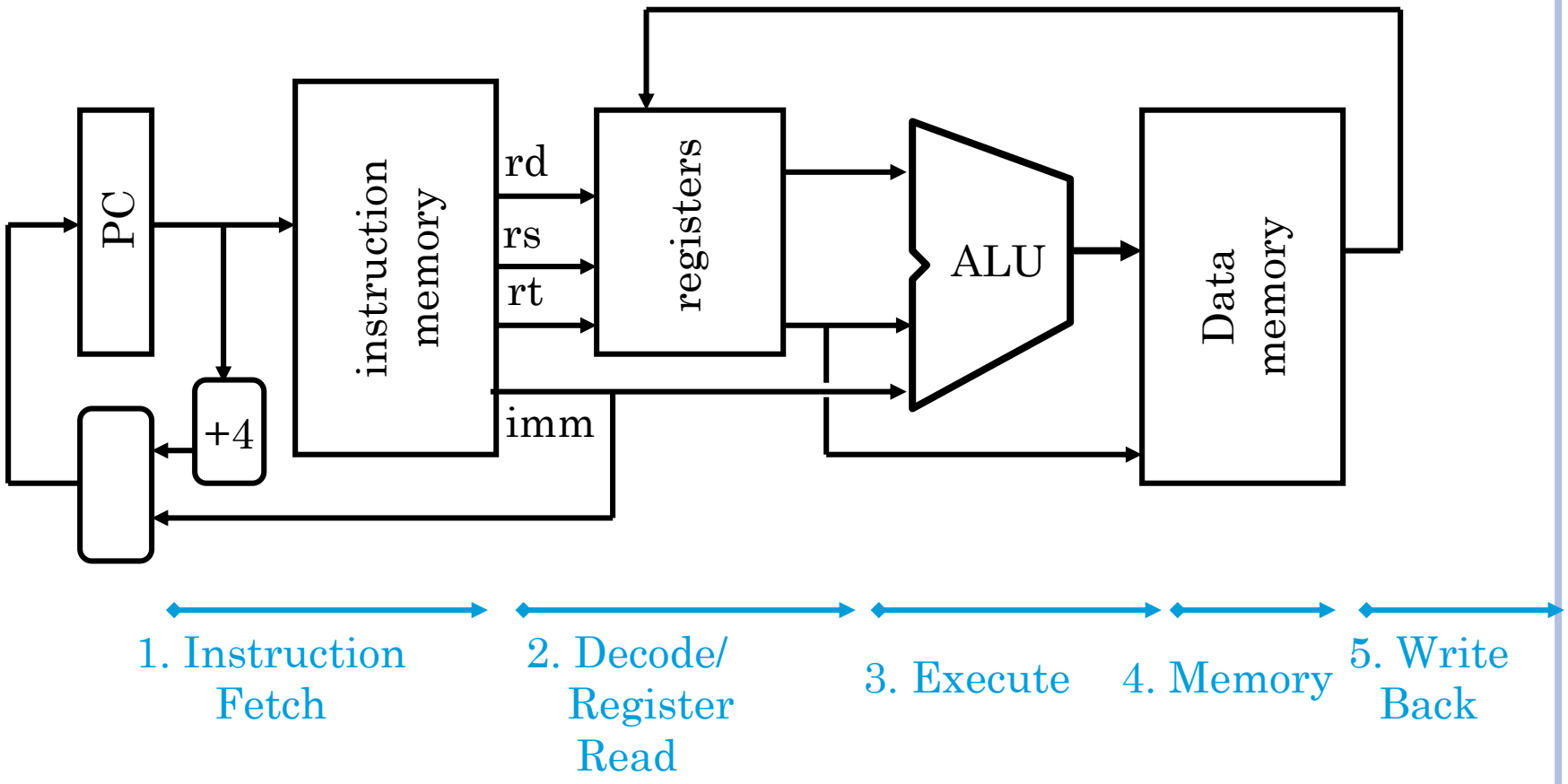
Under Graduate Course
(B. Tech-Information Technology, 2nd Semester)
Jan 2020-July 2020

By

Dr. Satish Kumar Singh

Associate Professor
Indian Institute of Information Technology, Allahabad
Email:sk.singh@iiita.ac.in

Datapath Computer Architecture



Computer Performance

- Latency:
  - Time to completely execute a certain task
    - For example, time to read a sector from disk is disk access time or disk latency
- Throughput:
  - Amount of work that can be done over a period of time
- Performance Enhancement:
  - Parallel Processing
    - Pipeline
    - Vector processing
    - Array processing

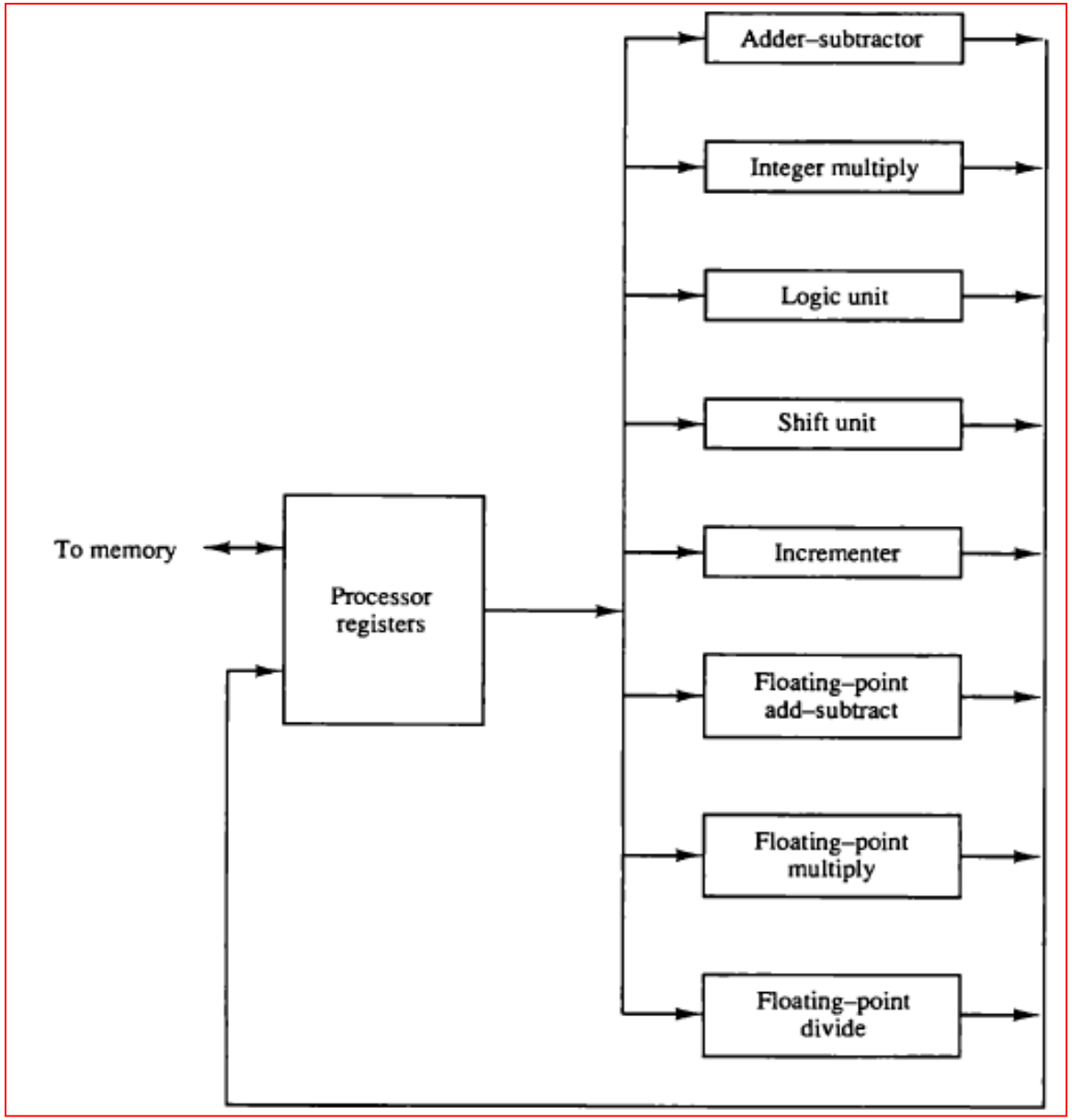
Parallel Processing

- Varity of techniques adopted to improve the computer performance in terms of throughput, while applying the simultaneous data processing tasks.
- Virtually computational speed is enhanced.
- Concurrent data processing.
- Purpose:
  - Computer processing capability speed-up.
  - Increase in throughput.
- Parallel processing is commercially feasible with the advent of cost effective hardware design technologies .

Parallel Processing cont…

- Various levels of parallel processing
  - Data level: serial vs parallel
  - Function level: Multiple functional units
    - Functions may be identical or different.
  - Architecture level

Parallel Processor cont....



Parallel Processor cont….

- Complex control requirement for parallel processing using multifunctional organization.
- Coordination is quite difficult.

Parallel Processor cont….

- Basis of Parallel Processing
  - Internal organization of processor,
  - Structural interconnects between processors,
  - Information flow.
- Instruction Stream
  - Sequence of instructions read from memory
- Data Stream
  - Operations performed on data
- NOTE:
  - Parallel processing in instruction or data streams or both.

Flynn's classification

- Single instruction stream, single data stream (SISD).
  - Parallel processing by means of Multiple processing elements with multiple functionality or pipelining.
- Single instruction stream, multiple data stream (SIMD).
  - Parallel processing by means of Multiple processing elements with same functionality.
- Multiple instruction stream, single data stream (MISD).
  - Only of theoretical interest.
- Multiple instruction stream, multiple data stream (MIMD).
  - Parallel processing by means of Multiprocessor and multicomputer system arrangement.

PIPELINING

- Pipelining:
  - Process of decomposing the sequential process into sub-processes being executed in special, respective dedicated segments, operation concurrently.
  - It is the collection of processing segments through which binary information flows.
  - Final result is obtained after passing the data from all the segments.

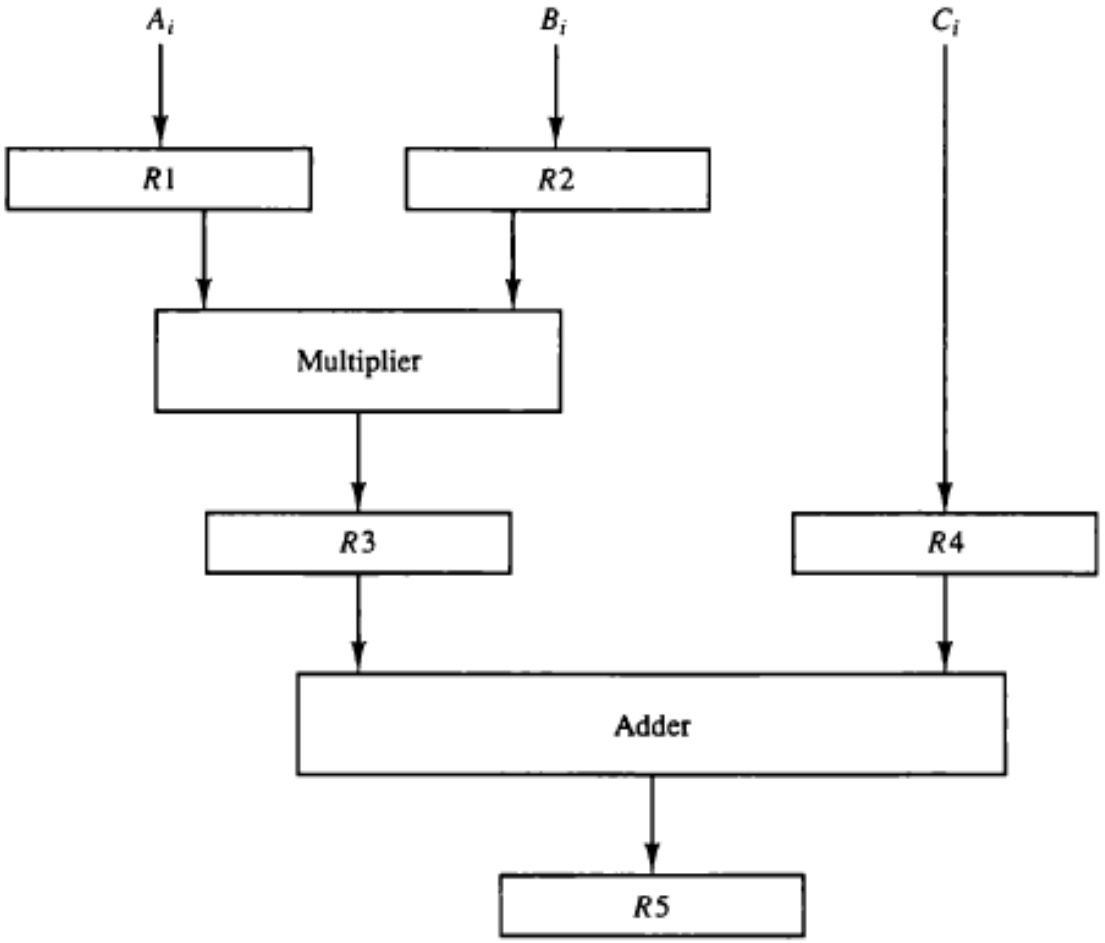
PIPELINE ⟷ ASSEMBLY LINE

Example of Pipeline Processing

- Overlapped computations require ISOLATION / BUFFER registers between two segments to facilitate the pipelining process.

$A_i * B_i + C_i$ for $i = 1, 2, 3, \ldots, 7$

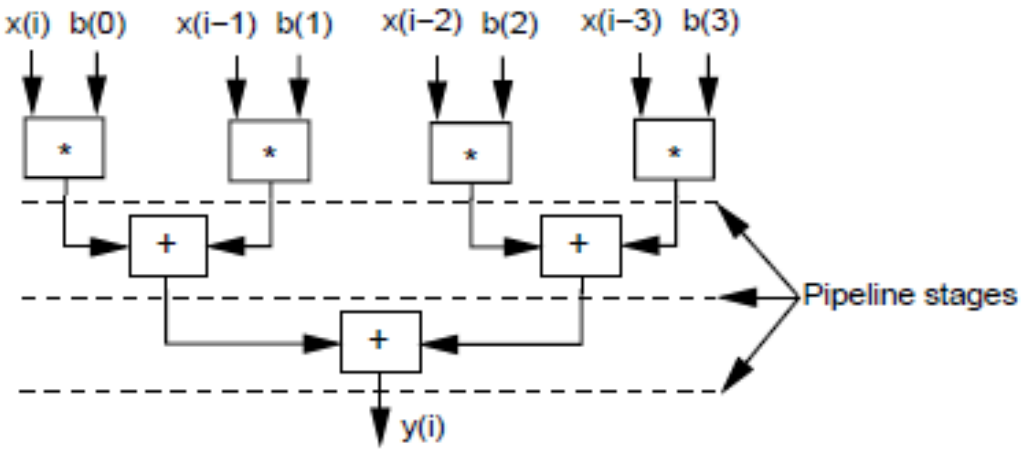
$R1 \leftarrow A_i$, $R2 \leftarrow B_i$	Input A_i and B_i
$R3 \leftarrow R1 * R2$, $R4 \leftarrow C_i$	Multiply and input C_i
$R5 \leftarrow R3 + R4$	Add C_i to product



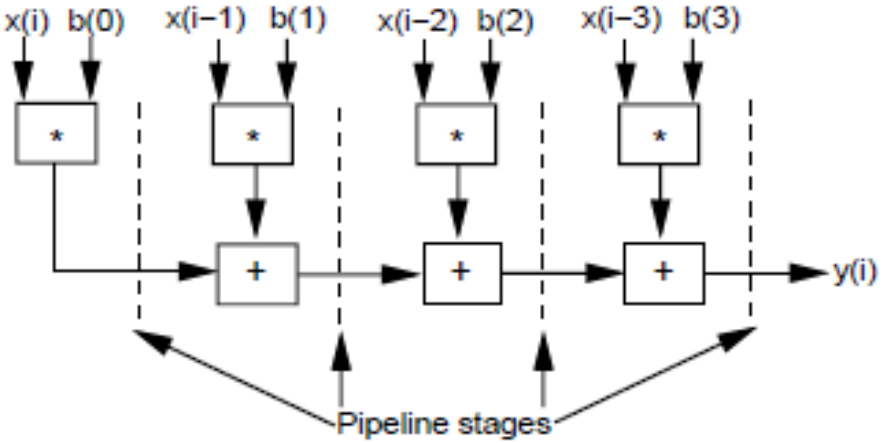
General Consideration of Pipelining

- Decomposability of operation into sub-operations of about the same complexity level can be implemented by pipeline processor.
- Pipeline processing is mostly efficient for those applications with repeated task for different set of data.

Two Pipeline architectures

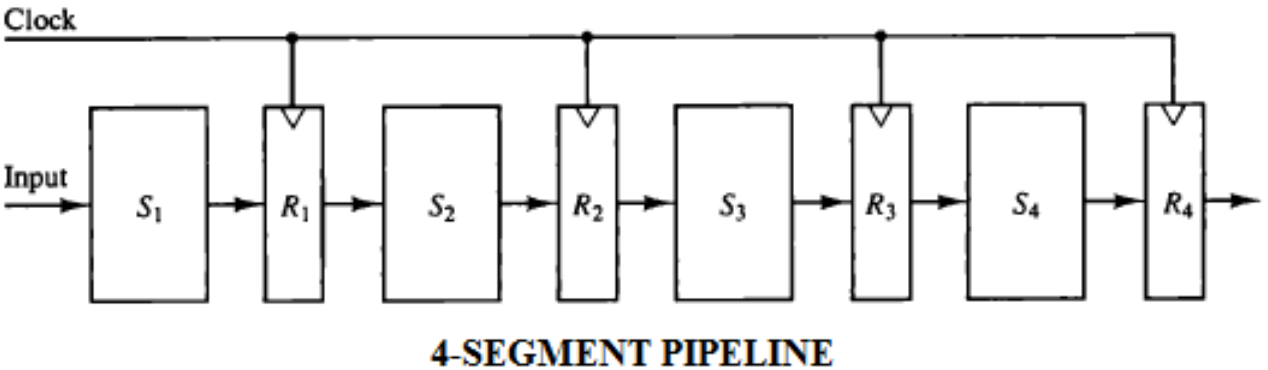


Three stage pipeline



Four stage pipeline

PIPELINE SPEED-UP

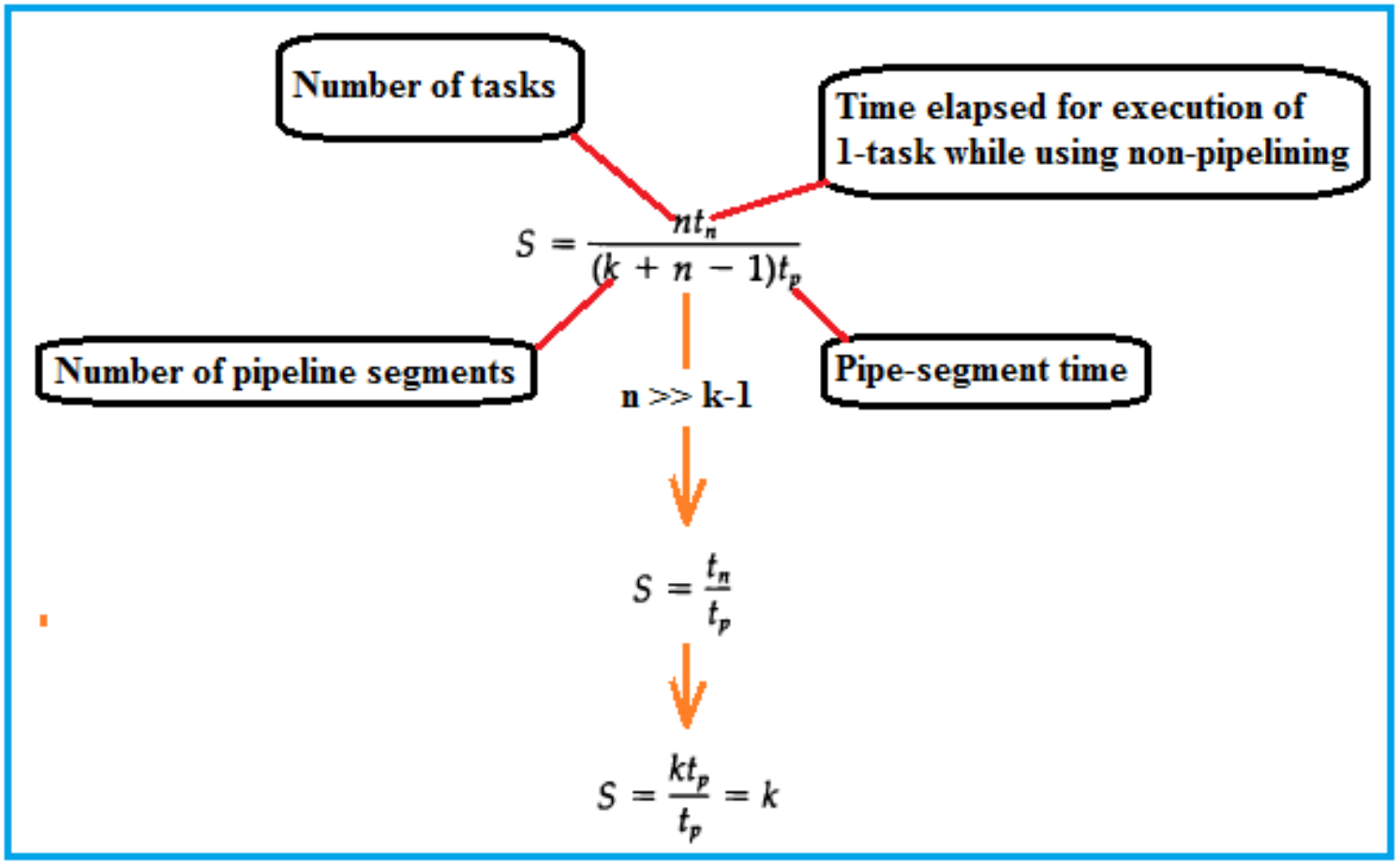


4-SEGMENT PIPELINE

Segment:	1	2	3	4	5	6	7	8	9
1	T_1	T_2	T_3	T_4	T_5	T_6			
2		T_1	T_2	T_3	T_4	T_5	T_6		
3			T_1	T_2	T_3	T_4	T_5	T_6	
4				T_1	T_2	T_3	T_4	T_5	T_6

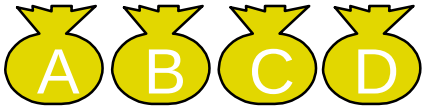
Clock cycles

SPACE-TIME DIAGRAM

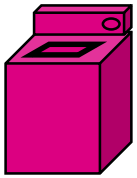

Number of tasks
Time elapsed for execution of 1-task while using non-pipelining
$S = \frac{nt_n}{(k + n - 1)t_p}$
Number of pipeline segments
Pipe-segment time
n >> k-1
$S = \frac{t_n}{t_p}$
$S = \frac{kt_p}{t_p} = k$

Laundry example

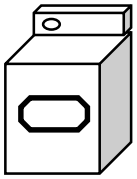
- Ann, Brian, Cathy, Dave each have one load of clothes to wash, dry, fold, and put away



  - Washer takes 30 minutes



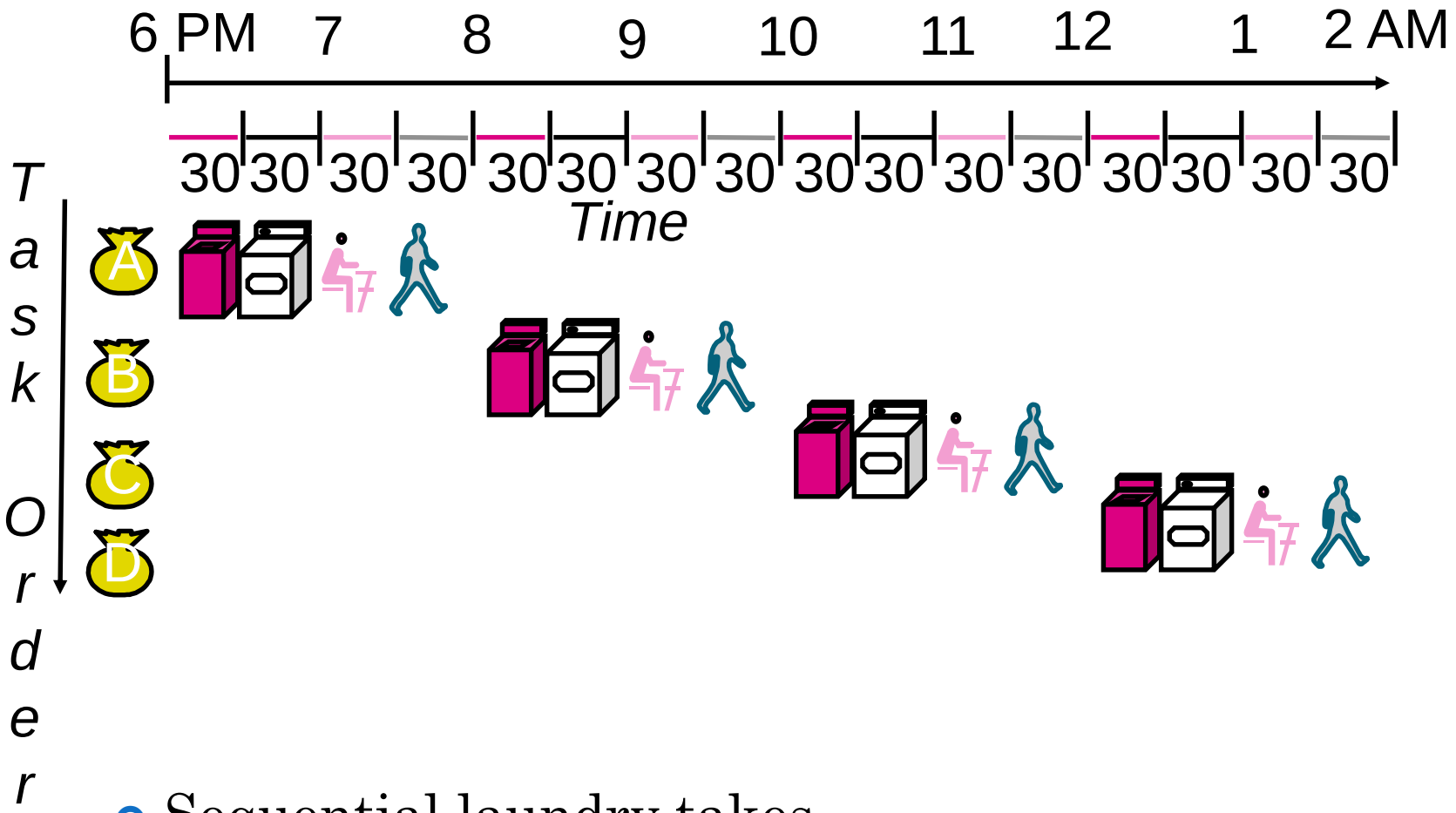
  - Dryer takes 30 minutes



  - "Folder" takes 30 minutes

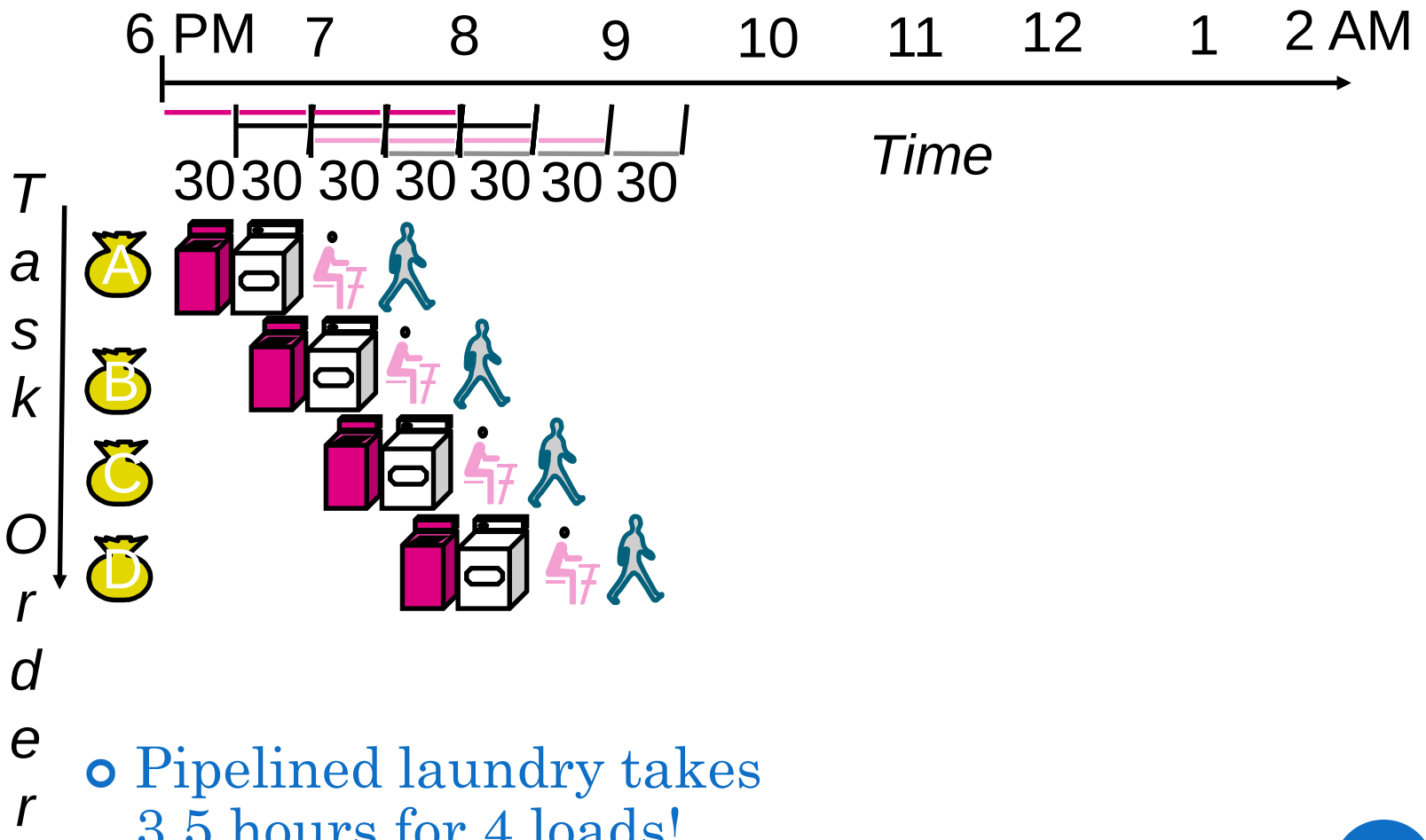
  - "Stasher" takes 30 minutes to put clothes into drawers

Sequential Laundry



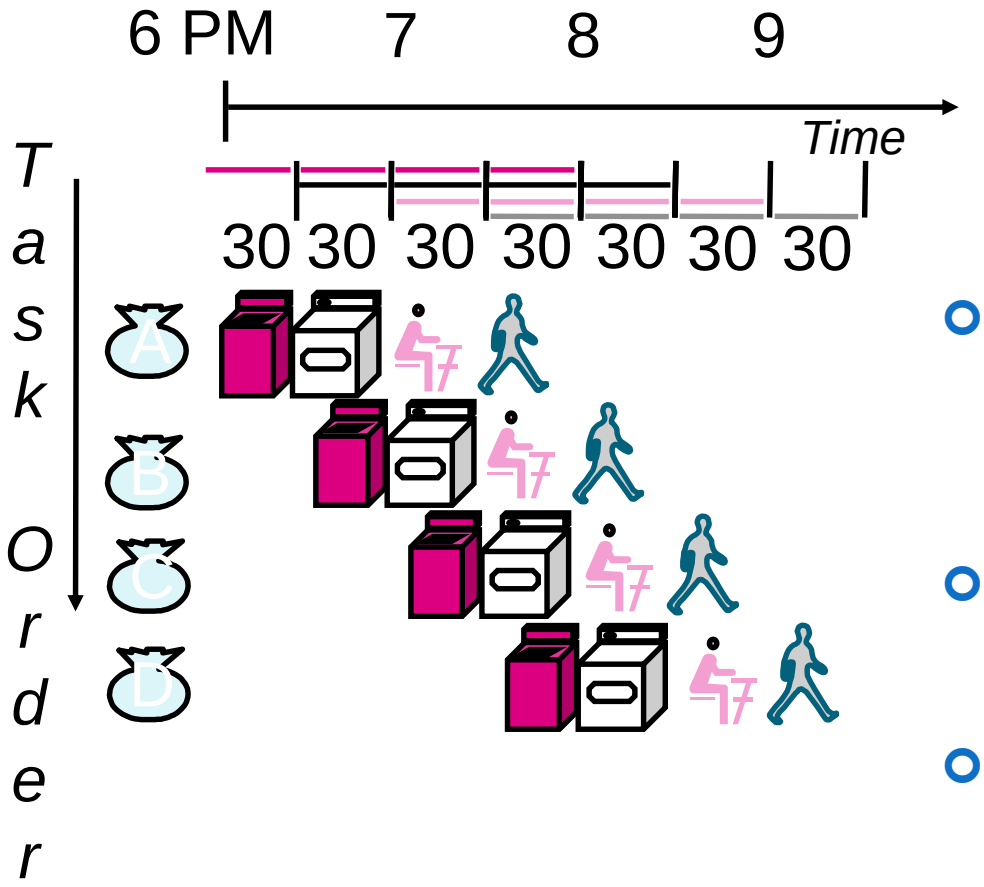
- Sequential laundry takes 8 hours for 4 loads

Pipelined Laundry



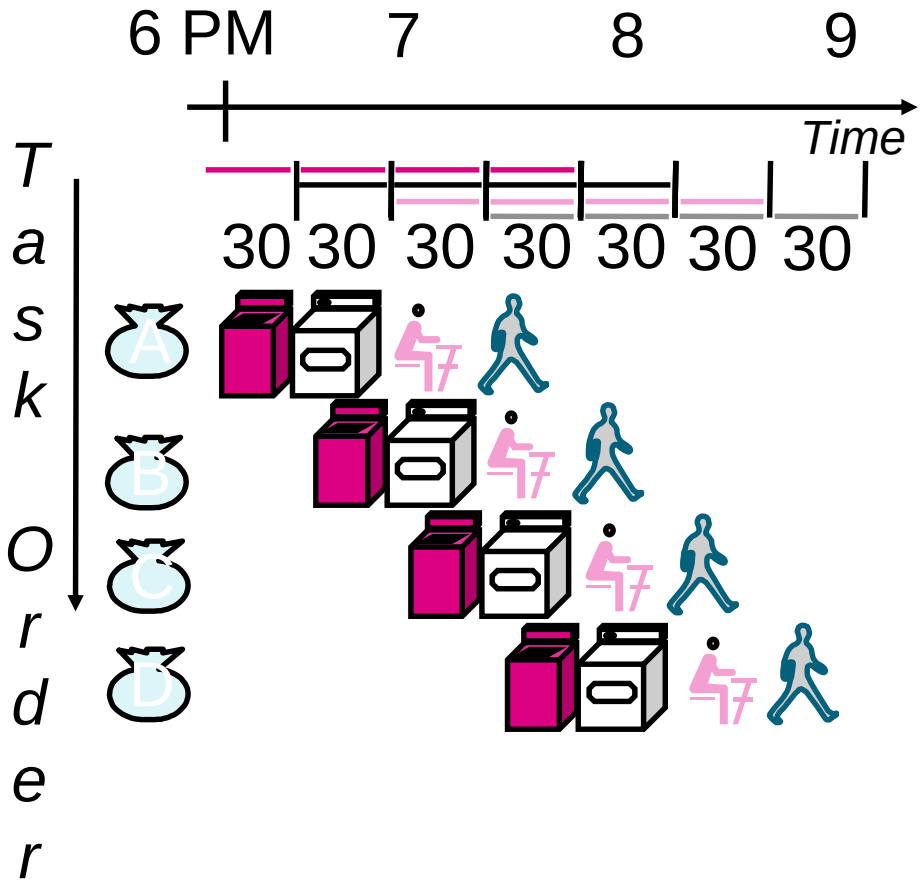
- Pipelined laundry takes 3.5 hours for 4 loads!

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

Pipelining Lessons (2/2)

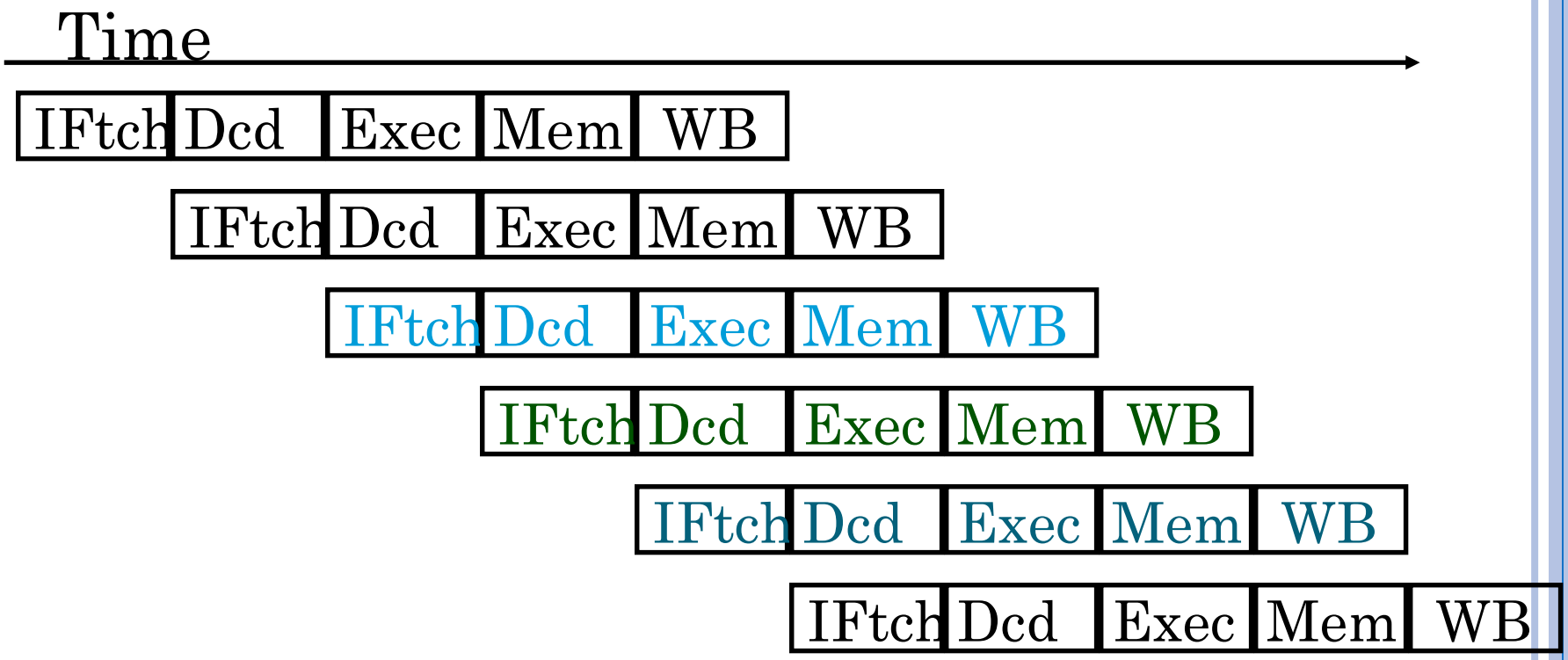


- Suppose new Washer takes 20 minutes, new Stasher takes 20 minutes. How much faster is pipeline?
- Pipeline rate limited by slowest pipeline stage
- Unbalanced lengths of pipe stages also reduces speedup

Steps in Executing MIPS

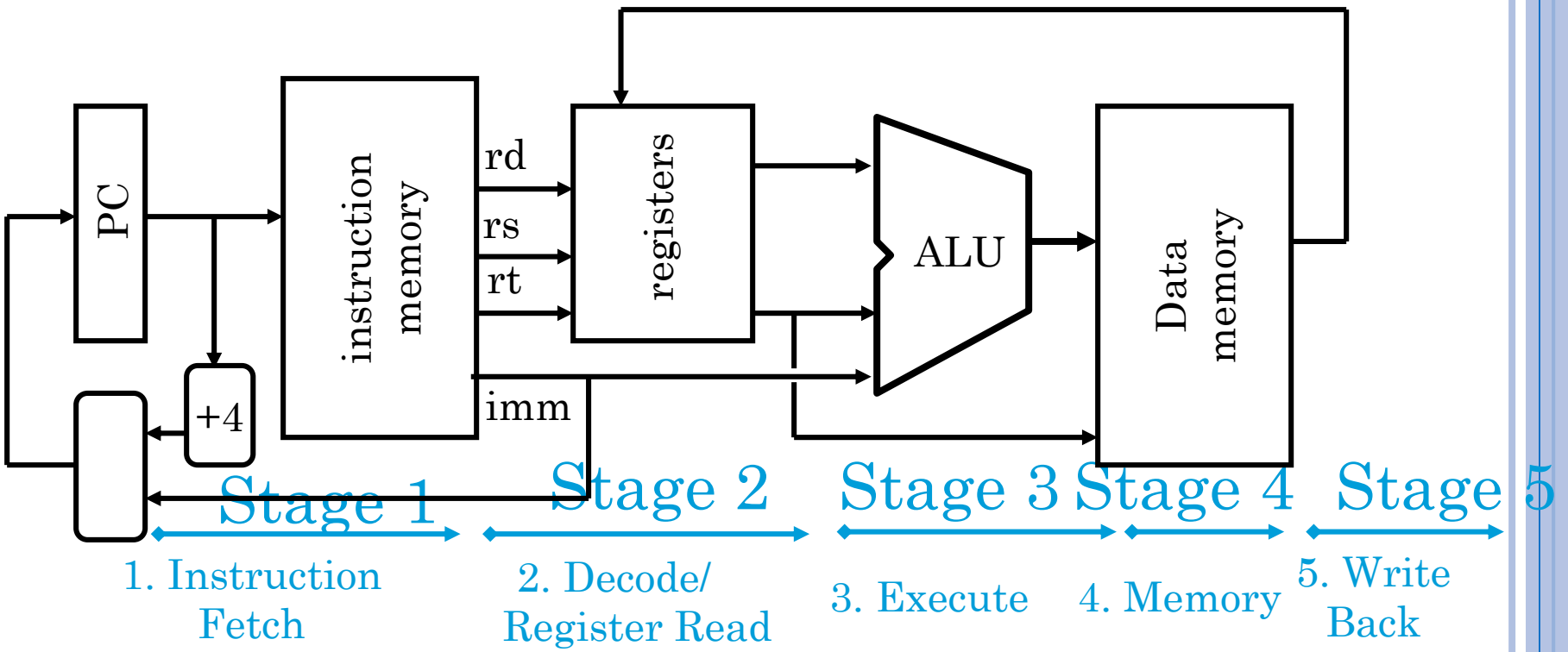
1) IFetch: Fetch Instruction, Increment PC
2) Decode Instruction, Read Registers
3) Execute:
   Mem-ref: Calculate Address
   Arith-log: Perform Operation
4) Memory:
   Load: Read Data from Memory
   Store: Write Data to Memory
5) Write Back: Write Data to Register

Pipelined Execution Representation

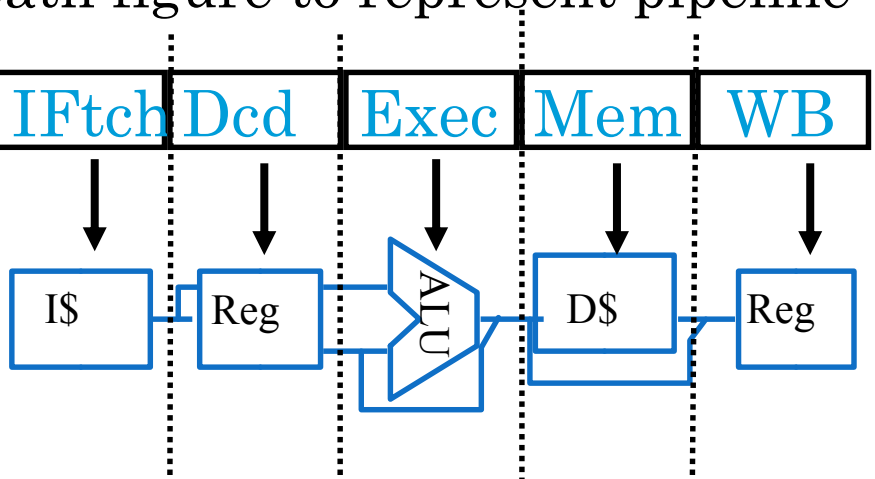


- Every instruction must take same number of steps, also called pipeline "stages", so some will go idle sometimes

Review: Datapath for MIPS

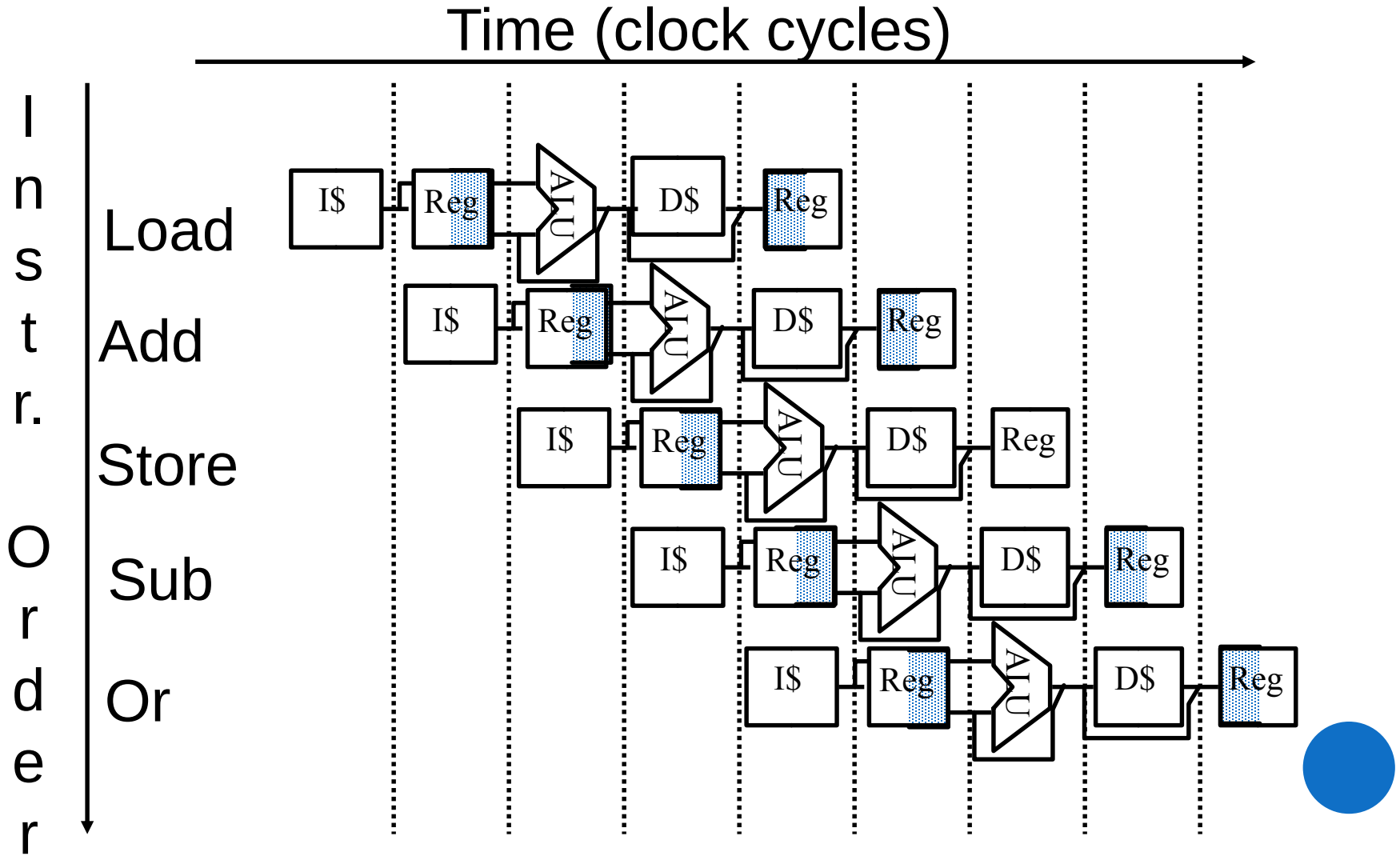


- Use datapath figure to represent pipeline



Graphical Pipeline Representation

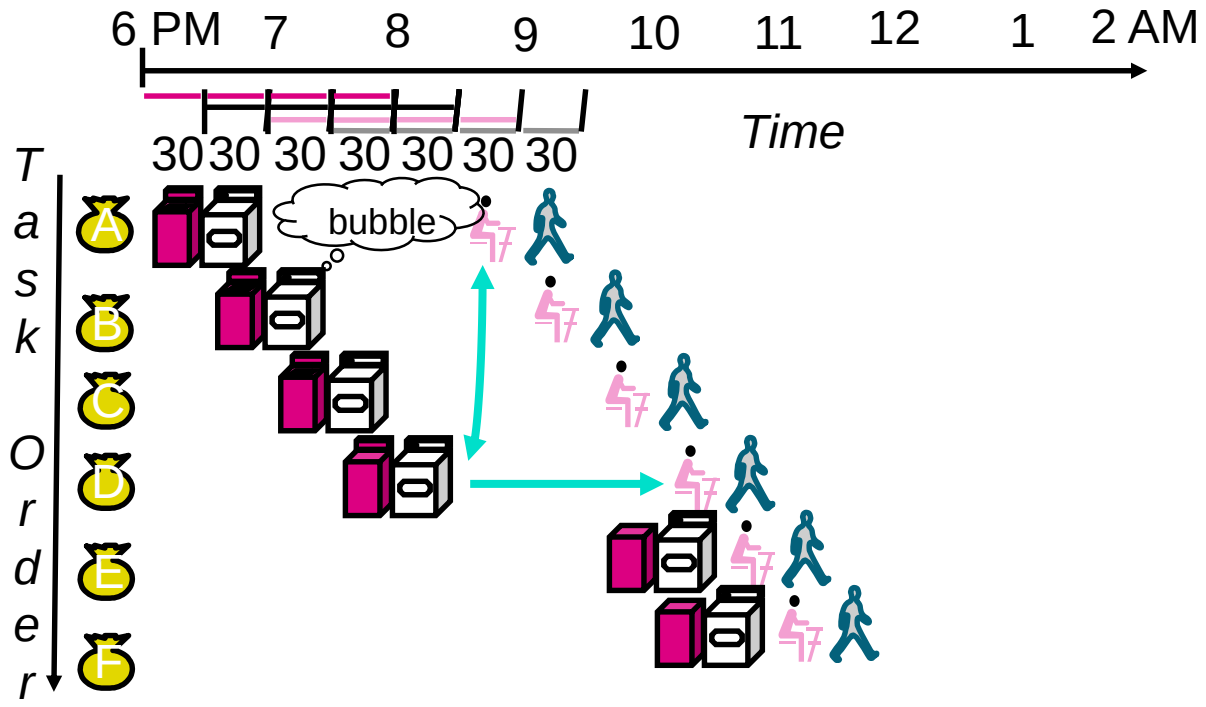
(In Reg, right half highlight read, left half write)



Example

- Suppose 2 ns for memory access, 2 ns for ALU operation, and 1 ns for register file read or write
- Nonpipelined Execution:
  - lw : IF + Read Reg + ALU + Memory + Write Reg = 2 + 1 + 2 + 2 + 1 = 8 ns
  - add: IF + Read Reg + ALU + Write Reg = 2 + 1 + 2 + 1 = 6 ns
- Pipelined Execution:
  - Max(IF,Read Reg,ALU,Memory,Write Reg) = 2 ns

Pipeline Hazard: Matching socks in later load



A depends on D; stall since folder tied up

Thank you