Parul® University
Vadodara, Gujarat

NAAC GRADE A++

Information and Communication Technology

Computer Organization-1

Dr. M Krishnam Raju
Assistant Professor
Electronics & Communication Engineering

Content

1. **Register Transfer:** Register Transfer language, Bus design using multiplexer and Tristate buffer, Memory Transfers, Arithmetic Micro- Operations, Logic Micro-Operations, Shift Micro-Operations, Arithmetic logical shift unit.
2. **Basic Computer Design:** Instruction codes, Computer registers, computer instructions, Timing and Control, Instruction cycle, Memory- Reference Instructions, Register Reference Instructions, IO Reference Instructions, Interrupt, Design of Accumulator Unit.

Register Transfer language

- Register Transfer Language (RTL) is a low-level language that is used to describe the functioning of a digital circuit and, more specifically, the transfer of information between registers.
- It provides how data moves from one register to the other and how data is processed within the digital system.
- Through RTL, there is a capability of creating abstraction levels where high-level design descriptions can be created and easily linked to low-level hardware implementation in designing, simulating, as well as synthesizing digital circuits.
- **Micro-operations:** The operation executed on the data store in registers are called micro-operations. They are detailed low-level instructions used in some designs to implement complex machine instructions.
- **Register Transfer:** The information transformed from one register to another register is represented in symbolic form by replacement operator is called Register Transfer.

Register Transfer language

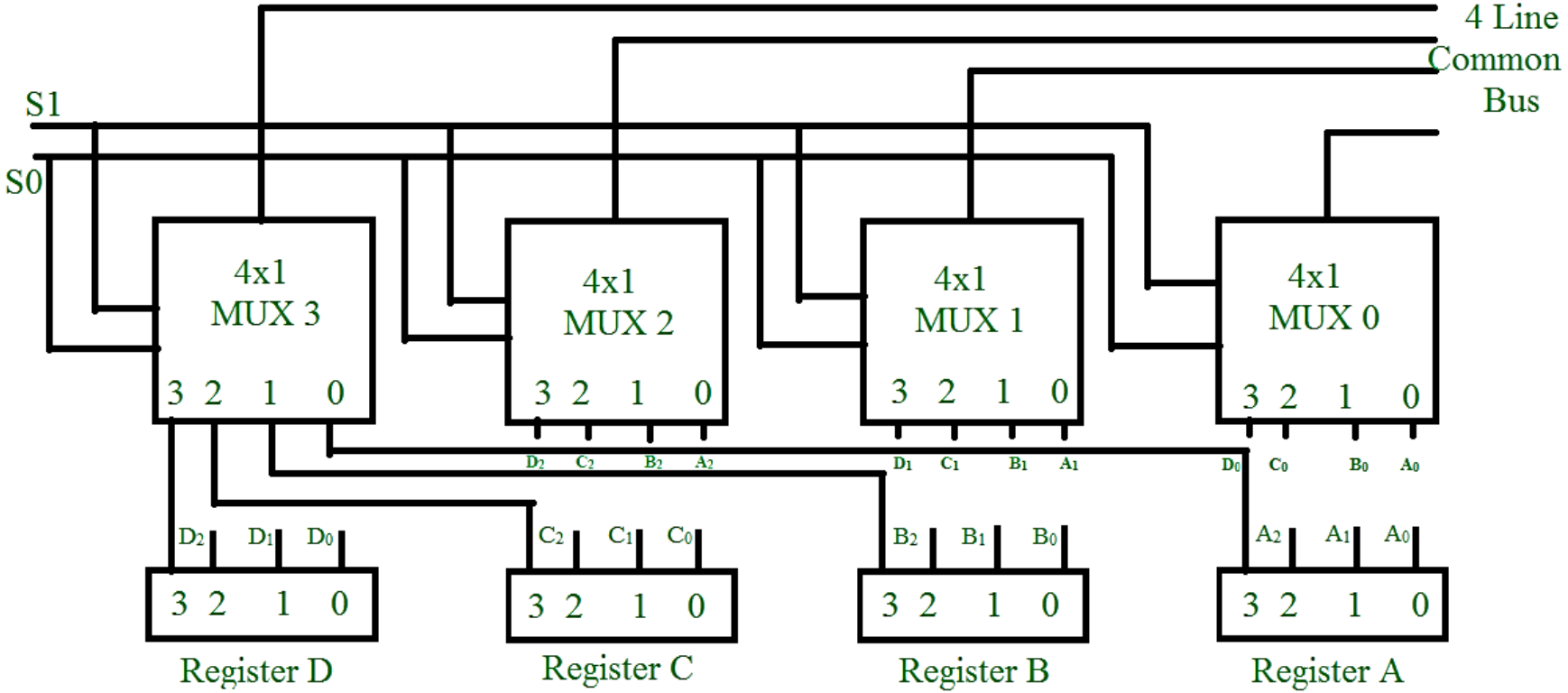
Simple Data Transfer:
R1 ← R2 (Copy the contents of register R2 into register R1)

Arithmetic Operation:
R3 ← R1 + R2 (Add R1 and R2, store the result in R3)

Memory Read:
R1 ← [MAR] (Load into R1 the value stored at the memory address given by the Memory Address Register, MAR)

Memory Write:
[MAR] ← R1 (Store the contents of R1 into the memory location pointed to by MAR)

Bus design using multiplexer and Tristate buffer



A typical computer has many registers and we need to transfer the information between these registers. A way to transfer the information is using the common bus system. In this article we shall discuss the common bus system using multiplexers.

Bus Design Using Multiplexer and Tristate Buffer

- The construction of this bus system for 4 registers is shown above. The bus consists of 4×1 multiplexers with 4 inputs and 1 output and 4 registers with bits numbered 0 to 3. There are 2 select inputs S0 and S1 which are connected to the select inputs of the multiplexers.
- The output 1 of register A is connected to input 0 of MUX 1 and similarly other connections are made as shown in the diagram. The data transferred to the bus depends upon the select lines.

Select Lines combination S1S0	Register Selected
00	Register A
01	Register B
10	Register C
11	Register D

Bus Design Using Multiplexer and Tristate Buffer

Feature	Multiplexer (MUX)	Tri-state Buffer
Main Function	Selects one input from many and forwards it to output	Connects a device to a shared bus or disconnects it
Control Signal	Uses select lines to pick the input	Uses an enable signal to connect/disconnect output
Output Behavior	Always outputs one selected input	Output can be 0, 1, or High-Impedance (Z)
Connection Style	One device is connected via MUX logic	All devices are physically connected, but only one drives the bus
Use Case	Useful when data selection is needed (e.g., choosing data source)	Useful for bus systems with many drivers (e.g., memory, registers)
Scalability	Harder to manage with many inputs	Easier to add more devices
Example	4-to-1 MUX selects 1 input out of 4	A register drives a bus only when its enable is active

Memory Transfers

Memory transfers refer to operations where data is moved between memory and registers.

Register ← Memory

- Load data from memory into a register.
- RTL Notation Example: R1 ← M[1000]

Memory ← Register

- Store data from a register into memory.
- RTL Notation Example: M[1000] ← R1

Memory Addressing: You often need a register that holds the memory address (like a Memory Address Register, MAR) and a register to hold the data (MDR — Memory Data Register).

Control Signals: Micro-operations send signals to specify whether it's a read or write operation.

Arithmetic Micro-Operations

Arithmetic Micro-Operations are basic operations performed at the hardware level, typically on the contents of registers.

Operation	RTL Notation	Meaning
Transfer	R1 ← R2	Copy the contents of register R2 into R1.
Addition	R1 ← R2 + R3	Add contents of R2 and R3 and store the result in R1.
Subtraction	R1 ← R2 - R3	Subtract contents of R3 from R2 and store in R1.
Increment	R1 ← R1 + 1	Add 1 to the contents of R1.
Decrement	R1 ← R1 - 1	Subtract 1 from the contents of R1.
Add with Carry	R1 ← R2 + R3 + Cin	Add R2, R3, and carry-in bit (Cin).
Subtract with Borrow	R1 ← R2 - (R3 + 1)	Subtract R3 and borrow bit from R2.
Two's Complement	R1 ← -R2	Take two's complement of R2 and store in R1 (negate).
Clear	R1 ← 0	Set all bits of R1 to 0.

Logic Micro-Operations

Operation	Symbol	Meaning
AND	$\wedge$ or AND	Bitwise AND
OR	$\vee$ or OR	Bitwise OR
XOR	$\oplus$ or XOR	Bitwise Exclusive-OR
NOT	$\neg$ or NOT	Bitwise Complement

RTL Statement	Meaning
R2 ← R1 AND R3	Perform bitwise AND between R1 and R3, store result in R2
R2 ← R1 OR R3	Perform bitwise OR between R1 and R3, store result in R2
R2 ← R1 XOR R3	Perform bitwise XOR between R1 and R3, store result in R2
R2 ← NOT R1	Perform bitwise NOT (complement) of R1, store in R2

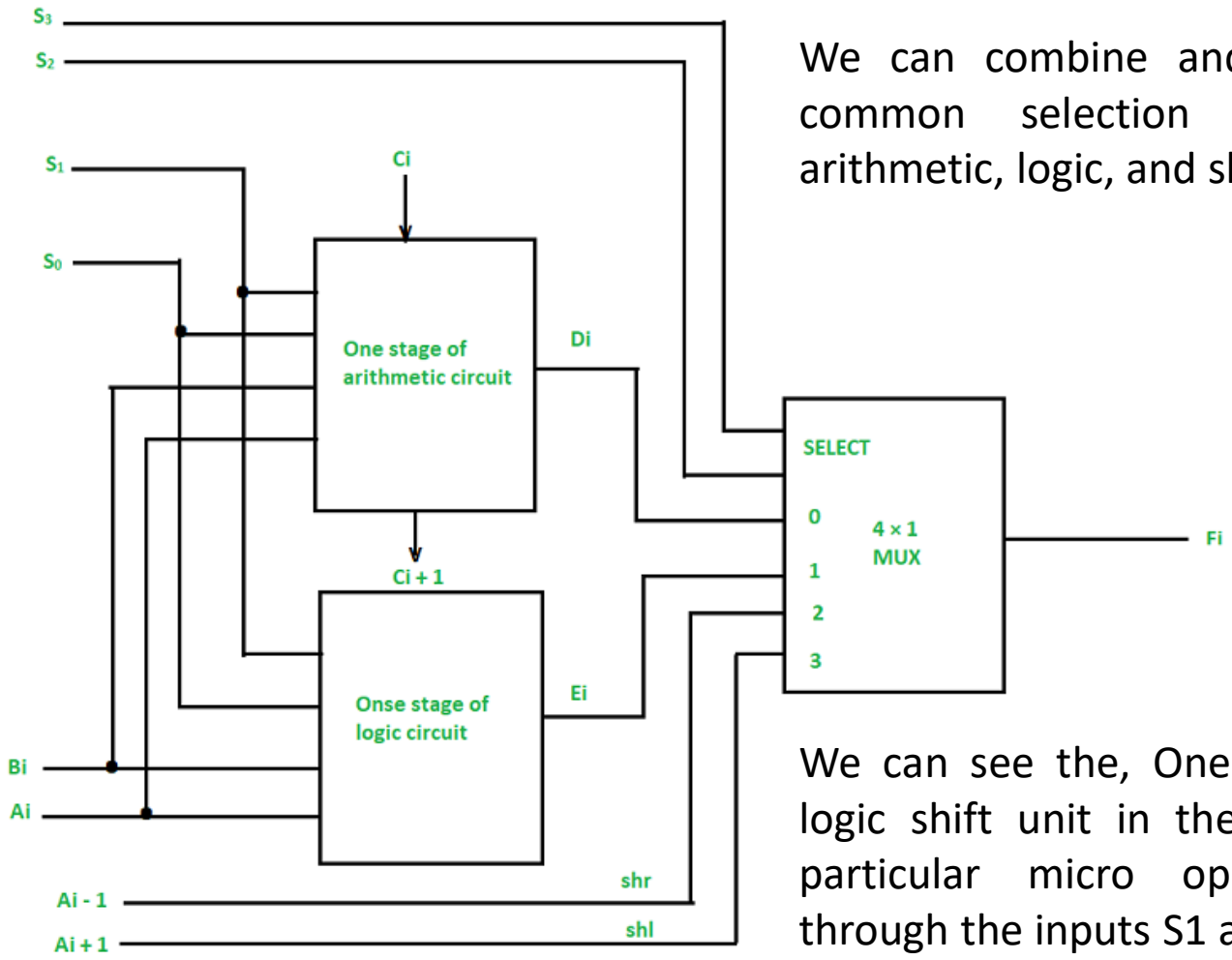
Shift Micro-Operations

Operation	RTL Representation	Description
Logical shift left	R ← R << 1	Shift all bits left; insert 0 into LSB.
Logical shift right	R ← R >> 1	Shift all bits right; insert 0 into MSB.
Arithmetic shift right	R ← ashR(R)	Shift right; copy MSB into new MSB.
Rotate left	R ← ROL R	Rotate bits left; MSB to LSB.
Rotate right	R ← ROR R	Rotate bits right; LSB to MSB.

Arithmetic Logical Shift Unit

- Arithmetic Logic Shift Unit (ALSU) is a member of the Arithmetic Logic Unit (ALU) in a computer system. It is a digital circuit that performs logical, arithmetic, and shift operations.
- Rather than having individual registers calculating the micro operations directly, the computer deploys a number of storage registers which is connected to a common operational unit known as an arithmetic logic unit or ALU.
- The Arithmetic Logic Unit performs an operation that leads as a result and gets transferred to a destination register.
- Arithmetic Logic Unit may be a combinatory circuit in order that the complete register transfer operation from the supply registers through the ALU and into the destination register is performed throughout one clock pulse amount.
- Sometimes, the shift micro operations are performed in a separate unit, but sometimes it is made as a part of full ALU.

Arithmetic Logical Shift Unit



We can combine and make one ALU with common selection variables by adding arithmetic, logic, and shift circuits.

We can see the, One stage of an arithmetic logic shift unit in the diagram above. Some particular micro operations are selected through the inputs S1 and S0.

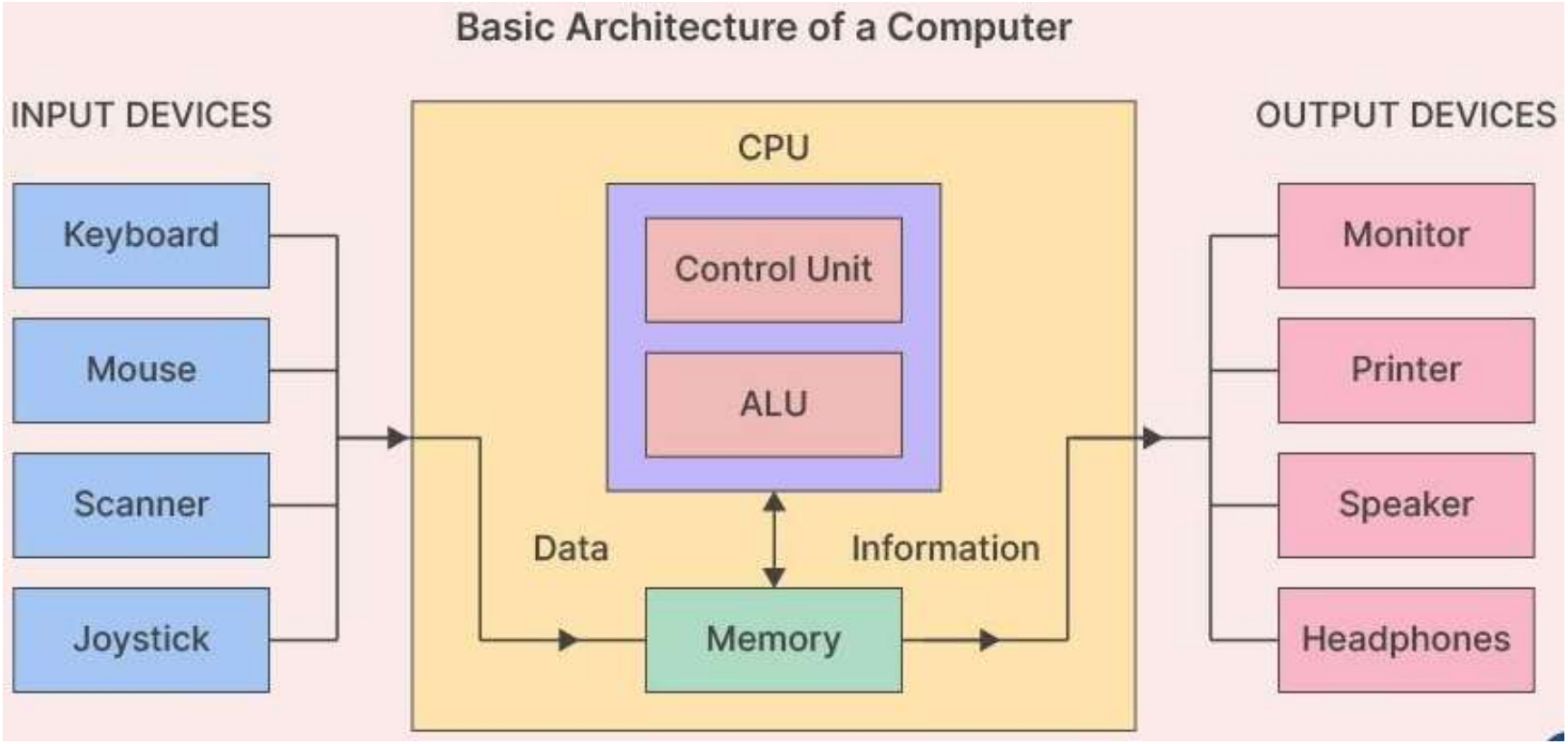
Arithmetic Logical Shift Unit

4 x 1 multiplexer at the output chooses between associate arithmetic output between Ei and a logic output in Hi. The data in the multiplexer are selected through inputs S3 and S2 and the other two data inputs to the multiplexer obtain the inputs Ai – 1 for the shr operation and Ai + 1 for the shl operation.

Operation Select					Operation	Function
S_3	S_2	S_1	S_0	C_{in}		
0	0	0	0	0	F = A	Transfer A
0	0	0	0	1	F = A + 1	Increment A
0	0	0	1	0	F = A + B	Addition
0	0	0	1	1	F = A + B + 1	Add with carry
0	0	1	0	0	F = A + B'	Subtract with borrow
0	0	1	0	1	F = A + B' + 1	Subtraction
0	0	1	1	0	F = A - 1	Decrement A
0	0	1	1	1	F= A	Transfer A
0	1	0	0	×	F = A ^ B	AND
0	1	0	1	×	F = A v B	OR
0	1	1	0	×	F = A XOR B	XOR
0	1	1	1	×	F = A'	Complement A
1	0	×	×	×	F = shr A	Shift right A into F
1	1	×	×	×	F = shl A	Shift left A into F

The circuit whose one stage is given in the below diagram provides 8 arithmetic operations, 4 logic operations, and 2 shift operations, and Each operation is selected by the 5 variables S3, S2, S1, S0, and Cin.

Basic Computer Design



Basic Computer Design

- Input Unit: Devices like a keyboard or mouse that feed data into the computer.
- Output Unit: Devices like a monitor or printer that show results.
- Memory Unit: Stores data and instructions.
  - Primary Memory (RAM, ROM)
  - Secondary Memory (hard drive, SSD)
- Central Processing Unit (CPU): The "brain" of the computer.
  - Control Unit (CU): Directs the flow of data.
  - Arithmetic Logic Unit (ALU): Performs calculations and logic operations.
- Bus: A communication system that transfers data between components (includes data bus, address bus, and control bus).

Instruction Codes

Types of Instructions

Depending on how the instruction is interpreted, it could be:

Memory-reference instructions are the most common type of instructions in a computer's instruction set.

- These instructions typically involve accessing data stored in the system's memory. To perform operations like reading a value, modifying it, or writing it back, the instruction must specify a memory address where the data is located.
- Examples of memory-reference instructions include loading data from memory into a register, storing data from a register back to memory, and performing arithmetic operations directly involving memory locations.
- Because so much of a computer's work involves moving and manipulating data stored in memory, these instructions are fundamental to system operation.

Instruction Codes

Types of Instructions

Register-reference instructions are designed to operate directly on the CPU's internal registers rather than external memory.

- Registers are small, very fast storage locations within the CPU used to hold temporary data and intermediate results.
- Instructions that work directly with registers are typically faster than memory-reference instructions because they avoid the time-consuming process of accessing external memory.
- Examples include operations like clearing a register, complementing its content, shifting its bits, or performing simple arithmetic operations between registers. These instructions are essential for speeding up computations and efficient CPU operations.

Register Transfer language

Types of Instructions

Input-output (I/O) instructions handle the communication between the CPU and external devices such as keyboards, monitors, printers, and disk drives.

- Rather than operating on memory or internal registers, these instructions are designed to read data from an input device or send data to an output device.
- I/O instructions are crucial for making the computer interact with the outside world, allowing users to input data for processing and receive results after processing.
- Specialized I/O control instructions are used to manage the transfer of information, handle device status checking, and coordinate device communication without interfering with the main computational work.

Computer registers

- Registers are small, fast storage locations inside a computer's CPU used to hold temporary data and instructions during processing.
- Registers are located directly inside the CPU (Central Processing Unit), making them the fastest type of memory.
- Registers operate at the same speed as the CPU clock, much faster than RAM or other types of memory.
- Registers are very small in size (typically 32-bit, 64-bit, or sometimes 128-bit), because large registers would slow down the CPU.
- Types of Registers
  - Accumulator (ACC): Stores results of arithmetic and logic operations.
  - Program Counter (PC): Holds the address of the next instruction to execute.
  - Instruction Register (IR): Holds the current instruction being executed.
  - Memory Address Register (MAR): Holds the address of memory to access.
  - Memory Data Register (MDR): Holds the data being transferred to/from memory.
  - Status Register/Flag Register: Holds flags like zero, carry, overflow, etc.

Computer Instructions

- Computer instructions are commands given to a computer's processor to perform specific operations.
- Every task a computer does (adding numbers, displaying a webpage, saving a file) is broken down into very small steps — instructions.
- Types of Instructions
  - Arithmetic Instructions (e.g., Add, Subtract)
  - Logical Instructions (e.g., AND, OR, NOT operations)
  - Data Transfer Instructions (e.g., Move data from one place to another)
  - Control Instructions (e.g., Jump, Branch, Call function)
- A computer's processor can only understand a specific set of instructions, called its Instruction Set (like x86, ARM, RISC-V).
- Instructions go through the Fetch → Decode → Execute cycle inside the CPU:
  - Fetch: Get the instruction from memory.
  - Decode: Understand what needs to be done.
  - Execute: Perform the action.

Timing and Control

- **Timing and Control Unit** of a microprocessor is a critical internal component responsible for coordinating all activities within the processor. It generates precise timing signals that regulate the execution of instructions, ensuring that each operation occurs in the correct sequence and at the right moment.
- **Control Unit** interprets the instructions fetched from memory and issues necessary control signals to direct the flow of data between the Arithmetic Logic Unit (ALU), registers, and external memory or I/O devices. It manages tasks such as instruction decoding, execution control, and synchronization of internal operations with external buses.
- **Timing signals** are derived from the system clock, and these signals help in managing machine cycles and instruction cycles, ensuring proper fetch, decode, execute, and write-back stages.

Timing and Control

- A time diagram is a graphical representation. The 8085 instruction timing diagram represents the execution time of each instruction in graphical format. Execution time is given in T-states.
- The 8085 microprocessor has a set of control signals and data signals that play an important role in the execution of instructions. In this article, I will explain in detail what a timing diagram is and how to draw a timing diagram of different instructions
  - **Clock Signal:** The time required to execute an instruction is called a clock cycle.
  - **Machine Cycle:** The time required to access memory or input/output devices is called a machine cycle. The 8085 has 5 basic machine cycles i.e., load opcode, read from memory, write to memory, read I/O, and write I/O.
  - **T-State:** A machine cycle and an instruction cycle take several clock periods. The portion of an operation performed in one system clock period is called a T-state.
  - **Control Signals:** The control signal controls the operations. Common signals are ALE (address block enable), RD (read), WR (write), and IO/M (input/output) memory.

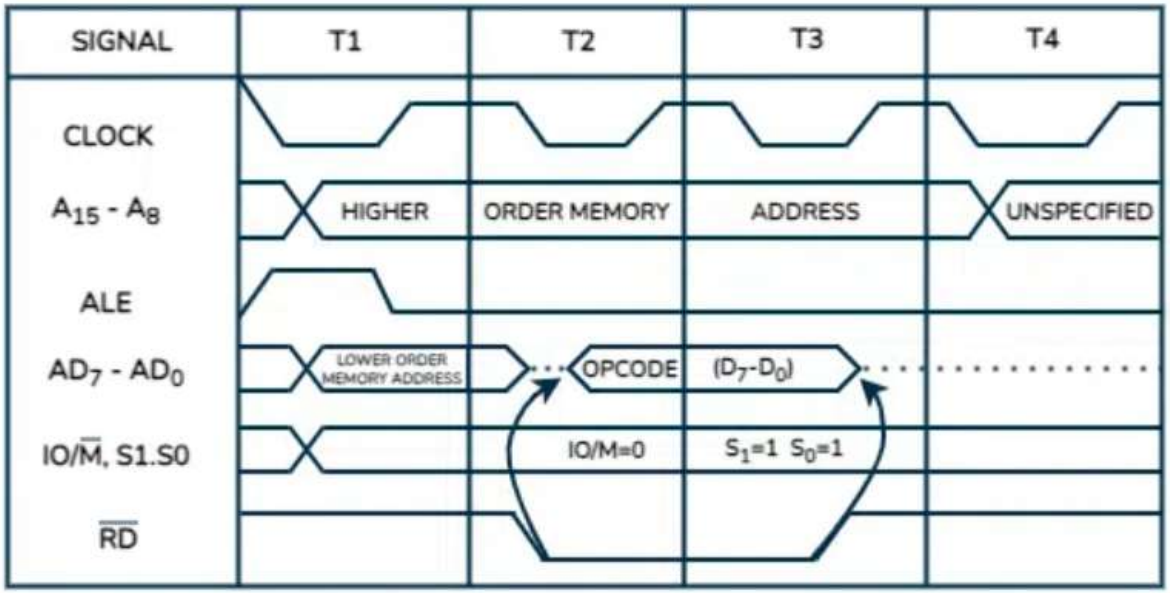
Timing and Control

Machine Cycle of 8085

- Opcode Fetch {4T- state}
- Memory Read {3T- state}
- Memory Write {3T- state}
- I/O Read {3T- state}
- I/O Write {3T- state}

Opcode Fetch Machine Cycle of 8085

Each processor instruction has a one byte opcode. Operation codes are stored in memory. Thus, the processor performs an opcode load machine cycle to load the opcode from memory.



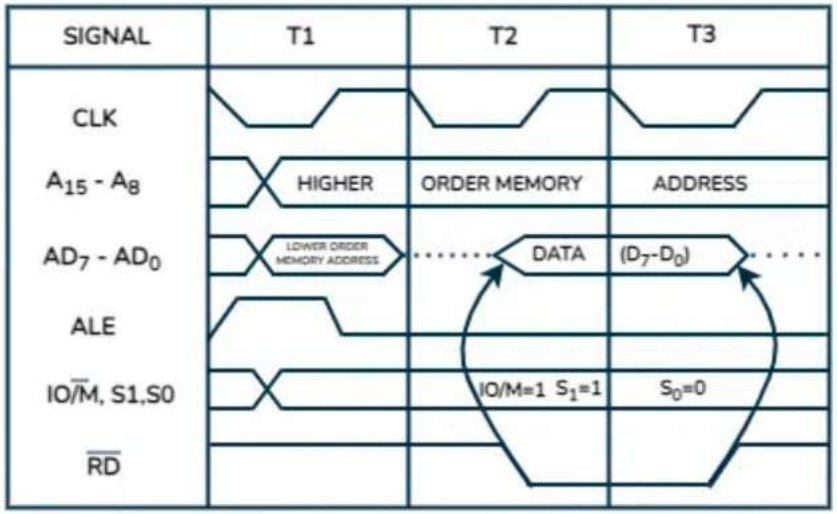
Timing and Control

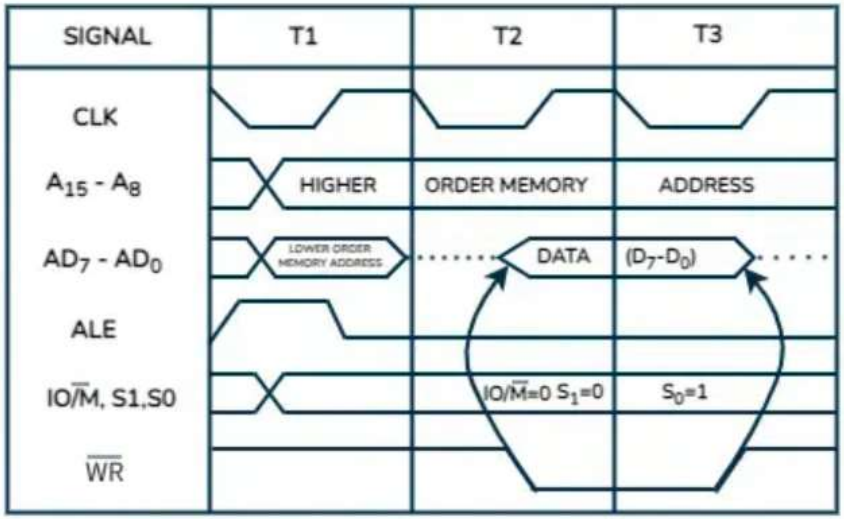
Memory Read Machine Cycle of 8085

- A memory read machine cycle is executed by the processor to read a data byte from memory.

Memory Write Machine Cycle of 8085

A write-to-memory machine cycle is executed by the processor to write a data byte to memory.

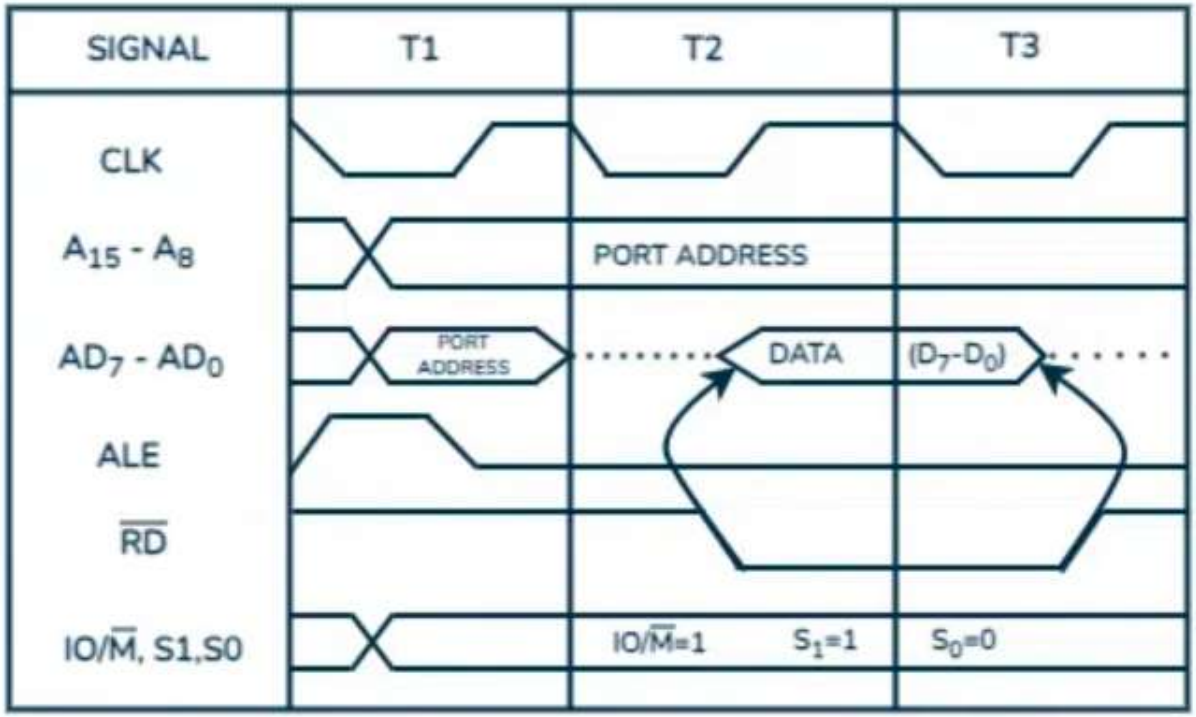




Timing and Control

I/O Read Machine Cycle

A reader I/O cycle is performed by the processor to read a data byte from an I/O port or peripheral that is I/O mapped in the system.

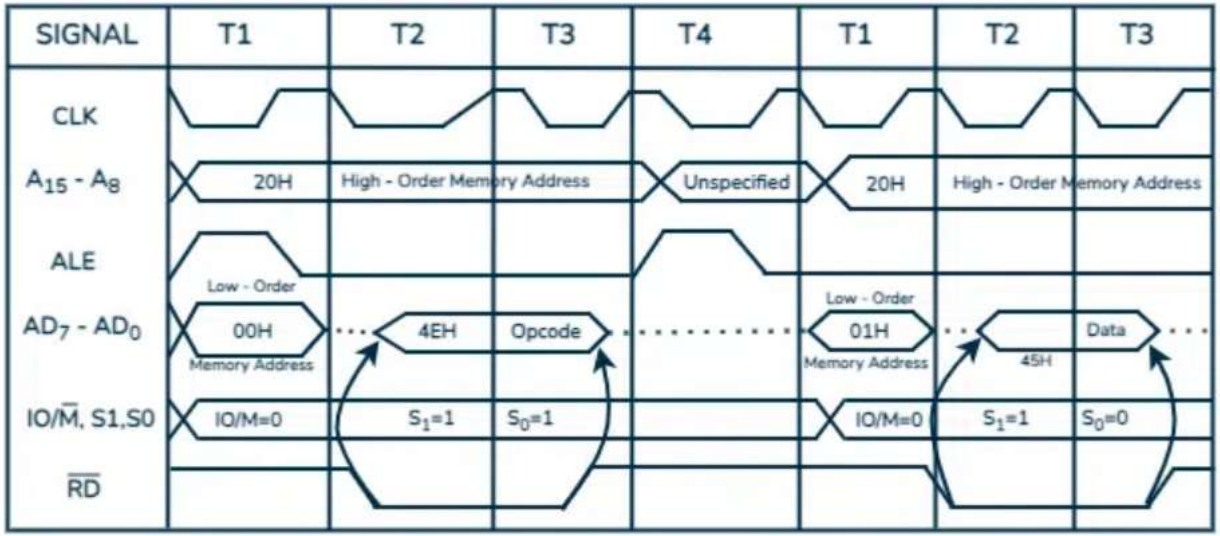


Timing and Control

Timing Diagram of MVI A, 45

- Loading opcode 4EH from memory 2000H. (Operation Code Loading Machine Cycle)
- Read (move) data 45H from memory 2001H. (memory reading).

Addresss	Mnemonics	Opcode
2000H	MVI A,45H	4EH
2001H		45H



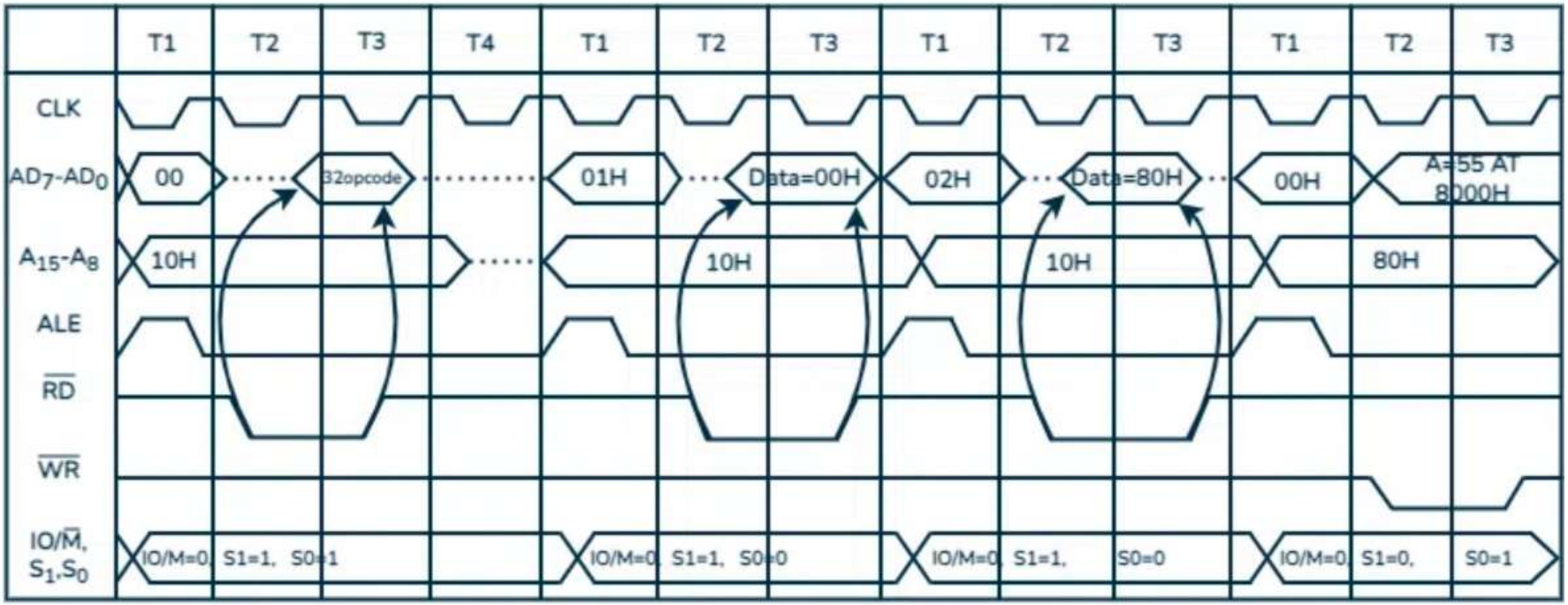
Timing and Control

Timing Diagram of STA 8000H

- STA stands for Store Accumulator - the contents of the accumulator are stored at (1000H).
- The transaction code of STA indicates 32H. It is provided by 1000H memory (see picture). - In the machine cycle.
- Then read the sub address (00). - Memory reading machine cycle
- Read higher memory location (80) – Memory read machine cycle.
- Consider the combination of the two addresses and write the details of the accumulator for 8000. Assume the memory address for the instruction and let the content of accumulator is C7H. So, C7H from accumulator is now stored in 8000H.

Address	Mnemonics	Opcode
1000H	STA 8000H	32H
1001H		00H
1002H		80H

Timing and Control



Memory-Reference Instructions

Memory-reference instructions are machine instructions that involve accessing memory locations.

Instruction	Meaning
AND	Logical AND between AC and memory, result in AC
ADD	Add content of memory to AC
LDA	Load content of memory into AC
STA	Store content of AC into memory
BUN	Branch unconditionally
BSA	Branch and Save Return Address
ISZ	Increment and Skip if Zero

Register Reference Instructions

Symbol	Meaning	Operation Performed
CLA	Clear AC	AC ← 0
CLE	Clear E	E ← 0
CMA	Complement AC	AC ← AC' (bitwise complement)
CME	Complement E	E ← E'
CIR	Circulate right (rotate AC + E right)	(AC, E) ← (AC, E) shifted right
CIL	Circulate left (rotate AC + E left)	(AC, E) ← (AC, E) shifted left
INC	Increment AC	AC ← AC + 1
SPA	Skip next instruction if AC is positive	If AC > 0, skip next instruction
SNA	Skip next instruction if AC is negative	If AC < 0, skip next instruction
SZA	Skip next instruction if AC is zero	If AC = 0, skip next instruction
SZE	Skip next instruction if E is zero	If E = 0, skip next instruction
HLT	Halt	Stop the computer

IO Reference Instructions

Symbol	Meaning	Operation Performed
INP	Input character to AC	AC(0–7) ← INPR, AC(8–15) ← 0
OUT	Output character from AC	OUTR ← AC(0–7)
SKI	Skip next instruction if input flag is set	If IF = 1, skip next instruction
SKO	Skip next instruction if output flag is set	If OF = 1, skip next instruction
ION	Turn interrupt system on	Enable interrupts (IEN ← 1)
IOF	Turn interrupt system off	Disable interrupts (IEN ← 0)

Interrupt

- Interrupts are signals that temporarily stop the CPU current execution and force it to jump to a special service routine (ISR — Interrupt Service Routine).

Based on Source

Type	Meaning
Hardware Interrupt	Interrupt comes from external hardware (e.g., TRAP, RST 7.5 in 8085)
Software Interrupt	Interrupt generated by executing a program instruction (e.g., RST n instructions in 8085)

Based on Masking Ability

Type	Meaning
Maskable Interrupt	Can be disabled or ignored by setting specific control bits (e.g., RST 7.5, RST 6.5, RST 5.5)
Non-Maskable Interrupt	Cannot be disabled; CPU must always respond (e.g., TRAP in 8085)

Interrupt

Based on Triggering Mechanism

Type	Meaning
Edge-Triggered Interrupt	Activated on a signal **transition** (rising or falling edge). Example: RST 7.5 in 8085.
Level-Triggered Interrupt	Activated when the interrupt signal is held at a certain logic level (high or low). Example: INTR, RST 6.5, RST 5.5 in 8085.

Based on Timing

Type	Meaning
Synchronous Interrupt	Occurs at predictable times related to program execution (e.g., divide-by-zero error).
Asynchronous Interrupt	Occurs at unpredictable times due to external events (e.g., a key press or hardware failure).

Interrupt

Based on Priority

Type	Meaning
Vectored Interrupt	Interrupt has a fixed service routine address (vector address is known). Example: TRAP, RST 7.5, etc.
Non-Vectored Interrupt	No fixed address; the external device must supply the address during interrupt acknowledge. Example: INTR in 8085.

Design of Accumulator Unit

Components of Accumulator Unit

Component	Function
Register (Accumulator Register)	Stores the current value.
ALU (Arithmetic Logic Unit)	Performs operations like ADD, SUB, AND, etc.
Input Multiplexer (MUX)	Chooses input for the accumulator: memory data, bus data, or ALU result.
Control Signals	Manage actions: Load, Clear, Hold.
Flags Register (optional)	Updates status flags: Zero (Z), Sign (S), Carry (C), etc.

Parul® University

NAAC GRADE A++

https://paruluniversity.ac.in/