Computer Organization and Software Systems

Session 7

Dr. Lucy J. Gudino
WILP & Department of CS & IS

BITS Pilani
Pilani Campus

Control Unit Design

BITS Pilani
Pilani Campus

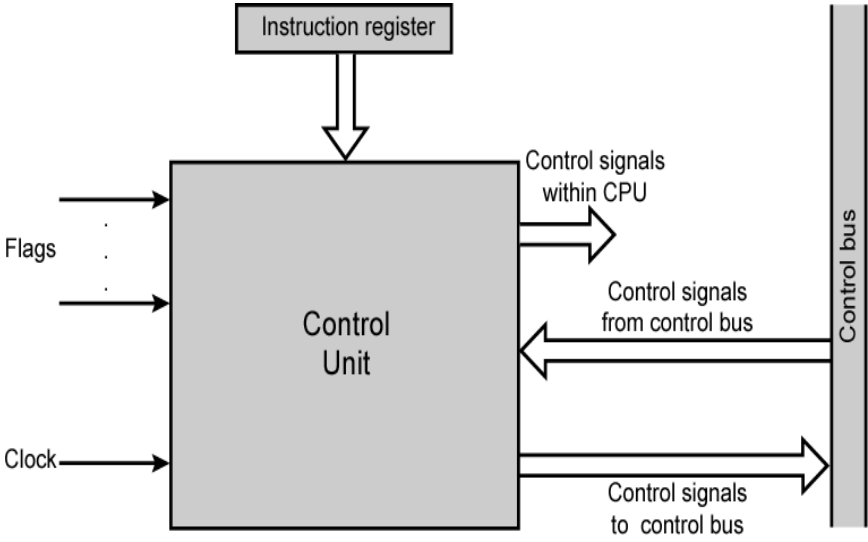
Control Unit implementation

Hardwired control unit (RISC)

Microprogrammed control unit(CISC)

Hardwired Implementation

- Control unit inputs
  - Flags and control bus
    - Each bit means something
  - Instruction register
    - Op-code causes different control signals for each different instruction
    - Unique logic for each op-code
  - Clock



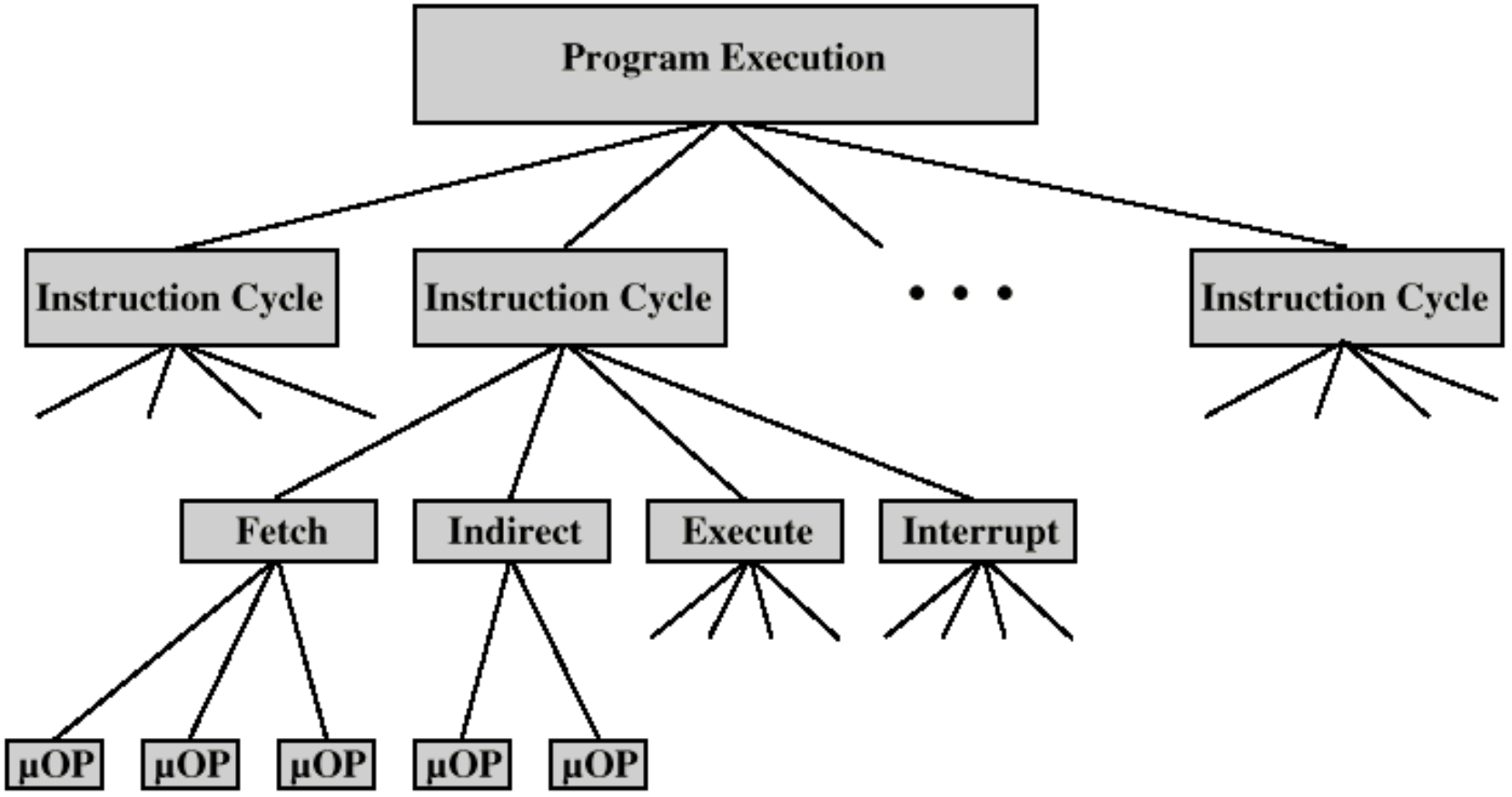
Problems With Hard Wired Designs

- Complex sequencing & micro-operation logic
- Difficult to design and test
- Inflexible design
- Difficult to add new instructions

Microprogrammed Control Unit

- A computer executes a program
- Fetch/execute cycle/interrupt cycle…

Constituent Elements of Program Execution



Example: ADD R1,X

ADD R1,X - add the contents of location X to R1, and stores the result in R1

Fetch Sequence :

t1: MAR ← (PC)

t2: MBR ← (memory)

PC ← (PC) +I

t3: IR ← (MBR)

OR

t1: MAR <- (PC)

t2: MBR <- (memory)

t3: PC <- (PC) + I

IR <- (MBR)

Example: ADD R1,X

Execute Cycle (ADD):

t4: MAR ← ($IR_{address}$)

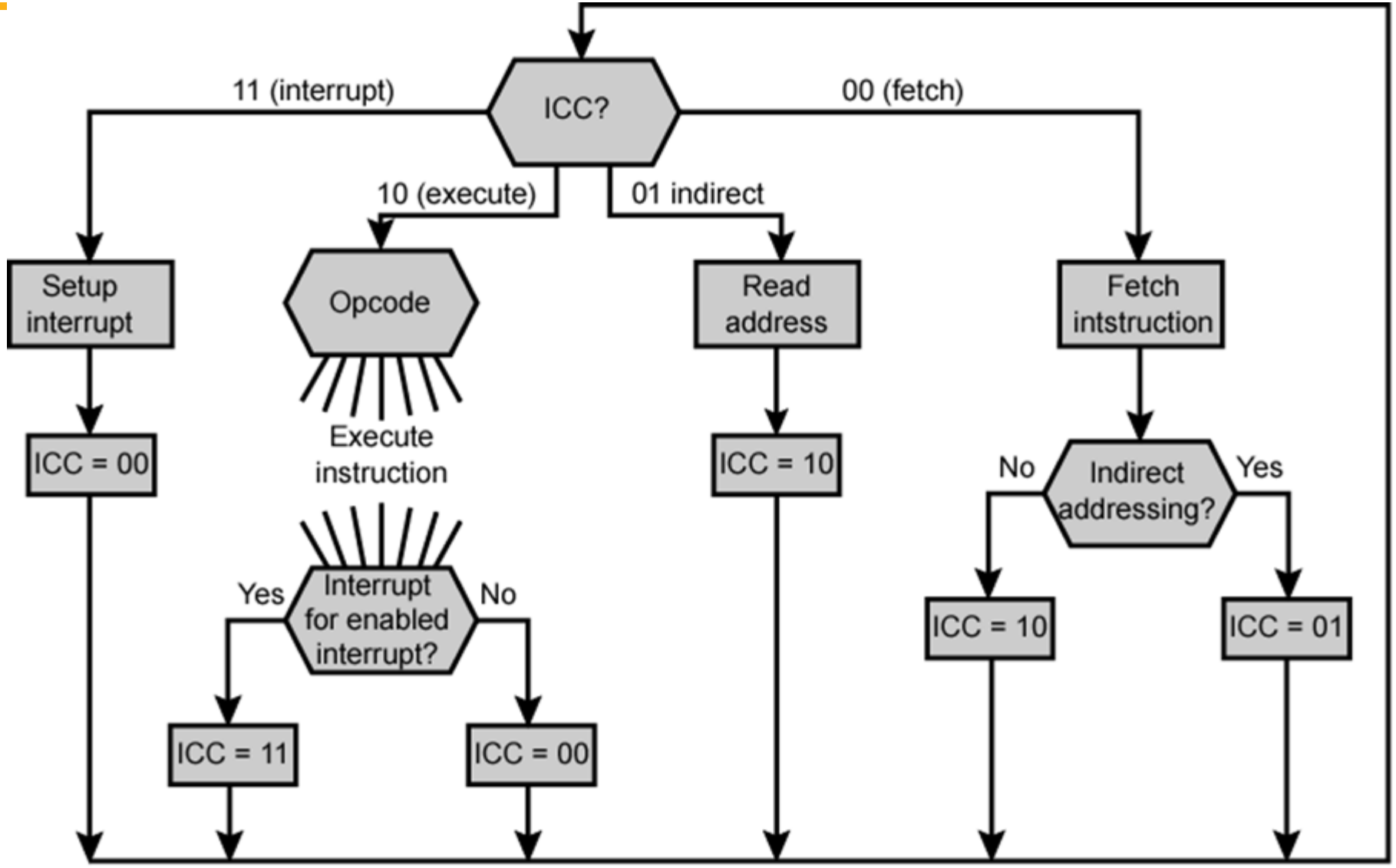
t5: MBR ← (memory)

t6: R1 ← R1 + (MBR)

Instruction Cycle

- Different phases : Instruction fetch, indirect, execute and interrupt
- Each phase decomposed into sequence of elementary micro-operations
- Execute cycle
  - One sequence of micro-operations for each opcode
- Need to tie sequences together
- Assume new 2-bit register
  - Instruction cycle code (ICC) designates which part of cycle processor is in
    - 00: Fetch
    - 01: Indirect
    - 10: Execute
    - 11: Interrupt

Flowchart for Instruction Cycle



Functions of Control Unit

- The control unit performs two basic tasks:
  - Sequencing
    - Causing the CPU to step through a series of micro-operations
  - Execution
    - Causing the performance of each micro-op

Example: ADD R1, X

t1: MAR ← (PC)

t2: MBR ← (memory)

PC ← (PC) +1

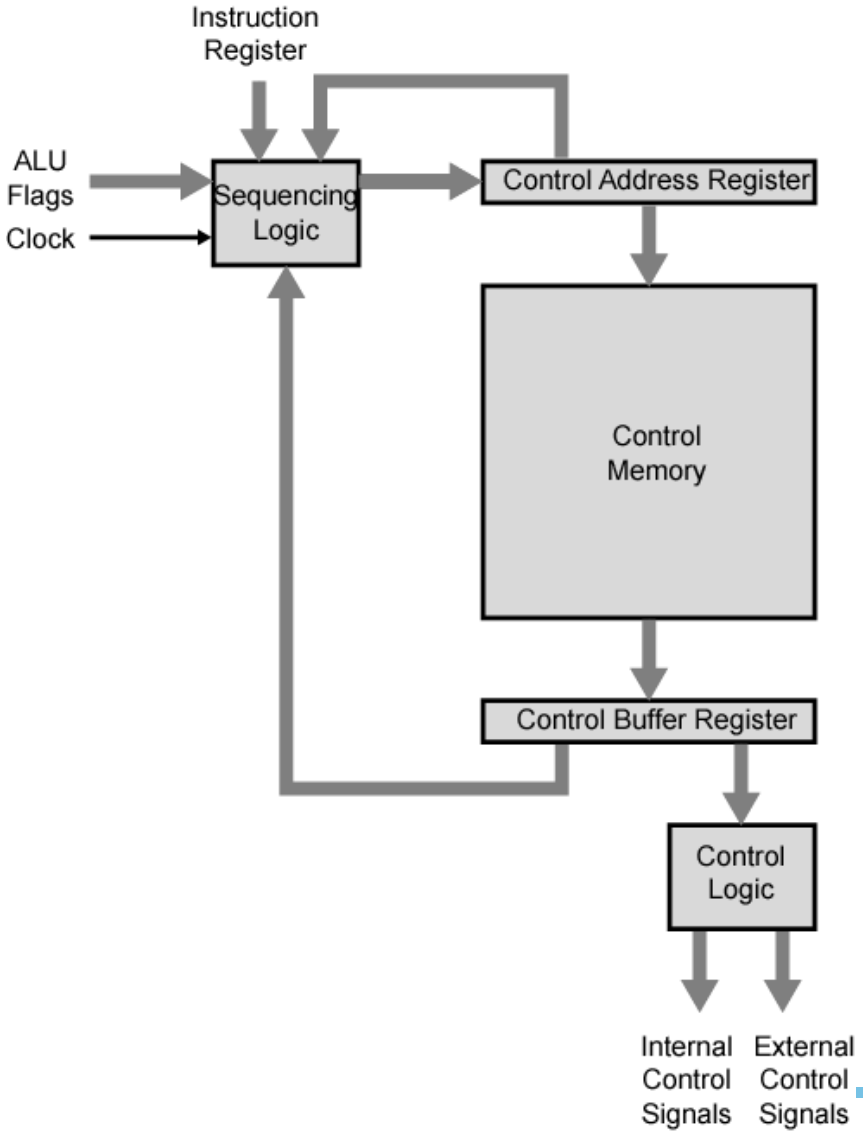
t3: IR ← (MBR)

t4: MAR ← ($IR_{address}$)

t5: MBR ← (memory)

t6: R1 ← R1 + (MBR)

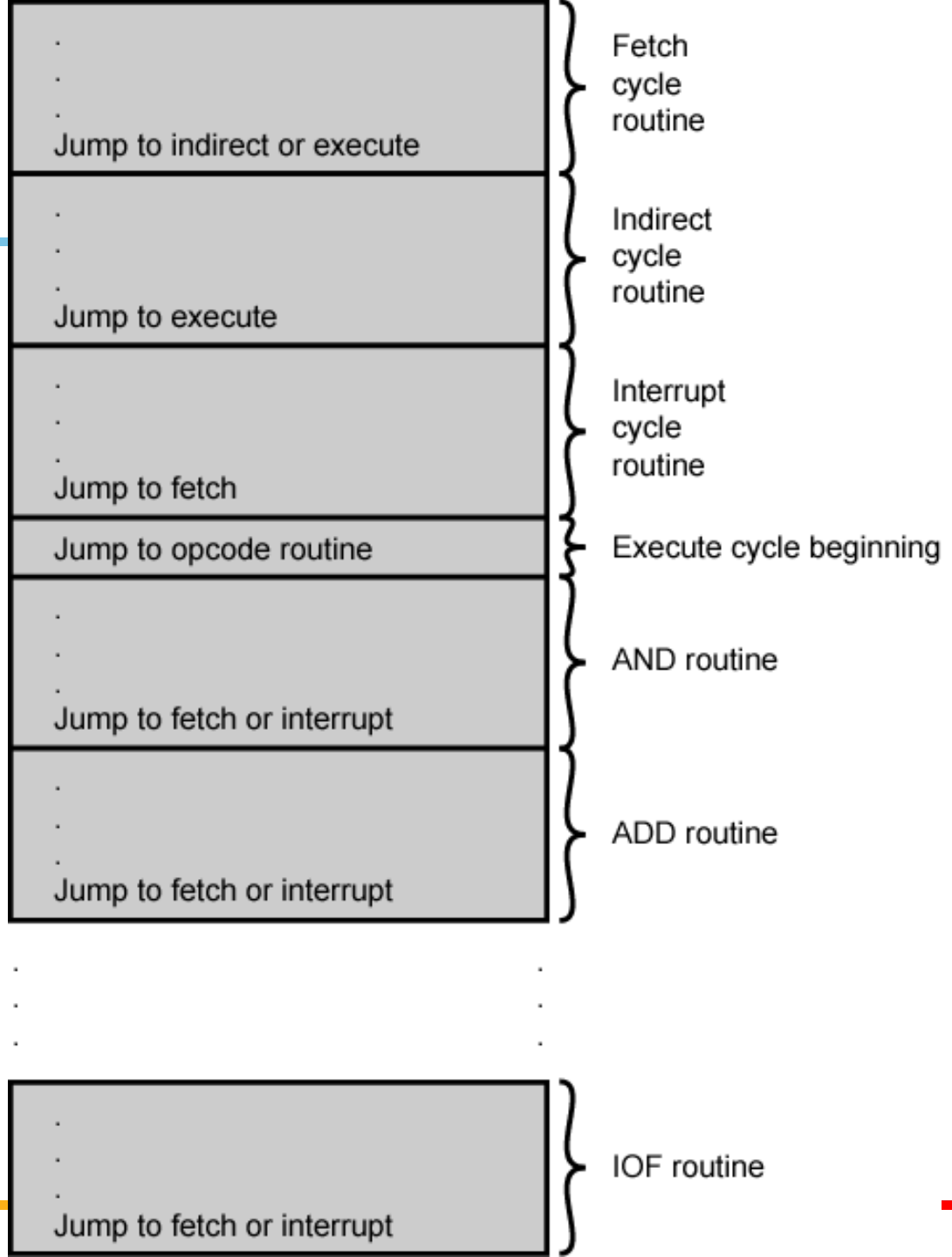
Microprogrammed Control Unit



Control Signals

- Inputs to the control unit
  - Clock
    - One micro-instruction (or set of parallel micro-instructions) per clock cycle
  - Instruction register
    - Op-code for current instruction
    - Determines which micro-instructions are performed
  - Flags
    - State of CPU
    - Results of previous operations
  - From control bus
    - Interrupts
    - Acknowledgements

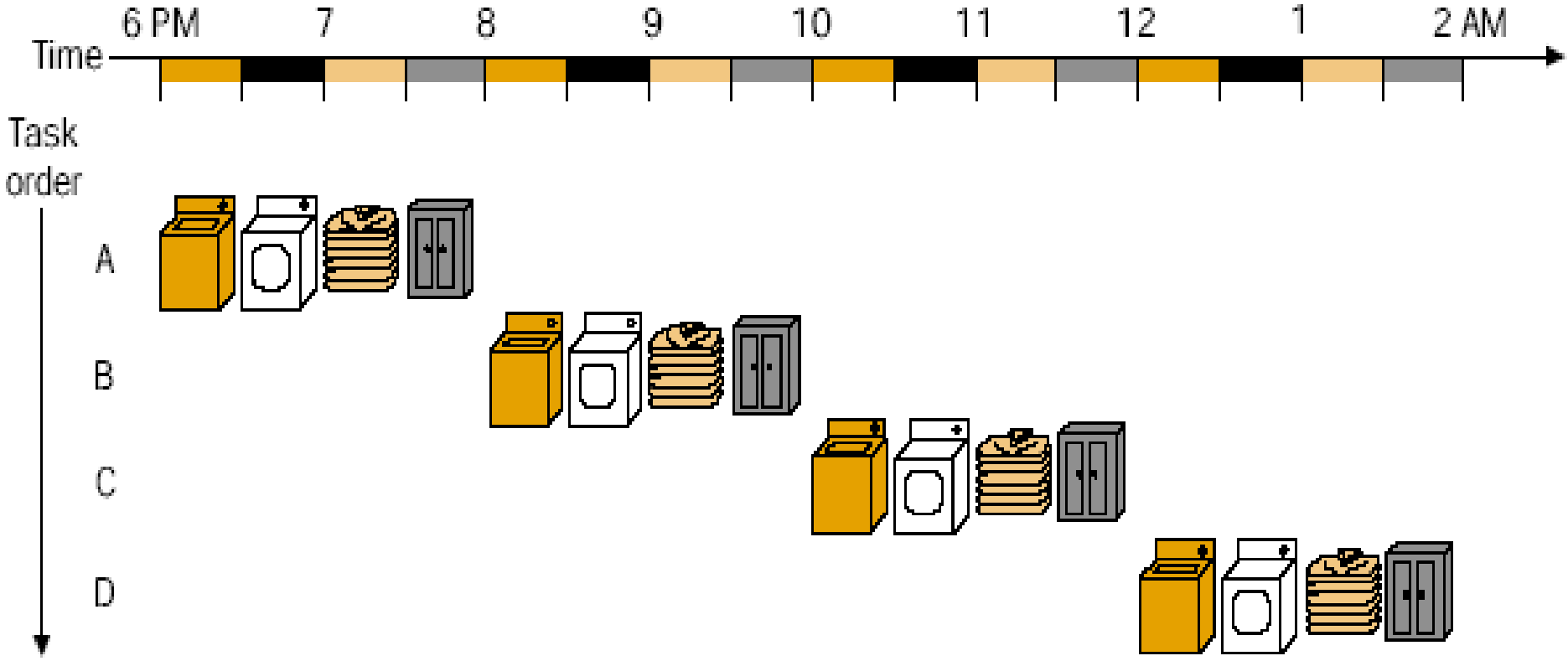
Organization of Control Memory



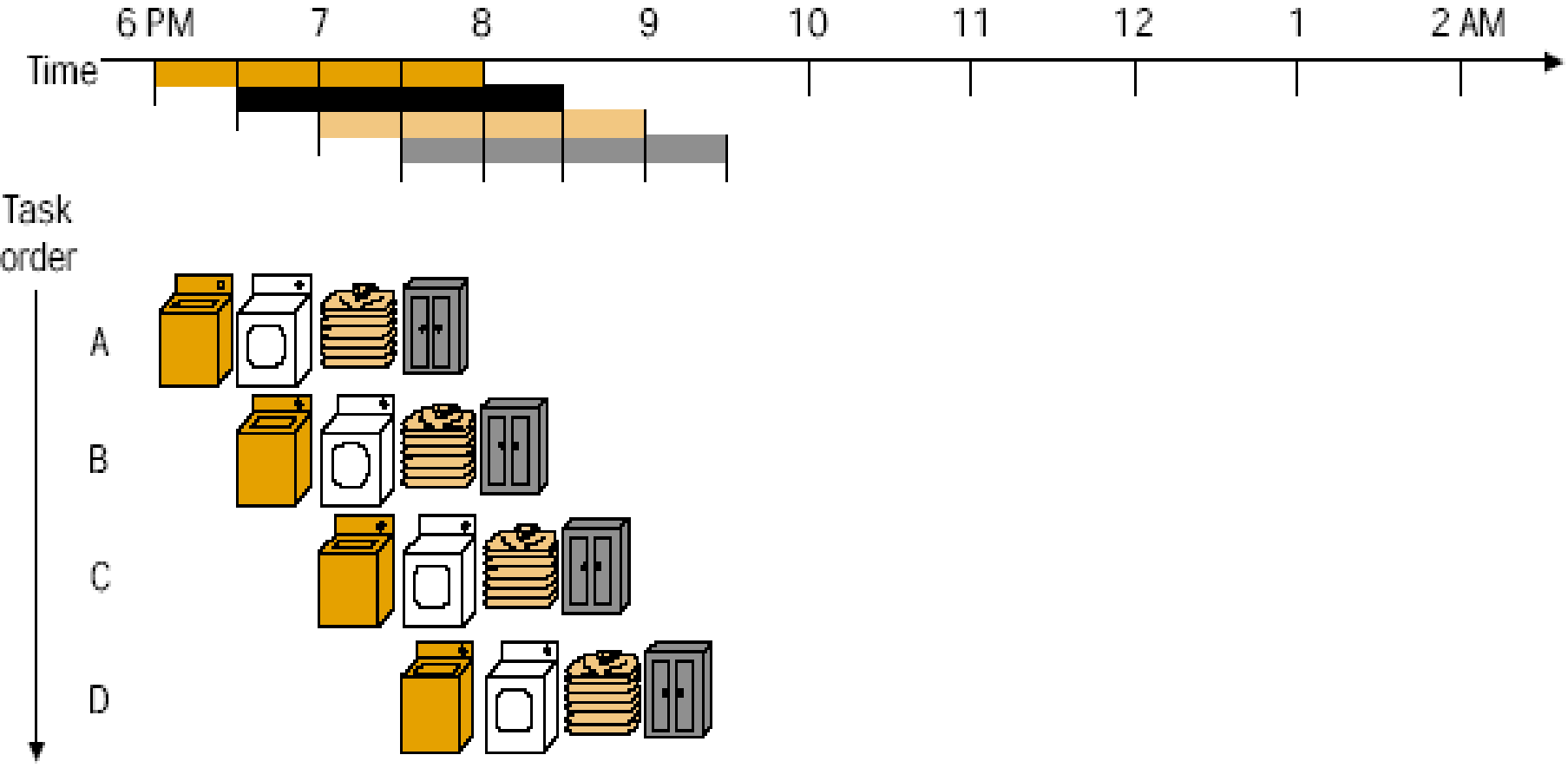
Pipeline

BITS Pilani
Pilani Campus

Laundry System



Laundry System……



Pipelining

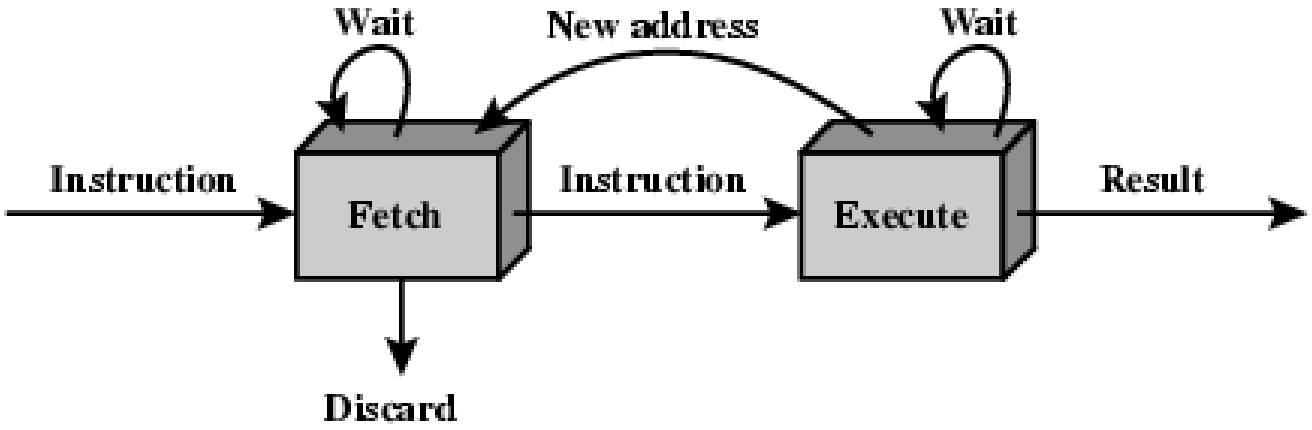
- An overlapped parallelism: overlapped execution of multiple operations
- Pipelining
  - Subdivide the input task into a sequence of subtasks
  - Specialized hardware stage for each task
  - Concurrent operation of all the stages

Two segment instruction pipeline

- Contains
  - Instruction Fetch (IF)
  - Execute (EX)
- Example: 8086 microprocessor
- Instruction Fetch unit is implemented by means of first in first out buffer (Queue)

Two Stage Pipeline

(a) Simplified view



(b) Expanded view

Issues

1. The execution time will generally be longer than the fetch time
2. A conditional branch instruction makes the address of the next instruction to be fetched unknown.

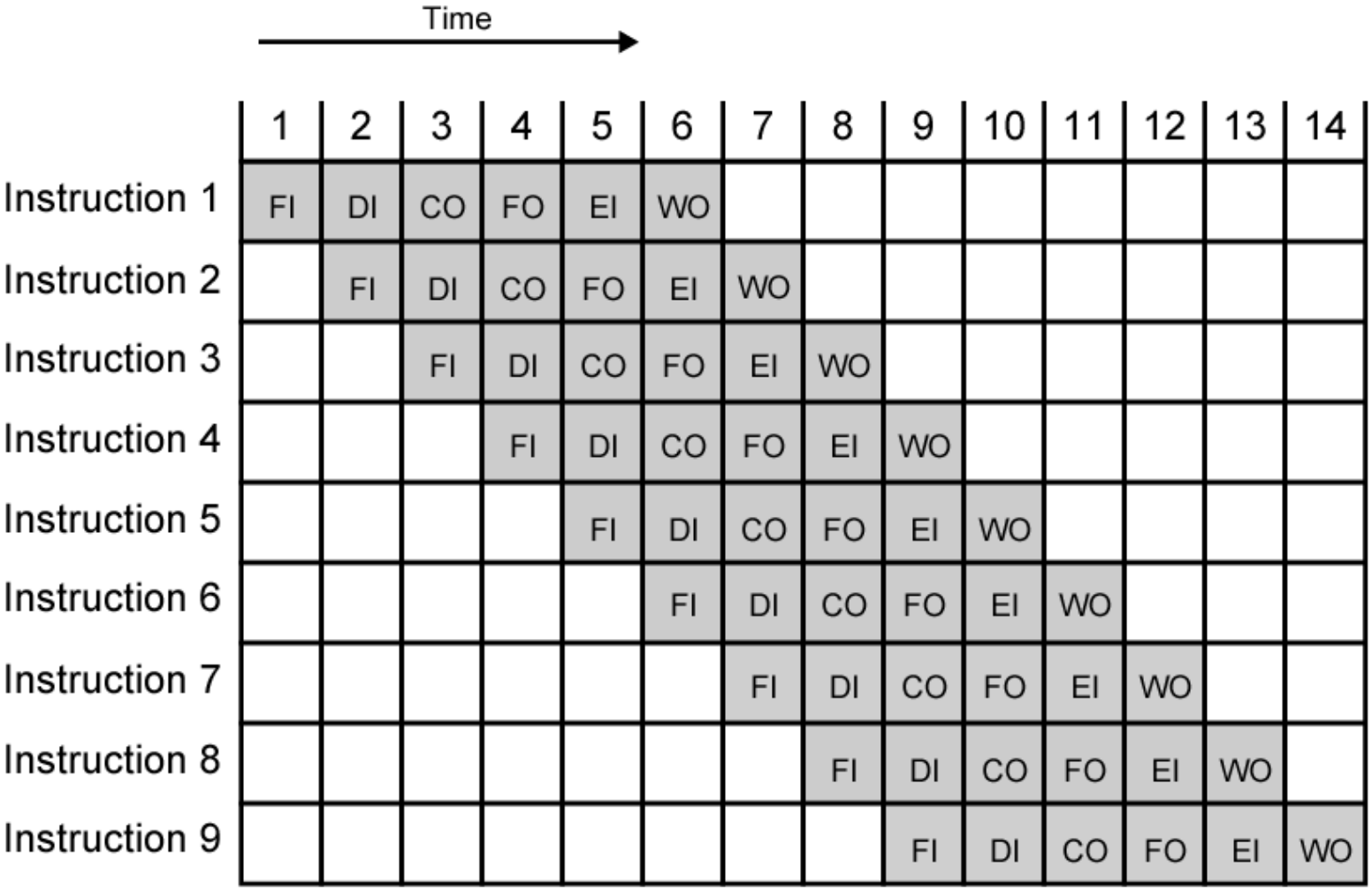
Four Segment instruction pipeline

- Contains
  - FI : fetch instruction
  - DA : Decode instruction and calculate effective address
  - FO :Fetch operand
  - EX: Execute instruction

Six stage pipeline

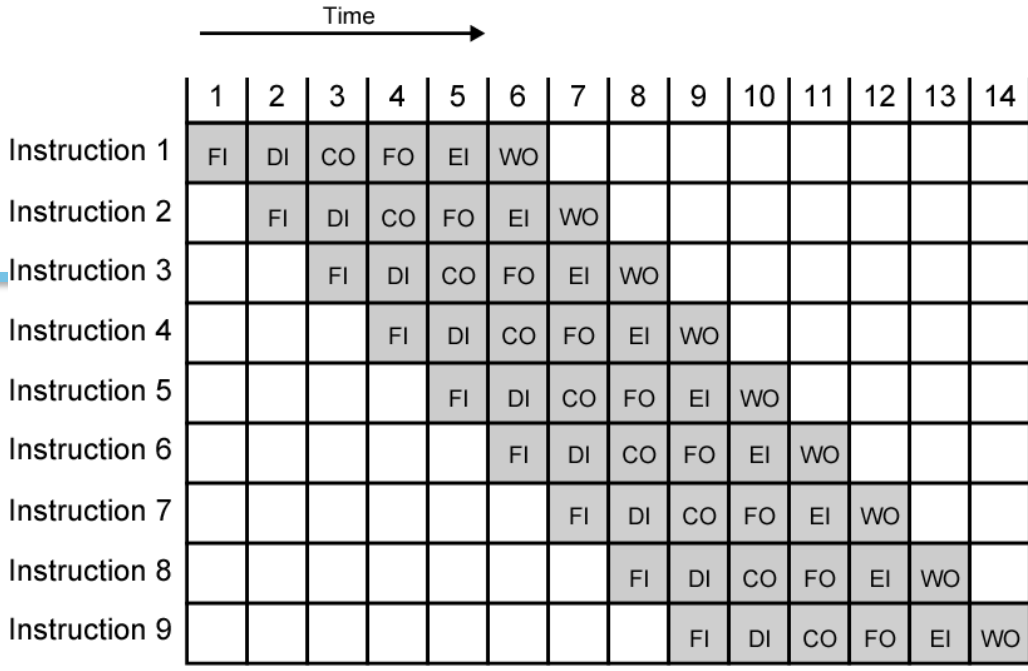
- Contains
  - FI : fetch instruction
  - DI : Decode instruction
  - CO : calculate effective address
  - FO :Fetch operand
  - EI : Execute instruction
  - WO : Write Operand

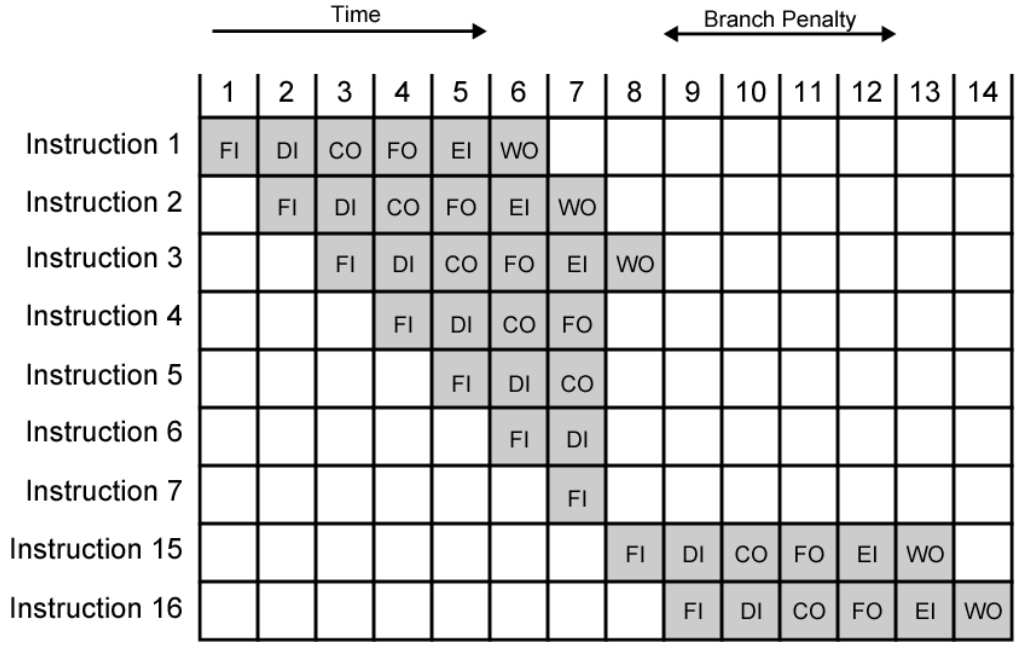
Timing Diagram for Instruction Pipeline Operation



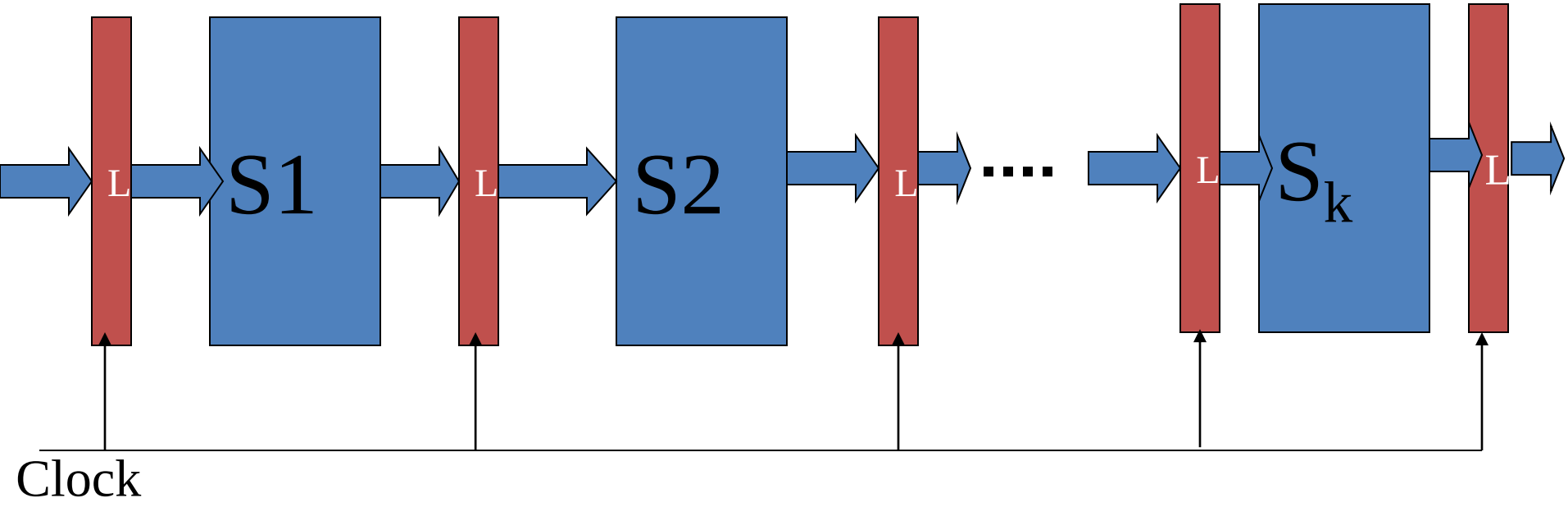
Important Points to be noted

- Do all the instructions need all the stages?
- At t=6, WO, FO and FI accesses the memory. Is there any issue?
- Is there any implication on having different time duration for different stages?
- Any issues with conditional branch?
- Dependency of CO stage on register used in previous stage





Structure of a pipeline



Classification

- Arithmetic pipelining
- Instruction pipelining
- Processor pipelining
- Unifunction and multifunction pipelining
- Static and Dynamic pipelining
- Scalar and Vector pipelining

Arithmetic pipelining

- Arithmetic and logic units of a computer can be segmentized for pipeline operations
- Usually found in high speed computers
- Example:
  - Star 100 → 4 stage
  - TI-ASC → 8 stage
  - Cray-1 → 14 stage
  - Cyber 205 → 26 stages
  - Intel Cooper Lake (3rd Gen Intel Xeon) = 14 stages
- Floating point adder pipeline

$$X = A*2^a$$

$$Y = B*2^b$$

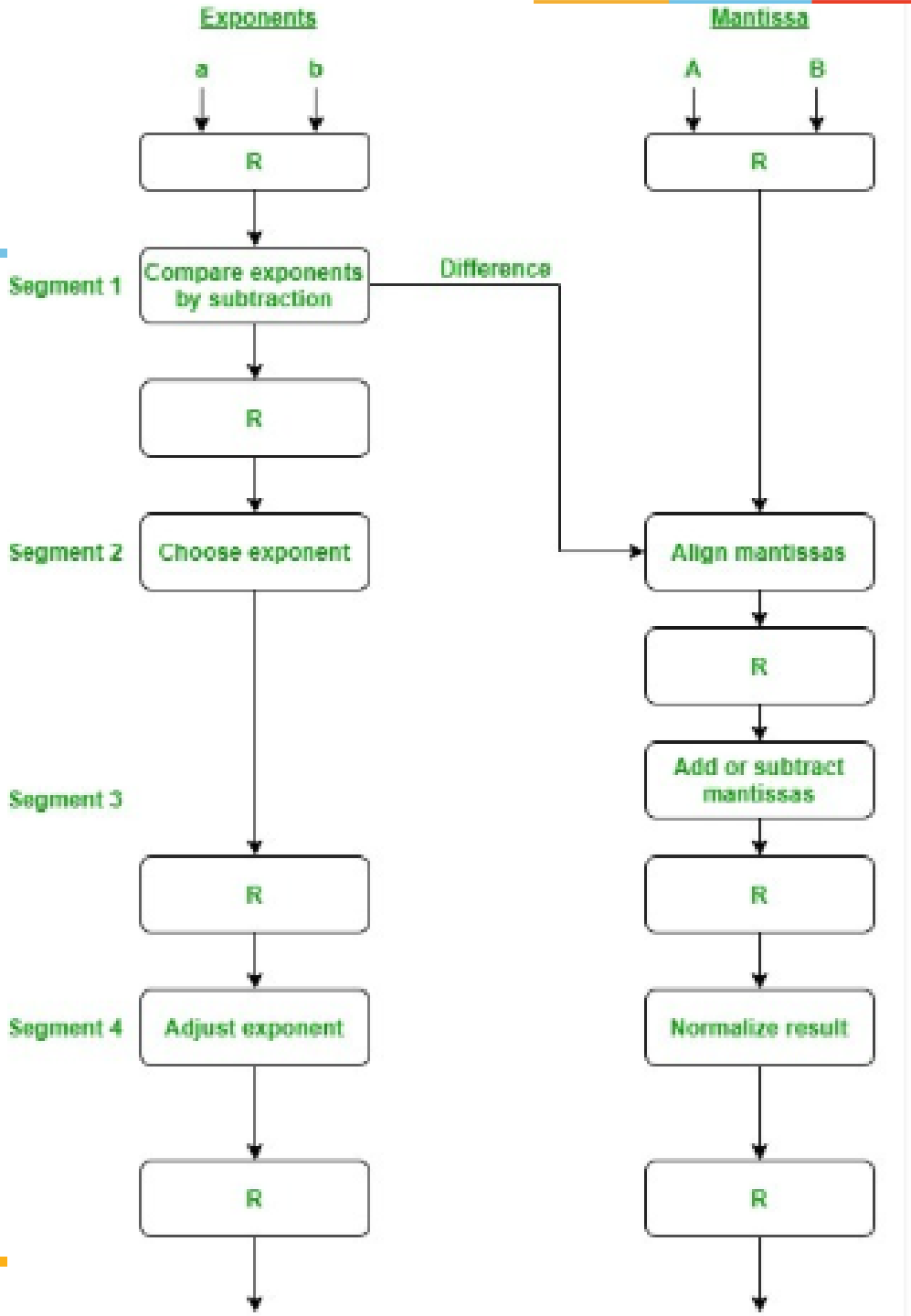
Floating point addition

1. Compare the exponents
2. Align the mantissas
3. Add or subtract the mantissas
4. Normalize the result

Numerical Example:

$X = 0.9504 * 10^3$

$Y = 0.8200 * 10^2$

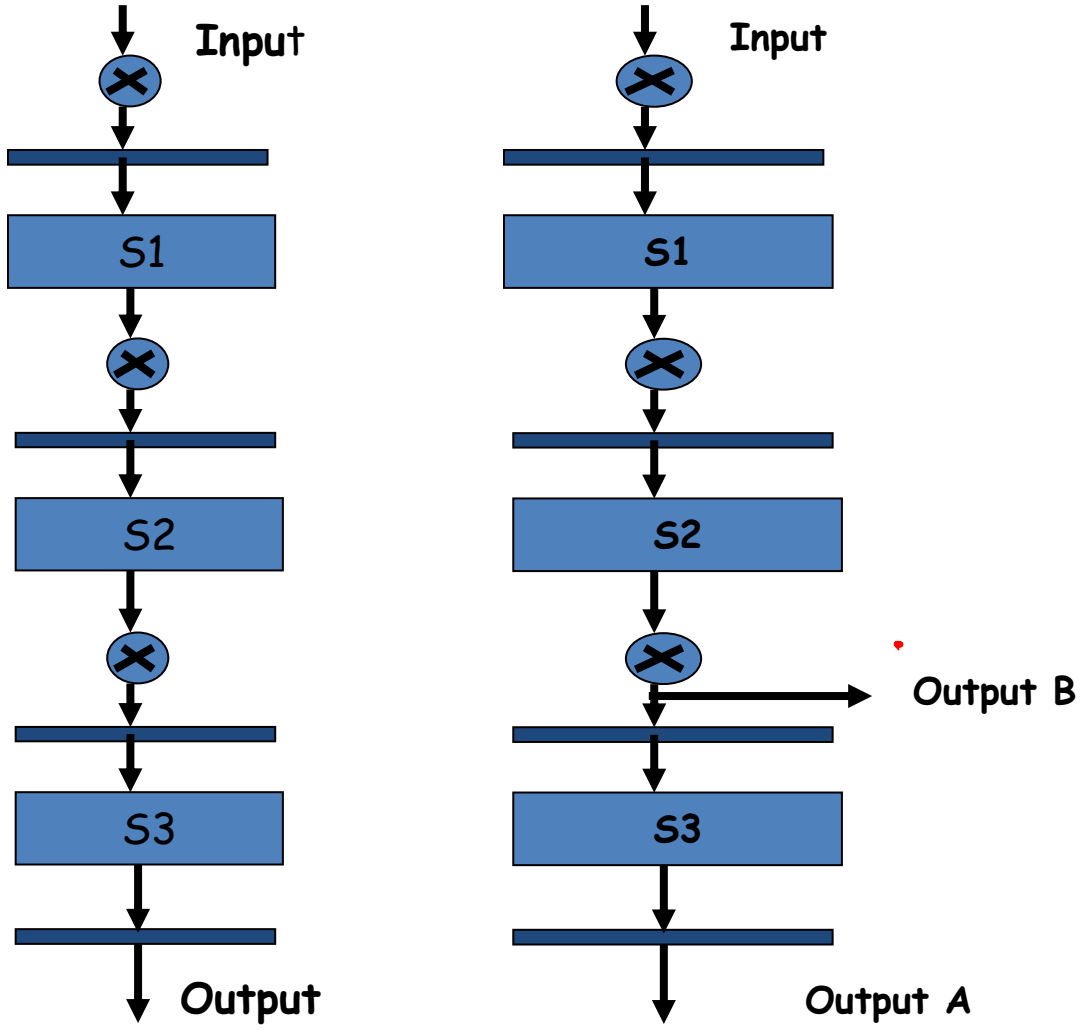


Instruction pipelining

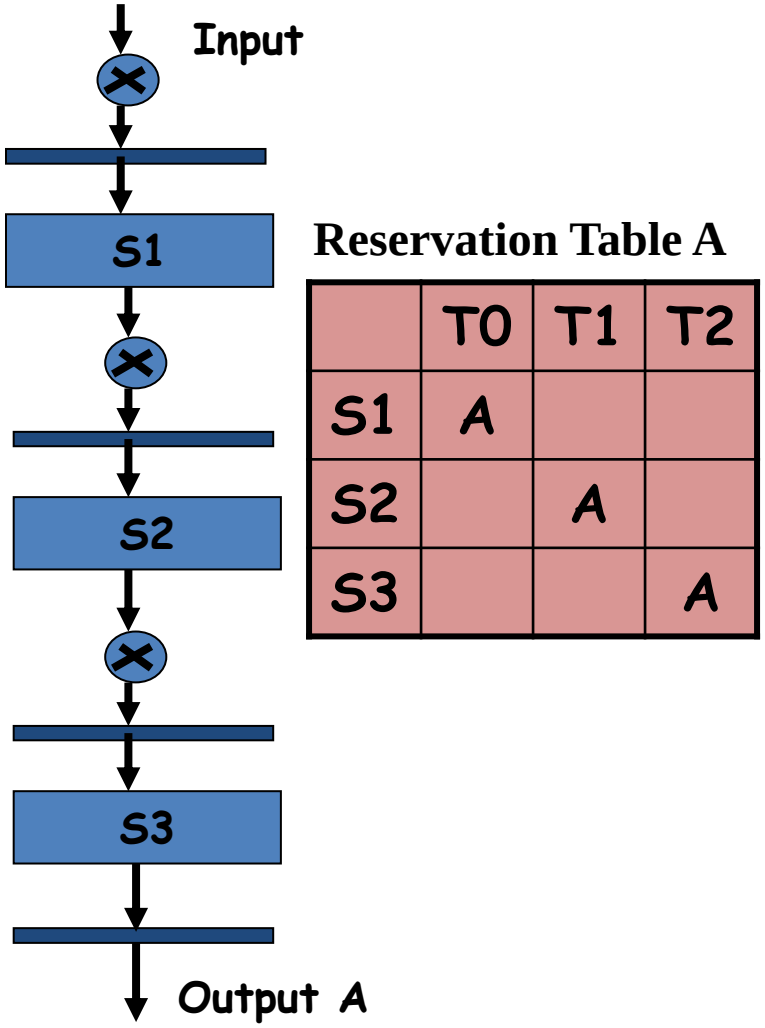
- The execution of a stream of instructions can be pipelined by overlapping the execution of the current instruction with the fetch, decode……of subsequent instructions
- Sequence of steps followed in most general purpose computer to process instruction
  1. IF: Fetch the instruction from memory
  2. ID: Decode the instruction
  3. CO: Calculate the effective address
  4. FO: Fetch the operands from memory
  5. EI: Execute the instruction
  6. WO: Store the result in the proper place

Unifunction and multifunction pipelining

- Unifunction
  - Pipeline with a fixed and dedicated function
  - Ex: Floating point adder
- Multifunction
  - Pipeline may perform different functions

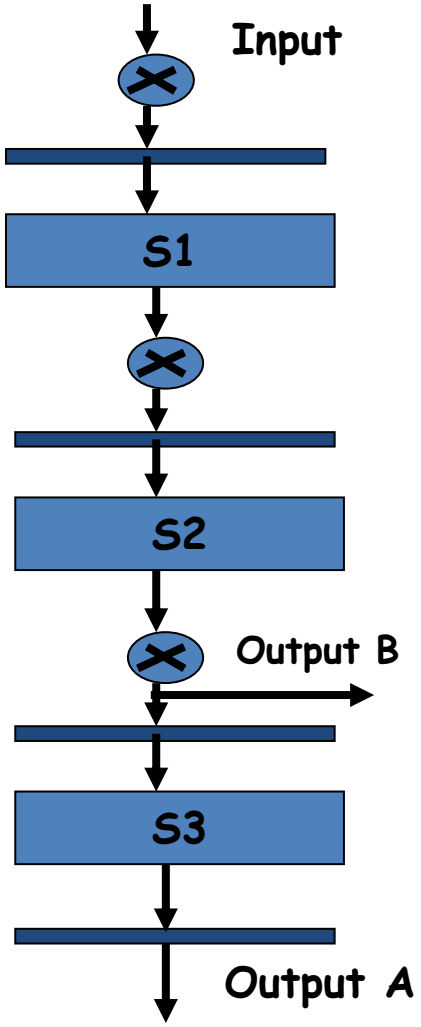


Uni-function Vs Multifunction



Reservation Table A

	T0	T1	T2
S1	A		
S2		A	
S3			A



Reservation Table A

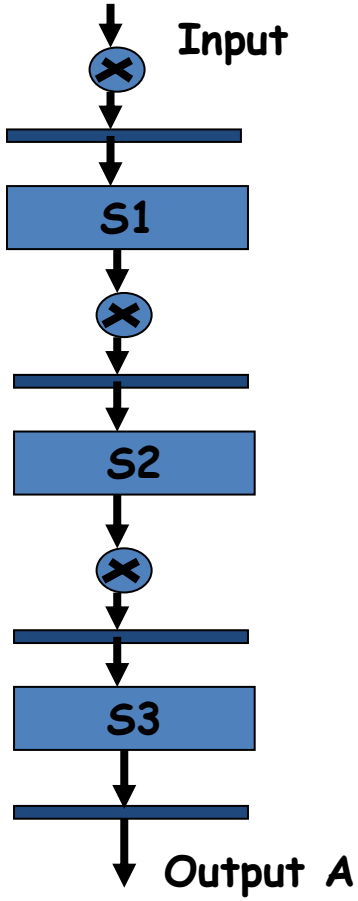
	T0	T1	T2
S1	A		
S2		A	
S3			A

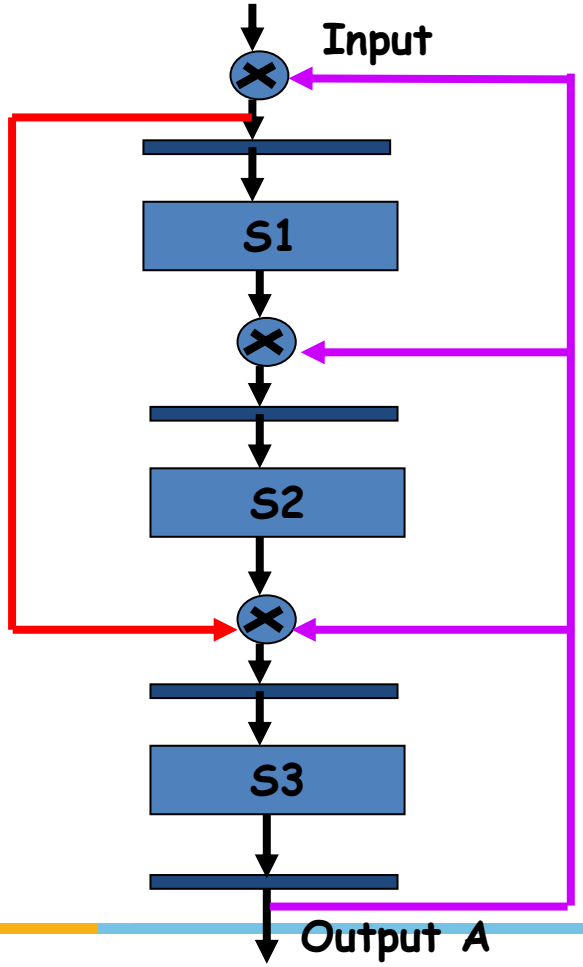
Reservation Table B

	T0	T1
S1	B	
S2		B

Linear and Nonlinear Pipelines

- Linear Pipeline: Without feed forward and feed back connection
- Nonlinear Pipeline with feed forward and/or feed back connection

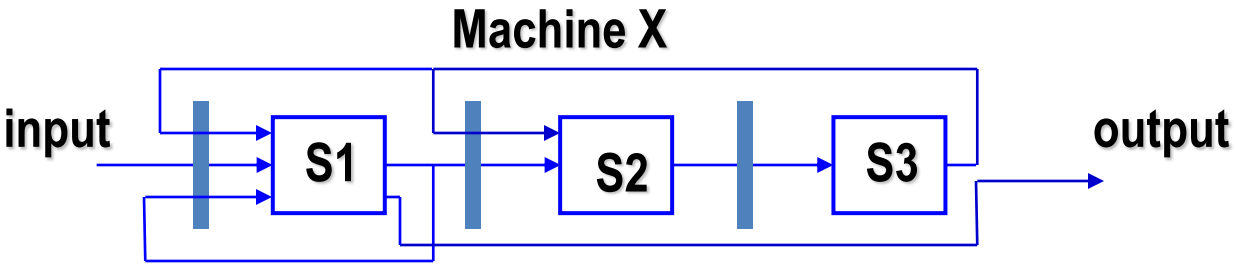




Reservation Table

- Is a two dimensional chart
- Used to show how successive pipeline stages are utilized or reserved

Reservation Table



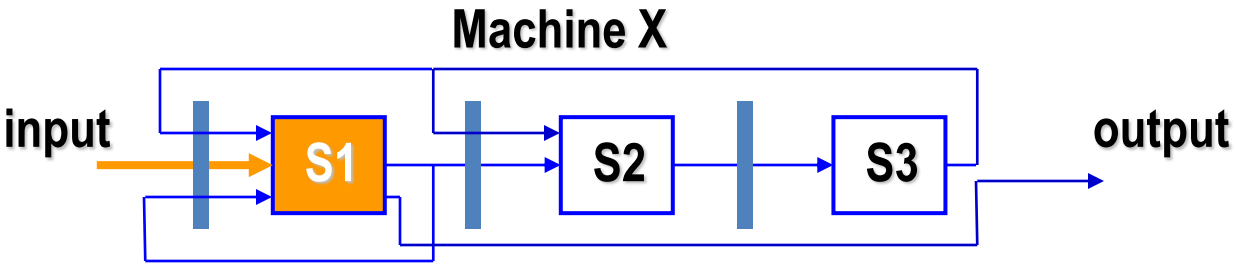
Reservation Table

Time →

Stage ↓

	0	1	2	3	4	5	6	7
S1	X	X					X	X
S2			X		X			
S3				X		X		

Reservation Table



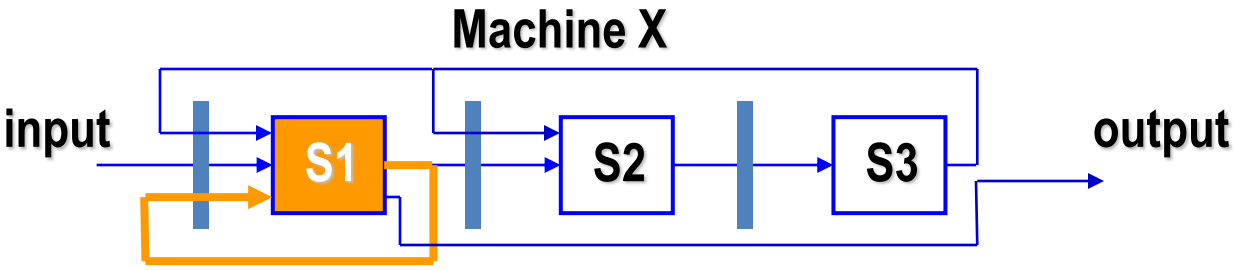
Reservation Table

Time →

Stage ↓

	0	1	2	3	4	5	6	7
S1	X	X					X	X
S2			X		X			
S3				X		X		

Reservation Table



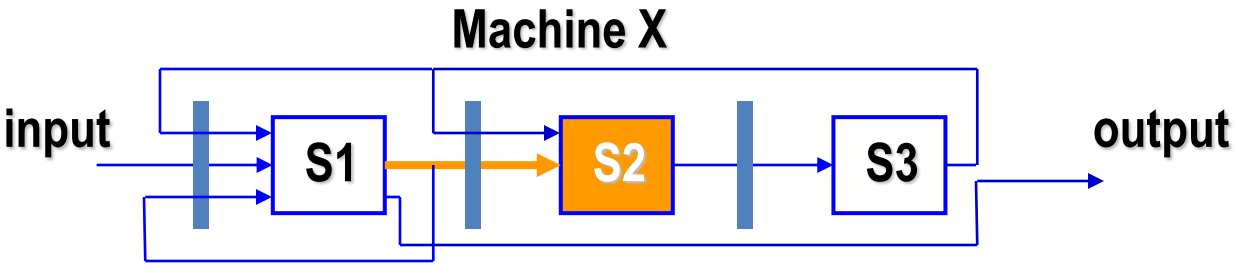
Reservation Table

Time →

Stage ↓

	0	1	2	3	4	5	6	7
S1	X	X					X	X
S2			X		X			
S3				X		X		

Reservation Table



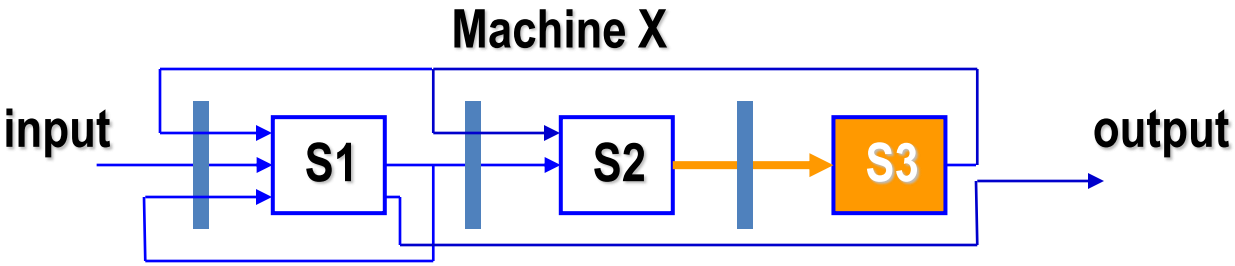
Reservation Table

Time →

Stage ↓

	0	1	2	3	4	5	6	7
S1	X	X					X	X
S2			X		X			
S3				X		X		

Reservation Table



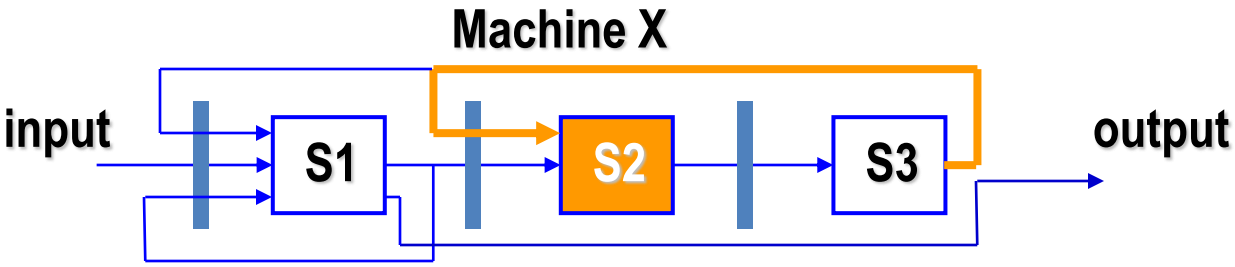
Reservation Table

Time →

Stage ↓

	0	1	2	3	4	5	6	7
S1	X	X					X	X
S2			X		X			
S3				X		X		

Reservation Table



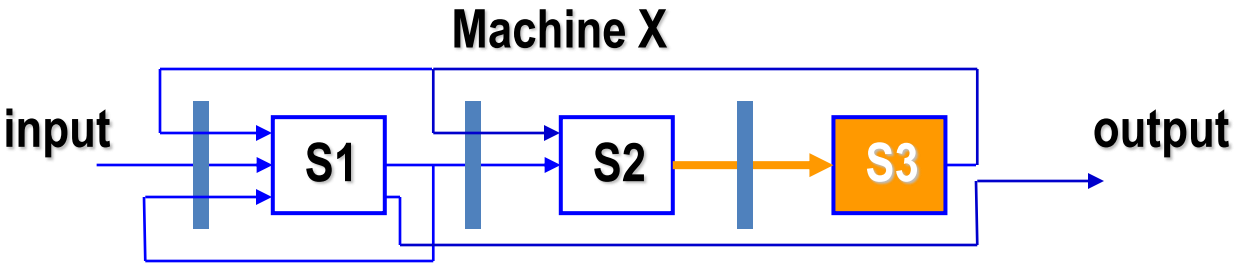
Reservation Table

Time →

Stage ↓

	0	1	2	3	4	5	6	7
S1	X	X					X	X
S2			X		X			
S3				X		X		

Reservation Table



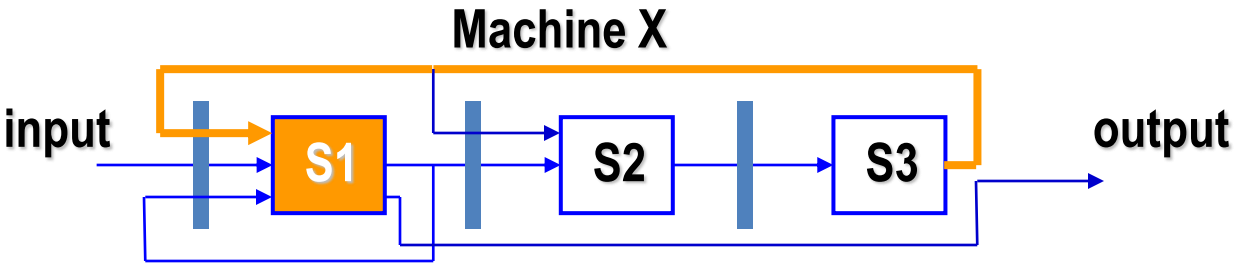
Reservation Table

Time →

Stage ↓

	0	1	2	3	4	5	6	7
S1	X	X					X	X
S2			X		X			
S3				X		X		

Reservation Table



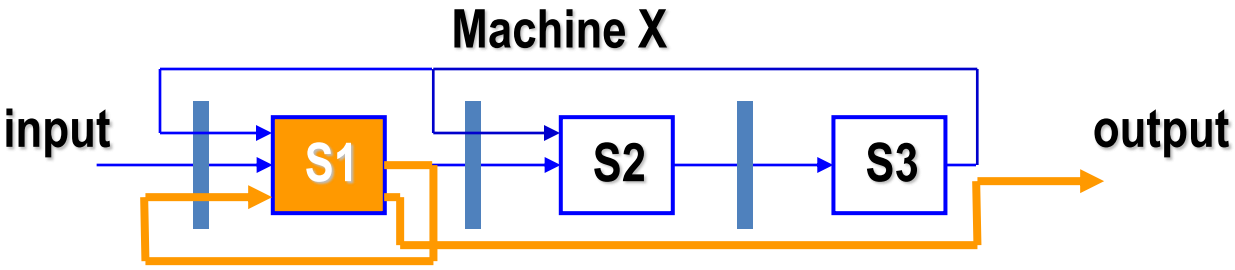
Reservation Table

Time →

Stage ↓

	0	1	2	3	4	5	6	7
S1	X	X					X	X
S2			X		X			
S3				X		X		

Reservation Table



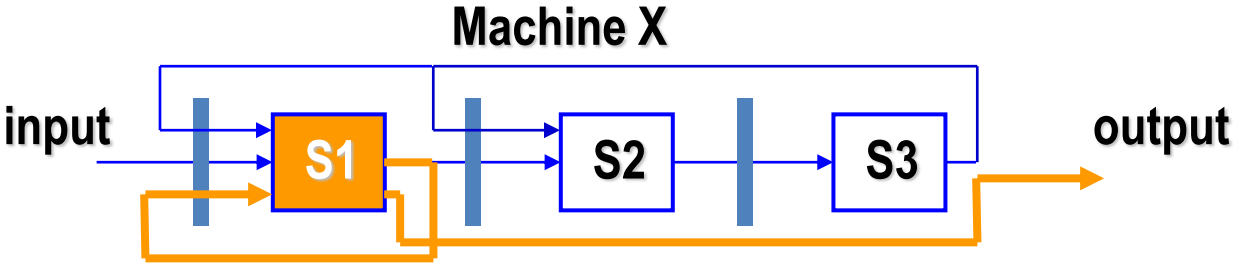
Reservation Table

Time →

Stage ↓

	0	1	2	3	4	5	6	7
S1	X	X					X	X
S2			X		X			
S3				X		X		

Reservation Table



Reservation Table

Time →

Stage ↓

	0	1	2	3	4	5	6	7
S1	X	X					X	X
S2			X		X			
S3				X		X		

Stage ↓

	0	1	2	3	4	5	6	7
S1	X	X					X	X
S2			X		X			
S3				X		X		

Static and Dynamic pipelining

- Dynamic pipeline allows more frequent changes in its configuration
- Require more elaborate sequencing and control mechanisms

Scalar and Vector pipelining

- Based on the operand types or instruction type
- Scalar pipeline processes scalar operands
- Vector pipeline operate on vector data and instructions.

Important Terms

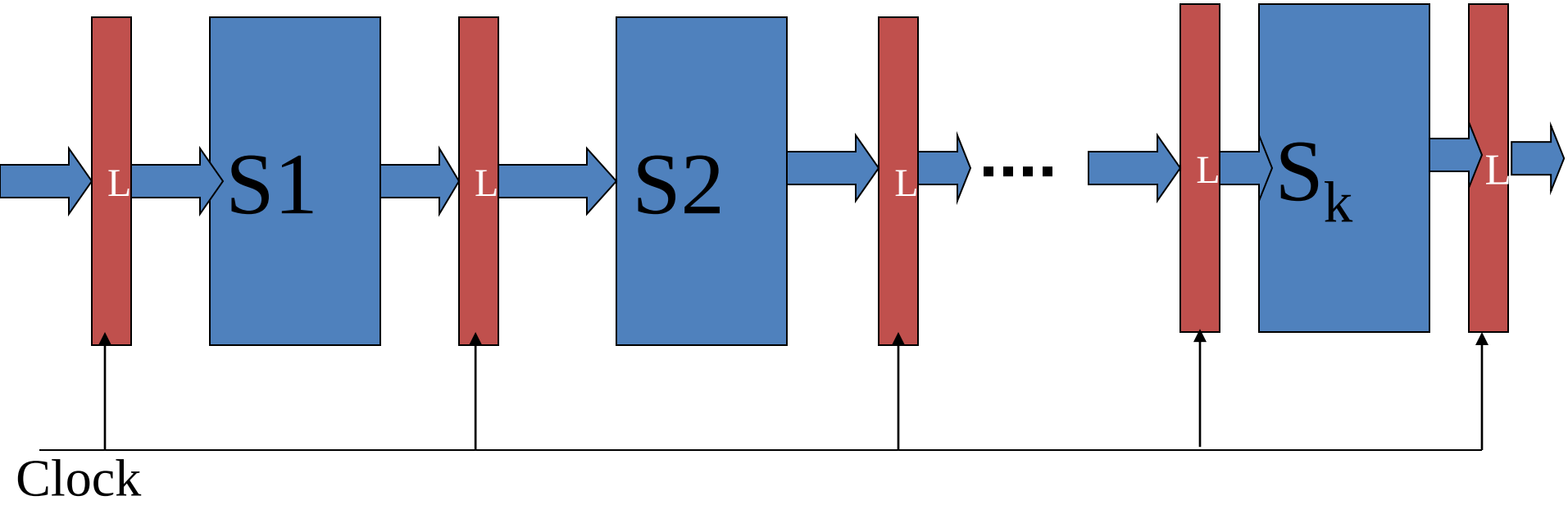


Fig: Structure of a pipeline

Important Terms

Clock period: τ

τ_i: time delay of S_i stage

τ_L: time delay of latch

$$\tau = \max\{\tau_i\} + \tau_L$$

Pipeline processor frequency $f = 1/\tau$

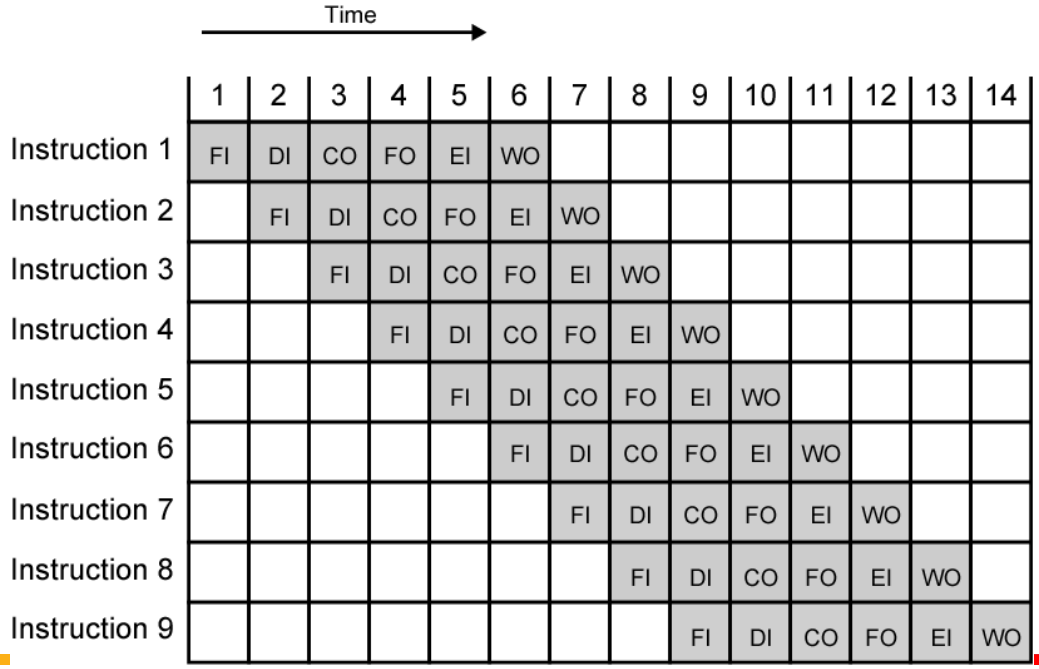
Important Terms

Time taken to complete n tasks by k stage pipeline is

$$T_k = [k + (n-1)]\tau$$

Time taken by the nonpipelined processor

?



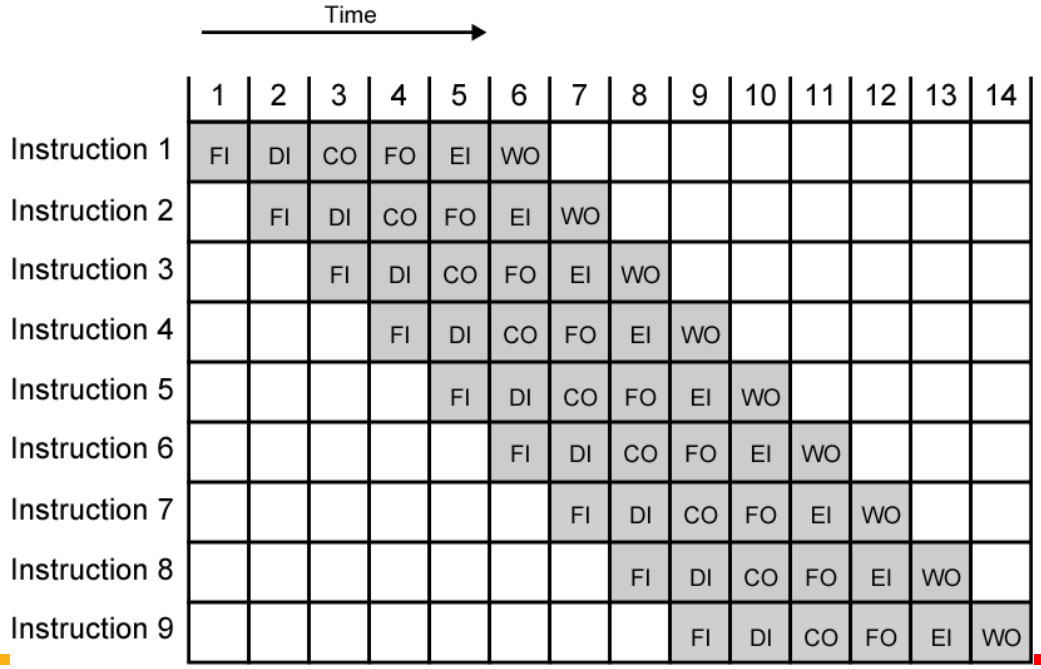
Important Terms

Time taken to complete n tasks by k stage pipeline is

$$T_k = [k + (n-1)]\tau$$

Time taken by the nonpipelined processor

$$T_1 = k^* n^* \tau$$



Important Terms

- Speedup: speedup of a k-stage linear-pipeline over an equivalent nonpipelined processor

$$S_k = \frac{T_1}{T_k}$$

$$= \frac{n*k*\tau}{[k+(n-1)]\tau}$$

$$= \frac{n*k}{[k+(n-1)]}$$

- The maximum speedup is $S_k \rightarrow k$ when $n \rightarrow INF$
- Maximum speedup is very difficult to achieve because of data dependencies between successive tasks, program branches, interrupts etc.

Important Terms

Efficiency: the ratio of actual speedup to ideal speedup

$$\eta = \frac{n.k}{k.[k+(n-1)]}$$

$$= \frac{n}{[k+(n-1)]}$$

- Maximum efficiency

$$\eta \to 1 \text{ as } n \to \infty$$

- Implies that the larger the number of tasks flowing through the pipeline, the better is its efficiency
- In steady state of a pipeline, we have n >> k, then efficiency should approach 1
- However, this ideal case may not hold all the time because of program branches and interrupts and data dependencies

Important Terms

Throughput : The number of tasks that can be completed by a pipeline per unit time

$$H_k = \frac{n}{[k+(n-1)]\tau} = \frac{nf}{[k+(n-1)]} = \eta f$$