Paper: Microprocessors and computer programming

Chapter: Microprocessor 8085 – Timing and control; Interrupts; Stack Memory; Interfacing devices.

Author: Priyamvada Bhardwaj

College/ Department: Ramjas College, University of Delhi

Table of Contents

Chapter 4: Microprocessor 8085 – Timing and control; Interrupts; Stack Memory; Interfacing devices.

Chapter4: Microprocessor 8085 – Timing and control; Interrupts; Stack Memory; Interfacing devices.

4.1 Introduction

After having studied the details of architecture of microprocessor 8085, it's various components and functions performed by it we need to go into further details of how these functions are performed by microprocessor (μP). One of the very important unit which co-ordinates the interaction between the μP and other devices including memory is timing and control unit. We will discuss control signals, timing diagrams and how the process of communication is managed by this unit. Stack memory is a read/write memory which again plays an important role in the functioning of μP. Interrupts are externally initiated signals with which the external user can ask μP to perform some task other than the one which it is performing and depending on the priority of the task μP has to perform it at that very moment or may even perform it after sometime. Interfacing devices are special hardware circuits through which the μP connects to the peripheral devices for e.g. keyboard, printers, display device etc. and the process through which this connection is made is called interfacing. In this chapter we will discuss these parameters i.e. timing and control unit, interrupts, stack memory and interfacing devices in detail.

4.2 Timing and control operation

Timing and control operations are performed by timing and control unit. These operations synchronize the process of communication between μP and peripheral devices by generating the control signals necessary for communication for e.g. $\overline{RD}$ and $\overline{WR}$ signals which indicate the availability of data on the data bus.

Whenever μP has to execute any instruction, the instruction is first received by the instruction register (8-bit) through A/D (address/data) bus. The instruction is then passed on to the instruction decoder where it is split, decoded and then passed to the control unit to generate necessary control signals for its execution. Besides doing this timing and control unit also checks whether any internal or external interrupt has occurred. If there is any kind of interrupt request this unit stops the execution of normal sequence of instructions to respond to it (interrupt) by generating the required control signals.

To execute any instruction μP has to pursue the following steps:

1) Identify the memory location from where the instruction is to be fetched
2) Decode the instruction using instruction decoder
3) Perform the function specified by the decoded instruction

All these operations are performed within a given time interval, which is provided by the clock of the system. With reference to the above mentioned operations certain terms can be defined. The definitions are as follows

1) **Instruction cycle:** Time required for completing the execution of an instruction is known as instruction cycle. The 8085 instruction cycle consists of one to six machines cycles or operations.

2) **Machine cycle:** It is the time required for completing a single operation. This operation can be accessing memory for read/write operation or accessing I/O device. There can be 3 to 6 clock periods or T-states in a machine cycle.
3) **T-states or clock cycles/periods (CLK):** T-state is equivalent to one clock period It is the time in which only a subdivision of the operation can be performed. The total number of T-states determines the size of the machine cycle required to perform an operation.

These three cycles can be represented with the help of a diagram which is shown in Fig.4.1 below.

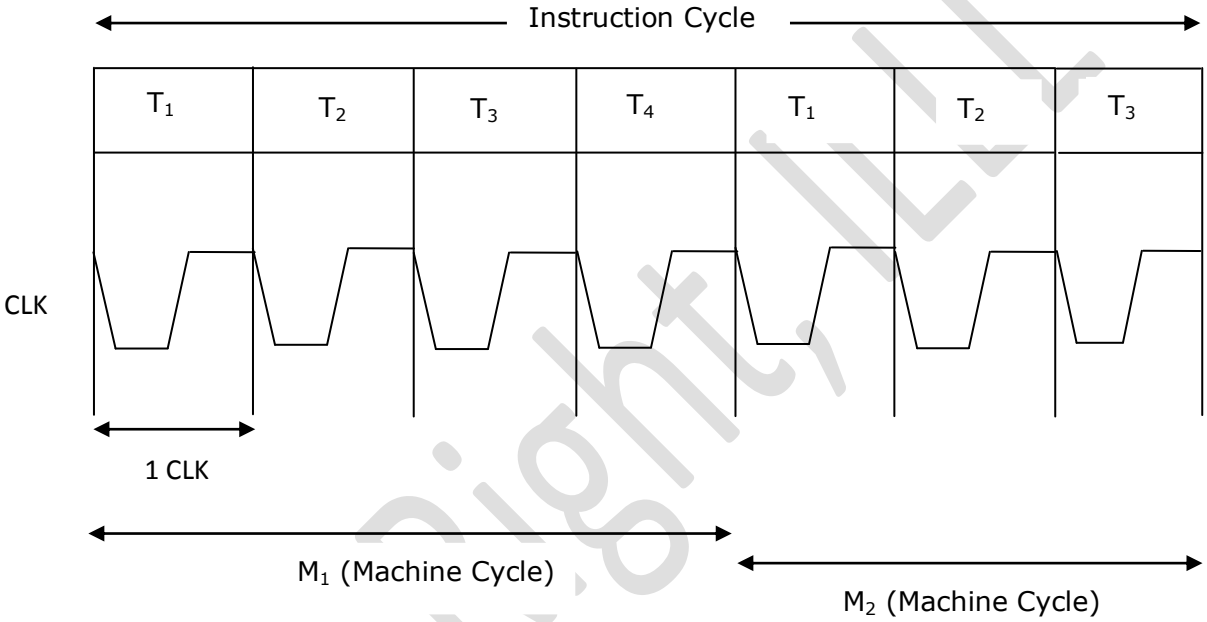


Fig 4.1 Representation of different cycles

4.2.1 Timing diagrams

A timing diagram of an instruction is a graphical representation of the time taken by the µP to fetch, decode and execute an instruction. The size of the instruction and the frequency of the µP decides the total amount of time taken to execute an instruction. This can be illustrated with the help of timing diagram. Consider the instruction MVI A, byte. It is basically a two byte instruction i.e. it requires two memory locations and this instruction can be written as follows:

Memory location	Mnemonics	Comments	Hexcode
2024	MVI A	move immediately 8-bit data in register A	3E
2025	data		data

So far as the timing diagram of this instruction is concerned it consists of two machine cycles M_1—opcode fetch and M_2—memory read. M_1 consists of 4 T-states and M_2 consists

of 3 T-states. Figure 4.2 (a) and (b) show the opcode fetch and memory read machine cycles respectively.

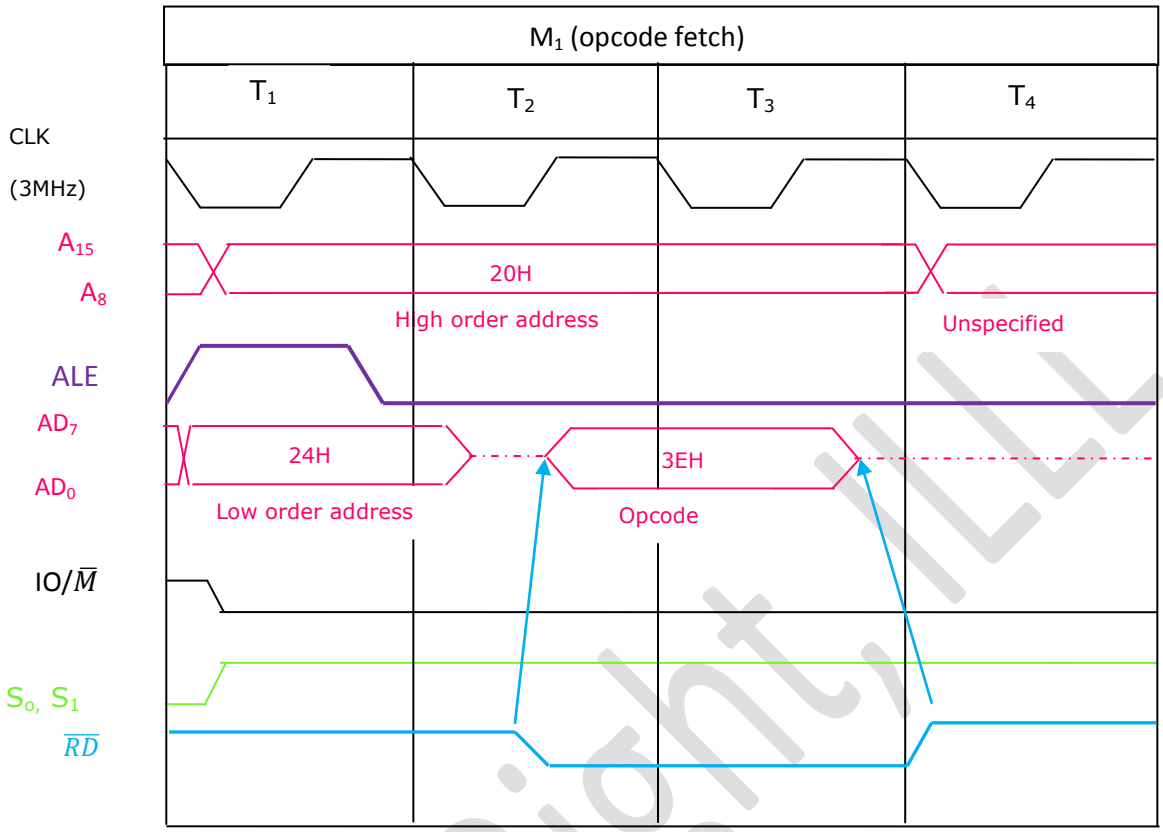


Figure 4.2 (a) Opcode fetch Machine cycle (M_1) for instruction MVI A , byte

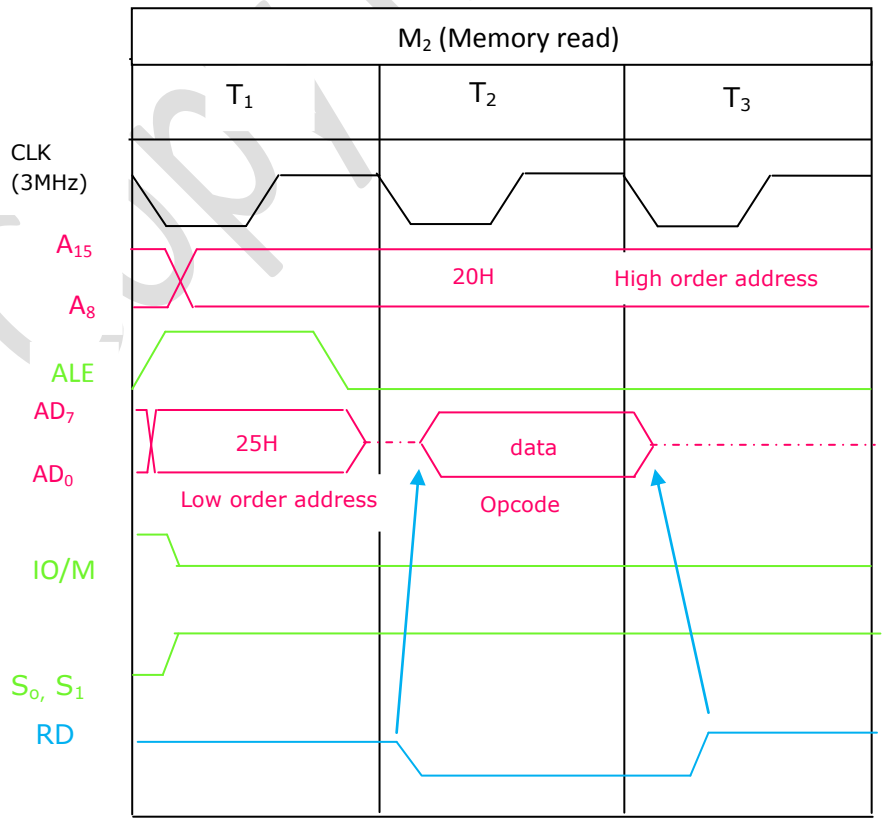


Figure 4.2 (b) Memory read Machine cycle (M_2) for instruction MVI A, byte

There are certain points that should be remembered while drawing the timing diagrams as they make it convenient to draw any timing diagram. These points are:

1) In each instruction cycle, there can be one or more than one machine cycles. The first machine cycle is always Opcode fetch. It can have four to six T-states.
2) The Memory read cycle requires three T-states
3) As is evident in both the figures that the high order address (A_8-A_{15}) bus carries high order address for the first 3 T-states.
4) ALE (Address Latch Enable) is a positive going pulse which is generated every time a machine cycle begins and remains positive for one T-state.
5) Low order address bus (AD_0-AD_7) carries low order address so long as ALE is positive.
6) Low order address bus (AD_0-AD_7) acts as data bus as soon as $\overline{RD}$ signal goes low.
7) If the µP is interacting with the memory i.e. reading instruction or data from the memory or writing information into the memory then $IO/\overline{M}$ is low and if µP is interacting with input or output device than it is high**.**

Calculation of execution time of Opcode fetch cycle, machine cycle and instruction cyle:

- Let the clock frequency f=5MHz
- One T-state= clock period (1/f)=0.2µs
- Execution time for Opcode fetch = 4 ×T = 4 × 0.2µs = 0.8 µs
- Execution time for Memory read = 3 ×T = 3 × 0.2µs = 0.6 µs
- Execution time for instruction = 7 ×T = 7 × 0.2µs = 1.4 µs

Value addition:

Generating Control Signals:

There are four control signals which are generated using three basic control signals namely $\overline{RD}$, $\overline{WR}$ and $IO/\overline{M}$. The four control signals are:

1) $\overline{MEMR}$ → Memory + Read
2) $\overline{MEMW}$ → Memory + Write
3) $\overline{IOR}$ → Input + Read
4) $\overline{IOW}$ → Output + Write

$\overline{MEMR}$ is used when µP is reading from memory and $\overline{IOR}$ is used when µP is reading from input device.
$\overline{MEMW}$ is used when µP is writing into the memory location and $\overline{IOW}$ is used when µP is sending data to the output device. Figure below shows how the control signals are generated.

Value addition:

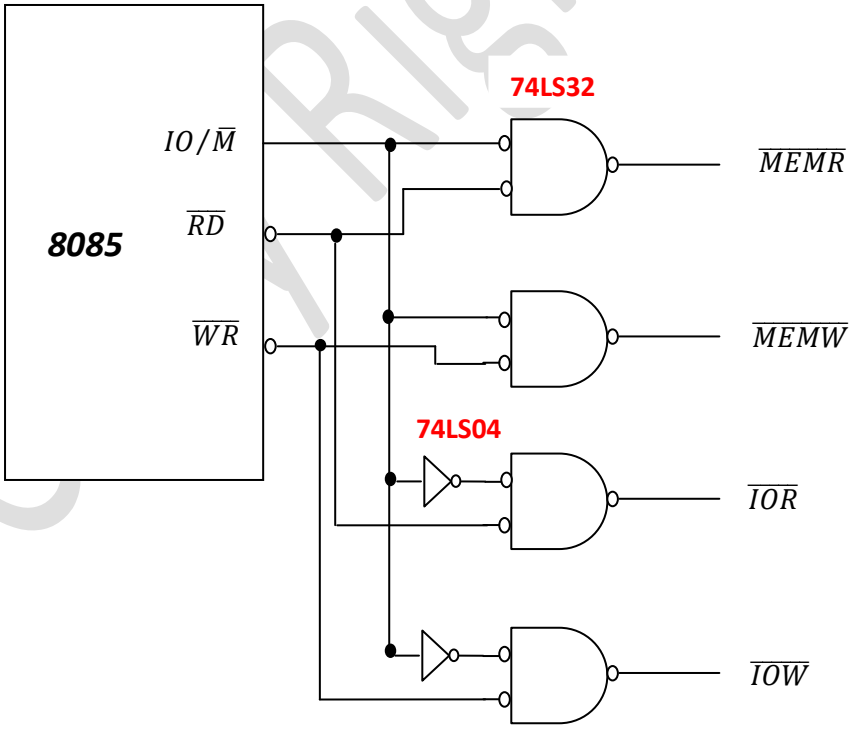


Figure: Schematic for generating control signals

4.3 Interrupts

An interrupt can be defined as any signal to the µP that alters the normal sequence of execution of a program. The interrupt can be introduced in the µP through an instruction written in the program or it can even be initiated by external device. Whenever µP receives any interrupt it's control gets shifted to some other location in order to execute a set of instructions called service routine which is written at that location. The µP resumes its operation after completing the service routine. The interrupt process in 8085 is controlled by the Interrupt Enable flip-flop, which is internal to the processor and can be set or reset by using software instructions. There exist two types of interrupts:

a) Software interrupts
b) Hardware interrupts

4.3.1 Software interrupts

The instruction set of 8085 includes eight RST (Restart) instructions referred to as software interrupts. These are one-byte instructions. Each of these instructions allows the transfer of the program execution to a specific location on page 00H. Therefore RST instructions act like a vector that points towards different memory locations.Table1 shows the list of different RST instructions.

Table 1: Software interrupts

Instruction	Hex Code	Vector location
RST0	C7	0000H
RST1	CF	0008H
RST2	D7	0010H
RST3	DF	0018H
RST4	E7	0020H
RST5	EF	0028H
RST6	F7	0030H
RST7	FF	0038H

We will discuss the functions performed by certain RESTART instructions like RST0, RST1 and RST5. The RST0 instruction service routine is stored in locations between 0000H and 0007H. When this instruction is given the Program Counter (PC) points to the memory location 0000H and its current address is loaded into the stack pointer. After the service routine is executed the PC returns back to the address of the next memory location by popping back the address from the stack pointer. Similarly RST1 service routine is stored in memory location 0008H to 000FH. In the same way when RST5 is inserted in the program it transfers the control to memory location 0028H. It is a break point service routine. The RST5 instruction displays the contents of accumulator and the flags when the A key is pressed and returns to the calling program when the Zero key is pressed.

4.3.2 Hardware interrupts

There are five hardware interrupts in 8085. They are:

1) INTR

2) RST 5.5

S. No.	Interrupts	Call locations
1	TRAP	0024H
2	RST 7.5	003CH
3	RST 6.5	0034H
4	RST 5.5	002CH
5	INTR	Non-vectored interrupt

3) RST 6.5
4) RST 7.5
5) TRAP

The interrupts are also classified as:

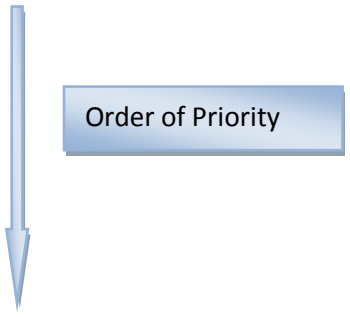
a) Vectored interrupts: Vectored interrupts are the ones which have some specific memory locations on page 00H associated with them and the control is automatically transferred to the respective memory location without any external hardware. Example of vectored interrupt is RST5.5, RST6.5, RST7.5 and TRAP.

b) Non-vectored interrupts: Non-vectored interrupts are the ones in which user has to specify the address along with the interrupt where the control is to be transferred. Example of non-vectored interrupt is INTR.

c) Maskable interrupts: Maskable interrupts are the one which µP need not attend immediately as and when it comes instead µP can ask it to wait for some time. Example of maskable interrupt is RST5.5, RST6.5, RST7.5 and INTR.

d) Non-Maskable interrupts: It is also known as NMI. It is the interrupt which µP has to attend as and when it comes i.e. it can't be ignored or asked to wait. Example of non-maskable interrupt is TRAP.

Table 2: Order of interrupts and their call locations.



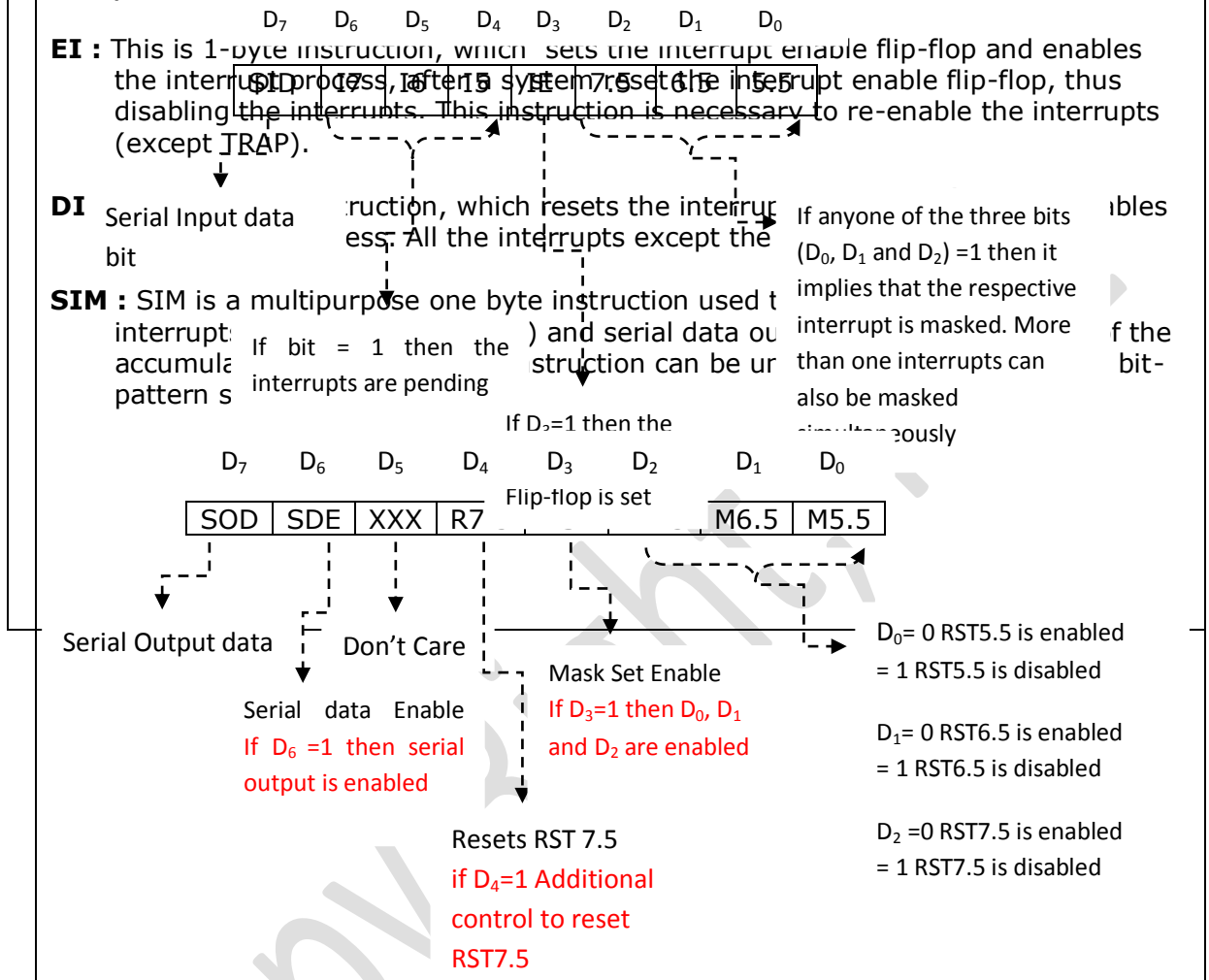
Value addition:

Value addition:

Before discussing various interrupts we need to discuss certain instructions in brief. These instructions are used to enable or disable the interrupts. The instructions are:

a) EI- Enable interrupt
b) DI- Disable interrupt
c) SIM- Set Interrupt Mask
d) RIM- Read interrupt Mask

RIM: This instruction is used to read the status of interrupts RST5.5, 6.5 and 7.5 and also to read serial data input bit. The bits of accumulator in case of RIM command can be interpreted as follows:



EI : This is 1-byte instruction, which sets the interrupt enable flip-flop and enables the interrupt process, after system reset the interrupt enable flip-flop, thus disabling the interrupts. This instruction is necessary to re-enable the interrupts (except TRAP).

DI ruction, which resets the interru ables ess. All the interrupts except the

SIM : SIM is a multipurpose one byte instruction used t interrupt) and serial data ou f the accumula struction can be ur bit-pattern s

INTR: While executing a program, the μP checks the INTR line on execution of each instruction. If the interrupt enable flip-flop is set and INTR is high, the μP after executing the current instruction resets the interrupt enable flip-flop and sends interrupt acknowledge signal ($\overline{INTA}$). In order to attend any other interrupt the μP has to enable interrupt enable flip-flop again.

With the help of INTR interrupt the user can also extend the interrupt capability of 8085 μP. The interrupt controller device used for this purpose is IC8259, which helps in catering upto eight peripheral devices.

RST 7.5, RST 6.5, RST 5.5: All the three interrupts are vectored and maskable interrupts. In order to enable these interrupts the μP requires two instructions:

a) EI (Enable Interrupt)

b) SIM (Set Interrupt Mask)

Besides these three interrupts are sensitive to different types of triggering as explained below:

RST 5.5 and RST 6.5: These are level sensitive interrupts which implies that the triggering level should be on until the µP completes the execution of current instruction. If the µP is not able to respond immediately to these interrupts then they should be stored by external hardware.

RST 7.5: It is positive-edge sensitive and can be triggered with a short pulse. In this case if the µP is unable to respond immediately then the request is stored internally by D-flip-flop.

TRAP: It is vectored, non-maskable interrupt. It is level-and edge-sensitive, which implies that the input should go high and remain in that state until it is acknowledged by the µP. In case of TRAP the instructions like EI (Enable interrupt), DI (Disable interrupt) and SIM (Set Interrupt Mask) are not required i.e. it need not be enabled, and it cannot be disabled. The triggering of various interrupts can be summarized in the form of diagram as shown below:

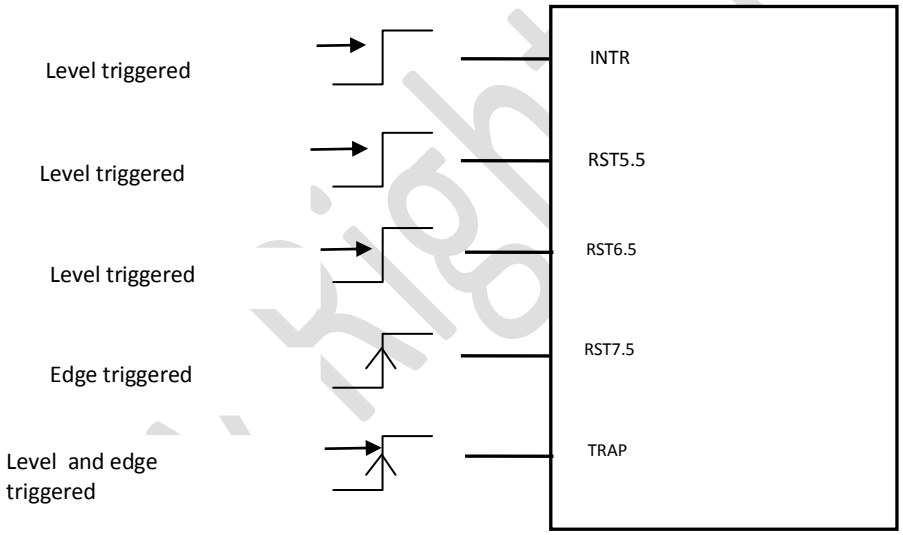


Figure 4.3 Triggering levels of various interrupts

4.4 Stack Memory

The stack memory can be described as a set of memory locations in the R/W memory that are used to store binary information temporarily during the execution of a program. The stack memory is used by both µP and programmer.

Whenever µP comes across any interrupt or subroutine the sequence of program execution is altered, in such a situation µP automatically places the address of the memory location present in the program counter on to the stack and after executing the interrupt or the subroutine the µP returns back to the original program by checking the address present in the stack.

Similarly the contents of the register can be stored on to the stack and retrieved from the stack by the programmer. During the execution of a program sometimes it becomes necessary to save the contents of certain registers and data in memory so that the registers may be used for other operations. After completing these operations the contents saved in memory can be transferred back to the registers. Memory locations for this purpose are set aside by the programmer in the beginning. The set of memory locations kept for this purpose is called stack.

Stack pointer is initialized in the beginning of the program by using the instruction LXI SP. This instruction loads the stack pointer register with the 16-bit memory address. Any area of the RAM can be used as stack but to prevent the program from being destroyed by the stack information, it is a general practice that the beginning of the stack is at the highest available memory location. The stack memory is based on last in first out principle (LIFO). The entering of data into the stack is called 'PUSH' operation and retrieving data from the stack is called 'POP' operation. The stack pointer always holds the address of the stack top.

Value addition:

LXI R_p: Load register pair immediate: It is a 3-byte command and is used to load the specified register pair with 16-bit data. For example if we want to load 3075H in register pair DE then the instruction is written as:

Instruction: LXI D,3075H
This instruction loads 75H in register E and 30H in register D.
Similarly if we write LXI SP, 3075H.then it loads stack pointer register (16-bit register) with 3075H.

General form of LXI instruction:

Opcode	**Operand**	**Comments**
LXI	Reg. pair, 16-bit data	Register pair can be B, D, H, SP (Stack pointer)

POP R_p: Pop off Stack to register pair: It is 1-byte instruction used to copy the contents from the stack memory to the register pair. When POP instruction is executed then following steps occur:

Suppose stack pointer contains memory address 3050H and data byte stored in 3050H is A7. The memory location 3051H contains C2, then on using POP B following steps are executed:

- The contents of memory location 3050H i.e. A7 are transferred to low-order register C
- Stack pointer is incremented by one to get the new value (SP+1) i.e. 3051H
- The contents of 3051H i.e. C2 is transferred to high-order register B and now the contents of register pair BC = C2A7
- The stack pointer is again incremented to get a new value (SP+1) i.e. 3052H

Therefore it is seen that when POP instruction is executed the contents of the memory location pointed out by the stack pointer register are copied to the low-order register (such as C, E, L and flag) of the operand. The stack pointer is incremented by 1 and the contents of that memory location are copied to the high-order register (B, D, H, A) of the operand.

General form of POP instruction:

Opcode	Operand	Comments
POP	Reg. pair	Register pair can be B, D, H, PSW(Program Status Word)

PSW represents accumulator and flag register, the accumulator is the high-order register and flag is low order register.

PUSH R_p: Push register pair onto the stack: It is 1-byte instruction used to copy the contents of the register pair onto the stack. When PUSH instruction is executed then following steps occur:

Suppose stack pointer contains memory address 2058H, register D contains 5AH and register E contains 63H, then on using PUSH D following steps are executed:

- Stack pointer (SP) is decremented by one i.e. (SP-1) which here happens to be 2057H.
- Contents of register D (High order register) i.e. 5AH are copied into memory location 2057H (M_{SP}-1)
- Stack pointer is again decremented by one i.e. (SP-1) which now becomes 2056H.
- The contents of register E (Low order register) are copied into the new value of memory location (M_{SP} – 1) i.e. 2056H.

Therefore in general we can say that when PUSH command is used the memory location address present in the stack pointer register is decremented and the contents of the high-order register (B, D, H, A) are copied into that location. The stack pointer register is decremented again and the contents of the low-order register (C, E, L and flag) are copied to that location.

General form of PUSH instruction:

Opcode	Operand	Comments
PUSH	Reg. pair	Register pair can be B, D, H, PSW(Program Status Word)

4.4.1 Stack Instructions

The commands used for writing into the stack memory and reading from it are PUSH and POP. The details of how these commands work are discussed in the next section. These instructions that write and read from the stack are called stack instructions. In these instructions indirect addressing mode is used, because a 16-bit register called stack pointer (SP) holds the address of the desired memory location.

4.4.2 How data is transferred into stack and retrieved from the stack using PUSH and POP command?

Suppose register B contains 6AH and register C contains B7H and we want to transfer this data to register pair DE using stack then first of all we write a short program for this purpose. Here we are explaining function of each instruction in the program with the help of diagrams.

Memory location	Mnemonics	Comments

3000	LXI SP, 3079H	This instruction loads stack pointer register with 3079H indicating to the μP that the memory space from memory location 3078H and moving upwards can be used as stack for temporary storage.
3003	LXI B, 6AB7H	This instruction loads register pair BC with 6AB7H.
3006	PUSH B	This instruction loads the contents of reg. B i.e. 6AH in location 3078H and contents of reg.C i.e. B7H in 3077H.

A		F	
B	6AH	C	B7H
D		E	
H		L	
SP		3079H	

Figure 4.4: Contents of register on execution of above instructions

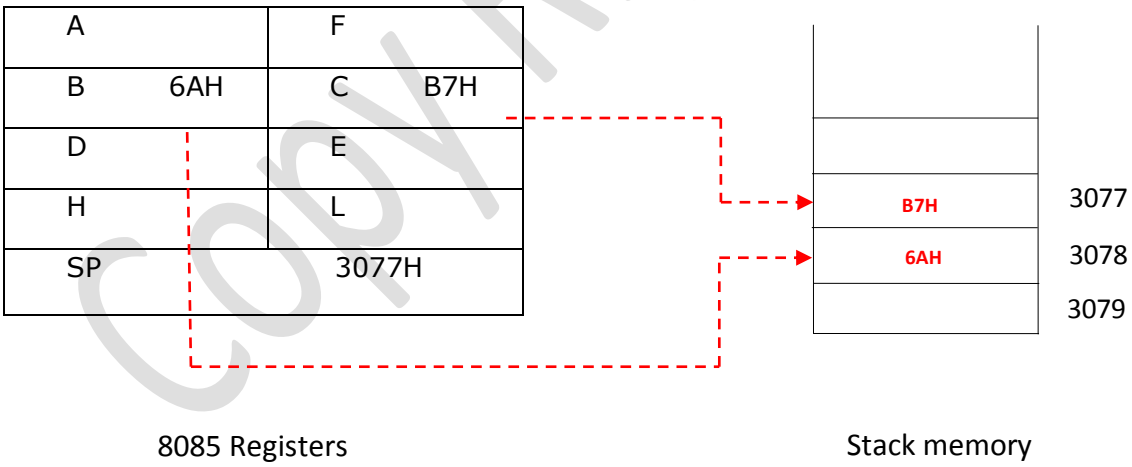


Figure 4.5: Contents of Register and stack memory after PUSH instruction is executed

Memory location	Mnemonics	Comments
3007	POP D	This instruction will transfer the contents of location 3077H to reg. E and 3078H to reg. D.

3008 HLT End of program.

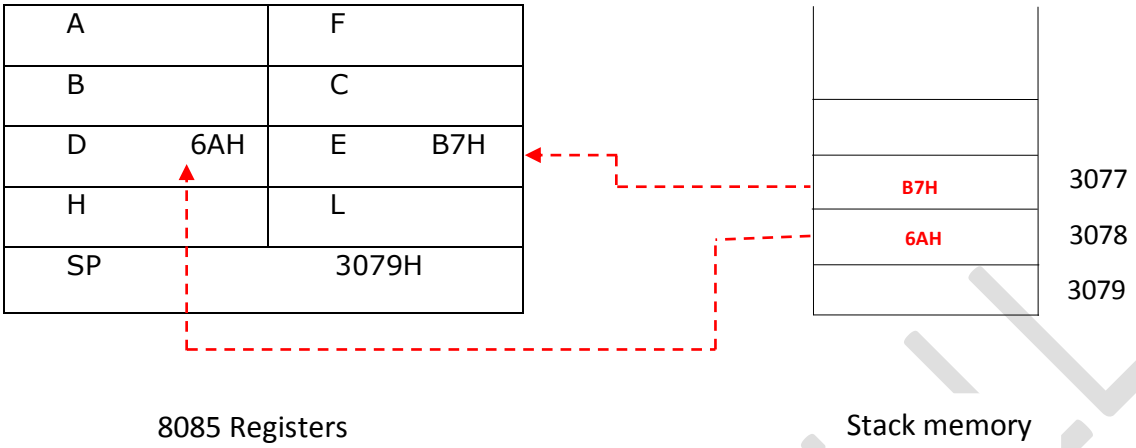


Figure 4.6: Contents of Register and stack memory after POP instruction is executed

4.5 Interfacing devices

Interfacing devices are basically hardware circuits that are specially designed to connect µP to external or peripheral devices like display unit, keyboard, printer or any other input/output device. The process through which this connection is made is called interfacing. This becomes clearer with the help of diagram shown below:

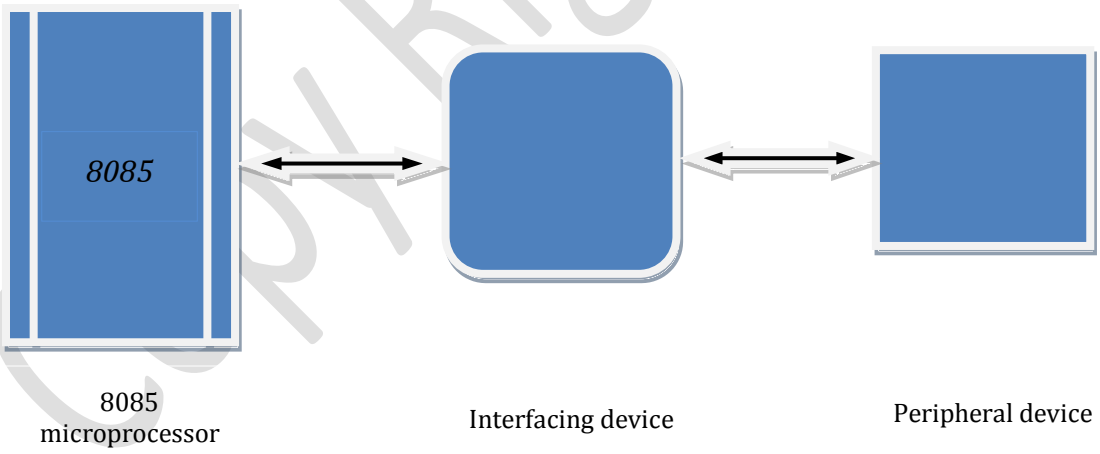


Figure 4.7 Connection of µP with peripheral device using interfacing device

There is a need for interfacing device because of difference in the characteristics of µP and peripheral devices. The difference can be in their operational speed as compared to µP or in their logic levels and power levels. The input and output devices are interfaced using programmable and non-programmable chips. Programmable devices are the ones which can be programmed by the user according to the requirement while non-programmable devices can't be programmed by the user.

In this chapter we will give a brief idea of some of the commonly used Programmable Peripheral Interfacing (PPI) devices/chips\ with 8085 microprocessor.

1. 8155/8156: Multi-functional programmable IO interfacing chip:

The Intel 8155 is a 40-pin IC. It contains 256 bytes RAM, two programmable I/O ports and a timer so it can be used as I/O interfacing chip or as a memory chip or as a timer chip. The IC 8156 is identical to the chip 8155 except that 8156 accepts active high chip enable (CE) signal and 8155 an active low CE signal. The main features of 8155/8156 can be summarized as follows:

1) 256-byte static RAM
2) Two 8-bit I/O ports
3) One 6-bit I/O port
4) One 14-bit binary counter/timer

The 8155 chip is specifically designed to be compatible with the 8085 μP, because control signals like$\overline{RD}$, $\overline{WR}$ and $IO/\overline{M}$ and special signal like ALE from 8085 can be directly connected to the device which in turn eliminates the need for generating control signals like $\overline{MEMR}$, $\overline{MEMW}$, $\overline{IOR}$ and $\overline{IOW}$, not only this it even eliminates the requirement of external demultiplexing of the low-order address bus AD_7-AD_0.

2. 8255: Programmable Peripheral interfacing chip:

It is programmable, parallel I/O device which is widely used to transfer data under various conditions. It's advantage is that it is economical when multiple I/O ports are used, it is flexible and versatile, therefore, it can be used with almost any μP. It is a 40-pin IC which operates on +5V power supply. It's main features can be summarized as follows:

1) Three 8-bit I/O ports constituted by 24-pins of the IC
2) Data bus buffer unit consisting of bidirectional data bus (D_O-D_7)
3) Read/Write control logic unit consisting of six input control signals (RF, WR, RESET, CS, A_o and A_1).
4) Group A and Group B control logic : The 8255 I/O ports are divided into these two groups. Each of the control block (Group A and Group B) receives commands from Read/write control logic and corresponding control word and accordingly controls the I/O port which comes under its block.

3. 8279: Display/Keyboard interface IC:

The IC 8279 is a programmable display and keyboard controller. This device is well suited for interfacing of matrix keyboard and seven-segment display in 8085 μP based system. There are two ways of interfacing a device with μP (a) hardware approach (b) software approach. The 8279 is a hardware approach of interfacing. In software approach the disadvantage is that the μP is occupied for a considerable amount of time in refreshing the display and checking the keyboard which is not the case if hardware approach is used. The disadvantage of using 8279 is that it is costly but advantage is that it saves processor time and the software development cost.

The 8279 IC is a 40 pin IC which requires 5V supply. It has got four major sections namely:

1) Keyboard section
2) Scan section
3) Display section
4) Microprocessor unit (MPU) interface

4. 8254/8253: Programmable interval timer:

It is a 24-pin IC which operates at +5V power supply. As it generates accurate time delays therefore it can be used as:

- Real-time clock
- Event counter
- Digital one shot
- Square wave generator
- Complex waveform generator
- Programmable rate generator
- Complex motor controller and
- Binary rate multiplier

. The 8254 is an upgraded version of IC 8253. The features of the two devices are almost identical except that 8254 can operate with higher clock frequency range and can latch the count and the status of the counters. The 8254/8253 IC can be divided into four functional blocks as follows:

1) Data bus buffer
2) Read/write logic
3) Control word register
4) Counter 0, counter 1 and counter 2.

5. 8257/8237 : DMA controller:

Direct Memory Access is an I/O technique which is commonly used for high speed data transfer. In this technique the µP leaves it's control over the buses to a DMA controller device. This DMA controller device manages data transfer between memory and peripheral device, thus bypassing the microprocessing unit. The DMA controlled data transfer does not require software; hence, it is faster than the µP controlled data transfer. This technique is used in any system that requires a high speed data transfer for example CRT system, hard disk drive system etc. The 8257 DMA controller consists of following parts:

1. Four DMA channels
2. Data bus buffer
3. Read/write logic
4. Control logic
5. Priority resolver
6. Mode set register
7. Status register

The 8237 is also a DMA controller device specially designed to improve the systems performance. It allows the peripheral devices to directly transfer

information from the system memory. Unlike 8257, memory-to-memory transfer capability is also provided in 8237.

6. **8259A** : Programmable interrupt controller:

This IC is used to relieve µP from the task of polling in multi-level priority interrupt system. It can manage upto 8 interrupting devices and can handle 64 levels of interrupts. It is primarily designed for use in real time interrupt driven microcomputer systems. Another important feature of 8259A is that there are several priority modes available to the programmer and therefore it can be configured by the user to match the system requirements. Besides this the priority modes can be re-configured at any time during the main program.

Besides the interfacing devices discussed above there are various other popularly used devices which are used for the purpose of interfacing and the names of some of them are mentioned below:

1) 8251: Universal synchronous/asynchronous receiver/transmitter.
2) 8212 : Latch
3) 0801/0802 : Digital to analog converter (DAC)
4) 0808/0809 : Analog to Digital converter (ADC)

SUMMARY

In this chapter we have discussed some of the very important units and features of 8085 µP like timing and control unit, stack memory, interfacing devices and interrupts. Timing and control unit provides synchronization between the slow peripheral devices and microprocessor. Timing diagrams and the calculations of execution time of an instruction has also been discussed.

What is stack memory and how it is used by the microprocessor and the programmer has been explained in detail. Commands like PUSH and POP have also been discussed.

Programmable Peripheral Interfacing devices like IC 8155, 8255, 8279, 8257 etc. have been briefly explained as to how they are used for interfacing different peripheral devices with 8085 µP and some of their salient features have also been mentioned.

Interrupt, which is an important feature of 8085 µP has been discussed. Different interrupts and commands required for implementing these interrupts have also been explained.

EXERCISES

Question Number	Type of question
1	Multiple choice questions

(1) Which of the following is first machine cycle of instruction cycle?

(a) Opcode fetch cycle (b) Operand fetch cycle

(c) I/O read cycle (d) Memory read cycle

(2) How many T-states are required for MVI M, 5DH?

(a) 4T-states (b) 10T-states

(c) 7T-states (d) 13T-states

(3) For how many T-states ALE remains high during a machine cycle?

(a) 1T-state (b) 2T-states

(c) 3T-states (d) 4T-states

(4) Calculate the execution time of an instruction having 13T-states if the microprocessor is operating at a frequency of 5MHz.

(a) 2.6μs (b) 3.6μs

Question Number	Type of question
2	Fill in the blanks

(9) Which of the following IC is used as programmable interrupt controller?

(a) 8255 (b) 8259A

(c) 8279 (d) 8257

(10) 8255 is a

(a) Programmable interrupt control IC (b) Programmable interval timer IC

(c) Keyboard and display interface IC (d) Programmable peripheral interface IC

Correct answers	(1) a	(6) d
	(2) b	(7) b
	(3) a	(8) d
	(4) a	(9) b
	(5) a	(10) d

Correct answers	(1) $\overline{\text{IOR}}$ (2) Timing diagram (3) ALE(Address Latch Enable) (4) RIM (5) PUSH (6) Accumulator, Flags (7) Interfacing devices

Microprocessor 8085 – Timing and control; Interrupts; Stack Memory; Interfacing devices.

Question Number	Type of question
3	Subjective questions

1) What is an instruction cycle?
2) Draw and explain the timing diagram of Mov D,A.
3) What is a stack? On what principal does it work?
4) Explain the working of POP instruction with the help of diagram.
5) What are vectored and non-vectored interrupts?
6) Distinguish between:
   (a) Software and hardware interrupt
   (b) Maskable and non-maskable interrupt
7) What is an interfacing device? Why interfacing is required?
8) Write short notes on:
   (a) 8257 DMA controller
   (b) 8155 programmable I/O interfacing chip.

GLOSSARY

Stack memory: It is a read/write memory which is used for temporarily storing the data during the execution of a program. Stack memory is shared by both the µP and the user.

Interrupt: It is an externally initiated signal through which the user can ask the µP to perform some other task in between the program it is executing.

Interfacing: The process through which the peripheral devices are connected to the µP is known as interfacing.

Timing diagram: It is a graphical representation of the time taken by the µP to fetch, decode and execute an instruction.

Service routine: A set of instructions which the µP executes on any interrupt request.

Instruction decoder: It identifies the instruction, takes the information from instruction register and decodes or translates the instruction to be performed.

Instructions register: It is a 8-bit register. When an instruction is fetched from memory then it is stored in this register.

Polling: Polling is a commonly used method in which the processor checks all the devices in sequential order to see if it requires service or not. However this technique wastes a lot of processor time, as it checks the status of all the devices all the time.

References

1. Suggested readings

- Microprocessor architecture, Programming and applications with the 8085 – Ramesh S. Gaonkar
- Fundamentals of Microprocessor and Microcomputers – B.Ram
- Microprocessor 8085 and its interfacing – Sunil Mathur
- 8085 microprocessor: Programing and interfacing – Srinath N. K.