

A Course Material on

# **Microprocessor and Microcontroller**

By

**MR.PADMANABAN**

**ASSISTANT PROFESSOR**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**PRATHYUSHA ENGINEERING COLLEGE**

## **EE8551 MICROPROCESSORS AND MICROCONTROLLERS L T P C 3 0 0 3**

### **OBJECTIVES:**

- To study the Architecture of uP8085 & uC 8051
- To study the addressing modes & instruction set of 8085 & 8051.
- To introduce the need & use of Interrupt structure 8085 & 8051.
- To develop skill in simple applications development with programming 8085 & 8051
- To introduce commonly used peripheral / interfacing
- 

### **UNIT I 8085 PROCESSOR**

**9**

Hardware Architecture, pinouts – Functional Building Blocks of Processor – Memory organization – I/O ports and data transfer concepts– Timing Diagram – Interrupts.

### **UNIT II PROGRAMMING OF 8085 PROCESSOR**

**9**

Instruction -format and addressing modes – Assembly language format – Data transfer, data manipulation& control instructions – Programming: Loop structure with counting & Indexing – Look up table - Subroutine instructions - stack.

### **UNIT III 8051 MICRO CONTROLLER**

**9**

Hardware Architecture, pinouts – Functional Building Blocks of Processor – Memory organization – I/O ports and data transfer concepts– Timing Diagram – Interrupts-Comparison to Programming concepts with 8085.

### **UNIT IV PERIPHERAL INTERFACING**

**9**

Study on need, Architecture, configuration and interfacing, with ICs: 8255 , 8259 , 8254,8237,8251, 8279 , - A/D and D/A converters & Interfacing with 8085& 8051.

### **UNIT V MICRO CONTROLLER PROGRAMMING & APPLICATIONS**

**9**

Data Transfer, Manipulation, Control Algorithms& I/O instructions – Simple programming exercises key board and display interface – Closed loop control of servo motor- stepper motor control – Washing Machine Control.

- Ability to understand and analyse, linear and digital electronic circuits.
- To understand and apply computing platform and software for engineering problems.

### **TEXT BOOKS:**

1. Krishna Kant, “Microprocessor and Microcontrollers”, Eastern Company Edition, Prentice Hall of India, New Delhi , 2007.
2. R.S. Gaonkar, „Microprocessor Architecture Programming and Application“, with 8085, Wiley
3. Soumitra Kumar Mandal, Microprocessor & Microcontroller Architecture, Programming & Interfacing using 8085,8086,8051,McGraw Hill Edu,2013.

### **REFERENCES:**

1. Muhammad Ali Mazidi & Janice Gilli Mazidi, R.D.Kinely „The 8051 Micro Controller and Embedded Systems“, PHI Pearson Education, 5th Indian reprint, 2003.
2. N.Senthil Kumar, M.Saravanan, S.Jeevananthan, „Microprocessors and Microcontrollers“, Oxford,2013.
3. Valder – Perez, “Microcontroller – Fundamentals and Applications with Pic,” Yeesdee Publishers, Tayler & Francis, 2013.

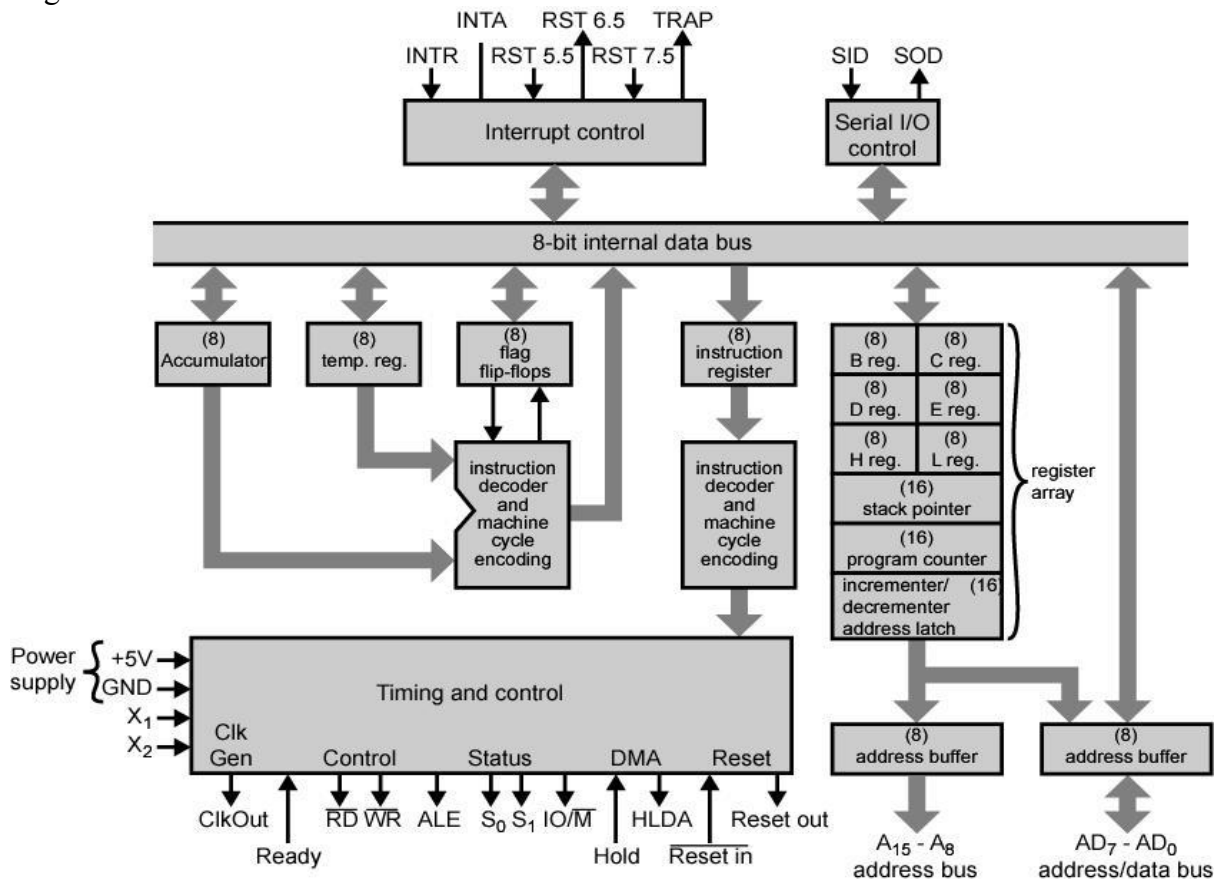
## UNIT I

### 8085 PROCESSOR

#### 1) 8085 Architecture:

The architecture of 8085 consist various components like:

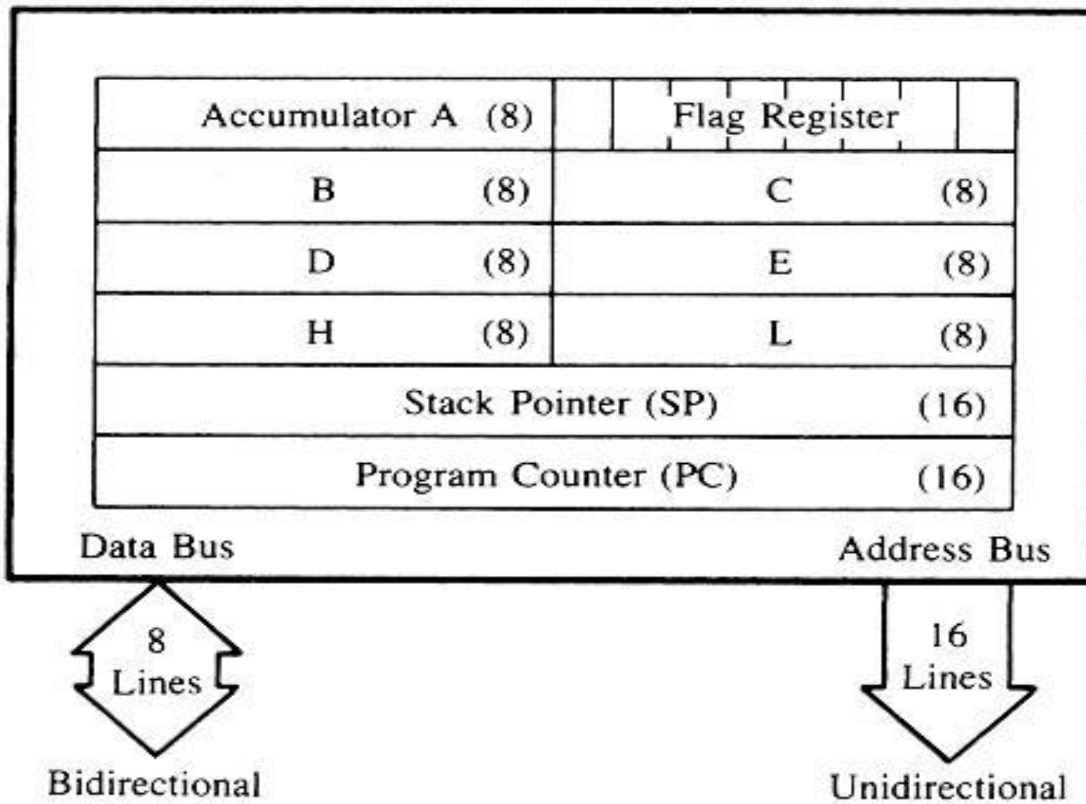
1. Accumulator & Register sets.
2. Program counter and stack pointer.
3. Flag Register.
4. ALU.
5. Instruction decoder and machine cycle encoder.
6. Address buffer.
7. Address/data buffer.
8. Increment/Decrement latch.
9. Interrupt control.
10. Serial I/O like SOD,SID.
11. Timing and Control circuit.



#### Accumulator:

- The accumulator is an 8-bit register then is part of the arithmetic/logic unit(ALU).
- This register is used to store to store 8-bit data this data is used to perform arithmetic & logical operation.
- The result of an operation is stored in the accumulator.

- The accumulator is also identified as register A.
- The accumulator is used for data transfer between an I/O port and memory location.



### Registersets:

- The 8085 simulator has six general-purpose registers to store 8-bit data; these are identified as **B, C, D, E, H** and **L**. They can be combined as register pair like **BC, DE** and **HL** – to perform some 16-bit operations.
- The programmer can use these registers to store or copy data into the registers by using data copy instructions.
- Out of these six registers, four 8-bit registers are scratch pad registers which are accessible to the programmer and hence can be used to temporarily store data during a program execution.
- And the two registers **H** and **L** are utilized in indirect addressing mode. In this mode, the memory location i.e. the address is specified by the contents of the registers.

### Program Counter (PC):

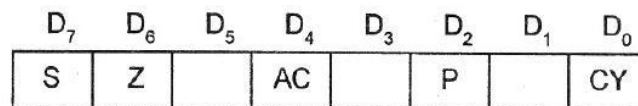
- 16 bit register which holds the memory address of the next instruction to be executed in the next step.
- This 16-bit register deals with sequencing the execution of instructions. This register is a memory pointer. Memory locations have 16-bit addresses, and that is why this is a 16-bit register.
- 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 instruction 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):

- Stack pointer is used during subroutine calling and execution.
- 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.

### Flag or status register:

- The ALU includes five 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), and Auxiliary Carry (AC) flags.
- The most commonly used flags are Zero, Carry, and Sign. The microprocessor uses these flags to test data conditions.
- For example, after an addition of two numbers, if the sum in the accumulator is larger than eight bits, the flip-flop uses to indicate a carry -- called the Carry flag (CY) -- is set to one.
- When an arithmetic operation results is zero, the flip-flop called the Zero(Z) flag is set to one.
- The Figure shows an 8-bit register, called the flag register, adjacent to the accumulator..



Bit positions of various flags in the flag register of 8085

- **Flag** is an 8-bit register containing 5 no.s of 1-bit flags:
  1. Sign - If the result of the latest arithmetic operation is having MSB (most- significant bit) „1“ (meaning it is a negative number), then the sign flag is set. Otherwise, it is reset to „0“ which means it is a positive number.
  2. Zero - If the result of the latest operation is zero, then zero flag will be set; otherwise it be reset.
  3. Auxiliary carry - set if there was a carry out from bit 3 to bit 4 of the result.
  4. Parity - set if the parity (the number of set bits in the result) is even. i.e., If the result of the latest operation is having even number of „1“s, then this flag will be set. Otherwise this will be reset to „0“. This is used for error checking.
  5. Carry - set if there was a carry during addition, or borrow during subtraction/comparison. Otherwise it will be reset.

### Instruction register or Decoder:-

- Instruction register holds the instruction that is currently being processed.
- Once the instruction is fetch from the memory, it is reloaded in the instruction register for some time, after the decoder decode the instruction performing some event or task.

### Address buffer:

- The remaining higher order address lines form the address buffer ranging from[A15-A8].This is having the unidirectional buffer

### Address/data buffer:

- The address bus will be having 16 address lines[A15-A0] .In which A7-A0 are called as lower addressing lines and these are multiplexed with data lines[D7-D0] to form multiplexed address /data buffer .The address/data buffer is the bidirectional bus.

### Increment/Decrement Address Latch:

- It increments/ decrements the address before sent to the address buffer

### Interrupts:

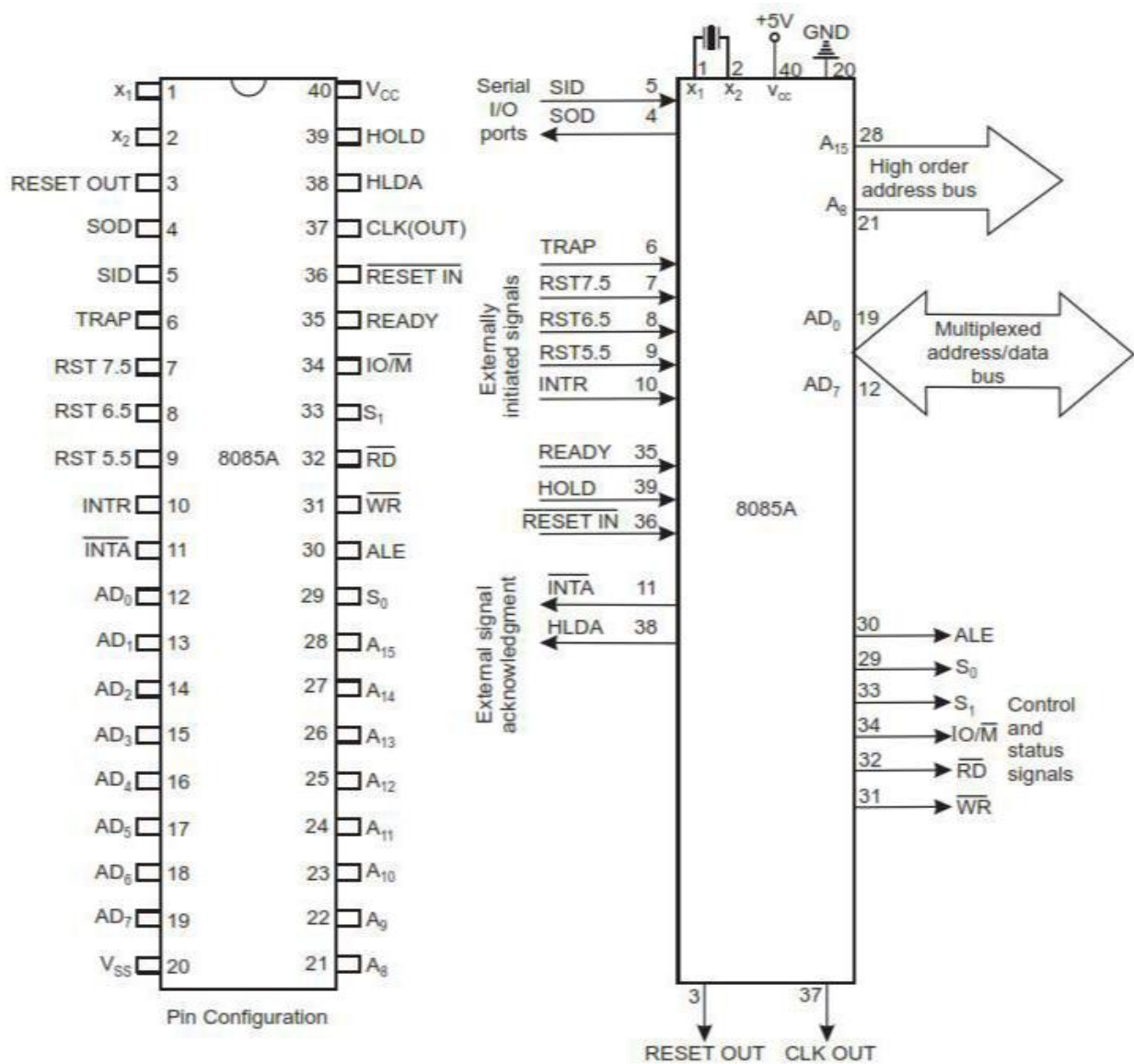
The processor has 5 interrupts. They are presented below in the order of their priority (from lowest to highest):

- **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. RST 5.5, RST6.5 and RST7.5 interrupts can be enabled or disabled individually using SIM Instruction

### Serial I/O control

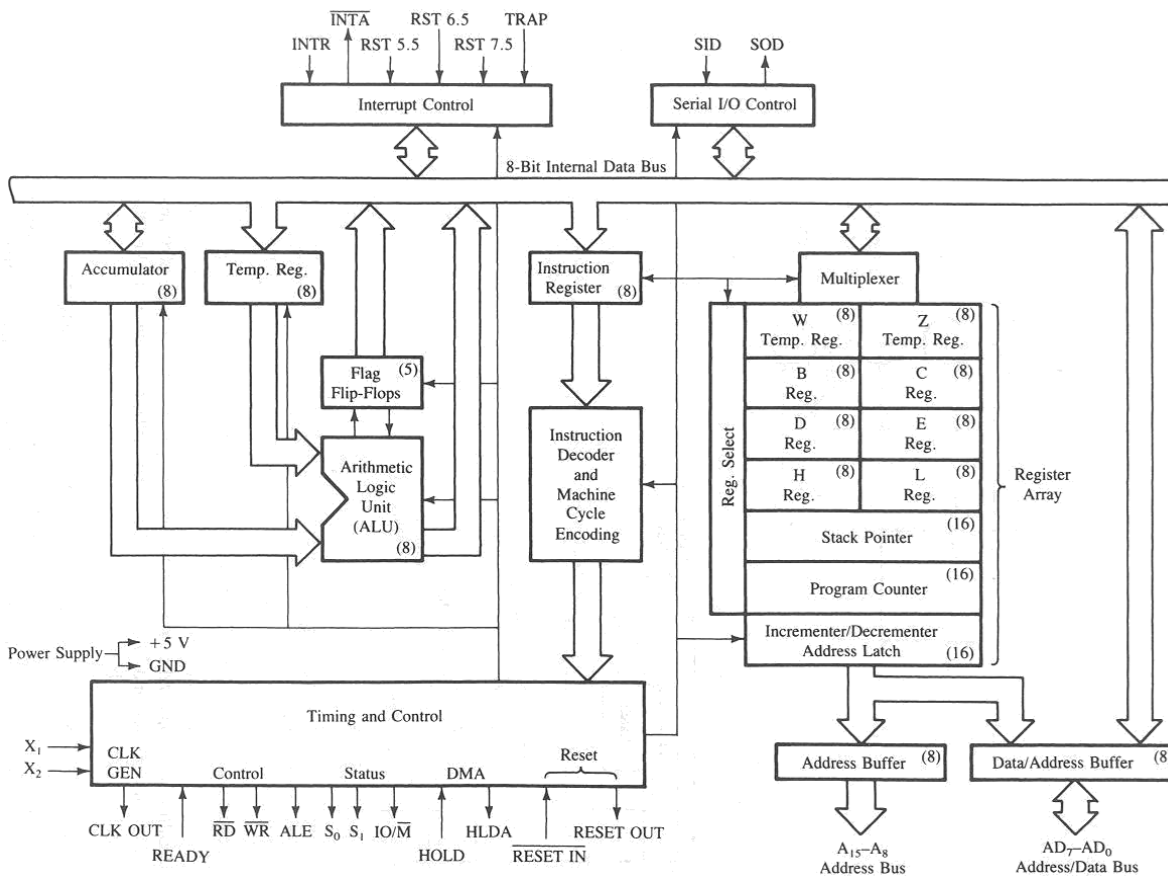
- There are control signals used for controlling 8085 these are subdivided into 2 types:
  1. SID(serial input data):  
This is used for transferring of data into the memory serially.
  2. SOD(serial output data):  
This is used for transferring of data from memory to external devices
- Interrupt control is used to transfer the ISR to the CPU.

## 2) PIN DIAGRAM



### Timing and Control Unit:

- The timing and control unit accepts information from the instruction decoder and generates different control signal. This unit synchronizes all the microprocessor operation and generates control and status signal necessary for communication between the microprocessor and peripherals.



**A8 - A15 (Output 3 State):**

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

**AD0 - AD7 (Input / Output 3 state):**

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

**ALE (Output):**

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

**S0, S1 (Output):**

Data Bus Status. Encoded status of the bus cycle:

- S1 S0
- 0 0 HALT
- 0 1 WRITE
- 1 0 READ
- 1 1 FETCH S1 can be used as an advanced R/W status.

**RD (Output 3state):**

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

**WR (Output 3state):**

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

**READY (Input):**



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

### **HOLD (Input):**

- ✓ 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 buses as soon as the completion of the current machine cycle.
- ✓ Internal processing can continue. The processor can regain the buses only after the Hold is removed. When the Hold is acknowledged, the Address, Data, RD, WR, and IO/M lines are 3stated.

### **HLDA (Output):**

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

### **INTR (Input):**

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

### **INTA (Output):**

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

### **RESTART INTERRUPTS:**

These three inputs have the same timing as INTR except they cause an internal RESTART to be automatically inserted.

- RST 7.5 Highest Priority
- RST 6.5
- RST 5.5 Lowest Priority

### **TRAP (Input):**

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

### **RESET IN (Input):**

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

### **RESET OUT (Output):**

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

### **X1, X2 (Input):**

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

### **CLK (Output):**

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

### **IO/M (Output):**

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

**SID (Input):**

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

**SOD (output):**

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

**Vcc:** +5 volt supply.

**Vss:** Ground Reference.

### 3) MEMORY ORGANIZATION

**Memory Interfacing**

The memory is made up of semiconductor material used to store the programs and data. Three types of memory is,

Process memory

Primary or main memory

Secondary memory

**Typical EPROM and Static RAM**

A typical semiconductor memory IC will have n address pins, m data pins (or output pins).

Having two power supply pins (one for connecting required supply voltage (V and the other for connecting ground).

The control signals needed for static RAM are chip select (chip enable), read control (output enable) and write control (write enable).

The control signals needed for read operation in EPROM are chip select (chip enable) and read control (output enable).

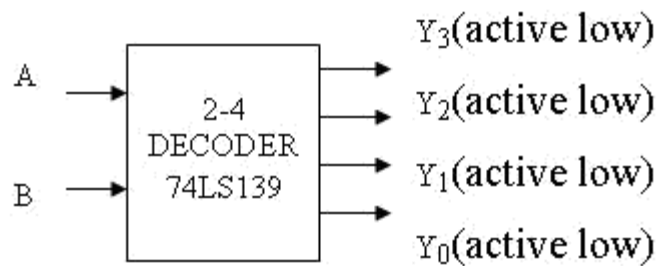
**Decoder**

It is used to select the memory chip of processor during the execution of a program. \_\_\_\_\_

No of IC's used for decoder is,

2-4 decoder (74LS139)

3-8 decoder (74LS138)



**Block Diagram of 2-4 Decoder**

**Truth Table for 2-4 Decoder**

Input		Output			
B	A	$\bar{Y}_3$	$\bar{Y}_2$	$\bar{Y}_1$	$\bar{Y}_0$
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	1

**Example for Memory Interfacing**

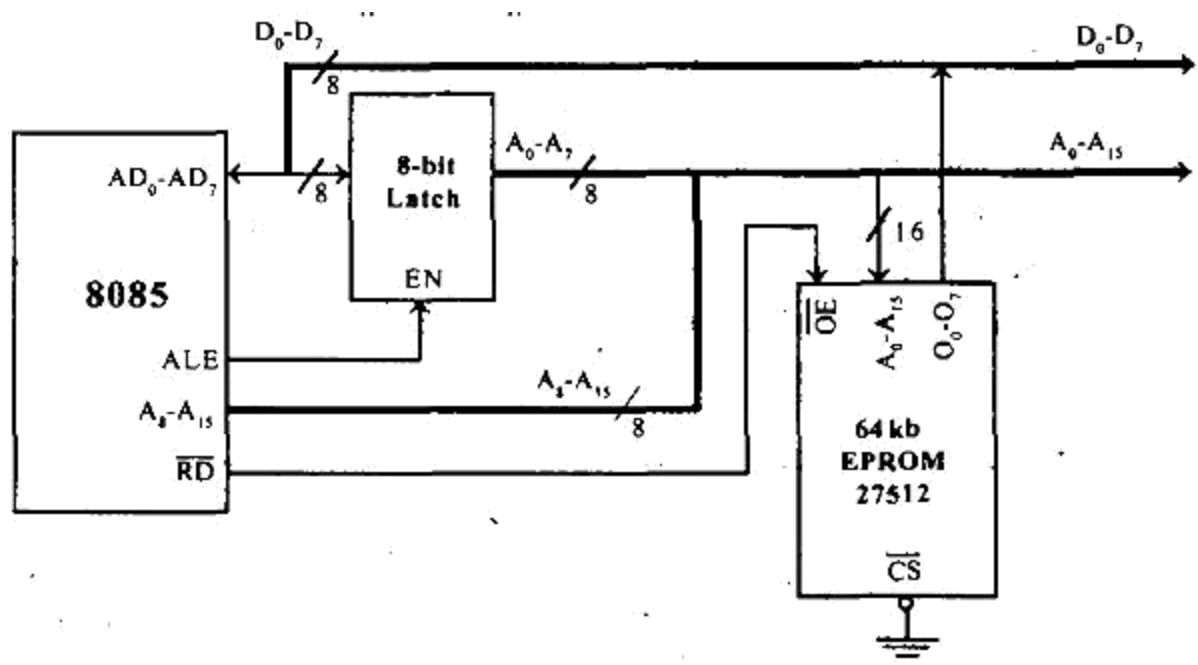
Consider a system in which the full memory space 64kb is utilized for EPROM memory. Interface the EPROM with 8085 processor.

The memory capacity is 64 Kbytes. i.e  
 $2^n = 64 \times 1000$  bytes where  $n =$  address lines.

So,  $n = 16$ .

In this system the entire 16 address lines of the processor are connected to address input pins of memory IC in order to address the internal locations of memory.  
 The chip select (CS) pin of EPROM is permanently tied to logic low (i.e., tied to ground).

Since the processor is connected to EPROM, the active low RD pin is connected to active low output enable pin of EPROM.  
 The range of address for EPROM is 0000H to FFFFH.



Memory Interfacing

#### 4) TIMING DIAGRAM

Timing Diagram is a graphical representation. It represents the execution time taken by each instruction in a graphical format. The execution time is represented in T-states.

##### **Instruction Cycle**

The time required to execute an instruction is called instruction cycle.

##### **Machine Cycle**

The time required to access the memory or input/output devices is called machine cycle.

##### **T-State**

The machine cycle and instruction cycle takes multiple clock periods.

A portion of an operation carried out in one system clock period is called as T-state.

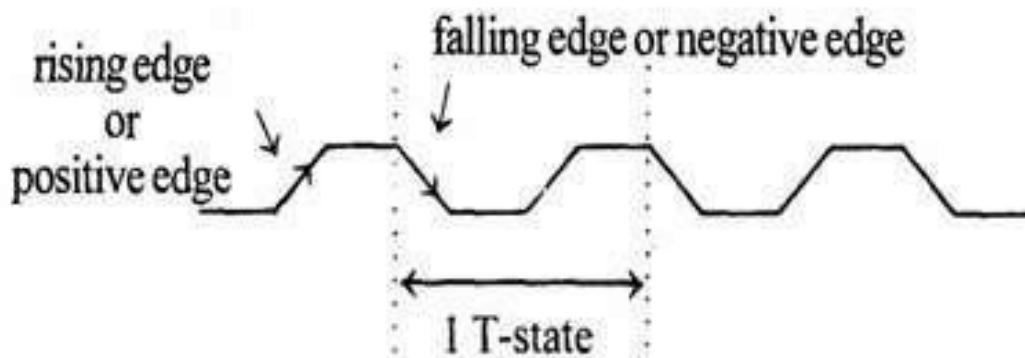
#### **Machine cycles of 8085**

The 8085 microprocessor has 5 (seven) basic machine cycles. They are

1. Opcode fetch cycle (4T)

2. Memory read cycle (3 T)
3. Memory write cycle (3 T)
4. I/O read cycle (3 T)
5. I/O write cycle (3 T)

*Time period,  $T = 1/f$ ; where  $f =$  Internal clock frequency*



**Clock Signal**

### **1. Opcode fetch machine cycle of 8085 :**

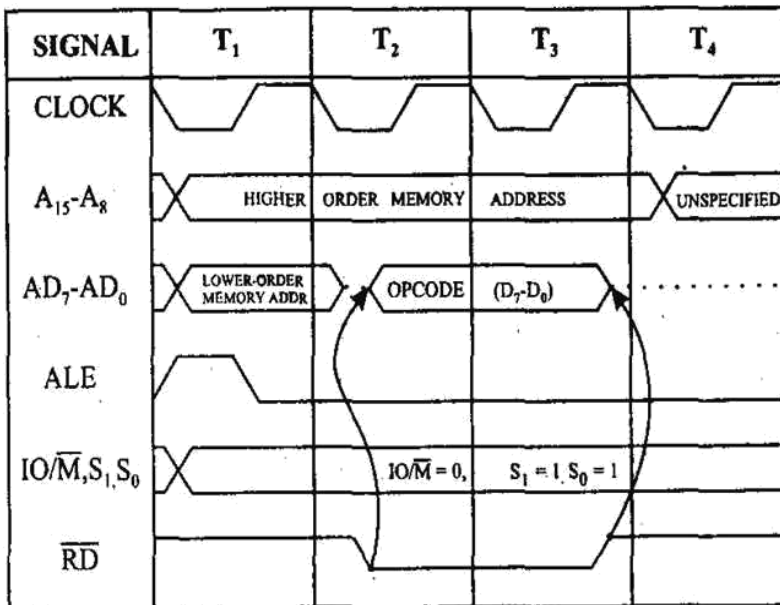
Each instruction of the processor has one byte opcode.

The opcodes are stored in memory. So, the processor executes the opcode fetch machine cycle to fetch the opcode from memory.

Hence, every instruction starts with opcode fetch machine cycle.

The time taken by the processor to execute the opcode fetch cycle is 4T.

In this time, the first, 3 T-states are used for fetching the opcode from memory and the remaining T-states are used for internal operations by the processor.

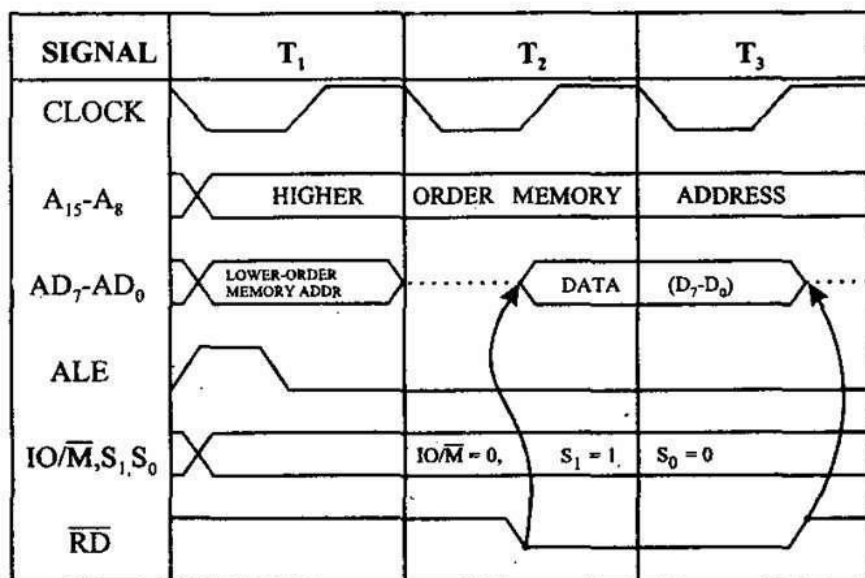


## 2. Memory Read Machine Cycle of 8085:

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.

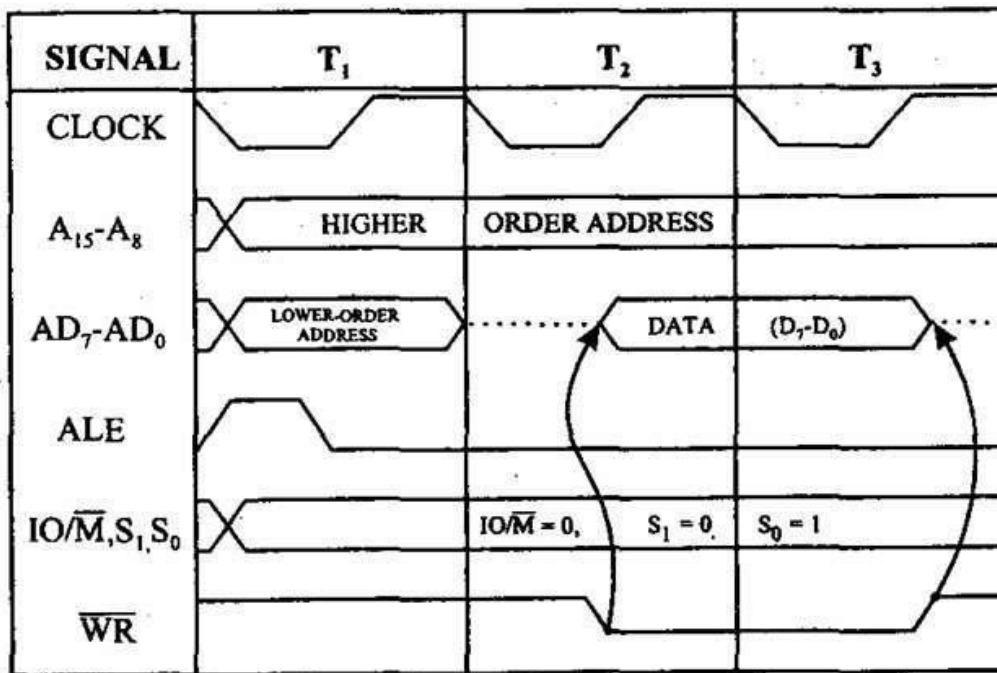


Memory Read Machine Cycle

### 3. Memory Write Machine Cycle of 8085

The memory write machine cycle is executed by the processor to write a data byte in a memory location.

The processor takes, 3T states to execute this machine cycle.



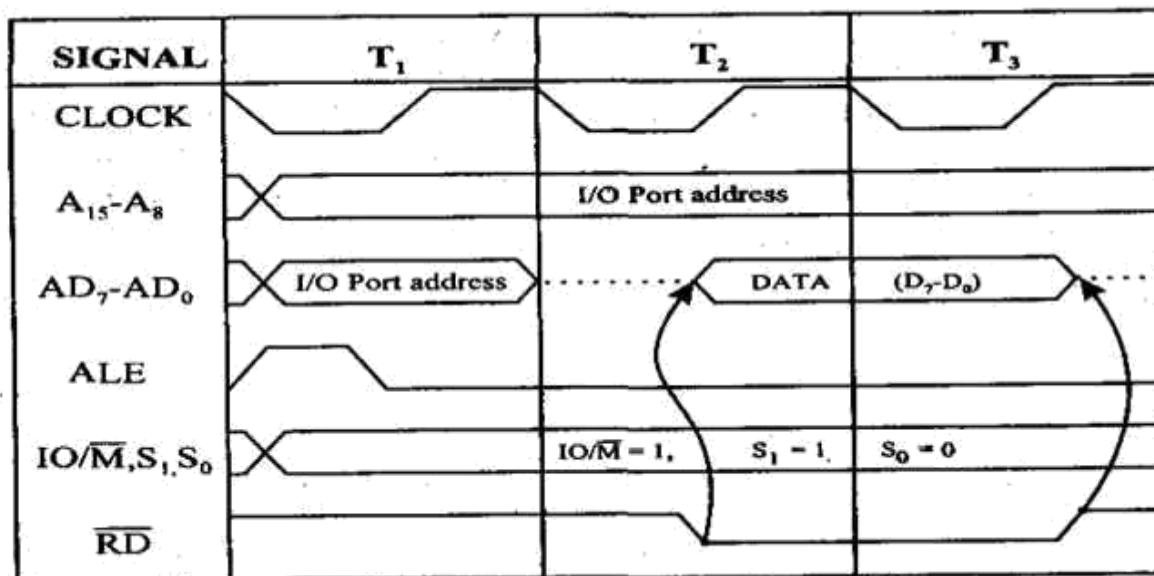
Memory Write Machine Cycle

### 4. I/O Read Cycle of 8085

The I/O Read cycle is executed by the processor to read a data byte from I/O port or from the peripheral, which is I/O, mapped in the system.

The processor takes 3T states to execute this machine cycle.

The IN instruction uses this machine cycle during the execution.



I/O Read Cycle

### Timing diagram for STA 526AH

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

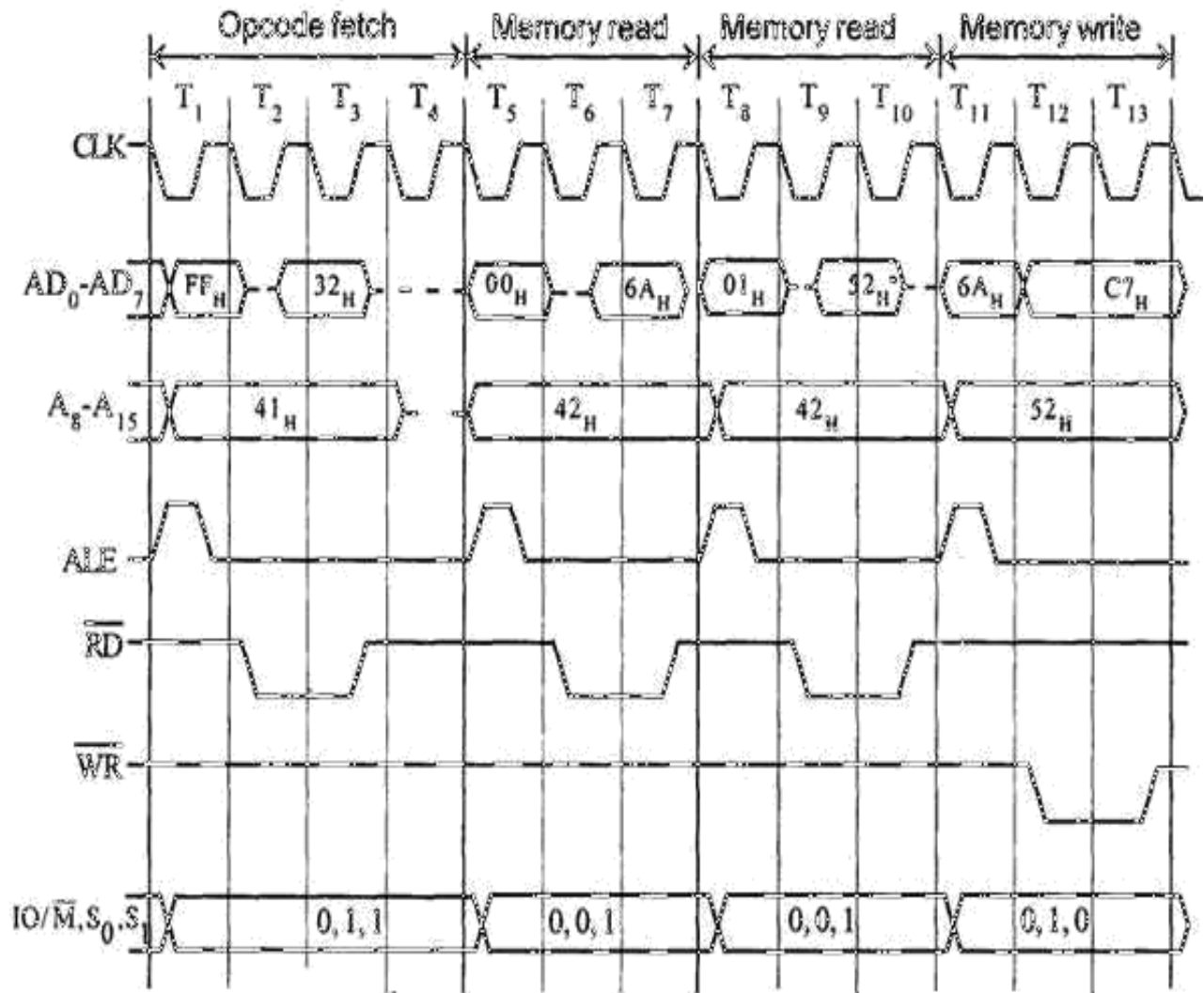
Read the higher order memory address (52).- Memory Read Machine Cycle

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.





**Timing Diagram for STA 526A H**

**Timing diagram for INR M**

Fetching the Opcode 34H from the memory 4105H. (OF cycle)

Let the memory address (M) be 4250H. (MR cycle -To read Memory address and data)

Let the content of that memory is 12H.

Increment the memory content from 12H to 13H. (MW machine cycle)

## 5) **INTERRUPTS:**

Interrupt is signals send by an external device to the processor, to request the processor to perform a particular task or work.

Mainly in the microprocessor based system the interrupts are used for data transfer between the peripheral and the microprocessor.

The processor will check the interrupts always at the 2nd T-state of last machine cycle.

If there is any interrupt it accept the interrupt and send the INTA (active low) signal to the peripheral.

The vectored address of particular interrupt is stored in program counter.

The processor executes an interrupt service routine (ISR) addressed in program counter.

It returned to main program by RET instruction.

### **Types of Interrupts:**

It supports two types of interrupts.

Hardware

Software

### **Software interrupts:**

The software interrupts are program instructions. These instructions are inserted at desired locations in a program.

The 8085 has eight software interrupts from RST 0 to RST 7. The vector address for these interrupts can be calculated as follows.

Interrupt number \* 8 = vector address

For RST 5,  $5 * 8 = 40 = 28H$

**Vector addresses of all interrupts.**

Interrupt	Vector address
RST 0	0000 <sub>H</sub>
RST 1	0008 <sub>H</sub>
RST 2	0010 <sub>H</sub>
RST 3	0018 <sub>H</sub>
RST 4	0020 <sub>H</sub>
RST 5	0028 <sub>H</sub>
RST 6	0030 <sub>H</sub>
RST 7	0038 <sub>H</sub>

**Hardware interrupts:**

An external device initiates the hardware interrupts and placing an appropriate signal at the interrupt pin of the processor.

If the interrupt is accepted then the processor executes an interrupt service routine.

The 8085 has five hardware interrupts

- (1) TRAP      (2) RST 7.5      (3) RST 6.5      (4) RST 5.5      (5) INTR

**(1)TRAP:**

This interrupt is a non-maskable interrupt. It is unaffected by any mask or interrupt enable.

TRAP has the highest priority and vectored interrupt.

TRAP interrupt is edge and level triggered. This means that the TRAP must go high and remain high until it is acknowledged.

In sudden power failure, it executes a ISR and send the data from main memory to backup memory.

The signal, which overrides the TRAP, is HOLD signal. (i.e., If the processor receives HOLD and TRAP at the same time then HOLD is recognized first and then TRAP is recognized).

There are two ways to clear TRAP interrupt.

- 1.By resetting microprocessor (External signal)
- 2.By giving a high TRAP ACKNOWLEDGE (Internal signal)

### **(2)RST 7.5:**

The RST 7.5 interrupt is a maskable interrupt.

It has the second highest priority.

It is edge sensitive. ie. Input goes to high and no need to maintain high state until it recognized.

Maskable interrupt. It is disabled by,

- 1.DI instruction
- 2.System or processor reset.
- 3.After reorganization of interrupt.

Enabled by EI instruction.

### **(3)RST 6.5 and 5.5:**

The RST 6.5 and RST 5.5 both are level triggered. . ie. Input goes to high and stay high until it recognized.

Maskable interrupt. It is disabled by,

- 1.DI, SIM instruction
- 2.System or processor reset.

3. After reorganization of interrupt.

Enabled by EI instruction.

The RST 6.5 has the third priority whereas RST 5.5 has the fourth priority.

INTR is a maskable interrupt. It is disabled by,

1. DI, SIM instruction
2. System or processor reset.
3. After reorganization of interrupt

Enabled by EI instruction.

Non- vectored interrupt. After receiving INTA (active low) signal, it has to supply the address of ISR.

It has lowest priority.

It is a level sensitive interrupts. ie. Input goes to high and it is necessary to maintain high state until it recognized.

The following sequence of events occurs when INTR signal goes high.

1. The 8085 checks the status of INTR signal during execution of each instruction.
2. If INTR signal is high, then 8085 complete its current instruction and sends active low interrupt acknowledge signal, if the interrupt is enabled.
3. In response to the acknowledge signal, external logic places an instruction OP CODE on the data bus. In the case of multibyte instruction, additional interrupt acknowledge machine cycles are generated by the 8085 to transfer the additional bytes into the microprocessor.
4. On receiving the instruction, the 8085 save the address of next instruction on

stack and execute received instruction.

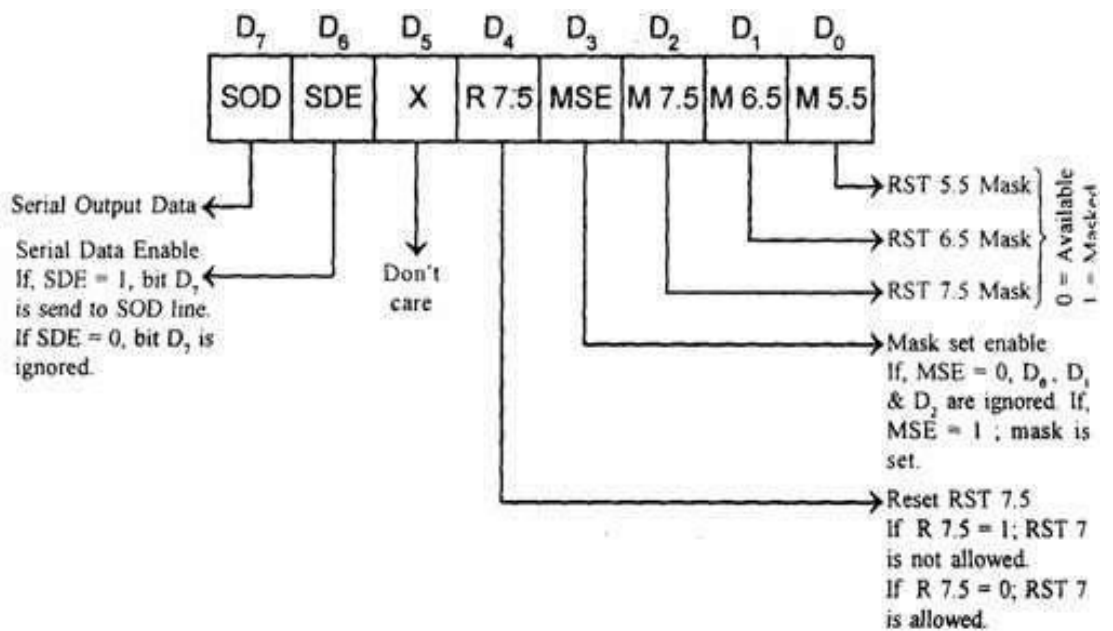
### SIM and RIM for interrupts:

The 8085 provide additional masking facility for RST 7.5, RST 6.5 and RST 5.5 using

SIM instruction.

The status of these interrupts can be read by executing RIM instruction.

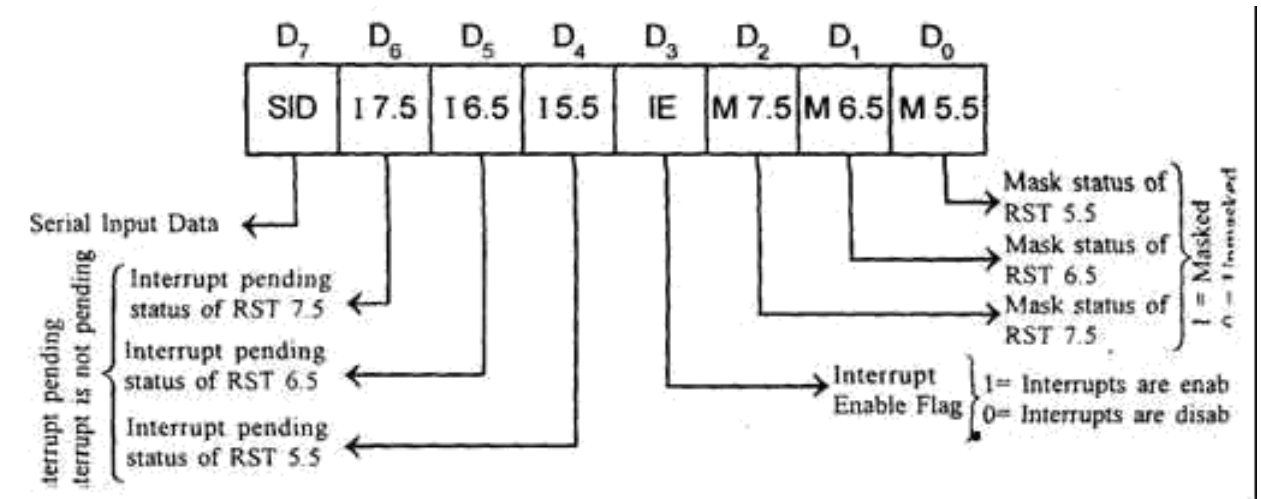
The masking or unmasking of RST 7.5, RST 6.5 and RST 5.5 interrupts can be performed by moving an 8-bit data to accumulator and then executing SIM instruction.



### 8 bit data to be loaded into the Accumulator

The status of pending interrupts can be read from accumulator after executing RIM instruction.

When RIM instruction is executed an 8-bit data is loaded in accumulator, which can be interpreted as shown in fig.



### Format of 8 bit data in Accumulator after executing RIM Instruction 8085 Bus Structure

#### • Address Bus :

- Consists of 16 address lines: A0 – A15
- Address locations: 0000 (hex) – FFFF (hex)
- Can access 64K ( = 2<sup>16</sup> ) bytes of memory, each byte has 8 bits
- Can access 64K × 8 bits of memory
- Use memory to map I/O, Same instructions to use for
- accessing I/O devices and memory

#### • Data Bus :

- Consists of 8 data lines: D0 – D7
- Operates in bidirectional mode
- The data bits are sent from the MPU to I/O & vice versa
- Data range: 00 (hex) – FF (hex)

#### • Control Bus:

- – Consists of various lines carrying the control signals such as read / write enable, flag bits.

## UNIT II

### PROGRAMMING OF 8085 MICROPROCESSOR

#### 1) INSTRUCTION FORMAT

An **instruction** is a command to the microprocessor to perform a given task on a specified data. Each instruction has two parts: one is task to be performed, called the **operation code** (opcode), and the second is the data to be operated on, called the **operand**. The operand (or data) can be specified in various ways. It may include 8-bit (or 16-bit) data, an internal register, a memory location, or 8-bit (or 16-bit) address. In some instructions, the operand is implicit.

#### **Instruction word size**

The 8085 instruction set is classified into the following three groups according to word size:

1. One-word or 1-byte instructions
2. Two-word or 2-byte instructions
3. Three-word or 3-byte instructions

In the 8085, "byte" and "word" are synonymous because it is an 8-bit microprocessor. However, instructions are commonly referred to in terms of bytes rather than words.

#### **2.1.1 One-Byte Instructions**

A 1-byte instruction includes the opcode and operand in the same byte.

Operand(s) are internal register and are coded into the instruction

Task	Op code	Operand	Binary Code	Hex Code
Copy the contents of the accumulator in the register C.	MOV	C,A	0100 1111	4FH
Add the contents of register B to the contents of the accumulator.	ADD	B	1000 0000	80H
Invert (compliment) each bit in the accumulator.	CMA		0010 1111	2FH

These instructions are 1-byte instructions performing three different tasks. In the first instruction, both operand registers are specified. In the second instruction, the operand B is specified and the accumulator is assumed. Similarly, in the third instruction, the accumulator is assumed to be the implicit operand. These instructions are stored in 8-bit binary format in memory; each requires one memory location.

#### **MOV rd, rs**

rd <-- rs copies contents of rs into rd.

Coded as 01 ddd sss where ddd is a code for one of the 7 general registers which is



the destination of the data, sss is the code of the source register.

Example: MOV A,B

Coded as 01111000 = 78H = 170 octal (octal was used extensively in instruction design of such processors).

### ADD r

$A \leftarrow A + r$

### Two-Byte Instructions

In a two-byte instruction, the first byte specifies the operation code and the second byte specifies the operand. Source operand is a data byte immediately following the opcode. For example

Task	Opcode	Operand	Binary Code	Hex Code	
Load an 8-bit data byte in the accumulator.	MVI	A, Data	0011 1110	3E	First Byte
			DATA	Data	Second Byte

The instruction would require two memory locations to store in memory.

### MVI r,data

$r \leftarrow \text{data}$

Example: MVI A,30H coded as 3EH 30H as two contiguous bytes. This is an example of immediate addressing.

### ADI data

$A \leftarrow A + \text{data}$

OUT port

0011 1110

DATA

where port is an 8-bit device address.  $(\text{Port}) \leftarrow A$ . Since the byte is not the data but points directly to where it is located this is called direct addressing.

### Three-Byte Instructions

In a three-byte instruction, the first byte specifies the opcode, and the following two bytes specify the 16-bit address. Note that the second byte is the low-order address and the third byte is the high-order address. opcode + data byte + data byte

Task	Opcode	Operand	Binary code	Hex Code	
Transfer the program sequence to the memory location 2085H.	JMP	2085H	1100 0011	C3	First byte
			1000 0101	85	Second Byte
			0010 0000	20	Third Byte

This instruction would require three memory locations to store in memory.

Three byte instructions - opcode + data byte + data byte

### **LXI rp, data16**

rp is one of the pairs of registers BC, DE, HL used as 16-bit registers. The two data bytes are 16-bit data in L H order of significance. rp <-- data16

LXI H,0520H coded as 21H 20H 50H in three bytes. This is also immediate addressing.

### **LDA addr**

A <-- (addr) Addr is a 16-bit address in L H order.

Example: LDA 2134H coded as

3AH 34H 21H. This is also an example of direct addressing.

## **2) THE 8085 ADDRESSING MODES**

The instructions MOV B, A or MVI A, 82H are to copy data from a source into a destination. In these instructions the source can be a register, an input port, or an 8-bit number (00H to FFH). Similarly, a destination can be a register or an output port. The sources and destination are operands. The various formats for specifying operands are called the ADDRESSING MODES. For 8085, they are:

1. Immediate addressing.
2. Register addressing.
3. Direct addressing.
4. Indirect addressing.

### **(1)Immediate addressing**

Data is present in the instruction. Load the immediate data to the destination provided.

**Example:** MVI R,data

### **(2)Register addressing**

Data is provided through the registers.

**Example:** MOV Rd, Rs

### **(3)Direct addressing**

Used to accept data from outside devices to store in the accumulator or send the data stored in the accumulator to the outside device. Accept the data from the port 00H and store them into the accumulator or Send the data from the accumulator to the port 01H.

**Example:** IN 00H or OUT 01H

### **(4)Indirect Addressing**

This means that the Effective Address is calculated by the processor. And the contents of the address (and the one following) is used to form a second address. The second address is where the data is stored. Note that this requires several memory accesses; two accesses to retrieve the 16-bit address and a further access (or accesses) to retrieve the data which is to be loaded into the register.

## **3) INSTRUCTION SET CLASSIFICATION**

An **instruction** is a binary pattern designed inside a microprocessor to perform a specific function. The entire group of instructions, called the **instruction set**, determines what functions the microprocessor can perform. These instructions can be classified into the following five functional categories: data transfer (copy) operations, arithmetic operations, logical operations, branching operations, and machine-control operations.

### **Data Transfer Group**

The data transfer instructions move data between registers or between memory and registers.

MOV Move

MVI Move Immediate

LDA Load Accumulator Directly from Memory

STA Store Accumulator Directly in Memory

LHLD Load H & L Registers Directly from Memory

SHLD Store H & L Registers Directly in Memory

An 'X' in the name of a data transfer instruction implies that it deals with a register pair (16-bits);

LXI Load Register Pair with Immediate data

LDAX Load Accumulator from Address in Register Pair

STAX Store Accumulator in Address in Register Pair

XCHG Exchange H & L with D & E XTHL Exchange

Top of Stack with H & L

### **Arithmetic Group**

The arithmetic instructions add, subtract, increment, or decrement data in registers or memory.

ADD Add to Accumulator

ADI Add Immediate Data to Accumulator

ADC Add to Accumulator Using Carry Flag

ACI Add Immediate data to Accumulator Using Carry

SUB Subtract from Accumulator

SUI Subtract Immediate Data from Accumulator

SBB Subtract from Accumulator Using Borrow (Carry) Flag

SBI Subtract Immediate from Accumulator Using Borrow (Carry) Flag

INR Increment Specified Byte by One DCR Decrement Specified Byte  
by One

INX Increment Register Pair by One

DCX Decrement Register Pair by One

DAD Double Register Add; Add Content of Register  
Pair to H & L Register Pair

### **Logical Group**

This group performs logical (Boolean) operations on data in registers and memory and on

condition flags.

The logical AND, OR, and Exclusive OR instructions enable you to set specific bits in the

accumulator ON or OFF.

ANA Logical AND with Accumulator

ANI Logical AND with Accumulator Using Immediate Data

ORA Logical OR with Accumulator

OR Logical OR with Accumulator Using Immediate

Data XRA Exclusive Logical OR with Accumulator XRI

Exclusive OR Using Immediate Data

The Compare instructions compare the content of an 8-bit value with the contents of the accumulator;

CMP Compare

CPI Compare Using Immediate Data

The rotate instructions shift the contents of the accumulator one bit position to the left or right:

RLC Rotate Accumulator Left

RRC Rotate Accumulator Right

RAL Rotate Left Through Carry

RAR Rotate Right Through Carry

Complement and carry flag instructions:

CMA Complement Accumulator

CMC Complement Carry Flag

STC Set Carry Flag

### **Branch Group**

The branching instructions alter normal sequential program flow, either unconditionally or

conditionally. The unconditional branching instructions are as follows:

JMP Jump

CALL Call

RET Return

Conditional branching instructions examine the status of one of four condition flags to determine

whether the specified branch is to be executed. The conditions that may be specified are as

follows:

NZ Not Zero ( $Z = 0$ )

Z Zero ( $Z = 1$ )

NC No Carry ( $C = 0$ )

C Carry ( $C = 1$ )

PO Parity Odd ( $P = 0$ )

PE Parity Even ( $P = 1$ )

P Plus ( $S = 0$ )

M Minus ( $S = 1$ )

Thus, the conditional branching instructions are specified as follows:

Jumps Calls Returns

C CC RC (Carry)

INC CNC RNC (No Carry)

JZ CZ RZ (Zero)

JNZ CNZ RNZ (Not Zero)

JP CP RP (Plus)

JM CM RM (Minus)

JPE CPE RPE (Parity Even)

JPO CPO RPO (Parity Odd)

Two other instructions can affect a branch by replacing the contents or the program counter:

PCHL Move H & L to Program Counter

RST Special Restart Instruction Used

with Interrupts

POP Pop Two Bytes of Data off the Stack

XTHL Exchange Top of Stack with H & L

SPHL Move content of H & L to Stack Pointer

**I/O instructions**

IN Initiate Input Operation

OUT Initiate Output Operation

**Machine Control instructions**

EI Enable Interrupt System

DI Disable Interrupt System

HLT Halt

NOP No Operation

**SAMPLE PROGRAM**

(1)Write an assembly program to add two numbers Program

MVI D, 8BH

MVI C, 6FH

MOV A, C

1100 0011

1000 0101

0010 0000

ADD D

OUT PORT1

HLT

(2)Write an assembly program to multiply a number by 8 Program

MVI A, 30H

RRC

RRC

RRC

OUT PORT1

HLT

(3)Write an assembly program to find greatest between two numbers Program

MVI B, 30H

MVI C, 40H

MOV A, B

CMP C

JZ EQU

JC GRT

OUT PORT1

HLT

EQU: MVI A, 01H

OUT PORT1

```
HLT
GRT: MOV A, C
OUT PORT1
HLT
```

## **2.5 Programming using Loop structure with Counting and Indexing**

### **(i) 16 bit Multiplication**

```
ADDRESS
LABEL
MNEMONICS
OPCODE
START
L1
L2
LHLD 4200
SPHL
LHLD 4202
XCHG LXI
H,0000 LXI
B,0000
DAD SP
JNC L2
INX B DCX
D MOV
A,E ORA D
JNZ L1
SHLD 4204
MOV L,C
MOV H,B
SHLD 4206
HLT
```

### **(ii) Finding the maximum number in the given array**

```
ADDRESS
LABEL
MNEMONICS
OPCODE
START
L2
L3
L1
LDA 4500
MOV C, A
LXI H, 4501
MOV A,
M DCR C
INX H
```



JZ L1  
CMP M  
JC L2  
JMP L3  
STA 4520  
HLT

(iii) To sort the array of data in ascending order

ADDRESS  
LABEL  
MNEMONICS  
START  
L3  
L2  
L1  
MVI B, 00  
LXI H, 4200  
MOV C, M  
DCR C  
INX H  
MOV A,  
M INX H  
CMP M JC  
L1 MOV  
D, M  
MOV M,  
A DCX H  
MOV M,  
D INX H  
MVI B, 01  
DCR C  
JNZ L2  
DCR B  
JZ L3  
HLT

## **2.6 Programming using subroutine Instructions**

### **Generation of Square waveform using DAC**

ADDRESS  
LABEL  
MNEMONICS  
START  
DELAY  
L2  
L1  
MVI A,00H  
OUT C8 CALL  
DELAY

```
MVI A,FF
OUT C8
CALL DELAY
JMP START
MVI B,05H
MVI C,FF
DCR C
JNZ L1
DCR B
JNL L2
RET
```

## **2.7 Programming using Look up table**

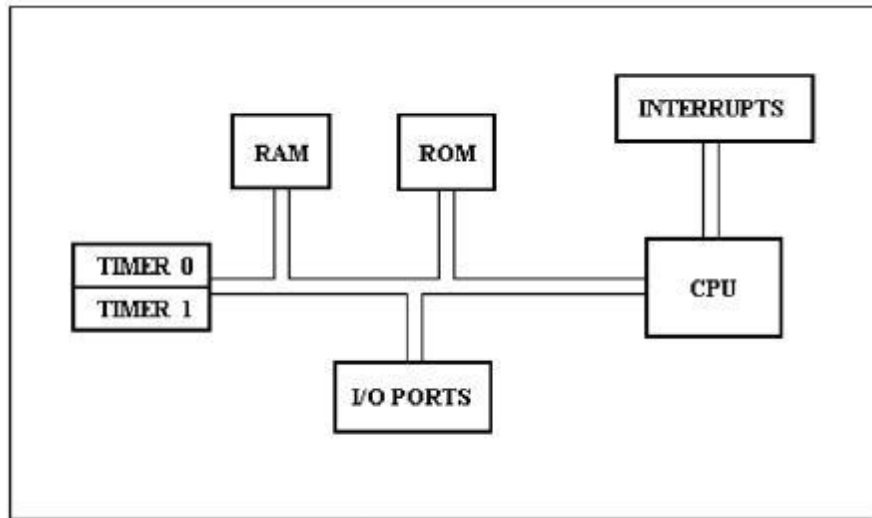
```
ADDRESS
LABEL
MNEMONICS
OPCODE
START
L2
MVI B,08
MVI A,00(DISPLAY MODE
SETUP)
OUT C2
MVI A,CC(CLEAR DISPLAY)
OUT C2
MVI A,90(WRITE DISPLAY
RAM)
OUT C2
MVI A, FF(CLEAR DISPLAY
RAM)
OUT C0
DCR B
JNZ L1
IN C2
ANI 07
JZ L2
MVI A, 40(SET TO READ FIFO
RAM)
OUT C2
IN C0
ANI 0F
MOV L, A
MVI H, 42
MOV A, M
OUT C0
JMP L2
LOOKUP TABLE
```

4200 0C 9F 4A 0B  
4204 99 29 28 8F  
4208 08 09 88 38  
420C 6C 1A 68 E8

## UNIT III

### 8051 MICROCONTROLLER

#### 1) ARCHITECTURE OF 8051:



#### **Memory Organization**

- Logical separation of program and data memory
- Separate address spaces for Program (ROM) and Data (RAM) Memory

Allow Data Memory to be accessed by 8-bit addresses quickly and manipulated by 8-bit CPU

Program Memory

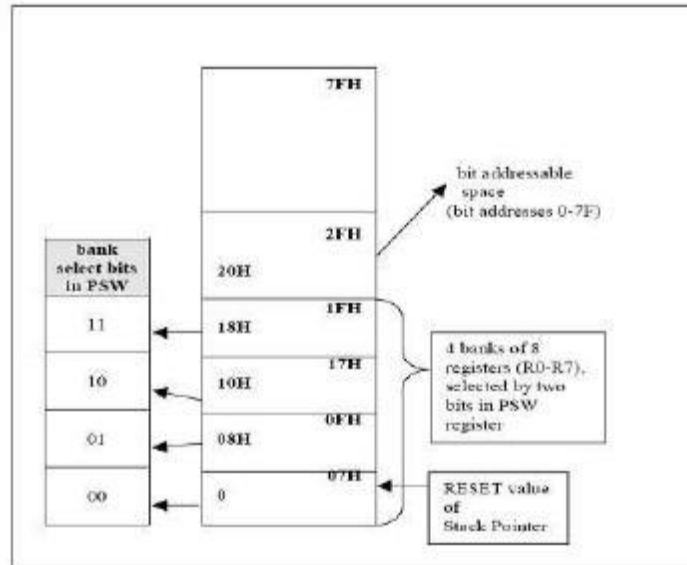
- Only be read, not written to
- The address space is 16-bit, so maximum of 64K bytes -
- Up to 4K bytes can be on-chip (internal) of 8051 core

- PSEN (Program Store Enable) is used for access to external Program Memory

Data Memory

- Includes 128 bytes of on-chip Data Memory which are more easily accessible directly by its instructions
- There is also a number of Special Function Registers (SFRs)

-Internal Data Memory contains four banks of eight registers and a special 32-byte long segment which is bit addressable by 8051 bit-instructions -External memory of maximum 64K bytes is accessible by “movx”



### Interrupt Structure

- The 8051 provides 4 interrupt sources
- Two external interrupts
- Two timer interrupts

### Port Structure

- The 8051 contains four I/O ports
- All four ports are bidirectional
- Each port has SFR (Special Function Registers P0 through P3) which works like a latch, an output driver and an input buffer
- Both output driver and input buffer of Port 0 and output driver of Port 2 are used for accessing external memory
- Accessing external memory works like this
- Port 0 outputs the low byte of external memory address (which is timemultiplexed with the byte being written or read)
- Port 2 outputs the high byte (only needed when the address is 16 bits wide)
- Port 3 pins are multifunctional
- The alternate functions are activated with the 1 written in the corresponding bit in the port SFR

Port Pin	Alternate Function
P3.2	~INT0 (external interrupt)
P3.3	~INT1 (external interrupt)
P3.4	T0 (Timer/Counter 0 external input)
P3.5	T1 (Timer/Counter 1 external input)
P3.6	~WR (external Data Memory writestrobe)
P3.7	~RD (external DataMemory readstrobe)

### Timer/Counter

- The 8051 has two 16-bit Timer/Counter registers
- Timer 0
- Timer 1
- Both can work either as timers or event counters
- Both have four different operating modes

### 2) INSTRUCTION FORMAT

An **instruction** is a command to the microprocessor to perform a given task on a specified data. Each instruction has two parts: one is task to be performed, called the **operation code** (opcode), and the second is the data to be operated on, called the **operand**. The operand (or data) can be specified in various ways. It may include 8-bit (or 16-bit ) data, an internal register, a memory location, or 8-bit (or 16-bit) address.

In some instructions, the operand is implicit.

#### Instruction word size

The 8051 instruction set is classified into the following three groups according to word size:

1. One-word or 1-byte instructions
2. Two-word or 2-byte instructions
3. Three-word or 3-byte instructions

#### One-Byte Instructions

A 1-byte instruction includes the opcode and operand in the same byte.

Operand(s) are internal register and are coded into the instruction.

These instructions are 1-byte instructions performing three different tasks. In the first instruction, both operand registers are specified. In the second instruction, the operand B is specified and the accumulator is assumed. Similarly, in the third instruction, the accumulator is assumed to be the implicit operand. These instructions are stored in 8-bit binary format in memory; each requires one memory location.

#### Two-Byte Instructions

In a two-byte instruction, the first byte specifies the operation code and the second byte specifies the operand. Source operand is a data byte immediately following the opcode.

#### Three-Byte Instructions

In a three-byte instruction, the first byte specifies the opcode, and the following two bytes specify the 16-bit address. Note that the second byte is the low-order address and the third byte is the high-order address.

### 3) Addressing Modes of 8051

The 8051 provides a total of five distinct addressing modes.

- (1) immediate
- (2) register
- (3) direct
- (4) register indirect
- (5) indexed

#### (1) Immediate Addressing Mode

- The operand comes immediately after the op-code.
- The immediate data must be preceded by the pound sign, "#".

```
MOV A, #25H      ;load 25H into A
MOV R4, #62      ;load the decimal value 62 into R4
MOV B, #40H      ;load 40H into B
MOV DPTR, #4521H ;DPTR=4512H
```

#### (2) Register Addressing Mode

- Register addressing mode involves the use of registers to hold the data to be manipulated

```
MOV A, R0 ;copy the contents of R0 into A
MOV R2, A ;copy the contents of A into R2
ADD A, R5 ;add the contents of R5 to contents of A
ADD A, R7 ;add the contents of R7 to contents of A
MOV R6, A ;save accumulator in R6
```

#### (3) Direct Addressing Mode

- It is most often used to access RAM locations 30 - 7FH.

-This is due to the fact that register bank locations are accessed by the register names of R0 - R7.

-There is no such name for other RAM locations so must use direct addressing

-In the direct addressing mode, the data is in a RAM memory location whose address is known, and this address is given as a part of the instruction

```

MOV A, 4      ;is same as
MOV A, R4     ;which means copy R4 into A

MOV A, 7      ;is same as
MOV A, R7     ;which means copy R7 into A

```

#### (4) Register Indirect Addressing Mode

- A register is used as a pointer to the data.
- If the data is inside the CPU, only registers R0 and R1 are used for this purpose. ● R2 - R7 cannot be used to hold the address of an operand located in RAM when using indirect addressing mode.
- When R0 and R1 are used as pointers they must be preceded by the @ sign.

#### (5) Indexed Addressing Mode

- Indexed addressing mode is widely used in accessing data elements of look-up table entries located in the program ROM space of the 8051.
- The instruction used for this purpose is :  
MOVC A, @ A+DPTR
- The 16-bit register DPTR and register A are used to form the address of the data element stored in on-chip ROM.
- Because the data elements are stored in the program (code) space ROM of the 8051, the instruction MOVC is used instead of MOV. The "C" means code.
- In this instruction the contents of A are added to the 16-bit register DPTR to form the 16-bit address of the needed data

#### 4) Interrupt Structure

- 8051 provides 4 interrupt sources
- 2 external interrupts
- 2 timer interrupts
- They are controlled via two SFRs, IE and IP
- Each interrupt source can be individually enabled or disabled by setting or clearing a bit in IE (Interrupt Enable). IE also exists a global disable bit, which can be cleared to disable all interrupts at once
- Each interrupt source can also be individually set to one of two priority levels by setting or clearing a bit in IP (Interrupt Priority)
- A low-priority interrupt can be interrupted by high-priority interrupt, but not by another low-priority one
- A high-priority interrupt can't be interrupted by any other interrupt source
- If interrupt requests of the same priority level are received simultaneously, an internal polling sequence determines which request is serviced, so within each priority level there is a second priority structure
- This internal priority structure is determined by the polling sequence, shown in the following table



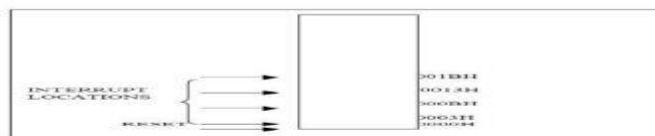
Source	Priority Within Level
IE0	highest
TF0	
IE1	lowest
TF1	

### External Interrupt

- External interrupts  $\sim$ INT0 and  $\sim$ INT1 have two ways of activation
- Level-activated
- Transition-activated
- This depends on bits IT0 and IT1 in TCON
- The flags that actually generate these interrupts are bits IE0 and IE1 in TCON
- On-chip hardware clears that flag that generated an external interrupt when the service routine is vectored to, but only if the interrupt was transition-activated
- When the interrupt is level-activated, then the external requesting source is controlling the request flag, not the on-chip hardware

### Handling Interrupt

- When interrupt occurs (or correctly, when the flag for an enabled interrupt is found to be set (1)), the interrupt system generates an LCALL to the appropriate location in Program Memory, unless some other conditions block the interrupt
- Several conditions can block an interrupt
- An interrupt of equal or higher priority level is already in progress
- The current (polling) cycle is not the final cycle in the execution of the instruction in progress
- The instruction in progress is RETI or any write to IE or IP registers
- If an interrupt flag is active but not being responded to for one of the above conditions, must be still active when the blocking condition is removed, or the denied interrupt will not be serviced
- Next step is saving the registers on stack. The hardware-generated LCALL causes only the contents of the Program Counter to be pushed onto the stack, and reloads the PC with the beginning address of the service routine
- In some cases it also clears the flag that generated the interrupt, and in other cases it doesn't. It clears an external interrupt flag (IE0 or IE1) only if it was transitionactivated.
- Having only PC be automatically saved gives programmer more freedom to decide how much time to spend saving other registers. Programmer must also be more careful with proper selection, which register to save.
- The service routine for each interrupt begins at a fixed location. The interrupt locations are spaced at 8-byte interval, beginning at 0003H for External Interrupt 0, 000BH for Timer 0, 0013H for External Interrupt 1 and 001BH for Timer 1.



## I/O Ports

- The 8051 contains four I/O ports
- All four ports are bidirectional
- Each port has SFR (Special Function Registers P0 through P3) which works like a latch, an output driver and an input buffer
- Both output driver and input buffer of Port 0 and output driver of Port 2 are used for accessing external memory
- Accessing external memory works like this
- Port 0 outputs the low byte of external memory address (which is timemultiplexed with the byte being written or read)
- Port 2 outputs the high byte (only needed when the address is 16 bits wide) ■ Port 3 pins are multifunctional
- The alternate functions are activated with the 1 written in the corresponding bit in the port SFR

## Timers

The 8051 comes equipped with two timers, both of which may be controlled, set, read, and configured individually. The 8051 timers have three general functions: 1) Keeping time and/or calculating the amount of time between events, 2) Counting the events themselves, or 3) Generating baud rates for the serial port.

one of the primary uses of timers is to measure time. We will discuss this use of timers first and will subsequently discuss the use of timers to count events. When a timer is used to measure time it is also called an "interval timer" since it is measuring the time of the interval between two events.

## Timer SFR

8051 has two timers which each function essentially the same way. One timer is TIMER0 and the other is TIMER1. The two timers share two SFRs (TMOD and TCON) which control the timers, and each timer also has two SFRs dedicated solely to itself (TH0/TL0 and TH1/TL1).

Table 4.4 SFR

### 13-bit Time Mode (mode 0)

Timer mode "0" is a 13-bit timer. This is a relic that was kept around in the 8051 to maintain compatability with its predecessor, the 8048. Generally the 13-bit timer mode is not used in new development.

When the timer is in 13-bit mode, TLx will count from 0 to 31. When TLx is incremented from 31, it will "reset" to 0 and increment THx. Thus, effectively, only 13 bits of the two timer bytes are being used: bits 0-4 of TLx and bits 0-7 of THx. This also means, in essence, the timer can only contain 8192 values. If you set a 13-bit timer to 0, it will overflow back to zero 8192 Machine cycles later.

Again, there is very little reason to use this mode and it is only mentioned so you wont be surprised if you ever end up analyzing archaic code which has been passed down through the generations (a generation in a programming shop is often on the order of about 3 or 4 months).

SFR Name Description SFR Address

TH0 Timer 0 High Byte 8Ch  
TL0 Timer 0 Low Byte 8Ah  
TH1 Timer 1 High Byte 8Dh  
TL1 Timer 1 Low Byte 8Bh  
TCON Timer Control 88h  
TMOD Timer Mode 89h

### **16-bit Time Mode (mode 1)**

Timer mode "1" is a 16-bit timer. This is a very commonly used mode. It functions just like 13-bit mode except that all 16 bits are used.

TLx is incremented from 0 to 255. When TLx is incremented from 255, it resets to 0 and causes THx to be incremented by 1. Since this is a full 16-bit timer, the timer may contain up to 65536 distinct values. If you set a 16-bit timer to 0, it will overflow back to 0 after 65,536 machine cycles.

### **8-bit Time Mode (mode 2)**

Timer mode "2" is an 8-bit auto-reload mode. What is that, you may ask? Simple. When a timer is in mode 2, THx holds the "reload value" and TLx is the timer itself. Thus, TLx starts counting up. When TLx reaches 255 and is subsequently incremented, instead of resetting to 0 (as in the case of modes 0 and 1), it will be reset to the value stored in THx.

### **Split Timer Mode (mode 3)**

Timer mode "3" is a split-timer mode. When Timer 0 is placed in mode 3, it essentially becomes two separate 8-bit timers. That is to say, Timer 0 is TL0 and Timer 1 is TH0. Both timers count from 0 to 255 and overflow back to 0. All the bits that are related to Timer 1 will now be tied to TH0.

While Timer 0 is in split mode, the real Timer 1 (i.e. TH1 and TL1) can be put into modes 0, 1 or 2 normally--however, you may not start or stop the real timer 1 since the bits that do that are now linked to TH0. The real timer 1, in this case, will be incremented every machine cycle no matter what.

## **USING TIMERS AS EVENT COUNTERS**

We've discussed how a timer can be used for the obvious purpose of keeping track of time. However, the 8051 also allows us to use the timers to count events. How can this be useful? Let's say you had a sensor placed across a road that would send a pulse every time a car passed over it. This could be used to determine the volume of traffic on the road. We could attach this sensor to one of the 8051's I/O lines and constantly monitor it, detecting when it pulsed high and then incrementing our counter when it went back to a low state. This is not terribly difficult, but requires some code. Let's say we hooked the sensor to P1.0; the code to count cars passing would look something like this: `JNB P1.0,$ ;If a car hasn't raised the signal, keep waiting JB P1.0,$ ;The line is high which means the car is on the sensor right now INC COUNTER ;The car has passed completely, so we count it`

## **Serial Communication**

Some of the external I/O devices receive only the serial data. Normally serial communication is used in the Multi Processor environment. 8051 has two pins for serial communication.

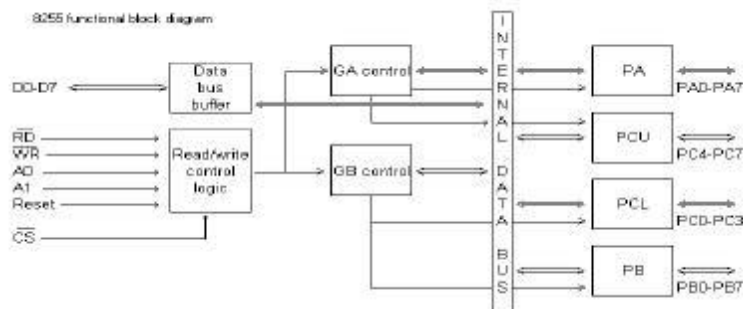
(1) SID- Serial Input data.

(2) SOD-Serial Output data.

## UNIT IV PERIPHERAL INTERFACING

### 1) Architecture of 8255

The parallel input-output port chip 8255 is also called as programmable peripheral input output port. The Intel's 8255 is designed for use with Intel's 8-bit, 16-bit and higher capability microprocessors. It has 24 input/output lines which may be individually programmed in two groups of twelve lines each, or three groups of eight lines. The two groups of I/O pins are named as Group A and Group B. Each of these two groups contains a subgroup of eight I/O lines called as 8-bit port and another subgroup of four lines or a 4-bit port. Thus Group A contains an 8-bit port A along with a 4-bit port C upper. The port A lines are identified by symbols PA0-PA7 while the port C lines are identified as PC4-PC7. Similarly, Group B contains an 8-bit port B, containing lines PB0-PB7 and a 4-bit port C with lower bits PC0- PC3. The port C upper and port C lower can be used in combination as an 8-bit port C. Both the port C are assigned the same address. Thus one may have either three 8-bit I/O ports or two 8-bit and two 4-bit ports from 8255. All of these ports can function independently either as input or as output ports. This can be achieved by programming the bits of an internal register of 8255 called as control word register ( CWR ). This buffer receives or transmits data upon the execution of input or output instructions by the microprocessor. The control words or status information is also transferred through the buffer.

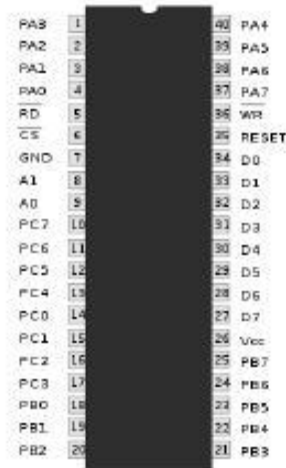


### 2) PIN DIAGRAM OF 8255

The signal description of 8255 are briefly presented as follows :

- **PA7-PA0**: These are eight port A lines that acts as either latched output or buffered input lines depending upon the control word loaded into the control word register.
- **PC7-PC4** : Upper nibble of port C lines. They may act as either output latches or input buffers lines. This port also can be used for generation of handshake lines in mode 1 or mode 2.
- **PC3-PC0** : These are the lower port C lines, other details are the same as PC7-PC4 lines.
- **PB0-PB7** : These are the eight port B lines which are used as latched output lines or buffered input lines in the same way as port A.
- **RD** : This is the input line driven by the microprocessor and should be low to indicate read operation to 8255.
- **WR** : This is an input line driven by the microprocessor. A low on this line indicates write operation.

**CS** : This is a chip select line. If this line goes low, it enables the 8255 to respond to RD and WR signals, otherwise RD and WR signal are neglected



**A1-A0** : These are the address input lines and are driven by the microprocessor. These lines A1-A0 with RD, WR and CS from the following operations for 8255. These address lines are used for addressing any one of the four registers,i.e. three ports and a control word register as given in table below

- In case of 8086 systems, if the 8255 is to be interfaced with lower order data bus, the A0 and A1 pins of 8255 are connected with A1 and A2 respectively.

**D0-D7** :

These are the data bus lines those carry data or control word to/from the Microprocessor.

**RESET** :

A logic high on this line clears the control word register of 8255. All ports are set as input ports by default after reset.

**Operational Modes of 8255**

There are two main operational modes of 8255:

1. Input/output mode
2. Bit set/reset mode

**Input/Output Mode**

There are three types of the input/output mode. They are as follows:

• **Mode 0**

In this mode, the ports can be used for simple input/output operations without handshaking. If both port A and B are initialized in mode 0, the two halves of port C can be either used together as an additional 8-bit port, or they can be used as individual 4-bit ports. Since the two halves of port C are independent, they may be used such that one-half is initialized as an input port while the other half is initialized as an output port. The input output features in mode 0 are as follows:

1. O/p are latched.
2. I/p are buffered not latched.
3. Port do not have handshake or interrupt capability.

### • **Mode 1**

When we wish to use port A or port B for handshake (strobed) input or output operation, we initialise that port in mode 1 (port A and port B can be initialised to operate in different modes, ie, for eg, port A can operate in mode 0 and port B in mode 1). Some of the pins of port C function as handshake lines.

For port B in this mode (irrespective of whether is acting as an input port or output port), PC0, PC1 and PC2 pins function as handshake lines.

If port A is initialised as mode 1 input port, then, PC3, PC4 and PC5 function as handshake signals. Pins PC6 and PC7 are available for use as input/output lines.

The mode 1 which supports handshaking has following features: 1. Two ports i.e. port A and B can be use as 8-bit i/o port. 2. Each port uses three lines of port c as handshake signal and remaining two signals can be function as i/o port. 3. interrupt logic is supported. 4. Input and Output data are latched.

### • **Mode 2**

Only group A can be initialised in this mode. Port A can be used for bidirectional handshake data transfer. This means that data can be input or output on the same eight lines (PA0 - PA7). Pins PC3 - PC7 are used as handshake lines for port A. The remaining pins of port C (PC0 - PC2) can be used as input/output lines if group B is initialised in mode 0. In this mode, the 8255 may be used to extend the system bus to a slave microprocessor or to transfer data bytes to and from a floppy disk controller.

### **Bit Set/Reset (BSR) mode**

In this mode only port b can be used (as an output port). Each line of port C (PC0 - PC7) can be set/reset by suitably loading the command word register. no effect occurs in input-output mode. The individual bits of port c can be set or reset by sending the signal OUT instruction to the control register.

## **3)PROGRAMMABLE INTERRUPT CONTROLLER(8259)**

8 levels of interrupts.

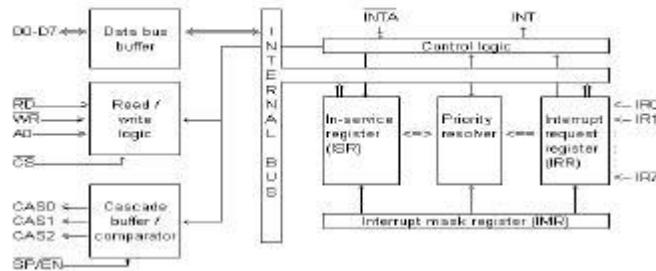
- Can be cascaded in master-slave configuration to handle 64 levels of interrupts.
- Internal priority resolver.
- Fixed priority mode and rotating priority mode.
- Individually maskable interrupts.
- Modes and masks can be changed dynamically.
- Accepts IRQ, determines priority, checks whether incoming priority > current level being serviced, issues interrupt signal.
- In 8085 mode, provides 3 byte CALL instruction. In 8086 mode, provides 8 bit vector number.
- Polled and vectored mode.
- Starting address of ISR or vector number is programmable.
- No clock required

CS	1	28	Vcc
WR	2	27	A0
RD	3	26	INTA
D7	4	25	IR7
D6	5	24	IR6
D5	6	23	IR5
D4	7	E259	IR4
D3	8	PIC	IR3
D2	9		IR2
D1	10		IR1
D0	11		IR0
CAS0	12		INT
CAS1	13		SP/EN
gnd	14		CAS2

### Pin diagram

D0-D7	Bi-directional, tristated, buffered data lines. Connected to data bus directly or through buffers
RD-bar	Active low read control
WR-bar	Active low write control
A0	Address input line, used to select control register
CS-bar	Active low chip select
CAS0-2	Bi-directional, 3 bit cascade lines. In master mode, PIC places slave ID no. on these lines. In slave mode, the PIC reads slave ID no. from master on these lines. It may be regarded as slave-select.
SP-bar / EN-bar	Slave program / enable. In non-buffered mode, it is SP-bar input, used to distinguish master/slave PIC. In buffered mode, it is output line used to enable buffers
INT	Interrupt line, connected to INTR of microprocessor
INTA-bar	Interrupt ack, received active low from microprocessor
IR0-7	Asynchronous IRQ input lines, generated by peripherals.

8259 internal block diagram





**ICW1 (Initialisation Command Word One)**

A0	D7	D6	D5	D4	D3	D2	D1	D0
0	A7	A6	A5	1	LTIM	ADI	SINGL	IC4

D0: IC4: 0=no ICW4, 1=ICW4 required

D1: SINGL: 1=Single PIC, 0=Cascaded PIC

D2: ADI: Address interval. Used only in 8085, not 8086. 1=ISR's are 4 bytes apart (0200, 0204, etc) 0=ISR's are 8 byte apart (0200, 0208, etc)

D3: LTIM: level triggered interrupt mode: 1=All IR lines level triggered. 0=edge triggered

D4-D7: A5-A7: 8085 only. ISR address lower byte segment. The lower byte is

A7	A6	A5	A4	A3	A2	A1	A0
----	----	----	----	----	----	----	----

of which A7, A6, A5 are provided by D7-D5 of ICW1 (if ADI=1), or A7, A6 are provided if ADI=0. A4-A0 (or A5-A0) are set by 8259 itself.

**ADI=1 (spacing 4 bytes)**

IRQ	A7	A6	A5	A4	A3	A2	A1	A0
IR0	A7	A6	A5	0	0	0	0	0
IR1	A7	A6	A5	0	0	1	0	0
IR2	A7	A6	A5	0	1	0	0	0
IR3	A7	A6	A5	0	1	1	0	0
IR4	A7	A6	A5	1	0	0	0	0
IR5	A7	A6	A5	1	0	1	0	0
IR6	A7	A6	A5	1	1	1	0	0
IR7	A7	A6	A5	1	1	1	0	0

**ADI=0 (spacing 8 bytes)**

IRQ	A7	A6	A5	A4	A3	A2	A1	A0
IR0	A7	A6	0	0	0	0	0	0
IR1	A7	A6	0	0	1	0	0	0
IR2	A7	A6	0	1	0	0	0	0
IR3	A7	A6	0	1	1	0	0	0
IR4	A7	A6	1	0	0	0	0	0
IR5	A7	A6	1	0	1	0	0	0
IR6	A7	A6	1	1	0	0	0	0
IR7	A7	A6	1	1	1	0	0	0

1	R	SL	EOI	0	0	L3	L2	L1
	R	SL	EOI	Action				
EOI	0	0	1	Non specific EOI (L3L2L1=000)				
	0	1	1	Specific EOI command (Interrupt to clear given by L3L2L1)				
Auto rotation of priorities (L3L2L1=000)	1	0	1	Rotate priorities on non-specific EOI				
	1	0	0	Rotate priorities in auto EOI mode set				
	0	0	0	Rotate priorities in auto EOI mode clear				
Specific rotation of priorities (Lowest priority ISR=L3L2L1)	1	1	1	Rotate priority on specific EOI command (resets current ISR bit)				
	1	1	0	Set priority (does not reset current ISR bit)				
	0	1	0	No operation				

**OCW3 (Operational Command Word Three)**

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	D7	ESMM	SMM	0	1	MODE	RIR	RIS

ESMM	SMM	Effect
0	X	No effect
1	0	Reset special mask
1	1	Set special mask

• **ICW2 (Initialisation Command Word Two)**

Higher byte of ISR address (8085), or 8 bit vector address (8086).

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	A15	A14	A13	A12	A11	A10	A9	A8

• **ICW3 (Initialisation Command Word Three)**

A0		D7	D6	D5	D4	D3	D2	D1	D0
1	Master	S7	S6	S5	S4	S3	S2	S1	S0
	Slave	0	0	0	0	0	ID3	ID2	ID1

- Master mode: 1 indicates slave is present on that interrupt, 0 indicates direct interrupt
- Slave mode: ID3-ID2-ID1 is the slave ID number. Slave 4 on IR4 has ICW3=04h (0000 0100)

• **ICW4 (Initialisation Command Word Four)**

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	SFNM	BUF	M/S	AEOI	Mode

- SFNM: 1=Special Fully Nested Mode, 0=FNM
- M/S: 1=Master, 0=Slave
- AEOI: 1=Auto End of Interrupt, 0=Normal
- Mode: 0=8085, 1=8086

• **OCW1 (Operational Command Word One)**

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	M7	M6	M5	M4	M3	M2	M1	M0

IR<sub>n</sub> is masked by setting M<sub>n</sub> to 1; mask cleared by setting M<sub>n</sub> to 0 (n=0..7)

• **OCW2 (Operational Command Word Two)**

A0	D7	D6	D5	D4	D3	D2	D1	D0
----	----	----	----	----	----	----	----	----

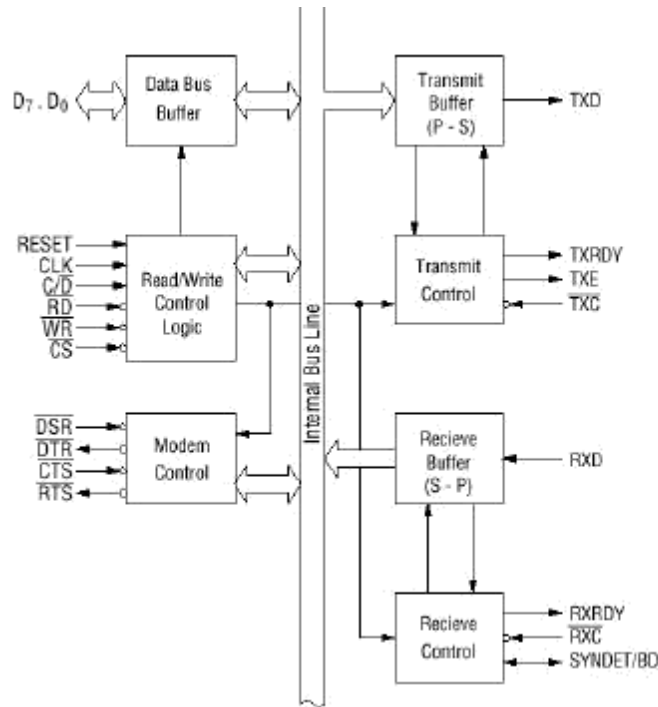
## 4) 8251 UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (USART)

The 8251 is a USART (Universal Synchronous Asynchronous Receiver Transmitter) for serial data communication. As a peripheral device of a microcomputer system, the 8251 receives parallel data from the CPU and transmits serial data after conversion. This device also receives serial data from the outside and transmits parallel data to the CPU after conversion

### Control Words

There are two types of control word.

1. Mode instruction (setting of function)
2. Command (setting of operation)



### 1) Mode Instruction

Mode instruction is used for setting the function of the 8251. Mode instruction will be in "wait for write" at either internal reset or external reset. That is, the writing of a control word after resetting will be recognized as a "mode instruction." Items set by mode instruction are as follows:

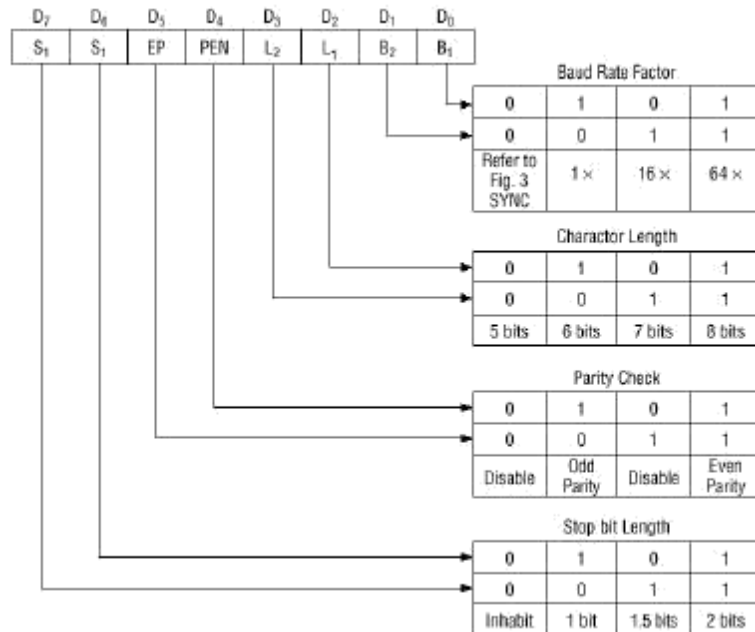
- Synchronous/asynchronous mode
- Stop bit length (asynchronous mode)
- Character length
- Parity bit
- Baud rate factor (asynchronous mode)
- Internal/external synchronization (synchronous mode)
- Number of synchronous characters (Synchronous mode)

The bit configuration of mode instruction is shown in Figure. In the case of synchronous mode, it is necessary to write one-or-two byte sync characters. If sync characters were written, a function will be set because the writing of sync characters constitutes part of mode instruction.

### Command

Command is used for setting the operation of the 8251. It is possible to write a command whenever necessary after writing a mode instruction and sync characters. Items to be set by command are as follows:

- Transmit Enable/Disable
- Receive Enable/Disable

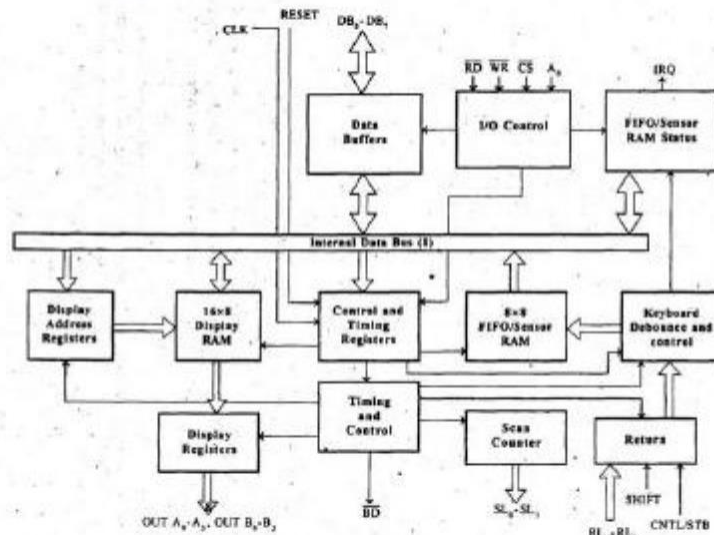
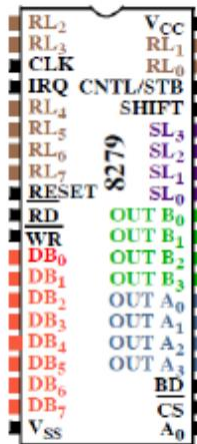


## 5) Programmable Keyboard/Display Interface - 8279

A programmable keyboard and display interfacing chip. Scans and encodes up to a 64-key keyboard. Controls up to a 16-digit numerical display. Keyboard section has a built-in FIFO 8 character buffer. The display is controlled from an internal 16x8 RAM that stores the coded display information.

**A0:** Selects data (0) or control/status (1) for reads and writes between micro and 8279.

- **BD:** Output that blanks the displays.
- **CLK:** Used internally for timing. Max is 3 MHz.
- **CN/ST:** Control/strobe, connected to the control key on the keyboard
- **CS:** Chip select that enables programming, reading the keyboard, etc.
- **DB7-DB0:** Consists of bidirectional pins that connect to data bus on micro.
- **IRQ:** Interrupt request, becomes 1 when a key is pressed, data is available.
- **OUT A3-A0/B3-B0:** Outputs that send data to the most significant/least significant nibble of display.
- **RD(WR):** Connects to micro's IORC or RD signal, reads data/status registers.
- **RESET:** Connects to system RESET.
- **RL7-RL0:** Return lines are inputs used to sense key depression in the keyboard matrix.
- **Shift:** Shift connects to Shift key on keyboard



The display section has eight output lines divided into two groups A0-A3 and B0-B3.

- The output lines can be used either as a single group of eight lines or as two groups of four lines, in conjunction with the scan lines for a multiplexed display.
- The output lines are connected to the anodes through driver transistor in case of common cathode 7-segment LEDs.
- The cathodes are connected to scan lines through driver transistors.
- The display can be blanked by BD (low) line.
- The display section consists of 16 x 8 display RAM. The CPU can read from or write into any location of the display RAM.

Scan section:

- The scan section has a scan counter and four scan lines, SL0 to SL3.
- In decoded scan mode, the output of scan lines will be similar to a 2-to-4 decoder.
- In encoded scan mode, the output of scan lines will be binary count, and so an external decoder should be used to convert the binary count to decoded output.
- The scan lines are common for keyboard and display.
- The scan lines are used to form the rows of a matrix keyboard and also connected to digit drivers of a multiplexed display, to turn ON/OFF. CPU interface section:

- The CPU interface section takes care of data transfer between 8279 and the processor.
- This section has eight bidirectional data lines DB0 to DB7 for data transfer between 8279 and CPU.
- It requires two internal address A =0 for selecting data buffer and A = 1 for selecting control register of 8279.
- The control signals WR (low), RD (low), CS (low) and A0 are used for read/write to 8279.
- It has an interrupt request line IRQ, for interrupt driven data transfer with processor.
- The 8279 requires an internal clock frequency of 100 kHz. This can be obtained by dividing the input clock by an internal prescaler.
- The RESET signal sets the 8279 in 16-character display with two -key lockout keyboard modes.

## 6) Keyboard Interface of 8279

The keyboard matrix can be any size from 2x2 to 8x8. Pins SL2-SL0 sequentially scan each column through a counting operation. The 74LS138 drives 0's on one line at a time. The 8279 scans RL pins synchronously with the scan. RL pins incorporate internal pull-ups, no need for external resistor pull-ups. The 8279 must be programmed first. First three bits given below select one of 8 control registers (opcode).

- 000DDMMM

**Mode set:** Opcode 000.

DD sets displays mode.

MMM sets keyboard mode.

DD field selects either: • 8- or 16-digit display

- Whether new data are entered to the rightmost or leftmost display position.

<i>DD</i>	<i>Function</i>
000	Encoded keyboard with 2-key lockout
001	Decoded keyboard with 2-key lockout
010	Encoded keyboard with N-key rollover
011	Decoded keyboard with N-key rollover
100	Encoded sensor matrix
101	Decoded sensor matrix
110	Strobed keyboard, encoded display scan
111	Strobed keyboard, decoded display scan

**Encoded:** SL outputs are active-high, follow binary bit pattern 0-7 or 0-15.

**Decoded:** SL outputs are active-low (only one low at any time). Pattern output: 1110, 1101, 1011, 0111.

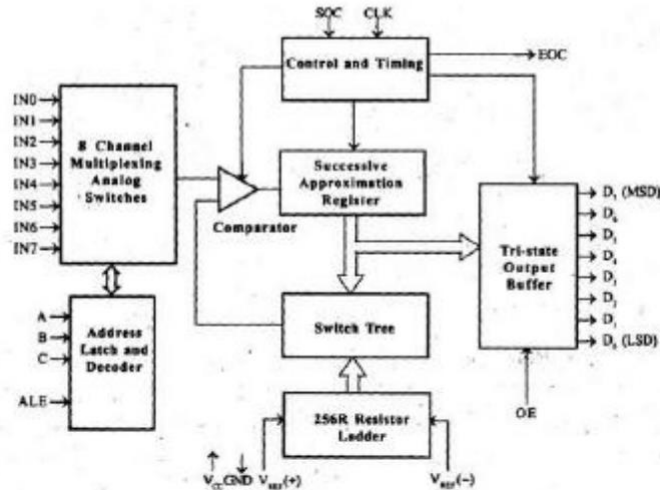
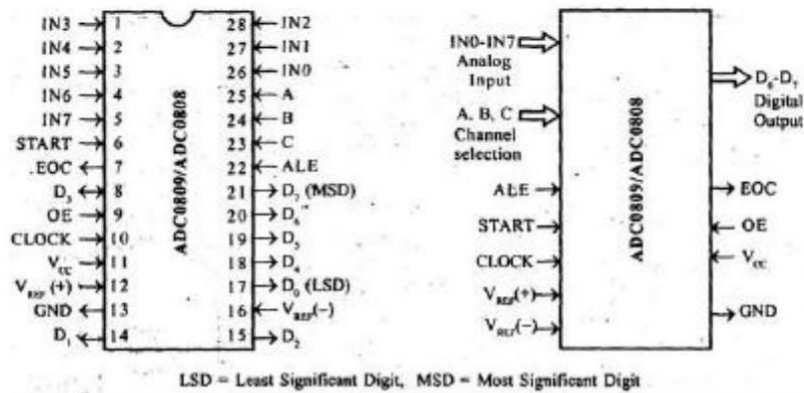
**Strobed:** An active high pulse on the CN/ST input pin strobes data from the RL pins into an internal FIFO for reading by micro later.

**2-key lockout/N-key rollover:** Prevents 2 keys from being recognized if pressed simultaneously/Accepts all keys pressed from 1st to last.

## 7)ADC Interfacing with 8085 Microprocessor

### Features

- The ADC0809 is an 8-bit successive approximation type ADC with inbuilt 8-channel multiplexer.
- The ADC0809 is suitable for interface with 8086 microprocessor.
- The ADC0809 is available as a 28 pin IC in DIP (Dual Inline Package).
- The ADC0809 has a total unadjusted error of  $\pm 1$  LSD (Least Significant Digit).
- The ADC0808 is also same as ADC0809 except the error. The total unadjusted error in ADC0808 is  $\pm 1/2$  LSD.



The successive approximation register (SAR) performs eight iterations to determine the digital code for input value. The SAR is reset on the positive edge of START pulse and start the conversion process on the falling edge of START pulse. A conversion process will be interrupted on receipt of new START pulse. The End-Of-Conversion (EOC) will go low between 0 and 8 clock pulses after the positive edge of START pulse. The ADC can be used in continuous conversion mode by tying the EOC output to START input. In this mode an external START pulse should be applied whenever power is switched ON. The 256R ladder network has been provided instead of conventional R/2R ladder because of its inherent monotonic, which guarantees no missing digital codes. Also the 256R resistor network does not cause load

variations on the reference voltage. The comparator in ADC0809/ADC0808 is a chopper-stabilized comparator. It converts the DC input signal into an AC signal, and amplifies the AC signal using high gain AC amplifier. Then it converts AC signal to DC signal. This technique limits the drift component of the amplifier, because the drift is a DC component and it is not amplified/passed by the AC amplifier. This makes the ADC extremely insensitive to temperature, long term drift and input offset errors. In ADC conversion process the input analog value is quantized and each quantized analog value will have a unique binary equivalent. The quantization step in ADC0809/ADC0808 is given by,

$$Q_{\text{step}} = \frac{V_{\text{REF}}}{2^8} = \frac{V_{\text{REF}(+)} - V_{\text{REF}(-)}}{256_{10}}$$

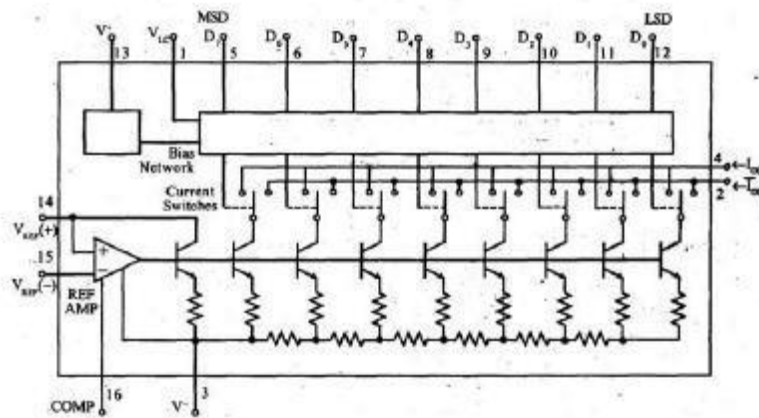
The digital data corresponding to an analog input ( $V_{in}$ ) is given by,

$$\text{Digital data} = \left( \frac{V_{in}}{Q_{\text{step}}} - 1 \right)_{10}$$

## 8) DAC Interfacing with 8085 Microprocessor

To convert the digital signal to analog signal a Digital-to-Analog Converter (DAC) has to be employed.

- The DAC will accept a digital (binary) input and convert to analog voltage or current.
- Every DAC will have "n" input lines and an analog output.
- The DAC require a reference analog voltage ( $V_{\text{ref}}$ ) or current ( $I_{\text{ref}}$ ) source.
- The smallest possible analog value that can be represented by the n-bit binary code is called resolution.
- The resolution of DAC with n-bit binary input is  $1/2^n$  of reference analog value.



The DAC0800 is an 8-bit, high speed, current output DAC with a typical settling time (conversion time) of 100 ns.

It produces complementary current output, which can be converted to voltage by using simple resistor load.

- The DAC0800 require a positive and a negative supply voltage in the range of  $\pm 5V$  to  $\pm 18V$ .
- It can be directly interfaced with TTL, CMOS, PMOS and other logic families.



- For TTL input, the threshold pin should be tied to ground ( $V_{LC} = 0V$ ).
- The reference voltage and the digital input will decide the analog output current, which can be converted to a voltage by simply connecting a resistor to output terminal or by using an op-amp I to V converter.
- The DAC0800 is available as a 16-pin IC in DIP.

**UNIT V**  
**MICRO CONTROLLER PROGRAMMING & APPLICATIONS**

**1) ARITHMETIC INSTRUCTIONS**

Mnemonic	Operation	Addressing modes	Execution time
ADD A, <byte>	$A = A + \text{<byte>}$	Dir, Ind, Reg, Imm	1
ADDC A, <byte>	$A = A + \text{<byte>} + C$	Dir, Ind, Reg, Imm	1
SUBB A, <byte>	$A = A - \text{<byte>} - C$	Dir, Ind, Reg, Imm	1
INC A	$A = A + 1$	Accumulator only	1
INC <byte>	$\text{<byte>} = \text{<byte>} + 1$	DPTR only	1
INC DPTR	$DPTR = DPTR + 1$	Dir, Ind, Reg	2
DEC A	$A = A - 1$	Accumulator only	1
DEC <byte>	$\text{<byte>} = \text{<byte>} - 1$	Dir, Ind, Reg	1
MUL AB	$B:A = B \times A$	ACC and B only	4
DIV AB	$A = \text{Int}[A/B]$ $B = \text{Mod}[A/B]$	ACC and B only	4
DA A	Decimal Adjust	Accumulator only	1

## 2) LOGICAL INSTRUCTIONS

Mnemonic	Operation	Addressing modes	Execution time
ANL A, <byte>	A = A .AND. <byte>	Dir, Ind, Reg, Imm	1
ANL A, <byte>	<byte> = <byte> .AND. A	Dir	1
ANL <byte>, #data	<byte> = <byte> .AND. #data	Dir	2
ORL A, <byte>	A = A .OR. <byte>	Dir, Ind, Reg, Imm	1
ORL A, <byte>	<byte> = <byte> .OR. A	Dir	1
ORL <byte>, #data	<byte> = <byte> .OR. #data	Dir	2
XRL A, <byte>	A = A .XOR. <byte>	Dir, Ind, Reg, Imm	1
XRL A, <byte>	<byte> = <byte> .XOR. A	Dir	1
XRL <byte>, #data	<byte> = <byte> .XOR. #data	Dir	2
CRL A	A = 00H	Accumulator only	1
CPL A	A = .NOT. A	Accumulator only	1
RL A	Rotate ACC Left 1 bit	Accumulator only	1
RLC A	Rotate Left through Carry	Accumulator only	1
RR A	Rotate ACC Right 1 bit	Accumulator only	1
RRC A	Rotate Right through Carry	Accumulator only	1
SWAP A	Swap Nibbles in A	Accumulator only	1

## 3) DATA TRANSFER INSTRUCTION THAT ACCESS THE INTERNAL MEMORY

Mnemonic	Operation	Addressing modes	Execution time
MOV A, <src>	A = <src>	Dir, Ind, Reg, Imm	1
MOV <dest>, A	<dest> = A	Dir, Ind, Reg	1
MOV <dest>, <src>	<dest> = <src>	Dir, Ind, Reg, Imm	2
MOV DPTR, #data 16	DPTR = 16-bit immediate constant	Imm	2
PUSH <src>	INCSP: MOV "@SP", <src>	Dir	2
POP <dest>	MOV <dest>, "@SP": DECSP	Dir	2
XCH A, <byte>	ACC and <byte> exchange data	Dir, Ind, Reg	1
XCHD A, @Ri	ACC and @Ri exchange low nibbles	Ind	1

#### 4) )DATA TRANSFER INSTRUCTION THAT ACCESS THE EXTERNAL MEMORY

Address width	Mnemonic	Operation	Execution time
8 bits	MOVX A, @Ri	Read external RAM @Ri	2
8 bits	MOVX @Ri, A	Write external RAM @Ri	2
16 bits	MOVX A, @DPTR	Read external RAM @DPTR	2
16 bits	MOVX @DPTR, A	Write external RAM @DPTR	2

#### 5)LOOK UP TABLES

Mnemonic	Operation	Execution time
MOVC A, @A+DPTR	Read Program Memory at (A + DPTR)	2
MOVC A, @A+PC	Read Program Memory at (A + PC)	2

#### 6)BOOLEAN INSTRUCTION

Mnemonic	Operation	Execution time
ANL C, bit	C = C .AND. bit	2
ANL C, /bit	C = C .AND. .NOT. bit	2
ORL C, bit	C = C .OR. bit	2
ORL C, /bit	C = C .OR. .NOT. bit	2
MOV C, bit	C = bit	1
MOV bit, C	bit = C	2
CRL C	C = 1	1
CRL bit	bit = 0	1
SETB C	C = 1	1
SETB bit	bit = 1	1
CPL C	C = .NOT. C	1
CPL bit	bit = .NOT. bit	1
JC rel	Jump if C = 1	2
JNC rel	Jump if C = 0	2
JB bit, rel	Jump if bit = 1	2
JNB bit, rel	Jump if bit = 0	2
JBC bit, rel	Jump if bit = 1; CLR bit	2

## 7) JUMP INSTRUCTIONS

Mnemonic	Operation	Execution time
ANL C, bit	C = C .AND. bit	2
ANL C, /bit	C = C .AND. .NOT. bit	2
ORL C, bit	C = C .OR. bit	2
ORL C, /bit	C = C .OR. .NOT. bit	2
MOV C, bit	C = bit	1
MOV bit, C	bit = C	2
CRL C	C = 1	1
CRL bit	bit = 0	1
SETB C	C = 1	1
SETB bit	bit = 1	1
CPL C	C = .NOT. C	1
CPL bit	bit = .NOT. bit	1
JC rel	Jump if C = 1	2
JNC rel	Jump if C = 0	2
JB bit, rel	Jump if bit = 1	2
JNB bit, rel	Jump if bit = 0	2
JBC bit, rel	Jump if bit = 1; CLR bit	2

## 8) Interfacing Keyboard to 8051 Microcontroller

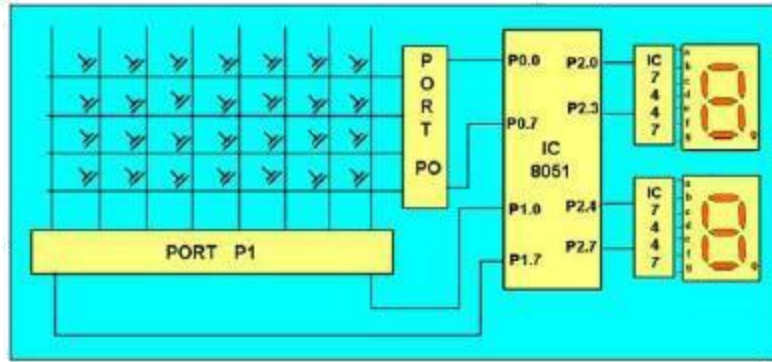
The key board here we are interfacing is a matrix keyboard. This key board is designed with a particular rows and columns. These rows and columns are connected to the microcontroller through its ports of the micro controller 8051. We normally use 8\*8 matrix key board. So only two ports of 8051 can be easily connected to the rows and columns of the key board. When ever a key is pressed, a row and a column gets shorted through that pressed key and all the other keys are left open. When a key is pressed only a bit in the port goes high. Which indicates microcontroller that the key is pressed. By this high on the bit key in the corresponding column is identified. Once we are sure that one of key in the key board is pressed next our aim is to identify that key. To do this we firstly check for particular row and then we check the corresponding column the key board.

To check the row of the pressed key in the keyboard, one of the row is made high by making one of bit in the output port of 8051 high . This is done until the row is found out. Once we get the row next out job is to find out the column of the pressed key. The column is detected by contents in the input ports with the help of a counter. The content of the input port is rotated with carry until the carry bit is set.

The contents of the counter is then compared and displayed in the display. This display is designed using a seven segment display and a BCD to seven segment decoder IC 7447.

The BCD equivalent number of counter is sent through output part of 8051 displays the number of pressed key.





## SEVEN SEGMENT DISPLAY

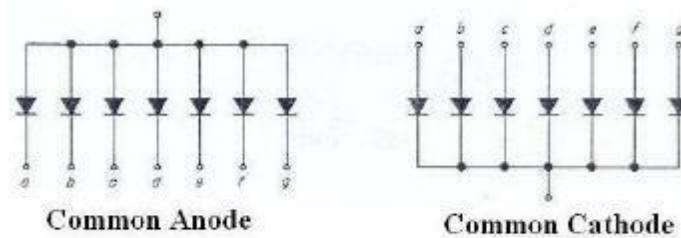
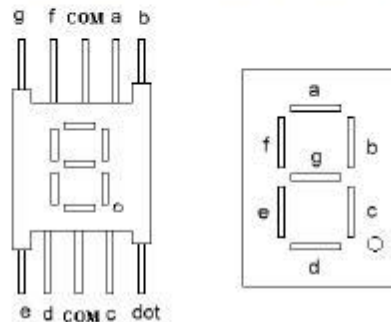


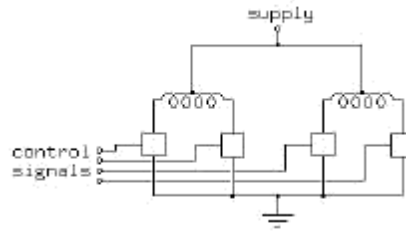
Fig 5.3 Interfacing LEDs to 8051 Microcontroller



## 9) Interfacing Stepper Motor with 8051 Microcontroller

Step motor is the easiest to control. Its handling simplicity is really hard to deny – all there is to do is to bring the sequence of rectangle impulses to one input of step controller and direction information to another input. Direction information is very simple and comes down to "left" for logical one on that pin and "right" for logical zero. Motor control is also very simple every impulse makes the motor operating for one step and if there is no impulse the motor won't start. Pause between impulses can be shorter or longer and it defines revolution rate. This rate cannot be infinite because the motor won't be able to "catch up" with all the impulses (documentation on specific motor should contain such information). The picture below represents the scheme for connecting the step motor to microcontroller and appropriate program code follows.

The key to driving a stepper is realizing how the motor is constructed. A diagram shows the representation of a 4 coil motor, so named because 4 coils are used to cause the revolution of the drive shaft. Each coil must be energized in the correct order for the motor to spin.



### Step angle

It is angle through which motor shaft rotates in one step. step angle is different for different motor . selection of motor according to step angle depends on the application , simply if you require small increments in rotation choose motor having smaller step angle. No of steps require to rotate one complete rotation =  $360 \text{ deg.} / \text{step angle in deg.}$

### INTERFACING TO 8051.

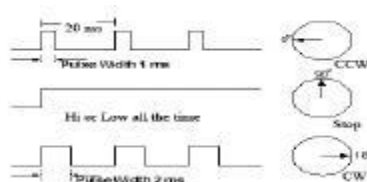
To cause the stepper to rotate, we have to send a pulse to each coil in turn. The 8051 does not have sufficient drive capability on its output to drive each coil, so there are a number of ways to drive a stepper, Stepper motors are usually controlled by transistor or driver IC like ULN2003. Driving current for each coil is then needed about 60mA at +5V supply.

### Servo Motor

Servos are DC motors with built in gearing and feedback control loop circuitry. And no motor drivers required. They are extremely popular with robot, RC plane, and RC boat builders. Most servo motors can rotate about 90 to 180 degrees. Some rotate through a full 360degreesormore.

However, servos are unable to continually rotate, meaning they can't be used for driving wheels, unless they are modified (how to modify), but their precision positioning makes them ideal for robot legs and arms, rack and pinion steering, and sensor scanners to name a few. Since servos are fully self contained, the velocity and angle control loops are very easy to impliment, while prices remain very affordable. To use a servo, simply connect the black wire to ground, the red to a 4.8-6V source, and the yellow/white wire to a signal generator (such as from your microcontroller). Vary the square wave pulse width from 1-2 ms and your servo is now position/velocity controlled.

Pulse width modulation (PWM) is a powerful technique for controlling analog circuits with a processor's digital outputs. PWM is employed in a wide variety of applications, ranging from measurement and communications to power control and conversion. The general concept is to simply send an ordinary logic square wave to your servo at a specific wave length, and your servo goes to a particular angle (or velocity if your servo is modified).The wavelength directly maps to servo angle.



### Controlling the Servo Motor

- **PWM**

Pulse width modulation (PWM) is a powerful technique for controlling analog circuits with a processor's digital outputs. PWM is employed in a wide variety of applications, ranging from measurement and communications to power control and conversion. The general concept is to simply send an ordinary logic square wave to your servo at a specific wave length, and your servo goes to a particular angle (or velocity if your servo is modified). The wavelength directly maps to servo angle.

- **Programmable Counter Array (PCA)**

The PCA is a special modules in Philips P89V51RD2 which includes a special 16-bit Timer that has five 16-bit capture/compare modules associated with it. Each of the modules can be programmed to operate in one of four modes: rising and/or falling edge capture, software timer, high-speed output, or pulse width modulator. Each module has a pin associated with it in port 1. Module 0 is connected to P1.3 (CEX0), module 1 to P1.4 (CEX1), etc. Registers CH and CL contain current value of the free running up counting 16-bit PCA timer. The PCA timer is a common time base for all five modules and can be programmed to run at: 1/6 the oscillator frequency, 1/2 the oscillator frequency, the Timer 0 overflow, or the input on the ECI pin (P1.2). The timer count source is determined from the CPS1 and CPS0 bits in the CMOD SFR. In the CMOD SFR there are three additional bits associated with the PCA. They are CIDL which allows the PCA to stop during idle mode, WDTE which enables or disables the Watchdog function on module 4, and ECF which when set causes an interrupt and the PCA overflow flag CF (in the CCON SFR) to be set when the PCA timer overflows. The Watchdog timer function is implemented in module 4 of PCA. Here, we are interested only PWM mode.

- **8051 Pulse width modulator mode**

All of the PCA modules can be used as PWM outputs. Output frequency depends on the source for the PCA timer. All of the modules will have the same frequency of output because they all share one and only PCA timer. The duty cycle of each module is independently variable using the module's capture register CCAPnL. When the value of the PCA CL SFR is less than the value in the module's CCAPnL SFR the output will be low, when it is equal to or greater than the output will be high. When CL overflows from FF to 00, CCAPnL is reloaded with the value in CCAPnH. this allows updating the PWM without glitches. The PWM and ECOM bits in the module's CCAPMn register must be set to enable the PWM mode. For more details see P89V51RD2 datasheet. This is an example how to control servos with 8051 by using PWM. The schematic is shown below. I use P1.4 (CEX1) to control the left servo and P1.2 (CEX2) to control the right servo. Here, I use GWS servo motor model S03T STD. I need three states of duty cycle:

- 20 ms to Stop the servo
- 1 ms to Rotate Clockwise

**Calculation for duty cycle (for XTAL 18.432 MHz with 6 Clock/Machine cycle)**



- Initial PWM Period = 20mS (18.432MHz /6-Cycle Mode)
- Initial PCA Count From Timer0 Overflow
- 1 Cycle of Timer0 =  $(1/18.432\text{MHz}) \times 6 = 0.326 \text{ uS}$
- Timer0 AutoReload = 240 Cycle = 78.125 uS
- 1 Cycle PCA =  $[(1/18.432\text{MHz}) \times 6] \times 240 = 78.125 \text{ uS}$
- Period 20mS of PCA =  $20\text{ms}/78.125\text{us} = 256 \text{ (CL Reload)}$
- CL (20mS) = 256 Cycle Auto Reload
- Load CCAPxH (1.0mS) =  $256-13 = 243 \text{ (243,244,...,255 = 13 Cycle)}$
- Load CCAPxH (2.0mS) =  $255-26 = 230 \text{ (230,231,...,255 = 26 Cycle)}$
- 2 ms to Rotate Counter-clockwise

## Washing Machine Control

Many washing m/c shell in the market has mechanical controlled sequence for activated the timer and the sequence back and forth for their motor; washing motor or spinning motor. Spinning motor control only has one direction only, and its simple could be changed to the discrete mechanical timer which sell on the market. But washing motor control has 2 direction for this purpose, it means to squeeze the clothes, it must go to forward and then reversed. The sequence is like this :

- First, go to forward direction for about a few seconds
- Than stop, while the chamber is still rotate
- Second, go back to reverse direction for about a few seconds • Than stop, while the chamber is still rotate
- And so on, back and forth, until the the timer elapsed

## SCHEMATIC

Timing sequence like the above description, can be implemented with many way, by using discrete electronic components, timer, using a program or a microcontroller or microprocessor, etc. Because I am learning the PIC microcontroller for right now, I will implement this function using this microcontroller, but for you who familiar with another kind of microcontroller my adapted it to your purpose. By using PIC micro, it can be made more compact. First I plan to make 2 buttons, 1 for set the timer and another for reset the timer or for the emergency stop push button. Then to know the timer works or not, I need a visual display. For this purpose I will use 7-segmen display showing the rest of the timer. To run the motor sequence of course I need a pair of relays (power relays, about 3 Amperes output), one for forward and another for reverse option. I will use the very common family of PIC micro, ie : 16F84A, because this is the most popular type and very simples used and very much used. Also can be obtained easily in the market. But this is the medium type of PIC micro family. It has 1kByte of memory (EEPROM type) and 13 I/O pins. It can be reprogrammable thousands times. Because the I/O just only 13 pins, I used a BCD to 7-segmen chip. So it will left a few I/O pins for expanded in the future. You can omitted this chip for timing sequence purpose and save one IC price, because the I/O just exactly enough.

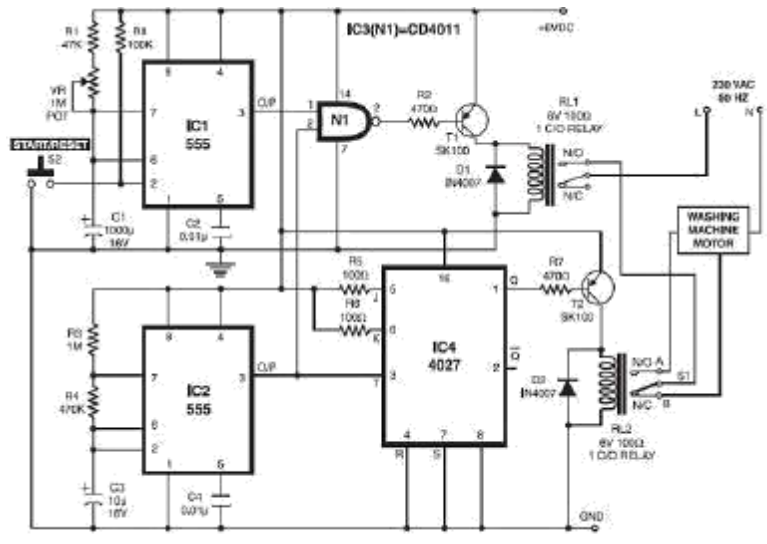
- I/O port A-0 = SET push button

- I/O port A-1 = RST push button
- I/O port A-2 = Reserved
- I/O port A-3 = Reserved
- I/O port A-4 = Reserved
- I/O port B-0 = Forward Relay (Run motor forward)
- I/O port B-1 = Reverse Relay (Run motor reverse)
- I/O port B-2 = Activated unit 7-segmen (multiplexed)
- I/O port B-3 = Activated ten 7-segmen (multiplexed)
- I/O port B-4 = BCD data A (for 7-segmen)
- I/O port B-5 = BCD data B (for 7-segmen)
- I/O port B-6 = BCD data C (for 7-segmen)
- I/O port B-7 = BCD data D (for 7-segmen)
- Also integrated power supply to run it modularly

The I/O can be configured as input pin or output pin bit-ly. It is up to you to choose the I/O pin number goes to what function, but it infect the program firmware of course. Once you choose, then it is just like that, except you also change both, the program and the hardware.

### **Working of Washing Machine**

The direction of rotation can be controlled When switch S1 is in position A, coil L1 of the motor receives the current directly, whereas coil L2 receives the current with a phase shift due to capacitor C. So the rotor rotates in clockwise direction (see Fig. 2(a)). When switch S1 is in position B, the reverse happens and the rotor rotates in anti-clockwise direction. Thus switch S1 can change the rotation direction. The motor cannot be reversed instantly. It needs a brief pause between switching directions, or else it may get damaged. For this purpose, another spin direction control timer (IC2) is employed. It is realised with an IC 555. This timer gives an alternate „on“ and „off“ time duration of 10 seconds and 3 seconds, respectively. So after every 10 seconds of running (either in clockwise or anticlockwise direction), the motor stops for a brief duration of 3 seconds. The values of R3 and R4 are calculated accordingly. The master timer is realised with monostable IC555 (IC1) and its „on“ time is decided by the resistance of 1-megaohm potentiometer VR. A 47-kilo-ohm resistor is added in series so that even when the VR knob is in zero resistance position, the net series resistance is not zero.



UNIT 1  
TWO MARKS

1. What is the function of program counter in 8085 microprocessor? [may 2013]

Program counter stores the address of next instruction to be fetched. Thus it is used as pointer to the instruction.

2. List the control and status signal of 8085 microprocessor and mention its need?[dec 12,08,06]

**ALE (Output):**

✓ Address Latch Enable: It occurs during the first clock cycle of a machine state and enables the address to get latched into the on chip latch of peripherals.

✓ The falling edge of ALE is set to guarantee set up and hold times for the address information. ALE can also be used to store the status information.

**RD(Output3state):**

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

**WR (Output3state):**

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

**IO/M,S<sub>0</sub>,S<sub>1</sub>:**

✓ IO/M indicates whether input output operation or memory operation is being carried out.

✓ S<sub>0</sub> and S<sub>1</sub> indicated the type os machine cycle in progress.

**READY(Input):**

If Ready is high during a read or write cycle, it indicates that the memory or peripheral is ready to send or receive data .

If Ready is low, the CPU will wait for Ready to go high before completing the read or write cycle.

3. Specify the size of data, address, memory word and capacity of 8085 microprocessor[may 2011]

- ✓ Size of data bus =8 bits
- ✓ Size of address bus=16bit
- ✓ Size of memory word=8 bit
- ✓ Memory capacity=64kb

4. What is the need of ALE signal in 8085 microprocessor?[may 2010]

- ✓ This is used to demultiplex AD0 to AD7 lines to A0-A7 and D0-D7.
- ✓ The separation of address line and data line is achieved by connecting a external latch to AD0-AD7 lines and enabling the latch when ALE signal is active.

5. What is tristate logic?[june 2009]

- ✓ Logic output has two states LOW and HIGH corresponding to the logic value 0 and 1.
- ✓ However some output have a third electrical state that is not logical called high impedance or floating state.
- ✓ In this state the output behaves as if it is not even connected to circuit except small leakage current that may flow into or out of the output pin. This state is tristate logic.

6. List the five interrupt pins available in 8085[may 2010]

- The five interrupt pins available in 8085 are
- TRAP
- RST 7.5
- RST 6.5
- RST 5.5
- INTR

7. What is stack and what is the function of stack pointer[Dec 07]

- ✓ The stack is a reserved area of memory in the RAM where temporary information may be stored.
- ✓ A 16 bit stack pointer is used to hold the address of most recent stack entry.

8. Define the function of parity flag and zero flag in 8085[may 12]

- Parity Flag (PF): This flag is used to indicate the parity of result. If lower order 8-bits of the result contains even number of 1's, the Parity Flag is set and for odd number of 1's, the Parity Flag is reset.
- Zero Flag(ZF) :It is set; if the result of arithmetic or logical operation is zero else it is reset.

9. What is the important control signal in 8085 microprocessor[Dec08]

The important signal in 8085 microprocessor are: ALE,IO/M,RD/WR.

10. Why interfacing is needed for I/O devices?

Generally I/O devices are slow devices. Therefore the speed of I/O devices does not match with the speed of microprocessor. And so an interface is provided between system bus and I/O devices.

11. Why address bus is unidirectional?

The address is an identification number used by the microprocessor to identify or access a memory location or I / O device. It is an output signal from the processor. Hence the address bus is unidirectional.

12. What are machine language and assembly language programs?

The software developed using 1's and 0's are called machine language, programs. The software developed using mnemonics are called assembly language programs.

13. What are the basic units of a microprocessor ?

The basic units or blocks of a microprocessor are ALU, an array of registers and control unit

14. Steps involved to fetch a byte in 8085?

- The pc places the 16-bit memory address on the address bus
- The control unit sends the control signal RD to enable the memory chip
- The byte from the memory location is placed on the data bus

- The byte is placed in the instruction decoder of the microprocessor and the task is carried out according to the instruction.

15. Define instruction cycle, machine cycle and T-state?

- ✓ Instruction cycle is defined as the time required completing the execution of an instruction.
- ✓ Machine cycle is defined as the time required completing one operation of accessing memory, I/O or acknowledging an external request.
- ✓ T –cycle is defined as one subdivision of the operation performed in one clock period.

16. How many machine cycles does 8085 have, mention them?

- Opcode fetch
- Memory read
- Memory write
- I/O read
- I/O write
- Interrupt acknowledge

## UNIT 2

### 1.What is an instruction?

An instruction is a binary pattern entered through an input device to command the microprocessor to perform that specific function.

**2.How many operations are there in the instruction set of 8085 microprocessor?** There are 74 operations in the 8085 microprocessor

**3.List out the five categories of the 8085 instructions.Give e.g. of the instructions for each group?[December 2011]**

- ✓ Data transfer group–MOV,MVI,LXI
- ✓ Arithmetic group–ADD,SUB,INR.
- ✓ Logical group-ANA,XRA,CMP.
- ✓ Branch group–JMP,JNZ,CALL.
- ✓ Stack/I/O and machine control group–PUSH,POP,IN,HLT.

**4.Explain the use of branch instruction and give example.[may 2012]**

JMP instruction permanently changes the program counter. A CALL instruction leaves information on the stacks that the original program execution sequence can be resumed.

**5.Explain the purpose of the I/O instructions IN and OUT[may 10]**

- ✓ The IN instruction is used to move data from an I/O port into the accumulator.
- ✓ The OUT instruction is used to move data from the accumulator to an I/O port.
- ✓ The IN and OUT instructions are used only on microprocessor, which use a separate address space for interfacing.

**6.What is the difference between the shift and rotate instructions?**

A rotate instruction is a closed loop instruction.that is the data moved out at one end is put back in at the other end.The shift instruction loses the data that is moved out of the last bit locations.

**7. List the four instructions which control the interrupt structure of the 8085 microprocessor?[may 2013]**

- ✓ DI(disable interrupts)
- ✓ EI(enable interrupts)
- ✓ RIM(read interrupt masks)
- ✓ SIM(set interrupt masks)

**8.Mention the categories of machine control group of instruction [may 2013]**

The instructions of 8085 can be categorized into the following five

1. EI
2. DI
3. NOP
4. HLT
5. SIM
6. RIM

**9.Explain LDA, STAANDDAA instructions**

LDA copies the data byte into the accumulator from the memory location specified by the 16-bit address.STA copies the data byte from the accumulator in the memory location specified by 16-bitaddress. DAA changes the content of the accumulator from binary to 4-bit BCD digits

**10.Explain the different instruction format set [june 09]**

The instruction set is grouped into the following formats One by the instruction MOV C,A Two byte instruction MVI A,39H Three byte instruction JMP2345H

**11.What is the use of addressing modes, mention the different types?[may 2012]**

The various formats of specifying the operands are called as addressing modes, it is used to access the operands or data. The different types are as follows

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

**12.Define stack and stack related instructions?[may 2013 and December 2012]**

The stack is a group of memory locations in the R/W memory that is used for the temporary storage of binary information during the execution of the program. The stack related instructions are PUSH and POP

**13.Why do we use XRAA instruction?**

The XRAA instruction is used to clear the contents of the accumulator and store the value 00H

#### 14. Compare CALL and PUSH instructions

CALL	PUSH
When CALL is executed the microprocessor automatically stores the 16-bit address of the instruction next to CALL on the stack	The program uses the instruction PUSH to save the contents of the register pair on the stack
When CALL is executed the stack pointer is decremented by two	When PUSH is executed the stack pointer register is decremented by two

#### 15. What are subroutines?

- ✓ Procedures are a group of instructions stored as a separate program in memory and it is called from the main program in memory and it is called from the main program when ever required.
- ✓ The type of procedure depends on where the procedures are stored in memory. If it is in the same code segment as that of the main program then it is an near procedure otherwise it is a far procedure.

### UNIT 3

#### 1. What is meant by microcontroller

A device which contains the microprocessor with integrated peripherals like memory, serial ports, parallel ports, timer/counter, interrupt controller, data acquisition interfaces like ADC, DAC is called microcontroller.

#### 2. Explain DJNZ instruction of Intel8051 microcontroller ? [June 2013]

DJNZ Rn Decrement the content of the register Rn and jump if not zero.

DJNZ direct, rel Decrement the content of direct 8-bit address and jump if not zero

#### 3. State the function of RS1 and RS0 bits in the flag register of Intel8051 microcontroller?

- RS1, RS0-Register bank select bits
- RS1, RS0-Bank
- Bank0
- Bank1
- Bank2
- Bank3

#### 4. Give the alternate functions for the port pins of port3? [Dec 11 May 11]

- RDWR T1 T0
- INT1 INT0 TXDRXD
- RD-Read data control output
- WR-Write data control output
- T1- Timer/counter1 external input or test pin T0- Timer/counter0 external input or test pin INT1-Interrupt1 input pin
- INT0- interrupt0 input pin
- TXD-Transmit data pin for serial port in UART mode
- RXD-Receive data pin for serial port in UART mode



**5. Specify the single instruction, which clears the most significant bit of Register of 8051, without affecting the remaining bits.**

Single instruction, which clears the most significant bit of Register of 8051, without affecting the remaining bits, is CLR B.7.

**6. What are the applications of microcontroller [may 2012]**

- Calculators
- Traffic light control system
- Game machine
- Military applications.

**7. Give the memory size of 8051 microcontroller [may 2010]**

The 8051 can access up to 64 kilobyte of program memory and 64 kilobyte of data memory.

**8. Name the special function registers available in 8051 [Dec 10]**

- Accumulator
- B Register
- Program Status Word.
- Stack Pointer.
- Data Pointer
- Port 0
- Port 1
- Port 2
- Port 3
- Interrupt priority control register. Interrupt enable control register.

**9. Explain the register IE format of 8051. [dec 2011]**

- EA ET2 ES
- ET1 EX1 ET0 EX0
- EA - Enable all control bit.
- ET2 - Timer 2 interrupt enable bit.
- ES - Enable serial port control bit.
- ET1 - Enable Timer 1 control bit.
- EX1 - Enable external interrupt 1 control bit.
- ET0 - Enable Timer 0 control bit.
- EX0 - Enable external interrupt 0 control bit.

**10. Compare Microprocessor and Microcontroller.**

Microprocessor      Microcontroller

1. Microprocessor contains ALU, general Microcontroller contains the circuitry of Purpose register counter, clock timing microprocessor and in addition it has Circuit and interrupt circuit.

Built-in ROM, RAM, I/O devices and counter

2. It has many instructions to move data. It has many instructions to move between memory and CPU. Data between memory and CPU.

**11.Name the five interrupt sources of 8051?[may 08 dec08]**

- The interrupt are: Vector address
- External interrupt0:IE0:0003H
- Timers interrupt0: TF0:000BH
- External interrupt1:IE1:0013
- Timersinterrupt1:TF1:001BH
- Serial interrupt
- Receiveinterrupt:RI:0023H Transmitinterrupt:TI:0023H

**12.Write a program to subtract the contents of RI of Bank0 from the contents of R0of Bank2.**

```
MOVPSW,#10
MOVA,R0
MOVPSW,#00
SUBBA,R1
```

**13.List the features of 8051 microcontroller?[may 04]**

The features are

- Single supply +5 volt operation using HMOS technology.
- 4096 bytes program memory on chip (not on 8031)
- 128 data register banks
- Four register mode,16-bit timer/counter.
- Extensive Boolean processing capabilities.
- 64 KB external RAM size
- 32 bi-directional individually addressable I/O lines.
- 8 bit CPU optimized for control applications.

**14.List the addressing modes of 8051?[dec09 may 13]**

- Direct addressing
- Register addressing
- Register in direct addressing Implicit addressing Immediate addressing
- Index addressing
- Bit addressing

**15. What are the modes of operation used in 8253?**

Each of the three counters of 8253 can be operated in one of the following six modes of operation.

1. Mode 0 (Interrupt on terminal count)
2. Mode 1 (Programmable monoshot)
3. Mode 2 (Rate generator)
4. Mode 3 (Square wave generator)

UNIT 4

**1.What is the use of 8051 chip?**

- ✓ Intel's 8251A is a universal synchronous asynchronous receiver and transmitter compatible with Intel's Processors. This may be programmed to operate in any of the serial communication modes built into it.

- ✓ This chip converts the parallel data in to a serial stream of bits suitable for serial transmission. It is also able to receive a serial stream of bits and converts it in to parallel data bytes to be read by a microprocessor.

## 2. What is the various programmed data transfer method?

- ✓ Synchronous data transfer
- ✓ Asynchronous data transfer
- ✓ Interrupt driven data transfer

## 3. What is synchronous data transfer?

It is a data method which is used when the I/O device and the microprocessor match in speed. The transfer a data to or from the device ,the user program issues a suitable instruction addressing the device. The data transfer is completed at the end of the execution of this instruction.

## 4. What are the features used model in 8255?[MAY 09]

- ✓ Two groups A and group B are available for stored Data transfer.
- ✓ Each group contains one 8-bit data I/O port and one 4-bit control/data port.
- ✓ The 8-bit data port can be either used as input or output port. The inputs and outputs both are latched.

## 5. What are the modes of operation used in 8253?[MAY 08 MAY 09]

Each of the three counters of 8253 can be operated in one of the following six modes of operation.

- ✓ Mode0 (Interrupt on terminal count)
- ✓ Mode1 (Programmable mono shot)
- ✓ Mode2 (Rate generator)
- ✓ Mode3 (Square wave generator)
- ✓ Mode4 (Software triggered strobe)
- ✓ Mode5 (Hardware triggered strobe)

## 6. Give the different types of command words used in 8259A

The command words of 8259A are classified in two groups

- ✓ Initialization command words(ICWs)
- ✓ Operation command words(OCWs)

## 7. Define scan counter?

- ✓ The scan counter has two modes to scan the key matrix and refresh the display. In the encoded mode ,the counter provides binary count that is to be externally decoded to provide the scan lines for keyboard and display
- ✓ .In the decoded scan mode, the counter internally decodes the least significant 2bit and provides a decoded 1 out of 4scan on SL3-SL3. The keyboard and display both are in the same mode at a time.

## 8. What are the modes used in keyboard modes?

- ✓ Scanned Keyboard mode with 2Key Lockout

- ✓ Scanned keyboard with N-Key Rollover.
- ✓ Scanned Keyboard Special Error Mode.
- ✓ Scanned Matrix Mode.

### **9.What are the modes used in display modes?**

#### 1.Left Entry Mode

In the left entry mode,the data is entered from the left side of the display unit.

#### 2.Right Entry Mode

In the right entry mode,the first entry to be displayed is entered on the right most display.

### **10.What is the use of modem control unit in 8251?**

The modem control unit handles the modem handshake signals to coordinate the communication between them on demand the USART.

### **11.List the operation modes of 8255?**

- ✓ Mode0-Simple Input/Output.
- ✓ Mode1-Strobe Input/Output (handshakemode)
- ✓ Mode2-Strobe bi-directional mode.

### **12.What is a control word?**

It is a word stored in a register (control register) used to control the operation of a program digital device.

### **13.What is the purpose of control word written to control register in 8255?**

The control words written to control register specify an I/O function for each I/O port. The bit D7of the control word determines either the I/O functions of the BSR function.

### **14.What is the size of ports in 8255?**

- ✓ Port-A:8-bits
- ✓ Port-B:8-bits
- ✓ Port-CU :4-bits
- ✓ Port-CL:4-bits

### **15.What is an USART?**

USART stands for universal Synchronous/Asynchronous Receiver/Transmitter.It is a programmable communication interface that can communicate by using either synchronous or asynchronous serial data.

## UNIT 5

### **1.Write a program using 8051 assembly language to change the data 55h stored in the lower byte of the data pointer register to AAH using rotate instruction?**

```
MOV DPL,#55H
MOV A,DPLRL A
LABEL :SJMP Label
```

**2.Explain the content of the accumulator after the execution of the following program segments?**

```
MOV A,#3CH
MOVR4,#66H ANLA,R4
A3C R466
A24
```

**3.Write a program to load accumulator a, DPH and DPL with 30H?**

```
MOVA,#30
MOVDPH
AMOVDPL,A
```

**4.Write a program to perform multiplication of 2 Nos using 8051?**

```
MOVA,#data1
MOVB,#data2
MULAB
MOVDPTR,#5000
MOV@DPTR,A(lower value)
INCDPTR
MOVA,B
MOVX@DPTR,A
```

**5.Write a program to mask the 0th&7<sup>th</sup> bit using 8051?**

```
MOVA,#data
ANLA,#81
MOVDPTR,#4500
MOVX@DPTR
,A LOOPS JMP LOOP
```

**6.Write about CALL statement in 8051?**

- ✓ There are two subroutine CALL instructions. they are
  - \*LCALL(Long CALL)
  - \*ACALL(Absolute CALL)
- ✓ Each increments the pc to the 1stbyte of the instruction & pushes them into this tack.

**7. Write about the jump statement?**

There are three forms of jump.they are  
LJMP(Long-jump)-address16  
AJMP(Absolute jump)-address11  
Sjmp(shortjump)-relative address

**8.Write a program to load accumulator DPH&DPL using 8051?**

```
MOVA,#30
MOVDPH,
AMOVDPL,A
```

**9. Write a program to find 2's complement using 8051?**

```
MOVA,R0  
CPLA INCA
```

**10. Write a program to add two 8-bit numbers using 8051?**

```
MOVA,#30H  
ADDA,#50H
```

**11. Write a program to swap two numbers using 8051?**

```
MOVA,#data  
SWAPA
```

**12. Write a program to subtract two 8-bit numbers & exchange the digits using 8051?**

```
MOVA,#9F  
MOVR0,#40  
SUBBA,R0  
SWAPA
```

**13. Write a program to subtract the contents of R1 of bank 0 from the contents of R0 of bank 2 using 8051?**

```
MOVPSW,#10  
MOVA,R0  
MOVPSW,#00  
SUBBA,R1
```

**14. Explain the operating mode 0 of 8051 serial ports?**

In this mode serial enters & exits through RXD, TXD output the shift clock 8 bits are transmitted or received 8 data bits (LSB first). the baud rate is fixed at 1/12 the oscillator frequency.

**15. Explain the operating mode 2 of 8051 serial ports?**

In this mode 11 bits are transmitted (through TXD) or received (through RXD). A start bit (0), 8 data bits (LSB first), a programmable 9th data bit & a stop bit (1). ON transmit the 9th data bit (TB\* in SCON) can be assigned the value of 0 or 1. or for reg: the parity bit (p, in the PSW) could be moved into TB8.