UNIT I – 8085 MICROPROCESSOR

1.1 Introduction

Microcomputer: The term microcomputer is generally synonymous with personal computer, or a computer that depends on a microprocessor.

- Microcomputers are designed to be used by individuals, whether in the form of PCs, workstations or notebook computers.
- A microcomputer contains a CPU on a microchip (the microprocessor), a memory system (typically ROM and RAM), a bus system and I/O ports, typically housed in a motherboard.

Microprocessor: A silicon chip that contains a CPU. In the world of personal computers, the terms microprocessor and CPU are used interchangeably.

- A microprocessor (sometimes abbreviated μP) is a digital electronic component with miniaturized transistors on a single semiconductor integrated circuit (IC).
- One or more microprocessors typically serve as a central processing unit (CPU) in a computer system or handheld device.
- Microprocessors made possible the advent of the microcomputer.
- At the heart of all personal computers and most working stations sits a microprocessor.
- Microprocessors also control the logic of almost all digital devices, from clock radios to fuel-injection systems for automobiles.
- Three basic characteristics differentiate microprocessors:
- Instruction set: The set of instructions that the microprocessor can execute.
- Bandwidth: The number of bits processed in a single instruction.
- Clock speed: Given in megahertz (MHz), the clock speed determines how many instructions per second the processor can execute.
- In both cases, the higher the value, the more powerful the CPU. For example, a 32-bit microprocessor that runs at 50MHz is more powerful than a 16-bit microprocessor that runs at 25MHz.

- In addition to bandwidth and clock speed, microprocessors are classified as being either RISC (reduced instruction set computer) or CISC (complex instruction set computer)

1.2 8085 Microprocessor

The Intel 8085 is an 8-bit microprocessor introduced by Intel in 1977. It was binary compatible with the more-famous Intel 8080 but required less supporting hardware, thus allowing simpler and less expensive microcomputer systems to be built. The "5" in the model number came from the fact that the 8085 requires only a +5-Volt (V) power supply rather than the +5 V, −5 V and +12 V supplies the 8080 needed. The main features of 8085 μP are:

- It is an 8-bit microprocessor.
- It is manufactured with N-MOS technology.
- It has 16-bit address bus and hence can address up to 2^{16}= 65536 bytes (64KB) memory locations through A0–A15.
- The first 8 lines of address bus and 8 lines of data bus are multiplexed AD0–AD7
- Data bus is a group of 8 lines D0–D7.
- It supports external interrupt request.
- A 16-bit program counter (PC)
- A 16-bit stack pointer (SP)
- Six 8-bit general purpose register arranged in pairs: BC, DE, HL.
- It requires a signal +5V power supply and operates at 3.2 MHZ single phase clock.
- It is enclosed with 40 pins DIP (Dual in line package).

1.3 8085 Architecture

8085 consists of various units as shown in Fig. 1 and each unit performs its own functions. The various units of a microprocessor are listed below

- Accumulator
- Arithmetic and logic Unit
- General purpose register
- Program counter
- Stack pointer
- Temporary register
- Flags
- Instruction register and Decoder

- Timing and Control unit
- Interrupt control
- Address buffer and Address-Data buffer
- Address bus and Data bus

Accumulator

Accumulator is nothing but a register which can hold 8-bit data. Accumulator aids in storing two quantities. The data to be processed by arithmetic and logic unit is stored in accumulator. It also stores the result of the operation carried out by the Arithmetic and Logic unit. The accumulator is also called an 8-bit register. The accumulator is connected to Internal Data bus and ALU (arithmetic and logic unit). The accumulator can be used to send or receivedata from the Internal Data bus.

Arithmetic and Logic Unit

There is always a need to perform arithmetic operations like +, -, *, / and to perform logical operations like AND, OR, NOT etc. So, there is a necessity for creating a separate unit which can perform such types of operations. These operations are performed by the Arithmetic and Logic Unit (ALU). ALU performs these operations on 8-bit data. But these operations cannot be performed unless we have an input (or) data on which the desired operation is to be performed. So, from where do these inputs reach the ALU? For this purpose, accumulator is used. ALU gets its Input from accumulator and temporary register. After processing the necessary operations, the result isstored back in accumulator.

General Purpose Registers

Apart from accumulator 8085 consists of six special types of registers called General Purpose Registers. These general-purpose registers are used to hold data like any other registers. The general-purpose registers in 8085 processors are B, C, D, E, H and L. Each register can hold 8-bit data. Apart from the above function these registers can also be used to work in pairs to hold 16-bit data. They can work in pairs such as B-C, D-E and H-L to store 16-bit data. The H-L pair works as a memory pointer. A memory pointer holds the address of a particular memory location. They can store 16-bit address as they work in pair.

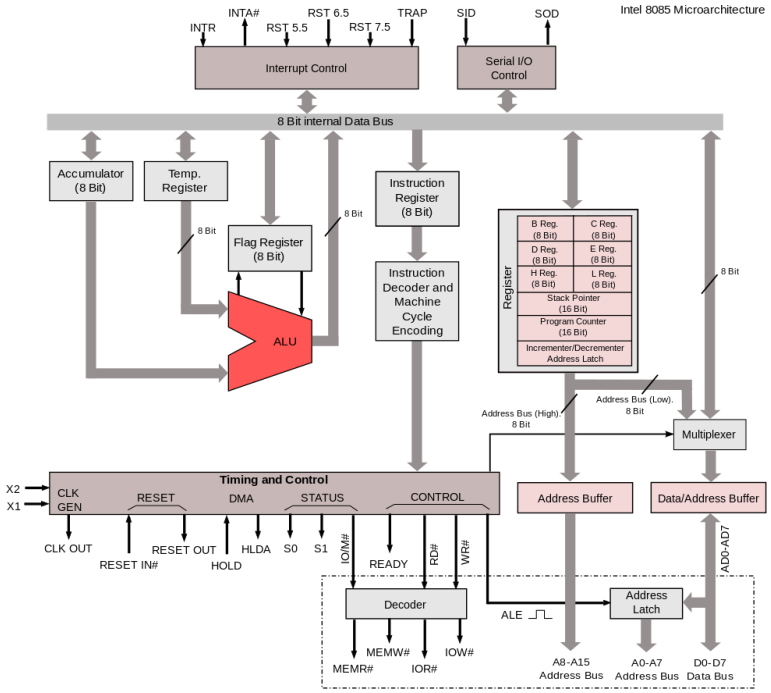

Fig. 1.1 8085 Architecture

Program Counter and Stack Pointer

Program counter is a special purpose register.

Consider that an instruction is being executed by processor. As soon as the ALU finished executing the instruction, the processor looks for the next instruction to be executed. So, there is a necessity for holding the address of the next instruction to be executed in order to save time. This is taken care by the program counter. A program counter stores the address of the next instruction to be executed. In other words, the program counter keeps track of the memory address of the instructions that are being executed by the microprocessor and the memory address of the next instruction that is going to be executed.

Microprocessor increments the program whenever an instruction is being executed, so that the program counter points to the memory address of the next instruction that is going to be executed. Program counter is a 16-bit register.

Stack pointer is also a 16-bit register which is used as a memory pointer. A stack is nothing but the portion of RAM (Random access memory).

So, does that mean the stack pointer points to portion of RAM?

Yes. Stack pointer maintains the address of the last byte that is entered into stack.

Each time when the data is loaded into stack, Stack pointer gets decremented. Conversely it is incremented when data is retrieved from stack.

Temporary Register

As the name suggests this register acts as a temporary memory during the arithmetic and logical operations. Unlike other registers, this temporary register can only be accessed by the microprocessor and it is completely inaccessible to programmers. Temporary register is an 8-bit register.

Flags

Flags are nothing but a group of individual Flip-flops. The flags are mainly associated with arithmetic and logic operations. The flags will show either a logical (0 or 1) (i.e.) a set or reset depending on the data conditions in accumulator or various other registers. A flag is actually a latch which can hold some bits of information. It alerts the processor that some event has taken place.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z		AC		P		CY

Fig. 1.2 Flag Register

Intel processors have a set of 5 flags.

1. Carry flag
2. Parity flag
3. Auxiliary carry flag
4. Zero flag
5. Sign flag

Consider two binary numbers.

For example

1100 0000

1000 0000

When we add the above two numbers, a carry is generated in the most significant bit. The number in the extreme right is least significant bit, while the number in extreme left is most significant bit. So, a ninth bit is generated due to the carry. So how to accommodate 9th bit in an 8-bit register?

For this purpose, the Carry flag is used. The carry flag is set whenever a carry is generated and reset whenever there is no carry. But there is an

auxiliary carry flag? What is the difference between the carry flag and auxiliary carry flag?

Let's discuss with an example. Consider the two numbers given below

0000 1100

0000 1001

When we add both the numbers a carry is generated in the fourth bit from the least significant bit. This sets the auxiliary carry flag. When there is no carry, the auxiliary carry flag is reset. So, whenever there is a carry in the most significant bit Carry flag is set. While an auxiliary carry flag is set only when a carry is generated in bits other than the most significant bit.

Parity checks whether it's even or add parity. This flag returns a 0 if it is odd parity and returns a 1 if it is an even parity. Sometimes they are also called as parity bit which is used to check errors while data transmission is carried out.

Zero flag shows whether the output of the operation is 0 or not. If the value of Zero flag is 0 then the result of operation is not zero. If it is zero the flag returns value 1.

Sign flag shows whether the output of operation has positive sign or negative sign. A value 0 is returned for positive sign and 1 is returned for negative sign.

Instruction Register and Decoder

Instruction register is 8-bit register just like every other register of microprocessor. Consider an instruction. The instruction may be anything like adding two data's, moving a data, copying a data etc. When such an instruction is fetched from memory, it is directed to Instruction register. So, the instruction registers are specifically to store the instructions that are fetched from memory. There is an Instruction decoder which decodes the information present in the Instruction register for further processing.

Timing and Control Unit

Timing and control unit is a very important unit as it synchronizes the registers and flow of data through various registers and other units. This unit consists of an oscillator and controller sequencer which sends control signals needed for internal and external control of data and other units. The oscillator generates two-phase clock signals which aids in synchronizing all the registers of 8085 microprocessor.

Signals that are associated with Timing and control unit are:

Control Signals: RD', WR', ALE

- ALE is used for provide control signal to synchronize the components of microprocessor and timing for instruction to perform the operation.
- RD (Active low) and WR (Active low) are used to indicate whether the operation is reading the data from memory or writing the data into memory respectively.

Status Signals: S0, S1, IO/M'

- IO/M (Active low) is used to indicate whether the operation belongs to the memory or peripherals.

Table 1.1 Status signals and the status of data bus

IO/M' (Active Low)	S1	S2	Data Bus Status (Output)
0	0	0	Halt
0	0	1	Memory WRITE
0	1	0	Memory READ
1	0	1	IO WRITE
1	1	0	IO READ
0	1	1	Op code fetch
1	1	1	Interrupt acknowledge

DMA Signals: HOLD, HLDA, READY

- HOLD: 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 the bus as soon as the completion of the current bus transfer. Internal processing can continue. The processor can regain the bus only after the HOLD is removed. When the HOLD is acknowledged, the Address, Data RD, WR and IO/M' lines are tri-stated.
- HLDA: Hold Acknowledge: Indicates that the CPU has received the HOLD request and that it will relinquish the bus in the next clock cycle HLDA goes low after the Hold request is removed. The CPU takes the bus one half-clock cycle after HLDA goes low.
- READY: This signal synchronizes the fast CPU and the slow memory, peripherals. 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 an integral number of clock cycle for READY to go high before completing the read or write cycle. READY must conform to specified setup and hold times.

Reset Signals: Reset in, Reset Out

RESET IN: A low on this pin;

- Sets the program counter to zero (0000H)
- Resets the interrupt enables and HLDA flip-flops.
- Tri-states the data bus, address bus and control bus.
- Affects the content of processors internal registers randomly.

On Reset, The Program counter sets to 0000h which causes the 8085 to execute; the first instruction from address 0000H.

- RESET OUT: This active high signal indicates that the processor; is being reset. This signal is synchronized to the processor clock and it can be used to reset other devices connected in the system.

Interrupt control

As the name suggests this control interrupts a process. Consider that a microprocessor is executing the main program. Now whenever the interrupt signal is enabled or requested the microprocessor shifts the control from main program to process the incoming request and after the completion of request, the control goes back to the main program. For example, an Input/output device may send an interrupt signal to notify that the data is ready for input. The microprocessor temporarily stops the execution of main program and transfers control to I/O device. After collecting the input data, the control is transferred back to main program. Interrupt signals present in 8085 are:

- INTR
- RST 7.5
- RST 6.5
- RST 5.5
- TRAP

INTR is maskable 8080A compatible interrupt. When the interrupt occurs the processor fetches from the bus one instruction, usually one of these instructions: One of the 8 RST instructions (RST0 - RST7). The processor saves current program counter into stack and branches to memory location N * 8 (where N is a 3 - bit number from 0 to 7 supplied with the RST instruction).

CALL instruction (3-byte instruction). The processor calls the subroutine, address of which is specified in the second and third bytes of the instruction.

RST5.5 is a maskable interrupt. When this interrupt is received the processor saves the contents of the PC register into stack and branches to 2CH (hexadecimal) address.

RST6.5 is a maskable interrupt. When this interrupt is received the processor saves the contents of the PC register into stack and branches to 34H (hexadecimal) address.

RST7.5 is a maskable interrupt. When this interrupt is received the processor saves the contents of the PC register into stack and branches to 3CH (hexadecimal) address.

TRAP is a non-maskable interrupt. When this interrupt is received the processor saves the contents of the PC register into stack and branches to 24H (hexadecimal) address.

All maskable interrupts can be enabled or disabled using EI and DI instructions. RST5.5, RST6.5 and RST7.5 interrupts can be enabled or disabled individually using SIM instruction.

Serial Input/output control

The input and output of serial data can be carried out using 2 instructions in 8085.

- SID-Serial Input Data
- SOD-Serial Output Data

Two more instructions are used to perform serial-parallel conversion needed for serial I/O devices.

- SIM
- RIM

Address buffer and Address-Data buffer

The contents of the stack pointer and program counter are loaded into the address buffer and address-data buffer. These buffers are then used to drive the external address bus and address-data bus. As the memory and I/O chips are connected to these buses, the CPU can exchange desired data to the memory and I/O chips.

The address-data buffer is not only connected to the external data bus but also to the internal data bus which consists of 8-bits. The address data buffer can both send and receive data from internal data bus.

Address bus and Data bus

We know that 8085 is an 8-bit microprocessor. So, the data bus present in the microprocessor is also 8-bits wide. So, 8-bits of data can be transmitted from or to the microprocessor. But 8085 processor requires 16-bit address bus as the memory addresses are 16-bit wide. The 8 most significant bits of the address are transmitted with the help of address bus and the 8 least significant bits are transmitted with the help of multiplexed address/data bus. The eight- bit data bus is multiplexed with the eight least significant bits of address bus. The address/data bus is time multiplexed. This means for few microseconds, the 8 least significant bits of address are generated, while for next few seconds the same pin generates the data. This is called Time multiplexing. But there are situations where there is a need to transmit both data and address simultaneously. For this purpose, a signal called ALE (address latch enables) is used. ALE signal holds the obtained address in its latch for a long time until the data is obtained and so when the microprocessor sends the data next time the address is also available at the output latch. This technique is called Address/Data demultiplexing.

1.4 Pin Diagram of 8085

The signals can be grouped as follows

1. Power supply and clock signals
2. Address bus
3. Data bus
4. Control and status signals
5. Interrupts and externally initiated signals
6. Serial I/O ports

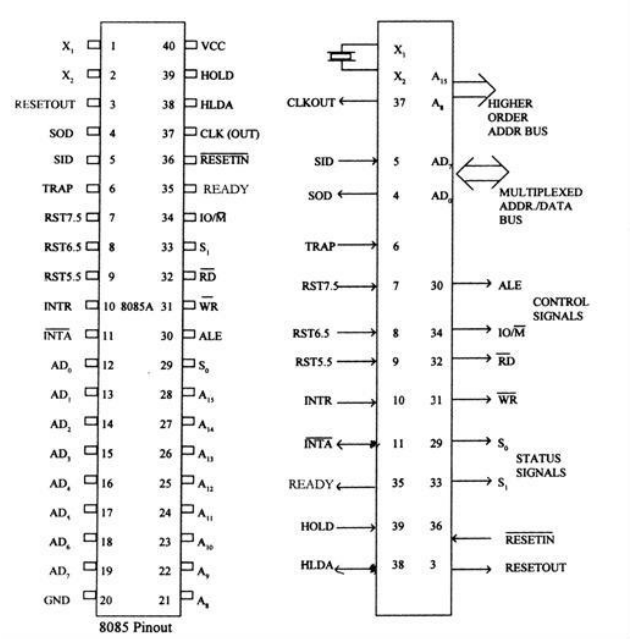

Fig. 1.3 Pin diagram of 8085

Power supply and Clock frequency signals

- Vcc + 5-volt power supply
- Vss Ground
- X1, X2: Crystal or R/C network or LC network connections to set the frequency of internal clock generator. The frequency is internally divided by two. Since the basic operating timing frequency is 3 MHz, a 6 MHz crystal is connected externally.
- CLK (output) – Clock Output is used as the system clock for peripheral and devices interfaced with the microprocessor.

Data Bus and Address Bus

AD0-AD7:-These are multiplexed address and data bus. So, it can be used to carry the lower order 8-bit address as well as the data. Generally, these lines are demultiplexed using the Latch. During the opcode fetch operation, in the first clock cycle the lines deliver the lower order address bus A0-A7. In the subsequent IO/M read or write it is used as data bus D0-D7. CPU can read or write data through these lines. A8-A15:- These are address bus used to address the memory location.

1.5 Instruction Set

The 8085 instruction set can be classified into the following five functional headings.

Data Transfer Instructions: Includes the instructions that moves (copies) data between registers or between memory locations and registers. In all data transfer operations, the content of source register is not altered. Hence the data transfer is copying operation.

Arithmetic Instructions: Includes the instructions, which performs the addition, subtraction, increment or decrement operations. The flag conditions are altered after execution of an instruction in this group.

Logical Instructions: The instructions which performs the logical operations like AND, OR, EXCLUSIVE-OR, complement, compare and rotate instructions are grouped under this heading. The flag conditions are altered after execution of an instruction in this group.

Branching Instructions: The instructions that are used to transfer the program control from one memory location to another memory location are grouped under this heading.

Machine Control Instructions: Includes the instructions related to interrupts and the instruction used to halt program execution.

1.6 Data Transfer Instructions

- These instructions move data between registers, or between memory and registers.
- These instructions copy data from source to destination.
- While copying, the contents of source are not modified.

Opcode	Operand	Description
MOV	Rd, Rs M, Rs Rd, M	Copy from source to destination.
MVI	Rd, Data M, Data	Move immediate 8-bit
LDA	16-bit address	Load Accumulator
LDAX	B/D Register Pair	Load accumulator indirect
LXI	Reg. pair, 16-bit data	Load register pair immediate
STA	16-bit address	Store accumulator direct
STAX	Reg. pair	Store accumulator indirect
XCHG	None	Exchange H-L with D-E

1.7 Arithmetic Instructions

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

Opcode	Operand	Description
ADD	R M	Add register or memory to accumulator
ADC	R M	Add register or memory to accumulator with carry
ADI	8-bit data	Add immediate to accumulator
ACI	8-bit data	Add immediate to accumulator with carry
SUB	R M	Subtract register or memory from accumulator
SUI	8-bit data	Subtract immediate from accumulator
INR	R M	Increment register or memory by 1
INX	R	Increment register pair by 1
DCR	R M	Decrement register or memory by 1
DCX	R	Decrement register pair by 1

1.8 Logical Instructions

These instructions perform various logical operations with the contents of the accumulator.

Opcode	Operand	Description
CMP	R M	Compare register or memory with accumulator
CMP	R M	Compare register or memory with accumulator
CPI	8-bit data	Compare immediate with accumulator
ANA	R M	Logical AND register or memory with accumulator
ANI	8-bit data	Logical AND immediate with accumulator
XRA	R M	Exclusive OR register or memory with accumulator
ORA	R M	Logical OR register or memory with accumulator
ORI	8-bit data	Logical OR immediate with accumulator
XRA	R M	Logical XOR register or memory with accumulator
XRI	8-bit data	XOR immediate with accumulator

1.9 Branching Instructions

This group of instructions alters the sequence of program execution either conditionally or unconditionally.

Opcode	Operand	Description
JMP	16-bit address	Jump unconditionally
Jx	16-bit address	Jump conditionally

1.10 Machine Control Instructions

These instructions control machine functions such as Halt, Interrupt, or do nothing.

Opcode	Operand	Description
HLT	None	Halt
NOP	None	No operation

EI	None	The interrupt enable flip-flop is set and all interrupts are enabled. No flags are affected.
DI	None	The interrupt enable flip-flop is reset and all the interrupts except the TRAP are disabled. No flags are affected.
SIM	None	This is a multipurpose instruction and used to implement the 8085 interrupts 7.5, 6.5, 5.5, and serial data output.
RIM	None	This is a multipurpose instruction used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data input bit.

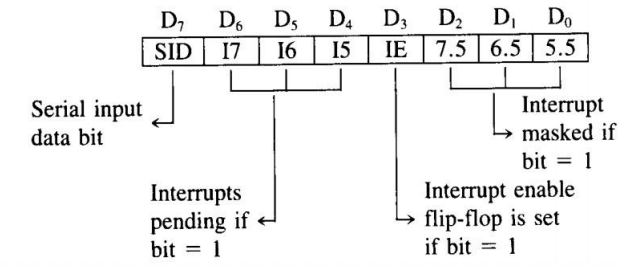

Fig. 1.4 SIM Instruction

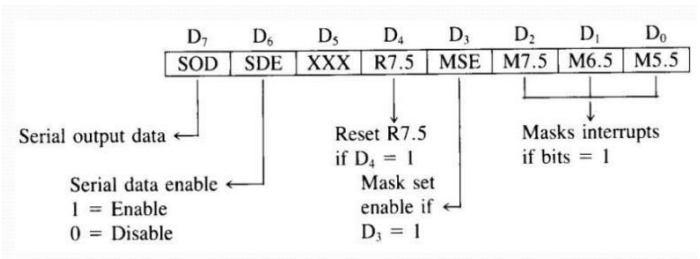

Fig. 1. 5 RIM Instruction

1.11 Addressing Modes

Every instruction of a program has to operate on a data. The method of specifying the data to be operated by the instruction is called Addressing. The 8085 has the following 5 different types of addressing.

- Immediate Addressing
- Direct Addressing
- Register Addressing
- Register Indirect Addressing
- Implied Addressing

Immediate Addressing

In immediate addressing mode, the data is specified in the instruction itself. The data will be a part of the program instruction.

EX. MVI B, 3EH - Move the data 3EH given in the instruction to B register; LXI SP, 2700H.

Direct Addressing

In direct addressing mode, the address of the data is specified in the instruction. The data will be in memory. In this addressing mode, the program instructions and data can be stored in different memory.

EX. LDA 1050H - Load the data available in memory location 1050H in to accumulator; SHLD 3000H

Register Addressing

In register addressing mode, the instruction specifies the name of the register in which the data is available.

EX. MOV A, B - Move the content of B register to A register; SPHL; ADD C.

Register Indirect Addressing

In register indirect addressing mode, the instruction specifies the name of the register in which the address of the data is available. Here the data will be in memory and the address will be in the register pair.

EX. MOV A, M - The memory data addressed by H L pair is moved to A register. LDAX B.

Implied Addressing

In implied addressing mode, the instruction itself specifies the data to be operated. EX. CMA - Complement the content of accumulator; RAL

1.12 Timing Diagrams

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. Each machine cycle is composed of many clock cycles. Since, the data and instructions, both are stored in the memory, the μP performs fetch operation to read the instruction or data and then execute the instruction. The 3-statussignals: IO / M', S1 and S0 are generated at the beginning of each machine cycle.The unique combination of these 3-status signals identifies read or write operation and remain valid for the duration of the cycle. Thus, time taken byany 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. Each machine cycle consists of many clock periods/ cycles, called T-states.

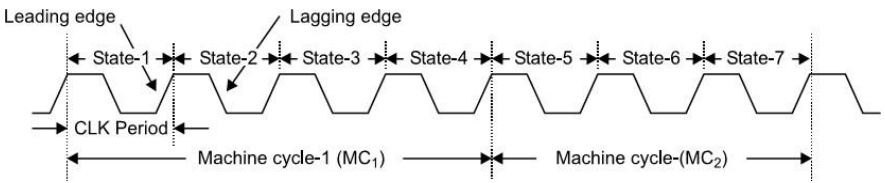

Fig. 1.6 Machine cycle showing clock periods

Each and every operation inside the microprocessor is under the control of the clock cycle. The clock signal determines the time taken by the microprocessor to execute any instruction. State is defined as the time interval between 2-trailing or leading edges of the clock. Machine cycle is the timerequired to transfer data to or from memory or I/O devices.

The 8085 microprocessor has 5 basic machine cycles. They are

- Opcode fetch cycle (4T)
- Memory read cycle (3 T)
- Memory write cycle (3 T)
- I/O read cycle (3 T)
- I/O write cycle (3 T)

Processor Cycle

The function of the microprocessor is divided into fetch and execute cycle of any instruction of a program. The program is nothing but number ofinstructions 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. Thus, 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.

Instruction Cycle (IC) = Fetch cycle (FC) + Execute Cycle (EC)

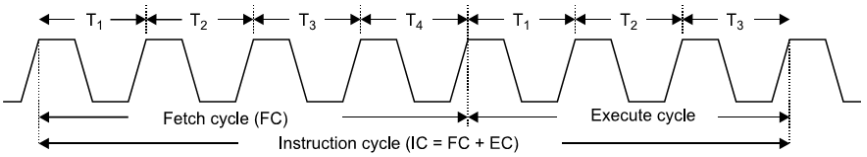

Fig. 1.7 Processor cycle

The 1st machine cycle of any instruction is always an Opcode 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.

In the opcode fetch cycle, the processor comes to know the nature of the instruction to be executed. The processor during (M1 cycle) puts the program counter contents on the address bus and reads the opcode of the instruction through read process. The T1, T2, and T3 clock cycles are used for the basic memory read operation and the T4 clock and beyond are used for its interpretation of the opcode. Based on these interpretations, the μP comes to know the type of additional information/data needed for the execution of the instruction and accordingly proceeds further for 1 or 2-machine cycle of memory read and writes.

Instruction Fetch (FC)⇒An instruction of 1 or 2 or 3-bytes is extracted from the memory locations during the fetch and stored in the μP's instruction register.

Instruction Execute (EC)⇒The instruction is decoded and translated into specific activities during the execution phase.

Opcode Fetch

The 1st step in communicating between the microprocessor and memory is reading from the memory. This reading process is called opcode fetch. The

process of opcode fetch operation requires minimum 4-clock cycles T1, T2, T3, and T4and is the 1st machine cycle (M1) of every instruction. In order to differentiate between the data byte pertaining to an opcode or an address, the machine cycle takes help of the status signal IO/ M, S1, and S0. The IO/ M= 0 indicates memory operation and S1 = S0 = 1 indicates Opcode fetch operation. The opcode fetch machine cycle M1 consists of 4-states (T1, T2, T3, and T4). The 1st 3-states are used for fetching (transferring) the byte from the memory and the 4th-state is used to decode it.

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 Figs. 5.3 (a), (b), and (c). The μP 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 μP sends a high on status signal S1 and S0 indicating fetch operation.
- The μP sends 16-bit address. AD bus has address in 1st clock of the 1st machine cycle, T1.
- AD7to 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 T3the RDsignal becomes high and memory is disabled.

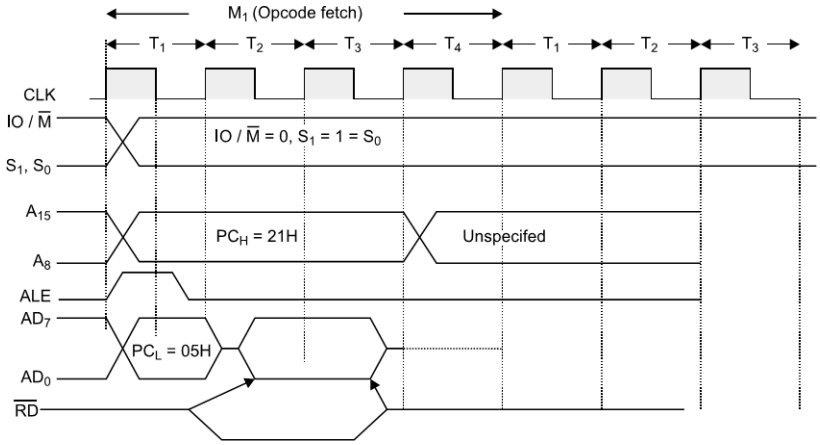

Fig. 1.8 Opcode Fetch

- During T4 the opcode is sent for decoding and decoded in T4.
- The execution is also completed in T4if the instruction is single byte.
- More machine cycles are essential for 2- or 3-byte instructions. The 1st machine cycleM1is meant for fetching the opcode. The machine cycles M2and M3are required either to read/ writedata or address from the memory or I/O devices.

Memory and I/O Read Cycle

The memory read machine cycle is executed by the processor to read a data byte from memory. The processor takes 3T states to execute this cycle. The instructions which have more than one-byte word size will use the machine cycle after the opcode fetch machine cycle.

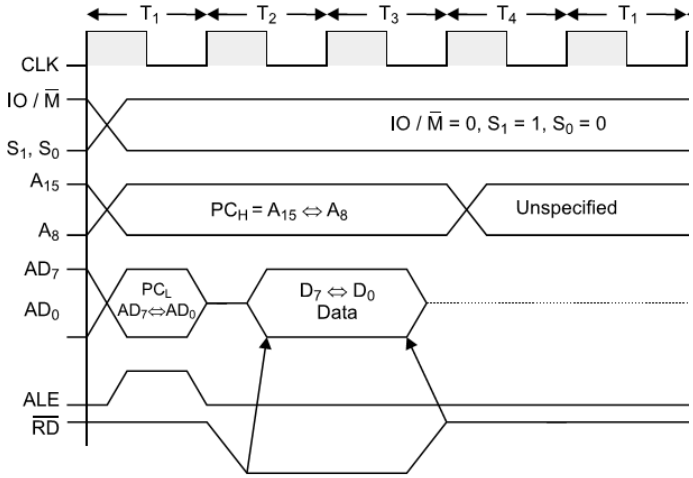

Fig. 1.9 Memory Read Cycle

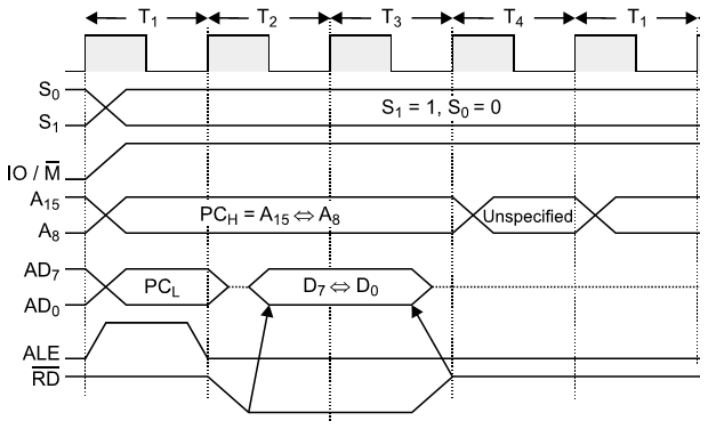

Fig. 1.10 I/O Read Cycle

The high order address (A15 ⇔A8) and low order address (AD7 ⇔AD0) are asserted on 1st low going transition of the clock pulse. The timing diagram for IO/M read are shown in Fig. The A15 ⇔A8 remains valid in T1, T2, and T3 i.e. duration of the bus cycle, but AD7⇔AD0 remains valid only in T1. Since it has to remain valid for the whole bus cycle, it must be saved for its use in the T2 and T3. ALE is asserted at the beginning of T1 of each bus cycle and is negated towards the end of T1. ALE is active during T1 only and is used as the clock pulse to latch the address (AD7⇔AD0) during T1. The RD' is asserted near the beginning of T2. It ends at the end of T3. As soon as the RD' becomes active, it forces the memory or I/O port to assert data. RD' becomes inactive towards the end of T3, causing the port or memory to terminate the data.

Memory and I/O Write Cycle

Immediately after the termination of the low order address, at the beginning of the T2, data is asserted on the address/data bus by the processor. WR' control is activated near the start of T2 and becomes inactive at the end of T3. The processor maintains valid data until after WR' is terminated. This ensures that the memory or port has valid data while WR' is active. It is clear from figures that for READ bus cycle, the data appears on the bus as a result of activating RD' and for the WR' bus cycle, the time the valid data is on the bus overlaps the time that the WR' is active.

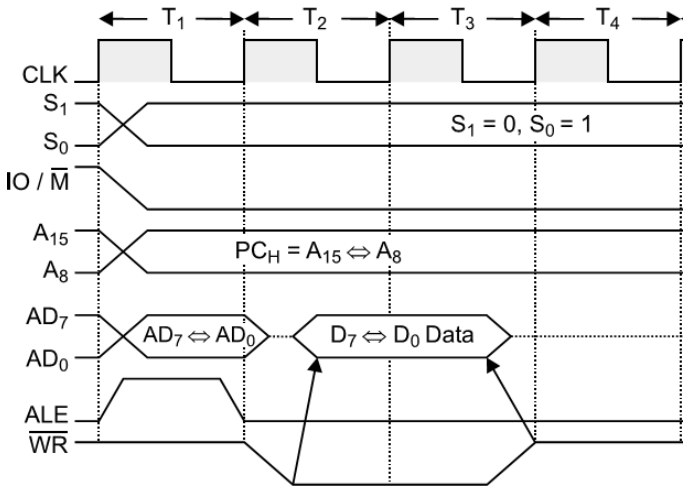

Fig. 1.11 Memory Write Cycle

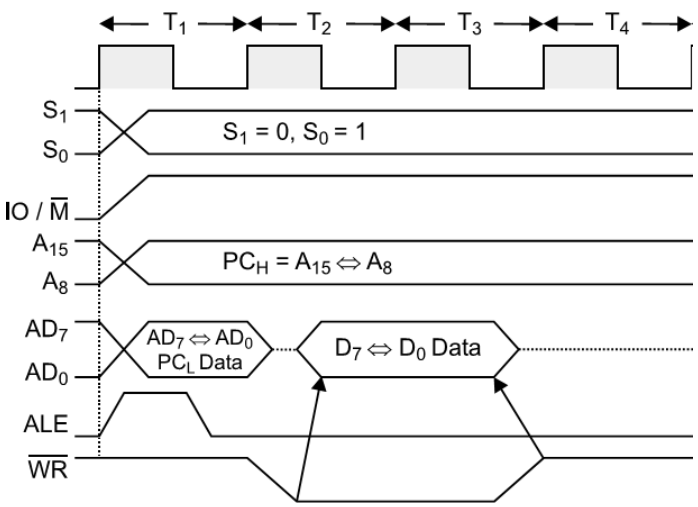

Fig. 1.12 I/O Write Cycle

Examples

Opcode fetch MOV B, C.

T1: The 1st clock of 1st machine cycle (M1) makes ALE high indicating address latch enabled which loads low-order address 00H on AD7⇔AD0 and high-order address 10H simultaneously on A15 ⇔A8. The address 00H is latched in T1.

T2: During T2 clock, the microprocessor issues RD control signal to enable the memory and memory places 41H from 1000H location on the data bus.

T3: During T3, the 41H is placed in the instruction register and RD= 1 (high) disables signal. It means the memory is disabled in T3 clock cycle. The opcode cycle is completed by end of T3 clock cycle.

T4: The opcode is decoded in T4 clock and the action as per 41H is taken accordingly. In other word, the content of C-register is copied in B-register.

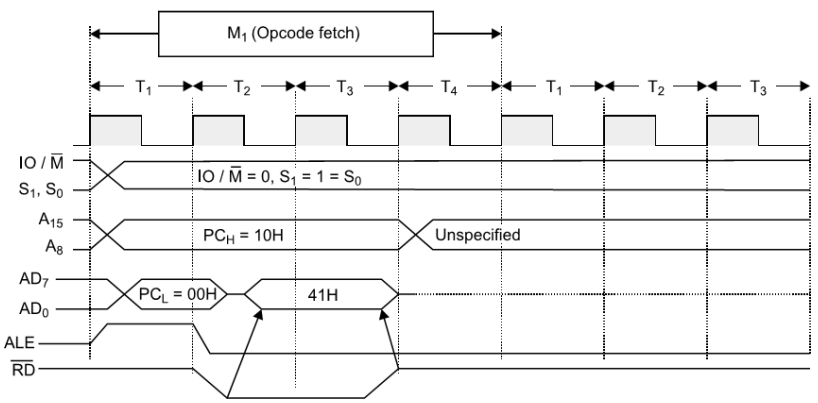

Fig. 1.13 Opcode Fetch (MOV B, C)

Timing diagram for STA 526A$_H$

- STA means Store Accumulator -The contents of the accumulator is stored in the specified address (526A).
- The opcode of the STA instruction is said to be 32H. It is fetched from the memory 41FFH (see fig). - *OF machine cycle*
- Then the lower order memory address is read (6A). - *Memory Read Machine Cycle*
- Read the higher order memory address (52). -*Memory Read Machine Cycle*
- The combination of both the addresses are considered and the content from accumulator is written in 526A. - *Memory Write Machine Cycle*

- Assume the memory address for the instruction and let the content of accumulator is C7H. So, C7H from accumulator is now stored in 526A.

Address	Mnemonics	Opcode
41FF	STA $526A_H$	32_H
4200		$6A_H$
4201		52_H

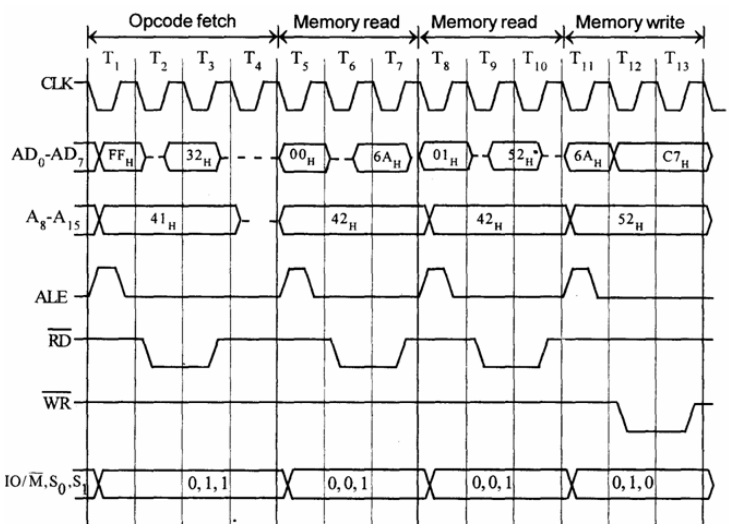

Fig. 1.14 Timing Diagram for STA $526A_H$

Timing diagram for IN $C0_H$

- Fetching the Opcode DB_H from the memory 4125_H.
- Read the port address $C0_H$ from 4126_H.
- Read the content of port $C0_H$ and send it to the accumulator.
- Let the content of port is $5E_H$.

Address	Mnemonics	Opcode
4125	IN $C0_H$	DB_H
4126		$C0_H$

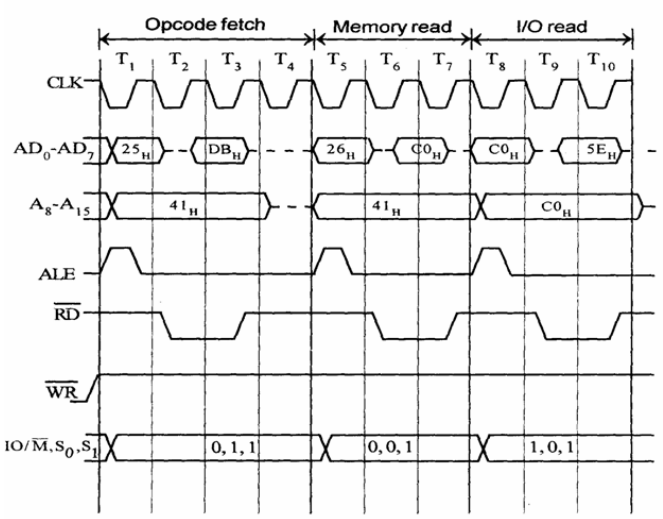

Fig. 1.15 Timing Diagram for IN $C0_H$

1.13 Assembly Language Programming

An assembly language is a low-level programming language for a computer, or other programmable device, in which there is a very strong (generally one- to-one) correspondence between the language and the architecture's machine code instructions. Each assembly language is specific to a particular computer architecture, in contrast to most high-level programming languages, which are generally portable across multiple architectures, but require interpreting or compiling.

Assembly language is converted into executable machine code by a utility program referred to as an assembler; the conversion process is referred to as assembly, or assembling the code.

Assembly language uses a mnemonic to represent each low-level machine operation or opcode. Some opcodes require one or more operands as part of the instruction, and most assemblers can take labels and symbols as operands to represent addresses and constants, instead of hard coding them into the program.

What is an Assembler?

An assembler is a software tool - a program -- designed to simplify the task of writing computer programs. If you have ever written a computer program directly in a machine-recognizable form such as binary or hexadecimal code, you will appreciate the advantages of programming in a symbolic assembly language.

Assembly language operation codes (opcodes) are easily remembered (MOV for move instructions, JMP for jump). You can also symbolically express addresses and values referenced in the operand field of instructions. Since you assign these names, you can make them as meaningful as the mnemonics for the instructions. For example, if your program manipulates a date as data, you can assign it the symbolic name DATE. If your program contains a set of instructions used as a timing loop (a set of instructions executed repeatedly until a specific amount of time has passed), you can name the instruction group TIMER.

What the Assembler Does

To use the assembler, you first need a source program. The source program consists of programmer written assembly language instructions. These instructions are written using mnemonic opcodes and labels. Assembly language source programs must be in a machine-readable form when passed to the assembler. TheIntellec development system includes a text editor that will help you maintain source programs as paper tape files or diskette files. You can then pass the resulting *source program file* to the assembler. The assembler program performs the clerical task of translating symbolic code into *object code*which can be executed by the 8080 and 8085 microprocessors. Assembler output consists of three possible files: the *object file*containing your program translated into object code; the *list file* printout of your source code, the assemble generated object code, and the symbol table; and the *symbol-crass- reference file,* a listing of the symbol-cross reference records.

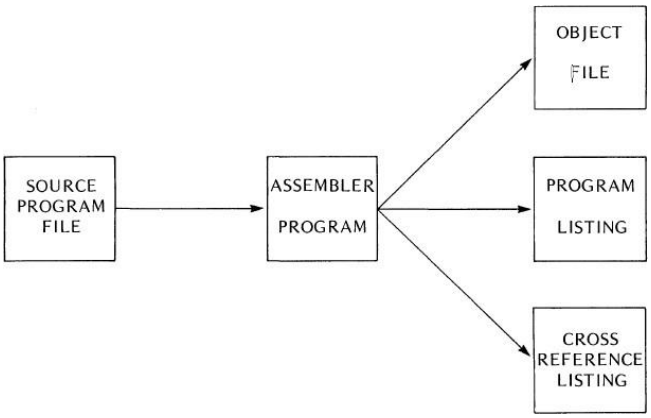

Fig. 1.16 Function of an Assembler

Example Programs

1. Statement: Store the data byte 32H into memory location 4000H.

Program 1

```
MVI A, 32H      : Store 32H in the accumulator
STA 4000H       : Copy accumulator contents at address 4000H
HLT             : Terminate program execution
```

Program 2

```
LXI H           : Load HL with 4000H
MVI M           : Store 32H in memory location pointed by HL register pair (4000H)
HLT             : Terminate program execution
```

Statement: Exchange the contents of memory locations 2000H and 4000H

Program 1

```
LDA 2000H       : Get the contents of memory location 2000H into accumulator
MOV B, A        : Save the contents into B register
LDA 4000H       : Get the contents of memory location 4000H into accumulator
STA 2000H       : Store the contents of accumulator at address 2000H
MOV A, B        : Get the saved contents back into A register
STA 4000H       : Store the contents of accumulator at address 4000H
```

Program 2

```
LXI H 2000H     : Initialize HL register pair as a pointer to memory location
                  2000H.
```

LXI D 4000H : Initialize DE register pair as a pointer to memory location 4000H.
MOV B, M : Get the contents of memory location 2000H into B register.
LDAX D : Get the contents of memory location 4000H into A register.
MOV M, A : Store the contents of A register into memory location 2000H.
MOV A, B : Copy the contents of B register into accumulator.
STAX D : Store the contents of A register into memory location 4000H.
HLT : Terminate program execution.

Sample problem

(4000H) = 14H
(4001H) = 89H
Result = 14H + 89H = 9DH

Source program

LXI H 4000H : HL points 4000H
MOV A, M : Get first operand
INX H : HL points 4001H
ADD M : Add second operand
INX H : HL points 4002H
MOV M, A : Store result at 4002H
HLT : Terminate program execution

Statement: Subtract the contents of memory location 4001H from the memorylocation 2000H and place the result in memory location 4002H.
Program - 4: Subtract two 8-bit numbers

Sample problem

(4000H) = 51H
(4001H) = 19H
Result = 51H - 19H = 38H

Source program

LXI H, 4000H : HL points 4000H

```
MOV A, M    : Get first operand
INX H       : HL points 4001H
SUB M       : Subtract second operand
INX H       : HL points 4002H
MOV M, A    : Store result at 4002H.
HLT         : Terminate program execution
```

Applications of Microprocessor in General Life

There are a lot of applications of Microprocessor in general life. Some of the applications are given below

- Mobile Phones
- Digital Watches
- Washing Machine
- Computer
- Lighting Control
- Traffic Control
- LAPTOP
- Modems
- Power Stations
- Television
- CD Player
- Multimeter
- CRO
- Wave generator
- More applications in medical

2