

MICROPROCESSOR AND MICROCONTROLLER BASED SYSTEMS (FOR CSE & IT)

UNIT – I

INTRODUCTION TO 8085

8085 Microprocessor - Architecture and its operation, Concept of instruction execution and timing diagrams, fundamentals of memory interface - Addressing modes

Basic Concepts of Microprocessors

Differences between: Microcomputer, Microprocessor and Microcontroller

- Microcomputer is a computer with a microprocessor as its CPU. Includes memory, I/O etc.
- Microprocessor is a silicon chip which includes ALU, register circuits & control circuits
- Microcontroller is a silicon chip which includes microprocessor, memory & I/O in a single package.

What is micro?

- Micro is a new addition.
 - In the late 1960's, processors were built using discrete elements.

These devices performed the required operation, but were too large and too slow.

- It went directly from discrete elements to a single chip. However, comparing today's microprocessors to the ones built in the early 1970's you find an extreme increase in the amount of integration.

What is a microprocessor?

- The word comes from the combination of micro and processor.
- Processor means a device that processes whatever. In this context processor means a device that processes numbers, specifically binary numbers, 0's and 1's.
- To process means to manipulate. It is a general term that describes all manipulation. Again in this content, it means to perform certain operations on the numbers that depend on the microprocessor's design. It is a programmable device that takes in numbers, performs on them arithmetic or logical operations according to the program stored in memory and then produces other numbers

As a Programmable device:

- The microprocessor can perform different sets of operations on the data it receives depending on the sequence of instructions supplied in the given program.
- By changing the program, the microprocessor manipulates the data in different ways as Instructions, Words, Bytes, etc.
- They processed information 8-bits at a time. That's why they are called "8-bit processors". They can handle large numbers, but in order to process these numbers, they broke them into 8-bit pieces and processed each group of 8-bits separately.

What is memory?

- Memory is the location where information is kept while not in current use. It is stored in memory
- Memory is a collection of storage devices. Usually, each storage device holds one bit. Also, in most kinds of memory, these storage devices are grouped into

groups of 8. These 8 storage locations can only be accessed together. So, one can only read or write in terms of bytes to and from memory.

- Memory is usually measured by the number of bytes it can hold. It is measured in Kilos, Megas and lately Gigas. A Kilo in computer language is $2^{10} = 1024$. So, a KB (KiloByte) is 1024 bytes. Mega is 1024 Kilos and Giga is 1024 Mega.
- When a program is entered into a computer, it is stored in memory. Then as the microprocessor starts to execute the instructions, it brings the instructions from memory one at a time.
- Memory is also used to hold the data.
- The microprocessor reads (brings in) the data from memory when it needs it and writes (stores) the results into memory when it is done.

A MICROPROCESSOR-BASED SYSTEM

From the above description, we can draw the following block diagram to represent a microprocessor-based system as shown in fig 1

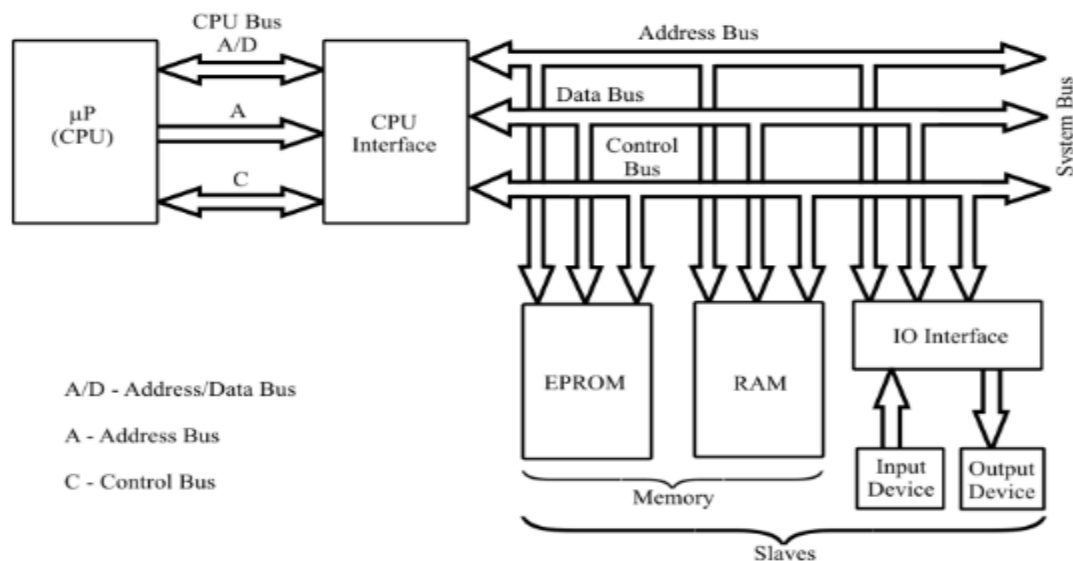


Fig.1. Microprocessor based system (organization of microcomputer)

In this system, the microprocessor is the master and all other peripherals are slaves. The master controls all peripherals and initiates all operations. The buses are group of lines that carry data, address or control signals. The CPU interface is provided to demultiplex the multiplexed lines, to generate the chip select signals and additional control signals. The system bus has separate lines for each signal.

All the slaves in the system are connected to the same system bus. At any time instant communication takes place between the master and one of the slaves. All the slaves have tristate logic and hence normally remain in high impedance state. The processor selects a slave by sending an address. When a slave is selected, it comes to the normal logic and communicates with the processor.

The EPROM memory is used to store permanent programs and data. The RAM memory is used to store temporary programs and data. The input device is used to enter program, data and to operate system. The output device is also used for examining the results. Since the speed of IO devices does not match with speed of microprocessor, an interface device is provided between system bus and IO device.

CENTRAL PROCESSING UNIT:

The CPU consists of ALU (Arithmetic and Logic Unit), Register unit and control unit. The CPU retrieves stored instructions and data word from memory; it also deposits processed data in memory.

a) ALU (Arithmetic and Logic Unit)

This section performs computing functions on data. These functions are arithmetic operations such as additions subtraction and logical operation such as AND, OR rotate etc. Result are stored either in registers or in memory or sent to output devices.

b) REGISTER UNIT:

It contains various register. The registers are used primarily to store data temporarily during the execution of a program. Some of the registers are accessible to the user through instructions.

c) CONTROL UNIT:

It provides necessary timing & control signals necessary to all the operations in the microcomputer. It controls the flow of data between the CPU and peripherals (input, output & memory). The control unit gets a clock which determines the speed of the CPU.

The CPU has three basic functions

- It fetches an instruction word stored in memory.
- It determines what the instruction is telling it to do.(decodes the instruction)
- It executes the instruction. Executing the instruction may include some of the following major tasks.
 - Transfer of data from reg. to reg. in the CPU itself.
 - Transfer of data between a CPU reg. & specified memory location.
 - Performing arithmetic and logical operations on data from a specific memory location or a designated CPU register.
 - Directing the CPU to change a sequence of fetching instruction, if processing the data created a specific condition.
 - Performing housekeeping function within the CPU itself in order to establish desired condition at certain registers.
 - It looks for control signal such as interrupts and provides appropriate responses.

- It provides states, control, and timing signals that the memory and input/output section can use.

There are three buses:

Address Bus:

It is a group of wires or lines that are used to transfer the addresses of Memory or I/O devices. It is unidirectional. In Intel 8085 microprocessor, Address bus was of 16 bits. This means that Microprocessor 8085 can transfer maximum 16 bit address which means it can address 65,536 different memory locations. This bus is multiplexed with 8 bit data bus. So the most significant bits (MSB) of address goes through Address bus (A7-A0) and LSB goes through multiplexed data bus (AD0-AD7).

Data Bus:

Data Bus is used to transfer data within Microprocessor and Memory/Input or Output devices. It is bidirectional as Microprocessor requires to send or receive data. The data bus also works as address bus when multiplexed with lower order address bus. Data bus is 8 Bits long. The word length of a processor depends on data bus, thats why Intel 8085 is called 8 bit Microprocessor because it have an 8 bit data bus.

Control Bus:

Microprocessor uses control bus to process data that is what to do with the selected memory location. Some control signals are Read, Write and Opcode fetch etc. Various operations are performed by microprocessor with the help of control bus. This is a dedicated bus, because all timing signals are generated according to control signal. The microprocessor is the master, which controls all the activities of the system. To perform a specific job or task, the microprocessor has to execute a program stored in memory. The program consists of a set of instructions stored in consecutive memory location. In order to execute the program the microprocessor issues address and control

signals, to fetch the instruction and data from memory one by one. After fetching each instruction it decodes the instruction and carries out the task specified by the instruction.

Memory

To execute a program:

- The user enters its instructions in binary format into the memory.
- The microprocessor then reads these instructions and whatever data is needed from memory, executes the instructions and places the results either in memory or produces it on an output device.

The three cycle instruction execution model

- To execute a program, the microprocessor “reads” each instruction from memory, “interprets” it, then “executes” it.
- To use the right names for the cycles
- The microprocessor fetches each instruction, decodes it, and then executes it. This sequence is continued until all instructions are performed.

The 8085 Machine Language

The 8085 (from Intel) is an 8-bit microprocessor. The 8085 uses a total of 246 bit patterns to form its instruction set. These 246 patterns represent only 74 instructions. The reason for the difference is that some (actually most) instructions have multiple different formats. Because it is very difficult to enter the bit patterns correctly, they are usually entered in hexadecimal instead of binary.

For example, the combination 0011 1100 which translates into “increment the number in the register called the accumulator”, is usually entered as 3C.

Assembly Language

Entering the instructions using hexadecimal is quite easier than entering the binary combinations. However, it still is difficult to understand what a program written in hexadecimal does. So, each company defines a symbolic code for the instructions.

These codes are called “mnemonics”.

The mnemonic for each instruction is usually a group of letters that suggest the operation performed.

- Using the same example from before,

00111100 translates to 3C in hexadecimal (OPCODE)

Its mnemonic is: “INR A”.

INR stands for “increment register” and A is short for accumulator.

It is important to remember that a machine language and its associated assembly language are completely machine dependent. In other words, they are not transferable from one microprocessor to a different one.

Assembling” The Program

How does assembly language get translated into machine language?

– There are two ways:

– 1st there is “hand assembly”.

- The programmer translates each assembly language instruction into its equivalent hexadecimal code (machine language). Then the hexadecimal code is entered into memory.
- The other possibility is a program called an “assembler”, which does the translation automatically.

8085 MICROPROCESSOR ARCHITECTURE

Features of 8085

- 8-bit general purpose μ p
- Capable of addressing 64 k of memory
- Has 40 pins as shown in fig 2
- Requires +5 v power supply
- Can operate with 3 MHz clock
- 8085 upward compatible

Pin Diagram of 8085

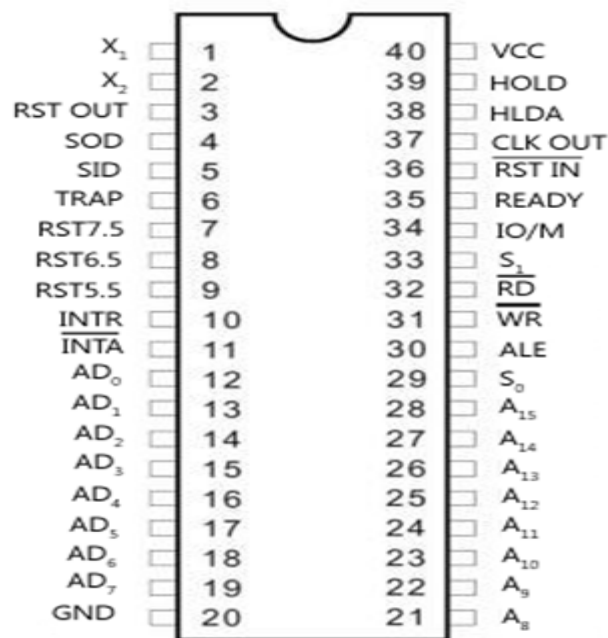


Fig .2 Pin Diagram of 8085

A8 - A15 (Output 3 State)

Address Bus: The most significant 8 bits of the memory address or the 8 bits of the I/O address, 3 states during Hold and Halt modes.

AD0 - AD7 (Input/Output 3state)

Multiplexed Address/Data Bus; Lower 8 bits of the memory address (or I/O address) appear on the bus during the first clock cycle of a machine state. It then becomes the data bus during the second and third clock cycles. 3 stated during Hold and Halt modes.

ALE (Output) Address Latch Enable

It occurs during the first clock cycle of a machine state and enables the address to get latched into the on chip latch of peripherals. The falling edge of ALE is set to guarantee setup and hold times for the address information. ALE can also be used to strobe the status information. ALE is never 3stated.

SO, S1 (Output)

Data Bus Status. Encoded status of the bus cycle: S1 S0 0 0 HALT 0 1 WRITE 1 0 READ 1 1 FETCH S1 can be used as an advanced R/W status.

RD (Output 3state)

READ: indicates the selected memory or I/O device is to be read and that the Data Bus is available for the data transfer.

WR (Output 3state)

WRITE: Indicates the data on the Data Bus is to be written into the selected memory or I/O location. Data is set up at the trailing edge of WR. 3 stated during Hold and Halt modes.

READY (Input)

If Ready is high during a read or write cycle, it indicates that the memory or peripheral is ready to send or receive data. If Ready is low, the CPU will wait for Ready to go high before completing the read or write cycle.

HOLD (Input)

It indicates that another Master is requesting the use of the Address and Data Buses. The CPU, upon receiving the Hold request will relinquish the use of buses as soon as the completion of the current machine cycle. Internal processing can continue. The processor can regain the buses only after the Hold is removed. When the Hold is acknowledged, the Address, Data, RD, WR, and IO/M lines are stated.

HLDA (Output)

HOLD ACKNOWLEDGE indicates that the CPU has received the Hold request and that it will relinquish the buses in the next clock cycle. HLDA goes low after the Hold request is removed. The CPU takes the buses one half clock cycle after HLDA goes low.

INTR (Input)

INTERRUPT REQUEST is used as a general purpose interrupt. It is sampled only during the next to the last clock cycle of the instruction. If it is active, the Program Counter (PC) will be inhibited from incrementing and an INTA will be issued. During this cycle a RESTART or CALL instruction can be inserted to jump to the interrupt service routine. The INTR is enabled and disabled by software. It is disabled by Reset and immediately after an interrupt is accepted.

INTA (Output)

INTERRUPT ACKNOWLEDGE: is used instead of (and has the same timing as) RD during the Instruction cycle after an INTR is accepted. It can be used to activate the 8259 Interrupt chip or some other interrupt port.

RESTART INTERRUPTS: These three inputs have the same timing as INTR except they cause an internal RESTART to be automatically inserted. RST 7.5 ~ Highest Priority RST 6.5 RST 5.5 Lowest Priority

TRAP (Input)

Trap interrupt is a nonmaskable restart interrupt. It is recognized at the same time as INTR. It is unaffected by any mask or Interrupt Enable. It has the highest priority of any interrupt.

RESET IN (Input)

Reset sets the Program Counter to zero and resets the Interrupt Enable and HLDA flipflops. None of the other flags or registers (except the instruction register) are affected. The CPU is held in the reset condition as long as Reset is applied.

RESET OUT (Output)

Indicates CPU is being reset also used as a system RESET. The signal is synchronized to the processor clock.

X1, X2 (Input)

Crystal or R/C network connections to set the internal clock generator X1 can also be an external clock input instead of a crystal. The input frequency is divided by 2 to give the internal operating frequency.

CLK (Output)

Clock Output for use as a system clock when a crystal or R/ C network is used as an input to the CPU. The period of CLK is twice the X1, X2 input period.

IO/M (Output)

IO/M indicates whether the Read/Write is to memory or I/O. Tristated during Hold and Halt modes.

SID (Input)

Serial input data line. The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.

SOD (output)

Serial output data line. The output SOD is set or reset as specified by the SIM instruction.

Vcc +5 volt supply.

Vss Ground Reference.

Signal Classification of 8085

The signal Classification of 8085 is as shown in fig3.

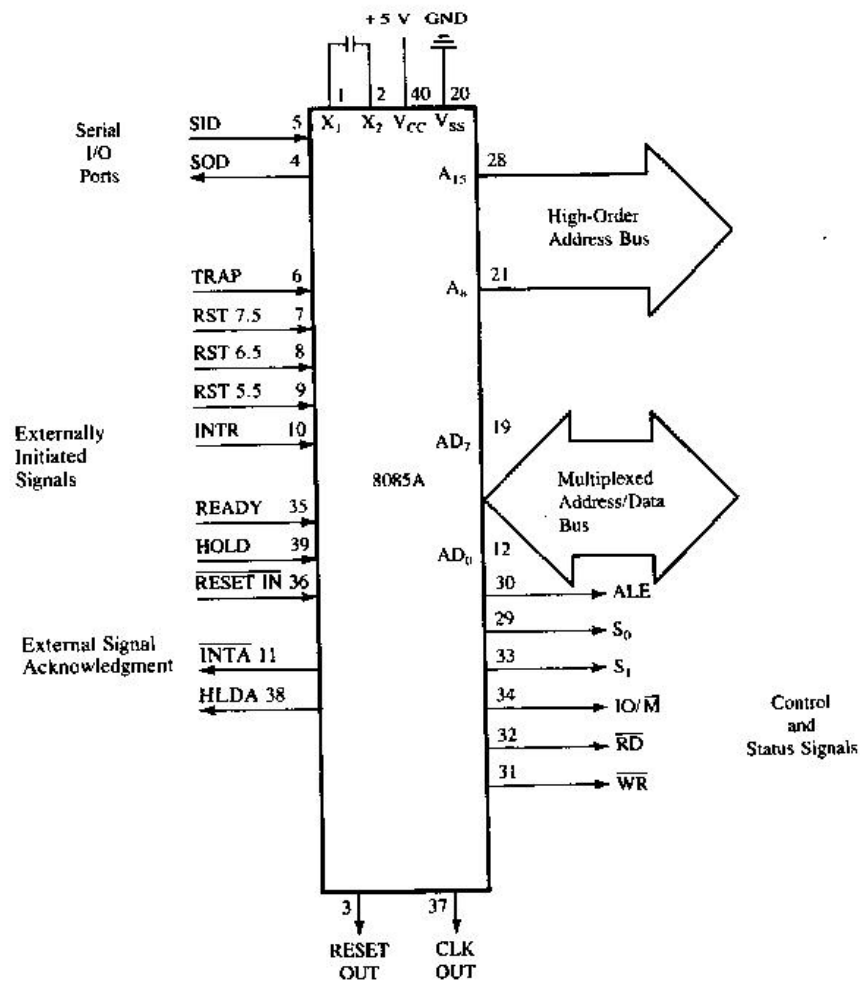


Fig: 3 Signal Classifications of 8085

System Bus – wires connecting memory & I/O to microprocessor

– ADDRESS BUS

- Unidirectional
- Identifying peripheral or memory location

– DATA BUS

- Bidirectional
- Transferring data

– CONTROL BUS

- Synchronization signals
- Timing signals
- Control signal

ARCHITECTURE OF INTEL 8085 MICROPROCESSOR

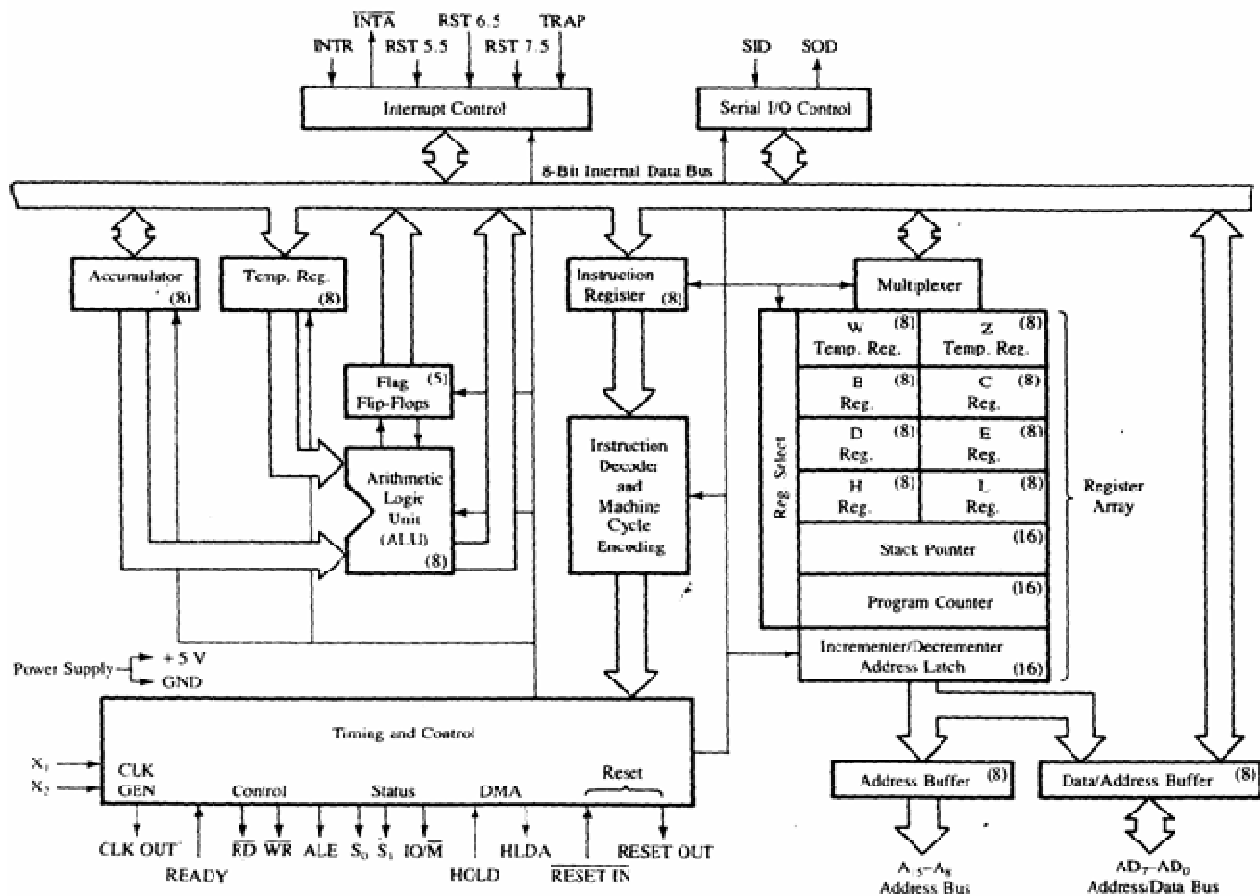


Fig:4 Architecture of intel 8085 microprocessor

The architecture of INTEL 8085 microprocessor is as shown in fig4.

Intel 8085 Microprocessor

Microprocessor consists of:

- Control unit: control microprocessor operations.
- ALU: performs data processing function.
- Registers: provide storage internal to CPU.
- Interrupts
- Internal data bus

The ALU

- In addition to the arithmetic & logic circuits, the ALU includes the accumulator, which is part of every arithmetic & logic operation.
- Also, the ALU includes a temporary register used for holding data temporarily during the execution of the operation. This temporary register is not accessible by the programmer.

Registers

General Purpose Registers

- B, C, D, E, H & L (8 bit registers)
- Can be used singly
- Or can be used as 16 bit register pairs

BC, DE, HL

- H & L can be used as a data pointer (holds memory address)
- Special Purpose Registers
- Accumulator (8 bit register)
- Store 8 bit data
- Store the result of an operation
- Store 8 bit data during I/O transfer Address

Flag Register

- 8 bit register – shows the status of the microprocessor before/after an operation
- S (sign flag), Z (zero flag), AC (auxillary carry flag), P (parity flag) & CY (carry flag)

D7	D6	D5	D4	D3	D2	D1	D0
S	Z	X	AC	X	P	X	CY

Sign Flag

- Used for indicating the sign of the data in the accumulator
- The sign flag is set if negative (1 – negative)
- The sign flag is reset if positive (0 –positive)

Zero Flag

- Is set if result obtained after an operation is 0
- Is set following an increment or decrement operation of that register

Carry Flag

```

          10110011
+         01001101
-----
          00000000
  
```

– Is set if there is a carry or borrow from arithmetic operation

```

          1011 0101      1011 0101
+         0110 1100  -   1100 1100
-----      -----

Carry 1      0010 0001  Borrow 1  1110 1001
  
```

Auxillary Carry Flag

– Is set if there is a carry out of bit 3

Parity Flag

- Is set if parity is even
- Is cleared if parity is odd

The Internal Architecture

We have already discussed the general purpose registers, the Accumulator, and the flags.

The Program Counter (PC)

- This is a register that is used to control the sequencing of the execution of instructions.
- This register always holds the address of the next instruction.
- Since it holds an address, it must be 16 bits wide.

The Stack pointer

- The stack pointer is also a 16-bit register that is used to point into memory.
- The memory this register points to is a special area called the stack.
- The stack is an area of memory used to hold data that will be retrieved soon.
- The stack is usually accessed in a Last In First Out (LIFO) fashion.

Non Programmable Registers

Instruction Register & Decoder

- Instruction is stored in IR after fetched by processor
- Decoder decodes instruction in IR

Internal Clock generator

- 3.125 MHz internally
- 6.25 MHz externally

The Address and Data Busses

- The address bus has 8 signal lines A8 – A15 which are unidirectional.
- The other 8 address bits are multiplexed (time shared) with the 8 data bits.

So, the bits AD0 – AD7 are bi-directional and serve as A0 – A7 and D0 – D7 at the same time.

- During the execution of the instruction, these lines carry the address bits during the early part, then during the late parts of the execution, they carry the 8 data bits.

In order to separate the address from the data, we can use a latch to save the value before the function of the bits changes.

Demultiplexing AD7-AD0

- From the above description, it becomes obvious that the AD7– AD0 lines are serving a dual purpose and that they need to be demultiplexed to get all the information.
- The high order bits of the address remain on the bus for three clock periods. However, the low order bits remain for only one clock period and they would be lost if they are not saved externally. Also, notice that the low order bits of the address disappear when they are needed most.
- To make sure we have the entire address for the full three clock cycles, we will use an external latch to save the value of AD7– AD0 when it is carrying the address bits. We use the ALE signal to enable this latch.

Demultiplexing AD7-AD0

Given that ALE operates as a pulse during T1, we will be able to latch the address. Then when ALE goes low, the address is saved and the AD7– AD0 lines can be used for their purpose as the bi-directional data lines.

Demultiplexing the Bus AD7 – AD0

- The high order address is placed on the address bus and hold for 3 clk periods,
- The low order address is lost after the first clk period, this address needs to be hold however we need to use latch
- The address AD7 – AD0 is connected as inputs to the latch 74LS373.
- The ALE signal is connected to the enable (G) pin of the latch and the OC – Output control – of the latch is grounded

ADDRESSING MODES

The microprocessor has different ways of specifying the data for the instruction. These are called “addressing modes”.

The 8085 has four addressing modes:

- | | |
|-------------|-----------|
| – Implied | CMA |
| – Immediate | MVI B, 45 |
| – Direct | LDA 4000 |
| – Indirect | LDAX B |

Load the accumulator with the contents of the memory location whose address is stored in the register pair BC).

Many instructions require two operands for execution. For example transfer of data between two registers. The method of identifying the operands position by the instruction format is known as the addressing mode. When two operands are involved in an instruction, the first operand is assumed to be in a register Mp itself.

Types of Addressing Modes

- Register addressing
- Direct addressing mode

- Register indirect addressing
- Immediate Addressing mode
- Implied addressing mode

Register Addressing

This type of addressing mode specifies register or register pair that contains data. ie (only the register need be specified as the address of the operands).

Example MOV B, A (the content of A is copied into the register B)

Direct Addressing Mode

Data is directly copied from the given address to the register.

Example LDA 3000H (The content at the location 3000H is copied to the register A).

Register Indirect Addressing

In this mode, the address of operand is specified by a register pair

Example MOV A, M (Move data from memory location specified by H-L pair to accumulator)

Immediate Addressing Mode

In this mode, the operand is specified within the instruction itself.

Example MVI A, 05 H (Move 05 H in accumulator.)

Implied Addressing Mode

This mode doesn't require any operand. The data is specified by opcode itself.

Example RAL,

CMP

INSTRUCTION SET OF 8085

An instruction is a binary pattern designed inside a microprocessor to perform a specific function. The entire group of instructions that a microprocessor supports is called Instruction Set. Since the 8085 is an 8-bit device it can have up to 28 (256) instructions. However, the 8085 only uses 246 combinations that represent a total of 74 instructions. Each instruction has two parts. The first part is the task or operation to be performed. This part is called the “opcode” (operation code). The second part is the data to be operated on. This part is called the “operand”.

Instruction Size

- Depending on the operand type, the instruction may have different sizes. It will occupy a different number of memory bytes.
 - Typically, all instructions occupy one byte only.
 - The exception is any instruction that contains immediate data or a memory address.
- Instructions that include immediate data use two bytes.
 - One for the opcode and the other for the 8-bit data.
- Instructions that include a memory address occupy three bytes.
 - One for the opcode, and the other two for the 16-bit address.

Classification of Instruction Set

- Data Transfer Instruction
- Arithmetic Instructions
- Logical Instructions
- Branching Instructions
- Machine Control Instructions

DATA TRANSFER INSTRUCTIONS

Opcode	Operand	Description
--------	---------	-------------

MOV	Rd, Rs M, Rs Rd, M	Copy from source to destination.
-----	--------------------------	----------------------------------

This instruction copies the contents of the source register into the destination register. The contents of the source register are not altered. If one of the operands is a memory location, its location is specified by the contents of the HL registers. Example: MOV B, C or MOV B, M.

Opcode	Operand	Description
MVI	Rd, Data M, Data	Move immediate 8-bit

The 8-bit data is stored in the destination register or memory. If the operand is a memory location, its location is specified by the contents of the H-L registers. Example: MVI B, 57H or MVI M, 57H.

Opcode	Operand	Description
LDA	16-bit address	Load Accumulator

The contents of a memory location, specified by a 16- bit address in the operand, are copied to the accumulator. The contents of the source are not altered. Example: LDA 2034H

Opcode	Operand	Description
--------	---------	-------------

LDAX	B/D Register Pair	Load accumulator indirect

The contents of the designated register pair point to a memory location. This instruction copies the contents of that memory location into the accumulator. The contents of either the register pair or the memory location are not altered. Example: LDAX B

Opcode	Operand	Description
STA	16-bit address	Store accumulator direct

The contents of accumulator are copied into the memory location specified by the operand. Example: STA 2500 H

Opcode	Operand	Description
STAX	Reg. pair	Store accumulator indirect

The contents of accumulator are copied into the memory location specified by the contents of the register pair. Example: STAX B

Opcode	Operand	Description
SHLD	16-bit address	Store H-L registers direct

The contents of register L are stored into memory location specified by the 16-bit address. The contents of register H are stored into the next memory location.

Example: SHLD 2550 H

Opcode	Operand	Description
XCHG	None	Exchange H-L with D-E

The contents of register H are exchanged with the contents of register D. The contents of register L are exchanged with the contents of register E. Example: XCHG

Opcode	Operand	Description
SPHL	None	Copy H-L pair to the Stack Pointer (SP)

This instruction loads the contents of H-L pair into SP. Example: SPHL

Opcode	Operand	Description
XTHL	None	Exchange H-L with top of stack

The contents of L register are exchanged with the location pointed out by the contents of the SP. The contents of H register are exchanged with the next location (SP + 1).

Example: XTHL

Opcode	Operand	Description
PCHL	None	Load program counter with

		H-L contents
--	--	--------------

The contents of registers H and L are copied into the program counter (PC). The contents of H are placed as the high-order byte and the contents of L as the low-order byte. Example: PCHL

Opcode	Operand	Description
PUSH	Reg. pair	Push register pair onto stack

The contents of register pair are copied onto stack. SP is decremented and the contents of high-order registers (B, D, H, A) are copied into stack. SP is again decremented and the contents of low-order registers (C, E, L, Flags) are copied into stack. Example: PUSH B

Opcode	Operand	Description
POP	Reg. pair	Pop stack to register pair

The contents of top of stack are copied into register pair. The contents of location pointed out by SP are copied to the low-order register (C, E, L, Flags). SP is incremented and the contents of location are copied to the high-order register (B, D, H, A). Example: POP H

Opcode	Operand	Description
OUT	8-bit port address	Copy data from

		accumulator to a port with 8- bit address
--	--	---

The contents of accumulator are copied into the I/O port. Example: OUT 78 H

Opcode	Operand	Description
IN	8-bit port address	Copy data to accumulator from a port with 8- bit address

The contents of I/O port are copied into accumulator. Example: IN 8C H

1.ARITHMETIC INSTRUCTIONS

These instructions perform arithmetic operations such as addition, subtraction, increment, and decrement.

Addition

Any 8-bit number, or the contents of register, or the contents of memory location can be added to the contents of accumulator. The result (sum) is stored in the accumulator. No two other 8-bit registers can be added directly. Example: The contents of register B cannot be added directly to the contents of register C.

Opcode	Operand	Description
ADD	R M	Add register or memory to accumulator

The contents of register or memory are added to the contents of accumulator. The result is stored in accumulator. If the operand is memory location, its address is specified by H-L pair. All flags are modified to reflect the result of the addition. Example: ADD B or ADD M

Opcode	Operand	Description
ADC	R M	Add register or memory to accumulator with carry

The contents of register or memory and Carry Flag (CY) are added to the contents of accumulator. The result is stored in accumulator. If the operand is memory location, its address is specified by H-L pair. All flags are modified to reflect the result of the addition. Example: ADC B or ADC M

Opcode	Operand	Description
ADI	8-bit data	Add immediate to accumulator

The 8-bit data is added to the contents of accumulator. The result is stored in accumulator. All flags are modified to reflect the result of the addition. Example: ADI 45 H

Opcode	Operand	Description
ACI	8-bit data	Add immediate to accumulator with carry

The 8-bit data and the Carry Flag (CY) are added to the contents of accumulator. The result is stored in accumulator. All flags are modified to reflect the result of the addition. Example: ACI 45 H

Opcode	Operand	Description
DAD	Reg. pair	Add register pair to H-L pair

The 16-bit contents of the register pair are added to the contents of H-L pair. The result is stored in H-L pair. If the result is larger than 16 bits, then CY is set. No other flags are changed. Example: DAD B

Subtraction

Any 8-bit number, or the contents of register, or the contents of memory location can be subtracted from the contents of accumulator. The result is stored in the accumulator. Subtraction is performed in 2's complement form. If the result is negative, it is stored in 2's complement form. No two other 8-bit registers can be subtracted directly.

Opcode	Operand	Description
SUB	R M	Subtract register or memory from accumulator

The contents of the register or memory location are subtracted from the contents of the accumulator. The result is stored in accumulator. If the operand is memory location, its address is specified by H-L pair. All flags are modified to reflect the result of subtraction. Example: SUB B or SUB M

Opcode	Operand	Description
--------	---------	-------------

SBB	R M	Subtract register or memory from accumulator with borrow

The contents of the register or memory location and Borrow Flag (i.e. CY) are subtracted from the contents of the accumulator. The result is stored in accumulator. If the operand is memory location, its address is specified by H-L pair. All flags are modified to reflect the result of subtraction. Example: SBB B or SBB M

Opcode	Operand	Description
SUI	8-bit data	Subtract immediate from accumulator

The 8-bit data is subtracted from the contents of the accumulator. The result is stored in accumulator. All flags are modified to reflect the result of subtraction. Example: SUI 45 H

Opcode	Operand	Description
SBI	8-bit data	Subtract immediate from accumulator with borrow

The 8-bit data and the Borrow Flag (i.e. CY) is subtracted from the contents of the accumulator. The result is stored in accumulator. All flags are modified to reflect the result of subtraction. Example: SBI 45 H

Increment/Decrement

The 8-bit contents of a register or a memory location can be incremented or decremented by 1. The 16-bit contents of a register pair can be incremented or decremented by 1. Increment or decrement can be performed on any register or a memory location.

Opcode	Operand	Description
INR	R M	Increment register or memory by 1

The contents of register or memory location are incremented by 1. The result is stored in the same place. If the operand is a memory location, its address is specified by the contents of H-L pair. Example: INR B or INR M

Opcode	Operand	Description
INX	R	Increment register pair by 1

The contents of register pair are incremented by 1. The result is stored in the same place. Example: INX H

Opcode	Operand	Description
DCR	R M	Decrement register or memory by 1

The contents of register or memory location are decremented by 1. The result is stored in the same place. If the operand is a memory location, its address is specified by the contents of H-L pair. Example: DCR B or DCR M

Opcode	Operand	Description
DCX	R	Decrement register pair by 1

The contents of register pair are decremented by 1. The result is stored in the same place. Example: DCX H

2.LOGICAL INSTRUCTIONS

These instructions perform logical operations on data stored in registers, memory and status flags. The logical operations are:

- AND
- OR
- XOR
- Rotate
- Compare
- Complement

AND, OR, XOR

Any 8-bit data, or the contents of register, or memory location can logically have

- AND operation
- OR operation
- XOR operation

with the contents of accumulator. The result is stored in accumulator.

Opcode	Operand	Description
ANA	R	Logical AND register or

	M	memory with accumulator
--	---	-------------------------

The contents of the accumulator are logically ANDed with the contents of register or memory. The result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of H-L pair. S, Z, P are modified to reflect the result of the operation. CY is reset and AC is set. Example: ANA B or ANA M.

Opcode	Operand	Description
ANI	8-bit data	Logical AND immediate with accumulator

The contents of the accumulator are logically ANDed with the 8-bit data. The result is placed in the accumulator. S, Z, P are modified to reflect the result. CY is reset, AC is set. Example: ANI 86H.

Opcode	Operand	Description
ORA	R M	Logical OR register or memory with accumulator

The contents of the accumulator are logically ORed with the contents of the register or memory. The result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of H-L pair. S, Z, P are modified to reflect the result. CY and AC are reset. Example: ORA B or ORA M.

Opcode	Operand	Description
ORI	8-bit data	Logical OR immediate with

		accumulator
--	--	-------------

The contents of the accumulator are logically ORed with the 8-bit data. The result is placed in the accumulator. S, Z, P are modified to reflect the result. CY and AC are reset. Example: ORI 86H.

Opcode	Operand	Description
XRA	R M	Logical XOR register or memory with accumulator

The contents of the accumulator are XORed with the contents of the register or memory. The result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of H-L pair. S, Z, P are modified to reflect the result of the operation. CY and AC are reset. Example: XRA B or XRA M.

Opcode	Operand	Description
XRI	8-bit data	XOR immediate with accumulator

The contents of the accumulator are XORed with the 8-bit data. The result is placed in the accumulator. S, Z, P are modified to reflect the result. CY and AC are reset. Example: XRI 86H.

Rotate

Each bit in the accumulator can be shifted either left or right to the next position as shown in fig5.

Opcode	Operand	Description
RLC	None	Rotate accumulator left

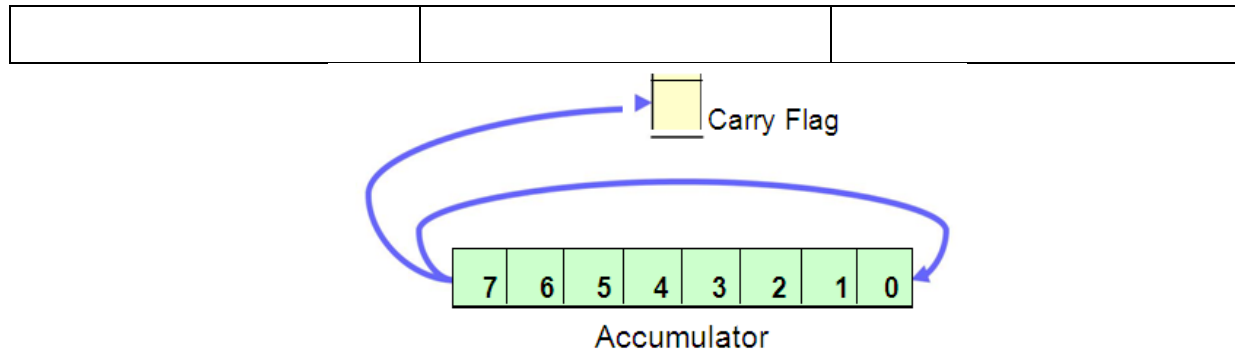


Fig 5. : Work flow of RLC

Each binary bit of the accumulator is rotated left by one position. Bit D7 is placed in the position of D0 as well as in the Carry flag. CY is modified according to bit D7. S, Z, P, AC are not affected. Example: RLC.

Opcode	Operand	Description
RRC	None	Rotate accumulator right

Each binary bit of the accumulator is rotated right by one position. Bit D0 is placed in the position of D7 as well as in the Carry flag. CY is modified according to bit D0. S, Z, P, AC are not affected. Example: RRC.

Opcode	Operand	Description
RAL	None	Rotate accumulator left through carry

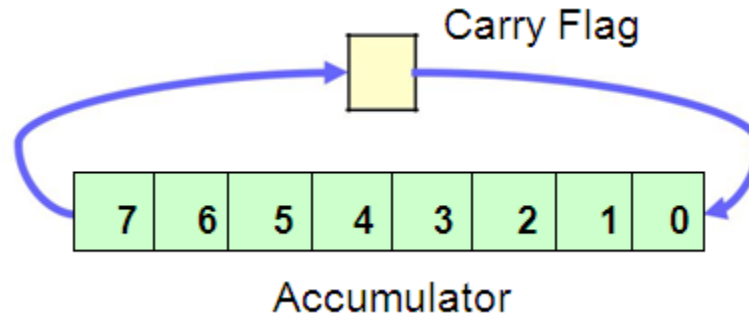


Fig 6. : Work flow of RAL

Each binary bit of the accumulator is rotated left by one position through the Carry flag as shown in fig 6. Bit D7 is placed in the Carry flag, and the Carry flag is placed in the least significant position D0. CY is modified according to bit D7. S, Z, P, AC are not affected. Example: RAL.

Opcode	Operand	Description
RAR	None	Rotate accumulator right through carry

Each binary bit of the accumulator is rotated right by one position through the Carry flag. Bit D0 is placed in the Carry flag, and the Carry flag is placed in the most significant position D7. CY is modified according to bit D0. S, Z, P, AC are not affected. Example: RAR.

COMPARE

Any 8-bit data, or the contents of register, or memory location can be compares for:

- Equality
- Greater Than
- Less Than

with the contents of accumulator. The result is reflected in status flags.

Opcode	Operand	Description
CMP	R M	Compare register or memory with accumulator

The contents of the operand (register or memory) are compared with the contents of the accumulator. Both contents are preserved. The result of the comparison is shown by setting the flags of the PSW as follows:

- if (A) < (reg/mem): carry flag is set
- if (A) = (reg/mem): zero flag is set
- if (A) > (reg/mem): carry and zero flags are reset.

Example: CMP B or CMP M

Opcode	Operand	Description
CPI	8-bit data	Compare immediate with accumulator

The 8-bit data is compared with the contents of accumulator. The values being compared remain unchanged. The result of the comparison is shown by setting the flags of the PSW as follows:

- if (A) < data: carry flag is set
- if (A) = data: zero flag is set
- if (A) > data: carry and zero flags are reset

Example: CPI 89H

COMPLEMENT

The contents of accumulator can be complemented. Each 0 is replaced by 1 and each 1 is replaced by 0.

Opcode	Operand	Description
CMA	None	Complement accumulator

The contents of the accumulator are complemented. No flags are affected. Example: CMA.

Opcode	Operand	Description
CMC	None	Complement carry

The Carry flag is complemented. No other flags are affected. Example: CMC.

Opcode	Operand	Description
STC	None	Set carry

The Carry flag is set to 1. No other flags are affected. Example: STC.

3. BRANCHING INSTRUCTIONS

The branching instruction alters the normal sequential flow. These instructions alter either unconditionally or conditionally.

Branch operations are of two types:

Unconditional branch-- Go to a new location no matter what.

Conditional branch-- Go to a new location if the condition is true.

Opcode	Operand	Description
JMP	16-bit address	Jump unconditionally

The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.

Example: JMP 2034 H.

Opcode	Operand	Description
Jx	16-bit address	Jump conditionally

The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW. Replace x with condition

Example: JZ 2034 H.

Jump conditionally

Opcode	Description	Status flag
JC	Jump if Carry	CY = 1
JNC	Jump if no carry	CY=0
JP	Jump if positive	S=0
JM	Jump if minus	S=1
JZ	Jump if Zero	Z=1
JNZ	Jump if no zero	Z=0
JPE	Jump if parity even	P=1
JPO	Jump if parity odd	P=0

Opcode	Operand	Description
CALL	16-bit address	Call unconditionally

The program sequence is transferred to the memory location specified by the 16-bit address given in the operand. Before the transfer, the address of the next instruction after CALL (the contents of the program counter) is pushed onto the stack.

Example: CALL 2034 H.

Opcode	Operand	Description
Cx	16-bit address	Call conditionally

The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW. Before the transfer, the address of the next instruction after the call (the contents of the program counter) is pushed onto the stack. Replace x with condition

Example: CZ 2034 H.

Call Conditionally

Opcode	Description	Status flag
CC	Call if carry	CY=1
CNC	Call if no carry	CY=0
CP	Call if positive	S=0
CM	Call if minus	S=1
CZ	Call if Zero	Z=1
CNZ	Call if no zero	Z=0
CPE	Call if parity even	P=1
CPO	Call if parity odd	P=0

Opcode	Operand	Description
RET	None	Return unconditionally

The program sequence is transferred from the subroutine to the calling program. The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.

Example: RET.

Opcode	Operand	Description
Rx	None	Call conditionally

The program sequence is transferred from the subroutine to the calling program based on the specified flag of the PSW. The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address. **Example:**

RZ. Replace x with condition

RETURN CONDITIONALLY

Opcode	Description	Status flag
RC	Return if carry	CY=1
RNC	Return if no carry	CY=0
RP	Return Call if positive	S=0
RM	Return if minus	S=1
RZ	Return if Zero	Z=1
RNZ	Return if no zero	Z=0
RPE	Return if parity even	P=1
RPO	Return if parity odd	P=0

Opcode	Operand	Description
RST	0-7	Restart (Software Interrupts)

The RST instruction jumps the control to one of eight memory locations depending upon the number. These are used as software instructions in a program to transfer program execution to one of the eight locations. Example: RST 3.

RESTART Address table

Instructions	Restart address
RST 0	0000 H
RST 1	0008 H
RST 2	0010 H
RST 3	0018 H
RST 4	0020 H
RST 5	0028 H
RST 6	0030 H
RST 7	0038 H

4.MACHINE CONTROL INSTRUCTIONS

The control instructions control the operation of microprocessor.

Opcode	Operand	Description
NOP	None	No operation

No operation is performed. The instruction is fetched and decoded but no operation is executed. Usually used for delay or to replace instructions during debugging.

Example: NOP

Opcode	Operand	Description
HLT	None	Halt

The CPU finishes executing the current instruction and halts any further execution. An interrupt or reset is necessary to exit from the halt state. **Example:** HLT

Opcode	Operand	Description
DI	None	Disable interrupt

The interrupt enable flip-flop is reset and all the interrupts except the TRAP are disabled.

No flags are affected.

Example: DI

Opcode	Operand	Description
EI	None	Enable interrupt

The interrupt enable flip-flop is set and all interrupts are enabled. No flags are affected. This instruction is necessary to re-enable the interrupts (except TRAP).

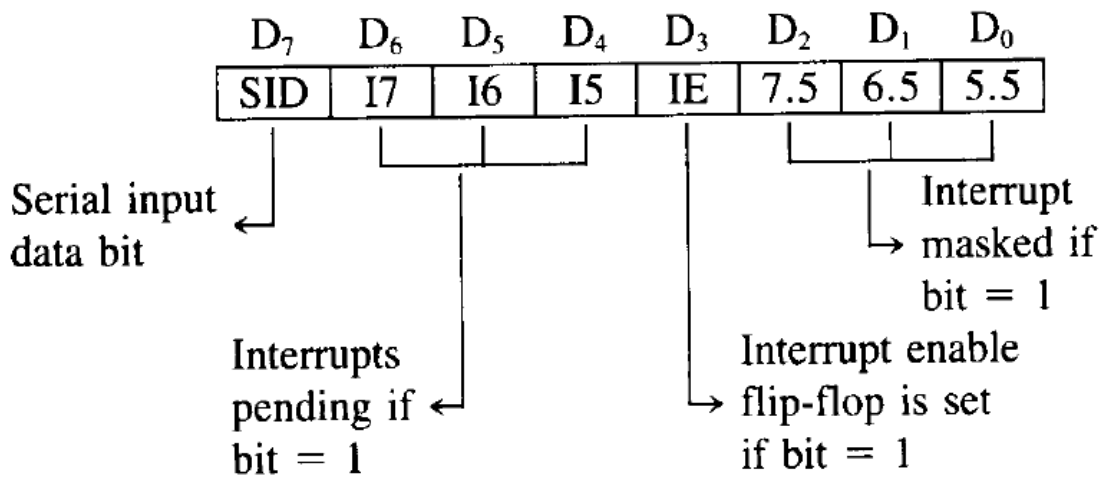
Example: EI

Opcode	Operand	Description
RIM	None	Read Interrupt Mask

This is a multipurpose instruction used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data input bit. The instruction loads eight bits in the accumulator with the following interpretations.

Example: RIM

RIM Instruction

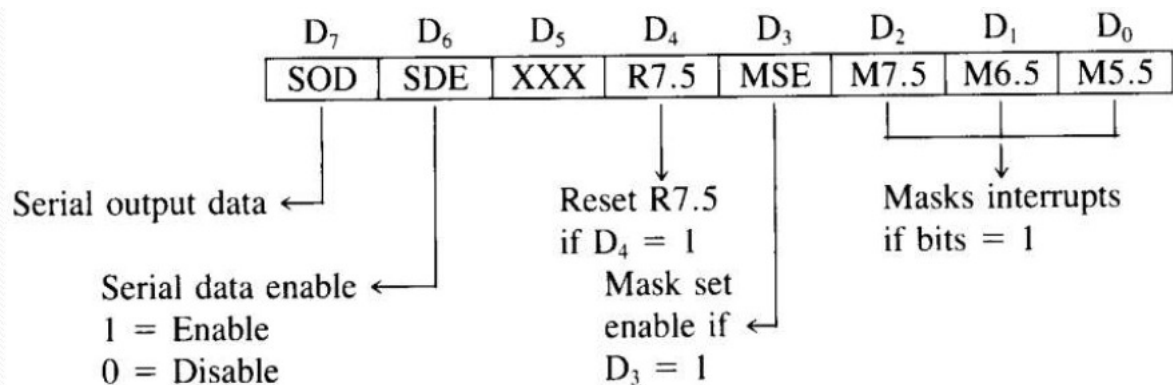


Opcode	Operand	Description
SIM	None	Set Interrupt Mask

This is a multipurpose instruction and used to implement the 8085 interrupts 7.5, 6.5, 5.5, and serial data output. The instruction interprets the accumulator contents as follows.

Example: SIM

SIM Instruction



TIMING DIAGRAM

Timing diagram is the display of initiation of read/write and transfer of data operations under the control of 3-status signals IO / M, S₁, and S₀. All actions in the microprocessor are controlled by either leading or trailing edge of the clock.

Machine Cycle

It is the time required by the microprocessor to complete the operation of accessing the memory devices or I/O devices. In machine cycle various operations like opcode fetch, memory read, memory write, I/O read, I/O write are performed.

T-State

Each clock cycle is called as T-states.

Each machine cycle is composed of many clock cycles. Since, the data and instructions, both are stored in the memory, the μ P performs fetch operation to read the instruction or data and then execute the instruction. The 3-status signals: IO / M, S₁, and S₀ are generated at the beginning of each machine cycle. The unique combination of these 3-status signals identify read or write operation and remain valid for the duration of the cycle.

Table 1 Machine Cycle Status And Control Signals

Machine cycle	Status			Controls		
	IO / $\overline{M}$	S ₁	S ₀	$\overline{RD}$	$\overline{WR}$	$\overline{INTA}$
Opcode Fetch (OF)	0	1	1	0	1	1
Memory Read	0	1	0	0	1	1
Memory Write	0	0	1	1	0	1
I/O Read (I/OR)	1	1	0	0	1	1
I/O Write (I/OW)	1	0	1	1	0	1
Acknowledge of INTR (INTA)	1	1	1	1	1	0
BUS Idle (BI) : DAD	0	1	0	1	1	1
ACK of RST, TRAP	1	1	1	1	1	1
HALT	Z	0	0	Z	Z	1
HOLD	Z	X	X	Z	Z	1

X $\Rightarrow$ Unspecified, and Z $\Rightarrow$ High impedance state

Table1 shows details of the unique combination of these status signals to identify different machine cycles. Thus, time taken by any μP to execute one instruction is calculated in terms of the clock period. The execution of instruction always requires read and writes operations to transfer data to or from the μP and memory or I/O devices. Each read/ write operation constitutes one machine cycle (MC1) as indicated in Fig.7. Each machine cycle consists of many clock periods/ cycles, called T-states.

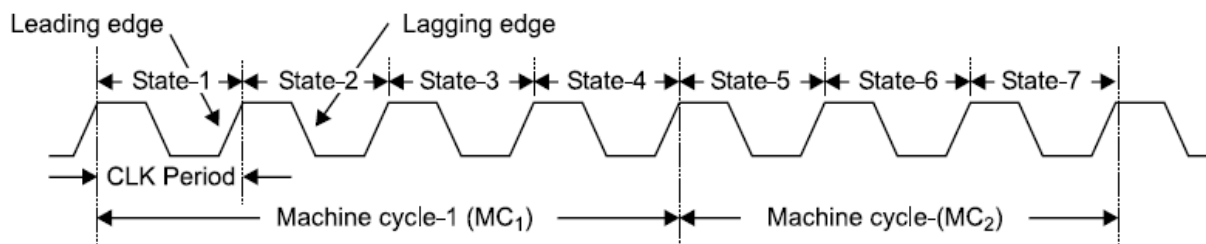


Fig.7 Machine cycle showing clock periods

Processor Cycle:

The functions of the microprocessor are divided into fetch and execute cycle of any instruction of a program. The program is nothing but number of instructions stored in the memory in sequence. In the normal process of operation, the microprocessor fetches (receives or reads) and executes one instruction at a time in the sequence until it executes the halt (HLT) instruction.

Instruction Cycle

An instruction cycle is defined as the time required to fetch and execute an instruction. For executing any program, basically 2-steps are followed sequentially with the help of clocks

- Fetch, and
- Execute.

The time taken by the μP in performing the fetch and execute operations are called fetch and execute cycle. Thus, sum of the fetch and execute cycle is called the instruction cycle as indicated in Fig. 8. Each read or writes operation constitutes a machine cycle. The instructions of 8085 require 1–5 machine cycles containing 3–6 states (clocks). The 1st machine cycle of any instruction is always an Op Code fetch cycle in which the processor decides the nature of instruction. It is of at least 4-states. It may go up to 6-states.

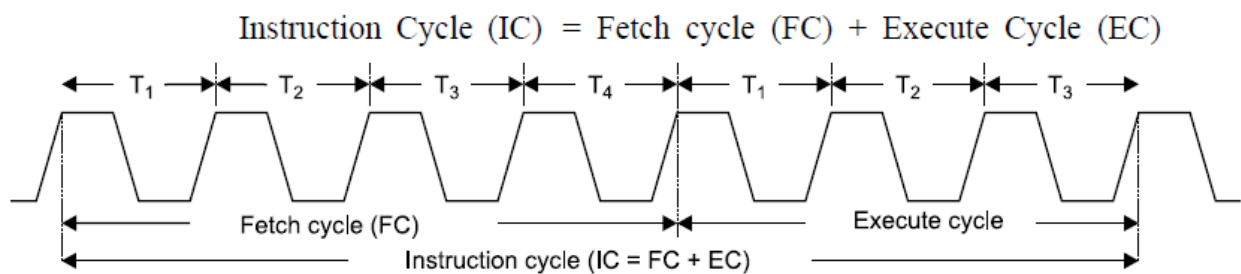


Fig.8 Processor cycle

Rules To Identify Number Of Machine Cycles In An Instruction:

- If an addressing mode is direct, immediate or implicit then No. of machine cycles = No. of bytes.
- If the addressing mode is indirect then No. of machine cycles = No. of bytes + 1.
Add +1 to the No. of machine cycles if it is memory read/write operation.
- If the operand is 8-bit or 16-bit address then, No. of machine cycles = No. of bytes +1.
- These rules are applicable to 80% of the instructions of 8085.

Timing Diagram of Opcode Fetch

The process of Opcode fetch operation requires minimum 4-clock cycles T1, T2, T3, and T4 and is the 1st machine cycle (M1) of every instruction.

Example

Fetch a byte 41H stored at memory location 2105H.

For fetching a byte, the microprocessor must find out the memory location where it is stored. Then provide condition (control) for data flow from memory to the microprocessor. The process of data flow and timing diagram of fetch operation are shown in Fig. 9. The microprocessor fetches Opcode of the instruction from the memory as per the sequence below

- A low IO/M means microprocessor wants to communicate with memory.
- The microprocessor sends a high on status signal S1 and S0 indicating fetch operation.
- The microprocessor sends 16-bit address. AD bus has address in 1st clock of the 1st machine cycle, T1.
- AD7 to AD0 address is latched in the external latch when ALE = 1.
- AD bus now can carry data.
- In T2, the RD control signal becomes low to enable the memory for read operation.
- The memory places opcode on the AD bus
- The data is placed in the data register (DR) and then it is transferred to IR.
- During T3 the RD signal becomes high and memory is disabled.
- During T4 the opcode is sent for decoding and decoded in T4.
- The execution is also completed in T4 if the instruction is single byte.
- More machine cycles are essential for 2- or 3-byte instructions. The 1st machine cycle M1 is meant for fetching the opcode. The machine cycles M2 and M3 are required either read/ write data or address from the memory or I/O devices.

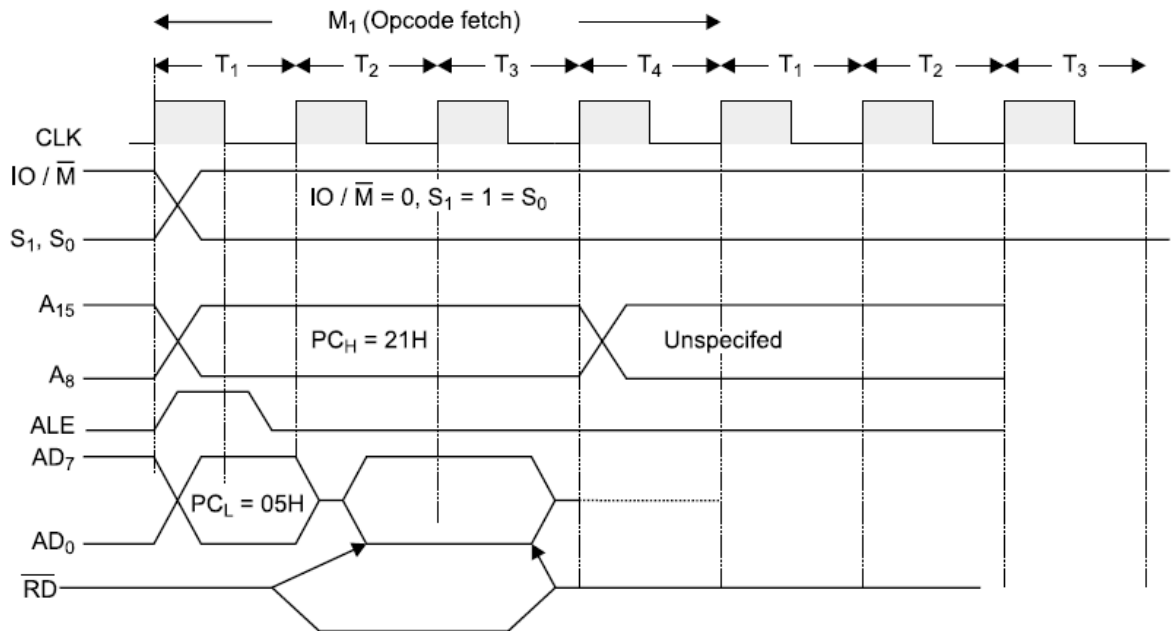


Fig. 9 Opcode fetch

Example For Opcode Fetch

- Explain the execution of MVI B, 05H stored at locations indicated below

Mnemonics	Machine code	Memory Locations
MVI B, 05H	06H	1000H
	05H	1001H

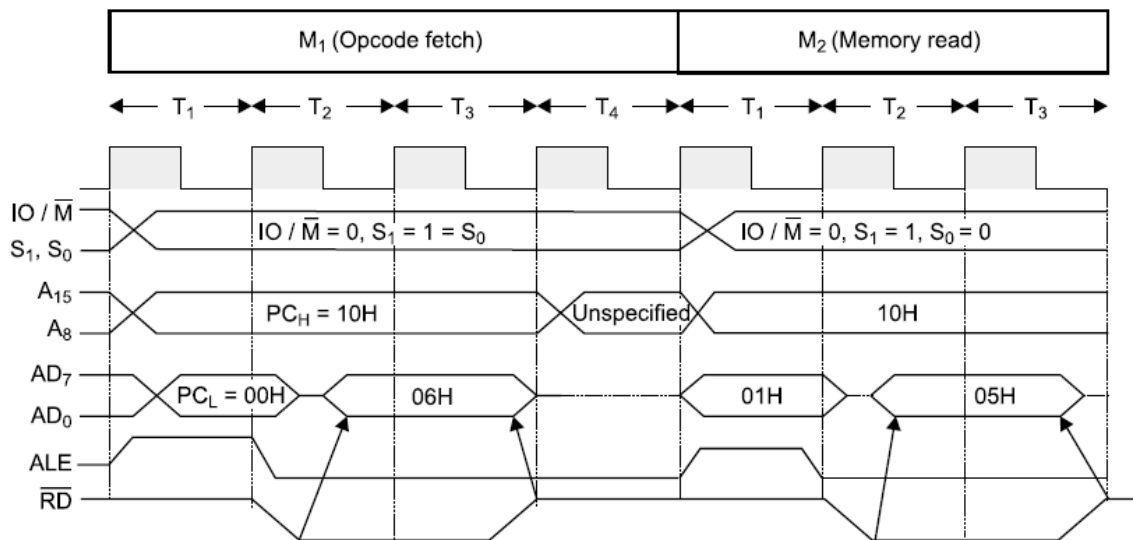


Fig. 10 Timing diagram for MVI B,05H

The MVI B, 05H instruction requires 2-machine cycles (M1 and M2). M1 requires 4-states and M2 requires 3-states, total of 7-states as shown in Fig. 10. Status signals IO/M, S1 and S0 specifies the 1st machine cycle as the op-code fetch. In T1-state, the high order address {10H} is placed on the bus A15 $\leftrightarrow$ A8 and low-order address {00H} on the bus AD7 $\leftrightarrow$ AD0 and ALE = 1. In T2 -state, the RD line goes low and the data 06 H from memory location 1000H are placed on the data bus. The fetch cycle becomes complete in T3-state. The instruction is decoded in the T4-state. During T4-state, the contents of the bus are unknown. With the change in the status signal, IO/M = 0, S1 = 1 and S0 = 0, the 2nd machine cycle is identified as the memory read. The address is 1001H and the data byte [05H] is fetched via the data bus. Both M1 and M2 perform memory read operation, but the M1 is called op-code fetch i.e., the 1st machine cycle of each instruction is identified as the opcode fetch cycle.

<i>Mnemonic</i>	<i>Instruction Byte</i>	<i>Machine Cycle</i>	<i>T-states</i>
MVI B,05H	Opcode	Opcode Fetch	4
	Immediate Data	Read Immediate Data	3
			7

TIMING DIAGRAM OF MEMORY READ

Operation:

- It is used to fetch one byte from the memory.
- It requires 3 T-States.
- It can be used to fetch operand or data from the memory.
- During T1, A8-A15 contains higher byte of address. At the same time ALE is high. Therefore Lower byte of address A0-A7 is selected from AD0-AD7 as shown in fig 11.
- Since it is memory ready operation, IO/M (bar) goes low.
- During T2 ALE goes low, RD (bar) goes low. Address is removed from AD0-AD7 and data D0-D7 appears on AD0-AD7.
- During T3, Data remains on AD0-AD7 till RD (bar) is at low signal.

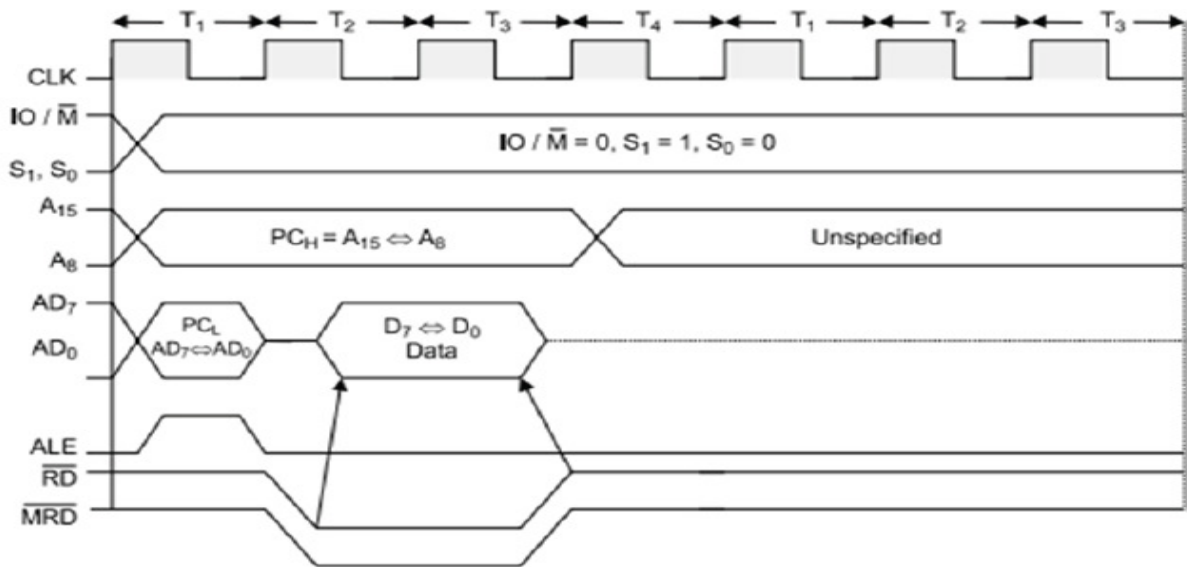


Fig 11.Memory read timing diagram

TIMING DIAGRAM OF MEMORY WRITE

Operation:

- It is used to send one byte into memory.
- It requires 3 T-States.
- During T1, $\overline{ALE}$ is high and contains lower address A0-A7 from AD0-AD7.
- A8-A15 contains higher byte of address.
- As it is memory operation, $\overline{IO/\overline{M}}$ (bar) goes low.
- During T2, $\overline{ALE}$ goes low, $\overline{WR}$ (bar) goes low and Address is removed from AD0-AD7 and then data appears on AD0-AD7 as in fig 12.
- Data remains on AD0-AD7 till $\overline{WR}$ (bar) is low.

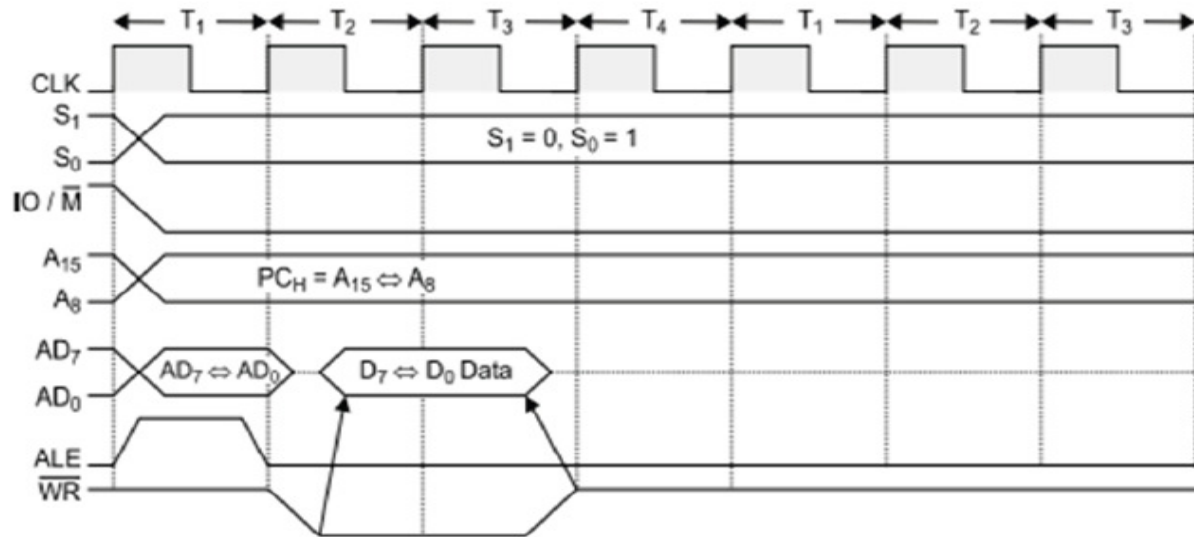


Fig 12.Memory Write timing diagram

TIMING DIAGRAM OF IO READ

Operation:

- It is used to fetch one byte from an IO port.
- It requires 3 T-States.
- During T1, The Lower Byte of IO address is duplicated into higher order address bus A8-A15 as in fig13.
- ALE is high and AD0-AD7 contains address of IO device.
- IO/M (bar) goes high as it is an IO operation.
- During T2, ALE goes low, RD (bar) goes low and data appears on AD0-AD7 as input from IO device.
- During T3 Data remains on AD0-AD7 till RD (bar) is low.

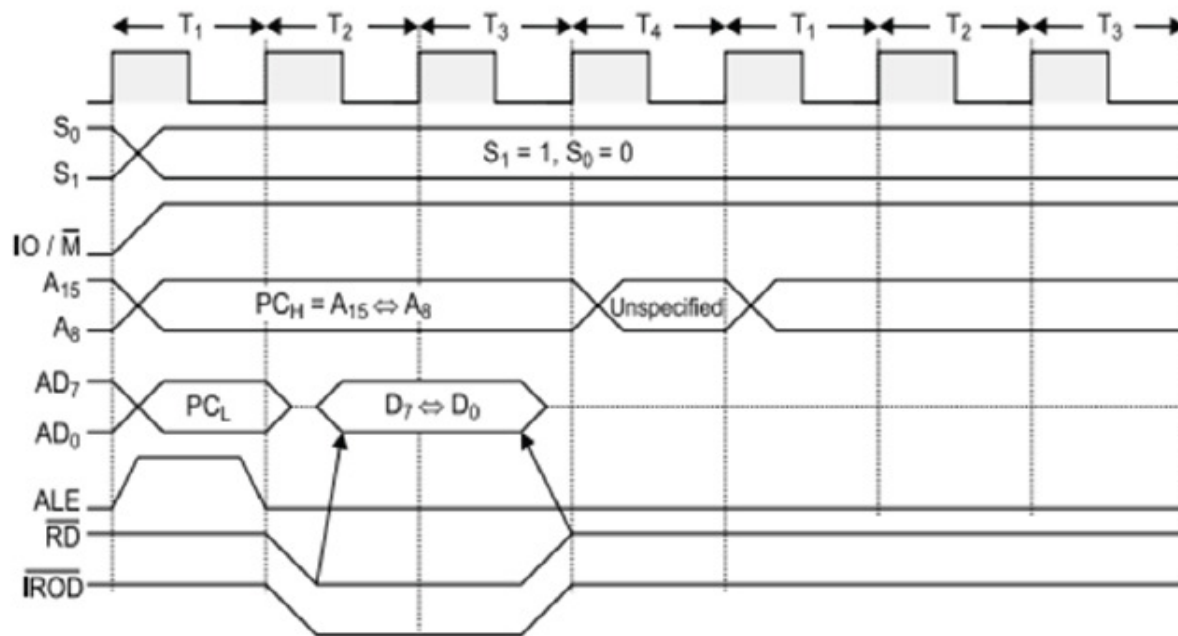


Fig 13.IO Read timing diagram

TIMING DIAGRAM OF IO WRITE

Operation:

- It is used to write one byte into IO device.
- It requires 3 T-States.
- During T₁, the lower byte of address is duplicated into higher order address bus A₈-A₁₅ as in fig 14.
- ALE is high and A₀-A₇ address is selected from AD₀-AD₇.
- As it is an IO operation IO/M (bar) goes low.
- During T₂, ALE goes low, WR (bar) goes low and data appears on AD₀-AD₇ to write data into IO device.
- During T₃, Data remains on AD₀-AD₇ till WR(bar) is low.

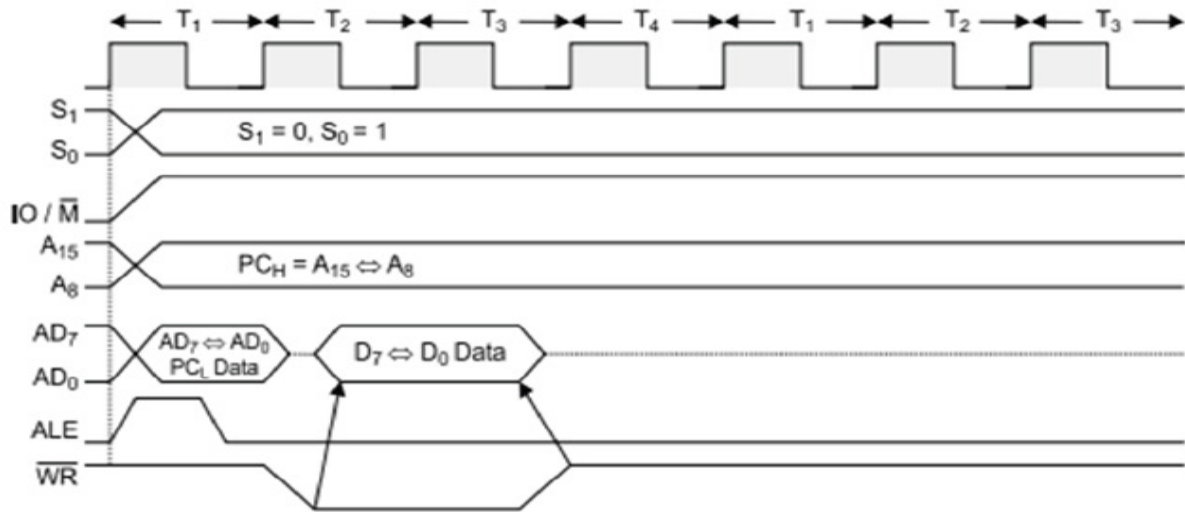


Fig 14. IO Write timing diagram