Unit I - Microprocessor Architecture and Interfacing

INTRODUCTION

- The Microprocessor is the central processing unit (CPU) of a computer.
- A microprocessor is a multipurpose, programmable, clock-driven, register- based electronic device.
- It reads binary instructions from storage device (memory).
- It accepts binary data as input and processes data according to those instructions, and provides results as output.
- The microprocessor operates in binary digits, 0 and 1, also known as **bits**.
- Each microprocessor recognizes and processes a group of bits called **word**.

Classification of Microprocessors:

Based on their word length: 8-bit and 16-bit microprocessor. (if a processor with an 8-bit (16-bit) word is known as an 8-bit (16-bit) microprocessor).

Based on the application: General purpose, and special purpose processors.

Based on the architecture and hardware: RISC(reduced instruction set computers), CISC(complex instruction set computers), VLIW(Very Long Instruction Word)

Note:

- **Bit:** a binary digit (0 or 1).
- **Word**: a group of bits.
- **Byte**: a group of 8-bits.
- **Nibble**: a group of 4 bits.
- **Kilobyte:** a collection of 1024 bytes (2_{10} bytes).
- **Megabyte:** a collection of 1024 kilobytes (2_{20} bytes).
-

HARDWARE ARCHITECTURE OF 8085 MICROPROCESSOR

The architecture of 8085 microprocessor includes the ALU (Arithmetic/Logic Unit), timing and control unit, instruction register and decoder, register array, interrupt control, and serial I/O control (shown in fig.1).

ARITHMETIC LOGIC UNIT (ALU)

- The arithmetic / logic unit performs the computing functions; it includes the accumulator, temporary register, arithmetic and logic circuits, and five flags.
- The temporary register is used to hold data during an arithmetic / logic operation.
- The result is stored in the accumulator, and the flags (flip-flops) are set or reset according to the result of the operation.

REGISTER ARRAY

- The 8085 microprocessor has six general-purpose registers to store 8-bit data; these are identified as B, C, D, E, H, and L (as shown in the figure 1).

- The programmer can use these registers to store or copy data into the registers by using data-copy instructions.
- Two additional registers, called temporary registers W and Z, are included in the register array. These registers are used to hold 8-bit data during the execution of some instructions. However, because they are used internally, they are not available to the programmer.

ACCUMULATOR

It is an 8-bit register, which is a part of arithmetic/logic unit (ALU). It is used to store 8-bit data and to perform arithmetic and logical operations. The result of an operation is stored in the accumulator. It is also identified as register A.

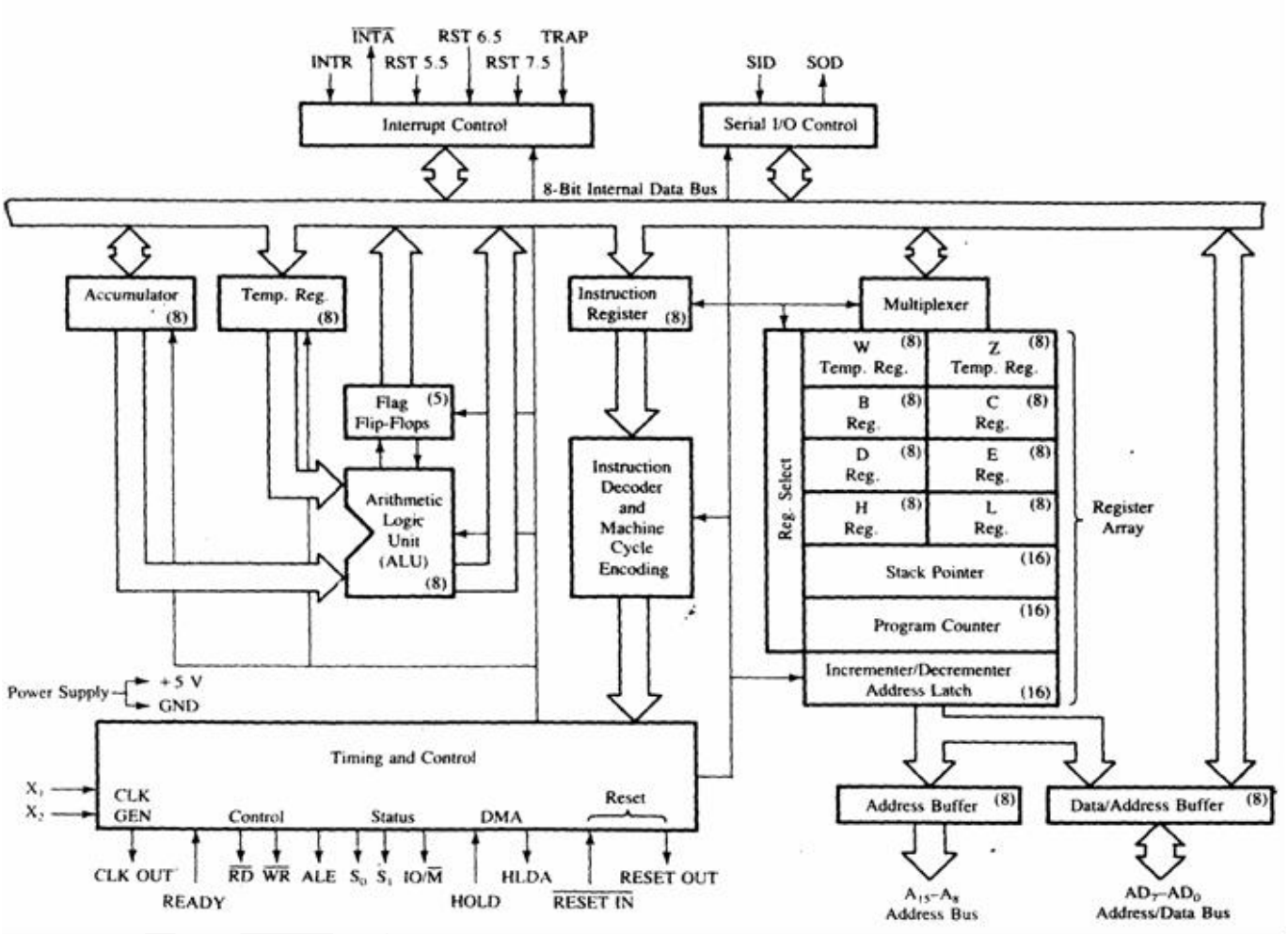
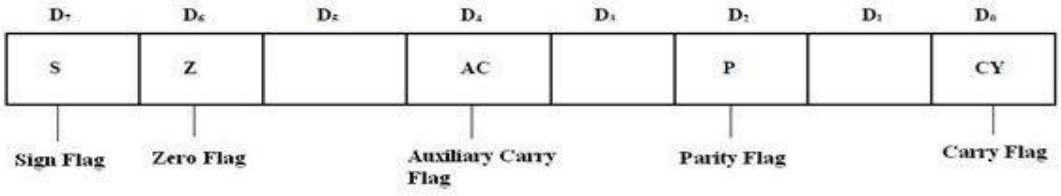


Figure 1: Hardware Architecture of 8085 Microprocessor

FLAGS

The ALU includes 5 flip-flops, which are set or reset after an operation according to data conditions of the result in the accumulator and other registers. They are called Zero (Z), Carry (CY), Sign(S), Parity (P), Auxiliary Carry (AC) flags; the bit positions in the flag register are shown in figure 2. The most



commonly used flags are zero, carry, and sign. the microprocessor uses these flags to test the data conditions.

1.Sign flag(S)

- After execution of an arithmetic or logic operation, if bit D7 (shown in figure.2) of the result is 1, the sign flag is set. This flag is used with signed numbers.
- In a given byte, if D7 is 1, the number will be viewed as a negative number; if it is 0, the number will be consider positive.
- In arithmetic operations with singed numbers, bit D7 is reserved for indicating the sign, and the remaining seven bits are used to represent the magnitude of a number.
- However, this flag is irrelevant for the operations of unsigned numbers. Therefore, for unsigned numbers, even if bit D7 of a result is 1and the flag is-set, it does not mean the result is negative.

2. Zero flag(Z):

The Zero flag is set if the ALU operation results in 0, and the flag is reset if the result is not 0.

3. Auxiliary Carry flag(AC)

- In an arithmetic operation, when a carry is generated by digit D3 and passed on to digit D4, the AC flag is set.
- The flag is used only internally for BCD (binary-coded decimal) operations and is not available for the programmer to change the sequence of a program with a jump instruction.

4. Parity flag(P)

After an arithmetic or logical operation, if the result has an even number of 1s, the flag is set. If it has an odd number of 1s, the flag is reset.

5. Carry flag(CY)

- If an arithmetic operation results in a carry, the Carry flag is set; otherwise it is reset.
- The Carry flag also serves as a borrow flag for subtraction.

PROGRAM COUNTER (PC)

- This 16-bit register deals with sequencing the execution of instructions.
- This register is a memory pointer. Memory locations have 16-bit addresses, and that is why this is a16-bit register.
- The microprocessor uses this register to sequence the execution of the instructions.
- The function of the program counter is to point to the memory address from which the next byte is to be fetched.
- When a byte (machine code) is being fetched, the program counter is incremented by one to point to the next memory location

STACK POINTER (SP)

- The stack pointer is also a 16-bit register used as a memory pointer.
- It points to a memory location in R/W memory, called the stack. The beginning of the stack is defined by loading 16-bit address in the stack pointer.

INSTRUCTION REGISTER AND DECODER

The instruction register and the decoder are part of the ALU. When an instruction is fetched from memory, it is loaded in the instruction register. The decoder decodes the instruction and establishes the sequence of events to follow. The instruction register is not programmable and cannot be accessed through any instruction.

TIMING AND CONTROL UNIT

This unit synchronizes all the microprocessor operations with the clock and generates the control signals necessary for communication between the microprocessor and peripherals. The control signals are similar to a sync pulse in an oscilloscope. The RD and WR signals are sync pulses indicating the availability of data on the data bus.

PIN CONFIGURATION OF 8085 MICROPROCESSOR (Pin Outs - Signals) :

The 8085 microprocessor is available on a 40-pin Dual-in-Line Package (DIP). The pin configuration is shown in Figure 3.

All the signals are classified into six groups

1) Address bus

2) Data bus

3) Control and status signals

4) Power supply and frequency signals

5) Externally initiated signals ,

6) Serial I/O ports.

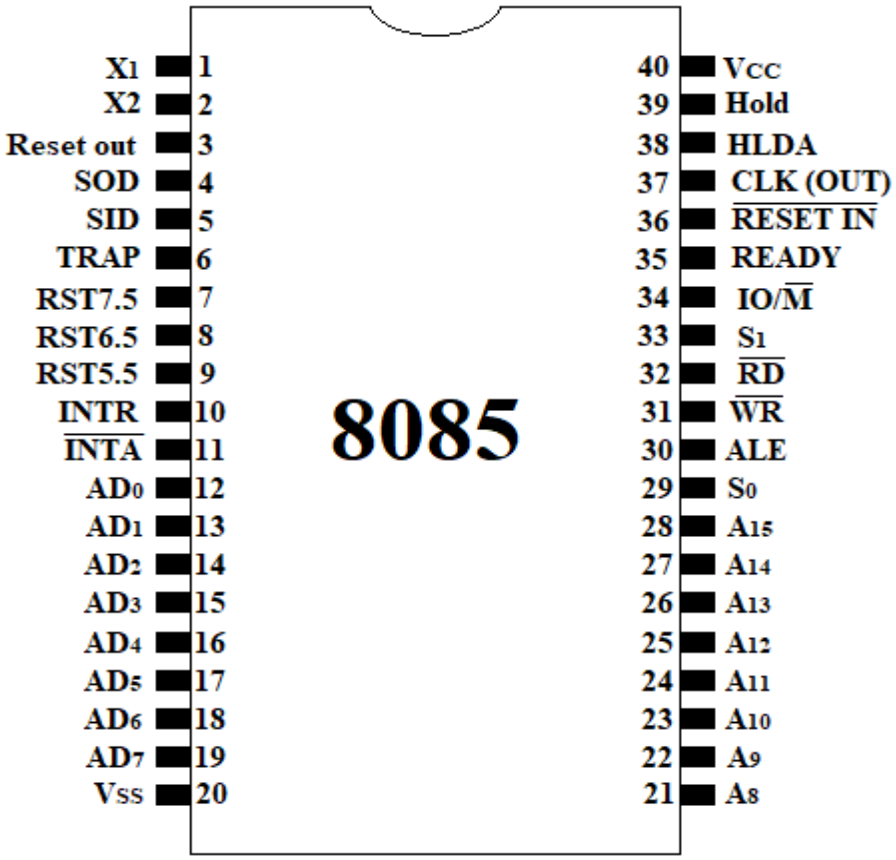


Figure 3: Pin configuration of 8085 microprocessor

1) Address bus (Pin 12-19, & 21-28)**:**

- ✓ The 8085 has 16 address signal lines /pins that are used as the address bus; however, these lines are split into two segments: A15 – A8 and AD7 –AD0.
- ✓ The eight signal lines, A15 – A8, are unidirectional and used for high-order address, of 16-bit address
- ✓ The signal lines AD7 –AD0 are used for a dual purpose.

2) Data bus / multiplexed address bus (Pin 12-19)**:**

- ✓ The signal lines AD7 –AD0 are bidirectional: they serve a dual purpose.
- ✓ They are used as the low-order address bus as well as the data bus.

3) Control and status signals

I. ALE – Address Latch Enable (Pin 30)**:**

- ✓ This is a positive going pulse generated every time the 8085 begins an operation; it indicates that the bits on AD7 –AD0 are address bits.
- ✓ This signal is used primarily to latch the low-order address from the multiplexed bus and generate a separate set of eight address lines, A7 –A0

II. RD – Read (Pin 32)**:**

- ✓ This is a Read control signal (active low).
- ✓ This signal indicates that the selected I/O or memory device is to be read and data are available on the data bus.

III. WR – Write (Pin 31)**:**

- ✓ This is a Read control signal (active low).
- ✓ This signal indicates that the data on the data bus are to be written into a selected memory or I/O location.

IV. IO/M (Pin 34)**:**

- ✓ This is a status signal used to differentiate between I/O and memory operations
- ✓ When it is high, it indicates an I/O operation.
- ✓ When it is low, it indicates a memory location.
- ✓ This signal is combined with **RD** and**WR** to generate I/O and memory control signals.

V. S0 and S1(Pin 29&33)**:**

- ✓ This status signals, similar to **IO/** , can identify various operations but they are rarely used in small systems.

4) Power supply and clock frequency

a) **X1, X2**(Pin 1&2)**:** A crystal (or RC, LC network) is connected at these two pins. The frequency is internally divided by two ; therefore , to operate a system at 3 MHz , the crystal should have a frequency of 6 MHz

b) **GND** (Pin 20)**:**Ground reference

c) **Vcc** (Pin 40)**:** +5v power supply

d) **CLK (OUT) – clock output**(Pin 37)**:** this signal can be used as the system clock for other devices.

5) Externally initiated signals

The 8085 has five interrupt signals that can be used to interrupt a program execution

i. **INTR** (Pin 10)**:**Interrupt Request

ii. **INTA** (Pin 11)**:**Interrupt Acknowledge: used to acknowledge an interrupt

iii. **RST 7.5, 6.5, 5.5 Restart Interrupts** (Pin7-9)**:** vectored interrupts that the program controls to specific memory locations. They have higher priorities than the INTR interrupt

iv. TRAP (Pin 6)**:** nonmaskable interrupt and highest priority

v. HOLD (Pin39)**:–:** This signal indicates that a peripheral such as DMA (Direct Memory Access) controller is requesting the use of the address and data bus.

vi. HLDA- Hold Acknowledge (Pin 38)**:–** This signal acknowledges the HOLD request.

vii. READY(Pin 35)**:** This signal is used to delay the microprocessor Read or Write cycles until a slow-responding peripheral is ready to send or accept data. When the signal goes low, the microprocessor waits for an integral number of clock cycles until it goes high.

viii. RESET: when reset pin is active, all internal operations suspended PC is cleared. Now the program execution can again begin at the zero memory address.

a. RST IN (Pin 36)**:** when the signal on this pin goes low, the program counter is set to zero, the buses are tri-stated, and the MPU is reset. **b. RST OUT** (Pin 3)**:** This signal indicates that the MPU is being rest. the signal can be used to reset other devices.

6) Serial I/O ports (Pin 4&5)**:**

The 8085 has two signals to implement the serial transmission:

1. **SID** (Serial Input Data) **2. SOD** (Serial Output Data)

Instruction Cycle

An instruction is a command given to the computer to perform a specified operation on given data. To perform a particular task a programmer writes a sequence of instructions, called a program. Program and data are stored in the memory. The CPU fetches one instruction from the memory at a time and executes it. It executes all the instructions of a program one by one to produce to the final result.

The necessary steps that a CPU carries out to fetch an instruction and necessary data from the memory, and to execute it, constitute an instruction cycle. An instruction cycle consists of a fetch cycle and executes cycle, In fetch cycle a CPU fetches opcode (the machine code of an instruction) from the memory. The necessary steps which are carried out to fetch an opcode from the memory, constitute a fetch cycle. The necessary steps which are carried out to get data, if any, from the memory and to perform the specific operation specified in an instruction, constitute an execute cycle.

IC=FC+EC

Fetch operation

The 1st byte of an instruction is its opcode. An instruction may be more than one byte long. The other bytes are data or operand address. The program counter (PC) keeps the memory address of the next instruction to be executed. In the beginning of a fetch cycle the content of the program counter, which is the address of the memory location where opcode is available, is sent to the memory. The memory places the opcode on the data bus so as transfer it to the CPU. The entire operation of fetching an opcode takes three clock cycles. A slow memory may take more time. In case of a slow memory the CPU has to wait till the memory sends the opcode. The clock cycle for which the CPU waits is called wait cycle. Most of the CPUs have been designed to introduce wait cycles to cope with slow memories.

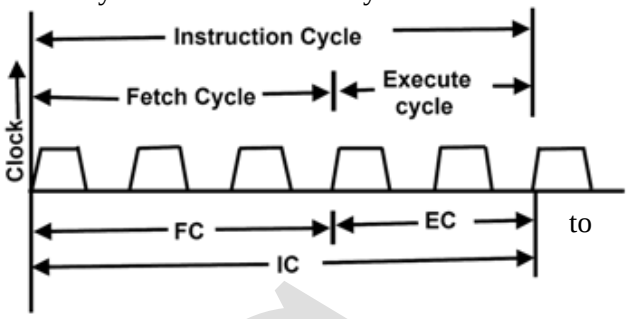


Figure (a) Instruction cycle showing FC, EC and IC

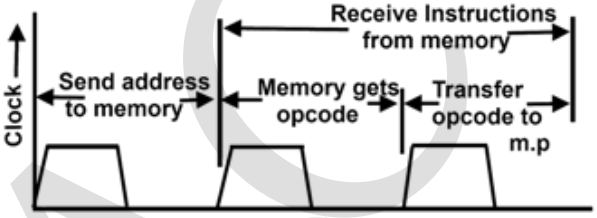


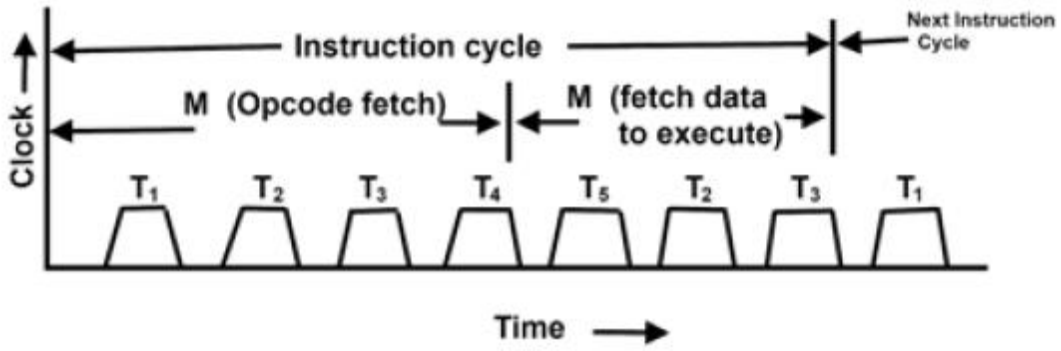
Figure (b) A Typical Fetch Cycle

Execute operation

The opcode fetched from the memory goes to the data register (DR) (data/address buffer in Intel 8085) and then to instruction register(IR). From the instruction register it goes to the decoder circuitry which decodes the instruction. The decoder circuitry is within the microprocessor. After the instruction is decoded, execution begins. If the operand is in the general purpose registers, execution is immediately performed. The time taken in decoding and execution is one clock cycle. If an instruction contains data or operand address which are still in the memory, the CPU has to perform some read operations to get the desired data. After receiving the data it performs execute operation. A read cycle is similar to a fetch cycle. In case of a read cycle the quantity received from the memory are data or operand address instead of an opcode. In some instructions write operation is performed. In write cycle data are sent from the CPU to the memory or an output device. Thus we see that in some cases an execute cycle may involve one or more read or write cycles or both. Figs.4 shows an instruction and fetch cycle respectively.

Machine Cycle

An instruction cycle consists of one or more machine cycles as shown in Figure. This figure is for MVI instruction. A machine cycle consists of a number of clock cycles. One clock cycle is known as state.



TIMING DIAGRAM

The graphical representation of the instruction execution in steps with respect to the time (clock signal) is called timing diagram.

Instruction Cycle : It is defined as the time required completing the execution of an instruction. The 8085instruction cycle has one to six machine cycles or one to six operations.

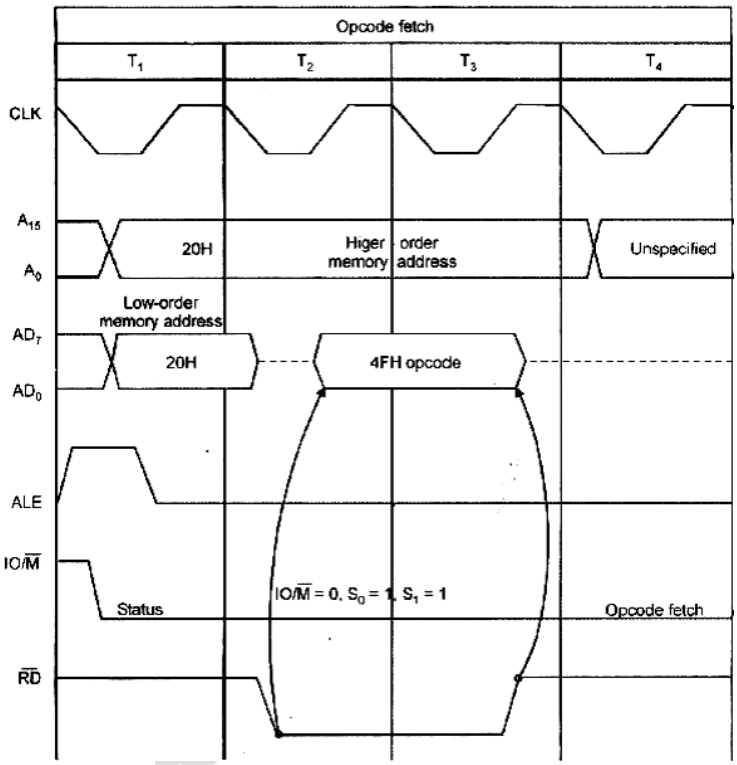
Machine Cycle : It is the time required to complete one operation of accessing the memory, I/O or Acknowledging an external request. This cycle may consist of three to six T-states.

T-State : It is defined as one subdivision of the operation performed in one clock period. Each T-state is exactly equal to one clock period.

Timing diagram for:

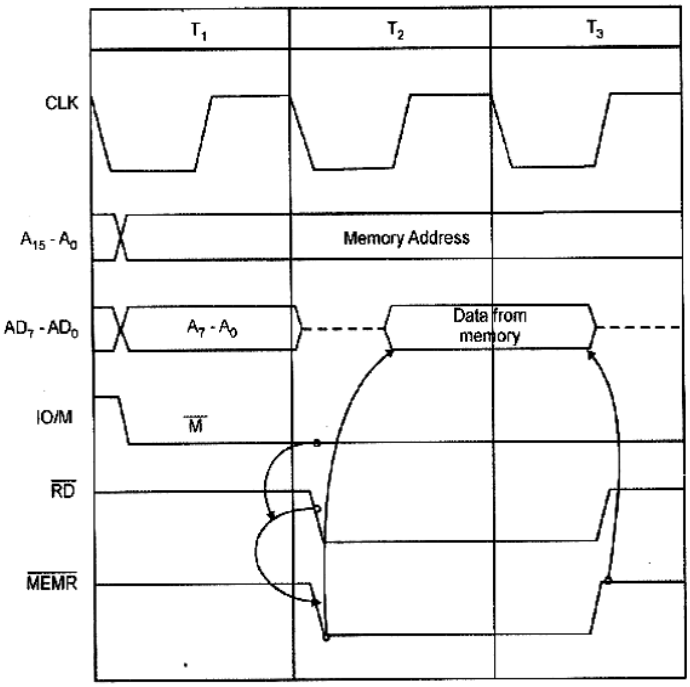
1. **Instruction Fetch, 2. Memory Read, 3. Memory Write, 4. Out Instruction, 5. IN Instruction**

1.**Timing diagram for Instruction Fetch**



- This instruction requires 4 or 6 clock periods.
- Timing diagram for instruction fetch operation is shown in Fig.
- The purpose of this instruction is to read the contents of the memory location containing an instruction addressed by the program counter to place it in the instruction register.
- The 8085 puts a LOW on the IO/M' line of the system bus, indicating memory operation.
- The 8085 sets S0=1 and S1=1 on the system bus indicating that it is a memory fetch instruction.
- The 8085 places the program counter high byte on the higher order address bus and the program counter low byte on the lower order address bus.
- 8085also sets the ALE signal HIGH.
- As soon as ALE goes LOW, the progran1counter low byte is latched by some supporting devices.
- At the beginning of T2 in MC1, the 8085puts the RD' line to LOW, indicating a READ operation.
- After sometime, 8085 loads the Opcode into the instruction register.
- During the T4 clock period in MC1, the 8085decodesthe instruction.

2. Timing diagram for Memory Read

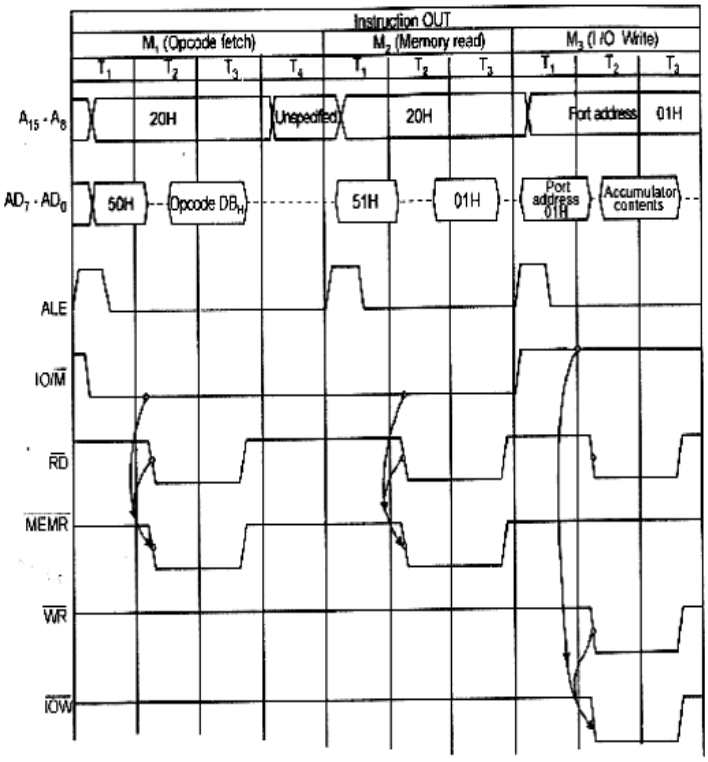


- The purpose of this instruction is to read the contents of the memory location.
- Timing diagram for memory READ operation is shown in Fig.2.
- The status of S0=0 and S1=1, indicating a memory READ operation.

- The 8085 places the contents of the high byte of the memory address register, such as the contents of the H register, on the higher order address lines.
- The 8085places the contents of the low byte of the memory address register, such as the contents of the L register, on the lower order address lines.
- The 8085 sets ALE to HIGH, indicating the beginning of MC2 As soon as ALE goes to LOW, the memory chip must latch the low byte of the address lines since the same lines are going to be used as data lines.
- The 8085 puts the RD signal to LOW, indicating a memory READ operation.
- The external logic gets the data from the memory location addressed by the memory address register, such as H, L pair and places the data into the register

3. **Timing diagram for Memory Write**

- The purpose of this instruction is to store the contentsof8085 register, such as accumulator, into a memory location addressed by a register Pair such as HL pair.
- Timing diagram for memory WRITE operation is shown in Fig.
- The status of S0=I and S1=0 and WR' =0 indicating a memory WRITE operation.
- The 8085 uses machine cycle MC1 to fetch and decode the instruction. It then performs WRITE operation in MC2.
- The 8085 places the contents of the high byte of the memory address register, such as the contents of the H register, on the higher order address lines.

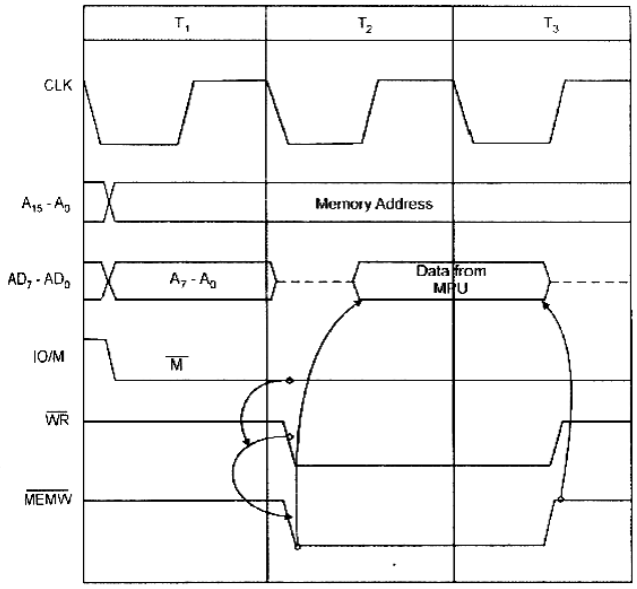


- The 8085places the contents of the low byte of the memory address register, such as the contents of the L register, on the lower order address lines.
- The 8085 sets ALE to HIGH, indicating the beginning of MC2 As soon as ALE goes to LOW, the memory chip must latch the low byte of the address lines since, the same lines are going to be used as data lines.
- The 8085 puts the WR signal to LOW, indicating a memory WRITE operation.
- The external logic gets the data from the lower order address lines and stores the data in the memory location addressed by the memory address register, such as HL pair.

4. **Timing diagram for Out Instruction**

- Timing diagram for OUT instruction is shown in Fig.

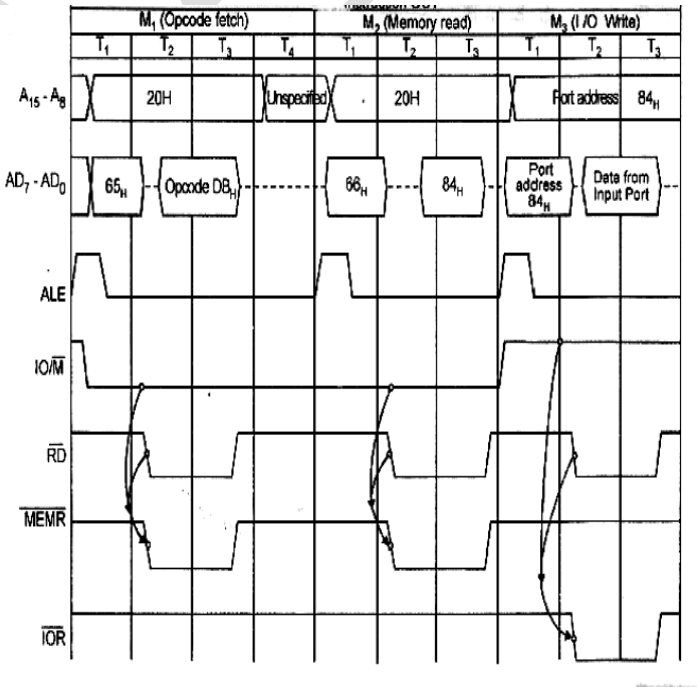
- In the M1cycle (OPCODE FETCH), the 8085 places memory address on the address bus.
- At the same time, ALE goes high and IO/M' goes low.
- AtT2,the processor sends READ signal which is combined with MEMR' signal and it fetches the instruction to read the port address.
- In M2 (MEMORY READ), the 8085 places next address and gets the device address via data bus.
- In M3 (I/O WRITE), the 8085 places device address on the address bus. The IO/ M' goes HIGH indicating I/O operation.



- At T2, the accumulator contents are placed in the data bus, followed by a control signal WR'. By ANDing IO/M' and WR' signals, IOW, can be generated to enable the output device.

5. Timing diagram for IN Instruction

- Timing diagram for IN instruction is shown in Fig.
- MI, M2 are identical to that of OUT instruction.
- In the M3cycle, the 8085 places the address of the input port on the address bus and asserts the RD' signal, which is used to generate I/O READ signal.
- It enables the input port, and the data from the input port are placed on the data bus and transferred into the accumulator.
- At T2,the accumulator contents are placed in the data bus, followed by a control signal WR'. By ANDing IO/M' and RD' signals, IOW' can be generated to enable an Input device.



Instruction Set of Intel 8085:

An instruction is a command given to the computer to perform a specified operation on given data. The instruction set of microprocessor is the collection of the instructions that the microprocessor is designed to execute. These instructions have been classified into the following groups:

1. Data Transfer Group
2. Arithmetic Group
3. Logical Group

4. Branch Control Group
5. I/O and Machine Control Group

1. **Data transfer instruction:**
   This instruction performs the following operations.
   - Load on 8-bit number in a register,
   - Transfer data from one register to another register
   - Transfer data from memory to register or vice versa
   - Transfer between I/O and Accumulator.
   - Transfer between register and Stack memory

   Example:
   MOV B, A ; the content of the register B is copied into the register A, and the content of register B remains unaltered.

   LDA 2500 ; the content of the memory location 2500 is loaded into the accumulator. But the content of the memory location 2500 remains unaltered.

2. **Arithmetic instruction:**
   The instructions of this group perform arithmetic operations such as addition, subtraction, increment or decrement of the content of a register or memory.

   Example:
   ADD B ; add the content of a register (B) to the content of A register.
   SUB C ; subtract the content of a register (C) from the content of A.
   INR D ; increment the content of a register.
   DCR D ; decrement the content of a register.

3. **Logical and bit manipulation instruction:**
   The instructions under this group perform logical operation such as AND, OR, compare, rotate etc.
   Example:
   ANA B ; logically AND the contents of a register (B) with the content of A.
   ORA E ; logically OR the contents of a register (E) with the content of A.
   XORA B ; Exclusive-OR the contents of a register (B) with the content of A.
   CMP B ; compare the contents of register with the contents of A for less than, Equal to, or greater than.

4. **Branch instruction:**
   This group includes the instruction for conditional and unconditional jump, subroutine call and return, and restart. Examples are: JMP, JC, JZ, JNZ,CALL, CZ, RST, etc.

   JMP 2050H ; Change the program sequence to the specified address.
   JZ 2080H ; Change the program sequence to the specified address if the zero flag is set.

5. **Machine control instruction:**
   This group includes the instructions for input/output ports, stacks and machine control. Examples are: IN, OUT, PUSH, POP, HLT, etc.

   HLT ; stop processing and wait.

   NOP ; do not perform any operation.

Instruction and Data Formats:

Intel 8085 is an 8-bit microprocessor. It handles 8-bit data at a time. One byte consists of 8 bits. A memory location for Intel 8085 microprocessor is designed to accommodate 8 bit data. If 16 bit data are to be stored, they are stored in consecutive memory locations. The address of a memory location is of 16 bits i.e. 2 bytes.

The various techniques to specify data for instructions are:

i. 8 bit or 16 bit data may be directly given in the instruction itself.

ii. The address of the memory location, I/O port or I/O device, where data resides may be given in the instruction itself.

iii. In some instructions only one register is specified. The content of the specified register is one of the operands. It is understood that the other operand is in the accumulator.

iv. Some instructions specify two registers. The contents of the registers are the required data.

v. In some instructions data is implied. The most instructions of this type operate on the content of the accumulator.

Due to different ways of specifying data for instructions, the machine codes of all instructions are not of the same length.

There are three types of the Intel 8085 instruction as described below:

1. **One-byte Instructions:** Includes Opcode and Operand in the same byte

Examples:

Opcode	Operand	Binary Code	Hex Code
MOV	C, A	0100 1111	4FH
ADD	B	1000 0000	80H
HLT		0111 0110	76H

2. **Two-byte Instructions:** First byte specifies Operation Code Second byte specifies Operand

Examples:

Opcode	Operand	Binary Code	Hex Code
MVI	A, 32H	0011 1110 0011 0010	3EH 32H
MVI	B, F2H	0000 0110 1111 0010	06H F2H

3. **Three-byte Instructions :** First byte specifies Operation Code Second & Third byte specifies Operand

Examples:

Opcode	Operand	Binary Code	Hex Code
LXI	H, 2050H	0010 0001 0101 0000 0010 0000	21H 50H 20H
LDA	3070H	0011 1010 0111 0000 0011 0000	3AH 70H 30H

Addressing Modes:

Each instruction requires certain data on which it has to operate. It has already been explained that there are various techniques to specify data for instructions. These techniques are called addressing modes. Intel 8085 uses the following addressing modes:

1. Direct addressing
2. Register addressing
3. Memory addressing
4. Immediate addressing
5. Implicit addressing

1. Direct Addressing modes:

In this mode of addressing the address of the operand or data is given in the instruction itself.

Examples are:

1. STA 2400H Store the content of the accumulator in the memory location 2400H.
2. MVI B, 05 Store the data 05_H in to the register B.

2. Register Addressing modes:

In this mode the operand is located in one of the general purpose registers. The opcode specifies the address of the registers in addition to the operation to be performed. Examples are:

1. MOV A,B Move the content of register B to register A.
2. ADD B Add the content of register B to the content of register A.

3. Memory addressing modes

When a data byte/word is located in memory, its address needs to be determined to access the data and to perform the specified operation. The address of the operand may be

- Specified directly in the instruction.
- Given indirectly in the instruction.
- Mentioned in the instruction with respect to the contents of the other register.
- Given indirectly in the instruction but with respect to the contents of the other register.

These modes are described below.

i).Direct addressing mode:

In this mode, the memory address of the operand is specified as part of the instruction.

For example:

LDA 2400_H ; the content of the memory address 2400_H loaded to accumulator.

ii) Indirect Addressing:

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

Examples are:

1. LXI H, 2500H — Load H-L pair with 2500H.
   MOV A, M — Move the content of the memory location, whose address is in H-L pair (i.e. 2500H) to the accumulator
   HLT — Halt.

In the above program the instruction MOV A, M is an example of register indirect addressing. For this instruction the operand is in the memory. The address of the memory is not directly given in the instruction. The address of the memory resides in H-L pair and this has already been specified by an earlier instruction in the program, i.e. LXI H 2500H.

2. LXI H, 2500 H — Load the H-L pair with 2500H.
   ADD M — Add the content of the memory location, whose address is in H-L pair (i.e. 2500H0, to the content of the accumulator.
   HLT — Halt.

In this program the instruction ADD M is an example of register indirect addressing.

4. Immediate Addressing:

In immediate addressing mode the operand is specified within the instruction itself, examples are:

1. MVI A, 05 — Move 05 in register A.
   3E, 05 — The instruction in the code form.
2. ADI 06 — Add 06 to the content of the accumulator.
   C6, 06 — The instruction in the code form.

In these instructions the 2^{nd} byte specifies data.

3. LXI H, 2500 is an example of immediate addressing. 2500 is 16-bit data which is given in the instruction itself. It is to be loaded into H-L pair.

5. Implicit Addressing:

There are certain instructions which operate on the content of the accumulator. Such instructions do not require the address of the operand. Examples are: CMA, RAL, RAR etc.

ADDRESS SPACE PARITITIONING:

The Intel 8085 uses a 16-bit wide address bus for addressing memories and I/O devices. Using 16-bit wide address bus it can access 2^{16}=64 K bytes of memory and I/O devices. The 64K addresses are to be assigned to memories and I/O devices for their addressing. There are two schemes for the allocation of addresses to memories and input/output devices:

1. Memory mapped I/O scheme
2. I/O Mapped I/O scheme

Memory Mapped I/O Scheme:

In memory mapped I/O scheme there is only one address space. Address space is defined as the set of all possible addresses that a microprocessor can generate. Some addresses are assigned to memories and some addresses to I/O devices. An I/O device is also treated as a memory location and one

address is assigned to it. Suppose that memory locations are assigned the addresses 2000 to 24FF. One address is assigned to each memory location. Any one of these addresses cannot be assigned to an I/O device. The addresses for I/O devices are different from the addresses which have been assigned to memories. The addresses which have not been assigned to memories can be assigned to I/O devices. For example, 2500, 2502 etc. May be assigned to I/O devices. One address is assigned to each I/O device.

In this scheme all the data transfer instructions of the microprocessor can be used for both memory as well as I/O devices. For example, MOV A, M will be valid for data transfer from the memory location or I/O device whose address is in H-L pair. If the H-L pair contains the address of a memory location, data will be transferred from the memory location to the accumulator. If the H-L pair contains the address of an I/O device, data will be moved from the I/O device to the accumulator. The memory mapped I/O scheme is suitable for a small system.

I/O Mapped I/O Scheme:

In this scheme the addresses assigned to memory locations can also be assigned to I/O devices. Since the same address may be assigned to a memory location or an I/O device, the microprocessor must issue a signal to distinguish whether the address on the address bus if for a memory location or an I/O device. The Intel 8085 issues an IO/M signal for this purpose. When this signal is high the address on the address bus is for an I/O device. When this signal is low, the address on the address bus is for a memory location. Two extra instructions IN and OUT are used to address I/O devices. The IN instructions is used to read to data of an input device. The OUT instruction is used o send data to an output device. This scheme is suitable for a large system.

MEMORY AND I/O INTERFACING:

Several memory chips and I/O devices are connected to a microprocessor. Fig.7.1 shows a schematic diagram to interface memory chips or I/O devices to a microprocessor. An address decoding circuit is employed to select the required I/O device or a memory chips. Fig.7.2 shows a schematic diagram of a decoding circuit. If IO/M is high the decoder 2 is activated and the required I/O device is selected. If IO/M is low, the decoder 1 is activated and the required memory chip is selected. A few MSBs of the address lines are applied to the decoder to elect a memory chip or an I/O device.

Memory Interfacing:

The address of a memory location or an I/O device is sent out by the microprocessor. The corresponding memory chip or I/O device is selected by a decoding circuit. The decoding task can be performed by a decoder, a comparator, a bipolar PROM or PLA (Programmed logic array). In this section the application of 74LS138, a 1 to 8 lines decoder will be illustrated. Fig.7.3 shows the interface of memory chips through 74LS138. G1, G2A and G2B are enable signals. To enable 74LS138, G1 should be high, and G2A and G2B should be low. A, B and C are select lines. By applying proper logic to select lines any one of the outputs can be selected. Y_0, Y_1,.....Y_7 are 8 output lines. An output lines goes low when it is selected. Other output lines remain high. Table 7.1 shows the truth table for 74LS138. When G1 is low or G2A is high or G2B is high, all output lines become high. Thus 74LS138 acts as decoder only when G1 is high, and G2A and G2B are low.

The memory locations for EPROM 1 will be in the range 0000 to 1FFF. These are the memory locations for ZONE 0 for the memory chip which is connected to the output line Y_0 of the decoder.

Similarly for ZONE 1 is 2000 to 3FFF and for ZONE 7 is E000 to FFFF. Table 7.2 shows the memory locations for various zones.

The entire memory address (64K for 8085) has been divided into 8 zones. Address lines A_{15}, A_{14} and A_{13} have been applied to the select lines A, B and C of the 74LS138. The logic applied to these line selects a particular memory device, an EPROM or a RAM. Other address lines A_0, A_1, A_2,....and A_{12} go directly to memory chip. They decide the address of the memory location within a selected memory chip. $IO/\overline{M}$ is connected to G2B. When $IO/\overline{M}$ goes low for memory read/write operation, G2B goes low. G1 is connected to $+5V_{d.c}$. Supply and G2A is grounded.

I/O Interfacing:

Fig. 7.4 shows the interface of I/O devices through decoder 74LS138. As the address of an I/O device is of 8 bits, only A_8-A_{15} lines of address bus are used for I/O addressing. The address lines A_8, A_9 and A_{10} have been applied to select lines A, B and C of the 74LS138. The address lines A_{11}-A_{15} are applied to G2B through a NAND gate. G2B becomes low only when all address lines A_{11}-A_{15} are high. G2A is grounded. $IO/\overline{M}$ is connected to G1. When $IO/\overline{M}$ goes high for I/O read/write operation, G1 goes high. Table 7.3 shows the addresses of I/O devices connected to 74LS138.

DATA TRANSFER SCHEMES:

In s microprocessor-based system or in a computer data transfer takes place between two devices such as microprocessor and memory, microprocessor and I/O devices, and memory and I/O device. Usually, semiconductor memories are compatible with microprocessor because the same technology is employed in the manufacturing of both semiconductor memories and microprocessors. Hence, there is less problem associated with the interfacing of memory. A wide variety of I/O devices having wide range of speed and other different characteristics are available. They use different manufacturing technology such as electronic, electrical, mechanical, electro-mechanical, optical etc. Due to the reasons designers face difficulties in interfacing I/O devices with microprocessor. Special interfacing circuitries have to be designed for the purpose.

A microprocessor based system or a computer may have several I/O devices of different speed. A slow I/O device can not transfer data when microprocessor issues instruction for the same because it takes some time to get ready. To solve the problem of speed mismatch between a microprocessor and I/O devices a number of data transfer techniques have been developed. The data transfer schemes are classified into the following two broad categories.

1. Programmed data transfer schemes
2. DMA (Direct Memory Access) data transfer scheme.

Programmed Data Transfer Schemes:

Programmed data transfer schemes are controlled b the CPU. Data are transferred from an I/O device to the CPU (or to the memory through the CPU) or vice versa under the control of programs which reside in the memory. These programs are executed by the CPU when an I/O device is ready to transfer data. The microprocessor executes the program to transfer data. Programmed data transfer schemes are employed when small amount of data are to be transferred. The programmed data transfer schemes are classified into the following three categories.

i. Synchronous data transfer scheme

ii. Asynchronous data transfer scheme

iii. Interrupt driven data transfer scheme

These schemes will be discussed in subsequent sections.

DMA Data Transfer Scheme:

In DMA data transfer scheme CPU does not participate. Data are directly transferred from an I/O device to the memory or vice versa. The data transfer is controlled by the I/O device or a DMA controller. This scheme is employed when large amount of data are to be transferred. If bulk data are transferred through the CPU, it takes appreciable time and the process becomes slow. An I/O device which wants to send data using DMA technique, sends the HOLD signal to the CPU. On receiving an HOLD signal from an I/O device the CPU gives up the control of buses as soon as the current machine cycle is completed. The CPU sends a hold acknowledge signal to the I/O device to indicate that it has received the HOLD request and it has released the buses. The I/O device takes over the control of buses and directly transfers data to the memory or reads data from the memory.

DMA data transfer scheme is a faster scheme as compared to programmed data transfer scheme. It is used to transfer data from mass storage devices such as hard disks, floppy disks etc. It is also used for high-speed printers. When data transfer is over, the CPU regains the control over the buses.

DMA data transfer schemes are of the following two types:

i. Burst mode of DMA data transfer

ii. Cycle stealing techniques of DMA data transfer

Burst mode of DMA data transfer:

A scheme of DMA data transfer, in which the I/O device withdraws the DMA request only after all the data bytes have been transferred, is called burst mode of data transfer. By this technique a block of data is transferred. This technique is employed by magnetic disks drives. In case of magnetic disks data transfer can not be stopped or slowed down without loss of data. Hence, block transfer is a must.

Cycle Stealing Technique:

In this technique a long block of data is transferred by a sequence of DMA cycles. In this method after transferring one byte or several bytes the I/O device withdraws DMA request. This method reduces interference in CPU's activities. The interference can be eliminated completely by designing an interfacing circuitry which can steal bus cycle for DMA data transfer only when the CPU is not using the system bus.

In DMA data transfer schemes I/O devices control data transfer and hence the I/O devices must have registers to store memory addresses and byte count. It must also have electronic circuitry to generate necessary control signals required for DMA operations. Usually, I/O devices do not have these facilities. To solve this problem DMA controllers have been designed and developed. Examples of DMA controller chips are: Intel 8237 A, 8257 etc, which will be discussed in subsequent sections.

Synchronous Data Transfer:

Synchronous means "at the same time." The device which sends data and the device which receives data are synchronous with the same clock. When the CPU and I/O devices match in speed, this technique of the data transfer is employed. The data transfer with I/O devices is performed executing IN or OUT instructions for I/O mapped I/O devices or using memory read/write instructions for memory mapped I/O devices. The IN instructions is used to read data from an input device or input port. The

OUT instruction is used to send data from the CPU to an output device or output port. As the CPU and the I/O device match in speed, the I/O device is ready to transfer data when IN or OUT instruction is issued by the CPU. The status of the I/O device i.e., whether it is ready or not, is not examined before data are transferred, as it is not needed.

The I/O devices compatible with microprocessors in speed are usually not available. Hence, this technique of data transfer is rarely used for I/O devices. However, memories compatible with microprocessors are available, and therefore, this technique is invariably used with compatible memory devices.

Asynchronous Data Transfer:

Asynchronous means "at irregular intervals". In this method data transfer is not based on per determined timing pattern. This technique of data transfer is used when the speed of an I/O device does not match the speed of the microprocessor, and the timing characteristic of I/O device is not predictable. In this technique the status of the I/O device i.e. whether the device is ready or not, is checked by the microprocessor before the data are transferred. The microprocessor irritates the I/O device to get ready and then continuously checks the status of the I/O device till the I/O device becomes ready to transfer data. When I/O device becomes ready, the microprocessor sends instructions to transfer data. This mode of data transfer is also called handshaking mode of data transfer because some signals are exchanged between the I/O device and microprocessor before the actual data transfer takes place. The microprocessor issues an irritating signal to the I/O device to get ready (or to start). When I/O device becomes ready it sends signals to the processor to indicate that it is ready. Such signals are called handshake signals.

Fig. 7.5 shows a schematic diagram for asynchronous data transfer. Asynchronous data transfer i used for slow I/O devices. This technique is an inefficient technique because the precious time of the microprocessor is wasted in waiting.

Fig. 7.6 shows a simple example of asynchronous data transfer. An A/D converter has been interfaced to the microprocessor to transfer data in asynchronous mode. The microprocessor sends a start of conversation signal, S/C to the A/D converter. The A/D converter being a slow device as compared to a microprocessor, takes some time to convert analog signal to digital signal. When conversion is over the A/D converter makes end of conversion signal, E/C till it becomes high. When E/C becomes high, the microprocessor issues instruction for data transfer.

Some simple I/O devices may not have status signal. In such a case the microprocessor goes on checking whether data are available on the port or not. A keyboard interfaced to a microprocessor through a port is an example of this type of data transfer scheme.

Asynchronous data transfer discussed so far use software approach. It can also implemented by hardware approach employing READY signal. An I/O device is interfaced to the microprocessor through READY signal. I/O device (or memory chip) and microprocessor have READY pins. When an I/O device or memory chip becomes ready to transfer data, it makes READY signal high. The microprocessor checks READY signal before data are transferred. If READY is low the microprocessor enters a wait state. The status of the READY signal is sensed by the microprocessor T_2 state of a machine cycle. The microprocessor remains in wait state till READY becomes high. This technique is commonly used by slow memory devices.

Interrupt Driven Data Transfer:

In this scheme the microprocessor initiates an I/O device to get ready, and then it executes its main program instead of remaining in a program loop to check the status of the I/O device. When the I/O device becomes ready to transfer data, it sends a high signal to the microprocessor through a special input line called an interrupt line. In other words interrupt the normal processing sequence of the microprocessor. On receiving an interrupt the microprocessor completes the current instruction at hand, and then attends the I/O device. It saves the contents of the program counter on the stack first, and then takes up a subroutine called ISS (Interrupt Service Subroutine). It executes ISS to transfer data from or to the I/O device. Different ISS are to be provided for different I/O devices. After completing the data transfer the microprocessor returns back to the main program which it was executing before the interrupt occurred. Interrupt driven data transfer is used for slow I/O devices. It is an efficient technique as compared to asynchronous data transfer scheme because precious time of the microprocessor is not wasted in waiting while an I/O device is getting ready.

Fig. 7.7 shows the interfacing of an A/D converter to transfer data employing interrupt driven data transfer scheme. The microprocessor sends first the start of conversion signal, S/C to the A/D converter. Thereafter, the microprocessor executes its main program. A/D converter is a slow device compared to a microprocessor. It takes some time to convert analog signal to its equivalent digital quantity. When A/D converter completes the task of conversion, it makes an end of conversion signal, E/C high. The E/C signal is connected to an interrupt line of the microprocessor. When interrupt line goes high, the microprocessor takes all necessary steps to transfer data from the A/D converter. After completing the data transfer the microprocessor returns back to execute the main program that it was executing prior to the interrupt.

INTERRUPTS OF INTEL 8085:

The Intel 8085 has five interrupt inputs namely TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR. The TRAP has the highest priority, followed by RST 7.5, RST 6.5 and RST 5.5. The INTR has the lowest priority. When interrupts are to be used they are enabled by software using the instruction EI (Enabled Interrupt) in the main program. Fig. 7.8 shows the schematic diagram of interrupts of Intel 8085. The instruction EI sets the interrupt enable flip-flop to enable the interrupts. The use of the instruction EI enables all the interrupts. The instruction DI (Disable Interrupt) is used to disable interrupts. In certain situation it may be desired to prevent the occurrence of interrupts while a particular task is being performed by the microprocessor. This can be done using DI instruction. The DI instruction resets the interrupts enable flip-flop and disables all the interrupts except nonmaskable interrupt TRAP, see Fig.7.8. The system RESET also resets the Interrupt Enable flip-flop.

When an interrupt line goes high processor completes its current instruction and saves program counter on the stack. It also resets Interrupts Enable flip-flop before taking up ISS so that the occurrence of further interrupts by other devices is prevented during the execution of ISS. All the interrupts except TRAP are disabled by resetting the Interrupt Enable flip-flop.

The resetting of this flip-flop can be done in one of the three ways: by software using instruction DI, system reset or by recognition of an interrupt request.

Before the program returns back from ISS to the main program all the interrupts are to be enabled again. This is done using instruction EI in ISS before using the instruction RET.

At many occasions the programmer may like to prevent the occurrence of a few of several interrupts while microprocessor is performing certain tasks. This is done by masking off those interrupts

which are not required to occur when certain task is being performed. The interrupts which can be masked off (i.e., made ineffective) are called maskable interrupts. Masking is done by software. The Intel 8085 has two categories of interrupts: maskable and nonmaskable. The TRAP is a nonmaskable interrupt. It need not be enabled. It cannot be disabled. It is not accessible to user. It is used for emergency situation such as power failure and energy shut-off. RST 7.5, RST 6.5 and RST 5.5 are maskable interrupts.

Hardware and Software Interrupts:

Interrupts caused by I/O devices are called hardware interrupt. The normal operation of a microprocessor can also be interrupted by abnormal internal conditions or special instructions. Such an interrupt is called a software interrupt. RST n instructions of the 8085 are used for software interrupt. When RST n instruction is inserted in a program, the program is executed upto the point where RST n has been inserted. This is used in debugging of a program. See an example in Section 5.10.

The internal abnormal or unusual conditions which prevent the normal processing sequence of a microprocessor are also called exceptions. For example, divide by zero will cause an exception. Intel literatures do not use the term exception, whereas Motorola literatures use the term exception. Intel includes exception in software interrupt.

When several I/O devices are connected to INTR interrupt line, an external hardware is used to interface I/O devices. The external hardware circuit generates RST n codes to implement the multiple interrupts scheme. This will be discussed later in this chapter.

Interrupts Call-Locations:

When an interrupt occurs the program is transferred to a specific memory location. Then the monitor transfers the program from the specific memory location to a memory location in RAM, from where the user can write the program for interrupt service sub-routine (ISS). For TRAP, RST 7.5, 6.5 and 5.5 the program is automatically transferred to specific memory locations without any external hardware. The necessary hardware is already provided within 8085. The specific memory locations for these interrupts are as follows:

Interrupts	**Call-Location in Hex**
TRAP	0024
RST 7.5	003C
RST 6.5	0034
RST 5.5	002C

An interrupt for which hardware automatically transfers the program to a specific memory location is known as vectored interrupt.

INTR CALL LOCATIONS:

There are 8 numbers of CALL-locations for INTR interrupt. Table 7.4 shows CALL-locations, RST n instructions and corresponding hex-code. For INTR external hardware is used to transfer program to specific CALL-location. The hardware circuit generates RST codes for this purpose. The INTR line is sampled by the microprocessor in the last state of the last machine cycle of each instruction. When INTR is high, the microprocessor saves the contents of the program counter on the stack and then sends an interrupts acknowledge signal, INTA to the external hardware. In response to INTA the external

hardware generates a RST n code. When microprocessor receives this code, it transfers program to the corresponding CALL-location. Upto 8 number of I/O devices can be connected to INTR through external hardware or interrupt controller. The external hardware recognizes which I/O device has interrupted and it generates proper RST code that causes microprocessor to take up ISS for that particular I/O device. An external hardware can be built employing priority encoder and a tri-state buffer, see Ref.9. Alternatively, Priority Interrupt controller 8214 can be used. Programmable interrupt controller 8259 is more versatile, and present trend is to use programmable interrupt controller. INTA is used to activate 8259 or some other hardware. The 8259 has been described later in this chapter.

RST 7.5, 6.5 and 5.5:

RST 7.5, RST 6.5 and RST 5.5 are maskable interrupts. These interrupts are enabled by software using instructions EI and SIM (Set Interrupt Mask). The execution of the instruction SIM enables/disables interrupts according to the bit pattern of the accumulator. Fig. 7.9 shows accumulator contents for SIM instruction.

Bits 0-2 set/rest the mask bits of interrupts mask register for RST 5.5, 6.7 and 7.5. Bit 0 is for RST 5.5 mask, bit 1 for RST 6.5 mask and bit 2 for RST 7.5 mask. If a bit is set to 1 the corresponding interrupt is masked off (disabled). If it is set to 0, the corresponding interrupt is enabled. Bit 3 is set to 1 to make bits 0-2 effective. Bit 4 is an additional control for RST 7.5. If it is set to 1, the flip-flop for RST 7.5 is reset. Thus RST 7.5 is disabled regardless of whether bit 2 for RST 7.5 is 0 or 1.

CALL-Locations and Hex-Codes for RST n

RST n	Hex-Code	CALL-Locations
RST 0	C7	0000
RST 1	CF	0008
RST 2	D7	0010
RST 3	DF	0018
RST 4	E7	0020
RST 5	EF	0028
RST 6	F7	0030
RST 7	FF	0038

Bits 6 and 7 are for serial data output. The instruction SIM is also used for serial data transmission. Bit 6 is to enable SOD. If SIM instruction is executed the content of the 7^{th} bit of the accumulator is output on SOD line of the microprocessor. The content of bit 7 may be either high (1) or low (0).

The instruction DI disables all the interrupts. This is not always desired. When the processor is performing a particular task it may be desired to prevent the occurrence of a few of the several interrupts. In such a situation the interrupts which are not desired to occur may be masked off using SIM instruction.

Triggering Levels:

TRAP is edge as well as level triggered. This means that TRAP should go high and stay high until it is acknowledged. In this way false triggering caused by noise and transients is avoided. TRAP has also a flip-flop which is not shown in Fig. 7.8. The flip-flop is cleared when interrupt is acknowledged so that future interrupt may be entertained.

RST 7.5is positive edge triggered interrupt. It can be triggered with a short duration pulse. RST 6.5 and 5.5 are level triggered interrupts. Triggering level for RST 6.5 and 5.5 has to be kept high until the microprocessor recognises these interrupts. If the processor does not recognise these interrupts immediately, their triggering level should be held by external hardware.