# Assembly language programming basics

- Each personal computer has a microprocessor that manages the computer's arithmetical, logical and control activities.
- Each family of processors has its own set of instructions for handling various operations such as getting input from keyboard, displaying information on screen and performing various other jobs.
- These set of instructions are called machine language instruction.
- A processor understands only machine language instructions, which are strings of 1's and 0's.
- However, machine language is too obscure and complex for using in software development.
- So, the low-level assembly language is designed for a specific family of processors that represents various instructions in symbolic code and a more understandable form.
- 8085 is a microprocessor with 8 bit word length.
- Its instruction set is designed by using various combination of these 8 bits.
- An **instruction** is a binary pattern entered through an input device in memory to command the microprocessor to perform the specific function.

Eg: 0011 1100 is an instruction that increments the number in the register called the accumulator by one.

1000 0000 is an instruction that adds the number in the register called B to the number in the accumulator and keeps the sum in the accumulator.

- 8085 has 246 such bit patterns, amounting to 74 different instructions for performing various operations.
- This 74 different instructions are called its **instruction set**.
- This binary language with predetermined instruction set is called 8085 **machine language**.
- Binary instruction 0011 1100 is equivalent to 3C hexadecimal.
- Even though the instructions are written in hexadecimal code, it is still difficult to understand a program written in hexadecimal number.
- So there will be a symbolic code for each instruction, called mnemonic.
- Which means operation to be performed by that instruction.
- The binary code 0011 1100 is represented by the mnemonic INR A

- INR A represents increment and A is Accumulator.
- i.e:- incrementing accumulator by one
- 10000 0000 represents ADD B
- ADD represents addition B represents contents in register B.
- Ie:- addition of contents in B and contents in accumulator.
- The complete set of 8085 mnemonics is called **8085 assembly language** and program written in these mnemonics is called an **assembly language program**
- Mnemonics can be written electronically on a computer and translated into binary code by using program called **assembler**
- Each defined instruction code performs a unique task.
- These instruction codes are called **operation code or opcode.**
- There are exact 74 basic functions.
- Size of 8085 microprocessor instruction code can either be 1byte, 2byte or 3byte
- Example:- MOV A,B

Move data present in register B to Accumulator

- Binary code corresponding to the instruction is 0111 1000 and its opcode is 78H

# 8085 Instruction Set

## Data transfer instructions

- An **instruction** of a computer is a command given to the computer to perform a specific operation on given data.
- In microprocessor, the instruction set is the collection of the instructions that the microprocessor is designed to execute.
- The programmer writes a program in assembly language using these instructions..
- 8 bit microprocessor can have 2^8=256 combinations of bit patterns.
- 8085 uses 246 combinations and 74 instructions.

This 74 different instructions are called **instruction set**

- **M –** Memory location pointed by HL register pair
- **Rd –** Destination register
- **Rs –** Source register
- **Opcode –** operation to be performed

- **Operand –** data to be operated

## Data transfer instruction

- This group of instructions copies data from source to destination.
- The content of the source is not altered.
- Instructions, which are used to transfer data from one register to another register, from memory to register or register to memory come under this group.

### 1.Copy from source to destination

**MOV Rd, Rs**

**M, Rs**

**Rd, M**

- This instruction copies the contents of the source register into the destination register.
- Contents of source register are not altered.
- Example: MOV B, C or MOV B, M

### 2.Move immediate 8-bit

**MVI Rd, data**

**M,data**

- The 8-bit data is stored in the destination register or memory.
- Example: MVI B, 5FH or MVI M, 57H

### 3. Load accumulator

**LDA 16-bit address**

- The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator
- Example :LDA 2034H

### 4. Load accumulator indirect

**LDAX B/D Reg.pair**

- The contents of the designated register pair point to a memory location.
- The instruction copies the contents of that memory location into the accumulator.

Example : LDAX B

## 5. Load register pair immediate

**LXI Reg.pair, 16-bit data**

- The instruction loads 16-bit data into the register pair designated in the operand.

  Example: LXI B, 2034H

## 6. Store accumulator indirect

**STAX Reg.pair**

- The contents of the accumulator are copied into the memory location specified by the contents of the operand (register pair)

  Example : STAX B

## 7. Store accumulator direct

**STA 16-bit address**

- The contents of the accumulator are copied into the memory location specified by the operand.

  Example : STA 4350H

## 8. Output data from accumulator to a port with 8-bit address

**OUT 8-bit port address**

Example : OUT F8H

## 9.Input data to a accumulator from a port with 8-bit address

**IN 8-bit port address**

Example: IN 8CH

# Arithmetic Instructions

## Add register or memory to accumulator : ADD R

**M**

- The contents of the operand are added to the contents of the accumulator and the result is stored in the accumulator.
- If the operand is a memory location, its location is specified by the contents of the HL registers.
- All flags are modified to reflect the result of the addition.
- Example : ADD B or ADD M

## Add register to accumulator with carry : ADC R

**M**

- The contents of the operand and the Carry flag are added to the content of the accumulator and the result is stored in the accumulator.
- Example : ADC B or ADC M

## Add immediate to accumulator : ADI 8-bit data

- The 8-bit data is added to the contents of the accumulator and the result is stored in the accumulator.

  Example : ADI 45H

## Add immediate to accumulator with carry: ACI 8-bit data

- The 8-bit data and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator.
- Example : ACI 45H

## Add register pair to H and L registers: DAD Reg.pair

- The 16-bit contents of the specified register pair are added to the contents of the HL register and the sum is stored in the HL register.
- If the result is larger than 16 bits, the CY flag is set.
- Example : DAD H

## Subtract register or memory from accumulator: SUB R

**M**

- The contents of the operand (register or memory) are subtracted from the contents of the accumulator, and the result is stored in the accumulator.
- Example : SUB B or SUB M

## Subtract source and borrow from accumulator : SBB R

**M**

- The contents of the operand and the Borrow flag are subtracted from the contents of the accumulator and the result is placed in the accumulator.
- Example: SBB B or SBB M

### Subtract immediate from accumulator : SUI 8-bit data

- The 8-bit data(operand) is subtracted from the contents of the accumulator and the result is stored in the accumulator.
- Example : SUI 45H

### Subtract immediate from accumulator with borrow : SBI 8-bit data

- The 8-bit data(operand) and the Borrow flag are subtracted from the contents of the accumulator and the result is stored in the accumulator.
- Example : SBI 45H

### Increment register or memory by 1 : INR R / M

- The contents of the designated register or memory are incremented by 1 and the result is stored in the same place.
- Example : INR B or INR M

### Increment register pair by 1 : INX R

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

### Decrement register or memory by 1 : DCR R / M

- The contents of the designated register or memory are decremented by 1 and the result is stored in the same place.
- Example: DCR B or DCR M

### Decrement register pair by 1 : DCX R

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

# 8085 Logical , shift and rotate instructions

## Logical AND register or memory with accumulator: ANA R

**M**

- The contents of the accumulator are logically ANDed with the contents of the operand (register or memory), and the result is placed in the accumulator.
- Example: ANA B or ANA M

## Logical AND immediate with accumulator : ANI 8-bit data

- The contents of the accumulator are logically ANDed with the 8-bit data (operand) and the result is placed in the accumulator.
- Example: ANI 86H

## Exclusive OR register or memory with accumulator: XRA R

**M**

- The contents of the accumulator are Exclusive ORed with the contents of the operand (register or memory) , and the result is placed in the accumulator.
- Example: XRA B or XRA M

## Exclusive OR immediate with accumulator: XRI 8-bit data

- The contents of the accumulator are Exclusive ORed with the 8-bit data (operand) and the result is placed in the accumulator.
- Example: XRI 86H

## Logical OR register or memory with accumulator: ORA R

**M**

- The contents of the accumulator are logically ORed with the contents of the operand (register or memory), and the result is placed in the accumulator.
- Example: ORA B or ORA M

## Logical OR immediate with accumulator: ORI 8-bit data

- The contents of the accumulator are logically ORed with the 8-bit data (operand) and the result is placed in the accumulator.
- Example: ORI 86H

### Rotate accumulator left : RLC none

- Each binary bit of the accumulator is rotated left by one position.
- Example :RLC
- Others are : RAL, RAR, RRC

### Compare register or memory with accumulator: CMP R

**M**

- The contents of the operand (register or memory) are compared with the contents of the accumulator.

  The result of the comparison is shown by setting the flags

### Compare immediate with accumulator : CPI 8-bit data

- The second byte (8-bit data) is compared with the contents of the accumulator.
- Example : CPI 89H

### Complement accumulator : CMA none

- The contents of the accumulator are complemented.
- Example : CMA

### Complement carry : CMC none

- The carry flag is complemented.
- Example: CMC

### Set Carry : STC none

- The Carry flag is set to 1.
- Example : STC

## 8085 Branch instructions

### Jump unconditionally : JMP 16-bit address

- The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.
- Example : JMP 2034H or JMP XYZ

### Jump conditionally : Opcode 16-bit address

- The program sequence is transferred to the memory location specified by the 16bit address given in the operand
- Example: JZ 2034H

| Opcode | Description | Flag Status |
|---|---|---|
| JC | Jump on Carry | CY=1 |
| JNC | Jump on no carry | CY=0 |
| JP | Jump on positive | S=0 |
| JM | Jump on minus | S=1 |
| JZ | Jump on zero | Z=1 |
| JNZ | Jump on no zero | Z=0 |
| JPE | Jump om parity even | P=1 |
| JPO | Jump on parity odd | P=0 |

### CALL 16-bit address

- 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 2034H or CALL XYZ

### RET none

- 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

## Program control instruction

### Halt : HLT none

- 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

### NOP none : No operation is performed.

- Example : NOP

# 8085 Addressing modes

- The way of specifying data to be operated by an instruction is called addressing mode.
- In 8085 microprocessor there are 5 types of addressing modes:

## Immediate Addressing Mode:-

- In immediate addressing mode the source operand is always data. If the data is 8-bit, then the instruction will be of 2 bytes.
- MVI B, 45H: move the data 45H immediately to register B
- ADI 45H
- ANI 74H
- SUI 37H

## Register Addressing Mode:-

- In register addressing mode, the data to be operated is available inside the register(s) and register(s) is (are) operands.
- MOV A,B : move the contents of register B to register A
- ADD B

## Direct Addressing Mode:-

- In direct addressing mode, the data to be operated is available inside a memory location and that memory location is directly specified as an operand.
- LDA 2050H : Load the contents of memory location into accumulator A.
- IN 07H
- OUT 05H

  STA 2000H

## Register Indirect Addressing Mode:-

- In register indirect addressing mode, the data to be operated is available inside a memory location and that memory location is indirectly specified by a register pair.
- MOV M,A
- LDAX B
- STAX B

### Implied/Implicit Addressing Mode:-

- In implied/implicit addressing mode the operand is hidden and the data to be operated is available in the instruction itself.
- CMA : finds and stores the 1's complement of the contents of accumulator A in A.
- RAL
- RLC
- STC

## Timing diagram

- Representation of the changes and variations in the status of signals with respect to time is referred to as a **timing diagram.**
- Higher order address bus: A18 to A15.
- Lower order address/data bus : AD0 to AD7
- ALE: ALE stands for Address Latch Enable.
- This output signal given by the microprocessor tells us whether the AD0-AD7 is carrying the address or is available for data transfer.
- ALE = 0 (AD0-AD7 is available for data transfer)

  ALE = 1 (AD0-AD7 is carrying the lower eight bits of the address)
- RD: This is an active low signal and simply tells us whether it is a read operation when it is low.
- WR: This is an active low signal and simply tells us whether it is a write operation when it is low.
- Processors work in synchronization with a periodic signal called 'clock signal'.
- The part of any operation carried out during one time period of this clock signal is known as a **T state**.
- Sometimes, 'T state' is also used to refer to a time of one clock period.
- For the execution of any instructions, basically two steps are followed – **fetch and then execute.**
- The time (or the number of 'T states') required to fetch and execute an instruction is called an **instruction cycle.**
- Process of reading the code for the instruction to be executed is called the **fetch cycle.**

- After fetching the code for the instruction that is supposed to be executed, the next step is to execute that instruction.
- This process is referred to as an **execute cycle.**
- The time required for the microprocessor to access memory or an IO device either for a read operation or a write operation is called a **machine cycle.**
- Every microprocessor instruction is made up of **opcode and operand.**
- Opcode is the equivalent Hex code for instruction mnemonic (ex: for opcode MVI A,8-bit,3E is an equivalent Hex value).
- This Hex value is predefined and available in data sheet of microprocessor.
- Operand is the 8-bit(data) or 16-bit number (data/address) mentioned in instruction.
- When we write program for microprocessor, these instructions (opcode+operand) get stored in program memory.
- In our example, opcode 3E get stored at address 2000H and operand 32H get stored at address 2001H.
- Every instruction is made of 1 or few machine cycles which is made of T-states.
- T-states are smallest unit of execution in microprocessor ,which is equivalent to 1 clock cycle.
- For this instruction (ex: MVI A,32H), we need 2 machine cycles (M1 and M2).
- Machine cycle M1 is of 4 T-states, where 3 T-states are used for opcode fetch from program memory and 1 T-state is used for its decoding.
- Similarly, Machine cycle M2 is made of 3 T-states which is needed for operand fetch from program memory.
- We need only 3 T-states in M2 since decoding is not needed for operand whereas opcode need extra 1 T-state for decoding process.
- **The 2 machine cycles are**

  **1.Opcode fetch**

  **2.Memory read**

## Opcode Fetch

**T1:** address (2000H) passing and making ALE high

**T2:** passing data (3E) in the location 2000H and making ALE low.

**T3:** Completing opcode fetch.

**T4:** Decoding the operand MVI A. ie; 3E

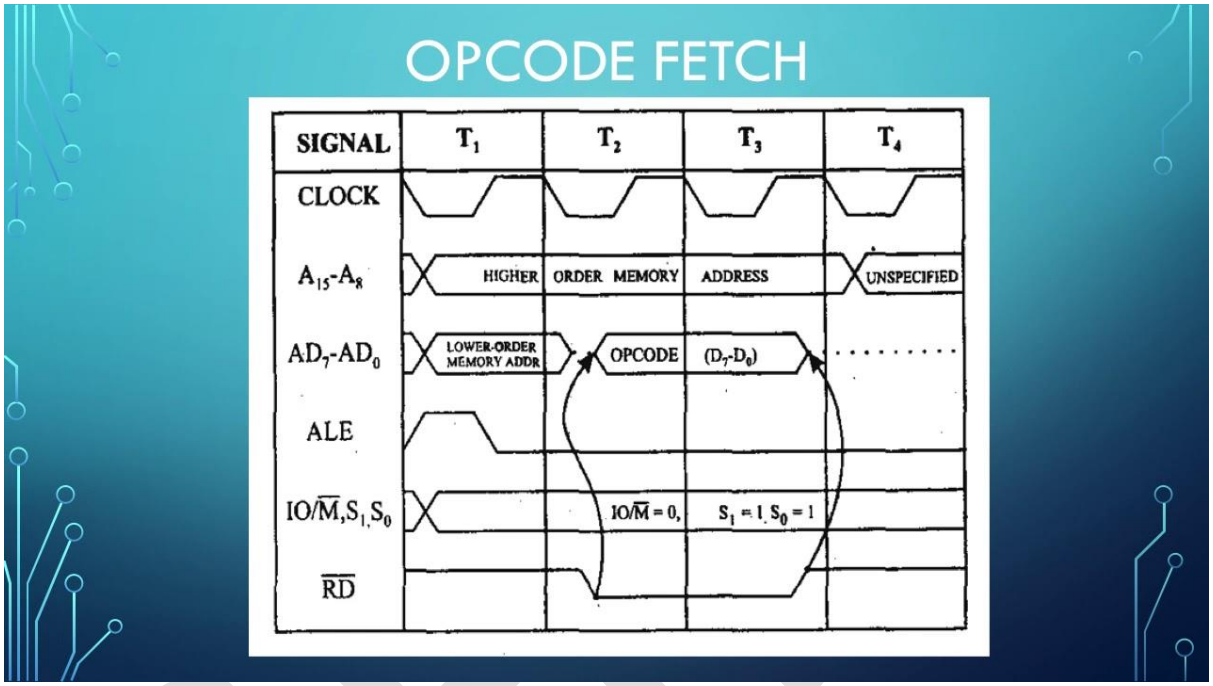
## Memory Read

**T1:** Address (2001H) passing and making ALE high

**T2:** Read data 32H from the location 2001H

**T3:** move data to Accumulator.

Thus total of 7 T states (2 machine cycles)



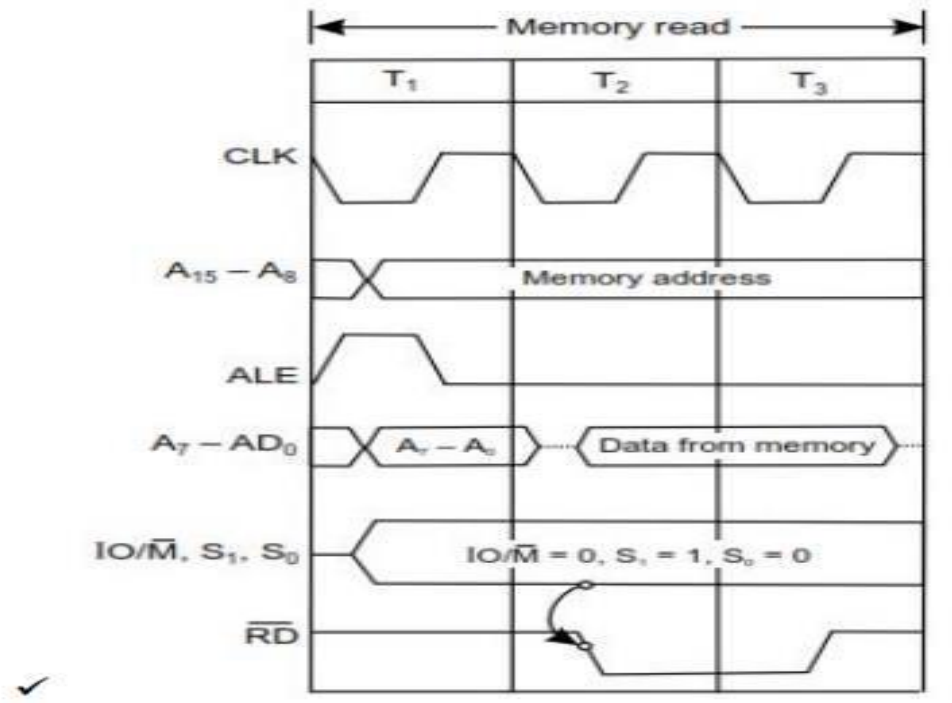
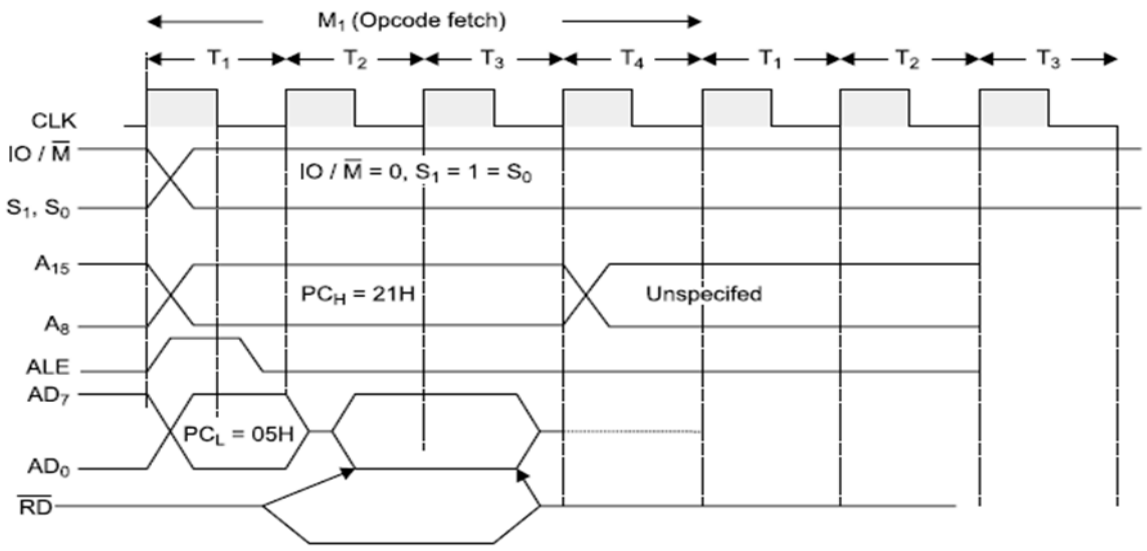
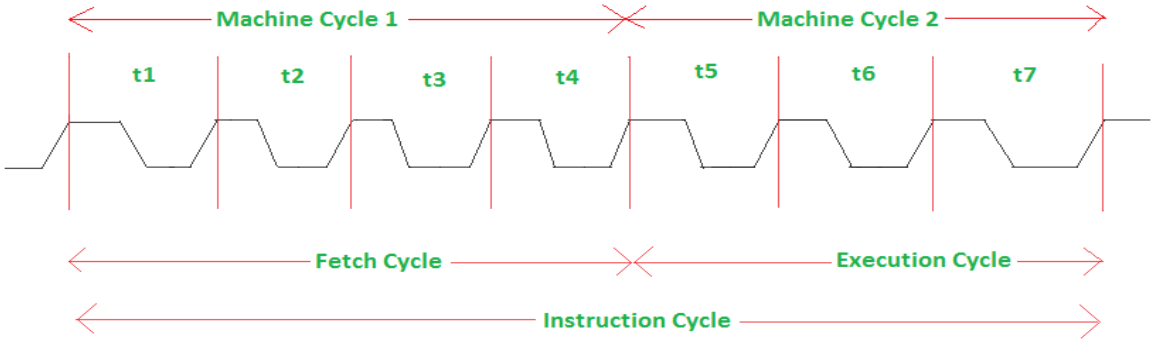




**Fig 1.9 Memory Read Machine Cycle**




## Opcode Fetch Machine Cycle

- Each instruction in 8085 microprocessor consists of two part- operation **code(opcode) and operand.**
- The opcode is a command such as ADD and the operand is an object to be operated on, such as a byte or the content of a register.
- **Instruction Cycle**: The time taken by the processor to complete the execution of an instruction.
- An instruction cycle consists of one to six machine cycles.
- **Machine Cycle:** The time required to complete one operation ; accessing either the memory or I/O device.
- A machine cycle consists of three to six T-states.
- **T-State:** Time corresponding to one clock period. It is the basic unit to calculate execution of instructions or programs in a processor.




**Instruction cycle in 8085 microprocessor**

**To execute a program, 8085 performs various operations as:**

- **Opcode fetch**
- **Operand fetch**
- **Memory read/write**
- **I/O read/write**

**External communication function are:**

- **Memory read/write**
- **I/O read/write**
- **Interrupt request acknowledge**

## Opcode Fetch Machine Cycle

- It is the first step in the execution of any instruction.

### T1 clock cycle.

- The content of PC is placed in the address bus: AD0-AD7 lines contains lower bit address and A8-A15 contains higher bit address.
- IO/M signal is low indicating that a memory location is being accessed.
- ALE is high, indicates that multiplexed AD0-AD7 act as lower order address bus.

### T2 Clock Cycle

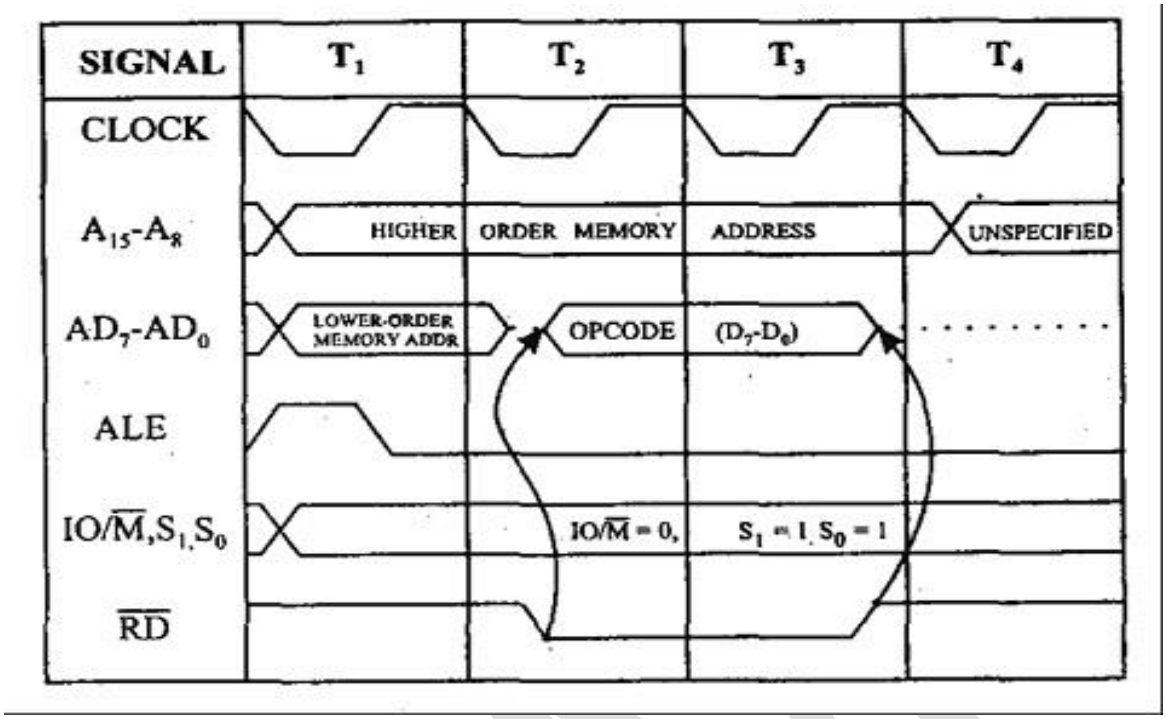
- Multiplexed address bus is now changed to data bus.
- The RD signal is made low by the processor.
- This signal makes the memory device load the data bus with the contents of the location addressed by the processor.

### T3 Clock Cycle

- The opcode available on the data bus is ready by the processor and moved to the instruction register.
- The RD signal is deactivated by making it logic 1.

### T4 Clock Cycle

- The processor decode the instruction in the instruction register and generate the necessary control signals to execute the instruction.




## BCD ADDITION PROGRAM

| Decimal | Binay (BCD) 8 4 2 1 |
|---|---|
| 0 | 0 0 0 0 |
| 1 | 0 0 0 1 |
| 2 | 0 0 1 0 |
| 3 | 0 0 1 1 |
| 4 | 0 1 0 0 |
| 5 | 0 1 0 1 |
| 6 | 0 1 1 0 |
| 7 | 0 1 1 1 |
| 8 | 1 0 0 0 |
| 9 | 1 0 0 1 |

**Eg: 82+73**

**1000 0010** **1111**

**0111 0011** **0110**

**1111 0101** **10101**

**LXI H,2000H**

**MOV A,M**

**INX H**

**ADD M**

**DAA(DECIMAL ADJUST AFTER ADDITION)**

**LXI H,3000H**

**MOV M,A**

**JNC DONE**

**MVI A,01H**

**INX H**

**MOV M,A**

**DONE HLT**