srivenkateshwaraa College of Engineering & Technology
(Approved by AICTE, New Delhi & Affiliated to Pondicherry University, Puducherry)
13-A, Pondy - Villupuram Main Road, Ariyur, Puducherry - 605 102.
ASPIRE TO EXCEL

ISO 21001

Department of Electrical and Electronics Engineering

Subject Name: **Microprocessors and Microcontrollers** Subject Code: **EE T63**

Prepared by:

Mrs.S.Punitha, Assistant Professor/EEE
Mr.V.Malarselvam, Assistant Professor/EEE

Verified by: **Approved by:**

UNIT – I

MICROPROCESSOR ARCHITECUTRE:

8085 Microprocessor architecture

–Registers, Arithmetic and logic section,

Timing and Control section and Interface section-

Machine cycles and bus timings–

Wait states–

Introduction to architecture of Z80 and MC6800 microprocessors.

MICROPROCESSOR

EVOLUTION OF MICROPROCESSOR

1ST GENERATION µP: 1969

- 4 Bit microprocessor
- Memory capacity : 4096 bytes
- It has 45 instructions
- The frequency is very low

CHARACTERISTICS:

- Low cost
- Slow in speed
- Low operational current
- Not compatible in TTL levels

OTHER 4 BIT PROCESSOR:

- Intel 4040
- Fairchild PPS25
- National IMP4
- Rockweln PPS4
- Microsystem INTLMCI

APPLICATION

- Calculator
- Game machine
- Home appliances
- Accounting system
- Intelligent instrumentation

2nd GENERATION µP: 1973

- Intel co-operation introduced the 8-bit µP 8008 and 8080

IMPROVEMENT OVER 1st GENERATION µP

- Memory capacity in 64kb
- TTL compatible
- Speed is high
- Clock frequency is in the range 1MHz – 6MHz
- NMOS technology is employed
- Offers faster speed and higher density than PMOS
- 8-bit processor is the other name for 2nd generation µP

OTHER 8-BIT PROCESSORS:

- Intel 8080
- Intel 8085
- fairchild F8
- Motorola M6800
- Motorola M6809
- National CMP
- Zilog 80/Z80

CHARACTERISTICS:

- long chip size

- 40 pin DIP
- More number of ON chip diode timing signals
- Ability to address larger memory location
- Ability to address more I/o codes
- Faster operation
- Most powerful instruction set
- Great number of levels of sub routine nesting
- Better interrupt handling capabilities

APPLICATION:

- Complex industrial controles
- Communication preprocessor control
- Process control system
- Instrumentation
- Intelligent terminals
- Military application

3rd GENERATION µP: 1978

- Released by Intel co-operation in 1978
- 16 bit µP ie. 8086
- 16 bit µP are called lllrd generation µP
- Memory range is 1MB to 16MB
- Works in multiprocessor environment
- Nses HMOS(high speed MOS) technology
- Offers better speed and high packing density than NMOS

SOME OTHER 3rd GENERATION **µP:**

- Provided with 40/48/68 pins

- High speed and very strong processing capability
- Easier to program
- Allow for dynamically re-locatable program
- Size of the internal register are 8.16/32 bits
- Processor has multiply or divide arithmetic hardware
- More powerful interrupt handling capability
- Flexible i/p ; o/p port addressing
- Different modes of operation

4^{th} GENERATION µP : 1980

- Released by Intel in 1980
- 32 bit µP and has great memory
- 32 bit µP are 4^{th} generation µP
- **INTEL 80386 :**
- first 32 bit µP in 1985
- 4GB of memory
- 25/33/40 NHz clock frequency

INTEL 80486 :

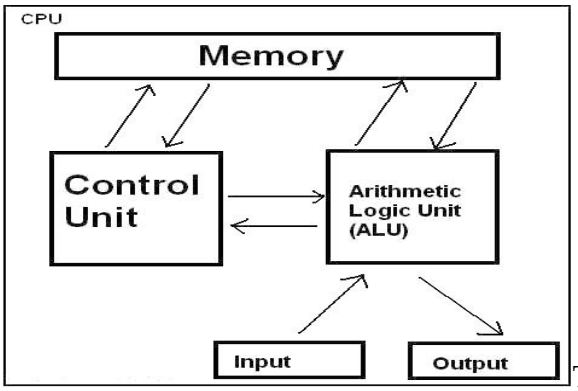
- Improved version of 80386
- Has CPU , an arithmetic co-processor
- Me,pry management unit and 8kb cache memory

MOTOROLA 68030 :

- 32 bit µP in 1985
- Most powerful

Microprocessing Unit (MPU)

The Microprocessing unit is similar to the term Central processing Unit (CPU) used in traditional computers.



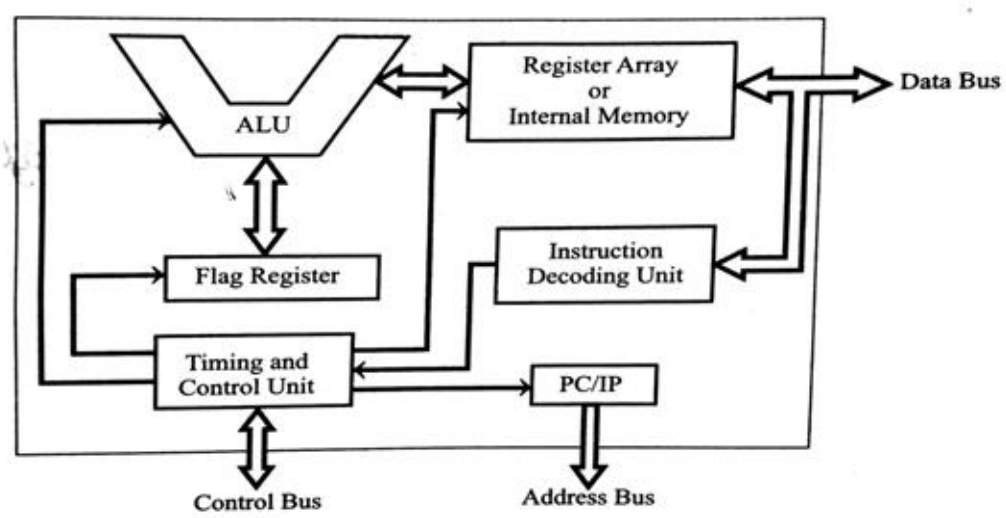
The Microprocessing unit as a device or a group of devices that can communicate with peripherals, provide timing signals, direct data flow and perform computing tasks as specified by the instructions in memory. The unit will have the necessary lines for the address bus, the data bus, and the control signals.

Microcomputer –a computer with a microprocessor as its CPU. Includes memory, I/O etc.

Microprocessor –silicon chip which includes ALU, register circuits & control circuits.

Microcontroller –silicon chip which includes microprocessor, memory & I/O in a single package.

General functional block diagram of Microprocessor.



Organization of Microprocessor.

The microprocessor is sometimes referred to as the 'brain' of the personal computer, and is responsible for the processing of the instructions which make up computer software. It houses the central processing unit, commonly referred to as the CPU.

CPU Structure

The simplified model consists of five parts, which are:

- **Arithmetic HYPERLINK "http://www.eastaughs.fsnet.co.uk/cpu/structure-alu.htm"& HYPERLINK "http://www.eastaughs.fsnet.co.uk/cpu/structure-alu.htm" Logic Unit (ALU)**
  The part of the central processing unit that deals with operations such as addition, subtraction, and multiplication of integers and Boolean operations. It receives control signals from the control unit telling it to carry out these operations. For more, click the title above.
- **Control Unit (CU HYPERLINK "http://www.eastaughs.fsnet.co.uk/cpu/structure-cu.htm")**
  This controls the movement of instructions in and out of the processor, and also controls the operation of the ALU. It consists of a decoder, control logic circuits, and a clock to ensure everything happens at the correct time. It is also responsible for performing the instruction execution cycle.

- **Register Array**
  This is a small amount of internal memory that is used for the quick storage and retrieval of data and instructions. All processors include some common registers used for specific functions, namely the program counter, instruction register, accumulator, memory address register and stack pointer.
- **System Bus**
  This is comprised of the control bus, data bus and address bus. It is used for connections between the processor, memory and peripherals, and transfers of data between the various parts.
- **Memory**
  The memory is not an actual part of the CPU itself, and is instead housed elsewhere on the motherboard. However, it is here that the program being executed is stored, and as such is a crucial part of the overall structure involved in program execution.

8085-CPU ORGANIZATION

FEATURES OF 8085

- 8085A is an 8-bit general – purpose microprocessor.
- It is a 40 pin dual - in – line package single chip integrated circuit.
- Only one +5v power supply is needed for its operation.
- It can operate with a 3 MHZ single – phase clock.
- The 8085A – 2 versions can operate at the maximum frequency of 5MHZ.
- The width of the data bus is 8-bit. The width of the address bus is 16 –bit . Therefore maximum of 64 kilobytes of memory locations(i.e 2^{16}=65,536=64KB) can be addressed directly by the 8085.
- The multiplexing of address/data bus allows for extra control signals.
- 8085 has one non- maskable (TRAP) and three maskable – vectored interrupts (RST 7.5,6.5& 5.5).
- It provides Serial Input Data (SID) and Serial Output Data(SOD) lines for simple serial interface.
- 8085 has an inbuilt clock oscillator circuit and requires externally only a crystal. The frequency of the crystal is internally divided by 2.

PIN DIAGRAM AND PIN DESCRIPTION OF 8085

- 8085 is a 40 pin IC, DIP.

- The microprocessor is a clock-driven semiconductor device consisting of electronic logic circuits manufactured by using either a large-scale integration (LSI) or very-large-scale integration (VLSI) technique.
- The microprocessor is capable of performing various computing functions and making decisions to change the sequence of program execution.
- In large computers, a CPU implemented on one or more circuit boards performs these computing functions.The microprocessor is in many ways similar to the CPU, but includes the logic circuitry, including the control unit, on one chip.
- The microprocessor can be divided into three segments for the sake clarity, arithmetic/logic unit (ALU), register array, and control unit.
  The signals from the pins can be grouped as follows:
    - Power supply and clock signals
    - Address bus
    - Data bus
    - Control and status signals
    - Interrupts and externally initiated signals
    - Serial I/O ports

  8085 microprocessor has two limitations
    - The low –order address bus of the 8085 microprocessor is multiplexed with the data bus. The buses need to be demultiplexed.

Appropriate control signals need to be generated to interface memory and I/O with the 8085

POWER SUPPLY AND CLOCK FREQUENCY SIGNALS:

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

ADDRESS BUS

The 8085 has 16 signal lines that are used as the address bus. These lines are split into two segments:

A_{15} –A_8 and AD_7- AD_0.

The eight signal lines A_{15} –$A_{8,}$ are unidirectional and used for the most significant bits, called higher order address busof a 16-bit address.

The eight signal lines AD_7- $AD_{0,}$ are bidirectional, they serve as a dual purpose.

MULTIPLEXED ADDRESS / DATA BUS

The eight signal lines AD_7- $AD_{0,}$ are bidirectional, they serve as a dual purpose. They are used as the low-order address bus as well as the data bus.

In executing an instruction, during the earlier part of thr cycle, these lines are used as the low –order address bus. During the later part of the cycle, these lines are used as the data bus. The low-order address bus can be separated from these signals by using a latch.

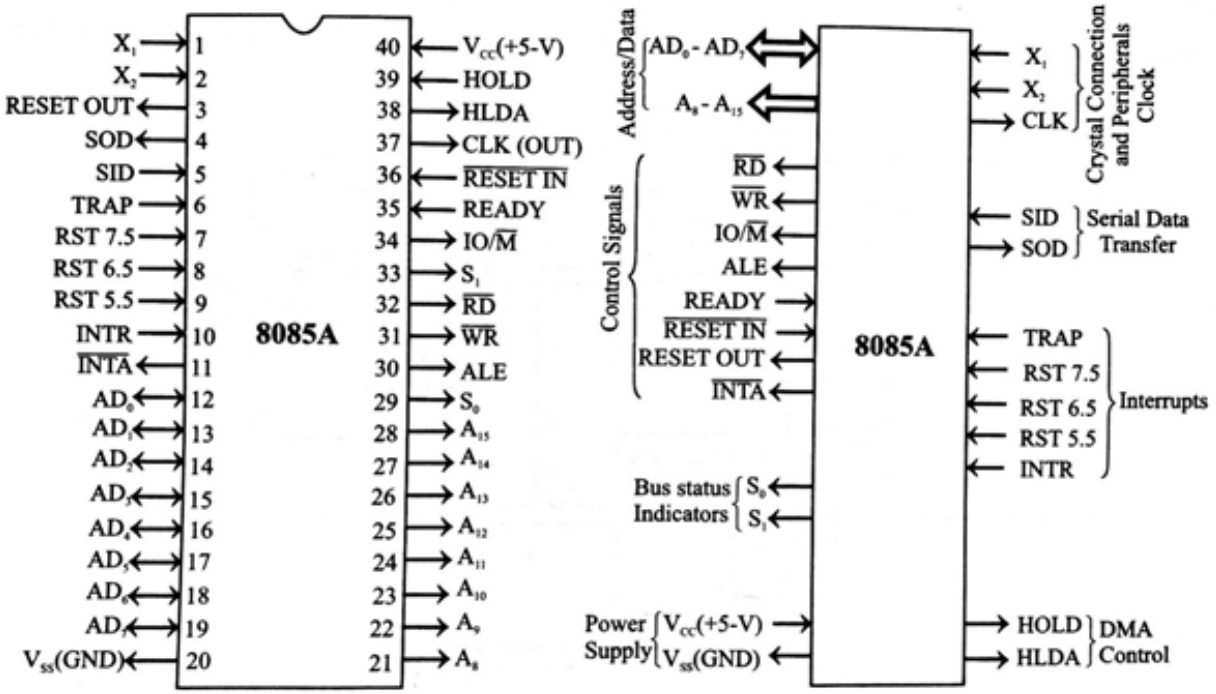


Fig (a) - Pin Diagram of 8085 Fig(b) - logical schematic of Pin diagram.

CONTROL AND STATUS SIGNALS

This group of signals includes two control signals (, three status signals (IO/, to identify the nature of the operation, and one special signal (ALE) to indicate the beginning of the operation.

ALE (output) - Address Latch Enable.

- This signal helps to capture the lower order address presented on the multiplexed address / data bus.
- RD (output 3-state, active low) - Read memory or IO device.
- This indicates that the selected memory location or I/O device is to be read and that the data bus is ready for accepting data from the memory or I/O device.
- WR (output 3-state, active low) - Write memory or IO device.
- This indicates that the data on the data bus is to be written into the selected memory location or I/O device.
- IO/M (output) - Select memory or an IO device.

- This status signal indicates that the read / write operation relates to whether the memory or I/O device.
- It goes high to indicate an I/O operation.
- It goes low for memory operations.

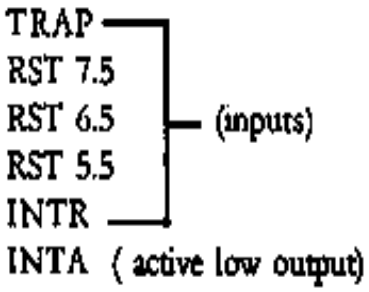
STATUS SIGNALS

- It is used to know the type of current operation of the microprocessor.

Status Signal			Machine Cycle
$\overline{S}_2$	$\overline{S}_1$	$\overline{S}_0$	
0	0	0	Interrupt acknowledge
0	0	1	Read IO port
0	1	0	Write IO port
0	1	1	Halt
1	0	0	Code access
1	0	1	Read memory
1	1	0	Write memory
1	1	1	Passive/Inactive

INTERRUPTS AND EXTERNALLY INITIATED OPERATIONS

- They are the signals initiated by an external device to request the microprocessor to do a particular task or work.
- There are five hardware interrupts called,



- On receipt of an interrupt, the microprocessor acknowledges the interrupt by the active low INTA (Interrupt Acknowledge) signal.

Reset In (input, active low)

- This signal is used to reset the microprocessor.
- The program counter inside the microprocessor is set to zero.
- The buses are tri-stated.

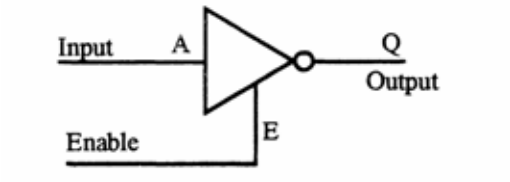
Reset Out (Output)

- It indicates CPU is being reset.

- Used to reset all the connected devices when the microprocessor is reset.

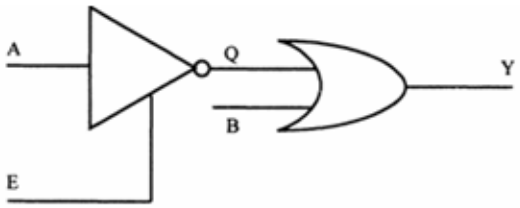
DIRECT MEMORY ACCESS (DMA)

Tri state devices:



- 3 output states are high & low states and additionally a high impedance state.
- When enable E is high the gate is enabled and the output Q can be 1 or 0 (if A is 0, Q is 1, otherwise Q is 0). However, when E is low the gate is disabled and the output Q enters into a high impedance state.

E	A	Q	State
1(high)	0	1	High
1	1	0	Low
0(low)	0	0	High impedance
0	1	0	High impedance



- For both high and low states, the output Q draws a current from the input of the OR gate.
- When E is low, Q enters a high impedance state; high impedance means it is electrically isolated from the OR gate's input, though it is physically connected. Therefore, it does not draw any current from the OR gate's input.
- When 2 or more devices are connected to a common bus, to prevent the devices from interfering with each other, the tri-state gates are used to disconnect all devices except the one that is communicating at a given instant.

- The CPU controls the data transfer operation between memory and I/O device. Direct Memory Access operation is used for large volume data transfer between memory and an I/O device directly.
- The CPU is disabled by tri-stating its buses and the transfer is effected directly by external control circuits.
- HOLD signal is generated by the DMA controller circuit. On receipt of this signal, the microprocessor acknowledges the request by sending out HLDA signal and leaves out the control of the buses. After the HLDA signal the DMA controller starts the direct transfer of data.

READY (input)

- Memory and I/O devices will have slower response compared to microprocessors.
- Before completing the present job such a slow peripheral may not be able to handle further data or control signal from CPU.
- The processor sets the READY signal after completing the present job to access the data.
- The microprocessor enters into WAIT state while the READY pin is disabled.

SINGLE BIT SERIAL I/O PORTS

- SID (input) - Serial input data line
- SOD (output) - Serial output data line
- These signals are used for serial communication.

ARCHITECTURE OF INTEL -8085 (Nov 2015) , (Nov 2014) ,(April 2015)

The functional block diagram or architecture of 8085 Microprocessor is very important as it gives the complete details about a Microprocessor. Fig. shows the Block diagram of a Microprocessor.

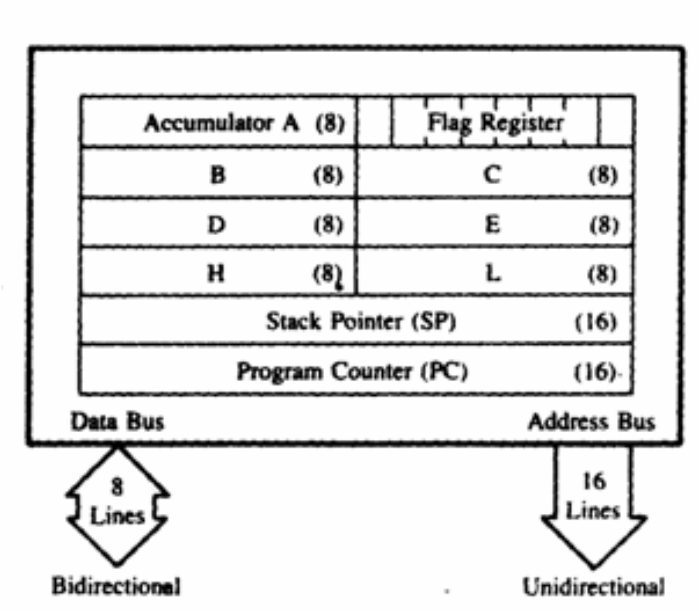
ACCUMULATOR:

- Accumulator is an 8-bit register.
- The accumulator is also called as A-register.
- It holds one of the data to be processed by ALU.
- It stores the result of the operation.
- The accumulator is connected to the 8 – bit internal data bus .
- The two-state output of the accumulator drives the ALU.

TEMPORARY REGISTER:

- The temporary register receives one of the data to be processed by ALU from external memory or general purpose registers.
- The other input for the ALU comes from the temporary register. This 8-bit register stores the operands of arithmetic – logic operations.
- For instance, during an ADD C the contents of the C register are copied in the temporary register during one T state and added to accumulator contents during another T state.

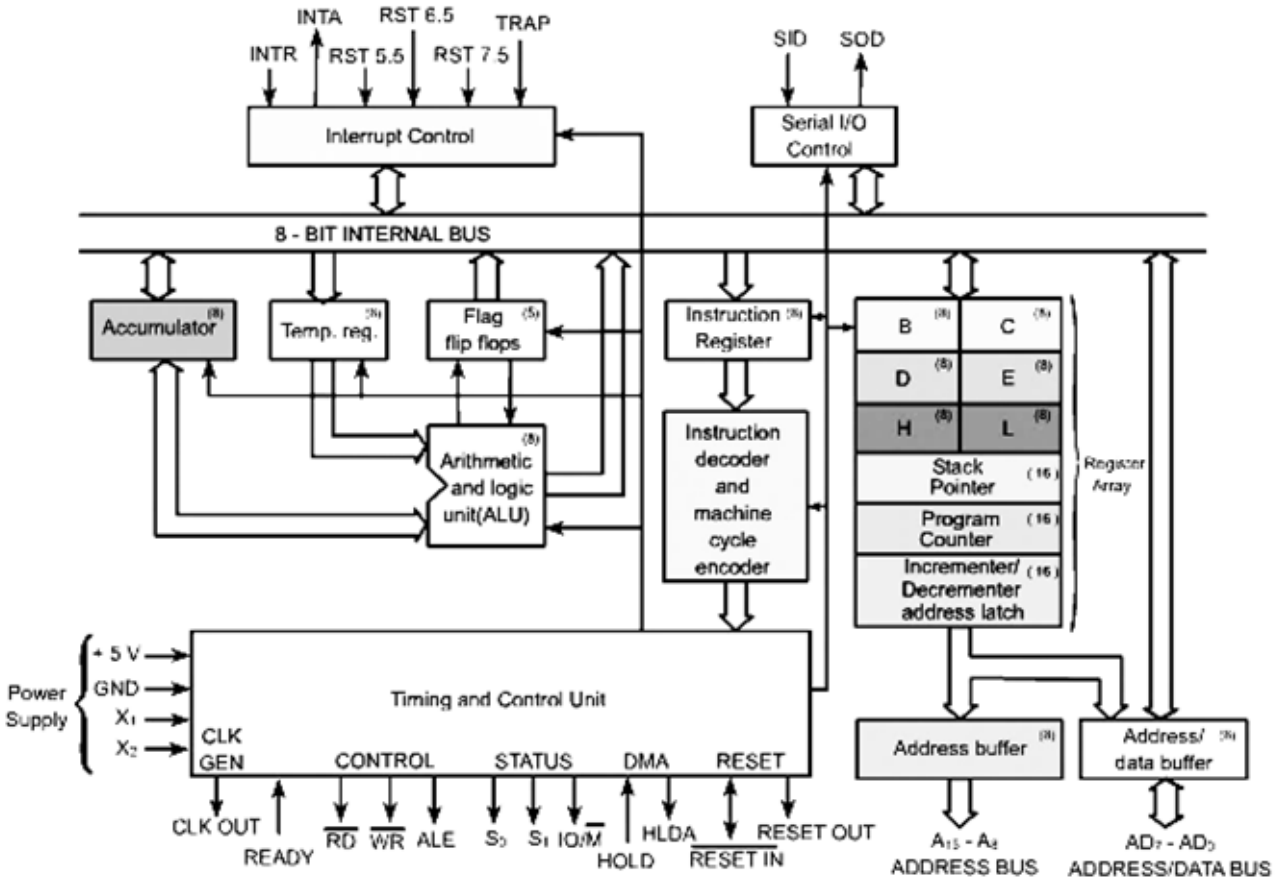
GENERAL PURPOSE REGISTERS:



- In INTEL 8085 microprocessor, there are six 8 –bit general – purpose registers.
  - They are B,C, D,E,H And L.
  - They may be used individually or combined as register pairs to perform some 16 – bit operations.
  - The permitted combinations of register pairs are B –C, D-E and H-L. The H-L registers pair, which is normally used to form a 16-bit memory pointer.
  - The register array (B,C,D,E,H and L) is like a small on-chip RAM with addressable memory locations.
  - Control signals select the register for a read or write operation. This means that the CPU can wither load or read a register contents to this data bus.

STACK POINTER (SP):

- Stack Pointer is a 16-bit register used as a memory pointer.
- It maintains the address of the last byte entered into the stack.
- The stack pointer is decremented each time when data is loaded into the stack and is incremented when data is retrieved from the stack.



Architecture of 8085 microprocessor

PROGRAM COUNTER (PC):

- This 16-bit register deals with sequencing the execution of instruction.
- The register is also a memory pointer.
- Memory locations have 16-bit address, and hence it 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 address of the next instruction to be executed.
- At the end of the execution of an instruction, the program counter is incremented by 1, pointing to the next memory location where the next instruction is available.

INCREMENTER/DECREMENTER:

- It can add 1 or subtract 1 from the contents of the Stack Pointer or Program Counter.

Arithmetic and Logic Unit (ALU): (Nov 2012) (Apr 2011)

- Arithmetic and Logic Unit is used to perform the arithmetic operations like addition, subtraction, multiplication, division, increment and decrement and logical operations like AND, OR and EX-OR.
- It receives the data from accumulator and registers.
- The ALU result is then stored back in the accumulator.
- According to the result it set or reset the flags**.**

FLAGS: [Explain Various flags of 8085 in detail] (April 2015)

Flag register is a group of five individual flip-flops. The content of the flag register will change (0 or 1) after the execution of arithmetic and logic operations. The microprocessor uses the flags for testing the data conditions. They are Zero (Z), Carry (CY), Sign (S), Parity (P), and Auxiliary Carry (AC) flags. The most commonly used flags are Sign, Zero, and Carry.

The bit position for the flags in flag register is,

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

1.Sign Flag (S):

After execution of any arithmetic and logical operation, if bit D_7 of the result is 1, then the **sign flag bit** is set. Otherwise it is reset.

The sign bit indicates the sign of the number (Positive or Negative) and becomes useful in the signed binary number system.

If D_7 is 1, the number will be viewed as negative number. If D_7 is 0, the number will be viewed as positive number.

2.Zero Flag (Z):

If the result of arithmetic and logical operation is zero, then **zero flag bit** is set otherwise it is reset.

Example :
```
  10110011
+ 01001101
----------
1 00000000
----------
```

3.Auxiliary Carry Flag (AC):

The auxiliary carry flag bit (AC) is set, when a carry is generated at digit D_3 position, and passed on to digit D_4. The flag is used only internally for Binary Coded Decimal (BCD) operations.

4.Carry Flag (CY):

The carry flag bit (CY) is set if a carry or borrow occurs during the arithmetic operation. The carry flag indicates that the operation resulted in overflow.

Example:

```
  1011 0101
+ 0110 1100
----------
Carry 1 0010 0001
```

Example:

```
  1011 0101
```

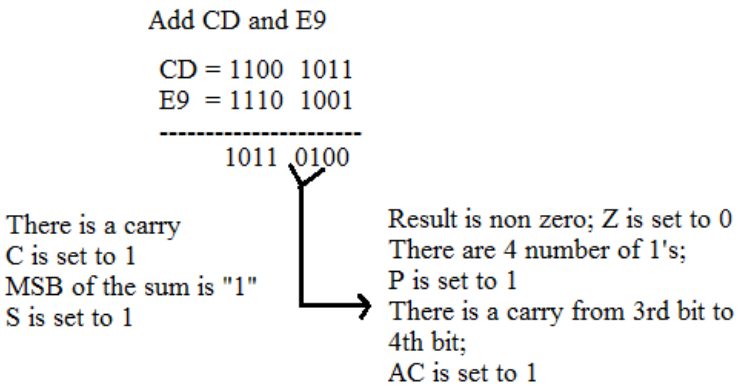
- 1100 1100

Borrow 1 1110 1001

5.Parity Flag (P):

If the result of arithmetic and logical operation contains even number of 1's then parity flag bit will be set and if it is odd number of 1's it will be reset.

Example for flag register:



INSTRUCTION REGISTER AND DECODER

- Instruction register and decoder is an 8-bit register.
- When an instruction is fetched from memory, it is loaded in the instruction register.
- The instruction decoder decodes the contents of the instruction register.
- It also determines the operation to be followed in executing the entire instruction and directs the timing and control unit accordingly.

- During the fetch cycle, the op-code of an instruction is stored in the instruction register. This op-code then drives the instruction decoder and machine- cycle encoder.

TIMING AND CONTROL

- The timing and control section of microprocessor includes an oscillator and controller-sequencer.
- The oscillator generates the two – phase clock signals (CLK and $\overline{CLK}$) that synchronize all registers.
- The controller-sequencer also produces the control signals needed for internal and external control.
- The controller – sequencer is micro programmed; it has a ROM that stores all the micro routines needed for executing the instruction.
- After each instruction is fetched and stored in the instruction register, the opcode is decoded to get the starting address of the desired micro routine.
- As each microinstruction is read out of the control ROM, control signals are sent to the internal and external data buses.
- The effect is to move data between registers, to perform arithmetic and logic operations, to input or output data, etc.
- The control ROM is sometimes called the control store.

INTERRUPT CONTROL

- Sometimes it is necessary to interrupt the execution of the main program to answer a request from an I/O device.
- For instance, an I/O device may send an interrupt signal to the interrupt control unit to indicate that data is ready for input.
- The computer temporarily stops the execution of main program, inputs the data, and then returns to the main program.
- The interrupt concept is analogous to reading a book (main program), hearing the phone (interrupt), answering the phone (servicing the interrupt), then returning back to reading (main program).

SERIAL I/O CONTROL

- Sometimes, I/O devices work with serial data rather than parallel. In this case, the serial data stream from an input device must convert to 8-bit parallel data before the computer can use it.
- Likewise, the 8-bit data out of a computer must be converted to serial form before a serial output device can use it.
- The Serial Input Data enters 8085 through pin 5 (SID – Serial Input Data) and leaves through pin 4 (SOD – Serial Input Data) and leaves through pin 4 (SOD – Serial Output Data).
- Two new instructions known as SIM and RIM allows us to perform the serial-parallel conversion needed for serial I/O devices.

ADDRESS, DATA, AND CONTROL BUSES

- Near the top of the Figure is an 8-bit internal data bus. This carries instructions and data between the CPU registers. The external buses are the ones we have to connect to other chips like memory I/O and so forth.
- Near the bottom left of the Figure is the external control bus ($\overline{RD}, \overline{WR}$, ALE). On the bottom right are the external address- data buses.
- The upper 8 address bits are on a separate bus always used for address bits; this upper section of the address bus is designated A_{15} - A_8.
- The lower 8-bit are multiplexed. This means that the eight lower bus lines are used for address bits during some T states and for data bits during other T states. This is why the bus is labelled address-data bus, and designed as AD_7 – AD_0.

ADDRESS BUFFER AND ADDRESS DATA-BUFFER

- At the bottom right in Figure are two buffer registers called the address buffer and the address – data buffer.
- The contents of the stack pointer or program counter can be loaded into the address buffer and address – data buffer. The output of these buffers then drives the external address bus and address –data bus. Memory and I/O chips are connected to these buses. In this way, the CPU can send the address of desired data to the memory or I/O chips.

- The 8-bit internal data bus is also connected to the address- data buffer. The bi-directional arrow indicates a three-state connection that allows the address-data buffer to send or receive data from the 8-bit internal data bus.

Timing diagrams and Machine cycle of 8085 Microprocessor.(April 2016)

- The 8085 instructions consist of one to five machine cycles.
- Actually the execution of an instruction is the execution of the machine cycles of that instruction in the predefined order.
- The timing diagram of an instruction are obtained by drawing the timing diagrams of the machine cycles of that instruction, one by one in the order of execution.

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.

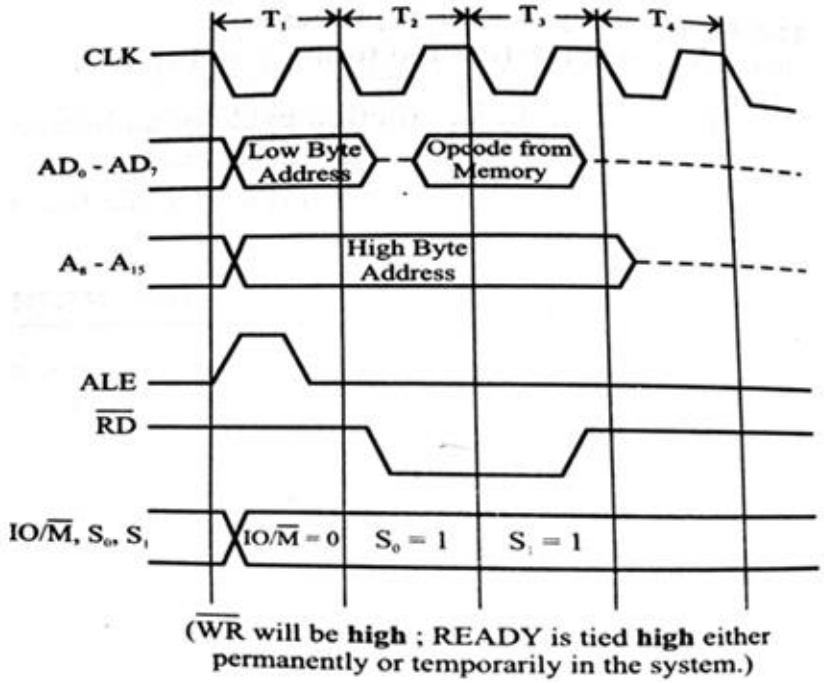
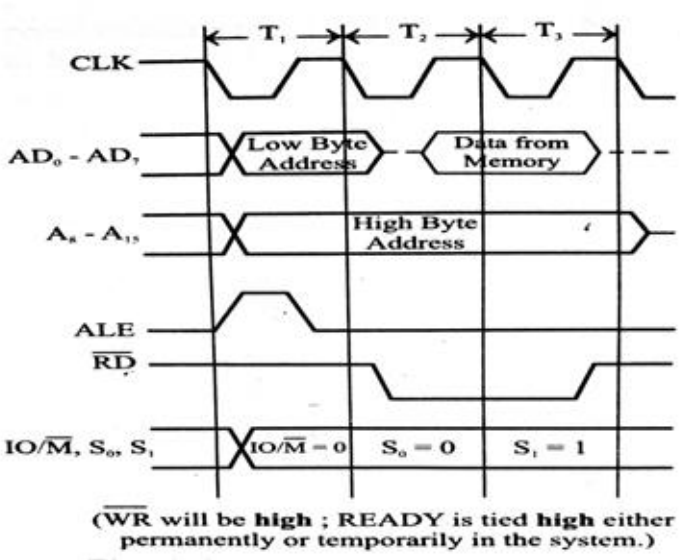


Fig - Timing Diagram for Op-code Fetch Machine Cycle

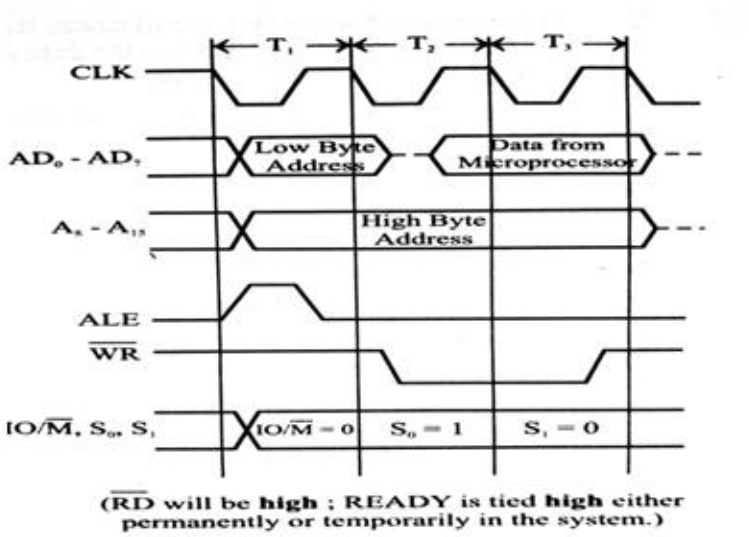
MEMORY READ MACHINE CYCLE OF 8085: (Nov/Dec 2014)

- 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 WRITE MACHINE CYCLE OF 8085:

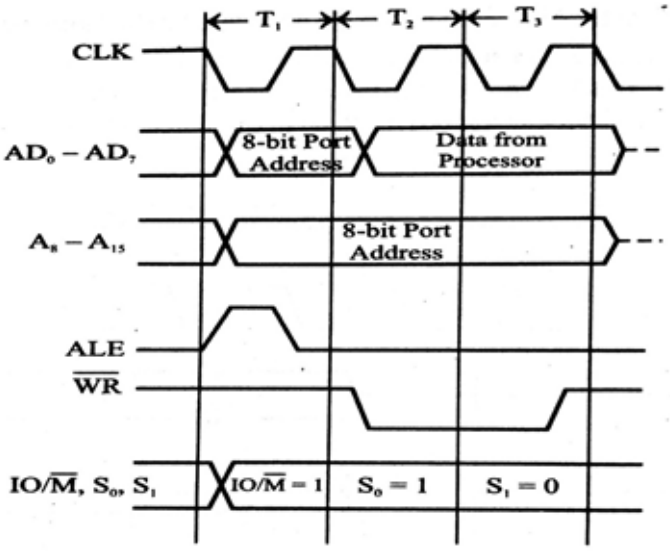
- The memory write machine cycle is executed by the processor to write a data byte to memory.
- The processor takes 3T states to execute this cycle.



(RD will be **high** ; READY is tied **high** either permanently or temporarily in the system.)

I/O WRITE CYCLE OF 8085:

- The I/O write machine cycle is executed by the processor to write a data byte in the I/O port or to a peripheral, which is I/O, mapped in the system.
- The processor takes, 3T states to execute this machine cycle.



(RD will be **high** ; READY is tied **high** either permanently or temporarily in the system.)

Fig - Timing Diagram for I/O Write Machine Cycle

I/O READ CYCLE OF 8085:

- The I/O read machine cycle is executed by the processor to read a data byte in the I/O port or from a peripheral, which is I/O, mapped in the system.
- The processor takes, 3T states to execute this machine cycle.

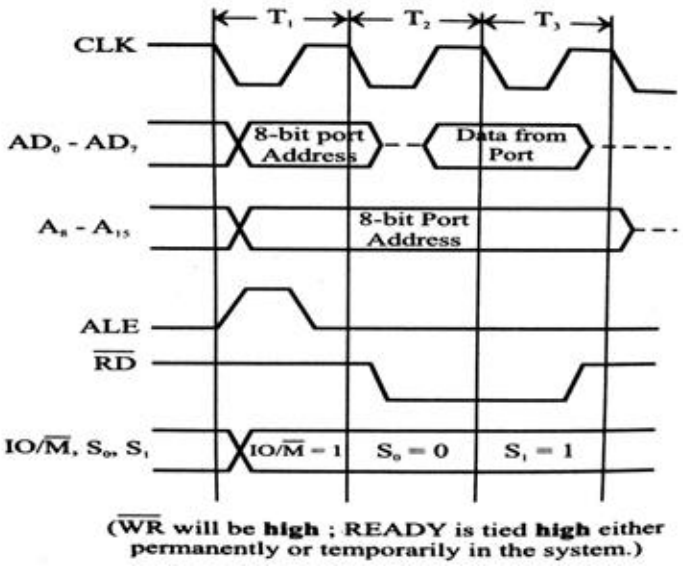


Fig - Timing Diagram for I/O Read Machine Cycle

Timing Diagram for STA 526AH in detail. (April/May 2014)

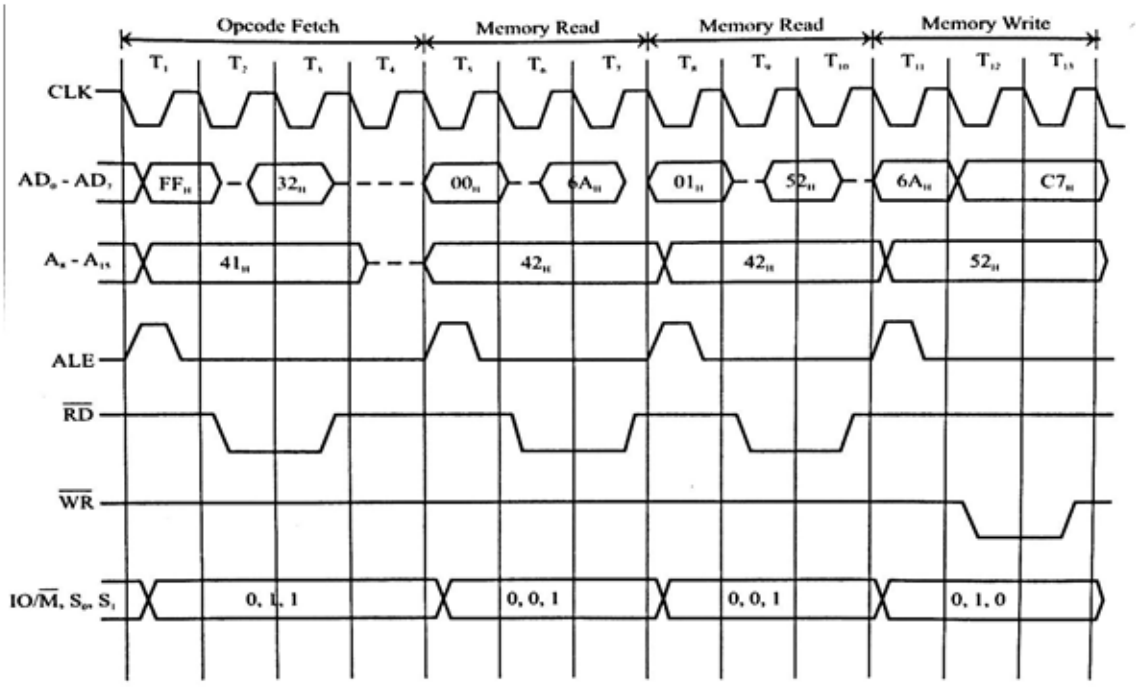
Timing Diagram of STA Instruction

The "STA addr16" instruction is used to store the content of the accumulator to a memory location. This instruction employs direct addressing. Let the content of the accumulator be $C7_H$ and it is desired to store the content of the accumulator to a memory location $526A_H$.

The STA addr16 instruction is a three byte instruction. The first byte is the opcode of the instruction 32_H. The second byte is low byte address $6A_H$ and the third byte is high byte address 52_H. Let the three bytes of the instructions be stored in memory locations $41FF_H$, 4200_H and 4201_H.

In order to execute this instruction, the 8085 microprocessor will first execute opcode fetch machine cycle to get the opcode, followed by two memory read cycles to read the address of data (i.e., to read second and third byte of instruction.) Then, the processor executes the memory write cycle to store the content of the accumulator in the memory. The status of various signals during execution of this instruction are shown in Fig. 3.14. (Readers are advised to refer to Section 3.2 for explanation of each machine cycle.)

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



Timing Diagram for IN C0H for 8085 Microprocessor.

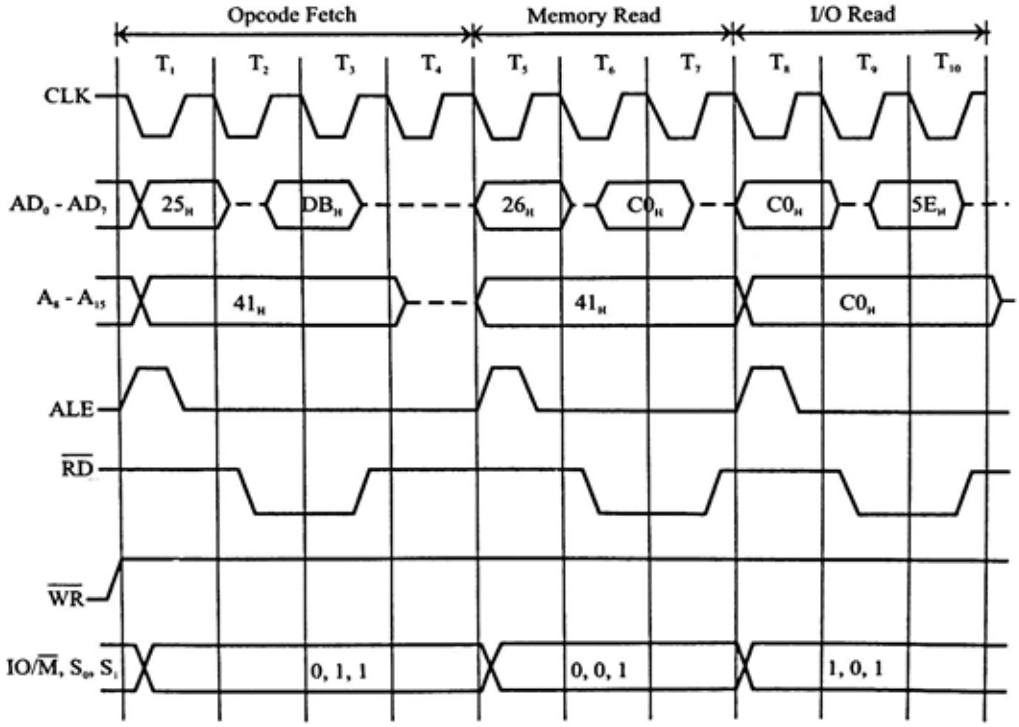
Timing Diagram of IN Instruction

The "IN addr8" instruction is used to read the content of an IO-mapped device/port and store in the accumulator. For addressing IO-mapped devices the 8085 microprocessor employs 8-bit address. Let the 8-bit address of the IO port be $C0_H$, and the content of IO port be $5E_H$.

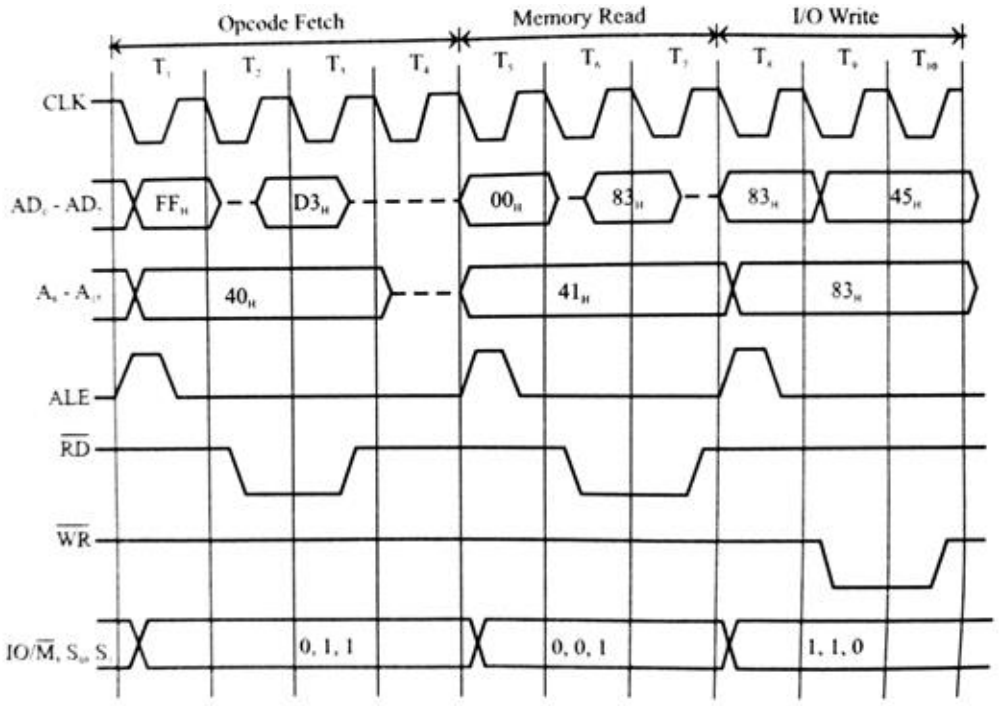
The IN addr8 instruction, is a two byte instruction. The first byte is the opcode of the instruction DB_H and the second byte is the IO port address $C0_H$. Let the two bytes of the instruction be stored in memory locations 4125_H and 4126_H.

In order to execute this instruction, the 8085 microprocessor will first execute the opcode fetch machine cycle to get the opcode, followed by the memory read cycle to read the IO port address (i.e., to read the second byte of the instruction.) Then, the processor executes IO read cycle to read the content of the IO port. The status of various signals during execution of this instruction are shown in Fig. 3.16.

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



Timing Diagram for OUT Instruction for 8085 Microprocessor. (November 2015)

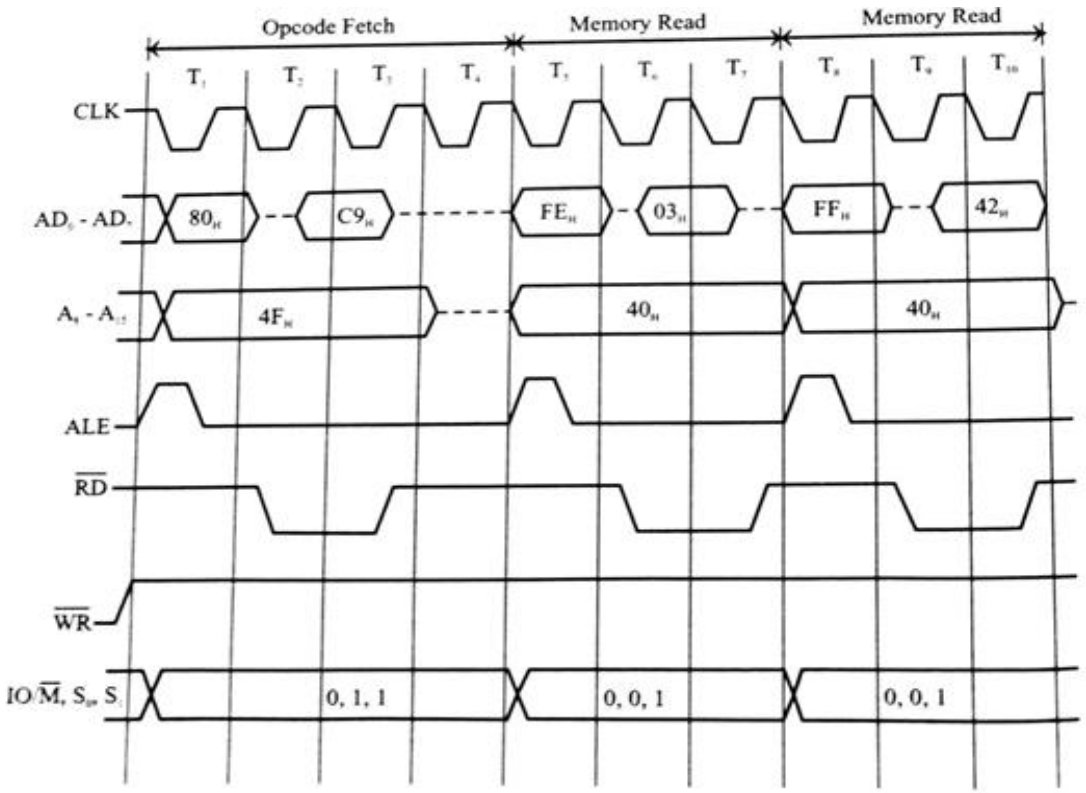


The "OUT addr8" instruction is used to output the content of the accumulator to the IO-mapped device/port. For addressing IO-mapped devices, the 8085 microprocessor employs 8-bit address. Let the 8-bit address of the IO port be 83_H and the content of the accumulator be 45_H.

The OUT addr8 instruction is a two byte instruction. The first byte is the opcode of the instruction $D3_H$, and the second byte is the IO port address 83_H. Let the two bytes of instruction be stored in memory locations $40FF_H$ and 4100_H.

In order to execute this instruction, the 8085 microprocessor will first execute the opcode fetch machine cycle to get the opcode, followed by the memory read cycle to read the IO port address. (i.e., to read the second byte of the instruction.) Then the processor executes the IO write cycle to write the content of the accumulator to the IO port. The status of various signals during execution of this instruction are shown in Fig. 3.17.

Timing diagram for RET instruction



:

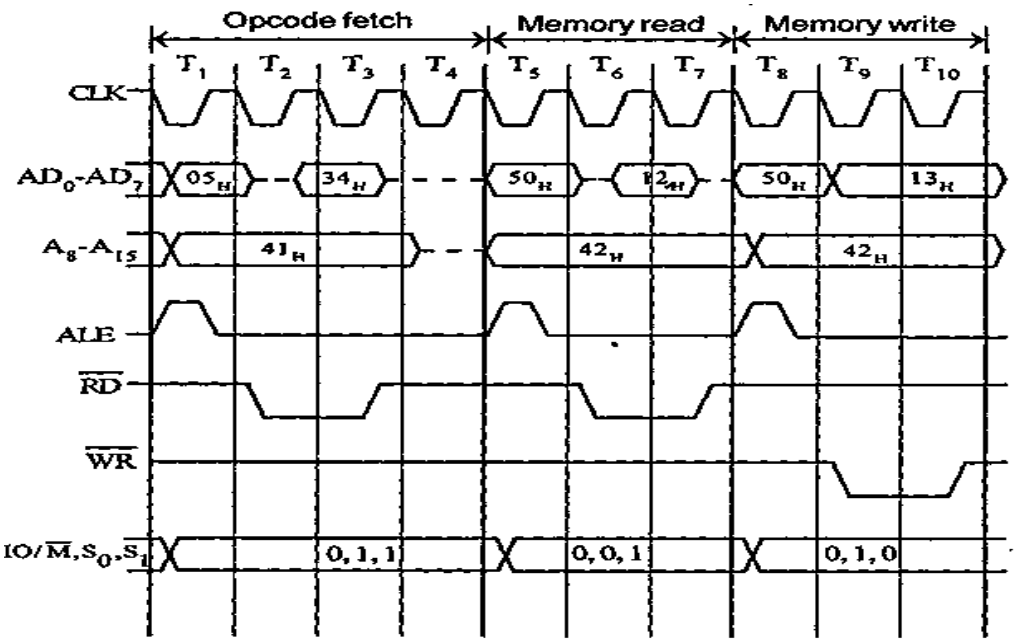
In a program while calling a subroutine/procedure using CALL instruction, the address of the next instruction of the program is stored in the top of the stack. The RET instruction is usually placed at the end of the subroutine/procedure. On execution of the RET instruction, the top of the stack is loaded in the **P**rogram **C**ounter (PC). For stack operation, the content of the **S**tack **P**ointer (SP) is used as the address. Let the content of SP be $40FE_H$ and the content of the stack memory locations $40FE_H$ and $40FF_H$ be 03_H and 42_H respectively.

The RET instruction is a one byte instruction and it is the opcode of instruction $C9_H$. Let this instruction be stored in memory location $4F80_H$. In order to execute this instruction, the 8085 processor will first execute the opcode fetch machine cycle to get the opcode $C9_H$. Then, it executes two memory read cycles to read the top of the stack memory.

During the memory read cycles, the content of the SP is used as the memory address. In the first read cycle, the content of the SP ($40FE_H$) is output on the address lines and the memory content (03_H) in this address is read and stored as low byte of the PC. In the second read cycle, the content of the SP is incremented by one ($40FE_H + 1 = 40FF_H$) and the output on the address lines, and the memory content (42_H) in this address is read and stored as high byte of PC. The status of various signals during execution of this instruction are shown in Fig. 3.20.

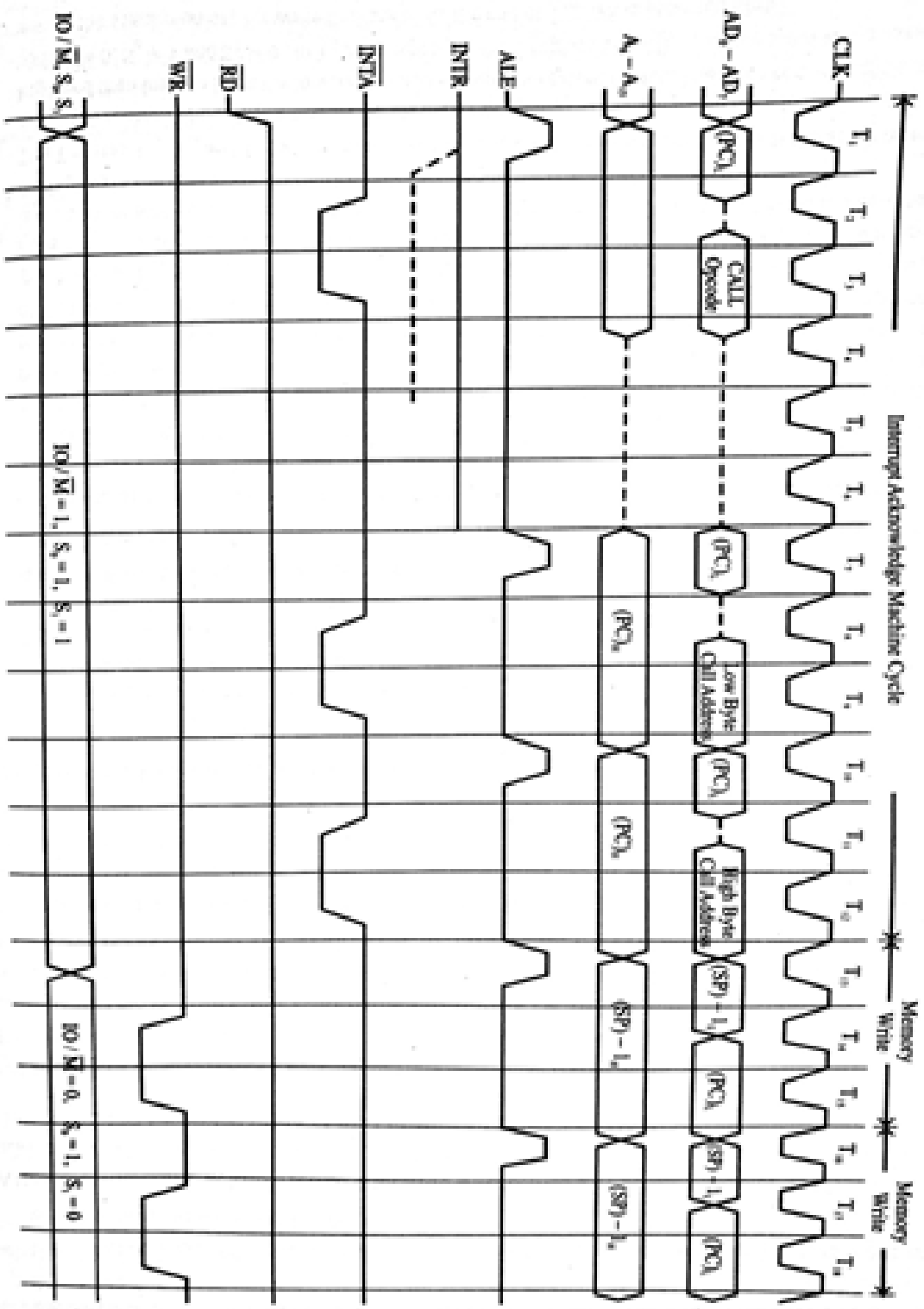
Timing diagram for INR M

- 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)



Address	Mnemonics	Opcode
4105	INR M	34_H

Timing diagram for CALL instruction:


CLK
AD7 – AD0
A15 – A8
ALE
INTR
INTA
RD
WR
IO/M, S1, S0
T1
T2
T3
T4
T5
T6
T7
T8
T9
T10
T1
T2
T3
(PC)
CALL Opcode
(PC)
Low Byte Call Address
(PC)
High Byte Call Address
(SP) – 1
(PC)
(SP) – 1
(PC)
(PC)H
(PC)L
(SP) – 1
(SP) – 1
IO/M = 1, S1 = 1, S0 = 1
IO/M = 0, S1 = 1, S0 = 0
Interrupt Acknowledge Machine Cycle
Memory Write
Memory Write

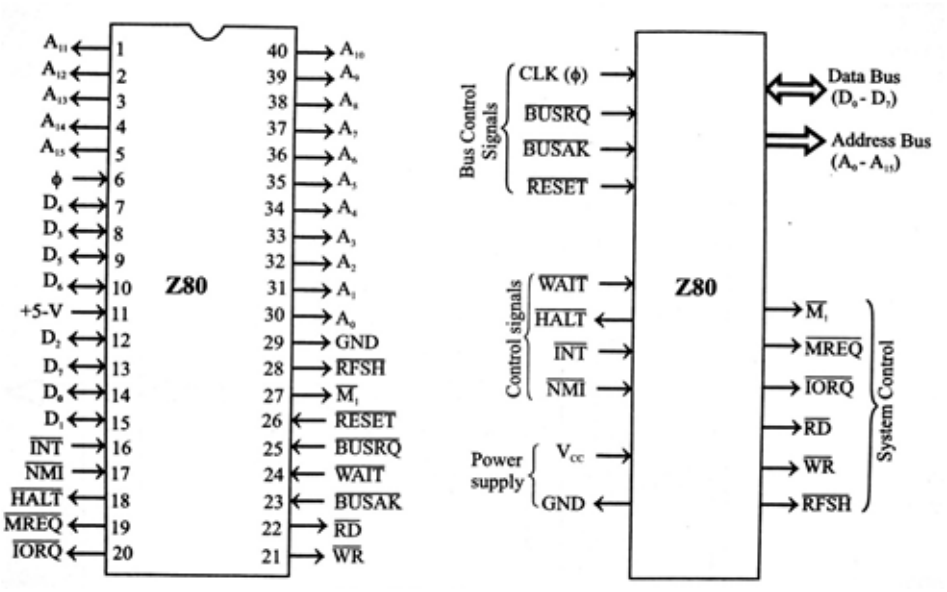
Architecture of Z80 Microprocessors.

The Z80 is one of the most talented 8 bit microprocessors, and many microprocessor-based systems are designed around the Z80. The Z80 microprocessor needs an external oscillator circuit to provide the operating frequency and appropriate control signals to communicate with memory and I/O.

The Z80 is a general-purpose 8 bit microprocessor with 16 address lines and requires a single +5V power supply. It is 40pin dual-in-line (DIP) package. The different versions of Z80 microprocessors such as Z80, Z80A, Z80B and Z80H are rated to operate at various frequencies ranging from 2.5MHz to 8MHz. Figure 3.1 shows the pin configuration of the Z80 microprocessor and its hardware model with logic signals. All the signals can be classified into six groups. 1. address bus 2. data bus 3. control signals 4. external requests 5. request acknowledge and special signals 6. power and frequency signals

Address Bus: The Z80 has 16 tri-state signal lines, A15 – A0 known as the address bus. These lines are unidirectional and capable of addressing 64K (216) memory registers. The address bus is used to send (or place) the addresses of memory registers and I/O devices.

Data Bus: The data bus consists of eight tri-state bidirectional lines D7 – D0 and is used for data transfer. On these lines, data can flow in either direction-from the microprocessor to memory and I/Os or vice versa.



Control and Status Signals:

This group consists of five individual output lines: three can be classified as status signals indicating the nature of the operation being performed, and two as control signals to read from and write into memory or I/Os.

- - Machine Cycle One: This is and active low signal indicating that an opcode is being fetched from memory. This signal is also used in an interrupt operation to generate an interrupt acknowledge signal.
- - Memory Request: This is an active low tri-state signal. This signal indicates that the address bus holds a valid address for a memory read or writes operation.
- - I/O Request: This is an active low tri-state line. This signal indicates that the low-order address bus (A7 – A0) holds a valid address for an I/O read or writes operation. This signal is also generated for an interrupt operation.
- - Read: This is an active low tri-state line. This signal indicates that the microprocessor is ready to read data from memory or an I/O device. This signal should be used in conjunction with for the Memory Read ( ) operation and withfor the I/O Read ( ) operation.
- – Write: This is an active low tri-state line. This signal indicates that the microprocessor has already placed a data byte on the data bus and is ready to write into memory or an I/O device. This signal should be used in conjunction with MREQ for the Memory Write ( ) operation and with IORQ for the I/O Write (IOWR) operation.

Power and Frequency Signals:

This group includes three signals as follows: -

- - Clock: This pin is used to connect a single phase frequency source. The Z80 does not include a clock circuit on its chip; the circuit must be built separately.
- +5V and GND - These pins are for a power supply and ground reference; the Z80 requires one +5V power source.

Z80 Programming Model

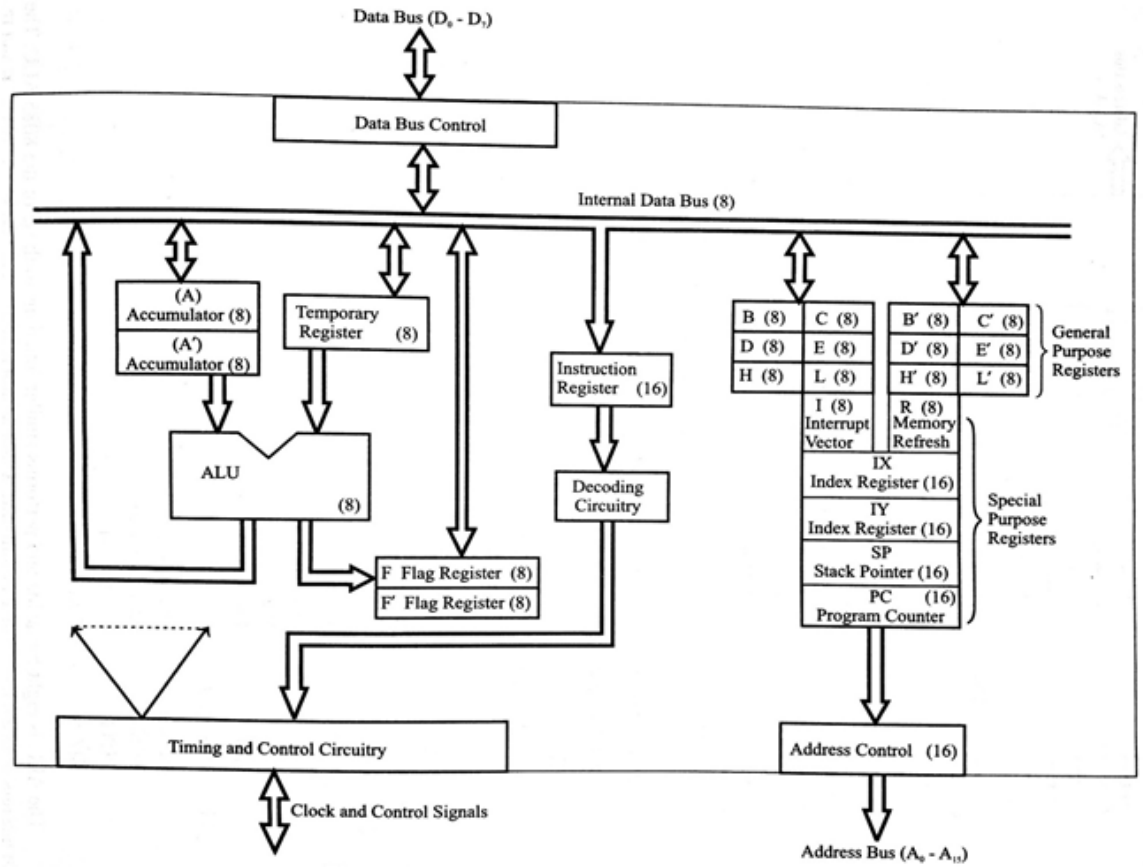
- In the last chapter, we developed a model to represent the internal structure of the MPU to process data. Now, we will describe a similar model of the Z80 microprocessor , The model includes an accumulator and a flag register, general-purpose register arrays, registers as memory pointers, and special-purpose registers.

General-Purpose Registers

- The Z80 microprocessor has six programmable general-purpose registers named B, C, D, E, H, and L, as shown in Figure 3.2. These are 8-bit registers used for storing data during the program execution. They can be combined as register pairs – BC, DE, and HL – to perform 16-bit operations or to hold memory addresses. The programmer can use these registers to load or copy data.

Accumulator

- The accumulator is an 8-bit register that is part of the arithmetic logic unit (ALU) and is also identified as register A. This register is used to store 8-bit data and to perform arithmetic and logic operations. The result of an operation performed in the ALU is stored in the accumulator.



Flag Register

The ALU includes six flip-flops that are set or reset according to data conditions after an ALU operation, and the status of these flip-flops, also known as flags, is stores in the 8-bit flag register. For example, after an addition in which the result generates a carry, the carry flip-flop

will be set and bit D0 in the flag register will show logic 1. The bit position of each flag is shown in Figure 3.2(b); bits D5 and D3 are unused. Among the six flags, the H (Half-Carry) and N (Add or Subtract) flags are used internally by the microprocessor for BCD (Binary Coded Decimal) operations. Each of the remaining four flags – S (Sign), Z (Zero), P/V (Parity or Overflow), and C (Carry) – has two Jump or Call instructions associated with it: one when the flag is set and the other when the flag is reset.

Architecture of MC6800 Microprocessors.(April/May 2016)

The MC6800 is a monolithic 8-bit microprocessor forming the central control function for Motorola's M68~ family. Compatible with TTL, the MC6B~, as with all M6800 system parts, requires only one + 5.O-volt power supply, and no external TTL devices for bus interface. The MC6800 is capable of addressing 64K bytes of memory with its 16-bit address lines. The 8-bit data bus is bidirectional as well as threestate, making direct memory addressing and multiprocessing applications realizable.

Program memory - program can be located anywhere in memory. Jump and subroutine call instructions can be used to jump anywhere in memory. Conditional and unconditional branches are limited to memory addresses positioned no farther than -125 - +129 bytes from the branch instruction.

Data memory - data can be anywhere in memory space

Stack memory - stack can be placed anywhere in memory space.

Interrupts

IRQ - maskable interrupt. When the interrupt occurs the program counter, index register, accumulators and condition code registers are stored in the stack, the further interrupts are disabled and the processor jumps to memory location address of which is stored in memory FFF8h - FFF9h. To return from the interrupt the processing routine should use RTI instruction. This interrupt can be enabled/disabled using CLI/SEI instructions.

NMI - non-maskable interrupt. When the interrupt occurs the program counter, index register, accumulators and condition code registers are stored in the stack, the further interrupts are disabled and the processor jumps to memory location address of which is stored in memory FFFCh - FFFDh. To return from the interrupt the processing routine should use RTI instruction. This interrupt can not be disabled.

SWI - software interrupt. This interrupt can be only invoked from the program. When the interrupt occurs the processor stores the program counter, index register, accumulators and condition code registers in the stack, disables the further interrupts and jumps to memory location address of which is stored in memory FFFAh - FFFBh. To return from the interrupt the processing routine should use RTI instruction. This interrupt can not be disabled.

Registers

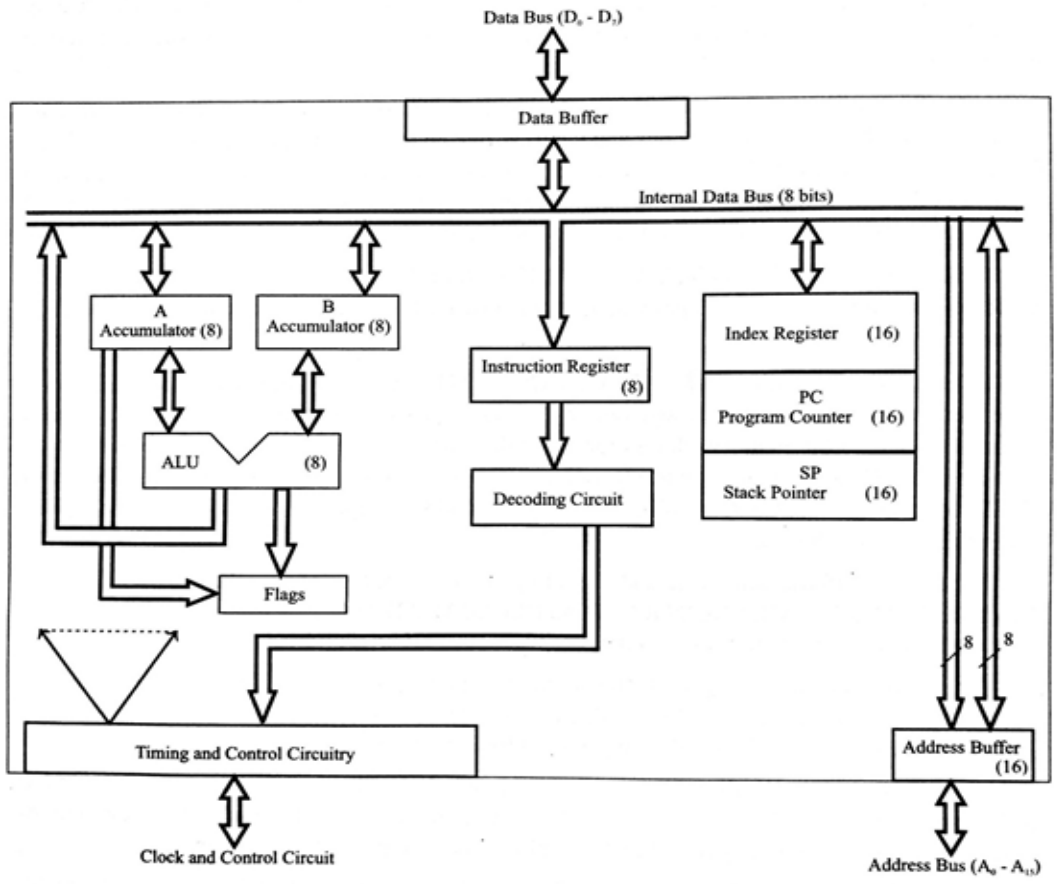
Accumulator A (ACCA) is an 8-bit register used for arithmetic and logic operations.

Accumulator B (ACCB) is an 8-bit register used for arithmetic and logic operations.

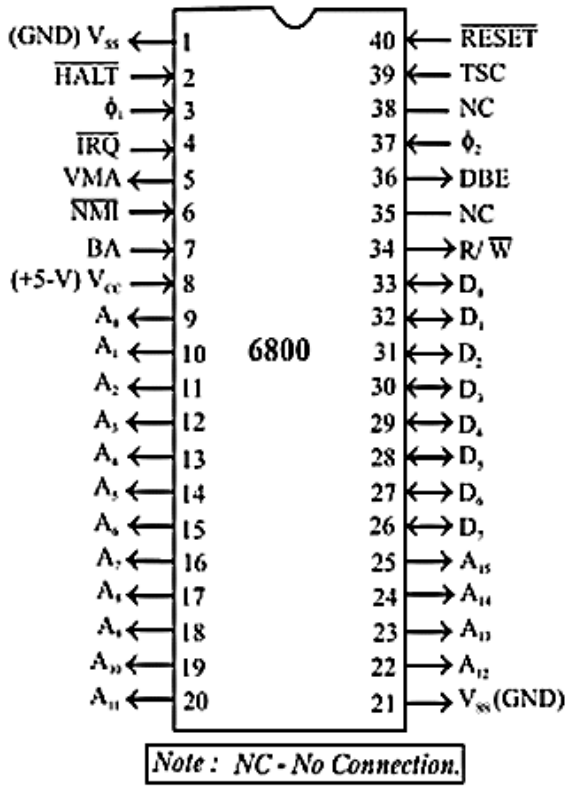
Index (IX) is a 16-bit register usually used for temporary storage or as an index when indexed addressing is used.

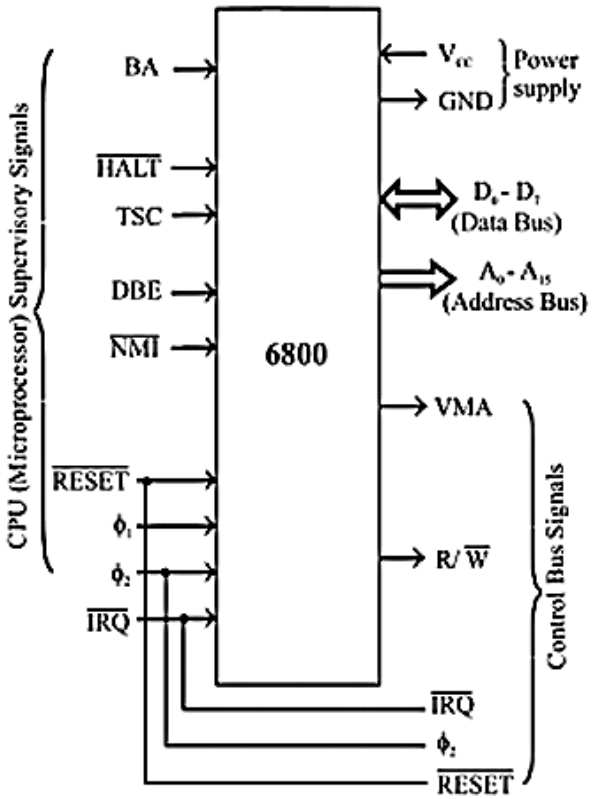
Program counter (PC) is a 16-bit register.

Stack pointer (SP) is a 16-bit register.



Motorola 6800: Pin Details





University Questions

2MARKS

1.What is meant by WAIT state? (Apr 2015) (Nov 2017)

2.State the function of Hold pin in 8085.(Apr 2015)

3.List four operation commonly performed by the Microprocessor?(Nov 2015)

4.Define Address bus. Why address bus is unidirectional .(Nov 2015) (May 2017)

5.Explain the function of the ALE signal of the 8085 Microprocessor(Dec 2014)

6.What is memory mapped I/O?(Dec 2014)

7.Define machine cycle.(Nov 2014)

8.Mention the address and data bus size of Z80 and MC6800 Microprocessor(Apr 2016)

9.Define a) instruction cycle. b) machine cycle. c) T-state (Apr 2016, May 2016)(May2018)

10.Write the significance of Arithmetic and logic section of microprocessor.(May 2016)

11.What are the operations performed by the ALU of any microprocessor?(Apr 2014)

12.What is the necessity of program counter in 8085 microprocessor?(Apr 2014)

13. Give the flag register format of 8085. (May2018) (Nov 2017)

14.List the operations performed by the timing and control section of a microprpcessor. (May2018)

15.Compare the microprocessors Z80 and 8085 in –terms of multiplexing of data lines and operating frequency. (May2018)

16. Name the types of buses in microprocessor. (Nov 2017)

17.Define Microprocessor. (May 2017)

18.What is the need for interfacing. (May 2017)

19.How are the address and data lines demultiplexed in 8085. (May 2017)

20.State the use of RESET IN and RESET OUT pins of 8085 processor. (Nov 2016)

21. Specify the size of data ,address, memory word and memory capacity of 8085 microprocessor. (Nov 2016)

22. Why databus is bi-directional. (Nov 2016)

11 MARKS

1.Draw and explain the bus architecture of 8085 (5) (April 2015).

2.Explain the various flags of 8085 microprocessor. (11) (April 2015)

3.Draw the architecture of 8085 microprocessor and explain each block in detail.(11)(Nov 2015) (Nov/Dec 2014) (Nov 2014) (May 2018) (Nov 2017) (May 2017) (Nov 2016) (or) Explain the salient features of8085 microprocessors with its functional block diagram.(May 2018)

4.Draw the timing diagram of 8085 microprocessor for the execution of IN instruction and explain the various control signal used.(11) (Nov 2015) (Nov 2016)

5.Explain the memory read cycle and memory write cycle with timing diagrams.(11) (Nov/Dec 2014) (Nov 2017)

6.Draw the timing diagram of memory read operation and explain. (7) (Nov 2014)

7.Describe the functional block diagram of 8085. (11) (April 2016)

8.Explain the various machine cycles supported by 8085.(11) (April 2016) (May 2017)

9.Draw the functional block diagram of 8085 microprocessor and explain the significance of each block. (11) (April/May 2016) (April/May 2014)

10.Explain the salient features of MC6800 CPU with its architectural diagram . How it differs from Z80? (11) (April/May 2016) (Nov 2016)

11.Draw the timing diagram of the instruction STA 526A architecture of 8085 and explain the various machine cycle involved in execution . (11) (April/May 2014) (May 2018)

12. Briefly explain about the Z80 and MC6800 architectures. (May 2018)

13. Draw the pin diagram of 8085 microprocessor . (Nov 2016)

****

srivenkateshwaraa College of Engineering & Technology
(Approved by AICTE, New Delhi & Affiliated to Pondicherry University, Puducherry)
13-A, Pondy - Villupuram Main Road, Ariyur, Puducherry - 605 102.
ASPIRE TO EXCEL
ISO 21001

Department of Electrical and Electronics Engineering

Subject Name: **Microprocessors and Microcontrollers** Subject Code: **EE T63**

Prepared by:

Mrs.S.Punitha, Assistant Professor/EEE
Mr.V.Malarselvam, Assistant Professor/EEE

Verified by: **Approved by:**

UNIT – II

8085 PROGRAMMING

Addressing modes
-Conditionflags-
Instructionset–
Programmingtechniques
–Arithmeticand logicoperationson8/16bitbinary/BCDnumbers,
Counterandtimedelay programs
–Stackand subroutines
-Code conversion.
Softwaredevelopment systems and assemblers.

Addressing modes of 8085 Microprocessor.(November 2015). (April/May 2016).

ADDRESSING MODES OF 8085

- Every instruction of a program has to operate on a data.
- The method of specifying the data to be operated by the instruction is called Addressing**.**
- The various ways of specifying the operand in the operand field of an instruction are called the addressing modes.

The various addressing modes are:

1. **Direct addressing mode**
2. **Immediate addressing mode**
3. **Register direct addressing mode**
4. **Register indirect addressing mode**
5. **Implicit addressing mode**

Present in Syllabus

Additional addressing modes :

6. Stack addressing mode
7. Indirect addressing mode
8. Indexed addressing mode
9. Relative addressing mode

Not in Syllabus.

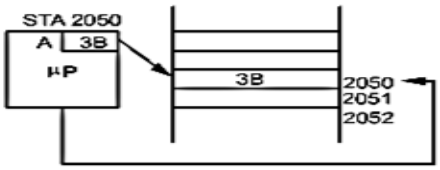
1. DIRECT ADDRESSING MODE:

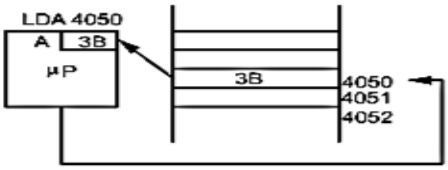
In direct addressing mode, the address of the operand is directly specified in the instruction. Except IN and OUT instructions all other direct addressing modes are 3-bytes long.

Examples:

a) STA 16-bit Address – The contents of the accumulator are copied to a memory location whose address is specified.

b) LDA 16-bit Address – The contents of the memory location whose address is specified in byte 2 and byte 3 of the instructions are copied to the Accumulator. It is 3-byte instruction.





2. IMMEDIATE ADDRESSING MODE:

In immediate addressing mode, the actual data (8-bit or 16-bit) is part of the instruction. The length of the instruction may be two or three bytes. The first byte specifies opcode. If it is a two byte instruction, the second byte specifies an 8-bit data. If it is a three byte instruction, the second and third bytes specify 16-bit data.

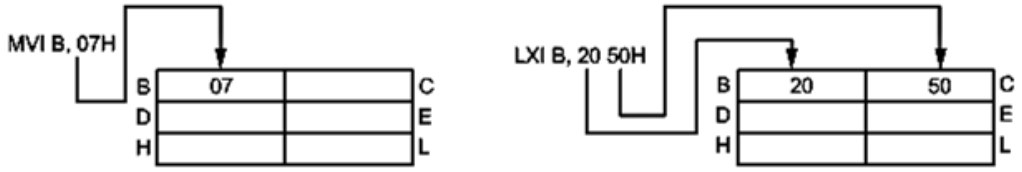
Examples:

a) MVI R, 8-bit Data

It is a 2 byte instruction. Byte 2 (8-bit data) of the instruction is immediately moved to register R (R may be A, B, C, D, E, H, L).

b) LXI R_P, 16-bit Data

It is a 3-byte instruction. Byte 2 of the instruction is immediately moved into the low-order register of the register pair $R_p$and byte 3 of the instruction immediately moved into the high-order register of the register pair.



3. REGISTER DIRECT ADDRESSING MODE:

In some instructions, general-purpose registers are specified as the address of the operands. Such instructions are called as register direct addressing mode instructions. These instructions are one byte long. Since the microprocessor need not fetch data from memory, this mode of addressing is faster than direct addressing mode. This mode of addressing is called as register addressing mode.

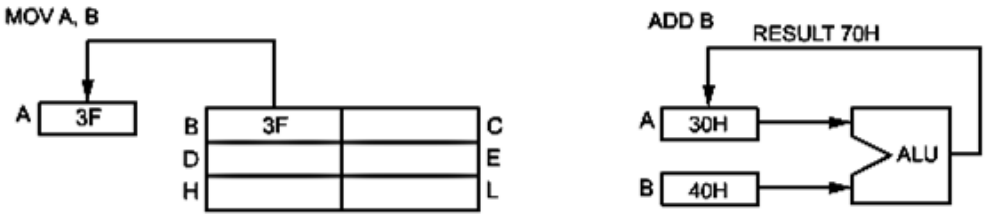
Examples:

a) MOV R_d, R_s

The contents of the source register (R_s) are moved to the destination register (R_d). It is a single byte instruction (R_d and $R_s$are general purpose registers).

b) ADD R

It is a single byte instruction. The contents of the register pair 'R_p'. (R_p may be BC, DE, HL) are incremented by 1.



4. REGISTER INDIRECT ADDRESSING MODE:

In register indirect addressing mode the contents of the specified register pair is used as the address of the operand. The register pair contains the 16-bit address of the memory location where the actual operand is stored. Usually the memory pointer (HL-register pair)contains the address of the memory location.

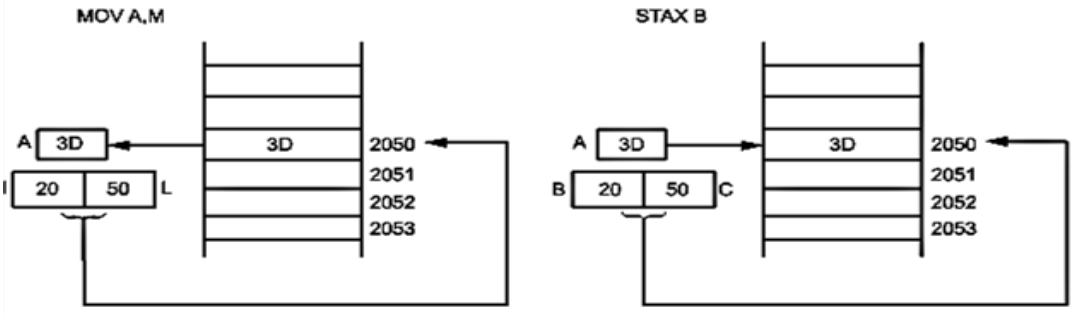
Examples

a) MOV R_d ,M

It is a single byte instruction. The contents of the memory location whose address is specified by the contents of the HL-register pair are moved to the destination register R_d (R_d may be any one of the general purpose register).

b) ADD M

It is a single byte instruction. The contents of the memory location whose address is specifies by the contents of the memory pointer (HL – register pair) are added with the contents of the accumulator and the result is placed in the accumulator.



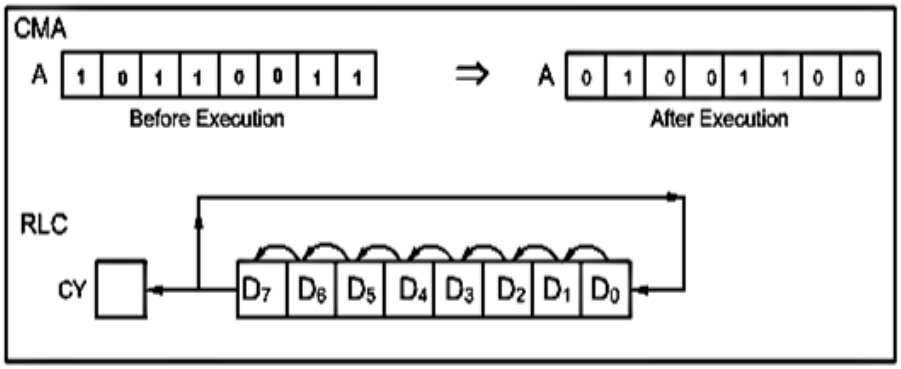
5. IMPLICIT ADDRESSING MODE

In implicit addressing mode, the address of a register (Accumulator in the case of 8085 containing the operand data) is implicitly stated in the opcode itself. In this addressing mode, the instructions are one byte long since the operand is in the accumulator.

Examples

a) CMA – It is a single byte instruction. The content of the accumulator is completed. There the operand (data) which is nothing but the contents of accumulator is specified within the instruction.

b) RLC – It is a single byte instruction. The contents of the accumulator are rotated left by one position.

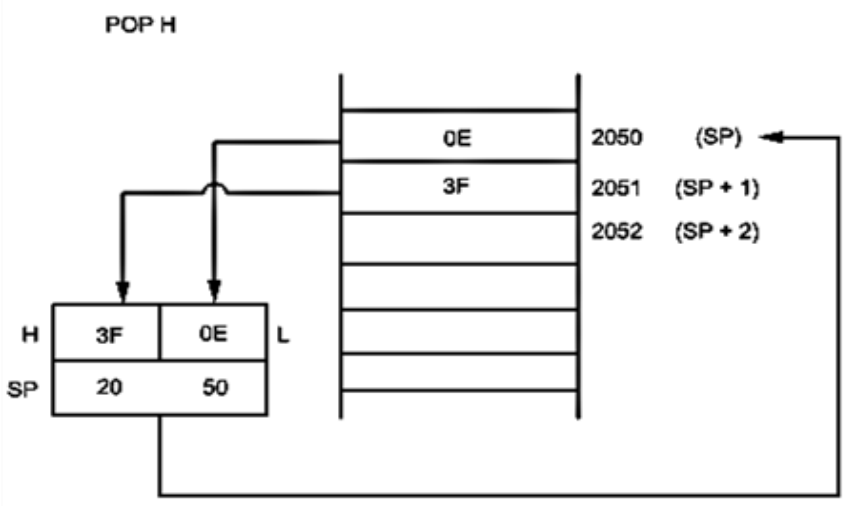


6. STACK ADDRESSING MODE

In this addressing the content of the stack pointer (16-bit register) is the address of the operand (data). It is similar to register addressing mode. Here the stack pointer content is the address of the stack memory.

Examples

POP R_p – It is a single byte instruction. The contents of the memory location pointed by the stack pointer are copied to the low-order-register. The stack pointer is incremented by 1 and contents of that memory location are copied to the high order register.

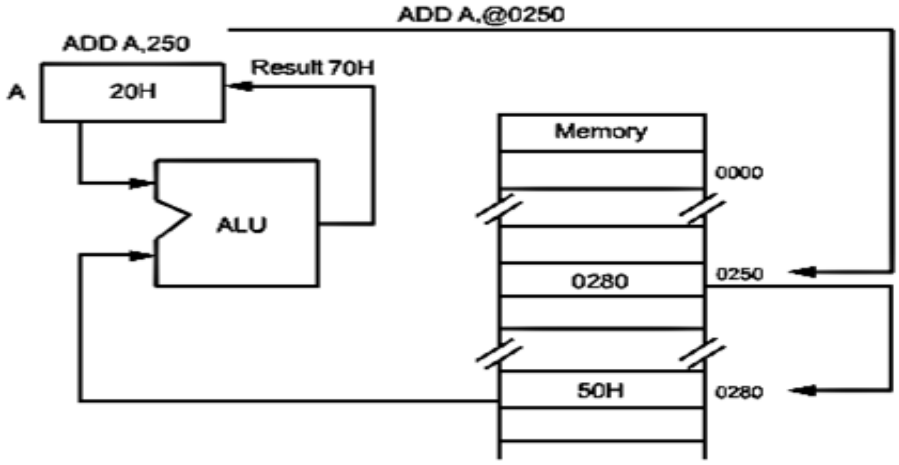


7. INDIRECT ADDRESSING MODE:

In Indirect addressing mode, the instruction points to an Address where the exact address of the operand is present.

Examples

ADD A, 2050 – In this instruction the contents of the accumulator are added with the content of memory location, whose Address is specified by the operand of the instruction.



8. INDEXED ADDRESSING MODE

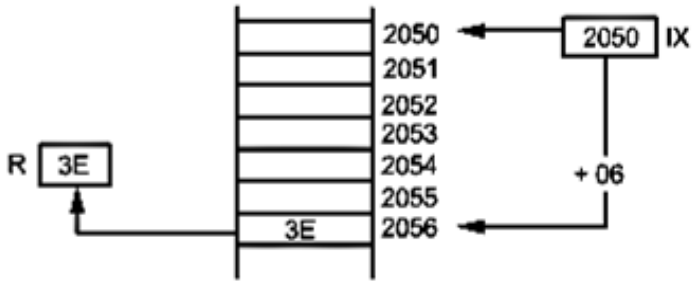
In this Addressing mode the Address of the operand is specified in relation to the contents of a 16-bit register called "Index Register". A displacement, which is to be added with the contents of the index, register is also given in the instruction itself. This type of Addressing mode is used in Z-80 microprocessor.

Examples

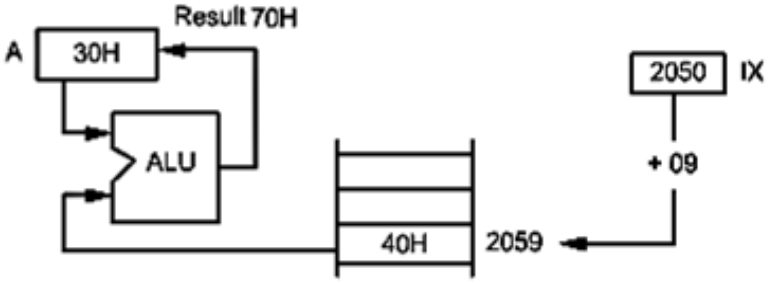
a) **LD R, (IX + d) –** This instruction will move the contents of the memory location specified by (IX + d) into specified register.

b) **ADD A, (IX +d) –** This instruction will add the contents of the memory location specified by (IX + d) with the contents of the accumulator.

LD R, (IX+d)
Let IX = 2050
d = 06
∴ LD R, (2050+06)
LD R, 2056



ADD A, (IX+d)
Let IX = 2050
d = 09
∴ ADD A, (2050+09)
ADD A,2059



9. RELATIVE ADDRESSING MODE

In this type of addressing mode, in order to find the effective address the address specified in the instruction (referred as offset) is added with the contents of PC. This mode of addressing is used in Motorola 6800 and Z – 80 microprocessor.
Offset is the displacement of the branching location from the Branch instruction.

Offset = address of the loop – address of next instruction to the branch instruction.

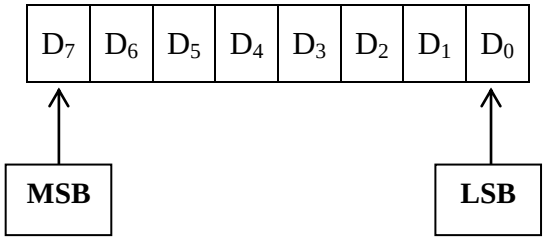
Instruction sets of 8085 Microprocessor. (November 2015).(April 2016).(April/May 2014).

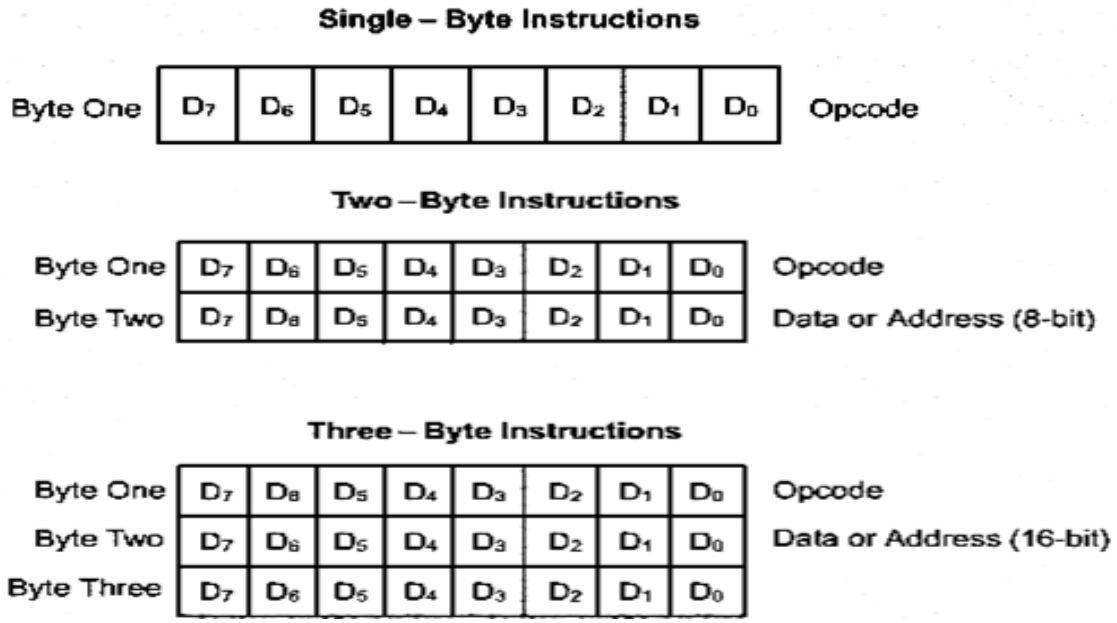
The 8085A implements a group of instructions that move data between registers, between a register and memory, and between registers and an I/O Port. It also has arithmetic and logic instructions, conditional branch instructions. The CPU recognizes these instructions only when they are coded in binary form.

- Instruction consists of
  - 74 operation codes, e.g. MOV ,ADD, SUI.
  - 246 Instructions, e.g. MOV A,B

Instruction and Data Formats

Data in the 8085A is stored in the form of 8-bit values.





The complete 8085 instruction set is described, grouped under five different functional headings, as follows:

1. DATATRANSFERINSTRUCTIONS

It includes the instruction that moves (copies) data between memory location and register. In all data transfer operations the content of source register / memory is not altered. Hence the data transfer is copying instruction.

Opcode	Operand	Description
Copyfrom source to destination		
MOV	**R_d,R_s**	Thisinstructioncopies the contents of the source Registerinto the destination register;the contents of Thesourceregisterarenotaltered. Ifoneoftheoperandsisa Memorylocation,itslocationisspecifiedbythecontentsof TheHL registers.
	M,R_s	
	R_d,M	

	Example:MOVB, C or MOVB, M

Loadaccumulator

LDA	**16-bitaddress**	Thecontents of a memory location, specifiedby a 16-bitaddressintheoperand,arecopiedtotheaccumulator. Thecontents of the source are not altered. **Example:**LDA 2034H

Storeaccumulator direct

STA	**16-bitaddress**	Thecontentsoftheaccumulatorarecopiedintothememory Locationspecifiedbytheoperand. Thisisa3-byteinstruction, Thesecondbytespecifiesthelow-orderaddressandthethird Bytespecifiesthe high-orderaddress. **Example:**STA 4350H

ExchangeH and L with D and E

XCHG	**none**	ThecontentsofregisterHareexchangedwiththecontentsof RegisterD,andthecontentsofregisterLareexchangedwith Thecontents of registerE. **Example:**XCHG

Pushregisterpair onto stack

PUS	**Reg.pair**	Thecontentsoftheregisterpairdesign at theoperandare Copiedontothestackinthefollowingsequence. Thestack Pointerregisterisdecrementedandthecontentsofthehigh-Orderregister(B,D,H,A)arecopiedintothatlocation. The Stackpointerregisterisdecrementedagainandthecontentsof Thelow-orderregister(C,E,L,flags)arecopiedtothat location. **Example:**PUSHB or PUSHA

Popoff stack to registerpair

POP	**Reg.pair**	Thecontentsofthememorylocationpointedoutbythestack Pointerregisterarecopiedtothelow-orderregister(C,E,L, Statusflags)oftheoperand. Thestackpointerisincremented By1andthecontentsofthatmemorylocationarecopiedto Thehigh-orderregister(B,D,H,A)oftheoperand. Thestack Pointerregisteris again incremented by 1. **Example:**POPH or POPA

Outputdata from accumulator to a port with 8-bit address

OUT	**8-bit port address**	ThecontentsoftheaccumulatorarecopiedintotheI/O port specifiedby the operand. Example: OUT F8H

2. ARITHMETICINSTRUCTIONS

It includes the instruction which performs addition, subtraction, increment, decrement operations. The flag conditions are altered after execution of an instruction in this group.

Opcode Operand Description

Addregisteror memory to accumulator

ADD	**R**	Thecontents of the operand (registeror memory) are Addedtothecontentsoftheaccumulatorandtheresultis stored in the accumulator. If the operand is a memory location,itslocationisspecifiedbythecontentsoftheHL registers. Allflagsaremodifiedtoreflecttheresultofthe addition. **Example:**ADD B or ADD M
ADD	**M**	

Subtractimmediate from accumulator

SUI	**8-bitdata**	The8-bitdata(operand)issubtractedfromthecontentsofthe Accumulatorandtheresultisstoredintheaccumulator. All Flagsare modified to reflectthe result of the subtraction. **Example:**SUI 45H

Incrementregisterpair by 1

INX	**R**	Thecontentsofthedesignatregisterpairareincremented by 1 and the result is stored in the same place. **Example:** INX H

Decimaladjust accumulator

DAA	**none**	Thecontentsoftheaccumulatorarechangedfromabinary Valuetotwo4-bitbinarycodeddecimal(BCD)digits. Thisis Theonlyinstructionthatusestheauxiliaryflagtoperformthe BinarytoBCDconversion,and theconversionprocedureis Describedbelow. S,Z,AC,P,CYflagsarealteredtoreflect the resultsof the operation. Ifthevalueofthelow-order4-bitsintheaccumulatoris Greaterthan9orifACflagisset,theinstructionadds6tothe low-orderfour bits. Ifthevalueofthehigh-order4-bitsintheaccumulatoris Greaterthan9oriftheCarryflagisset,theinstructionadds6 to the high-orderfour bits.**Example:** DAA

3. BRANCHINGINSTRUCTIONS

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

These operations are used to control the flow of program execution

1. **Jumps**
   - Conditional jumps
   - Unconditional jumps
2. **Call&Return**
   - Conditional Call & Return
   - Unconditional Call & Return

Opcode Operand Description

Jumpunconditionally

JMP	**R**	Theprogramsequenceistransferredtothememorylocation Specifiedby the 16-bit addressgiven in the operand. **Example:**JMP2034H or JMPXYZ

Opcode Description Flag Status

Jumps Conditionally

JC	Jumpon Carry	CY= 1
JNC	Jumpon no Carry	CY= 0
JP	Jumpon positive	S= 0
JM	Jumpon minus	S= 1
JZ	Jumpon zero	Z= 1
JNZ	Jumpon no zero	Z= 0
JPE	Jumpon parity even	P= 1
JPO	Jumpon parity odd	P= 0

Unconditionalsubroutine call

CALL	**16-bitaddress**	Theprogramsequenceistransferredtothememorylocation Specifiedbythe16-bitaddressgivenintheoperand. Before Thetransfer,theaddressofthenextinstructionafterCALL (thecontents of the program counter) is pushed onto the stack. Example:CALL 2034H or CALL XYZ

Opcode Description Flag Status

Conditional subroutine call

CC	Callon Carry	CY= 1
CNC	Callon no Carry	CY= 0
CP	Callon positive	S= 0
CM	Callon minus	S= 1
CZ	Callon zero	Z= 1
CNZ	Callon no zero	Z= 0
CPE	Callon parity even	P= 1
CPO	Callon parity odd	P= 0

4.LOGICALINSTRUCTIONS

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

Opcode Operand Description

Compareregisteror memory with accumulator

CMP	**R**	The contents of the operand (registeror memory) are compared withthecontentsoftheaccumulator. Both contentsarepreserved. Theresultofthecomparisonis shown by setting the flags of the PSW as follows:
		if(A)< (reg/mem):
		carry flag is
		set if (A)= (reg/mem):
		zero flag is set
		if(A)> (reg/mem): carry and zero flags are reset
		Example:CMP B or CMP M

	M	

LogicalAND registeror memory with accumulator

ANA	**R**	Thecontents of the accumulator are logically ANDedwith Thecontentsoftheoperand(registerormemory),andthe Resultisplacedintheaccumulator. Iftheoperandisa Memorylocation,itsaddressisspecifiedbythecontentsof HLregisters. S,Z,Paremodifiedtoreflecttheresultofthe operation.CY is reset. AC is set. **Example:**ANA B or ANA M
	M	

ExclusiveOR registeror memory with accumulator

XRA	**R**	Thecontents of the accumulator are Ex-ORed with
	M	Thecontentsoftheoperand(registerormemory),andthe
		Resultisplacedintheaccumulator. Iftheoperandisa
		Memorylocation,itsaddressisspecifiedbythecontentsof
		HLregisters. S,Z,Paremodifiedtoreflecttheresultofthe
		operation.CY and AC are reset.
		Example:XRA B or XRA M

5.MACHINE CONTROLINSTRUCTIONS

It includes the instructions related to interrupts and the instructions used to halt program execution.

Opcode	Operand	Description

Haltand enter wait state

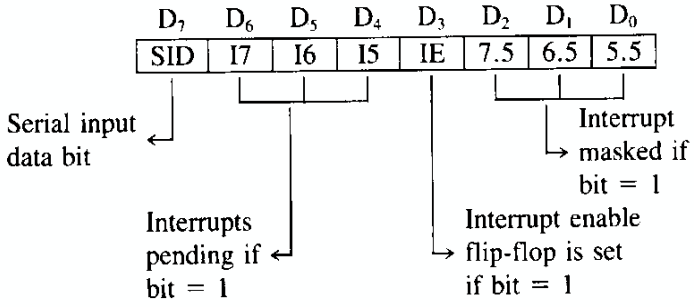
HLT	**none**	TheCPUfinishesexecutingthecurrentinstructionandhalts Anyfurtherexecution. Aninterruptorresetisnecessaryto Exitfrom the halt state. **Example**:HLT

Enableinterrupts

EI	**none**	Theinterruptenableflip-flopissetandallinterruptsare enabled. Noflagsareaffected. Afterasystemresetorthe Acknowledgementofaninterrupt,theinterruptenableflip-Flopisreset,thusdisablingtheinterrupts. Thisinstructionis Necessaryto reenable the interrupts(except TRAP). **Example**:EI

Readinterruptmask

RIM	**None**	Thisisamultipurposeinstructionusedtoreadthestatusof interrupts7.5,6.5,5.5andreadserialdatainputbit. The instruction loads eight bits in the accumulator with the Followinginterpretations. **Example:**RIM



Set interrupt mask

SIM	**None**	It is used to mask the hardware interrupts RST 7.5, RST 6.5, RST5.5.It is also used to send the data through SOD line. **Example:** RIM

Instruction DAA. (November 2014).

- **DAA-Decimal Adjust Accumulator**

- After BCD addition,this instruction is executed to get the result in BCD.when DAA instruction is executed the content of the accumulator is altered or adjusted as explained below:
    1. If the sum of the lower nibbles exceeds 09H or auxiliary carry is set,a correction 06H(0110) is added to lower nibble.
    2. If the sum of the upper nibble exceeds 09H or carry is set,a correction 06H(0110) is added to the upper nibble.

After executing this instruction all flags are modified to indicate the status of result.

One byte instruction

Implied addressing.

One machine cycle : Opcode fetch 4T states.

Total number of Instruction is =1

Programs

8 BIT ARITHMETIC OPERATIONS

Addition

```
MVI C,00H
LDA  4200 H
MOV  B,A
LDA  4201H
ADD  B
JNC LOOP
INR C
LOOP STA  4202 H
MOV A,C
STA  4203 H
HLT
```

Subtraction

```
MVI C,00H
LDA  4201H
MOV  B,A
LDA  4200H
SUB  B
```

```
JNC LOOP
 INR C
LOOP   STA  4202 H
MOV A,C
STA  4203 H
HLT
```

Multiplication

```
LXI  H,4200H
MOV  B,M
INX  H
MOV  C,M
MVI  A,00 H
LOOP      ADC  C
DCR  B
JNZ  LOOP
INX  H
MOV  M,A
HLT
```

Division

```
LDA 4201
MOV B,A
LDA 4200H
MVI C,00H
AGAIN    CMP B
JC LOOP
SUB B
INR C
JMP AGAIN
STA 4203
MOV A,C
STA 4202H
HLT
```

CODE Conversion

BINARY TO BCD CODE CONVERSION (April 2015). (Nov 2014) (April 2014)

```
LDA  4200
MVI  C,00
MVI  D,00
XYZ     CPI 64
JC  XXX
SUI  64
INR  C
JMP  XYZ
CPI  0A
JC  XYY
SUI   0A
INR  D
JMP  XXX
MOV C,A
MOV A,D
RLC
RLC
RLC
RLC
ADD C
STA 4201
MOV A,E
STA 4202
HLT
```

BCD TO BINARY CODE CONVERSION

```
LDA 4200
MOV  B,A
ANI 0F
MOV C,A
MOV A,B
ANI F0
RLC
RLC
RLC
RLC
MOV D,A
XRA A
Loop1 ADI  0A
```

```
DCR  D
JNZ Loop1
ADD C
STA 4400
HLT
```

LARGEST ELEMENT IN AN ARRAY

```
LXI H 4200
MOV B , M
MVI  A , 00
XXX   INX H
CMP M
JNC YYY
MOV A , M
YYY   DCR B
JNZ  XXX
STA 4500
HLT
```

SMALLEST ELEMENT IN AN ARRAY. (April 2015).

```
LXI H , 4200
MOV B , M
MVI A , FF
XYZ    INX H
CMP M
JC  CCC
MOV A , M
CCC     DCR B
JNZ   XYZ
STA 4500
HLT
```

SEARCH AN ELEMENT IN AN ARRAY

```
LXI H , 4200
MOV B , M
MVI A , FF
INX H
XYZ   CMP M
JZ  CCC
INX H
DCR B
JNZ   XYZ
MVI  A, FF
STA  4500
JMP   HLT
CCC   SHLD 4500
HLT
```

COUNTERS

UP COUNTER [IN HEXADECIMAL]

```
LOOP1 MVI A,00

LOOP  STA FFF9

CALL  06D3

CALL  DELAY

LDA FFF9

INR A

CPI 0A

JNZ LOOP

JMP LOOP1
```

UP COUNTER [IN DECIMAL]

```
START   MV1 A,00

LOOP   STA FFF9

CALL  06D3

CALL DELAY

LDA FFF9

INR A

DAA

CP1 10

JNZ LOOP

JMP START
```

DOWN COUNTER [IN HEXADECIMAL]

START MVI A,09

LOOP STA FFF9

CALL 06D3

CALL DELAY

LDA FFF9

DCR A

CPI 0A

JNZ **LOOP**

JMP **START**

DOWN COUNTER [IN DECIMAL]

START MV1 A,10

LOOP STA FFF9

CALL 06D3

CALL DELAY

LDA FFF9

AD1 99

DAA

CP1 FF

JC LOOP
JMP START

Program to count from 0 to 9 with a one second delay between each count. At the count of 9, the counter should reset itself to 0 and repeat the sequence continuously. Use register pair HL to set up the delay, and display each count at one of the output ports. Assume clock frequency of the 8085 microprocessor as 1 Mhz. (11) (April 2016).

MOD 10 COUNTER

LOOP1 MVI A,00

LOOP STA FFF9

CALL 06D3

CALL DELAY

LDA FFF9

INR A

CPI 0A

JNZ **LOOP**

JMP **LOOP1**

MOD 100 COUNTER

LOOP1 MVI A,00

LOOP STA FFF9

CALL 06D3

CALL DELAY

LDA FFF9

INR A

CPI 9A

JNZ **LOOP**

JMP **LOOP1**

1 SECOND DELAY:

```
DELAY     MV1 D,02
LOOP3     LX1 B,FFFF
LOOP2  DCX B
MOV A,B
ORA C
JNZ  LOOP2
DCR D
JNZ LOOP3
```

ASCENDING ORDER- SORTING

```
LDA 5400
SUI 01
MOV D,A
MOV E,A
LOOP2    MOV B,D
LXI H,5200
LOOP1    MOV A,M
INX H
CMP M
JC LOOP
MOV C,A
MOV A,M
MOV M,C
LOOP   DCX H
MOV M,A
INX H
DCR B
JNZ LOOP1
DCR E
JNZ LOOP2
HLT
```

DESCENDING ORDER-SORTING
(November 2014)

```
LDA 5400
SUI 01
MOV D,A
MOV E,A
LOOP2   MOV B,D
LXI H,5200
LOOP1    MOV A,M
INX H
CMP M
JNC LOOP
MOV C,A
MOV A,M
MOV M,C
LOOP    DCX H
MOV M,A
INX H
DCR B
JNZ LOOP1
DCR E
JNZ LOOP2
HLT
```

BCD Addition

```
LDA 9000
MOV B, A
LDA 9001
MVI C, 00
ADD B
DAA
JNC LOOP
INR C
STA 9002
MOV A, C
STA 9003
HLT
```

BCD SUBTRACTION

```
MVI E, 00
LDA 8600
MOV B, A
LDA 8601
MOV D, A
MVI A, 99
SUB D
ADI 01
ADD B
DAA
JNC LOOP
INR E
MOV C, A
MVI A, 99
SUB C
ADI 01
DAA
STA 8602
MOV A, E
STA 8603
```

16 bit addition

```
MVI B,00
LHLD 5000
XCHG
LHLD 5002
DAD D
JNC LOOP
INR B
LOOP   SHLD 5004
MOV A,B
STA 5006
HLT
```

16 bit subtraction

```
MVI B,00
LHLD 5000
XCHG
LHLD 5002
MOV A,L
SUB E
MOV L,A
MOV A,H
SBB D
MOV H,A
JNC LOOP
INR B
LOOP   SHLD 5004
MOV A,B
STA 5006
HLT
```

16 bit multiplication

```
LXI B , 0000
LHLD 4300
XCHG
LHLD 4302
SPHL
LXI H , 0000
MUL    DAD  SP
JNC  LOOP
INX B
LOOP    DCX D
MOV A , E
ORA D
JNZ  MUL
SHLD 4304
MOV L , C
MOV H , B
SHLD 4306
HLT
```

16 bit division

```
LXI D , 0000
LHLD 4300
REP   MOV C , L
MOV B , H
LHLD 4302
MOV A , L
SUB C
MOV L , A
MOV A , H
SBB B
MOV H , A
JC REM
INX D
JMP  REP
REM    DAD B
SHLD 4304
XCHG
SHLD 4306
HLT
```

Addition of three 16 bit numbers

```
DAD D
MOV D, B
MOV E, C
DAD D
XCHG
HLT
```

8 bit multiplication using logical instruction

```
MVI B 05
MVI B 05
MOV A, B
RLC
RLC
STA 3050
HLT
```

Factorial of a number

```
LXI H, 2000H
MOV B, M
MVI D, 01H
CALL MULTIPLY
DCR B
JNZ FACTORIAL
INX H
MOV M, D
HLT
MOV E, B
MVI A, 00H
ADD D
DCR E
JNZ MULTIPLYLOOP
MOV D, A
RET
```

Reversal of 8 bit number

```
LDA 2050
RLC
RLC
```

```
RLC
RLC
STA 3050
HLT
```

```
JMP 2007
MOV A, E
STA 3050
HLT
```

Square root

```
MVI D, 01
MVI E, 01
LDA 2050
SUB D
JZ 2011
INC D
INC D
INC E
```

Binary to Gray Conversion

```
STC
CMC
LDA 2050
MOV B,A
RAR
XRA B
STA 3050
HLT
```

Delay Subroutines . (November 2014).

8.7 DELAY ROUTINE

Delay routines are subroutines used for maintaining the timings of various operations in a microprocessor. In control applications, certain equipment need to be ON/OFF after a specified time delay. In some applications, a certain operation has to be repeated after a specified time interval. In such cases, simple time delay routines can be used to maintain the timings of the operations.

A delay routine is generally written as a subroutine (However it need not always be a subroutine. It can be even a part of the main program.) In delay routine, a count (number) is loaded in a register of the microprocessor. Then it is decremented by one and the zero flag is checked to verify whether the content of the register is zero or not. This process is continued until the content of register is zero. When it is zero, the time delay is over and the control is transferred to the main program to carry out the desired operation.

The delay time is given by the total time taken to execute the delay routine. It can be computed by multiplying the total number of T states required to execute the subroutine and the time for one T state of the processor. The total number of T states can be computed from the knowledge of the T states required for each instruction. The time for one T state of the processor is given by the inverse of the internal clock frequency of the processor. For example, if the 8085 microprocessor has 5 MHz quartz crystal then,

$$\text{Internal clock frequency} = \frac{5}{2} = 2.5 \text{ MHz}$$

$$\text{Time for one T-state} = \frac{1}{2.5 \times 10^6} = 0.4 \text{ ms}$$

Two examples of delay routines that can be used in 8085 assembly language programs are presented in this section with details of the timing calculations. For small time delays (< 0.5 milli-seconds) an 8-bit register can be used as the counter, but for large time delays (< 0.5 second) a 16-bit register should be used as the counter. For very large time delays (> 0.5 second), a delay routine can be repeatedly called in the main program. The disadvantage in delay routines is that the processor time is wasted. An alternate solution is to use a dedicated timer like 8253/8254 to produce time delays or to maintain timings of various operations.

University Questions

2 MARKS

1.Explain the difference between JMP instruction and CALL instruction.(Apr 2015)

2.State the difference between the SHIFT and ROTATE instruction.(Apr 2015)

3.Define T stack.(Nov 2015)

4.What is stack? (Nov 2015,apr 2016)

5.Differentiate immediate and direct addressing? (Dec 2014)

6.Explain any two logical instructions.(Dec 2014)

7.Which is the hardware interrupt having highest priority?(Nov 2014)

8.What is the vector address of the interrupt RST 6.5?(Nov 2014)

9.Distinguish between immediate and implied addressing mode of 8085.(Apr 2016).

10.Distinguish between stack and subroutine.(May 2016,May 2014)

11.What are assemblers? (May 2016)

12.calculate the time required to execute the instruction MVI using 8085 micro processor with 2Mhz clock frequency.(May 2014)

13.List the addressing modes of 8085.(May 2018)

14.Write a delay routine to produce a time delay of 0.5 msec.in 8085 processor –based system whose operating frequency is 3 MHZ. (May 2018)

15.What is meant by stack in 8085 based system?(Nov 2017)

16.What is assembler? (Nov 2017) (May 2016)

17.What are the different addressing modes of 8085.(May 2017)

18.Define the types of branching operations. (May 2017)

19.Define the function of parity flag and zero flag in 8085. (Nov 2016)

20.How address and data lines are demultiplexed in 8085. (Nov 2016)

21.Distinguish between stack and subroutine.(May 2016)

11 MARKS

1. Explain data masking with logical AND with one example. (11) (April 2015).

2. Write a program to convert 8 bit number to BCD. (6) (April 2015) (May 2018).

3. Write an ALP using 8085 instructions to find smallest elements in an array.(5) (April 2015).

4. Describe the instruction set of 8085 microprocessor. (11) (November 2015)(May 2018) (May 2017).

5. Discuss the various addressing models used in 8085 microprocessor. (11) (Nov 2015) (Nov 2017) (Nov 2016).

6. Explain the arithmetic instructions with examples. (11) (November/December 2014).

7. Write an assembly language program to convert a two digit BCD(8 bit) data to binary numbers. (11) (November/December 2014).

8. Write an assembly language program to sort an array of 'N' numbers in descending order. (11) (November 2014). (May 2017).

9. Explain DAA instruction with examples. (5) (November 2014).

10. Discuss about the software time delay technique using a register pair and compute the time delay for the maximum count 'FFFF'. (6) (November 2014).

11. Explain the different types of instructions in 8085. (11) (April 2016).

12. Write an 8085 program to count from 0 to 9 with a one second delay between each count. At the count of 9, the counter should reset itself to 0 and repeat the sequence continuously. Use register pair HL to set up the delay, and display each count at one of the output ports. Assume clock frequency of the 8085 microprocessor as 1 Mhz. (11) (April 2016).

13. List the various addressing modes of 8085. Explain them with at- least two examples, each from different instruction set.(11) (April/May 2016)(Nov 2016).

14. Explain the various logical and data transfer group of instructions used in 8085 microprocessor with examples. (11) (April/May 2014).

15.Write an assembly language program to convert 2 digit BCD code to binary code and explain with flow chart.(11) (April/May 2014).

16. Write an assembly language program to convert 2 digit BCD code to binary code and explain with flow chart.(11) (April/May 2014)(Nov 2017).

17. Write an assembly language program to multiply two 8 bit data using 8085 microprocessor . (Nov 2017).

18. (a)With suitable example, discuss about 8085 microprocessor instructions used for data manipulation. (Nov 2016).

(b)Write short note on Stack and subroutines. (Nov 2016).

19. Write an assembly language program to sort an array of 'N' numbers in ascending order. (11) (Nov 2016).

****

srivenkateshwaraa College of Engineering & Technology
(Approved by AICTE, New Delhi & Affiliated to Pondicherry University, Puducherry)
13-A, Pondy - Villupuram Main Road, Ariyur, Puducherry - 605 102.
ASPIRE TO EXCEL
ISO 21001

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

Subject Name: Microprocessors and Microcontrollers
Subject Code:EET63

Prepared by:

Mrs.S.Punitha, Assistant Professor/EEE
Mr.V.Malarselvam, Assistant Professor/EEE

Verified by: **Approved by:**

UNIT – III
UNITIII: MEMORY I/O INTERFACINGAND INTERRUPTS

Memory Interfacing-Compatibility between memory and microprocessor unit– Address space– Partitioning of address space– Interfacing input devices. Types of data transfer– 8085 Interrupt structure- vectored interrupts –Interfacing data converters.

2 MARKS

1. What is an Interrupt? Haw the interrupt are classified?

Interrupt is a signal send by an external device to the processor so as to request the processor to perform a particular task or work.

They are three methods of classifying interrupts

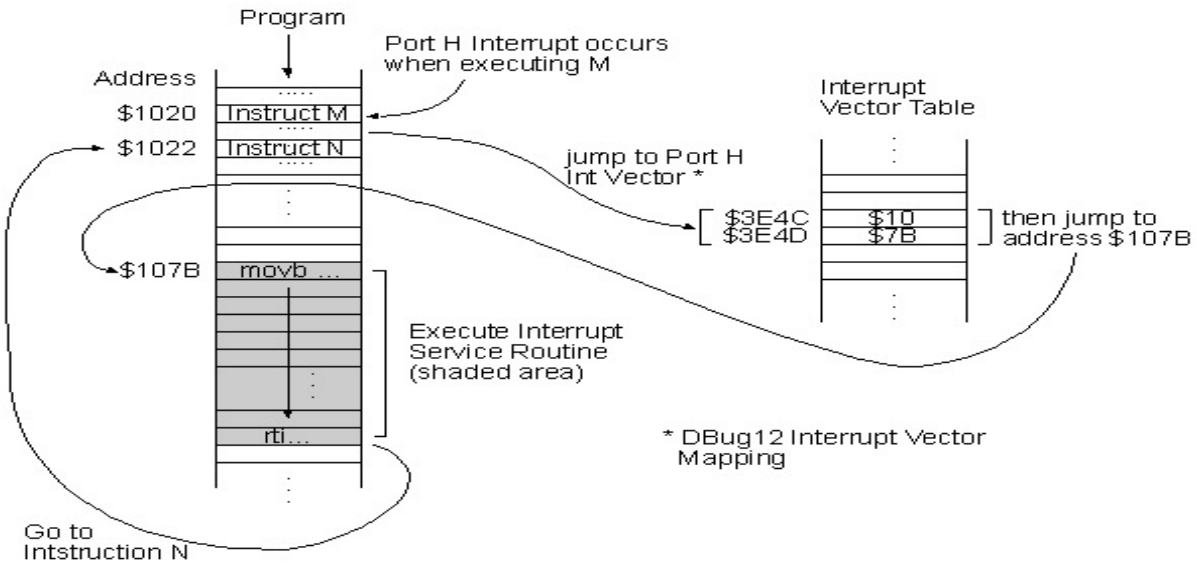
Method I :The interrupts are classified into Hardware and Software interrupts

Method II:The interrupts are classified into vectored and Non- Vectored interrupt.

2. What is Vectored and Non- Vectored interrupt?

When an interrupt is accepted, if the processor control branches to a specific address defined by the manufacturer then the interrupt is called vectored interrupt.

In Non-vectored interrupt there is no specific address for storing the interrupt service routine. Hence the interrupted device should give the address of the interrupt service routine.



3. **List the Software and Hardware interrupts of 8085 ?**

Software interrupts:

RST 0, RSTl, RST 2,
RST 3, RST 4, RST 5,
RST 6 and RST 7.

Hardware interrupts:

TRAP, RST 7.5, RST 6.5,
RST 5.5 and INTR.

4. **Write 8085 assembly language instructions to store the contents of the flagregister in memory location 2000H.**

```
PUSH PSW
POP B
MOV A,C
STA 2000H.
HLT
```

5. **When the 8085 processor checks for an interrupt?**

In the second T -state of the last machine cycle of every instruction, the 8085 processor checks whether an interrupt request is made or not.

6. **What is interrupt acknowledge cycle?**

The interrupt acknowledge cycle is a machine cycle executed by 8085 processor to get the address of the interrupt service routine in-order to service the interrupt device.

7. **How the interrupts are affected by system reset?**

Whenever the processor or system is resetted , all the interrupts except TRAP are disabled.

In order to enable the interrupts, El instruction has to be executed after a reset.

8. What is Software interrupts?

The Software interrupts are program instructions. These instructions are inserted at desired locations in a program. While running a program, if software interrupt instruction is encountered then the processor executes an interrupt service routine.

9. What is Hardware interrupt?

If an interrupt is initiated in a processor by an appropriate signal at the interrupt pin, then the interrupt is called Hardware interrupt.

10. What is the difference between Hardware and Software interrupt?

The Software interrupt is initiated by the main program, but the Hardware interrupt is initiated by an external device.

In 8085, the Software interrupt cannot be disabled or masked but the Hardware interrupt except TRAP can be disabled or masked.

11. What is TRAP?

The TRAP is non-maskable interrupt of8085. It is not disabled by processor reset or after reorganization of interrupt.

12. Whether HOLD has higher priority than TRAP or not?

The interrupts including mAP are recognized only if the HOLD is not valid, hence TRAP has lower priority than HOLD.

13. What is masking and why it is required?(Apr.2016)

Masking is preventing the interrupt from disturbing the current program execution. When the processor is performing an important job (process) and if the process should not be interrupted then all the interrupts should be masked or disabled.

In processor with multiple 'interrupts, the lower priority interrupt can be masked so as to prevent it from interrupting, the execution of interrupt service routine of higher priority interrupt.

14. When the 8085 processor accept hardware interrupt?

The processor keeps on checking the interrupt pins at the second T -state of last Machine cycle of every instruction. If the processor finds a valid interrupt signal and if the interrupt is unmasked and enabled then the processor accepts the interrupt. The acceptance of the interrupt is acknowledged by sending an OOA signal to the interrupted device.

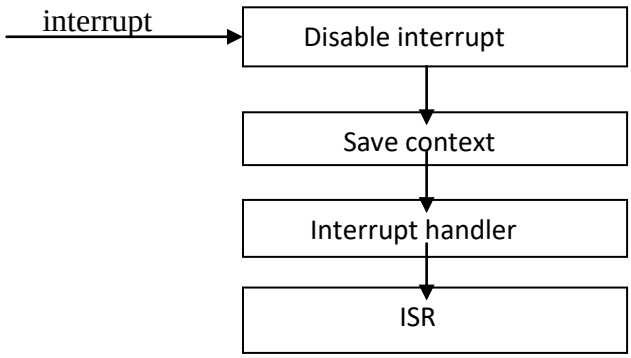
15. When the 8085 processor will disable the interrupt system?

The interrupts of 8085 except TRAP are disabled after anyone of the following operations

i. Executing El instruction.
ii. System or processor reset.
iii. After reorganization (acceptance) of an interrupt.

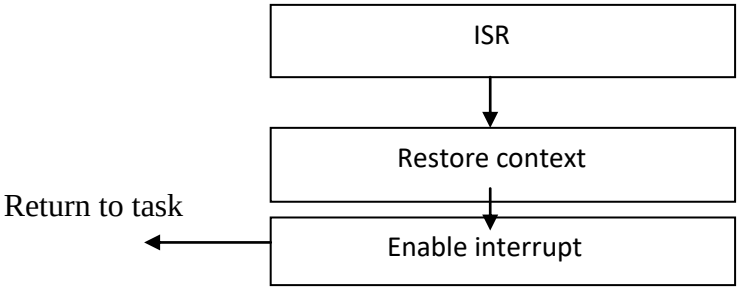
16. What is the function performed by Dl instruction?

The function of Dl instruction is to enable the disabled interrupt system.



17. What is the function performed by El instruction?

The El instruction can be used to enable the interrupts after disabling.



18. How the vector address is generated for the INTR interrupt of 8085?

For the interrupt INTR, the interrupting device has to place either RST opcode or CALL opcode followed by l6-bit address. I~RST opcode is placed then the corresponding vector address is generated by the processor. In case of CALL opcode the given l6-bit address will be the vector address.

19. How clock signals are generated in 8085 and what is the frequency of theinternal clock?

The 8085 has the clock generation circuit on the chip but an external quartz crystal or L C circuit or RC circuit should be connected at the pins XI and X2. The maximum internal clock frequency of 8085A is 3.03 MHz.

20. What happens to the 8085 processor when it is resetted?

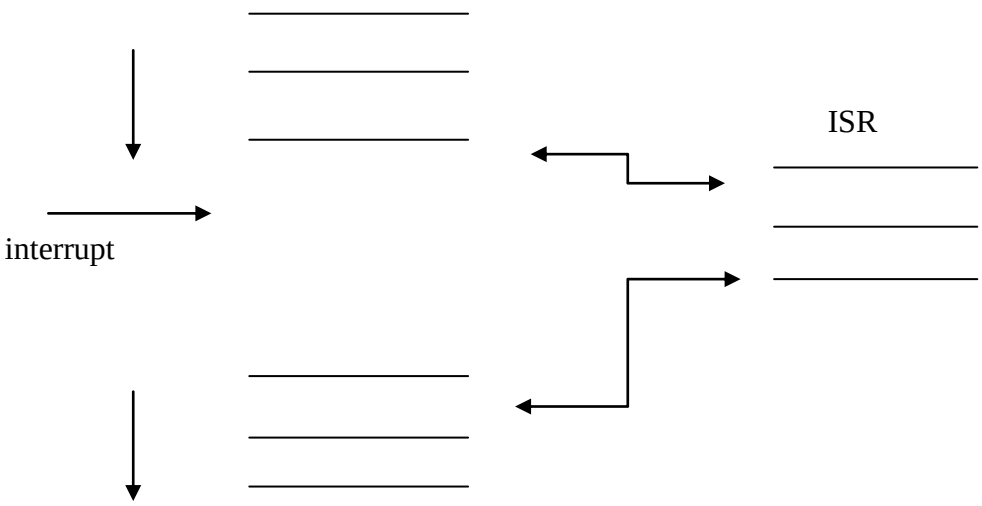
When the 8085 processor is resetted it execute the first instruction at the OOOOH location. The 8085 resets (clears) instruction register, interrupt mask bits and other registers.

21. How a microprocessor services an interrupt request?

When the processor recognizes an interrupt , it saves the processor status in stack then calls and executes an interrupt service routine (ISR). At the end of ISR, it restores the processor status and the program control is transferred to the main program.

22. What is the role of interrupt service routine?

For each interrupt the processor has to perform a specific job . An interrupt service routine has been developed in order to perform the operations required for a device that is interrupting the processor.



23. What is vectoring?

Vectoring is the process of generating the address of interrupt service routine to be loaded into the program counter .

24. List the type of signals that has to be applied to initiate a hardware interruptin 8085?

The TRAP is level and edge sensitive and so the interrupt signal has to take a low to high transition and then remain high until it is recognized .The RST 7.5 is edge sensitive and so the interrupt signal has to take a low to high transition and need not remain high until it is recognized. The RST 6.5 , RST 5.5 and INTR are level sensitive and so the interrupt signal should be high until the interrupt is recognized.

25. List the four interrupts which controls the interrupt structure of 8085.(Apr.2015)

- DI(Disable Interrupts)
- EI(Enable interrupts)
- RIM (Read Interrupt Masks)
- SIM (Set interrupt Masks)

26. **Explain priority interrupts of 8085.**

The 8085 microprocessor has five interrupt inputs. They are TRAP, RST 7.5, RST 6.5, RST 5.5, and INTR. These interrupts have a fixed priority of interrupt service. If two or more interrupts go high at the same time, the 8085 will service them on priority basis. The TRAP has the highest priority followed by RST 7.5, RST 6.5, RST 5.5. The priority of interrupts in 8085 is shown in the table.

TRAP	**1**
RST 7.5	**2**
RST 6.5	**3**
RST 5.5	**4**
INTR	**5**

27. What are vector interrupts? Give examples.(Apr./May.2014) or What is the significance of vectored interrupts?(Nov.2015)

A vectored interrupt is where the CPU actually knows the address of the Interrupt Service Routine in advance. All it needs is that the interrupting device sends its unique vector via a data bus and through its I/O interface to the CPU. The CPU takes this vector, checks an interrupt table in memory, and then carries out the correct ISR for that device. So the vectored interrupt allows the CPU to be able to know what ISR to carry out in software (memory). Eg. RST0, RST1, RST 7.5, TRAP etc

28.**What are the maskable and non-maskable interrupts of 8085? (Nov./Dec.2014)**

The TRAP is non-maskableinterrupt . The RST 7.5 ,RST 6.5 and RST 5.5 are maskable interrupts . The INTR of 8085 can also be disabled by DI-Instructions.

29.How the interrupt INTR of 8085 can be expanded?

The interrupt INTR of 8085 can be expanded of eight interrupts using 8-to-3 Priority encoder . It can also be expanded to 8 interrupts using one number of 8259(Programmable interrupt controller) or upto 64 interrupts using 8259's in cascaded mode.

30.How to check whether an 8085 is masked or not?

The masking status of an 8085 interrupt can be obtained by executing RIM instruction. When RIM instruction is executed an 8-bit data is loaded in accumulator. The bits Bo , B1 and B2 will give the masking status of RST 5.5, RST 6.5 and RST 7.5 respectively. If this bit is 1 , then the corresponding interrupt is masked otherwise it is unmasked .

SID	I7.5	I6.5	I5.5	IE	M7.5	M6.5	M5.5

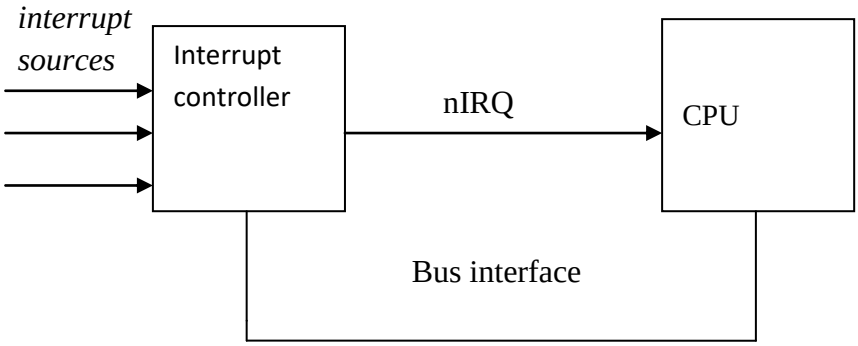
31. How to check the interrupt request pending status 8085 interrupt?

The pending status of an 8085 interrupt can be obtained by executing RIM instruction . When RIM instruction is executed an 8-bit data is loaded in accumulator . The bits B4 , B5 and B6 will give the pending status of RST 5.5, RST 6.5 and RST 7.5 respectively . If this bit is one then the interrupt is pending otherwise it is not pending.

D7	D6	D5	D4	D3	D2	D1	D0
SID	I7.5	I6.5	I5.5	IE	M7.5	M6.5	M5.5

32.What is the need for interrupt controller ?

The interrupt controller is employed to expand the interrupt input .It can handle the interrupt request from various devices and allow one by one to the processor.



33.How the vector addresses are generated for the hardware interrupts of 8085?

For the hardware interrupts trap, RST 7.5, RST 6.5,and RST 5.5 the vector addresses are generated by the processor itself. These addresses are fixed by the manufacturer.

34.How the vector addresses are generated for software interrupts?

For the software interrupts RST0 to RST 7, the vector addresses are generated internal to the processor. These vector addresses are fixed but the manufacturer.

35.List the INTEL predefined interrupts

1) Division by zero(type-0 interrupt)
2) Single step(type -1 interrupt)
3) Non-maskable interrupt(type-2 interrupt)
4) Breakpoint interrupt(type-3 interrupt)
5) Interrupt on overflow(type-4 interrupt)

36.How the hardware interrupt of 8085 can be masked or unmasked?

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

SOD	SID	X	R7.5	MSE	M7.5	M6.5	M5.5

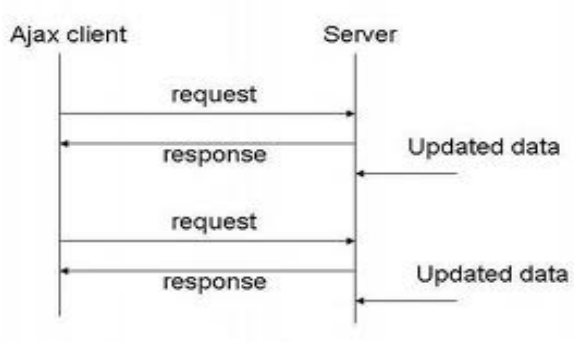
37.**How the status of maskable interrupts can be read in 8085 processor?**

The status of hardware interrupts like interrupt request pending or not, interrupts enabled or not,and masked or unmasked can 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.

S	I7.5	I6.5	I5.5	IE	M7.5	M6.5	M5.5

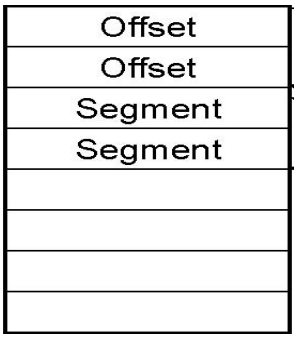
38.**What is polling? (Apr.2015)**

Polling is a CAM. In a master/slave scenario, the master queries each slave device in turn as to whether it has any data to transmit. If the slave answers yes then the device is permitted to transmit its data. If the slave answers no then the master moves on and polls the next slave device. The process is repeated continuously.



39.What is vector table? Where it is located?

The memory block consisting of vector addresses of all the 256 types of interrupts of 8086 is called vector table. The vector table is stored ib the first 1kb of physical memory space.



40. Classify the modes of operations in 8255 PPI.(Nov.2014)

- **Mode 0** − In this mode, Port A and B is used as two 8-bit ports and Port C as two 4-bit ports. Each port can be programmed in either input mode or output mode where outputs are latched and inputs are not latched. Ports do not have interrupt capability.
- **Mode 1** − In this mode, Port A and B is used as 8-bit I/O ports. They can be configured as either input or output ports. Each port uses three lines from port C as handshake signals. Inputs and outputs are latched.
- **Mode 2** − In this mode, Port A can be configured as the bidirectional port and Port B either in Mode 0 or Mode 1. Port A uses five signals from Port C as handshake signals for data transfer. The remaining three signals from Port C can be used either as simple I/O or as handshake for port B.

41. Compare memory mapped I/O and peripheral mapped /O.(Apr.2016),(Apr./May.2014)

S.No	Memory mapping of I/O device	I/O mapping of I/O device
1.	16-bit addresses are provided for I/O devices.	8-bit addresses are provided for I/O devices.
2.	The devices are accessed by memory read or memory write cycles.	The devices are accessed by I/O read or I/O write cycle. During these cycles the 8-bit address is available on both low order address lines and high order address lines.
3.	The I/O ports or peripherals can be treated like memory locations and so all instructions related to memory can be used for data transfer between I/O device and the processor.	Only IN and OUT instructions can be used for data transfer between I/O device and the processor
4.	In memory mapped ports the data can be moved from any register to ports and vice-versa.	In I/O mapped ports the data transfer can take place only between the accumulator and ports.

42.List the types of data transfer in 8085.(Apr./Map.2016)

The data transfer technique refers to the method of data transfer between the processor and peripheral devices. In a typical microcomputer, data transfer takes place between any two devices:

- Microprocessor and memory
- Microprocessor and I/O devices
- Memory and I/O devices

43. How to reset the bit D_0 of accumulator without altering other bits?(Nov.2014)

To change a single bit of an output without changing any others bits

```
LXI H, CWORD : Load address of control word into HL
MOV A, M     : transfer copy of present control word to
               Accumulator
ANI FE h     : set bit 0
MOV M,A      : update copy of control word
OUT PORT0    : output new control word
```

44. What is mean by memory interfacing?(Nov.2015)

The primary function of memory interfacing is that the microprocessor should be able to read from and write into a given register of a memory chip. To perform these operations the microprocessor should

- Be able to select the chip
- Identify the register
- Enable the appropriate buffer

11 MARKS

1. **Explain in details about Memory**

INTRODUCTION TO MEMORY

A memory unit is an integral part of any microcomputer system and its primary purpose is to store programs and data. In a broad sense, a microcomputer memory system can be logically divided into three groups. They are:

- Processor memory
- Primary or main memory
- Secondary memory

The processor memory refers to registers inside the microprocessor. These registers are used to hold the data and results temporarily when a computation is in progress. Since the registers of the processor are fabricated using the same technology as that of the microprocessor, there is no speed disparity between these registers and the processor. However, the cost involved in this approach forces a manufacturer to include only few registers in the microprocessor.

The primary or main memory refers to the storage area which can be directly accessed by the microprocessor. Therefore, all programs and data must be stored only in the primary memory prior to execution. In primary memories, the access time should be compatible with the read/ write time of the processor. Therefore, only semiconductor memories are used as primary memories and they (the latest versions) are fabricated using CMOS technology. Primary memory normally includes ROM, EPROM, static RAM, DRAM and NVRAM.

Secondary memory refers to the storage medium comprising slow devices such as magnetic tapes and disks (hard disk, floppy disc and **C**ompact **D**isc (CD)). They are called auxiliary or backup storage. These devices are used to hold large data files and huge programs such as operating systems, compilers, data bases, permanent programs, etc. The microcomputer system copies the required programs and data from the secondary memory to the main memory and work directly with the main memory only.

SEMICONDUCTOR MEMORY

The main or primary memory elements are semiconductor devices, because the semiconductor devices alone can work at high speeds and consume less power. Moreover, they can be fabricated as ICs and so they occupy less space.

A typical semiconductor memory IC will have **n** address pins (lines) and **m** data pins (lines). The capacity of the memory will be $\mathbf{2^n \times m}$ bits. Figure 6.1 shows a simplified functional block diagram of semiconductor memory. The functional blocks of the semiconductor memory are row address decoder, column address decoder, memory array, input buffer and output buffer.

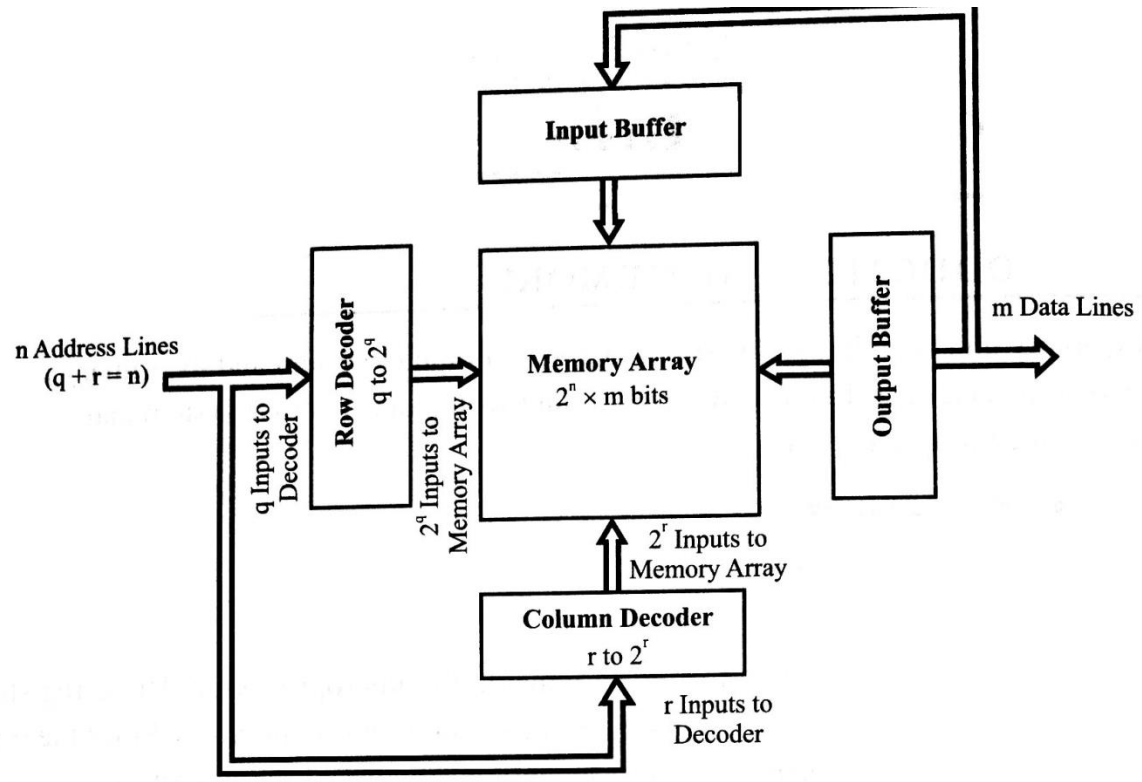


Fig. 6.1 : A simplified functional block diagram of a typical semiconductor memo

The input and output buffers are used to hold the data until valid time and also takes care of signal current level matching (Impedance matching). The **n** address lines are split into q lines and r lines, such that q + r = n. The q address lines are applied as input to row decoder and r address lines are applied as input to column decoder.

The output lines of the row and column decoder are used to form a matrix array of size, $2^q \times 2^r$ consisting of 2^n crossing points as shown in Fig. 6.2. Each crossing point is called a memory cell and can store one-bit of binary information. A typical memory array consists of **m** number of layers of matrix array as that of Fig. 6.2 and all of them are wired in parallel. When an address is send to memory IC, the row and column decoder will select one line each, which in turn will select one memory cell in each layer. Thus **m** memory cells are selected by an address. Then using the read or write control signals, the data can be read or stored in the selected memory cells.

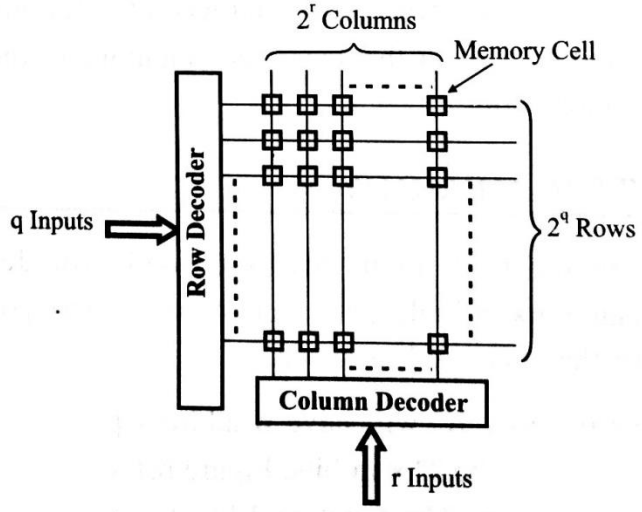


Fig. 6.2 : One layer of memory array

In the first version of semiconductor memory, the memory cells were made of passive elements like resistors and capacitors. Later, diodes were used instead of passive elements. With advancement in semiconductor technology, bipolar and MOS transistors were used to form memory cells. The latest technology used for fabricating memory cells are CMOS and HMOS which offers very low power and high speed operation.

The different types of semiconductor memory are ROM, PROM, EPROM, static RAM, DRAM and NVRAM. These semiconductor memories can be classified into volatile and non-volatile memory. If the information stored in a semiconductor memory is lost when the power supply to that IC is switched OFF, then the memory is called volatile. On the other hand, if the stored information is retained even if the power supply is switched OFF, then the memory is called non-volatile. ROM, PROM, EPROM and NVRAM are non-volatile memories. Static RAM and DRAM are volatile memories.

Semiconductor memories can also be classified into read only memory and read/write memory. In read only memories, the informations are stored permanently either during manufacturing or after manufacturing and then interfaced to the microcomputer system. The processor can only read the stored informations from these memories and cannot write into it. But in Read/Write memory, the processor can store (write) the information as well as read from it. ROM, PROM and EPROM are read only memories. NVRAM, static RAM and DRAM are read/write memories.

The other features of semiconductor memories are random access and non-destructive readout. In random access memory, the memory access time is independent of the memory location being accessed (i.e., the access time will be the same for first or last location). All semiconductor memories are random access memories. In semiconductor memories, a read operation by the processor will not destroy the stored information and for this reason the semiconductor memory is also called NDRO memory (**N**on-**D**estructive **R**ead-**O**ut memory).

ROM AND PROM

The ROM is a semiconductor memory which permits only a read access. The ROM functions as a memory array whose contents, once programmed, are permanently fixed and cannot be altered by the microprocessor to which the memory is interfaced. Other names for this type of memory are dead memory, fixed memory, permanent memory and **R**ead-**O**nly **S**tore (ROS). In ROM memory, the memory cell (or storage unit) will have a MOS transistor either with open gate or closed gate. The transistors with closed gate represent **1's** and with open gate represent **0's**. Since the configuration is fixed, they permanently store **1's** and **0's**.

ROM is a non-volatile memory, i.e., loss of power or system malfunction does not change the contents of the memory. Also, the ROM memories have the feature of random access, which means that the access time for a given memory location is same as that for all other locations. The process of storing information in ROM is called programming. The technique employed for storing information in ROM provides a convenient method for classifying ROMs into one of the following three categories:

1. **Custom programmed or Mask programmed ROM (ROM).**
2. **Programmable or Field programmable ROM (PROM).**
3. **Reprogrammable or Erasable-Programmable ROM (EPROM).**

The custom programmed ROMs are programmed by the manufacturer as specified by the user during fabrication and the contents cannot be changed after packaging. The programmable ROM's are one-time programmable by the user. The reprogrammable ROMs have facilities for programming as well as for erasing its content and reprogramming the memory. Reprogrammable ROMs are erased either by passing electrical current or ultraviolet light.

The programming of ROMs can be carried using a ROM (EPROM) programmer. Usually, the ROM programmer is a digital system interfaced to the **P**ersonal **C**omputer (PC). The information to be programmed is first stored as a file in the PC and converted to the required binary format using a conversion software. Then the information is transferred from the PC to the ROM programmer.

EPROM

The **R**ead **O**nly **M**emory (ROM) which has a reprogrammable feature is called EPROM (**E**rasable-**P**rogrammable **R**ead **O**nly **M**emory). The EPROM memory is non-volatile and also has the feature of random access. In an EPROM, the binary informations are entered using electrical impulses and the stored information is erased using ultraviolet rays. Typical erase time vary between 10 to 30 minutes.

In EPROM, the memory cell (storage location of a bit) consists of a MOS transistor with an isolated gate. The isolated gate is located between the normal control gate and the source/drain region of a MOS transistor. This gate may be charged with electrons during the programming operation and when charged with electrons, the transistor is permanently turned OFF. The state of the floating gate, charged or uncharged, is permanent because the gate is isolated in an extremely pure oxide.

The charge on the isolated gate may be removed if the device is irradiated with ultraviolet light. The ultraviolet light allows the electrons to recombine and discharge through the control gate. The process of charging and discharging are repeatable.

The EPROM is programmed by inserting the EPROM chip into the socket of a PROM programmer and providing addresses and voltage pulses at the appropriate pins of the chip. Usually, the PROM programmer is interfaced to a **P**ersonal **C**omputer (PC) and the informations to be programmed are downloaded from the PC.

EPROMs are manufactured by many semiconductor companies in the industry like INTEL, Hitachi, Toshiba, Cypress, etc. The manufacturers have a common industry standard, so that a product from different industry will be pin to pin compatible and slightly differ in electrical and switching characteristics. The various features of a 2764 (8 kb EPROM) manufactured by Cypress Semiconductor Corporation are discussed in this section.

2. Explain about interfacing Static RAM and EPROM and its Memory capacity in detail:

INTERFACING STATIC RAM AND EPROM

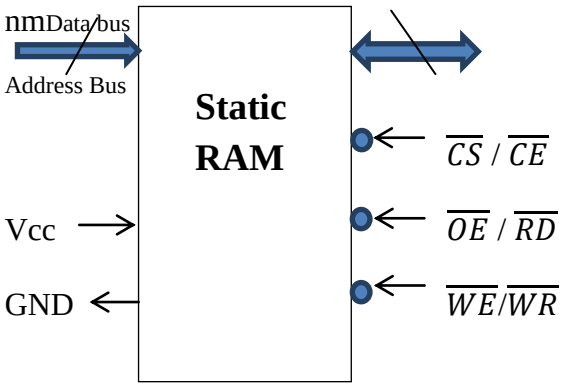
The primary function of memory interfacing is that the microprocessor should be able to read from and write into a set of semiconductor memory IC chips . Generally EPROM is interfaced for read operations and RAM is interfaced for read and write operations. The procedure for interfacing SRAM for Read/Write operation and EEPROM for read operation are similar.

Inorder to perform the read/write operation the memory access time should be less than the read/write time of processor, chip select signals should be generated for selecting a particular memory IC, suitable control signals has to be generated for read/write operation and a specific address should be allotted to each memory location.

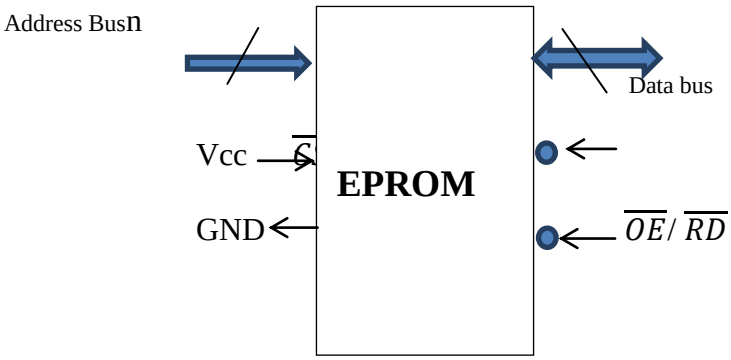
Hence memory interfacing deals with choosing memories with suitable access time. Designing address decoding circuit to generated chip select signals, generating control signals for read/write operation and allocation of addresses to various memory ICs and their locations.

Typical EPROM and Static RAM:

A typical semiconductor memory IC will have **n** address pins, **m** data pins (or output pins) and a minimum of two power supply pins (one for **Vcc** and other for **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 operations in EPROM are chip select (Chip enable), Read control (output enable) . A typical static RAM and EPROM are shown in figure.



A Typical static RAM IC



A typical EPROM IC in read mode

MEMORY CAPACITY:

A semiconductor memory IC will have **n** address pins and **m** data pins. Such a memory has $\mathbf{2^n}$ locations and each location can store m bit data. The size of data stored in each memory location is called memory word size of 1- byte are used.

The memory capacity is specified in kilo-bytes. If the memory IC has **m** data pins and **n** address pins,address pins, then the memory IC will have capacity of 2^n x m bits.When m=8 , the memory capacity is 2^n bytes. One kilobytes is 1024_{10} (=400_H) bytes. The relation between address pins and capacity of memory IC s are listed in table.

Relation between Number of address pins and memory capacity:

Number of address pins	Memory Capacity			Range of address in hexa
	In decimal	In kilo	In hexa	
10	2^{10} =1024	1k	400	000 to 3FF
11	2^{11} = 2x 2^{10}=2048	2k	800	000 to 7FF
12	2^{12} = 2^2x2^{10} = 4x2^{10}= 4096	4k	1000	000 to FFF
13	2^{13}= 2^3x2^{10} = 8x2^{10}=8192	8k	2000	0000 to 1FFF
14	2^{14}= 2^4x2^{10} = 16x2^{10}= 16384	16k	4000	0000 to 3FFF
15	2^{15}= 2^5x2^{10} = 32x2^{10}= 32768	32k	8000	0000 to 7FFF
16	2^{16}= 2^6x2^{10} = 64x2^{10}= 65536	64k	10000	0000 to FFFF
17	2^{17}= 2^7x2^{10} = 128x2^{10}=131072	128k	20000	00000 to 1FFFF
18	2^{18}= 2^8x2^{10} = 256x2^{10}=262144	256k	40000	00000 to 3FFFF
19	2^{19}= 2^9x2^{10} = 512x2^{10}= 524288	512k	80000	00000 to 7FFFF
20	2^{20}= 2^{10}x2^{10} = 1024x2^{10}=1048576	1024k=1M	100000	00000 to FFFFF

Choice of Memory ICs

The memory requirement of a system depends on the application for which it is designed. A system designer has a variety of choices for choosing memory ICs. The total memory requirement can be realized in a single IC or in multiple ICs.

The total memory requirement of the system will be split between EPROM and RAM memories. The EPROM memories are used for storing the monitor program, other permanent programs and data. The RAM memories are used for stack operations, temporary program and data storage.

The popular EPROM and static RAM ICs used with microprocessor and microcontroller systems and their capacity are listed here. Table-6.7 shows the number of address pins and data pins available on these ICs.

EPROM

2708 (1k × 8 = 8 kilo bits/1 kb)
2716 (2k × 8 = 16 kilo bits/2 kb)
2732 (4k × 8 = 32 kilo bits/4 kb)
2764 (8k × 8 = 64 kilo bits/8 kb)
27256 (32k × 8 = 256 kilo bits/32 kb)
27512 (64k × 8 = 512 kilo bits/64 kb)
27010 (128k × 8 = 1 Mega bits/128 kb)
27020 (256k × 8 = 2 Mega bits/256 kb)
27040 (512k × 8 = 4 Mega bits/512 kb)

Static RAM

6208 (1k × 8 = 8 kilo bits/1 kb)
6216 (2k × 8 = 16 kilo bits/2 kb)
6232 (4k × 8 = 32 kilo bits/4 kb)
6264 (8k × 8 = 64 kilo bits/8 kb)
62256 (32k × 8 = 256 kilo bits/32 kb)
62512 (64k × 8 = 512 kilo bits/64 kb)
62128 (128k × 8 = 1 Mega bits/128 kb)
62138 (256k × 8 = 2 Mega bits/256 kb)
62148 (512k × 8 = 4 Mega bits/512 kb)

Note : In this book kb refers to kilo bytes.

NUMBER OF ADDRESS AND DATA PINS IN MEMORY ICs

Memory IC EPROM/RAM	Capacity	Number of address pins	Number of data pins
2708/6208	1 kb	10	8
2716/6216	2 kb	11	8
2732/6232	4 kb	12	8
2764/6264	8 kb	13	8
27256/62256	32 kb	15	8
27512/62512	64 kb	16	8

Note : 16kb memory is not available as a standard product.

1. **Explain in details about Memory organisation in a Microprocessor system**

MEMORY ORGANIZATION IN A MICROPROCESSOR SYSTEM

A microprocessor-based system requires both EPROM and RAM. Hence the available memory space has to be divided between EPROM and RAM. This choice depends on the system designer as well as on the application for which the system is designed. In 16-bit system when memory is organized as two banks, for proper system functioning, the system designer should allot equal address space in the odd and even banks for both EPROM and RAM.

Some systems may require a large memory space and so full memory space is utilized. But in some systems, the memory requirement may be less and in this case the full memory space is not utilized. When full memory space is not utilized for memory, then the unused memory addresses can be used for addressing IO devices. Such IO devices are called memory-mapped IO devices and they can be accessed similar to that of a memory device.

The required EPROM memory capacity of the system can be implemented in one IC in an 8-bit system and two ICs in a 16-bit system (one for even and the other for odd bank) or in multiple ICs. Similarly, the RAM capacity of the system can be implemented in one IC in an 8-bit system and two ICs in a 16-bit system or in mulitple ICs. This choice depends on the availability of memory IC and the system designer. Some examples of memory organizations for 8085, 8086 and 8031/8051 processor-based system are discussed in the following section.

Memory Organization in an 8085-Based System

The 8085 microprocessor-based system requires both EPROM and RAM. Hence, the available memory space has to be divided between EPROM and RAM. The 8085 has 64 kb of addressable memory space and allotting this address space for EPROM and RAM depends on the system designer as well as on the application for which the system is designed.

In an 8085 system, the EPROM is mapped at the beginning of memory space. (i.e., 0000_H address is allotted to the EPROM memory location). Whenever the power supply is switched ON, the microprocessor chip will be resetted. This power-on reset will be implemented by the system designer. When the processor is resetted all the internal registers, the flag register and the program counter will be cleared. Hence after a reset, the program counter will have an address 0000_H and so the processor starts fetching and executing the instruction stored at 0000_H.

The system designer will store the monitor program starting from the address 0000_H. The monitor program should be executed to initialize the system peripherals, whenever the system is switched ON. To enable automatic execution of the monitor program, whenever the system is switched ON, the EPROM should be mapped from the 0000_H location in an 8085-based system. The monitor program is a permanent program written by the system designer to take care of system initialization. System initialization includes the following :

1. **Programming the 8279 for keyboard scanning and display refreshing.**
2. **Programming peripheral ICs 8259, 8257, 8255, 8251, 8254, etc.**
3. **Initializing stack.**
4. **Display a message on the LED's.**
5. **Initializing the interrupt vector table.**

Note : *8279 - Programmable keyboard/display controller.* *8257 - DMA controller.*
8259 - Programmable interrupt controller. *8251 - USART.*
8255 - Programmable peripheral interface. *8254 - Programmable timer.*

Consider a system in which the full memory space 64 kb is utilized for the EPROM memory. In this system, the entire 16 address lines of the processor are connected to the address input pins of the memory IC in order to address the internal locations of the memory and the Chip Select ($\overline{CS}$) pin of the EPROM is permanently tied to logic **low** (i.e., tied to ground) as shown in Fig. 6.15. Now the range of address for EPROM is 0000_H to $FFFF_H$.

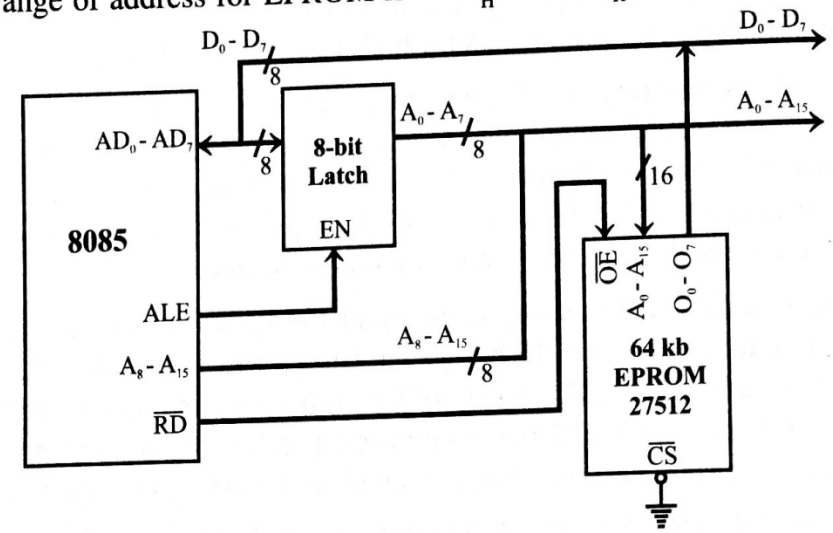


Fig. 6.15 : An example of implementing 64 kb EPROM in the 8085 system.

Consider a system in which the available 64kb memory space is equally divided between the EPROM and the RAM. Let us implement 32 kb memory capacity of EPROM using a single IC 27256. Similarly, 32 kb RAM capacity is implemented using a single IC 62256. A 32 kb memory requires 15 address lines and so the address lines A_0 - A_{14} of the processor are connected to 15 address pins of both EPROM and RAM, as shown in Fig. 6.16. The unused addres line A_{15} is used as chip select signal for selecting either EPROM or RAM. The A_{15} is directly connected to the $\overline{CS}$ pin of the EPROM and it is inverted and connected to the $\overline{CS}$ pin of the RAM. Therefore, the EPROM is selected when A_{15} = 0 and RAM is selected when A_{15} = 1. The address range of EPROM will be 0000_H to $7FFF_H$ and that of RAM will be 8000_H to $FFFF_H$.

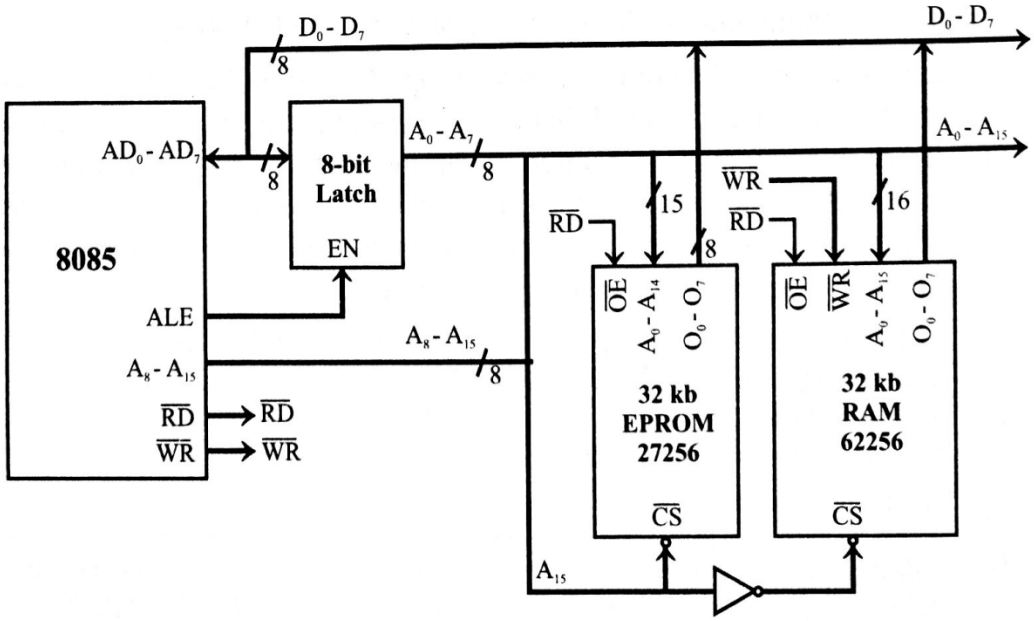


Fig. 6.16 : An example of implementing 32 kb EPROM and 32 kb RAM in an 8085 system.

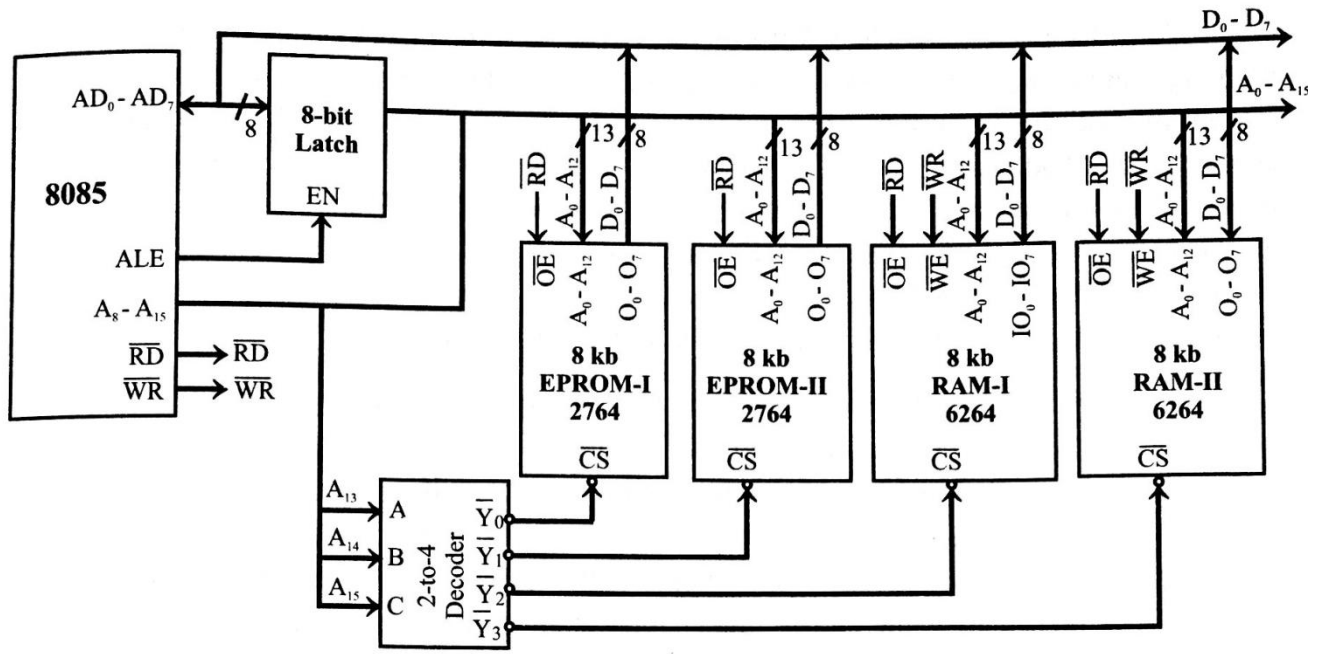


Fig. 6.17 : An example of implementing 16 kb EPROM and 16 kb RAM in an 8085 system.

TABLE - 6.11 : ADDRESS ALLOCATION FOR MEMORY ICs SHOWN IN FIG. 6.17

Device	Binary address																Hexa address
	Decoder enable/input			Input to address pins of memory IC													
	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
8 kb EPROM - I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0001
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFF
8 kb EPROM - II	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000
	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	2001
	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	2002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFF
8 kb RAM - I	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	4001
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	4002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	5FFF
8 kb RAM - II	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	6000
	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	6001
	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	6002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7FFF

Consider a system in which 32 kb memory space is implemented using four numbers of 8 kb memory. Let two numbers of 8 kb memory be EPROM and the remaining two numbers be RAM. Each 8 kb memory requires 13 address lines and so the address lines A_0 - A_{12} of the processor are connected to 13 address pins of all the memory ICs. The address lines A_{13} and A_{14} can be decoded using a 2-to-4 decoder to generate four chip select signals. These four chip select signals can be used to select one of the four memory IC at any one time. The address line A_{15} is used as enable for the decoder. The simplified schematic of this memory organization is shown in Fig. 6.17 and the addresses allotted to each memory IC are shown in Table-6.11.

Consider a system in which the 64 kb memory space is implemented using eight numbers of 8 kb memory. Each 8 kb memory requires 13 address lines and so the address lines A_0-A_{12} of the processor are connected to 13 address pins of all the memory ICs. The address lines A_{13}, A_{14} and A_{15} are decoded using a 3-to-8 decoder to generate eight chip select signals. These eight chip select signals can be used to select one of the eight memory ICs at any one time. The design example-85.2 presented at the end of this chapter is an example of implementing 64 kb address space using 8 numbers of 8 kb memory.

4. Explain in details about I/O structure of microprocessor

IO STRUCTURE OF A TYPICAL MICROCOMPUTER

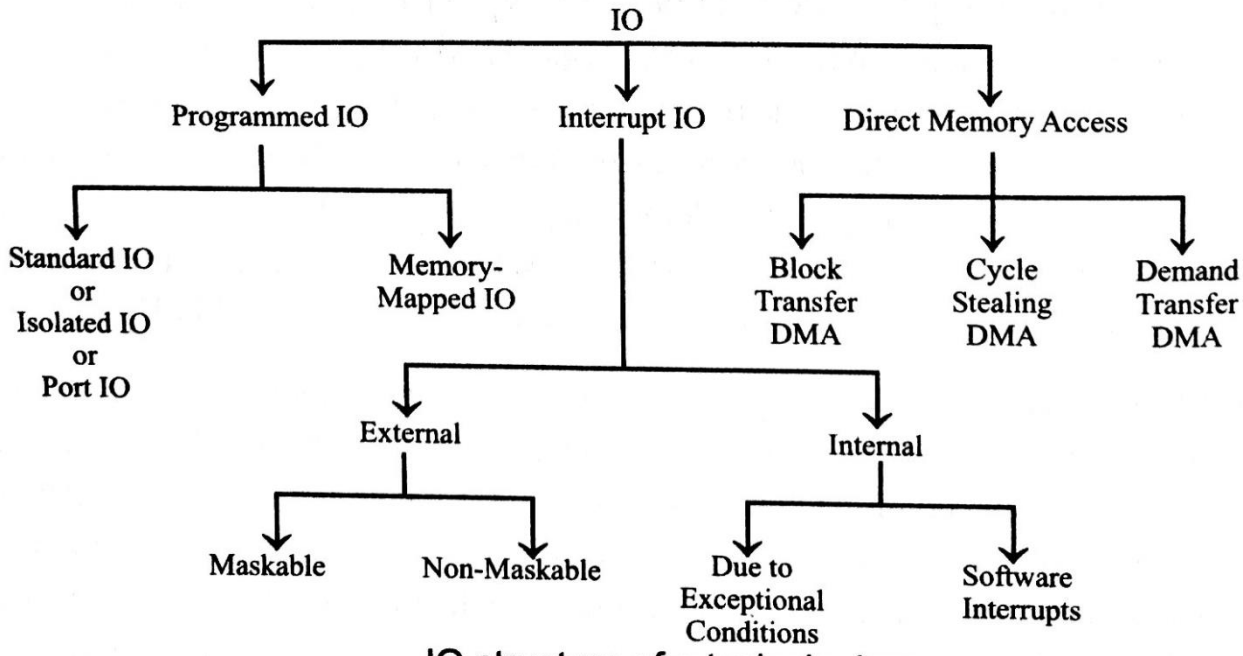
The IO devices connected to a microcomputer system provides an efficient means of communication between the microcomputer system and the outside world. These IO devices are commonly called peripherals and include keyboards, CRT displays, printers and disks (floppy disk, hard disk and Compact Disc (CD)).

The characteristics of the IO devices are normally different from the characteristics of the microprocessor. Since the characteristics of the IO devices are not compatible with that of the microprocessor, an interface hardware circuitry between the microprocessor and the IO device is necessary.

There are three major types of data transfer between the microcomputer and an IO device. They are :

- **Programmed IO**
- **Interrrupt driven IO**
- **Direct memory access (DMA)**

In programmed IO, the data transfer is accomplished through an IO port and controlled by a software. In interrupt driven IO, the IO device will interrupt the processor, and initiate data transfer. In DMA, the data transfer between the memory and IO can be performed by bypassing the microprocessor. Each type of data transfer scheme mentioned above, includes different methods of data transfer schemes. Figure 6.20 shows all the types of data transfer schemes in a microcomputer and it can also be called the IO structure of a microcomputer.



IO structure of a typical microcomputer.

INTERFACING IO AND PERIPHERAL DEVICES

IO devices are generally slow devices and so they are connected to the system bus through ports. The ports are buffer ICs which are used to temporarily hold the data transmitted from the microprocessor to the IO device or to hold the data transmitted from the IO device to the microprocessor.

For data transfer from the input device to the processor, the following operations are performed:

1. The input device will load the data to the port.
2. When the port receives a data, it sends a message to the processor to read the data.
3. The processor will read the data from the port.
4. After a data has been read by the processor, the input device will load the next data into the port.

For data transfer from processor to output device, the following operations are performed:

1. The processor will load the data to the port.
2. The port will send a message to the output device to read the data.
3. The output device will read the data from the port.
4. After the data has been read by the output device, the processor can load the next data to the port.

IO Mapping

The port and peripheral devices will have one logic **low/high** chip select pin. The processor can access the port/peripheral device by supplying the internal address and chip select signal. Therefore, the port and peripheral device interfacing (IO interfacing) deals with the allocation of various internal addresses and generation of chip select signals.

There are two ways of interfacing IO devices in a microprocessor/microcontroller-based system:

1. Memory-mapped IO device.
2. Standard IO-mapped IO device or Isolated IO mapping.

The interfacing of IO ports and controller/peripheral ICs are commonly referred as IO device mapping.

The 8085 and 8086 microprocessors supports both memory-mapped IO and IO-mapped IO. The 8031/8051 microcontroller supports only memory-mapped IO. Hence in a 8031/8051-based system, some of the memory addresses should be reserved for IO devices, and in these systems the IO devices are interfaced similar to that of memory devices.

IO mapping in 8085-based system

The two methods of interfacing IO devices in an 8085-based system are: Memory mapping of IO device and Standard IO mapping of IO device or Isolated IO mapping.

In memory mapping of IO devices the ports are allotted a 16-bit address like that of memory location. Some of the chip-select signals generated to select memory ICs are used for selecting the IO port devices. Hence, the processor treats the IO ports as memory locations for reading and writing (i.e., the devices which are mapped by memory mapping are accessed by executing the memory read cycle or the memory write cycle.)

In standard IO mapping or isolated IO mapping, a separate 8-bit address is allotted for IO ports and the peripheral ICs. The processor differentiates the IO-mapped devices from the memory-mapped devices in the following ways:

1. For accessing the IO-mapped devices, the processor executes the IO read or write cycle.
2. During IO read or write cycle, the 8-bit address is placed on both low order address lines and the high order address lines.
3. IO/$\overline{M}$ is asserted high to indicate the IO operation (for read as well as write).

A 8085 processor does not provide separate read ($\overline{RD}$) and write ($\overline{WR}$)signals for memory and IO devices. But it differentiates the memory and IO device accessed by an IO/$\overline{M}$ signal. The three signals $\overline{RD}$,$\overline{WR}$ and IO/$\overline{M}$ can be decoded as shown in Fig. 6.21 to provide separate read and write control signals for IO devices and memory devices.

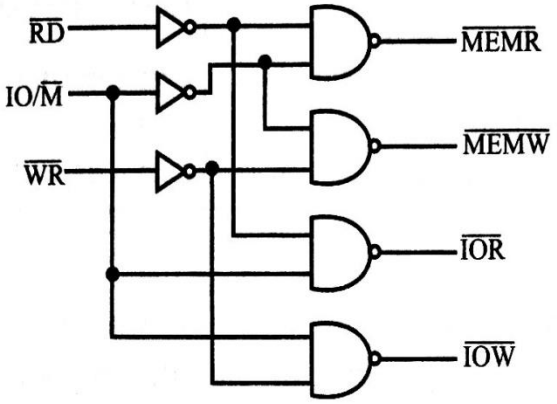


Fig. 6.21 : Circuit to generate separate read and write signals for memory and IO devices in an 8085 system.

When the devices are IO-mapped, then only the IN and OUT instructions have to be used for data transfer between the device and the processor. For IO-mapped devices, a separate decoder should be used to generate the required chip select signals.

5. Explain the difference between memory mapped I/o and standard IO mapped IO (11m-Dec-2014) or Differentiate between I/O mapped I/O and memory mapped I/o. (4m-April-2015)

S.No	Memory mapping of I/O device	I/O mapping of I/O device
1.	16-bit addresses are provided for I/O devices.	8-bit addresses are provided for I/O devices.
2.	The devices are accessed by memory read or memory write cycles.	The devices are accessed by I/O read or I/O write cycle. During these cycles the 8-bit address is available on both low order address lines and high order address lines.
3.	The I/O ports or peripherals can be treated like memory locations and so all instructions related to memory can be used for data transfer between I/O device and the processor.	Only IN and OUT instructions can be used for data transfer between I/O device and the processor
4.	In memory mapped ports the data can be moved from any register to ports and vice-versa.	In I/O mapped ports the data transfer can take place only between the accumulator and ports.

5.	When memory mapping is used for I/O devices, the full memory address space cannot be used for addressing memory. Hence memory mapping is useful only for small systems, where the memory requirement is less.	When I/O mapping is used for I/O devices then the full memory address space can be used for addressing memory. Hence it is suitable for systems which requires large memory capacity.
6.	In memory mapped I/O devices , a large number of I/O ports can be interfaced.	In I/O mapping only 256 ports (2^8 =256) can be interfaced.
7.	For accessing the memory mapped devices, the processor executes memory read or write cycle. During this cycle IO/$\overline{M}$ is asserted low (IO/$\overline{M}$ =0)	For accessing the I/O mapped devices, the processor executes I/O read or write cycle. During this cycle IO/$\overline{M}$ is asserted High (IO/$\overline{M}$ =1)

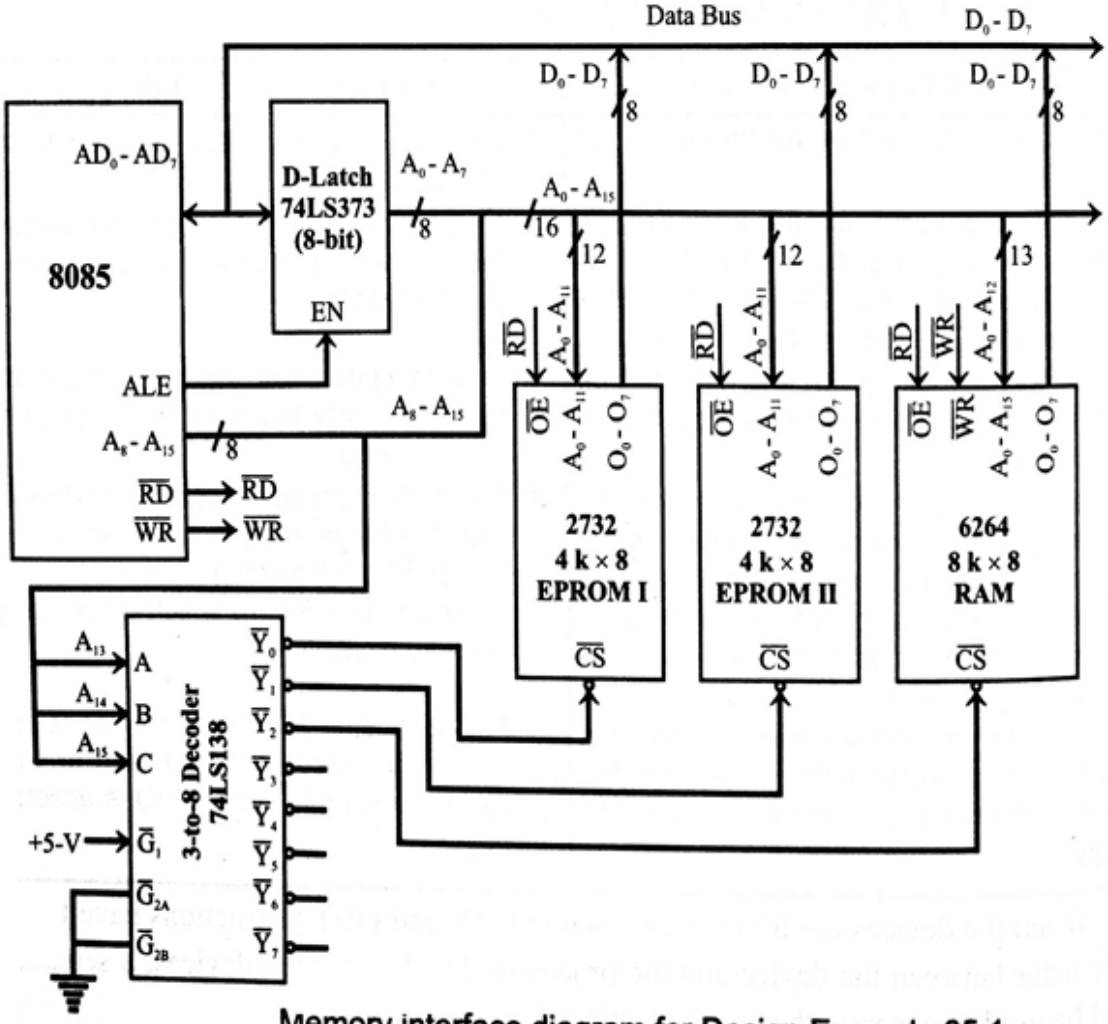
Example of Memory and IO interface in 8085 – Base System

6.1 Draw the memory interface diagram to 8085 processor with 2 numbers of 4 Kb EPROM and one number of 8Kb RAM. Explain the system and allocate binary addresses to memory IC's.(11m-Dec-2014)

Interface two numbers of 4 kb EPROM and one number of 8 kb RAM with 8085 processor. Explain the interface diagram and allocate binary addresses to the memory ICs.

Solution

The IC 2732 is selected for EPROM memory and the IC 6264 is selected for RAM memory. Both the memory ICs have time compatibility with an 8085 processor.



Memory interface diagram for Design Example 85.1.

The 4 kb EPROM IC requires 12 address lines (2^{12} = 4 k). The 8 kb RAM IC requires 13 address lines (2^{13} = 8 k). The address lines A_0 - A_{11} are connected to both EPROM and RAM address input pins. The address lines A_{13}, A_{14} and A_{15} are not used for memory address. Hence by decoding these address lines, we can generate chip select signals.

The 3-to-8 decoder, 74LS138 is employed to produce the chip-select signals for the system. The decoder has 8-output lines which can be used as 8-chip select signals. In this, three chip select signals are used for selecting the memory ICs and the remaining five can be used for selecting other peripheral ICs in the system or for future expansion of the memory capacity. The interface diagram is shown in Fig. 85.1. The address allotted to the memory ICs are shown in Table-85.1.

TABLE - 85.1 : ADDRESS ALLOCATION TABLE FOR DESIGN EXAMPLE 85.1

Memory IC	Binary address																Hexa address
	Decoder input			Input to memory address pins													
	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
EPROM I 2732	0	0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0000
	0	0	0	X	0	0	0	0	0	0	0	0	0	0	0	1	0001
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	0	0	X	1	1	1	1	1	1	1	1	1	1	1	1	0FFF
EPROM II 2732	0	0	1	X	0	0	0	0	0	0	0	0	0	0	0	0	2000
	0	0	1	X	0	0	0	0	0	0	0	0	0	0	0	1	2001
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	0	1	X	1	1	1	1	1	1	1	1	1	1	1	1	2FFF
RAM 6264	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	4001
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	5FFF

Note : "X" indicates the unused address line for the particular memory IC and they are considered as zero.

The EPROMs are mapped in the beginning of memory space. The remaining addresses can be allotted to the RAMs. The EPROM memory is mapped from 0000_H to $0FFF_H$ and 2000_H to $2FFF_H$. The RAM memory is mapped from 4000_H to $5FFF_H$.

DESIGN EXAMPLE 6.2

Interface three numbers of 8 kb EPROM and 5 numbers of 8 kb static RAM to microprocessor 8085 to have a total memory capacity of 64 kb.

Solution

The IC 2764 is selected for EPROM memory and the IC 6264 is selected for RAM memory. Both the memory ICs have time compatibility with an 8085 processor.

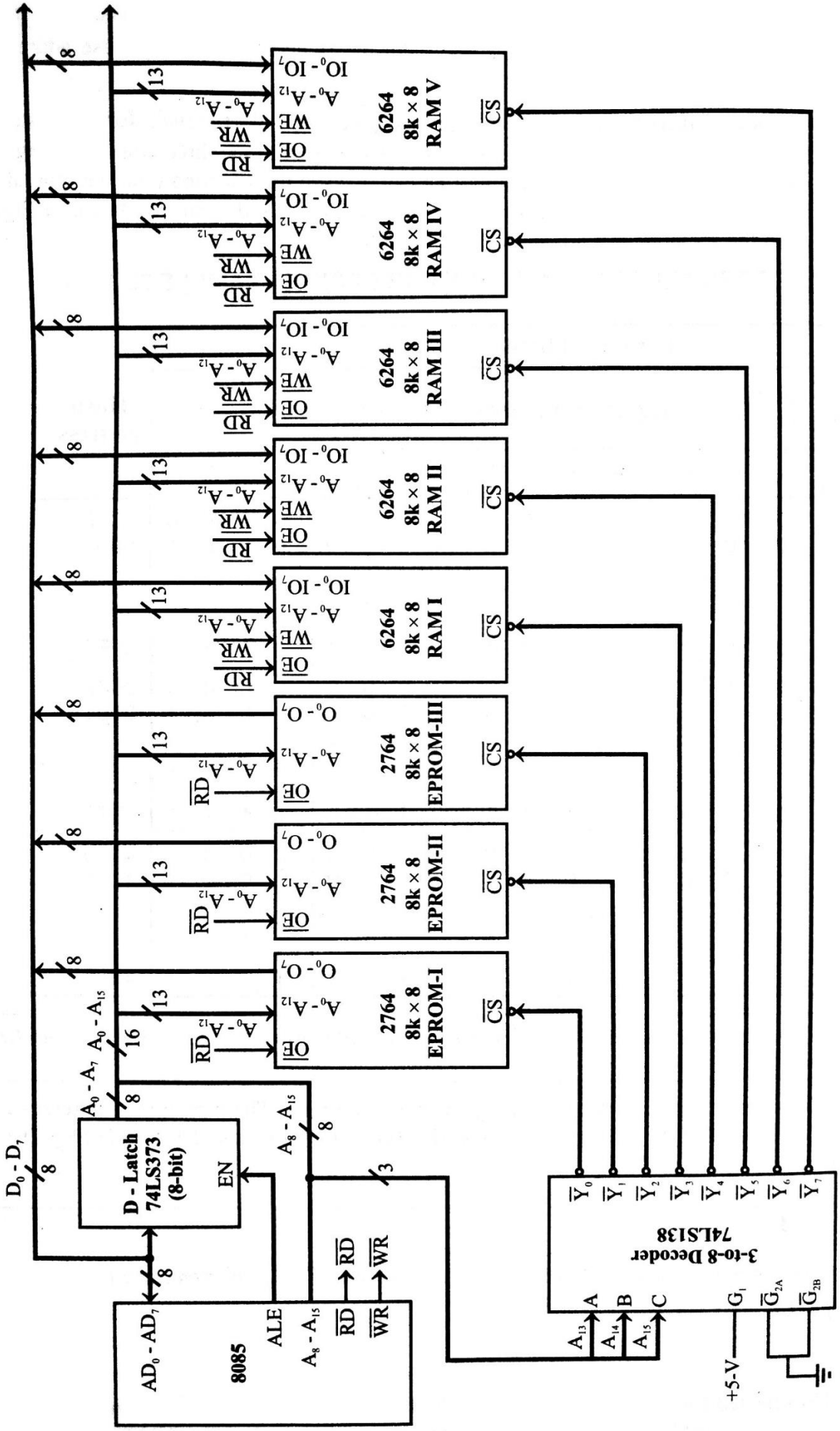


Fig. 85.2 : Memory interface diagram for Design Example 85.2.

TABLE - 85.2 : ADDRESS ALLOCATION TABLE FOR DESIGN EXAMPLE-85.2

Memory IC chip	Binary address															Hexa address
	Decoder input			Input to memory address pins												
	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
EPROM I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0001
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFF
EPROM II	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000
	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	2001
	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	2002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFF
EPROM III	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	4001
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	4002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	5FFF
RAM I	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	6000
	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	6001
	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	6002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7FFF
RAM II	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8000
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	8001
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	8002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	9FFF
RAM III	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	A000
	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	A001
	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	A002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	BFFF
RAM IV	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	C000
	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	C001
	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	C002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	DFFF
RAM V	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	E000
	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	E001
	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	E002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFF

The 8 kb EPROM IC requires 13 address lines (2^{13} = 8 k). The 8 kb RAM IC requires 13 address lines (2^{13} = 8 k). The address lines A_0 - A_{12} are connected to all the EPROMs and RAMs. Hence, A_0 - A_{12} will select the required memory location.The address lines A_{13}, A_{14} and A_{15} are not used for memory address. Hence by decoding these address lines we can generate chip select signals.

The 3-to-8 decoder, 74LS138 is employed to produce the chip-select signals for the system. The decoder has 8-output lines which can be used as 8-chip select signals. All the 8-chip select signals are used to select memory ICs. EPROM's are mapped at the beginning of memory space. The decoder will select a memory IC by decoding the address lines A_{13}, A_{14} and A_{15}. The address lines A_0 - A_{12} will select a particular memory location in the selected IC. The interface diagram is shown in Fig. 85.2 and address allocation table is shown in Table-85.2.

In this system the full memory capacity of 64 kb is utilized for memory. Hence the peripheral ICs and the IO ports should be IO-mapped in the system. The EPROM is mapped from 0000_H to $5FFF_H$. The RAM is mapped from 6000_H to $FFFF_H$. The EPROM capacity is 24 kb. The RAM capacity is 40 kb.

DESIGN EXAMPLE 6.3

Interface 2 kb RAM and 256 × 8 ROM with an 8085 processor to satisfy the total memory requirement of 8 kb RAM and 1 kb ROM.

Solution

The memory requirement of 8 kb RAM can be achieved with 4 numbers of 2 kb RAM. The memory requirement of 1 kb ROM can be achieved with 4 numbers of 256 × 8 ROM. (4 × 256 = 1024 = 1k). The 2 kb RAM requires 11 address lines (2^{11} = 2 k). The 256 × 8 ROM requires 8 address lines (2^8 = 256). The address lines A_0 - A_{10} are connected to the RAM ICs. Hence, they will select the required memory location in that IC. The address lines A_0 - A_7 are connected to the ROM ICs. Hence, they will select the required memory location in that IC.

Totally there are 8-memory ICs, hence we require 8-chip select signals. The 8-chip select signals can be generated by using a dual 2-to-4 decoder 74LS139. One of the 2-to-4 decoder is used to generate the chip select signals for ROM memory ICs and the other decoder is used to generate chip select signals for RAM memory ICs. The address lines A_8 and A_9 are used to generate chip select signals for ROM memory. The address lines A_{10} to A_{15} are logically ORed and used as enable for ROM decoder. The address lines A_{11} and A_{12} are used to generate chip select signals for RAM memory. The address lines A_{13} to A_{15} are logically NANDed and used as enable for RAM decoder.

The ROM memories are mapped in the beginning of memory space. The RAM memories are mapped at the end of memory space. The ROM memories are mapped from 0000_H to $03FF_H$. The RAM memories are mapped from $E000_H$ to $FFFF_H$.

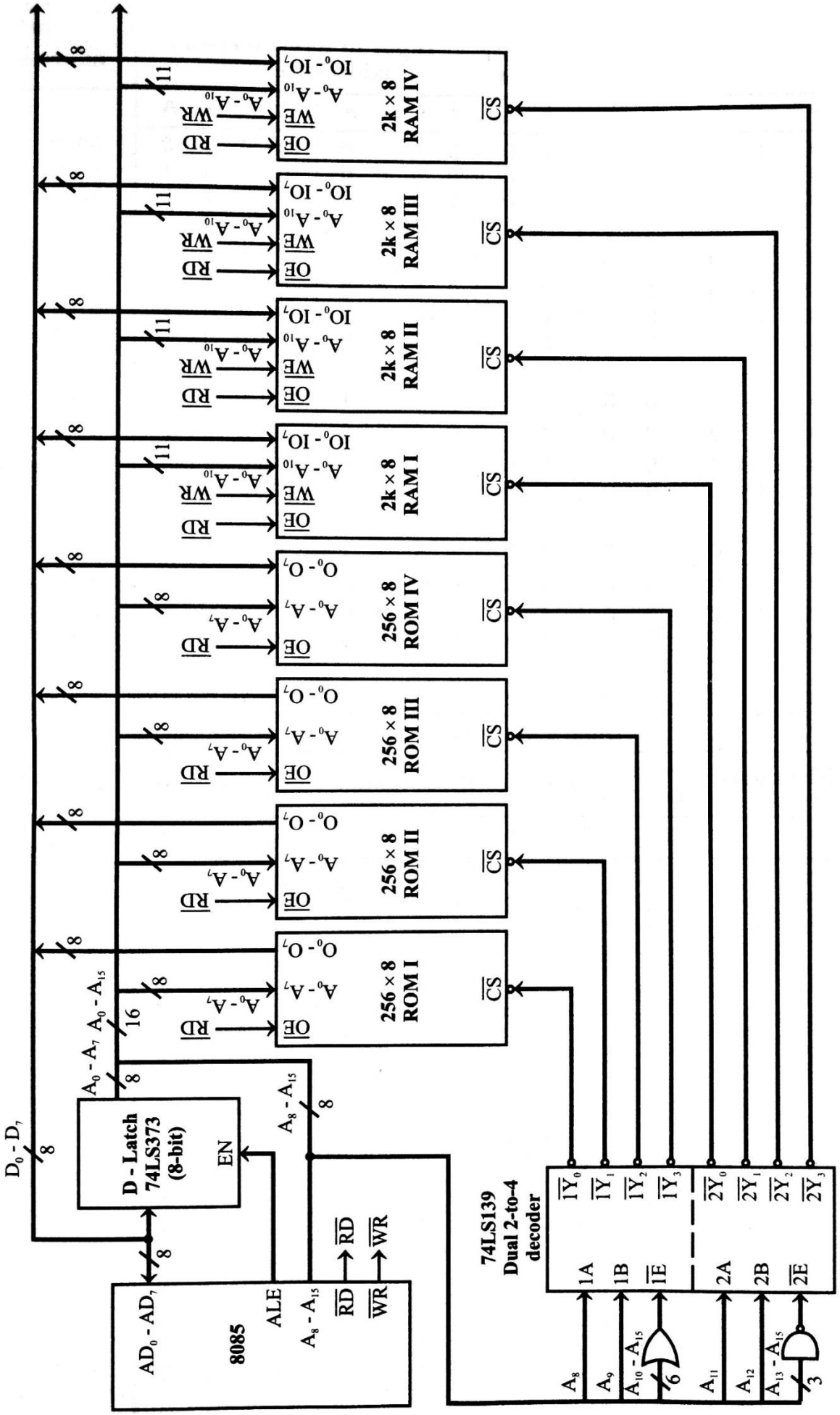


Fig. 85.3 : Memory interface diagram for Design Example 85.3.

TABLE - 85.3 : ADDRESS ALLOCATION TABLE FOR DESIGN EXAMPLE-85.3

Device	Binary address: ROM decoder enable A_{15} A_{14} A_{13} A_{12}	A_{11} A_{10}	Decoder input A_9 A_8	Input to ROM memory address pins A_7 A_6 A_5 A_4	A_3 A_2 A_1 A_0	Hexa address
256 × 8 ROM I	0 0 0 0	0 0	0 0	0 0 0 0	0 0 0 0	0000
	0 0 0 0	0 0	0 0	0 0 0 0	0 0 0 1	0001
	0 0 0 0	0 0	0 0	0 0 0 0	0 0 1 0	0002
	. . . .	. .	. .	. . . .	. . . .	.
	0 0 0 0	0 0	0 0	1 1 1 1	1 1 1 1	00FF
256 × 8 ROM II	0 0 0 0	0 0	0 1	0 0 0 0	0 0 0 0	0100
	0 0 0 0	0 0	0 1	0 0 0 0	0 0 0 1	0101
	0 0 0 0	0 0	0 1	0 0 1 0	0 0 1 0	0102
	. . . .	. .	. .	. . . .	. . . .	.
	0 0 0 0	0 0	0 1	1 1 1 1	1 1 1 1	01FF
256 × 8 ROM III	0 0 0 0	0 0	1 0	0 0 0 0	0 0 0 0	0200
	0 0 0 0	0 0	1 0	0 0 0 0	0 0 0 1	0201
	0 0 0 0	0 0	1 0	0 0 0 0	0 0 1 0	0202
	. . . .	. .	. .	. . . .	. . . .	.
	0 0 0 0	0 0	1 0	1 1 1 1	1 1 1 1	02FF
256 × 8 ROM IV	0 0 0 0	0 0	1 1	0 0 0 0	0 0 0 0	0300
	0 0 0 0	0 0	1 1	0 0 0 0	0 0 0 1	0301
	0 0 0 0	0 0	1 1	0 0 0 0	0 0 1 0	0302
	. . . .	. .	. .	. . . .	. . . .	.
	0 0 0 0	0 0	1 1	1 1 1 1	1 1 1 1	03FF

Device	Binary address: RAM decoder enable A_{15} A_{14} A_{13}	Decoder input A_{12} A_{11}	Input to RAM memory address pins A_{10} A_9 A_8	A_7 A_6 A_5 A_4	A_3 A_2 A_1 A_0	Hexa address
2 k × 8 RAM I	1 1 1	0 0	0 0 0	0 0 0 0	0 0 0 0	E000
	1 1 1	0 0	0 0 0	0 0 0 0	0 0 0 1	E001
	1 1 1	0 0	0 0 0	0 0 0 0	0 0 1 0	E002
	. . .	. .	. . .	. . . .	. . . .	.
	1 1 1	0 0	1 1 1	1 1 1 1	1 1 1 1	E7FF
2 k × 8 RAM II	1 1 1	0 1	0 0 0	0 0 0 0	0 0 0 0	E800
	1 1 1	0 1	0 0 0	0 0 0 0	0 0 0 1	E801
	1 1 1	0 1	0 0 0	0 0 0 0	0 0 1 0	E802
	. . .	. .	. . .	. . . .	. . . .	.
	1 1 1	0 1	1 1 1	1 1 1 1	1 1 1 1	EFFF
2 k × 8 RAM III	1 1 1	1 0	0 0 0	0 0 0 0	0 0 0 0	F000
	1 1 1	1 0	0 0 0	0 0 0 0	0 0 0 1	F001
	1 1 1	1 0	0 0 0	0 0 0 0	0 0 1 0	F002
	. . .	. .	. . .	. . . .	. . . .	.
	1 1 1	1 0	1 1 1	1 1 1 1	1 1 1 1	F7FF
2 k × 8 RAM IV	1 1 1	1 1	0 0 0	0 0 0 0	0 0 0 0	F800
	1 1 1	1 1	0 0 0	0 0 0 0	0 0 0 1	F801
	1 1 1	1 1	0 0 0	0 0 0 0	0 0 1 0	F802
	. . .	. .	. . .	. . . .	. . . .	.
	1 1 1	1 1	1 1 1	1 1 1 1	1 1 1 1	FFFF

DESIGN EXAMPLE 6.4

In a microprocessor system using 8085, the memory requirement is 8 kb EPROM and 8 kb RAM. For interfacing IO devices, three numbers of 8255 are required. Select suitable memories and explain how they are interfaced to the system. Interface the 8255 by memory mapping.

Solution

IC 2764 is selected for EPROM memory and IC 6264 is selected for RAM memory. Both the memory ICs have time compatibility with an 8085 processor.

The 8 kb EPROM, 2764 requires 13 address lines (2^{13} = 8 k). The 8 kb RAM, 6264 requires 13 address lines (2^{13} = 8 k). The address lines A_0 to A_{12} are connected to both EPROM and RAM memory ICs. The 8255 requires four internal addresses. Let us connect A_1 of 8085 to A_0 of 8255 and A_2 of 8085 to A_1 of 8255. The 8255 is memory-mapped in the system.

> ***Note :*** *The internal devices of an 8255 can be selected by connecting any two address lines of the processor to A_0 and A_1 of the 8255.*

For the memories and 8255's we require 5 chip select signals. Hence, we can use a 3-to-8 decoder 74LS138 for generating eight chip select signals by decoding the unused address lines A_{13}, A_{14} and A_{15}. The decoder enable pins are permanently tied to appropriate levels. In the eight chip select signals, five are used for selecting the memory ICs and 8255, and the remaining three can be used for future expansion. The memory/8255 interface diagram is shown in Fig. 85.4.

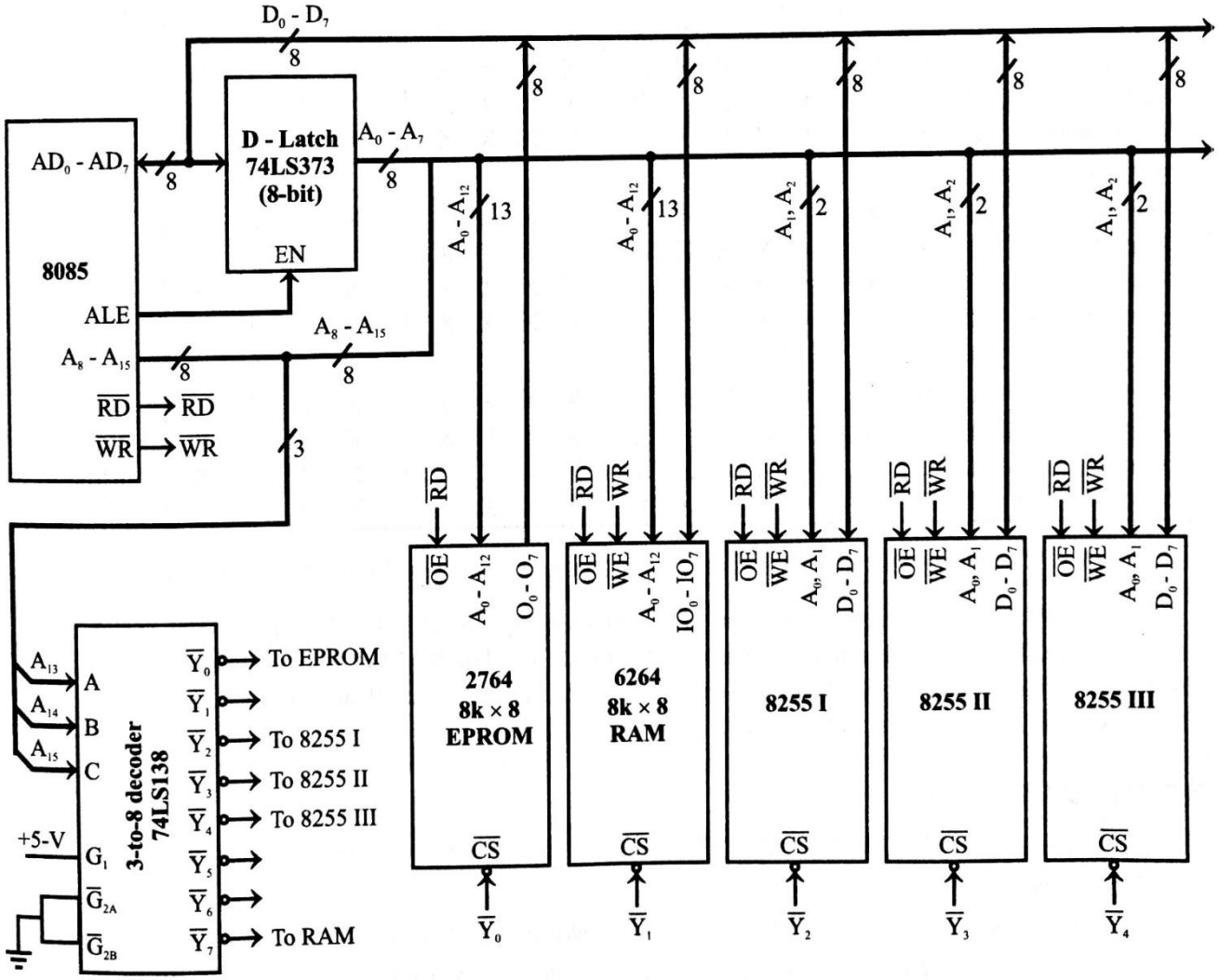


Fig. 85.4 : Memory interface diagram for Design Example 85.4.

TABLE - 85.4 : ADDRESS ALLOCATION TABLE FOR DESIGN EXAMPLE - 85.4

Device	Binary address															Hexa address
	Decoder input			Input to address pins of memory/8255												
	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
2764 EPROM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0001
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFF
6264 RAM	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	E000
	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	E001
	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	E002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFF
8255 I																	
Port-A	0	1	0	X	X	X	X	X	X	X	X	X	X	0	0	X	4000
Port-B	0	1	0	X	X	X	X	X	X	X	X	X	X	0	1	X	4002
Port-C	0	1	0	X	X	X	X	X	X	X	X	X	X	1	0	X	4004
Control register	0	1	0	X	X	X	X	X	X	X	X	X	X	1	1	X	4006
8255 II																	
Port-A	0	1	1	X	X	X	X	X	X	X	X	X	X	0	0	X	6000
Port-B	0	1	1	X	X	X	X	X	X	X	X	X	X	0	1	X	6002
Port-C	0	1	1	X	X	X	X	X	X	X	X	X	X	1	0	X	6004
Control register	0	1	1	X	X	X	X	X	X	X	X	X	X	1	1	X	6006
8255 III																	
Port-A	1	0	0	X	X	X	X	X	X	X	X	X	X	0	0	X	8000
Port-B	1	0	0	X	X	X	X	X	X	X	X	X	X	0	1	X	8002
Port-C	1	0	0	X	X	X	X	X	X	X	X	X	X	1	0	X	8004
Control register	1	0	0	X	X	X	X	X	X	X	X	X	X	1	1	X	8006

Note : "X" indicates that the address line is not used for the particular device and they are considered as zero.

The EPROM is mapped at the start of the memory space. The RAM is mapped at the end of the memory space. The EPROM is mapped from 0000_H to $1FFF_H$. The RAM is mapped from $E000_H$ to $FFFF_H$. The four internal devices of 8255 are the control register, port-A, port-B and port-C. A 16-bit address is allotted to each internal device of the 8255 as shown in Table-85.4.

DESIGN EXAMPLE 6.5

A system requires 1 kb EPROM and 16 kb RAM. Also the system has two numbers of 8255, one number of 8279, one number of 8251 and one number of 8254.

(8255 - Programmable peripheral interface, 8279-Keyboard/display controller, 8251 - USART and 8254 - Timer)

Draw the Interface diagram. Allocate addresses to all the devices. The peripheral IC's should be IO-mapped.

Solution

The IO devices in the system should be mapped by standard IO mapping. Hence, separate decoders can be used to generate chip select signals for memory ICs and peripheral ICs.

For 16 kb EPROM, we can provide two numbers of 2764 (8 k × 8) EPROM. For 16 kb RAM we can provide two numbers of 6264 (8 k × 8) RAM.

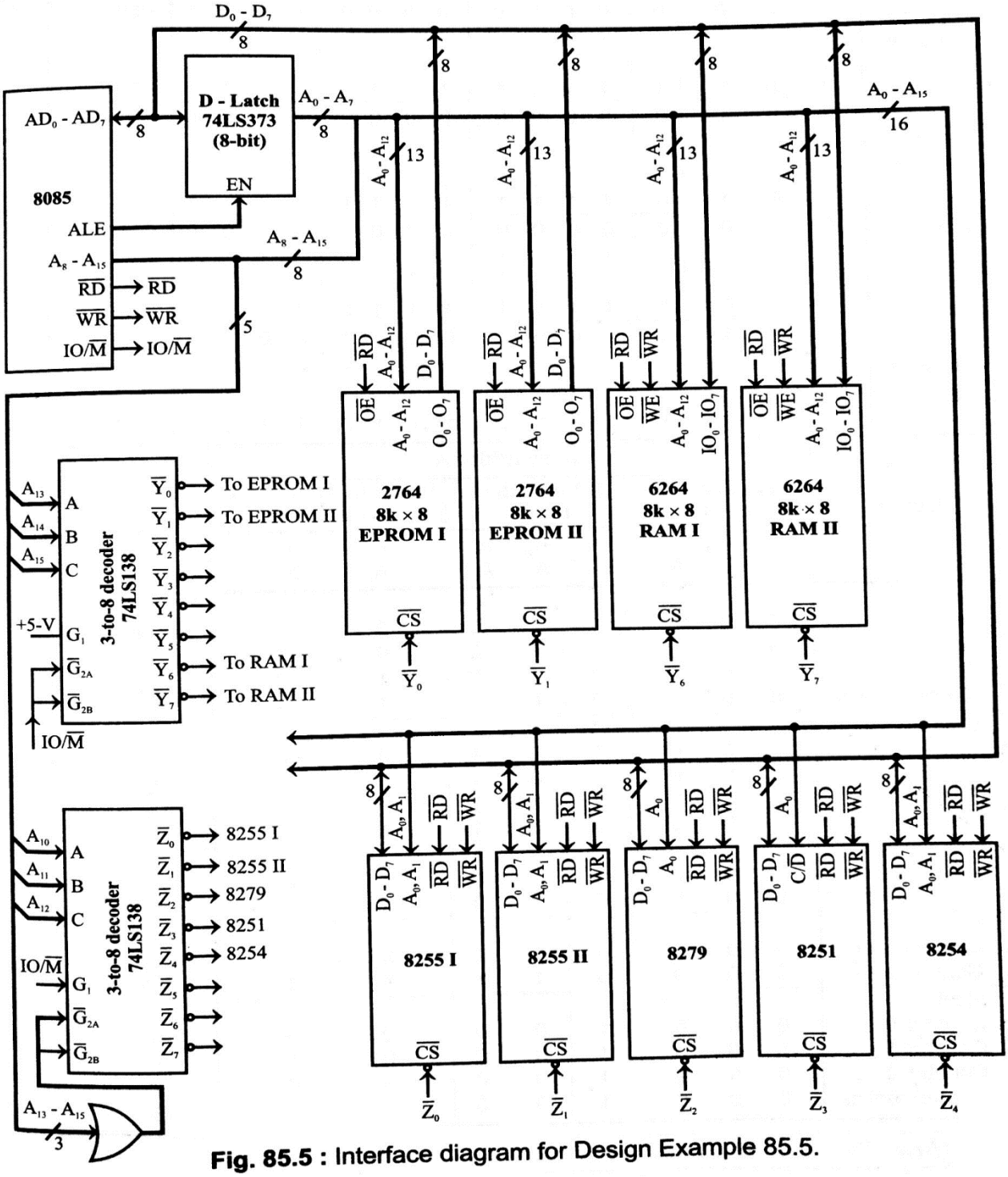


Fig. 85.5 : Interface diagram for Design Example 85.5.

TABLE - 85.5 : ADDRESS ALLOCATION TABLE FOR DESIGN EXAMPLE- 85.5

Device	Binary address																Hexa address
	Input to memory decoder			Input to memory address pins													
	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
2764 I 8 k × 8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFF
2764 II 8 k × 8	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	3FFF
6264 I 8 k × 8	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	C000
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	DFFF
6264 II 8 k × 8	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	E000
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFF

Device	Binary address								Hexa address
	IO decoder enable			IO decoder input			Input to IO device address pins		
	A_{15} A_7	A_{14} A_6	A_{13} A_5	A_{12} A_4	A_{11} A_3	A_{10} A_2	A_9 A_1	A_8 A_0	
8255 I									
Port-A	0	0	0	0	0	0	0	0	00
Port-B	0	0	0	0	0	0	0	1	01
Port-C	0	0	0	0	0	0	1	0	02
Control register	0	0	0	0	0	0	1	1	03
8255 II									
Port-A	0	0	0	0	0	1	0	0	04
Port-B	0	0	0	0	0	1	0	1	05
Port-C	0	0	0	0	0	1	1	0	06
Control register	0	0	0	0	0	1	1	1	07
8279									
Data register	0	0	0	0	1	0	X	0	08
Control register	0	0	0	0	1	0	X	1	09
8251									
Data register	0	0	0	0	1	1	X	0	0C
Control register	0	0	0	0	1	1	X	1	0D
8254									
Counter-0	0	0	0	1	0	0	0	0	10
Counter-1	0	0	0	1	0	0	0	1	11
Counter-2	0	0	0	1	0	0	1	0	12
Control register	0	0	0	1	0	0	1	1	13

Note : Don't care (X) is considered as zero.

A 8 kb memory requires 13 address lines (2^{13} = 8k). Hence, the address lines A_0 - A_{12} are used for selecting the memory locations. The unused address lines A_{13}, A_{14} and A_{15} are used as input to decoder 74LS138 (3-to-8-decoder) of the memory IC. The logic **low** enables of this decoder are tied to the IO/$\overline{M}$ of an 8085, so that this decoder is enabled for memory read/write operation. The other enable pins of the decoder are tied to appropriate logic levels permanently. Four outputs of the decoder are used to select memory ICs and the remaining four are kept for future expansion.

The EPROM is mapped in the beginning of memory space from 0000_H to $3FFF_H$. The RAM is mapped at the end of memory space from $C000_H$ to $FFFF_H$.

There are five peripheral ICs to be interfaced to the system. The chip select signals for these ICs are given through another 3-to-8 decoder 74LS138 (IO decoder). The input to this decoder is A_{10}, A_{11} and A_{12}. The address lines A_{13}, A_{14} and A_{15} are logically ORed and applied to **low** enable of the IO decoder. The logic **high** enable of the IO decoder is tied to the IO/$\overline{M}$ signal of an 8085, so that this decoder is enabled for IO read/write operation.

Here the higher order address lines can be used for decoding, because the processor outputs the 8-bit port address both on AD_0 to AD_7 and A_8 to A_{15}. The address lines A_0 and A_1 are used to select the internal devices of the peripheral ICs. The output of the decoder are used to select the ICs. Three outputs of the decoder will be spare for future expansion.

Note : Since the IO devices are IO-mapped in the system, 8-bit addresses have been allotted to them.

DESIGN EXAMPLE 6.6

In a microprocessor-based system using 8085, 8 kb EPROM and 8 kb RAM are needed. For interfacing IO devices two numbers of 8155 are required. Select suitable memories and explain how they are interfaced in the system. Interface the 8155 ports by IO mapping.

Solution

The IC 2764 (8 k × 8) is selected for EPROM memory and IC 6264 (8 k × 8) is selected for RAM memory. Both the memory ICs have time compatibility with an 8085 processor.

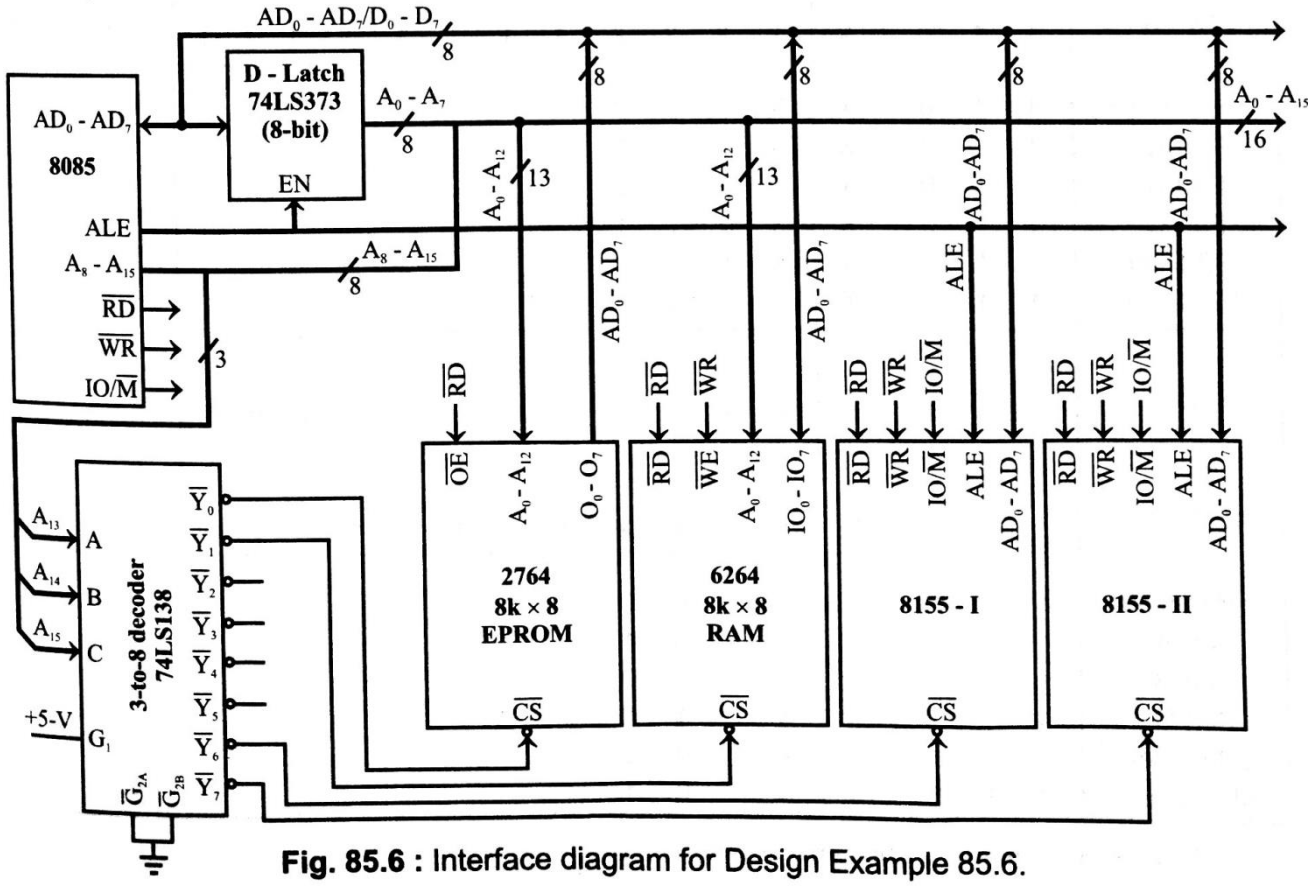


Fig. 85.6 : Interface diagram for Design Example 85.6.

TABLE - 85.6 : ADDRESS ALLOCATION TABLE FOR DESIGN EXAMPLE - 85.6

Device	Binary address																Hexa address
	Decoder input			Input to address pins of memory/8155													
	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
2764 8k × 8 EPROM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0001
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFF
6264 8k × 8 RAM	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000
	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	2001
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFF
8155 I RAM 256 × 8	1	1	0	X	X	X	X	X	0	0	0	0	0	0	0	0	C000
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	1	1	0	X	X	X	X	X	1	1	1	1	1	1	1	1	C0FF
Control register	1	1	0	X	X	0	0	0									C0
Port-A	1	1	0	X	X	0	0	1									C1
Port-B	1	1	0	X	X	0	1	0									C2
Port-C	1	1	0	X	X	0	1	1									C3
LSB timer	1	1	0	X	X	1	0	0									C4
MSB timer	1	1	0	X	X	1	0	1									C5
8155 II RAM 256 × 8	1	1	1	X	X	X	X	X	0	0	0	0	0	0	0	0	E000
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	1	1	1	X	X	X	X	X	1	1	1	1	1	1	1	1	E0FF
Control register	1	1	1	X	X	0	0	0									E0
Port-A	1	1	1	X	X	0	0	1									E1
Port-B	1	1	1	X	X	0	1	0									E2
Port-C	1	1	1	X	X	0	1	1									E3
LSB timer	1	1	1	X	X	1	0	0									E4
MSB timer	1	1	1	X	X	1	0	1									E5

Note : Don't care (X) is considered as zero.

The 8 kb memories require 13 address lines (2^{13} = 8 k). Hence, the address lines A_0 - A_{12} are used to select the memory locations.

In addition to 6264, each one of the 8155 chip provides a static RAM capacity of 256 bytes. The RAM locations of 8155 are selected by address lines A_0-A_6.

A 3-to-8 decoder, 74LS138 is used for generating chip select signals by decoding the address lines A_{13}, A_{14} and A_{15}.

The 8155 has an internal address latch and decoder to differentiate memory operation and IO operation. To utilize this facility the control signals ALE and IO/$\overline{M}$ are connected to an 8155.

The 8155 ports and memory locations can be selected from the decoder used for the memory devices. It differentiates the memory and IO operation from the IO/$\overline{M}$ signal. Eight bit addresses are allotted to the ports of 8155 and sixteen bit addresses are allotted to the RAM memory locations of an 8155.

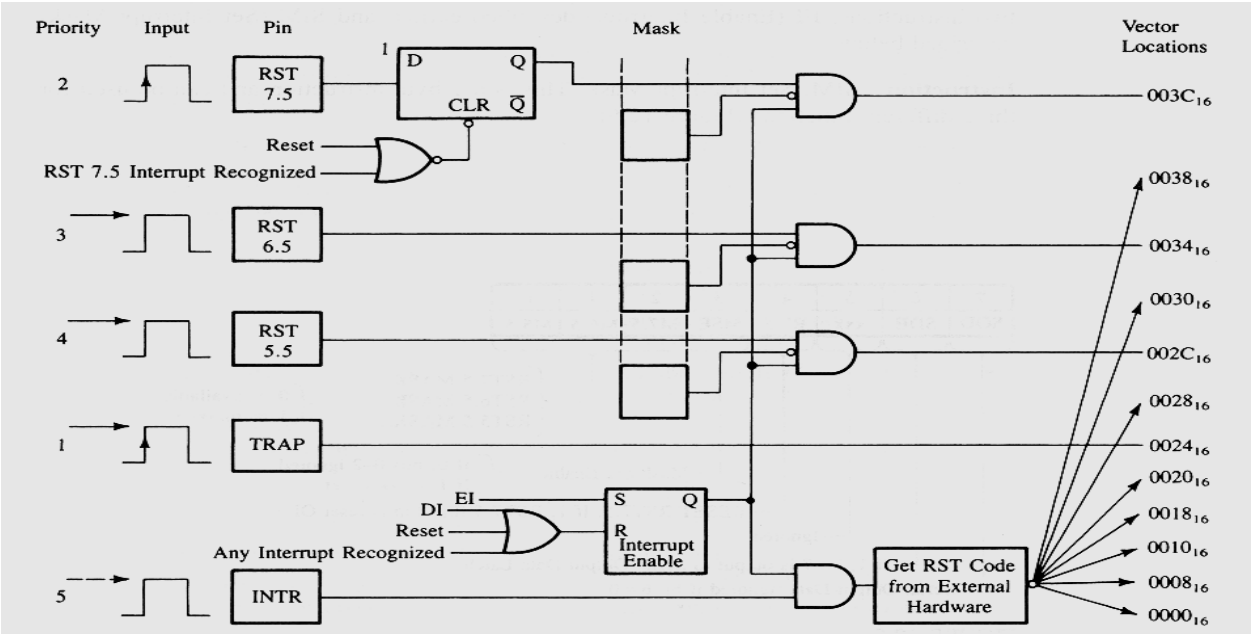
---

7. EXPLAINABOUT THE INTERRUPT STRUCTURE IN DETAIL. (April-2016), (May-2016), (Nov-2015), (April-2015)

Interrupt structure

- 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.
- Thee processor executes an interrupt service routine (ISR) addressed in Program counter.
- It returned to main program by RET instruction.

INTERRUPT STRUCTURE:



Types of Interrupts:

It supports two types of interrupts

a. Softwareinterruptsb. Hardware interrupt

a) SOFTWAREINTERRUPTS:

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

The 8085 has eight software interrupts from RST0 to RST 7. The vector address for these interrupts can be calculated as follows.

Interrupt number *8= Vector address

Vector address for interrupt RST 5 is 0028H

The table shows the vector address of all the interrupts.

Interrupt	Vector address
RST0	0000_H
RST1	0008_H
RST2	0010_H
RST3	0018_H
RST4	0020_H
RST5	0028_H
RST6	0030_H
RST7	0038_H

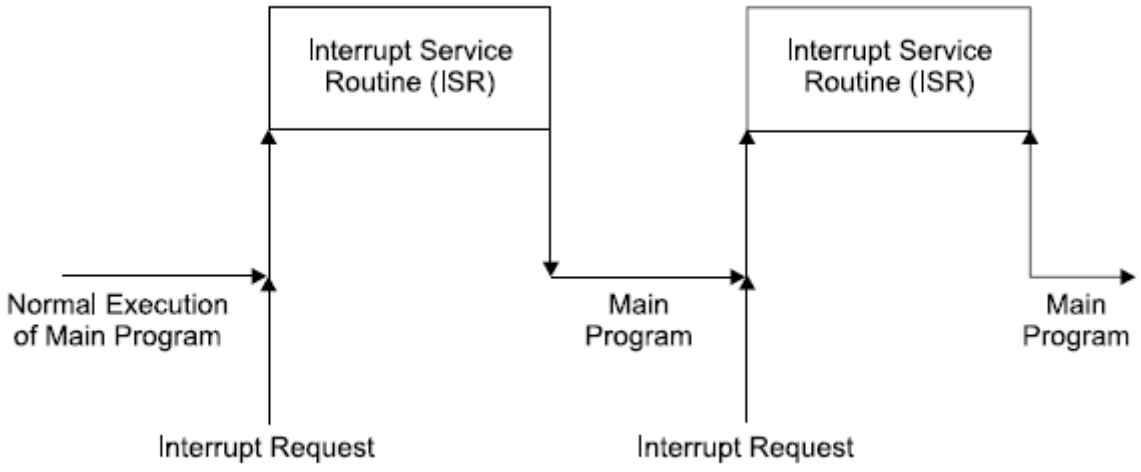
- The software instructions are program instructions. When a software interrupt instruction is executed, the processor executes an interrupt service subroutine (ISR) stored in the vector address of that software interrupt instruction.
- The software interrupts of 8085 are RSR 0, RST 1, RST 2, RST 3, RST 4, RST 5, RST 6 and RST 7 . The software interrupts of 8085 are vectored interrupts. The software interrupts cannot be masked and they cannot be disabled..
- The software interrupt instructions are included at the appropriate (or required) place in the main program.
- When the processor encounters the software instruction , it pushes the content of the PC (program counter) to stack.
- Then loads the vector address in PC and starts executing an ISR stored in this address.
- The last instruction of ISR will be RET instruction. When the RET instruction is executed, the
- Processors POP the content of the top of stack to PC.
  Hence the processor control returns to the main program after receiving the interrupt.

HARDWARE INTERRUPTS OF 8085:

Table 8085 interrupts

Interrupt	Maskable	Masking method	Vectored	Memory	Triggering method
INTR	Yes	DI/EI	No	No	Level sensitive
RST 5.5	Yes	DI/EI SIM	Yes	No	Level sensitive
RST 6.5	Yes	DI/EI SIM	Yes	No	Level sensitive
RST 7.5	Yes	DI/EI SIM	Yes	Yes	Edge sensitive
TRAP	No	None	Yes	No	Level and edge sensitive

INTERRUPTS	VECTOR ADDRESS
RST 7.5	003CH
RST 6.5	0034H
RST 5.5	002CH
TRAP	0024H



Interrupt execution

The hardware interrupts of 8085 are initiated by an external device by placing an appropriate signal at the interrupt pin of the processor.

- The processor keeps on checking the interrupt pins at the second T-state of last machine cycle of every instruction.
- If the processor finds a valid interrupt signal and if the interrupt is unmasked and enabled; then the processor accepts the interrupt. The acceptance of the hardware interrupt is acknowledged by sending an INTA signal to the interrupting device.
- When the interrupt is accepted; the processor saves the content of PC into the stack.
- The hardware interrupts of 8085 are TRAP , RST 7.5, RST 6.5, RST 5.5 and INTR . Except INTR all are vectored interrupts.
- In vectored interrupts the address to which the program control is transferred is fixed by the manufacturer.
- The vector addresses of hardware interrupts are given in table.
- TRAP is edge and level sensitive. Hence to initiate TRAP, the interrupt signal has to make a low to high transition and then it has to remain high until the interrupt is recognized.
- The RST 7.5 is edge sensitive. , and in order to initiate it ,the interrupt signal has to make a low to high transition and then it need not remain high until the interrupt is recognized.
- The RST 7.5, RST 6.5, RST 5.5 and INTR are level sensitive. Hence these interrupts should remain high, until it is recognized.
- The Trap is non maskable interrupt and RST 7.5, RST 6.5, RST 5.5 are maskable interrupt using SIM instruction. The status of maskable interrupts can be read into the accumulator by executing RIM instruction.
- All the interrupts except TRAP are disabled when the processor is resettedor can also be disabled by executing DI instruction.
- In order to enable them the processor has to execute EI instruction.

Priorities of Interrupts of an 8085

When all the interrupts are enabled, the priority of the hardware interrupts from highest to lowest is TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR. When the 8085 processor accept an interrupt, it will disable all the hardware interrupts except TRAP. Hence, in order to allow the higher priority interrupt while executing **I**nterrupt **S**ervice **R**outine (ISR) for lower priority interrupt, enable the interrupt system in the beginning of ISR of lower priority interrupt, by executing an EI instruction.

For example, if the processor accepts RST 5.5 interrupt, then it will disable RST 7.5, RST 6.5 and INTR interrupts. In order to allow the higher priority interrupt RST 7.5 and RST 6.5 while executing ISR of RST 5.5, the EI instruction should be executed in the beginning of ISR of RST 5.5.

The execution of software interrupt will not disable any hardware interrupt. Therefore while executing an ISR of software interrupts, the processor will recognize or allow the hardware interrupts.

8. Explain in details about enabling, disabling and masking of 8085 interrupts

Enabling, Disabling and Masking of 8085 Interrupts

TRAP

The interrupt TRAP is non-maskable and it cannot be disabled by DI instruction. Also, the TRAP is not disabled by system (processor) reset or after recognition of another interrupt. The only signal which can override TRAP is the 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.)

INTR

The interrupt INTR is disabled by any one of the following operations:

- Executing DI instruction
- System or processor reset
- After recognition (acceptance) of an interrupt

The interrupt INTR can be enabled by executing an EI instruction.

RST 7.5, RST 6.5 AND RST 5.5

The interrupts RST 7.5, RST 6.5 and RST 5.5 are disabled by any one of the following operations:

- Executing DI instruction
- System or processor reset
- After recognition (acceptance) of an interrupt

These hardware interrupts can be enabled by executing an EI instruction.

The 8085 provides 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 the 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 the accumulator and then executing the SIM instruction. The format of the 8-bit data is shown in Fig. **A**

The status of pending interrupts can be read from the accumulator after executing the RIM instruction. When RIM instruction is executed an 8-bit data is loaded in the accumulator, which can be interpreted as shown in Fig. **B**

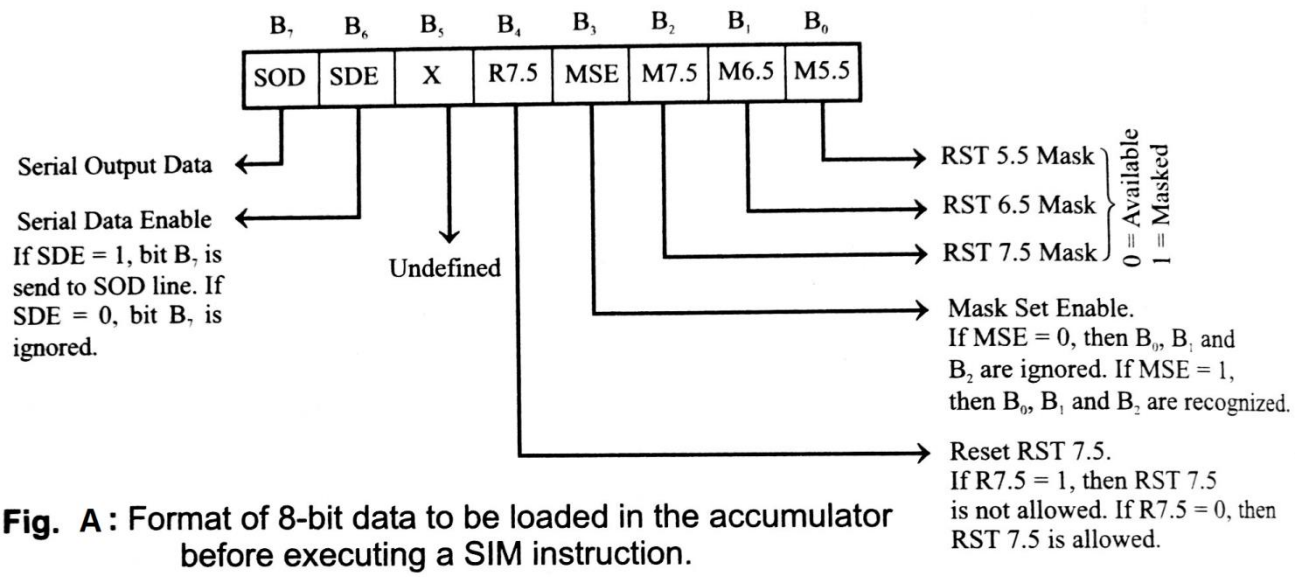


Fig. A : Format of 8-bit data to be loaded in the accumulator before executing a SIM instruction.

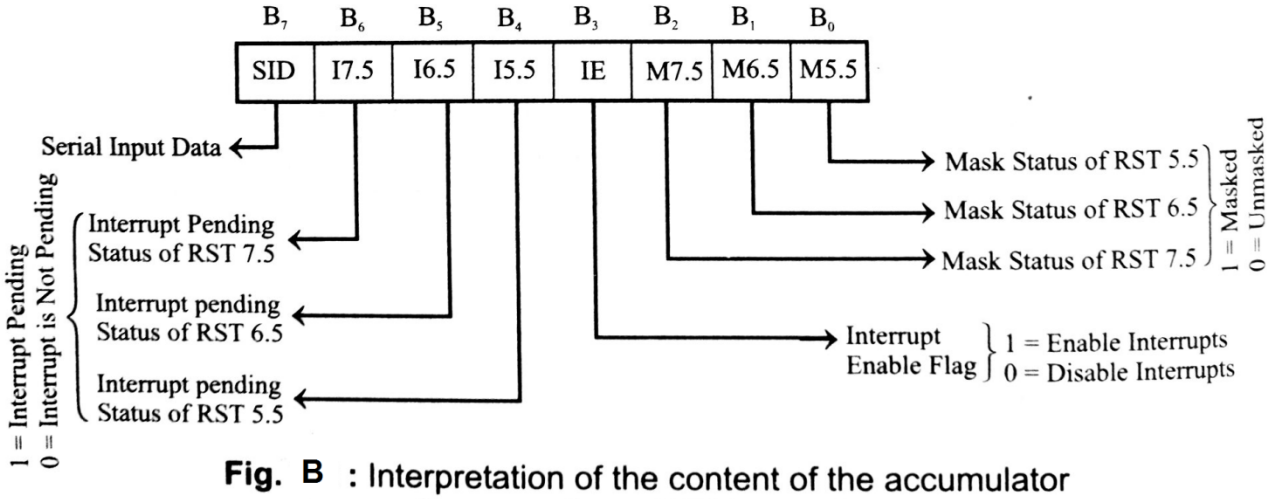


Fig. B : Interpretation of the content of the accumulator after executing a RIM instruction.

9.Explainclarification of data transfer take place in DMA controller.

DATA TRANSFER TECHNIQUES

The data transfer technique refers to the method of data transfer between the processor and peripheral devices. In a typical microcomputer, data transfer takes place between any two devices:

- Microprocessor and memory
- Microprocessor and I/O devices
- Memory and I/O devices

For effective data transfer between these devices, the timing parameters of the devices should be matched.

TYPES OF DATA TRANSFER:-

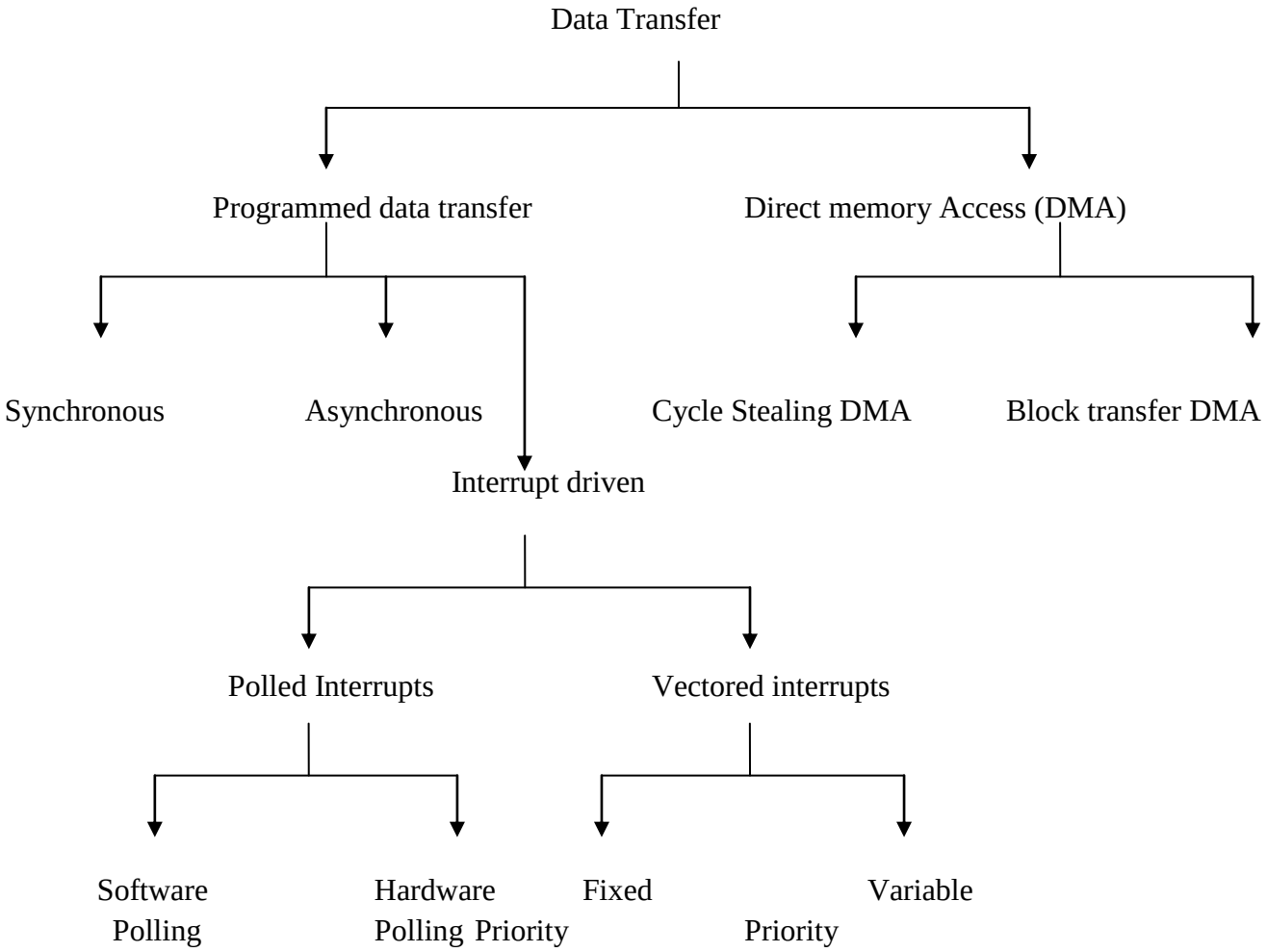
The data transfer technique has been broadly classified into the following two categories;

a)Programmed data transfer
b)Direct memory access (DMA) data transfer

a)PROGRAMMED DATA TRANSFER:

In programmed data transfer, a memory resident request the device for data transfer to or from one of the processor register. It is used when relatively small amount of data are to be transferred. They can be further classified into the following three types;

i. Synchronous data transfer
ii. Asynchronous data transfer
iii. Interrupt driven data transfer



i)SYNCHRONOUS DATA TRANSFER:

The synchronous data transfer scheme is the simplest of all data transfer schemes. In this scheme the processor does not check the readiness of the devices. The I/O device or peripheral should have matched timing parameters. Whenever data is to be obtained from the device or transferred to the device, the user program can issue a suitable instruction for the device. At the end of the execution of this instruction, the transfer would have been completed.

ii)ASYNCHRONOUS DATA TRANSFER:

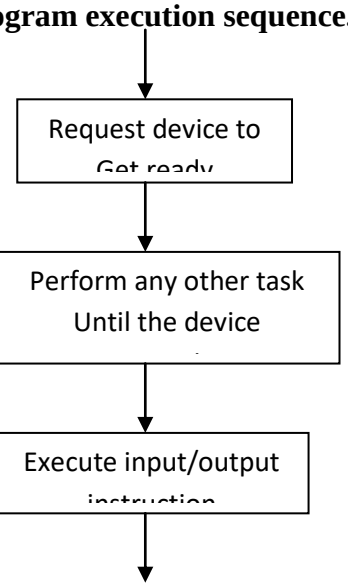
The asynchronous data transfer scheme is employed when the speed of the processor and I/O devices does not match. In this scheme the processor sends a request to the device for read/write operation. Then the processor keeps on polling the status of the devices. Once the device is ready, the processor execute a data transfer instruction to complete the process

iii)INTERRUPT DRIVEN DATA TRANSFER:

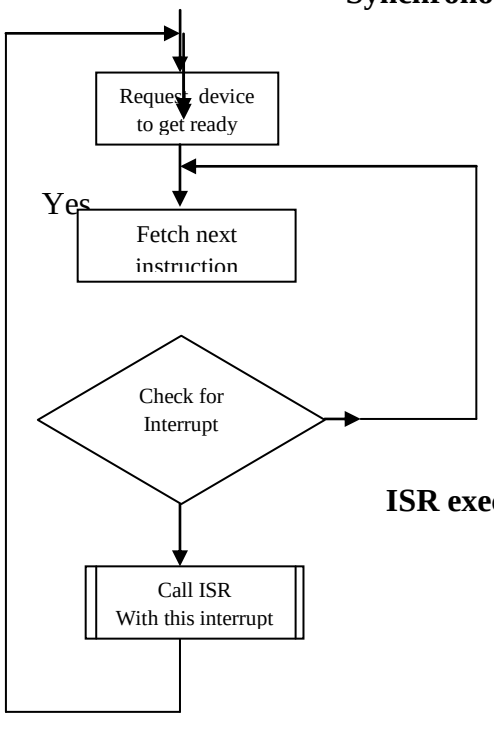
The interrupt driven data transfer scheme is the best method of data transfer for efficient utilization of processor time. In this scheme, the processor first initiates the I/O device for data transfer. After initiating the device, the processor will continue the execution of instructions in the program. Also at the end of every instruction the processor will check for a valid interrupt signal. If there is no interrupt then the processor will continue the execution.

When the I/O device is ready, it will interrupt the processor. On receiving an interrupt signal the processor will complete the current instruction execution and save the processor status in stack. Then the processor call an Interrupt Service Routine (ISR) to service the interrupting device. At the end of ISR, the processor status is retrieved from the stack and the processor starts executing its main program.

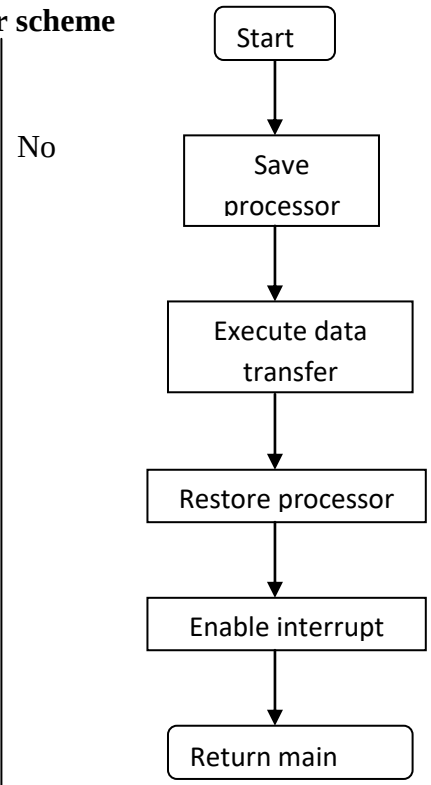
Main program execution sequence.



Synchronous data Transfer scheme



ISR execution sequence



b)DMA DATA TRANSFER:

Normally the data transfer from memory to I/O device or I/O device to memory can be achieved only through microprocessor. When data has to be transferred from memory to I/O device, first the processor send address and control signals to memory to read the data from memory. Then the processor send address and control signals to I/O device to write data to I/O device. Similarly, the data can be transferred from I/O device to memory.

In the data transfer method described above, the data cannot be directly transferred between memory and I/O devices, even though they are connected to common bus. This process is inevitable, because the processor cannot simultaneously select two devices. Hence a scheme called DMA has been developed in which the I/O device can access the memory directly for data transfer.

The DMA data transfer will be useful to transfer large amount of data between memory and I/O device in a short time.

The modes of DMA operations are two types;

i. Burst mode
ii. Cycle stealing

i)BURST MODE DATA TRANSFER:

As bus control is granted to the device controller, it continues with the controller till the data transfer is completed. After all the data has been transferred, the device interrupts the processor to indicate the completion to the user program. During this period the microprocessor is idling and it is in hold state.

The microprocessor exits from this state only after an interrupt is received or after the DMA request is withdrawn by the peripheral device. The duration of HOLD depends on I/O device speed, the memory speed and the number of bytes transferred. This type of DMA transfer is known as burst mode data transfer.

ii)CYCLE STEALING DATA TRANSFER:

The I/O device uses the concept of cycle stealing. The I/O device requests the processor for DMA cycle. When request is granted a byte or a word is transferred and DMA request is withdrawn. After sometime, when the device is again ready for data transfer, it repeats the above process.

Finally when the last data byte has been transferred, the device interrupts the processor indicating the end of the requested I/O operation. This type of DMA access is cycle stealing data transfer.

10.With a neat diagram describe the working of DMA controller. (7m-April-2015), or Draw the schematic diagram of DMA controller and explain how data transfer take place. (11m-nov-2015).

Programmable Peripheral Interface - INTEL 8255

The INTEL 8255 is a device used to implement parallel data transfer between processor and slow peripheral devices like ADC, DAC, keyboard, 7-segment display, LCD, etc.

The 8255 has three ports: Port-A, Port-B and Port-C. The ports A and B are 8-bit parallel ports. Port-A can be programmed to work in any one of the three operating modes as input or output port. The three operating modes are:

Mode - 0 → Simple IO port
Mode - 1 → Handshake IO port
Mode - 2 → Bidirectional IO port

The port-B can be programmed to work either in mode-0 or mode-1 as input or output port. The port-C pins (8 pins) have different assignments depending on the mode of ports A and B. If ports A and B are programmed in mode-0, then the port-C can perform any one of the following function:

1. As 8-bit parallel port in mode-0 for input or output.
2. As two numbers of 4-bit parallel port in mode-0 for input or output.
3. The individual pins of port-C can be set or reset for various control applications.

If port-A is programmed in mode-1/mode-2 and port-2 is programmed in mode-1 then some of the pins of port-C are used for handshake signals and the remaining pins can be used as input/ output lines or individually set/reset for control applications.

IO Modes of 8255

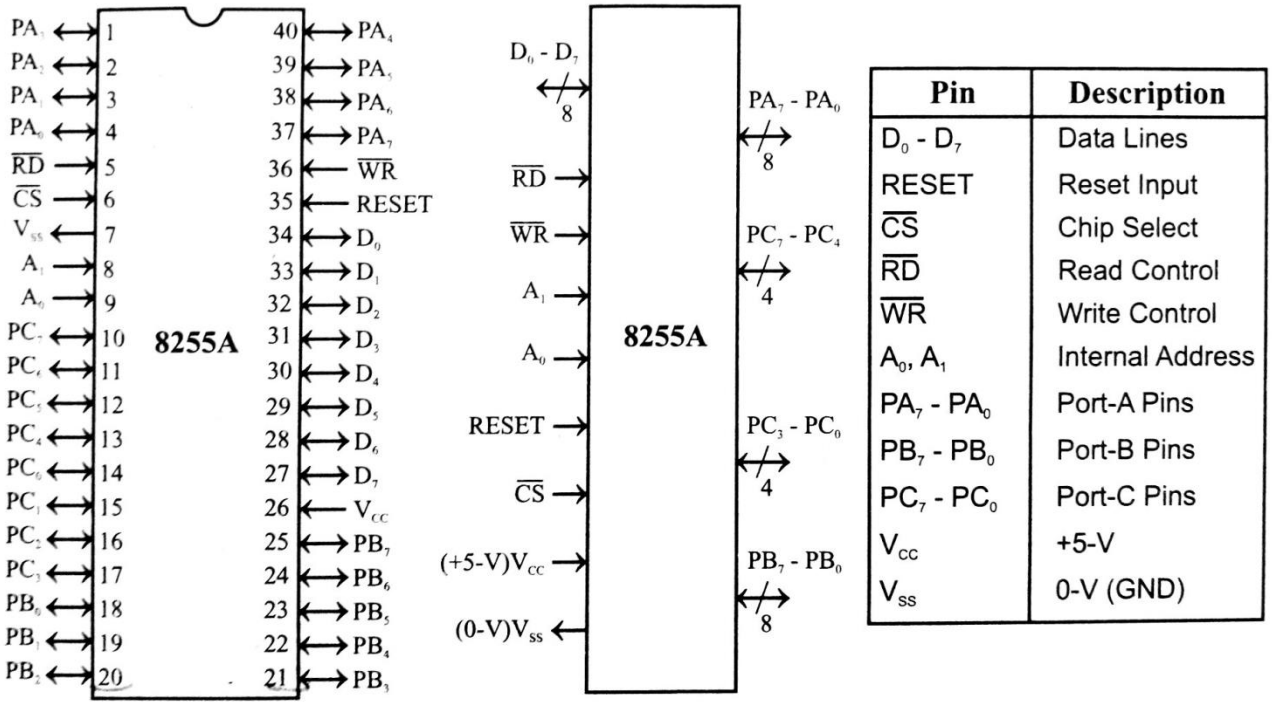
Mode-0 : In this mode, all the three ports can be programmed either as the input or the output port. In mode-0, the outputs are latched and the inputs are not latched. The ports do not have handshake or interrupt capability. The ports in mode-0 can be used to interface DIP switches, hexa-keypad, LEDs and 7-segment LEDs to the processor.

Mode-1 : In this mode, only ports A and B can be programmed either as the input or output port. In mode-1, handshake signals are exchanged between the processor and peripherals prior to data transfer. The port-C pins are used for handshake signals. Input and output data are latched. Interrupt driven data transfer scheme is possible.

Mode-2 : In this mode, the port will be a bidirectional port. (i.e., the processor can perform both read and write operations with an IO device connected to a port in mode-2.) Only port-A can be programmed to work in mode-2. Five pins of port-C are used for handshake signals. This mode is used primarily in applications such as data transfer between two computers or floppy disk controller interface.

Pins, Signals and Internal Block Diagram of 8255

The pin description of 8255 is shown in Fig. 9.15. It has 40 pins and requires a single +5-V supply. The internal block diagram of 8255 is shown in Fig. 9.16.



Pin	Description
D_0 - D_7	Data Lines
RESET	Reset Input
$\overline{CS}$	Chip Select
$\overline{RD}$	Read Control
$\overline{WR}$	Write Control
A_0, A_1	Internal Address
PA_7 - PA_0	Port-A Pins
PB_7 - PB_0	Port-B Pins
PC_7 - PC_0	Port-C Pins
V_{CC}	+5-V
V_{SS}	0-V (GND)

Fig. 9.15 : Pin description of 8255.

The ports are grouped as Group A and Group B. The group A has port-A, port-C upper and its control circuit. The group B comprises port-B, port-C lower and its control circuit. The read/ write control logic requires six control signals. These signals are:

$\overline{RD}$ (Read) : This control signal enables the read operation. When this signal is **low**, the microprocessor reads data from a selected IO port of the 8255A.

$\overline{WR}$ (Write) : This control signal enables the write operation. When this signal goes **low**, the microprocessor writes into a selected IO port or the control register.

RESET : This is an active **high** signal. It clears the control register and sets all ports in the input mode.

$\overline{CS}$, A_0 and A_1 : These are device select signals. The address lines A_0 and A_1 of 8255 can be connected to any two address lines of the processor to provide internal addresses.The A_0 and A_1 selects any one of the 4 internal devices as shown in Table-9.3. The 8255 will remain in **high impedance** state if the signal input to $\overline{CS}$ is **high** and the device can be brought to normal logic by making the signal input to $\overline{CS}$ as logic **low**.

TABLE - 9.3

Internal address		Device selected
A_1	A_0	
0	0	Port-A
0	1	Port-B
1	0	Port-C
1	1	Control Register

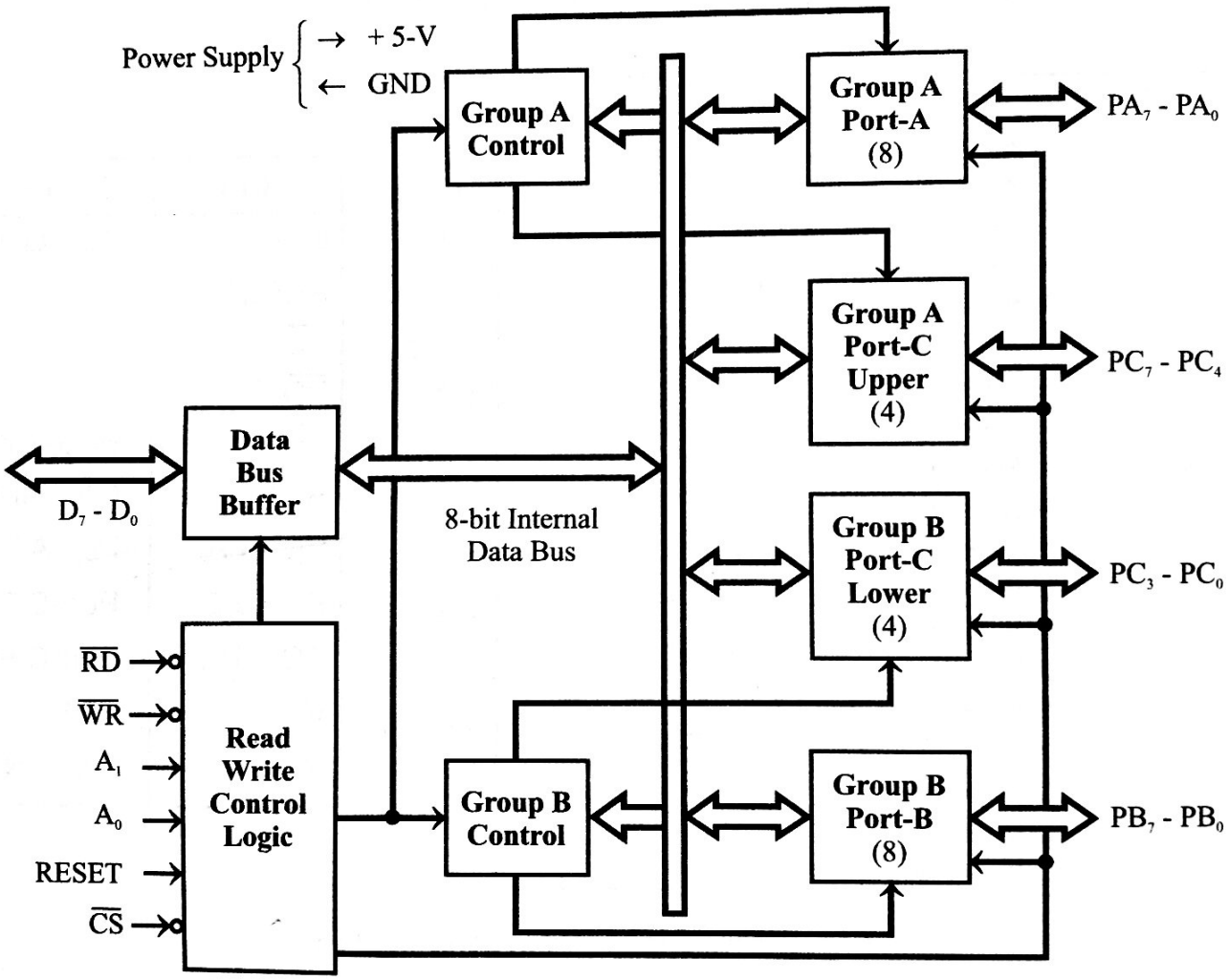


Fig. 9.16 : Internal block diagram of 8255.

11. Explain the operation of mode '2' in 8255 PPI in detail with necessary timing diagrams. (11m-Nov-2014)

Programming (or Initializing) 8255

The 8255 has two control words: IO Mode Set control Word (MSW) and Bit Set/Reset (BSR) control word. The MSW is used to specify IO functions and BSR word is used to set/reset individual pins of port-C. Both the control words are written in the same control register. The control register differentiates them by the value of bit B_7.The BSR control word does not affect the functions of ports A and B.

Bit B_7 of the control register specifies either the IO function or the bit set/reset function. If $B_7 = 1$, then the bits B_6- B_0 determine the IO functions in various modes. If bit $B_7 = 0$, then the bits B_6- B_0 determine the pin of port-C to be set or reset.

The 8255 ports are programmed (or initialized) by writing a control word in the control register. For setting IO functions and mode of operation, the IO mode set control word is sent to the control register. For setting/resetting a pin of port-C, the bit set/reset control word is sent to the control register. The format of the IO mode set control word is shown in Fig. 9.20 and the format of bit set/reset control word is shown in Fig. 9.21. The various functions (assignments) of port-C pins during the different operating modes of ports A and B are listed in Table-9.7.

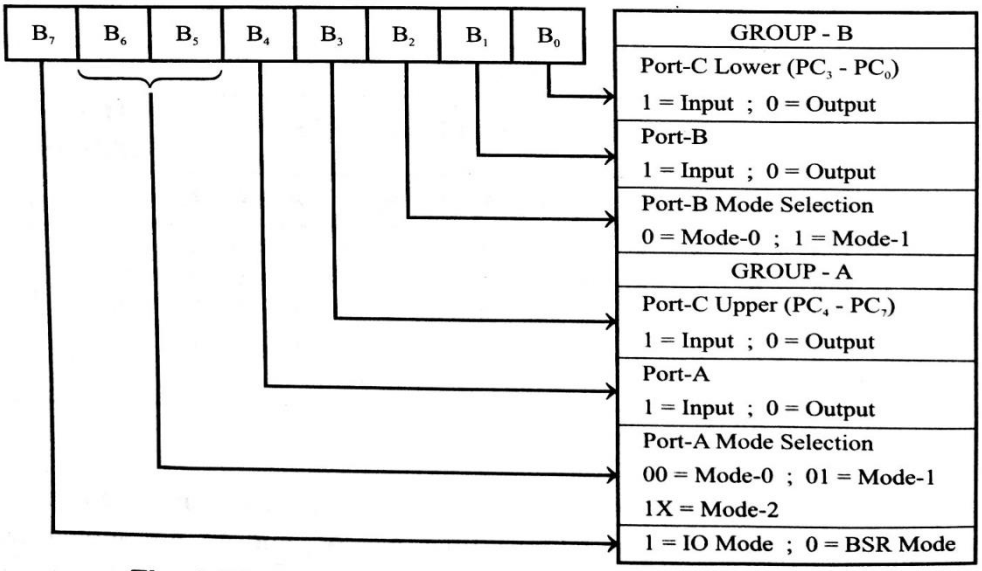
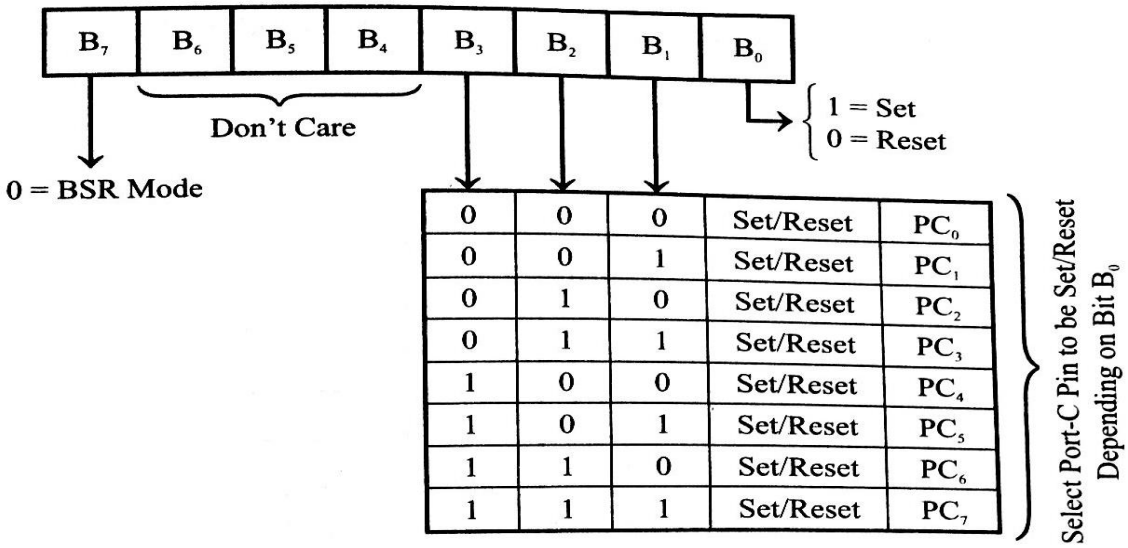


Fig. 9.20 : Format of IO mode set control word of 8255.



0	0	0	Set/Reset	PC_0
0	0	1	Set/Reset	PC_1
0	1	0	Set/Reset	PC_2
0	1	1	Set/Reset	PC_3
1	0	0	Set/Reset	PC_4
1	0	1	Set/Reset	PC_5
1	1	0	Set/Reset	PC_6
1	1	1	Set/Reset	PC_7

Fig. 9.21 : Format of Bit Set/Reset control word of 8255.

In handshake mode (i.e., in mode-1 and mode-2), the data transfer between the processor and the port can be implemented either by the interrupt method or by checking the status of the 8255 ports. In an interrupt-driven data transfer scheme, when the port is ready it interrupts the 8086 processor through the NMI/INTR pin for a read/write operation. In status check technique, the 8086 processor can check the status of ports A and B by reading port-C. When the port is ready for data transfer, the processor executes a read/write cycle.

The 8255 has two internal flip-flops as interrupt enables ($INTE_A$ and $INTE_B$) for port-A and port-B interrupt signals. In interrupt driven data transfer scheme, the 8255 generates an interrupt signal, only if these flip-flops are enabled by using BSR control word. The $INTE_A$ is enabled by setting PC_4 to **high** and $INTE_B$ is enabled by setting PC_2 to **high** using BSR control word. The interrupt signal can be disabled by resetting these two bits to zero using BSR control word.

PORT-C PIN ASSIGNMENTS

Functions of Ports A and B	PC_7	PC_6	PC_5	PC_4	PC_3	PC_2	PC_1	PC_0
Ports A and B in mode-0 Input/Output	IO	IO	IO	IO	IO	IO	IO	IO
Ports A and B in mode-1 Input ports	IO	IO	IBF_A	$\overline{STB}_A$	$INTR_A$	$\overline{STB}_B$	IBF_B	$INTR_B$
Ports A and B in mode-1 Output ports	$\overline{OBF}_A$	$\overline{ACK}_A$	IO	IO	$INTR_A$	$\overline{ACK}_B$	$\overline{OBF}_B$	$INTR_B$
Port-A in mode-2 Port-B in mode-0	$\overline{OBF}_B$	$\overline{ACK}_A$	IBF_A	$\overline{STB}_A$	$INTR_A$	IO	IO	IO

IO - Input /Output line	$\overline{OBF}$ - Output Buffer Full
$\overline{STB}$ - Strobe	$\overline{ACK}$ - Acknowledge
IBF - Input Buffer Full	The subscript A denotes port-A signal.
INTR - Interrupt Request	The subscript B denotes port-B signal.

When port-A and port-B are programmed in handshake mode (i.e., in mode-1 and mode-2), the port-C can be read to know the readiness of the ports for data transfer. The format of the status word read from port-C is shown in Fig. 9.22.

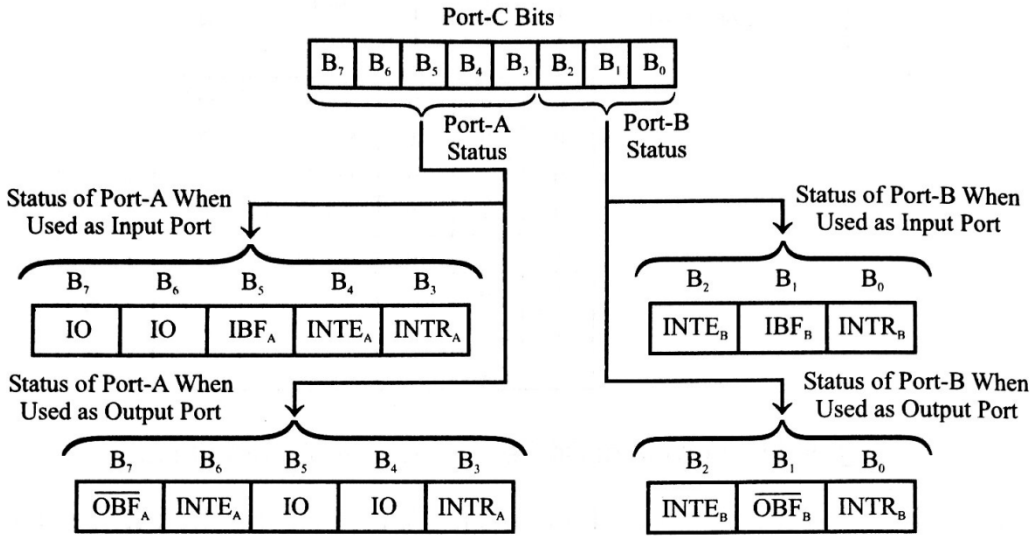


Fig. 9.22 : Format of status word of 8255 for handshake input and output operation.

8255 Handshake input port (Mode-1)

The signals used for data transfer between input device and 8085 microprocessor using port-A of 8255 as handshake input port (Mode-1) are shown in Fig. 9.23.

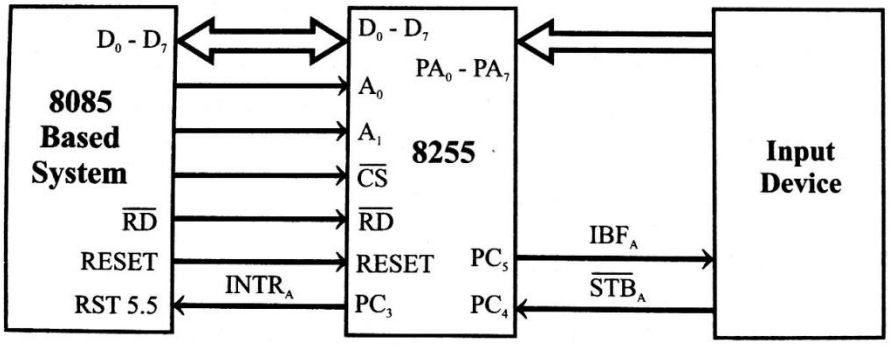


Fig. 9.23 : Port-A of 8255 as handshake input port (Mode-1).

1. The input device checks the IBF_A signal. If it is **low,** then the input device places the data on the port lines PA_0-PA_7 and asserts $\overline{STB}_A$ **low** and after a delay time $\overline{STB}_A$ is asserted **high.**
2. When $\overline{STB}_A$ is **low,** the 8255 asserts IBF signal **high** and at the rising edge of $\overline{STB}_A$ the data is latched to the port and $INTR_A$ is set **high.**
3. When $INTR_A$ goes **high,** the processor is interrupted through the RST 5.5 input pin to execute a subroutine for reading the data from the port. For a read operation, the processor asserts $\overline{RD}$ **low** and then **high.**
4. When $\overline{RD}$ is **low,** the $INTR_A$ is reset (asserted **low**) by 8255 and at the rising edge of $\overline{RD}$, the IBF is asserted **low** and the input device can send the next data.

8255 Handshake output port (Mode-1)

The signals used for data transfer between output device and an 8085 microprocessor using port-A of 8255 as the handshake output port (Mode-1) are shown in Fig. 9.24.

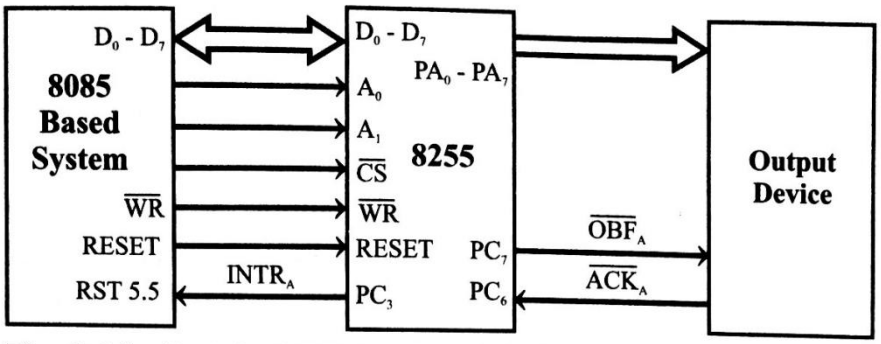


Fig. 9.24 : Port-A of 8255 as handshake Output port (Mode-1).

1. When the port is empty, the processor writes a byte in the port.
2. For writing a data to port, the processor asserts $\overline{WR}$ **low** and then **high.** At the rising edge of $\overline{WR}$, both the $INTR_A$ and $\overline{OBF}_A$ are asserted **low** by the 8255.
3. The $\overline{OBF}_A$ signal informs the output device that the data is ready. If the output device accepts the data then it sends an acknowledgement signal by asserting $\overline{ACK}_A$ **low** and then **high.**
4. When $\overline{ACK}_A$ is **low,** the $\overline{OBF}_A$ is asserted **high** by the 8255. When $\overline{ACK}_A$ is **high,** the $INTR_A$ is set (asserted **high**), to interrupt the processor.
5. When $INTR_A$ goes **high,** the processor is interrupted through RST 5.5 input pin to execute an interrupt service routine to load the next data in the output port.

8255 Bidirectional Port (Mode-2)

The signals used for data transfer between IO device and 8085 microprocessor using port-A of 8255 as a bidirectional port (Mode-2) are shown in Fig. 9.25.

Note : Only port-A can work in mode-2.

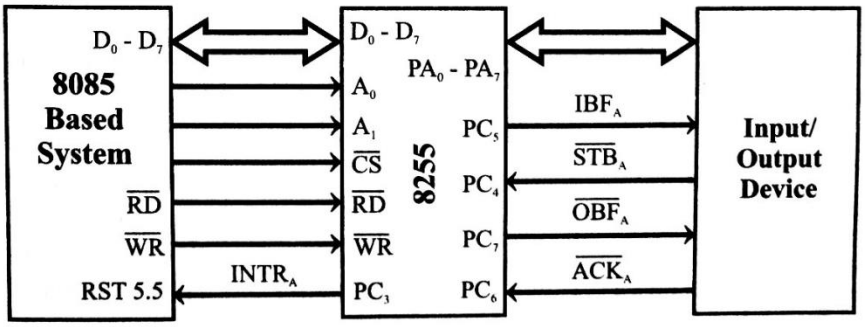


Fig. 9.25 : Port-A of 8255 as bidirectional port (Mode-2).

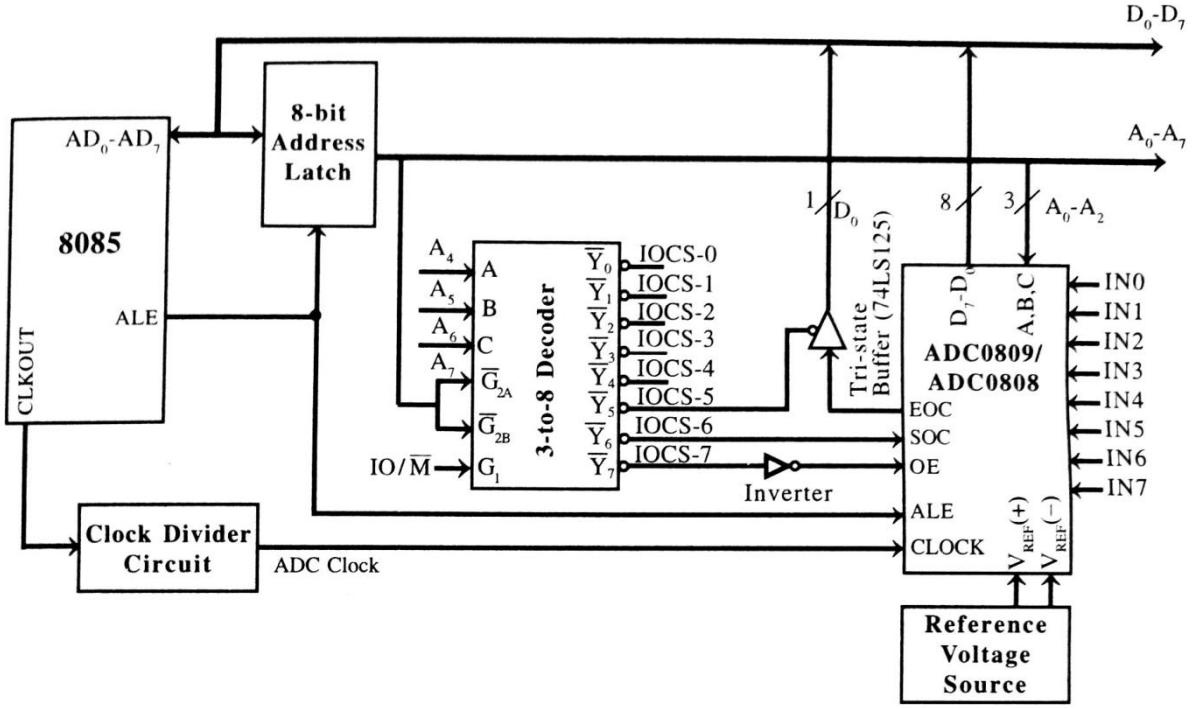
In mode-2, the port can be used either as an input port or as an output port. At any one time, the processor will perform either read or write operation. In mode-2, the read operation can be followed by write, or write operation can be followed by read. The signals involved and the operations performed for read operation are similar to mode-1 input port. The signals involved and the operations performed for write operation are similar to mode-1 output port.

12.With a neat diagram explain the ADC interface to microprocessor.(11m-April-2016)

Or

Explain the process of interfacing of ADC with 8085 microprocessor with relevant diagram.(11m-May-2014)

A simple schematic for interfacing ADC0809/ADC0808 with 8085 microprocessor is shown in Fig. 9.93. The ADC can be either memory-mapped or IO-mapped in the system. Here the ADC is IO-mapped in the system. The chip select signals for IO-mapped devices are generated by using a 3-to-8 decoder. The address lines A_4, A_5 and A_6 are used as input to decoder. The address line A_7 and the control signal IO/$\overline{M}$ are used as enable for the decoder. The decoder generates eight chip select signals (IOCS-0 to IOCS-7), and out of this three chip select signals are used for ADC interface.



Interfacing ADC0809/ADC0808 with 8086 microprocessor.

The chip select signal IOCS-6 is used to give **S**tart **O**f **C**onversion (SOC) signal to ADC along with a channel address. The chip select IOCS-5 is used to enable the tristate buffer provided for interfacing EOC with data bus. The chip select signal IOCS-7 is inverted and used to enable the output buffer of ADC whenever the digital data has to read from the ADC.

The output clock signal of an 8085 microprocessor is divided by suitable clock divider circuits and used as a clock signal for the ADC. A separate voltage source has to be provided to give an accurate reference voltage levels. The **E**nd **O**f **C**onversion (EOC) signal of ADC is connected to the bus line D_0 of the system through a tristate buffer, so that the processor can check for a valid EOC before reading the output buffer of the ADC.

The working of ADC 0809 with 8085 will be as follows:

1. First the processor selects a channel by sending an address and SOC pulse is asserted **high** and **low**.
2. Once address of the channel and SOC pulse are applied, the ADC will start converting the signal at the selected channel.
3. Then the processor keeps on polling the status of the EOC to verify whether it is set to one. (when the conversion is completed by ADC0809 the EOC is set to one.)
4. When the processor finds a valid EOC, then it will read the digital value from the output buffer of ADC.

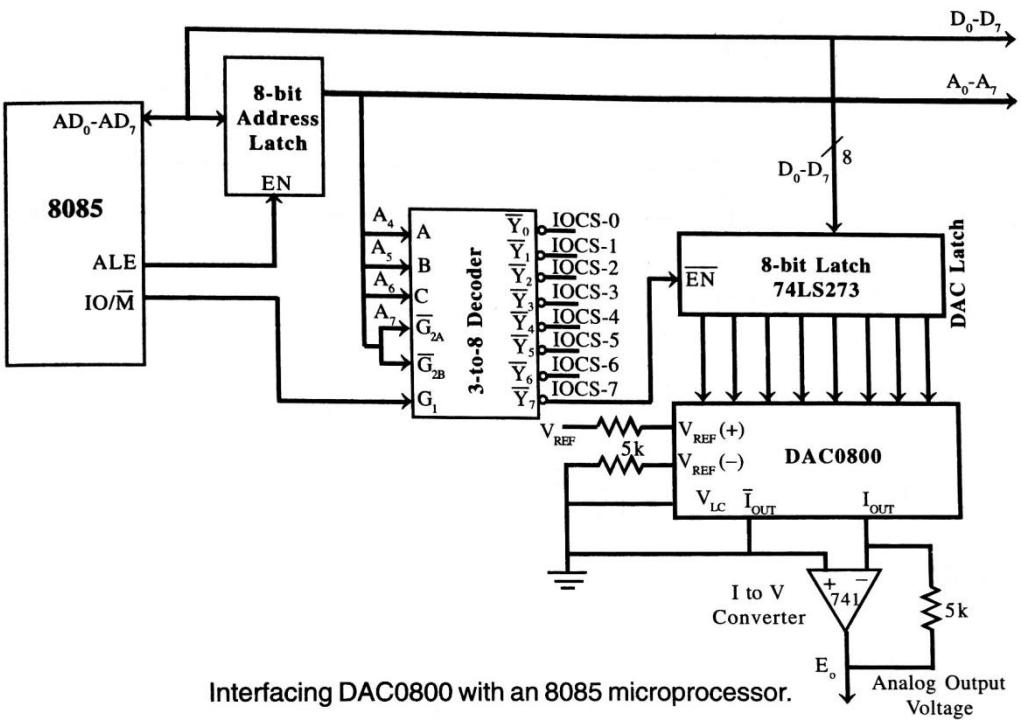
IO ADDRESS OF ADC0809/ADC0808 INTERFACED TO 8085 AS SHOWN IN

Operation performed	Binary address								Hexa address
	Decoder input/enable				Address input to ADC				
	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
SOC channel-0	0	1	1	0	x	0	0	0	60
SOC channel-1	0	1	1	0	x	0	0	1	61
SOC channel-2	0	1	1	0	x	0	1	0	62
SOC channel-3	0	1	1	0	x	0	1	1	63
SOC channel-4	0	1	1	0	x	1	0	0	64
SOC channel-5	0	1	1	0	x	1	0	1	65
SOC channel-6	0	1	1	0	x	1	1	0	66
SOC channel-7	0	1	1	0	x	1	1	1	67
Read EOC	0	1	0	1	x	x	x	x	50
Read ADC output	0	1	1	1	x	x	x	x	70

13. **With a neat diagram explain the DAC interface to microprocessor.**

Interfacing DAC0800 With 8085

The DAC0800 can be interfaced to an 8085 system bus by using an 8-bit latch and the latch can be enabled by using one of the chip select signals generated for IO devices. A simple schematic for interfacing DAC0800 with 8085 is shown in Fig. 9.85. In this schematic, the DAC0800 is interfaced using an 8-bit latch 74LS273 to the system bus. The 3-to-8 decoder 74LS138 is used to generate chip select signals for IO devices. The address lines A_4, A_5 and A_6 are used as input to decoder. The address line A_7 and the control signal $IO/\overline{M}$ are used as enable for the decoder. The decoder will generate eight chip select signals and in this, the signal IOCS-7 is used as enable for latch of the DAC. The IO address of the DAC is shown in Table-9.31.



Interfacing DAC0800 with an 8085 microprocessor.

In order to convert a digital data to analog value, the processor has to load the data to latch. The latch will hold the previous data until the next data is loaded. The DAC will take definite time to convert the data. The software should take care of loading successive data only after the conversion time. The DAC 0800 produces a current output, which is converted to voltage output using a I to V converter.

IO ADDRESS OF DAC LATCH SHOWN IN FIG.

Device	Binary address								Hexa address
	Decoder Input and enable				Unused address lines				
	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
DAC Latch 74LS273	0	1	1	1	x	x	x	x	70

UNIVERSITY QUESTIONS

PART-A

1.What is an Interrupt? Haw the interrupt are classified?

2.What is Vectored and Non- Vectored interrupt?

3.List the Software and Hardware interrupts of 8085 ?

4.What is masking and why it is required?

5.What is the role of interrupt service routine?

6.What is the need for interrupt controller ?

7. How the hardware interrupt of 8085 can be masked or unmasked?

Part-B

1. Explain the interrupt mechanism, types and priority of 8085 microprocessor.(11m-April-2016)

2. With a neat diagram explain the ADC interface to microprocessor.(11m-April-2016)

3. Draw the circuit diagram of an 8085 system, having 4KB EPROM and two 8KB RAM IC's. The starting address of the EPROM is 0000H and that of RAM is 8000H.(11m-May-2016)

4. What is the need of interrupt structures? Explain the hardware interrupt structure of 8085.(11m-May-2016).

5. With a neat diagram describe the working of DMA controller. (7m-April-2015)

6. Differentiate between I/O mapped I/O and memory mapped I/o. (4m-April-2015)

7. Explain the interrupt mechanism of 8085 microprocessor. (6m-April-2015)

8. Write a program for block transfer of data bytes. (5m-April-2015)

9. Draw the schematic diagram of DMA controller and explain how data transfer take place. (11m-nov-2015).

10. Explain the vectored interrupt in detail with their respective memory location.(11m-Nov-2015)

11. Explain the difference between memory mapped I/o and standard IO mapped IO. (11m-Dec-2014)

12. Draw the memory interface diagram to 8085 processor with 2 numbers of 4 Kb EPROM and one number of 8Kb RAM. Explain the system and allocate binary addresses to memory IC's.(11m-Dec-2014)

13. Draw the hardware arrangement for data acquisition with 0809 ADC interfaced with 8085 μP. Also write an assembly language program to acquire data from channel 'o'. (11m-Nov-2014)

14. Explain the operation of mode '2' in 8255 PPI in detail with necessary timing diagrams. (11m-Nov-2014)

15. Design a memory system for the 8085 microprocessor such that it should contain 8KB of EPROM and 8KB of RAM.(11m-May-2014)

16. Explain the process of interfacing of ADC with 8085 microprocessor with relevant diagram.(11m-May-2014)

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

18. Design a system in which the available 64kb memory space is equally divided between EPROM and RAM. Interface the EPROM and RAM with 8085 processor.

19. Design a system in which 32kb memory space is implemented using four numbers of 8kb memory. Interface the EPROM and RAM with 8085 processor.

20. Design a system in which the 64kb memory space is implemented using eight numbers of 8kb memory. Interface the EPROM and RAM with 8085 processor.

Department of Electrical and Electronics Engineering

Subject Name: **Microprocessors and Microcontrollers** Subject Code: **EE T63**

Prepared by:

Mrs.S.Punitha, Assistant Professor/EEE
Mr.V.Malarselvam, Assistant Professor/EEE

Verified by: **Approved by:**

Unit-IV

Study of Architecture and programming of ICs: Programmable Peripheral device(8255), Timer/ Counter (8253), Programmable keyboard displayinterfaces (8279) - Programmable interrupt controller (8259) - USART (8251). Microprocessor Applications-stepper motor control - temperature control-traffic light control.

8255 Programmable Peripheral Interface

Most of µp application will have two processes in common

- Reading data from an external device
- Writing data into another external device.

Special interface circuits are available for this purpose. Such interface circuits are called peripheral interface circuits or I/O ports.The INTEL 8255 is one such peripheral interface chip. It can be used with INTEL 8085 CPU.

Introduction:

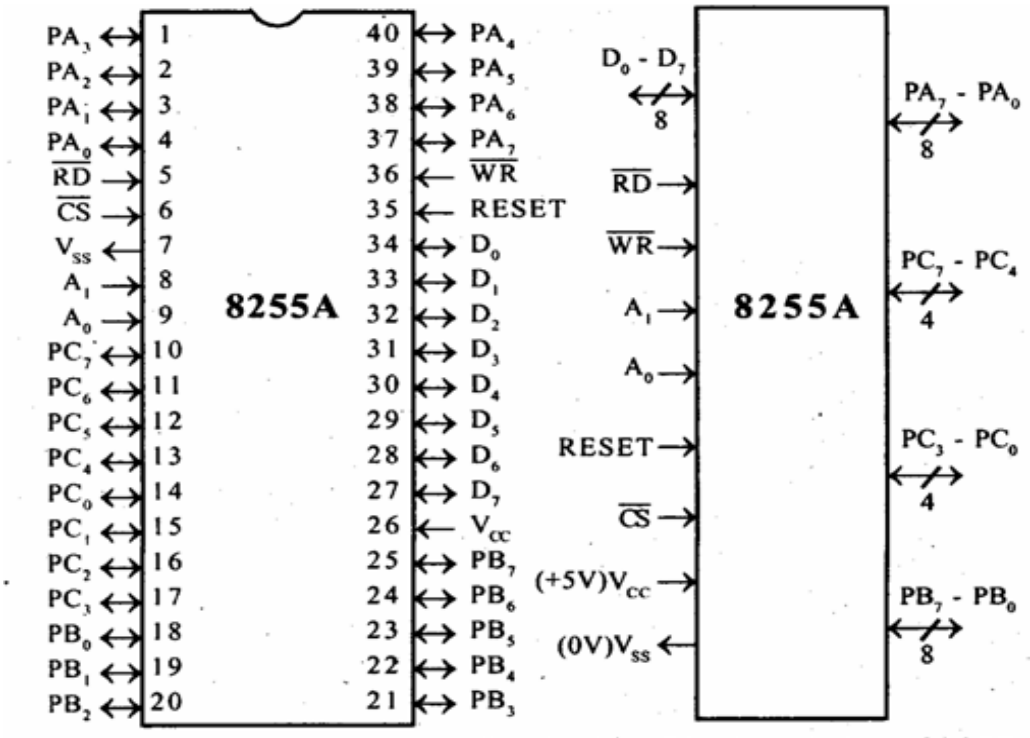
- The 8255A is a widely used programmable, parallel I/O device.
- It can be programmed to transfer data between peripheral devices and µp.
- It is compatible with all INTEL and most other microprocessor.
- It is completely TTL compatible.

Features:

- It is an important general purpose I/O device that can be used with almost any INTEL µp.
- It is packed in 40 pin DIP.
- All I/O pins of pins of 8255 have 2.5 mA DC driving capacity (i.e. sources current of 2.5 mA).
- Out of 40 pins, 24 pins are used as I/O pins. these 24 pins are divided into 3port(A,B,C)
- These ports can be programmed as inputs or outputs.

- The 8 bits of port C can be used as individual bits or be grouped in two 4-bit ports:$C_{upper(cu)}$ and $C_{lower\ (cl)}$ for the purpose of programming.
- These port are programmable as
    1. GROUP A:port A(PA_7-PA_0) & c_u(PC_7-PC_4)
    2. GROUP B:port B(PB_7-PB_0)& c_l (PC_3-PC_0)

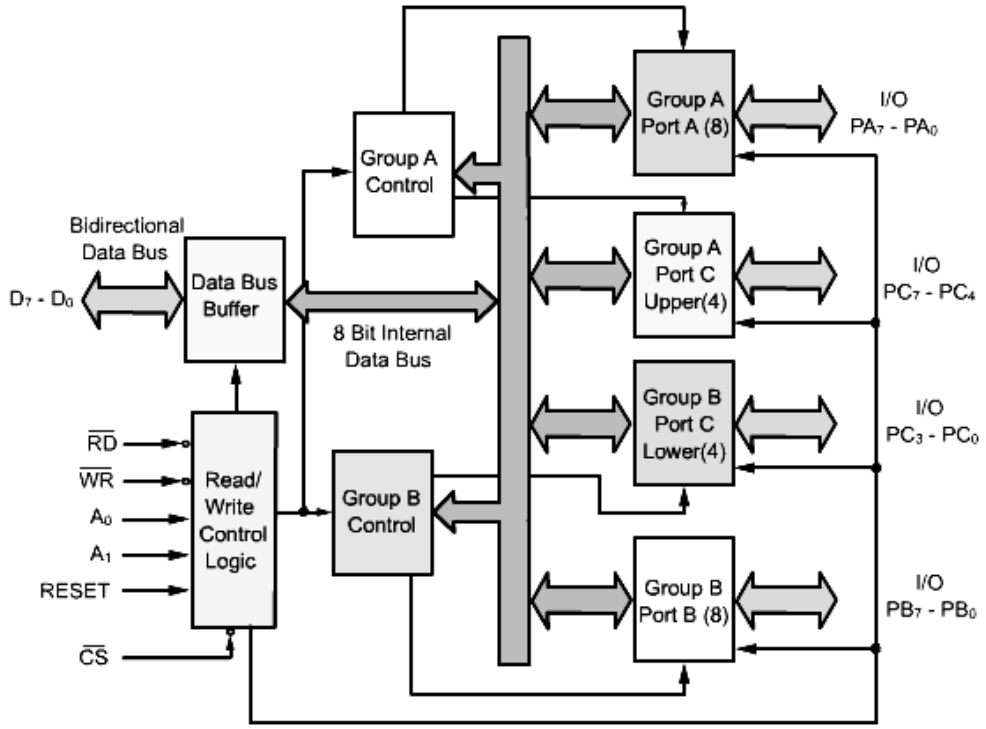
Pin diagram of 8255



Pin symbols	Function
D0-D7(data bus)	Bi-directional, tri-state data bus connected to system data bus used to transfer data and control word from µp(8085) to 8255 or to receive data or status.
PA0-PA7(port A)	8-bit Bi-directional I/O pins used to send data to output device and to receive data from input device.
PB0-PB7(port B)	8-bit Bi-directional I/O pins used to send data to output device and to receive data from input device
PC0-PC7	8-bit Bi-directional I/O pins divided into 2groups PC_L(PC_3-PC_0)and PC_U(PC_7-PC_4)
$^{-}_{rd}$(read)	When the pin is low,thecpu can read the data in the data in the ports or the status word through buffer.
$\overline{\ }_{WR}$(write)	When the input pin is low,thecpu can write the data in the data on the ports or in control register through the data buffer.

$\overline{CS}$(chip select)	This is an active low input which can be enabled for data transfer operation b/w CPU and the 8255.
RESET	This is an active high input used to reset 8255.
A_0 and A_1	These along with RD and WR inputs control selection of the control/status registers or one of the three ports.

Block diagram of 8255:



Data bus transfer:

- This tri-state bi-directional buffer is used to interface the internal data bus of 8255 to the system data us.
- Input or output instructions executed by the CPU either read data from or write data into the buffer.
- Output data from the CPU to the port or control register and input data to the CPU from the ports or status register are all passed through the buffer.

Control logic:

- The control logic block accepts control bus signals as well as inputs from the address bus and issues commands to the individual group control blocks (group A control and group B control).
- It issues appropriate enabling signals to access the required data/control words or status word.
- The read/write control logic requires six control signals. These signals are given below.

1. RD (low): This control signal enables the read operation. When this signal is low, the microprocessor reads data from a selected I/O port of the 8255A.

2. WR (low): This control signal enables the write operation. When this signal goes low, the microprocessor writes into a selected I/O port or the control register.

3. RESET: This is an active high signal. It clears the control register and set all ports in the input mode.

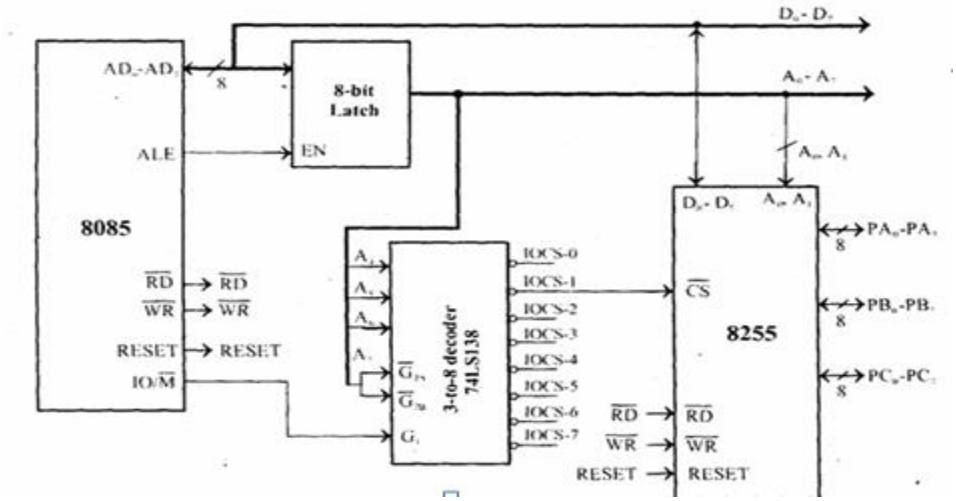
4. CS (low), A0 and A1: These are device select signals. They are,

Internal Devices	A_1	A_0
Port A	0	0
Port B	0	1
Port C	1	0
Control Register	1	1

Group A and group B controls:

- Each of the group A and group B control blocks receives control words from the CPU and issues appropriate commands to the ports associated with it.
- The group A control block controls port A and PC_7 –PC_4 while the group B control block controls port B and (PC_3-PC_0).
- **Port A:**This has an 8-bit latched and buffered output and an 8-bit input latch. it can be programmed in three modes:
    1. Mode-0:simple input/output
    2. Mode-1:input/output with handshake
    3. Mode-2:bi-directional i/o data transfer
- **Port B:** This has an 8-bit data I/O latch/buffer and an 8-bit data input buffer. it can be programmed in mode 0 and mode 1.
- **Port C:** This has one 8-bit unlatched input buffer and an 8-bit output latch/buffer. it can be separated into two parts and each be used as control signals for ports A and B in the handshake mode. it can be programmed for bit set/reset operation.

Interfacing of 8255 with 8085



- The 8255 can be either memory mapped or I/O mapped in the system. In the schematic shown in above is I/O mapped in the system.
- Using a 3-to-8 decoder generates the chip select signals for I/O mapped devices.
- The address lines A4, A5 and A6 are decoded to generate eight chip select signals (IOCS-0 to IOCS-7) and in this, the chip select IOCS- 1 is used to select 8255.
- The address line A7 and the control signal IO/M (low) are used as enable for the decoder.
- The address line A0 of 8085 is connected to A0 of 8255 and A1 of 8085 is connected to A1 of 8255 to provide the internal addresses.
- The data lines D0-D7 are connected to D0-D7 of the processor to achieve parallel data transfer.
- The I/O addresses allotted to the internal devices of 8255 are listed in table.

Internal Device	Binary Address								Hexa Address
	Decoder input and enable				Input to address pins of 8255				
	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
Port-A	0	0	0	1	x	x	0	0	10
Port-B	0	0	0	1	x	x	0	1	11
Port-C	0	0	0	1	x	x	1	0	12
Control Register	0	0	0	1	x	x	1	1	13

Note : *Don't care "x" is considered as zero.*

Operation modes:

The operation of the ports can be classified into two broad groups

- I/O mode and
- Bit set/reset mode.

Bit set-reset (BSR) mode:

- The individual bits of port C can be set or reset by sending out a single OUT instruction to the control register.
- When port C is used for control/status operation, this feature can be used to set or reset individuals bit.

I/O modes:

Mode 0:

- In this mode ports A and B are used as two simple 8-bit I/O ports and ports C as two 4-bit ports.
- Each port can be programmed to function as simply an input port or output port.
- The input/output features in mode 0 are as follows:
    1. Outputs are latched
    2. Inputs are buffered ,not latched
    3. Ports do not have handshake or interrupt capability.

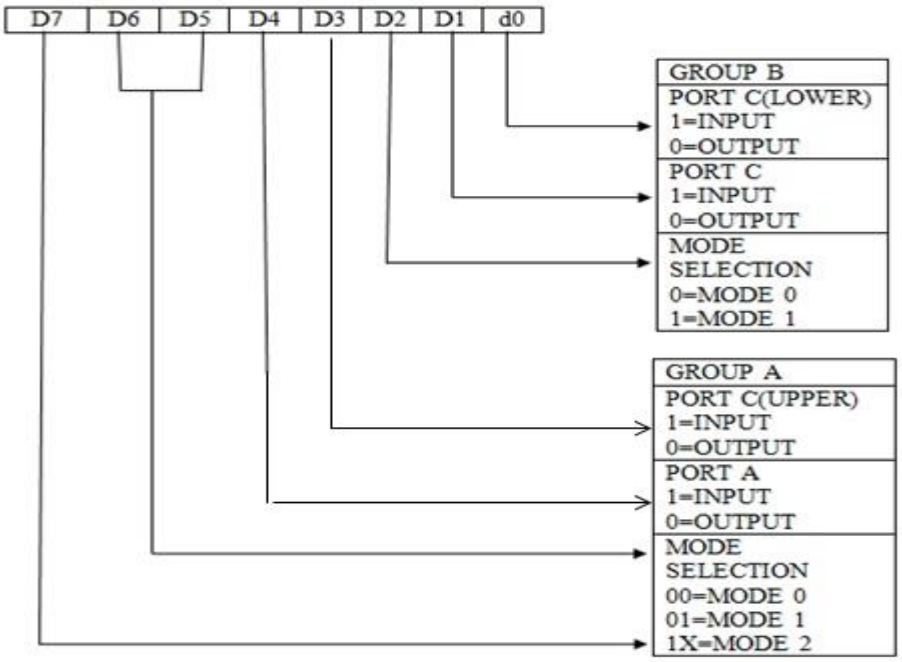
Mode 1:

- Under mode 1 the port A or port B can be set for input or output operation.
- The bits from port C are used as control signals.
- These signals are used for handshaking.
- This supports handshaking has following features.
    1. Port A & b function as 8-bit I/O ports. they can be configured either as inputs or output ports.
    2. Each port uses three lines from port C as handshake signals.
    3. The remaining two lines of port C can be used for simple I/O functions.
    4. Input and output data are latched.
    5. Interrupt logic is supported.

Mode 2:

- In mode 3 the port A is used as a bi-directional port with simultaneous input and output capability.
- Port C is used for handshake.

Control Word:



A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Operation
					INPUT (READ) OPERATION
0	0	0	1	0	Port A to data bus
0	1	0	1	0	Port B to data bus
1	0	0	1	0	Port C to data bus
					Output(write) operation
0	0	1	0	0	Data bus to port A
0	1	1	0	0	Data bus to port B
1	0	1	0	0	Data bus to port C
1	1	1	0	0	Data bus to control register
					Disable function
x	x	x	x	1	Data bus tri -stated
1	1	0	1	0	Illegal condition
x	x	1	1	0	Data bus tri-stated

APPLICATIONS:

Printer operation:

The data transfer operation between a printer and CPU can be carried out as explained below:

1. The CPU places an 8 bit data in the output port.
2. The operation circuit sends out OBF (output buffer full), active low level signal to the data is available for taking inside the printer memory buffer.
3. If the printer buffer is empty enough to receive the data, it will read the data into the pointer buffer and if it is ready to receive next byte of data, then it will acknowledge the output port through the ACK active low signal.
4. The active ACK level is sensed by the interface circuit and an interrupt is raised to inform the CPU to place the next data from the computer memory to the output port.
5. Steps 1, 2, 3 and 4 are replaced.
   This operation is continued till the active ACK signal is received from the printer.

Programmable Timer 8253

- The 8253 solves one of the most common problems in any microcomputer system-the generation of accurate time delays under software control.
- Instead of setting up timing loops in system software, the programmer configures the 8253 with the desired quantity, and then upon command the 8253 will count out the delay and interrupt the CPU when it has completed its tasks.

Introduction:

- The 8253 is a programmable interval timer/counter designed for use with Intel microcomputer systems.
- It is a general purpose, multi-timing element that can be treated as an array of i/o ports in the system software.
- The 8253 solves one of the most common problems in any microcomputer system, the generation of accurate time delays under software control.
- Instead of setting up timing loops in software, the programmer configures the 8253 to match his requirements and programs one of the counters for the desired delay.
- After the desired delay, the 8253 will interrupt the CPU. Software overhead is minimal and variable length delays can easily be accommodated.
- The 8253/54 includes three identical 16 bit counters that can operate independently.
- To operate a counter, a 16 bit count is loaded in its register and, on command; it begins to decrement the count until it reaches 0.
- At the end of the count, it generates a pulse that can be used to interrupt the CPU.
- The counter can count either in binary or BCD.
- In addition, a count can be read by the CPU while the counter is decrementing.

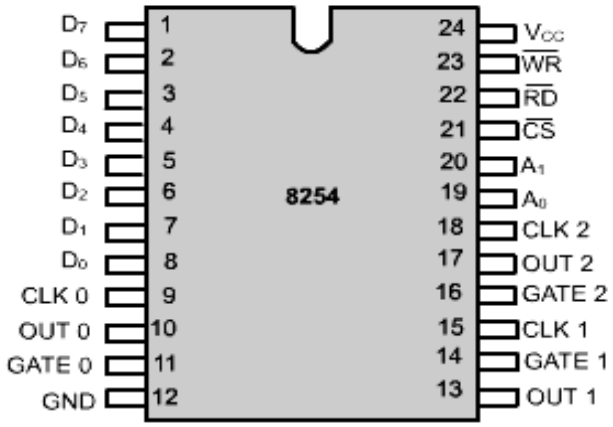
Features:

- Three independent 16-bit down counters.
- 8254 can handle inputs from DC to 10 MHz (5 MHz 8254-5, 8 MHz 8254 10 MHz 8254-2) whereas 8253 can operate upto 2.6 MHz.
- Three counters are identical, presentable and can be programmed for either binary or BCD count.
- Counter can be programmed in six different modes.
- Compatible with all Intel and most other microprocessors.

Data bus buffer:

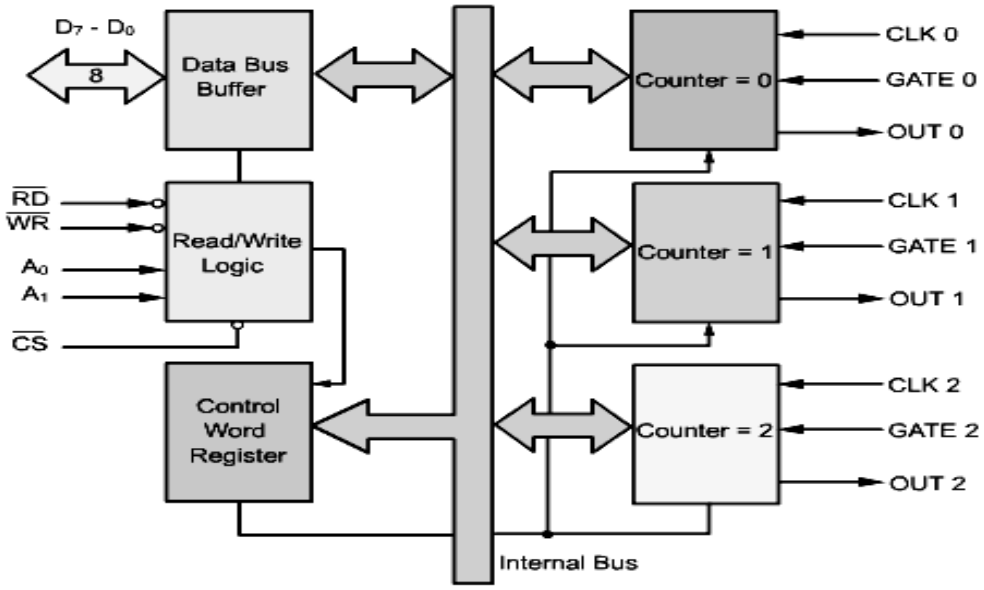
- This tri-state, bi-directional, 8-bit buffer is used to interface the 8253/54 to the system data bus.
- The Data bus buffer has three basic functions.
    1. Programming the 8253/54 in various modes
    2. Loading the count registers.
    3. Reading the count values.

Pin diagram:



D_7 - D_0	Data Bus (Bidirectional)
CLK N	Counter Clock Inputs
GATE N	Counter Gate Inputs
OUT N	Counter Outputs
$\overline{RD}$	Read Counter
$\overline{WR}$	Write Command or Data
$\overline{CS}$	Chip Select
A_0 - A_1	Counter Select
V_{CC}	+ 5 volts
GND	0 Volts

Block Diagram:



Read/Write Logic:

- The Read/Write logic has five signals: ¯R¯D, ¯W¯R, ¯C¯S and the address lines A_0 and A_1.
- In peripheral I/O mode, the RD and WR signals are connected to IOR and IOW, respectively
- In memory-mapped I/O, these are connected to MEMR and MEMW.Address lines A_0 and A_1 of the CPU are usually connected to lines A_0 and A_1 of the CPU are usually connected to lines A_0 and A_1 of the 8253, and CS is tied to a decoded address.
- The control word register and counters are selected according to the signals on lines A_0 and A_1.

Control word register:

- This register is accessed when lines A0 and A1 are at logic 1.
- It is used to a write a command word which specifies the counter to be used (binary or BCD), its mode and either a read or write operation.

A1	A0	SELECTION
0	0	COUNTER 0
0	1	COUNTER 1
1	0	COUNTER 2
1	1	CONTROL WORD REGISTER

Counter:

- These three functional blocks are identical in operation.
- Each counter consists of a single, 16 bit, pre-settable, down counter.
- The counter can operate in either binary or BCD and its input, gate and output are configured by the selection of modes stored in the control word register.
- The counters are fully independent.

- The programmer can read the contents of the three counters without disturbing the actual count in process.

Operational description:

- The complete functional definition of the 8254 is programmed by the system software.
- Once programmed the 8254 is ready to perform whatever timing tasks it is assigned to accomplish.

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

- The register is accessed when lines A0 and A1 are at logic 1.
- It is used to write a command word which specifies the counter to be used.
- Its mode & either a read or write operation.

SC1	SC0	
0	0	Select Counter 0
0	1	Select Counter 1
1	0	Select Counter 2
1	1	Read back command

The bits for D7 and D6 of the control word select anyone of the three counters or the read back command.(SC-select counter D7 to D6). The bits D5 and D4 decide the various read/write operation.

RW1	RW2	
0	0	Counter latch command
0	1	Read/write least significant byte only
1	0	Read/write most significant byte only
1	1	Read/write least significant byte first, then most significant byte

The bits D3, D2 & D1 decide the mode in which the 8254 has to operate the six modes.

M2	M1	M0	
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

M-Mode(D3,D2 &D1)

The bits D0 of the control word decides whether counter should count in BCD or Binary.

BCD	
0	Binary counter 16-bits
1	Binary coded decimal(BCD)counter(4 decades)

8254 Modes of Operation

- Mode 0: Events counter.
- Mode 1: One shot pulse
- Mode 2: Continuous pulses, one clock pulse wide
- Mode 3: Continuous square-wave as long as Gate is 1
- Mode 4: Software triggered one-shot pulse
- Mode 5: Hardware triggered one-shot pulse

Mode definition:

Read and write operations

The 8253 /54 can be programmed to provide various types of output through write operations, or to check a count while counting through read operations. The details of these operations are given below.

Read operations

In some applications especially in event counters, it is necessary to read the values of the count in progress. This can be done by either of two methods.

- One method involves reading a count after inhibiting (stopping) the counter to be read. It is known as **reading by halting the count.**
- The second method involves reading a count while the count is in progress (known as reading on the fly). It is known as **reading while counting**.

In first method, counting is stopped (or inhibited) by controlling the gate input or the clock input of the selected counter, and two I/O read operations are performed by the MPU. The first I/O operation reads the low order byte, and the second I/O operation reads the higher order byte. In the second method, an appropriate control word is written into the control register to latch a count in the output latch, and two I/O read operations are performed by the MPU.

Write operation:

To initialize a counter, the following steps are necessary.

- Write a control word into the control register.
- Load the low order byte of a count in the counter register.
- Load the high order byte of a count in the counter register.

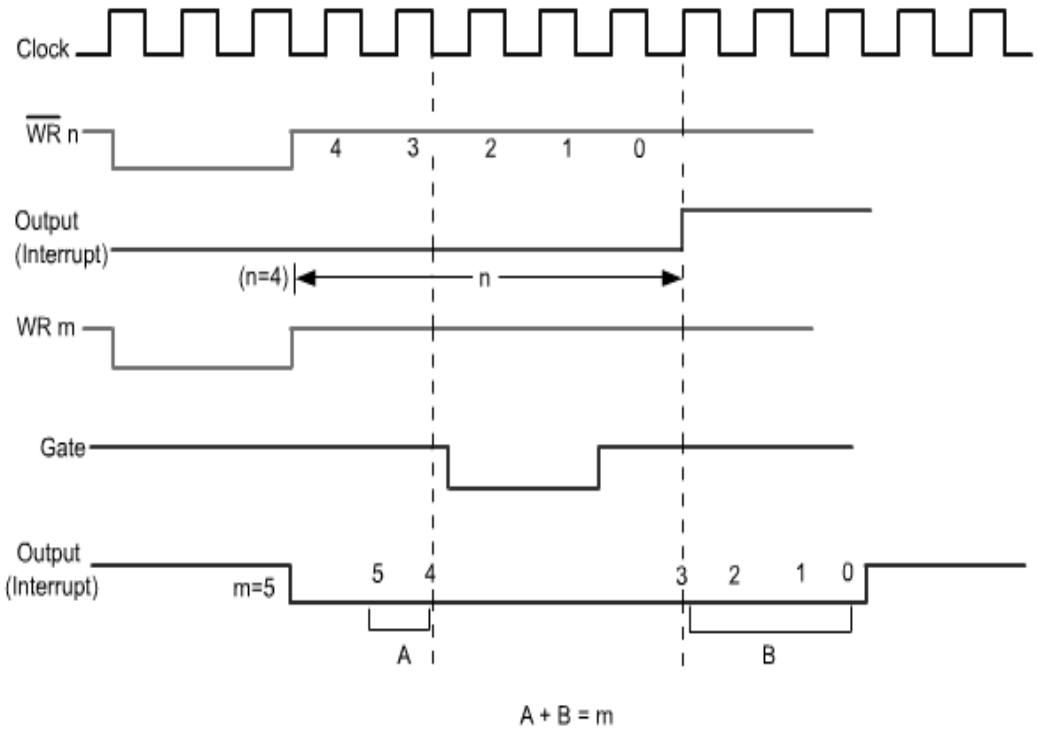
With a clock and an appropriate gate signal to one of the counters, the above steps should start the counter and provide appropriate output according to the control word.

Modes of operation:

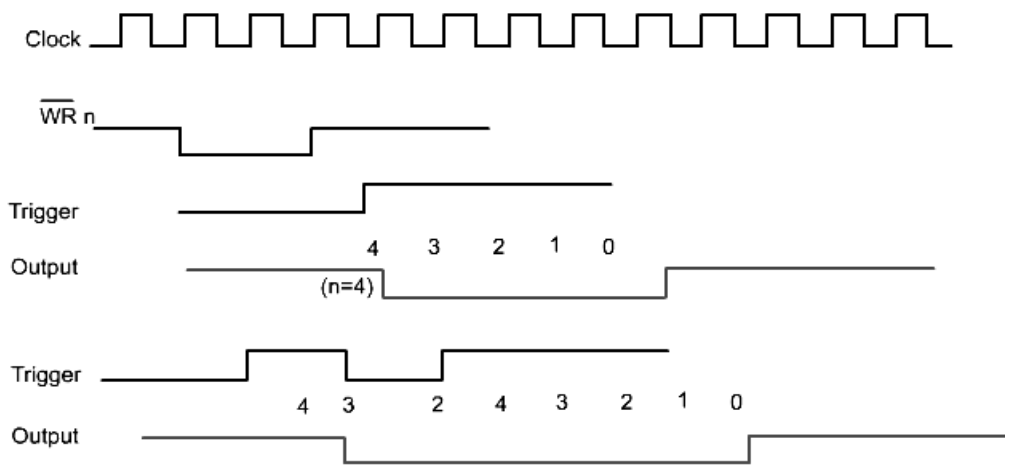
As mentioned earlier, the 8254 can operate in six different modes; we will describe briefly various modes of the 8254.

MODE 0: Interrupt on terminal count

- The output goes low on setting the mode, and goes high after the desired count.
- In this mode, initially the OUT is low.
- Once count is loaded in the register, the counter is decremented every cycle, and when the count reaches zero, the OUT goes high.
- This can be used as an interrupt.
- The OUT remains high until a new count or a command word is loaded.
- Figure shows that the counting (m=s) is temporarily stopped when the gate is disabled (G=0), and continued again when the gate is at login 1.



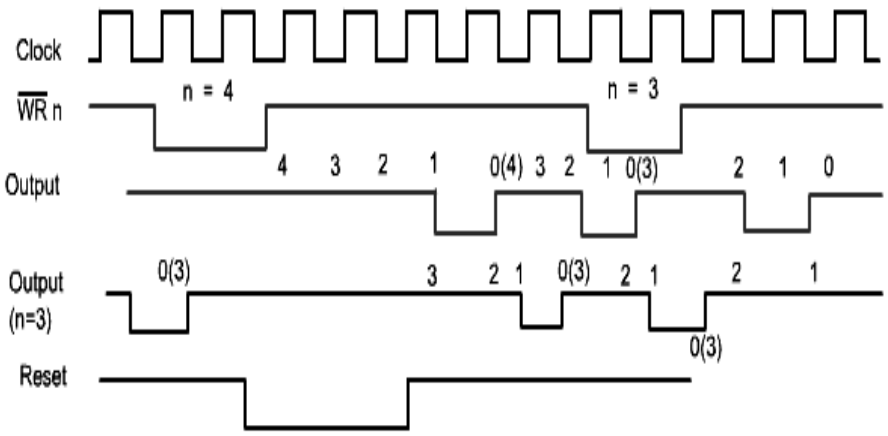
MODE 1: Hardware-Retriggerable One-Shot



- The output goes low on a gate input, and goes high on terminal count.
- In this mode, the OUT is initially high.
- When the gate is triggered, the out goes low, and at the end of the count, the OUT goes high again, thus generating a one-shot pulse.

MODE 2: Rate generator

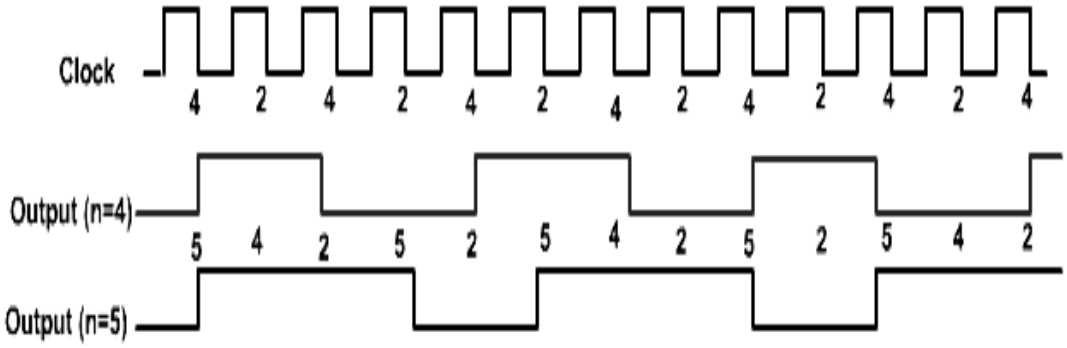
- It is equivalent to a 'clock pulse divide by n' counter.
- The output pulse frequency is 1/n of the input pulse frequency, where n is the count number.



- This mode is used to generate pulse equal to the clock period at a given interval
- When a count is loaded, the OUT stays high until the count reaches 1, and then the OUT goes low for one clock Period.
- The count is reloaded automatically, and the pulse is generated continuously.
- The count = 1 is illegal in this mode.

MODE 3: Square wave generator

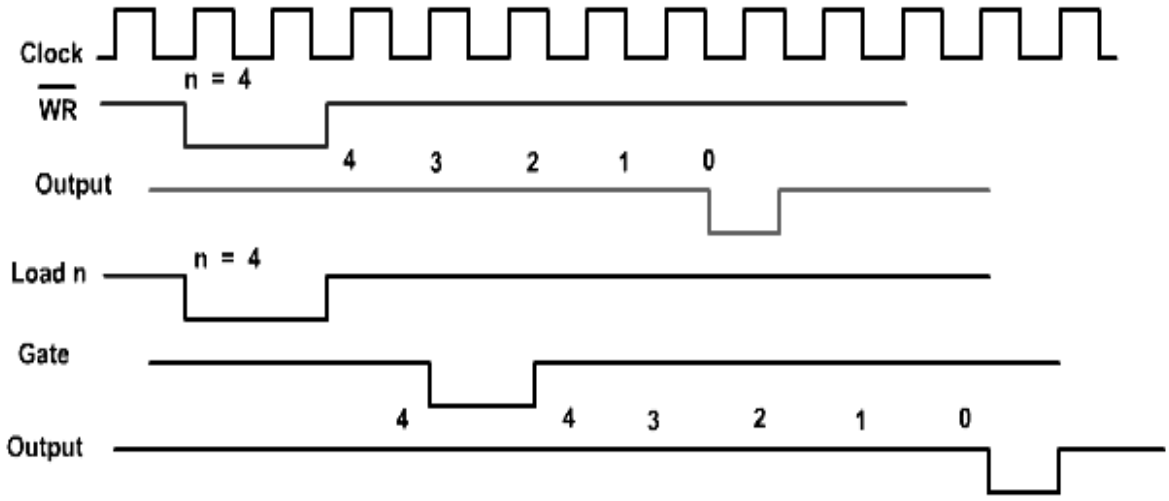
- This is similar to mode 2 with difference in minor operating details.
- In this mode, when a count is loaded, the OUT is high.
- The count is decremented by two at every clock cycle, and when it reaches zero, the OUT goes low, and the count is reloaded again.



- This is repeated continuously; thus, a continuous square wave with period equal to the period of the count is generated.
- In other word, the frequency of the square wave is equal to the frequency of the clock divided by the count.
- If the count (N) is odd the pulse stays high for (N+1)/2 clock cycles and stays low for (N-1)/2 clock cycles.

MODE 4: Software-triggered strobe

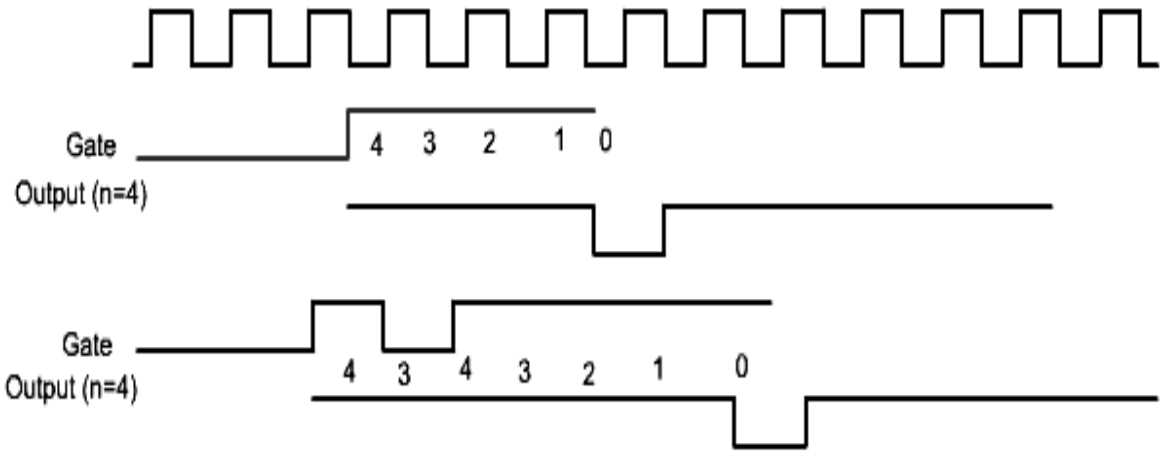
- The output goes high on setting the mode
- After terminal count, the output goes low for one clock period and then goes high again



- In this mode, the OUT is initially high; it goes low for one clock period at the end of the count.
- The count must be reloaded for subsequent outputs.

MODE 5: Hardware-triggered strobe

- This is similar to mode 4, but a trigger at the gate initiates the counting.
- This mode is similar to mode 4, except that it is triggered by the rising pulse at the gate.



- Initially, the OUT is high, and when the gate pulse is triggered from low to high, the count begins.
- At the end of the count, the OUT goes low for one clock period.

8259 is a Programmable Interrupt Controller

FEATURES OF 8259:

a. It is programmed to work with either 8085 or 8086 processor.
b. It manages 8-interrupts according to the instructions written into its control registers.
c. In 8086 processor, it supplies the type number of the interrupt and the type number is programmable. In 8085 processor, the interrupt vector address is programmable. The priorities of the interrupts are programmable.
d. The interrupts can be masked or unmasked individually.
e. The 8259s can be cascaded to accept a maximum of 64 interrupts.

FUNCTIONAL BLOCK DIAGRAM OF 8259:

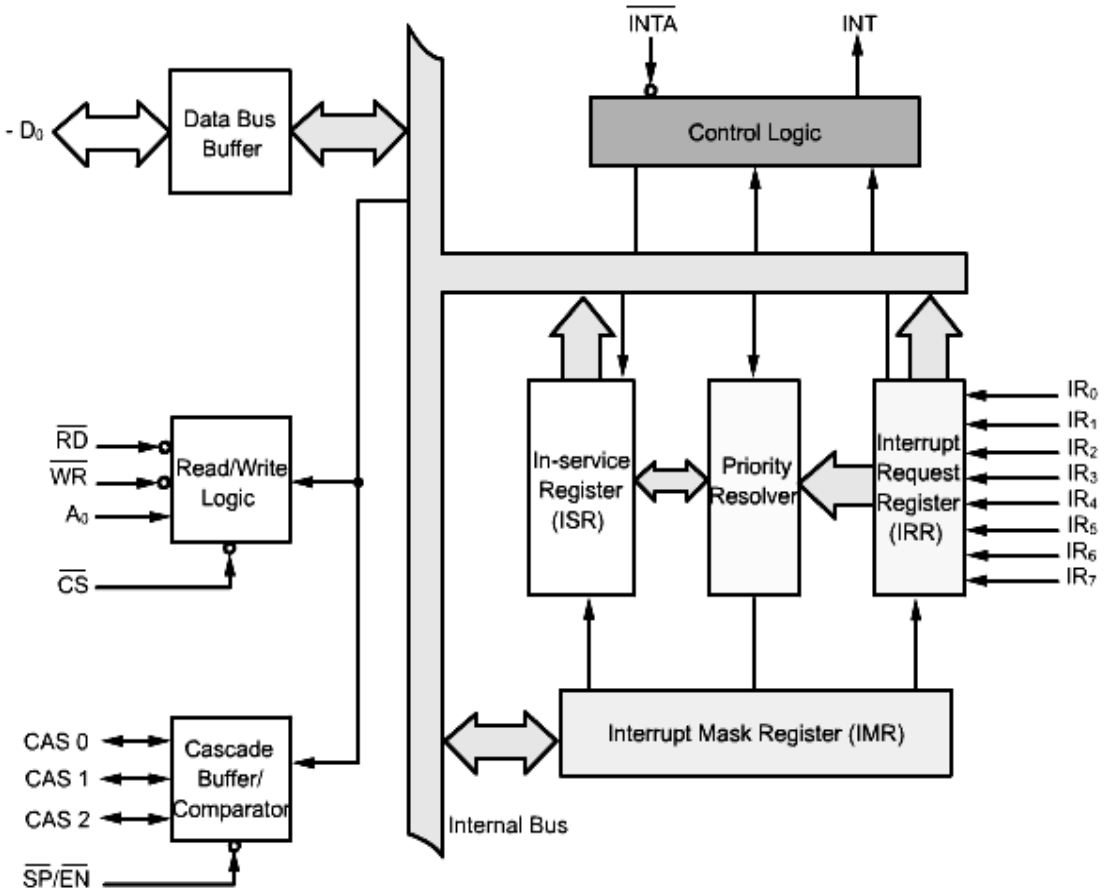
It has eight functional blocks. They are,

1. Control logic
2. Read Write logic
3. Data bus buffer
4. Interrupt Request Register (IRR)
5. In-Service Register (ISR)
6. Interrupt Mask Register (IMR)
7. Priority Resolver (PR)
8. Cascade buffer.

Functional block diagram of 8259

The data bus and its buffer are used for the following activities.

1. The processor sends control word to data bus buffer through D0-D7.
2. The processor read status word from data bus buffer through D0-D7
3. From the data bus buffer the 8259 send type number (in case of 8086) or the call opcode and address (in case of 8085) through D0-D7 to the processor.



Functional block diagram

- The processor uses the RD (low), WR (low) and A0 to read or write 8259.
- The 8259 is selected by CS (low).
- The IRR has eight input lines (IR0-IR7) for interrupts. When these lines go high, the request is stored in IRR. It registers a request only if the interrupt is unmasked.
- Normally IR0 has highest priority and IR7 has the lowest priority. The priorities of the interrupt request input are also programmable.
- First the 8259 should be programmed by sending Initialization Command Word (ICW) and Operational Command Word (OCW). These command words will inform 8259 about the following,

i. Type of interrupt signal (Level triggered / Edge triggered).
ii. Type of processor (8085/8086).
iii. Call address and its interval (4 or 8)

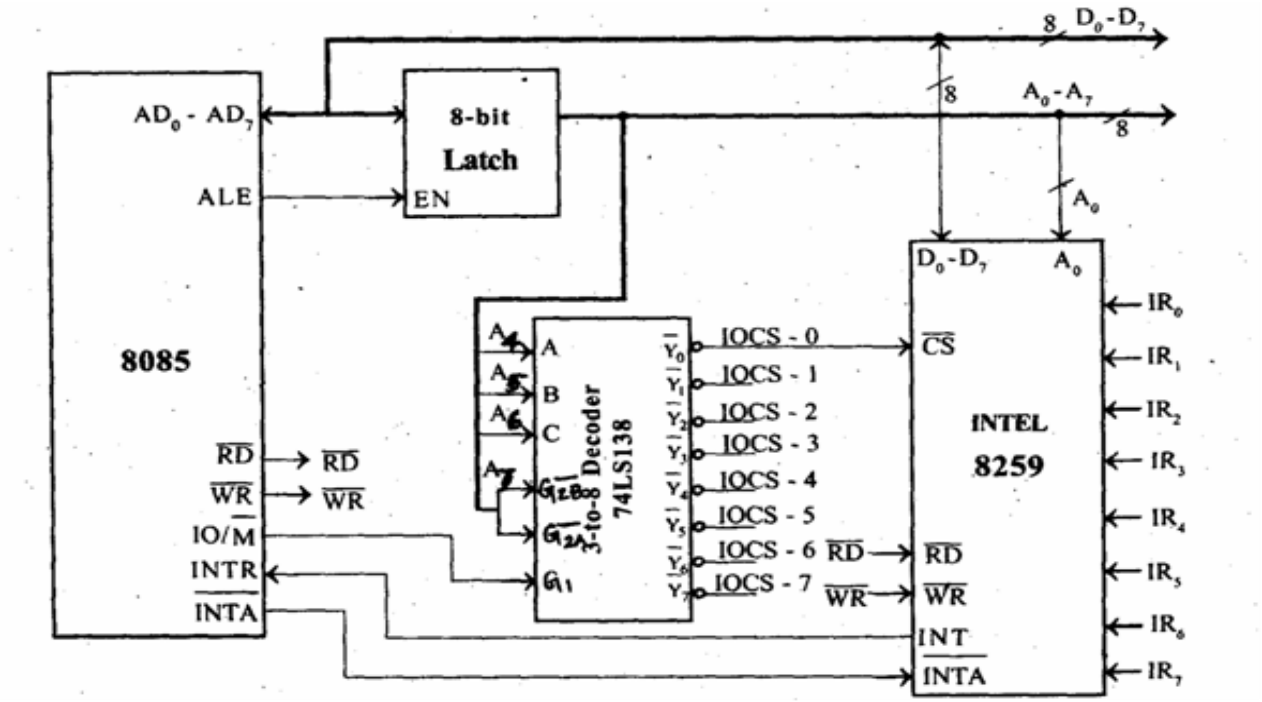
iv. Masking of interrupts.
v. Priority of interrupts.
vi. Type of end of interrupts.

- The interrupt mask register (IMR) stores the masking bits of the interrupt lines to be masked. The relevant information is send by the processor through OCW.
- The in-service register keeps track of which interrupt is currently being serviced.
- The priority resolver examines the interrupt request, mask and in-service registers and determines whether INT signal should be sent to the processor or not.
- The cascade buffer/comparator is used to expand the interrupts of 8259.
- In cascade connection one 8259 will be directly interrupting 8086 and it is called master 8259.
- To each interrupt request input of master 8259 (IR0-IR7), one slave 8259 can be connected. The 8259s interrupting the master 8259 are called slave 8259s.
- Each 8259 has its own addresses so that each 8259 can be programmed independently by sending command words and independently the status bytes can be read from it.

CASCADING 8259:

- The cascade pins (CAS0, CAS1 and CAS2) from the master are connected to the corresponding pins of the slave.
- For the slave 8259, the SP (low) / EN (low) pin is tied low to let the device know that it is a slave.
- The SP (low) / EN (low) pin can be used as input or output signal.
- In non-buffered mode it is used as input signal and tied to logic-I in master 8259 and logic-0 in slave 8259.
- In buffered mode it is used as output signal to disable the data buffers while data is transferred from 8259A to the CPU.

INTERFACING 8259 WITH 8085 MICROPROCESSOR



- It requires two internal address and they are A =0 or A = 1.
- It can be either memory mapped or I/O mapped in the system. The interfacing of 8259 to 8085 is shown in figure is I/O mapped in the system.
- The low order data bus lines D0-D7 are connected to D0-D7 of 8259.
- The address line A0 of the 8085 processor is connected to A0 of 8259 to provide the internal address.
- The 8259 require one chip select signal. Using 3-to-8 decoder generates the chip select signal for 8259.
- The address lines A4, A5 and A6 are used as input to decoder.
- The control signal IO/M (low) is used as logic high enables for decoder and the address line A7 is used as logic low enable for decoder.

The I/O addresses of 8259 are shown in table

	Binary Address								Hexa address
	Decoder input/ enable			Input to address pin of 8259					
	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
For A_0 of 8259 to be zero	0	0	0	0	x	x	x	0	00
For A_0 of 8259 to be one	0	0	0	0	x	x	x	1	01

Note : *Don't care "x" is considered as zero.*

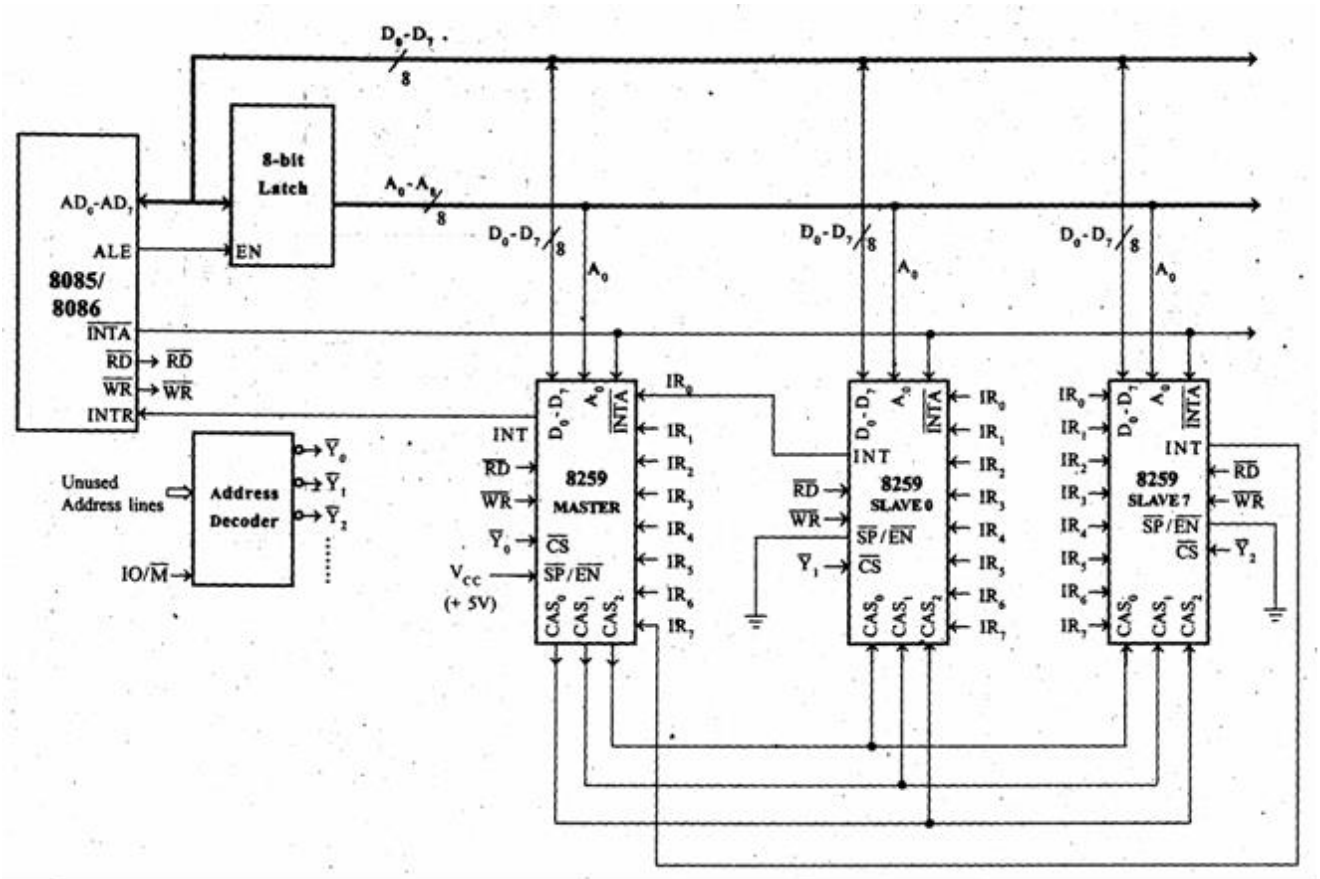


Fig - Cascade Connection of 8259

Working of 8259 with 8085 processor:

First the 8259 should be programmed by sending Initialization Command Word (ICW) and Operational Command Word (OCW). These command words will inform 8259 about the following,

i. Type of interrupt signal (Level triggered / Edge triggered).
ii. Type of processor (8085/8086).
iii. Call address and its interval (4 or 8)
iv. Masking of interrupts.
v. Priority of interrupts.
vi. Type of end of interrupts.

- Once 8259 is programmed it is ready for accepting interrupt signal. When it receives an interrupt through any one of the interrupt lines IR0-IR7 it checks for its priority and also checks whether it is masked or not.
- If the previous interrupt is completed and if the current request has highest priority and unmasked, then it is serviced.
- For servicing this interrupt the 8259 will send INT signal to INTR pin of 8085.

- In response it expects an acknowledge INTA (low) from the processor.
- When the processor accepts the interrupt, it sends three INTA (low) one by one.
- In response to first, second and third INTA (low) signals, the 8259 will supply CALL opcode, low byte of call address and high byte of call address respectively. Once the processor receives the call opcode and its address, it saves the content of program counter (PC) in stack and load the CALL address in PC and start executing the interrupt service routine stored in this call address.

8259 has four numbers of Initialization Command Word (ICW) and three numbers of Operational Command Word(OCW).

The command words are sent to the 8259 by selecting it by CS=0 and A_o=0 or 1. Certain command word are sent to the internal address, A_o =0and others with A_o=1.

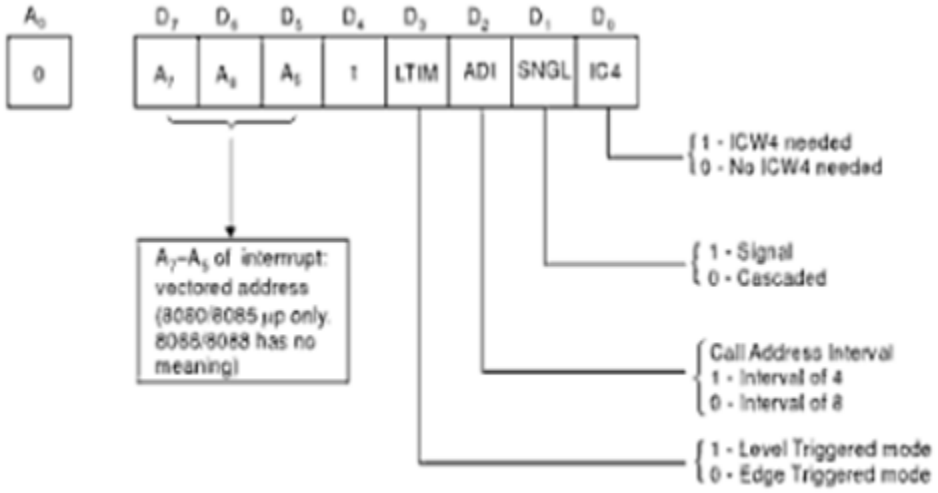
ICWs are used to program the following features of an 8259:

- Call address interval in case of an 8085
- Level or edge triggered
- Cascade mode or single
- Vector addresses or type number
- 8085 mode
- Auto or normal end of interrupt
- Specific fully nested mode

Initialization Command Word 1 (ICW1)

Whenever a command is received with $A_0 = 0$ and $D_4 = 1$, this is interpreted by 8259A as an initialization sequence during which the following events automatically occur.

(i) The edge sense circuit is reset, which means that following the initialization, an interrupt request (IR) input must make a low-to-high transition to generate an interrupt.
(ii) The interrupt mask register is cleared.
(iii) IR_7 input is assigned priority 7.
(iv) The slave mode address is set to 7.
(v) The special mask mode is cleared and status read is set to IRR.
(vi) If D_0-bit of ICW1(Fig. -) is 0, then all functions selected in ICW4 are set to zero.

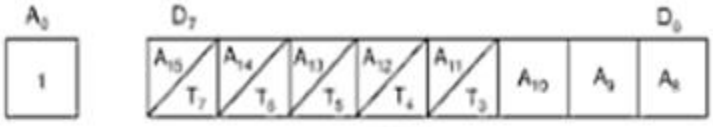


ICW1.

Note: For an 16-bit microprocessor (8086/8088) it is necessary that the call interval is 4.

Initialization Command Word 2 (ICW2)

$(A_8$–$A_{15})$ of interrupt vector address is for 8080/8085 microprocessor
T_3–T_7 vector type for 8088/8086 microprocessor.

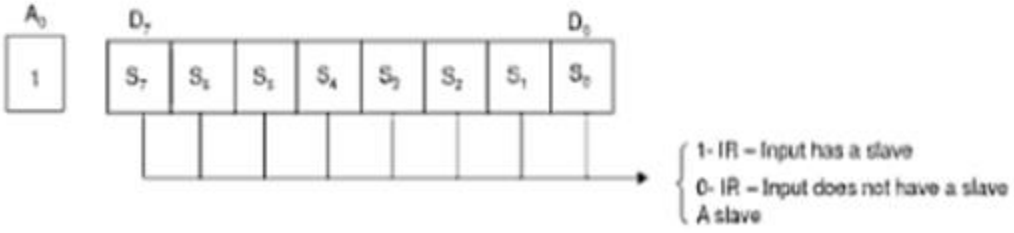


ICW2.

Initialization Command Word 3 (ICW3)

The word is used only when there are more than one 8259A's in the system and cascading is used, in which case SNGL = 0. This command word is loaded into the 8-bit slave register. The functions of this register are:

(i) In the master mode (either when SP = 1 in the buffer mode or M/S = 1 in ICW4) 1 is set for each slave in the system. The master then will release (for 8088/8086 only byte 2) the address of the slave 8259A through the cascade lines.

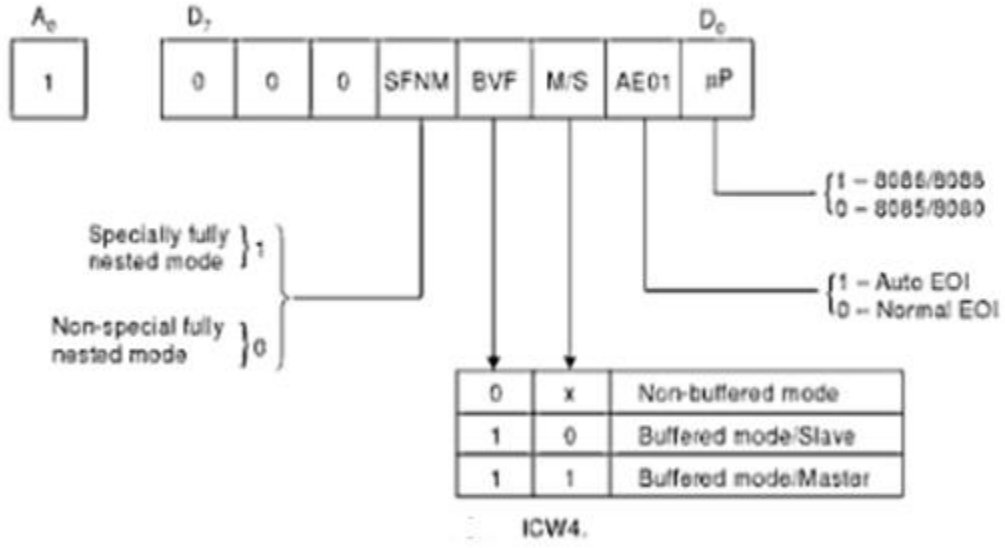


ICW3 for Master.

Initialization Command Word 4 (ICW4)

If AEOI = 1, then the 8259A will operate in automatic end of interrupt mode. In this mode the 8259A will automatically perform a non-specific EOI operation (i.e., the highest priority interrupt service register is closed) at the trailing edge of the final interrupt acknowledge pulse.

Note: From a system stand point, this mode should be used only when a nested multi-level interrupt structure is not required within a single 8259A. The AEOI mode can only be used in a master 8259A and not in a slave.



Non-buffered Mode

In this mode if the 8259A is the master, it has to have the $\overline{SP}/\overline{EN}$ pin high and if it is in the slave mode, the $\overline{SP}/\overline{EN}$ pin must be pulled low.

OCWs are used to read the status of the interrupts and also to program the following features of a 8259:

- Masking or unmasking of individual interrupts
- Specific od non-specific end of interrupt
- Priority modes

USART-8251A

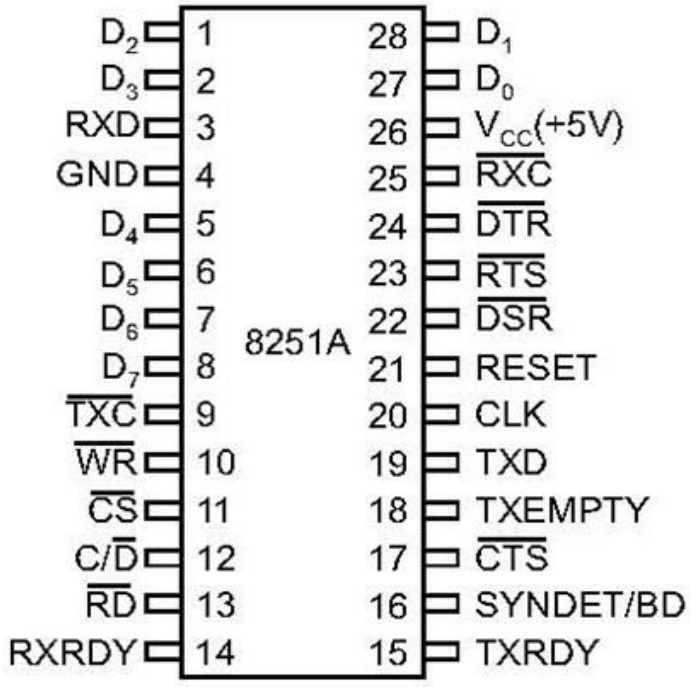
- To transmit byte data it is necessary to convert into eight serial bits. This can be done by using the parallel to serial converter.
- Similarly at the reception the serial bits must be converted into parallel 8 bit data. The serial to parallel converter is used to convert serial data bits into the parallel data.
- The devices are designed for these purposes are called Universal Asynchronous Receivers- Transmitter.

- A good example of UART is 8250 and USART is 8251. These devices are software programmable for numbers of data bits parity and number of stop bits.

Features:

- The Intel 8251A is an universal synchronous and asynchronous communication controller.
- It supports standard asynchronous protocol with :
- 5 to 8 Bit character format
- Odd, even or not parity generation and detection.
- Baud rate from DC to 19.2 Kbaud
- False start bit detection.
- Automatic break detect and handling.
- Break character generation
- It has built in baud rate generator
- It supports standard synchronous protocol with:
- 5 to 8 Bit character format
- Internal or external character synchronization
- Automatic sync insertion
- Baud rate from DC to 64 kbaud
- It allows full duplex transmission and reception
- It provides double buffering of data both in the transmission section and in the receiver section.
- It provides error detection logic, which detects parity, overrun and in the receiver section.
- It has Modern Control Logic, which support basic data set control signals.
- It provides separate clock inputs for receiver and transmitter section, thus providing an option of fixing different baud rates for the transmitter and receiver section.
- It is compatible with an extended range of Intel microprocessors
- It is fabricated in 28 pins DIP package and its all inputs and outputs are TTL compatible.
- It is available in standard as well as extended temperature range.

PIN DIAGRAM OF 8251A



Data Bus: Bi-directional, tri-state, 8-bit Data Bus. This pin allow transfer of bytes between the CPU and the 8251A

RD (Read): A low on this input allows the CPU to read data or status bytes from 8251A

$\overline{WR}$ (Write): A low on this input allows the CPU to write data or command word to the from 8251A

CLK (Clock): The CLK input is used to generate internal device timing. The frequency of CLK must be greater than 30 times the receiver or transmitter data bit rates.

RESET: A high on this input forces the 8251A into an "Idle" mode. The device will remain at "Idle" until a new set of control words is written into the 8251A to program its functional definition.

C/D (Control/Data): This input in conjuction with the WR and RD inputs, informs the 8251A that the word on the Data Bus is either a data character, control word or status information.

CS (Chip Select): A low on this input allows communication between CPU and 8251A

C/D	RD	WR	OPERATION
0	0	1	CPU reads data from USART
0	1	0	CPU writes data to USART
1	0	1	CPU reads status from USART
1	1	0	CPU writes command to USART
X	1	1	USART Bus floating

MODERN CONTROL SIGNALS:

The 8251A has a set of control inputs and outputs that can be used to simplify the interface to almost any modem.

$\overline{\text{DSR}}$ (Data Set Ready): This input signal is used to test modem conditions such as data set ready.
DTR (Data Terminal Ready): This output signal is used to tell modem that data terminal is ready.
RTS (Request To Send): This output signal is asserted to begin transmission.
CTS (CLEAR TO SEND): A low on this input enables the 8251A transmit serial data if the TxE bit in the command byte is set to a "one".

TRANSMITTER SIGNALS:

TxD: Transmit Data: This output signal outputs a composite serial stream of data on the falling edge of TxC
TxRDY (Transmitter Ready) : This output signal indicates the CPU that the transmitter is ready to accept the data character
TxE (Transmitter Empty*):*This output signal indicates that the transmitter has no character to transmit.
TxC (Transmitter Clock): This clock input controls the rate at which the character is to be transmitted
RECEIVER SIGNALS:
RxD (Receiver Data): This input receives a composite serial stream of data on the rising edge of RxC
RxRDY (Receiver Ready): This output indicates that the 8251A contains a character that is ready to be input to the CPU
RxC (Receiver Clock): This clock input controls the rate at which the character is to be received.

SYNDET (Sync Detect) / BRKDET (Break detect)

- This pin is used in synchronous mode for detection of synchronous characters and may be used as either input or output
- In asynchronous mode this pin goes high if receiver line stays low for more than 2 character times.
- It then indicates a break in the data stream.
- When used as an input (external sync detect mode) a positive signal will cause the 8251A to start receiving data characters on the rising edge of the next RXC

Block diagram of 8251 USART

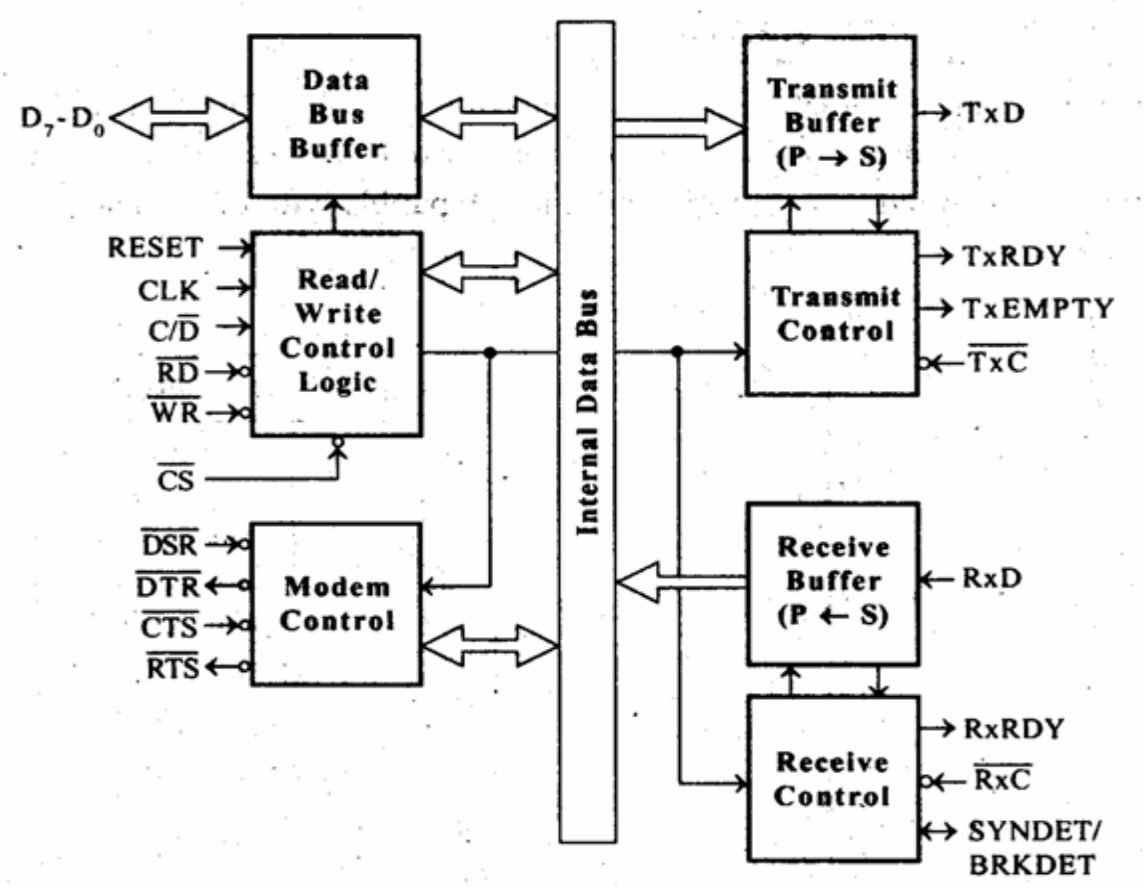
Data Bus Buffer

- This tri-state, bi-directional, 8-bit buffer is used to interface 8251 to the system data bus.

- Along with the data, control word, command words and status information are also transferred through the Data Bus Buffer.

Read/ write control logic:

- This functional block accepts input from the system control bus and generates control signal for overall device operation
- It decodes control signals on the 8085 control bus into signal which controls the internal and external I/O bus.
- It contains the control word register and command word register that stores the various control formats for the device functional definition



Transmit Buffer

- The transmit buffer accepts parallel data from the CPU, adds the appropriate framing information, serializes it, and transmits it on the TxD pin on the falling edge of $\overline{TxC}$
- It has two registers
- A buffer register to hold eight bits and an output register to convert eight bits into a stream of serial bits

- The CPU writes a bit in the buffer register, which is transferred to the output register when it is empty.
- The output register then transmits serial data on the TxD pin
- In the asynchronous mode the transmitter always adds START bit; depending on how the unit is programmed, it also adds an optional even or odd parity bit, and either 1, 1 ½ , or 2 STOP bits.
- In synchronous mode no extra bits (other than parity, if enable) are generated by the transmitter.

Transmit control

- It manages all activities associated with the transmission of serial data
- it accepts and issues signals both externally and internally to accomplish this functions

TxRDY (Transmit Ready)

- This output signal indicates CPU that buffer register is empty and the USART is ready to accept a data character
- It can be used as an interrupt to the system or, for polled operation, the CPU can check TxRDY using the status read operation
- This signal is reset when a data bit is loaded into the buffer register

TxE (Transmitter Empty*)*

- This is an output signal
- A high on this line indicates that the output buffer is empty
- In the synchronous mode, if the CPU has failed to load a new character in time, TxE will go high momentarily as SYN characters are loaded into the transmitter to fill the gap in transmission

TxC (Transmitter Clock)

- This clock controls the rate at which characters are transmitted by USART
- In the synchronous mode TxC is equivalent to the baud rate, and is supplied by modem
- In the asynchronous mode $\overline{\text{TxC}}$ is 1, 16,or 64 times baud rate
- The clock division is programmable
- It can be programmed by writing proper mode word in the mode set register

Receiver Buffer

- The receiver accepts serial data on the RxD line, converts this serial data to parallel format, check for bits or characters that are unique to the communication technique and sends an "assembled" character to the CPU
- When 8251A is in the asynchronous mode and it is ready to accept a character, it looks for a low level RxD line.
- When it receives the low level, it assumes that it is a START bit and enables an internal counter.
- At a count equivalent to one-half of a bit time, the RxD lines sampled again

- If the line is still low a valid START bit is detected and the 8251A proceeds to assemble the character
- After successful reception of a START bit the 8251A receives data, parity and STOP bits, and then transfers the data on the receivers input register
- The data is then transferred into the receiver buffer register
- In the synchronous mode the receiver simply receives the specified number of data bits and transfers them to the receiver input register and then to the receiver buffer register

Receiver Buffer

- It manages all receiver-related activities.
- Along with data reception , it does false start bit detection, parity error detection, framing error detection, sync detection and break detection

RxRDY (Receiver Ready)

- This is an output signal
- It goes high(active) when the USART has a character in the buffer register and is ready to transfer it to the CPU
- This line can be used either to indicate the status in the status register or to interrupt the CPU
- This signal is reset when a data bit from receiver buffer is read by the CPU

RxC (Receiver Clock)

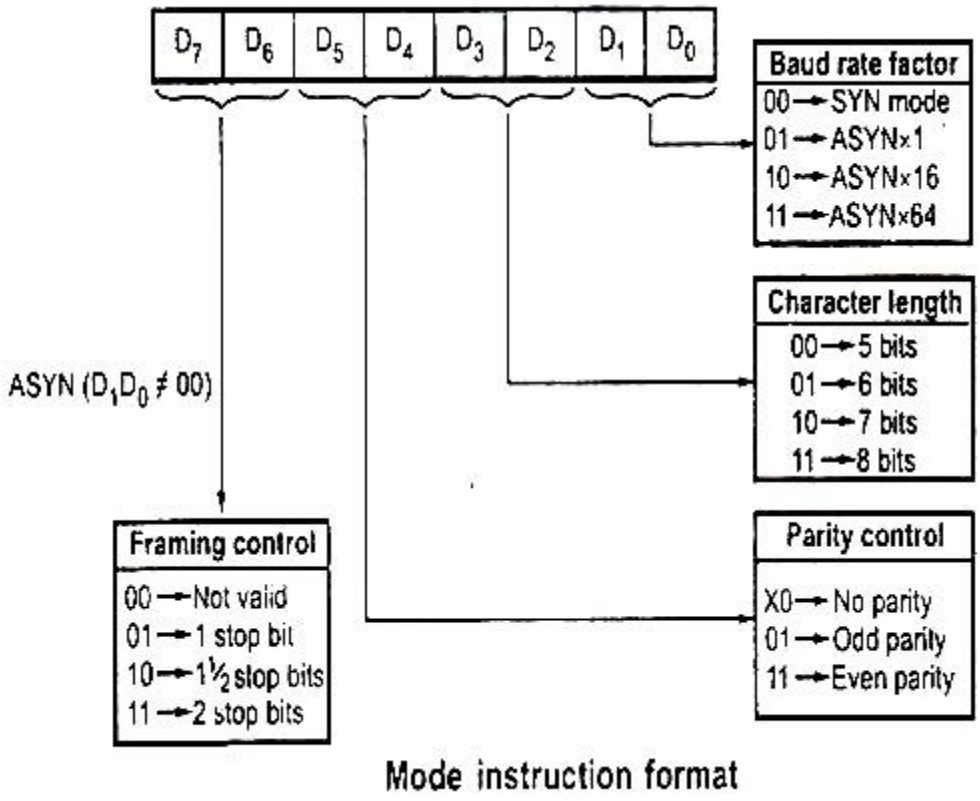
- This clock controls the rate at which characters is to be received by USART
- In the synchronous mode $\overline{\text{RxC}}$ is equivalent to the baud rate, and is supplied by modem
- In the asynchronous mode $\overline{\text{RxC}}$ is 1, 16,or 64 times baud rate
- The clock division is programmable
- It can be programmed by writing proper mode word in the mode set register

Modem control

- The 8251 has a set of control inputs and outputs that can be used to simplify the interface to almost any modem
- It provides control circuitry for the generation of $\overline{\text{RTS}}$ and $\overline{\text{DTR}}$ and the reception of ~~CTS~~ and ~~DSR~~
- A general purpose inverted output and general purpose input are provided
- The output is labeled $\overline{\text{DTR}}$ and the input is labeled $\overline{\text{DSR}}$
- $\overline{\text{DTR}}$ can be asserted by setting bit 2 of the command instruction;
- DSR can be sensed as bit 7 of the status register
- When used as a modem control signal $\overline{\text{DTR}}$ indicates that the terminal is ready to communicate and ~~DSR~~ indicates that it is ready for communication

8251A CONTROL WORDS

- The control words defines the complete functional definition of 8251A and they must be loaded before transmission or reception
- The control words of 8251A are split into two formats:
  1. Mode instruction
  2. Command instruction
- The instruction can be considered as four 2 –bit fields
- The first 2-bit field (D_1-D_0) determines whether the USART is to operate in the synchronous (00) or asynchronous mode.
- In asynchronous mode, this fields determines the division factor for clock to decide the baud rate.
- For example, if D_1 and D_0 are both ones, the RxC and TxC will be divided by 64 to establish the baud rate.
- The second 2-bit field (D_3-D_2) determines number of data bits in one character.
- With this 2-bit field we can set character length from 5-bits to 8-bits

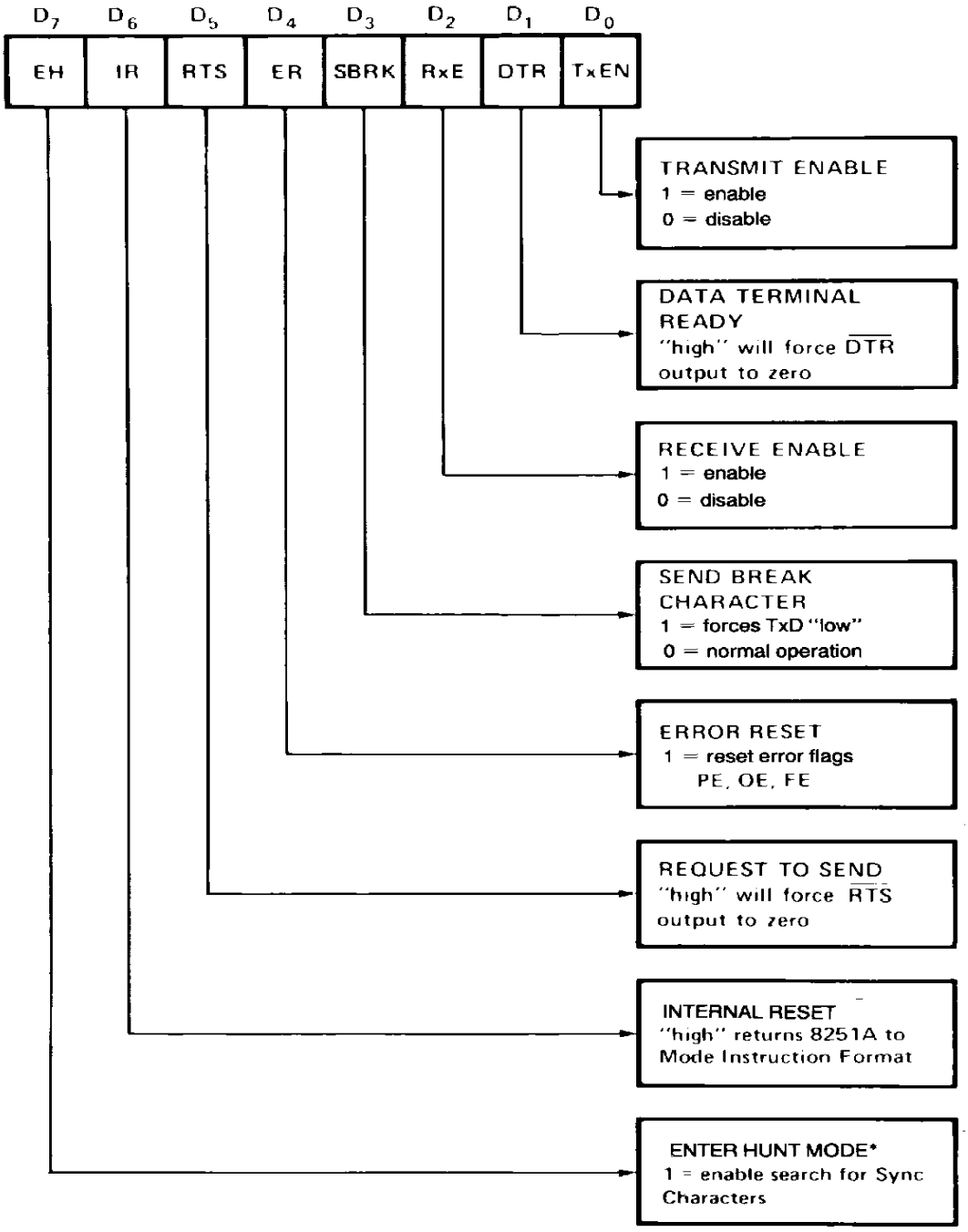


Mode instruction format

- The third 2-bit field, (D_5 _ D_4) controls the parity generation. The parity bit is added to the data bits only if parity is enabled.
- The last field, (D_7 _ D_6), has two meanings depending on whether operation is to be in the synchronous or asynchronous mode, (i.e. D_1 D_0 not equal to 00
- It controls the number of STOP bits to be transmitted with the character

- In synchronous mode, (D_1 D_0= 00) this field controls the synchronizing process.
- It decides whether to operate with external synchronization or internal synchronization and whether to transmit single synchronizing character or two synchronizing characters

COMMAND INSTRUCTION

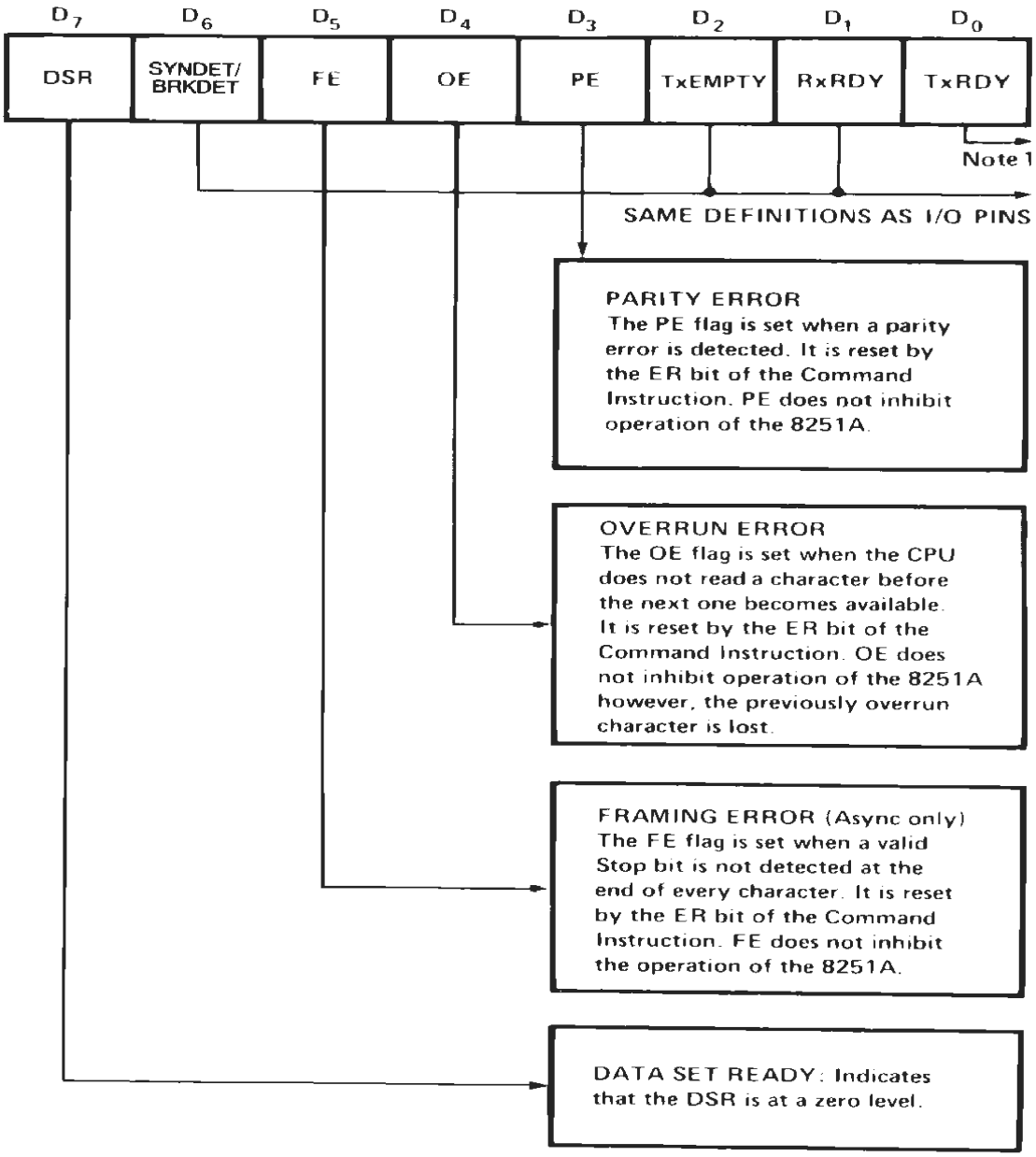


- After the mode instruction, command character should be issued to the USART.

- It controls the operation of the USART within the basic frame work established by the mode instruction
- It does function such as: enable transmit /receive, error reset and modem control

8251A STATUS WORDS

- In the data communication systems it is often necessary to examine the "status" of the transmitter and receiver
- It is also necessary for CPU to know if any error has occurred during communication
- The 8251A allow the programmer to read above mentioned information from the status register any time during the functional operation

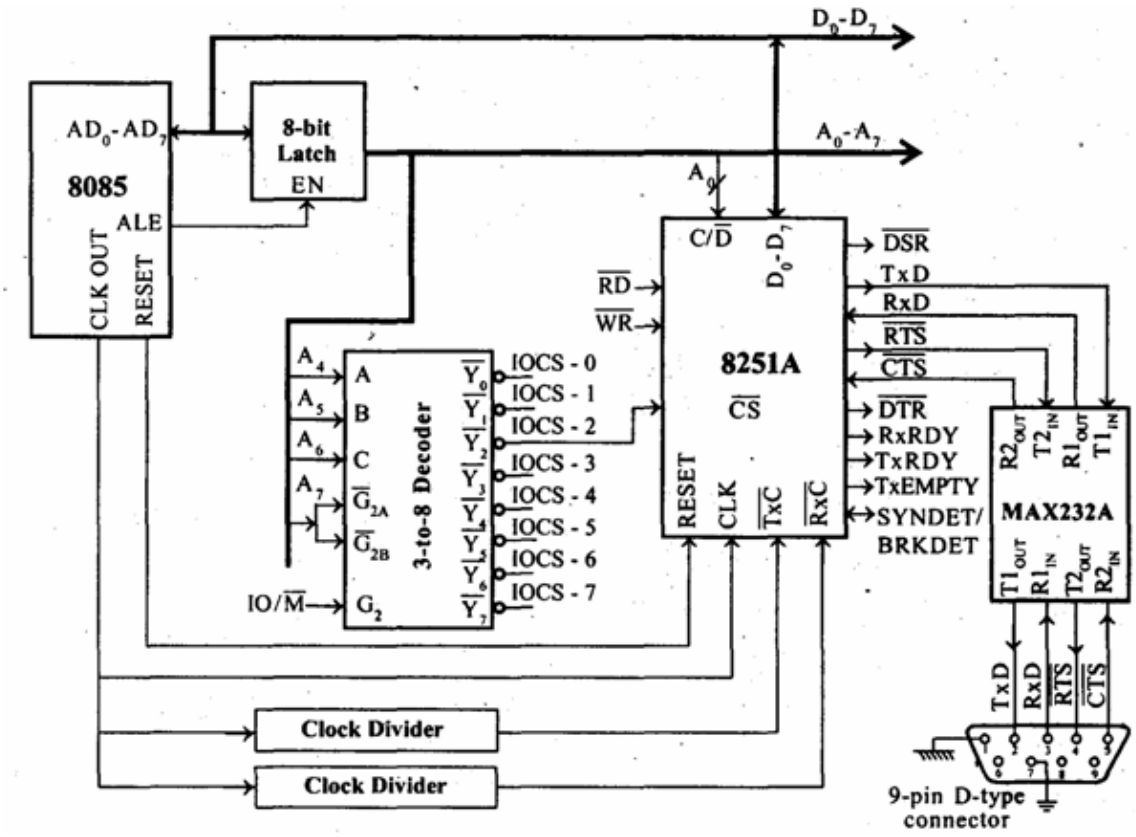


ERROR DEFINITIONS

- **Parity Error:** At the time of transmission of data an even or odd parity bit is inserted in the data stream. At the receiver end, if parity of the character does not match with the pre-defined parity, parity error occurs.
- **Overrun Error:** In the receiver section received character is stored in the receiver buffer. The CPU is supposed to read this character before reception of the next character. But if CPU fails in reading the character loaded in the receiver buffer, the next the received character replaces the previous one and the OVERRUN error occurs
- **Framing Error:** If valid stop bit is not detected at the end each character framing error occurs.
- All these errors, when occur, set the corresponding bits in the status register. These error bits are reset by setting ER bit in the command instruction.

Interfacing 8251 USART with 8085 Microprocessor

- The 825 1A can be either memory mapped or I/O mapped in the system.
- 8251A in I/O mapped in the system is shown in the figure.
- Using a 3-to-8 decoder generates the chip select signals for I/O mapped devices.
- The address lines A4, A5 and A6 are decoded to generate eight chip select signals (IOCS-0 to IOCS-7) and in this, the chip select signal IOCS-2 is used to select 8251A.
- The address line A7 and the control signal IO / M(low) are used as enable for decoder.
- The address line A0 of 8085 is connected to C/D(low) of 8251A to provide the internal addresses.
- The data lines D0 - D7 are connected to D0 - D7 of the processor to achieve parallel data transfer.
- The RESET and clock signals are supplied by the processor. Here the processor clock is directly connected to 8251A. This clock controls the parallel data transfer between the processor and 8251A.
- The output clock signal of 8085 is divided by suitable clock dividers like programmable timer 8254 and then used as clock for serial transmission and reception.
- The TTL logic levels of the serial data lines and the control signals necessary for serial transmission and reception are converted to RS232 logic levels using MAX232 and then terminated on a standard 9-pin D-.type connector.
- In 8251A the transmission and reception baud rates can be different or same.
- The device which requires serial communication with processor can be connected to this 9-pin D-type connector using 9-core cable
- The signals TxEMPTY, TxRDY and RxRDY can be used as interrupt signals to initiate interrupt driven data transfer scheme between processor and 8251 A.



I/O addresses of 8251A interfaced to 8085 is,

Internal Device of 8251A	Binary Address								Hexa Address
	Decoder input and enable				Input to address pin of 8251				
	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
Data buffer	0	0	1	0	x	x	x	0	20
Control register	0	0	1	0	x	x	x	1	21

KEYBOARD/DISPLAY CONTROLLER - INTEL 8279

The INTEL 8279 is specially developed for interfacing keyboard and display devices to 8085/8086/8088 microprocessor based system. The important features of 8279 are,

Simultaneous keyboard and display operations.
Scanned keyboard mode.
Scanned sensor mode.
8-character keyboard FIFO.
1 6-character display.
Right or left entry 1 6-byte display RAM.
Programmable scan timing.

Architecture and Signal Descriptions of 8279

- The keyboard display controller chip 8279 provides:

a) a set of four scan lines and eight return lines for interfacing keyboards

b) A set of eight output lines for interfacing display.

- Fig shows the functional block diagram of 8279 followed by its brief description.
- **I/O Control and Data Buffers** : The I/O control section controls the flow of data to/from the 8279. The data buffers interface the external bus of the system with internal bus of 8279.
- The I/O section is enabled only if CS is low. The pins A0, RD and WR select the command, status or data read/write operations carried out by the CPU with 8279.
- **Control and Timing Register and Timing Control** : These registers store the keyboard and display modes and other operating conditions programmed by CPU. The registers are written with A0=1 and WR=0. The Timing and control unit

controls the basic timings for the operation of the circuit. Scan counter divide down the operating frequency of 8279 to derive scan keyboard and scan display frequencies.

- **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 (Four externally decoded scan lines may drive upto 16 displays). In the decode scan

mode, the counter internally decodes the least significant 2 bits and provides a decoded 1 out of 4 scan on SL0-SL3(Four internally decoded scan lines may drive upto 4 displays). The keyboard and display both are in the same mode at a time.

- **Return Buffers and Keyboard Debounce and Control**: This section for a key closure row wise. If a key closer is detected, the keyboard debounce unit debounces the key entry (i.e. wait for 10 ms). After the debounce period, if the key continues to be detected. The code of key is directly transferred to the sensor

RAM along with SHIFT and CONTROL key status.

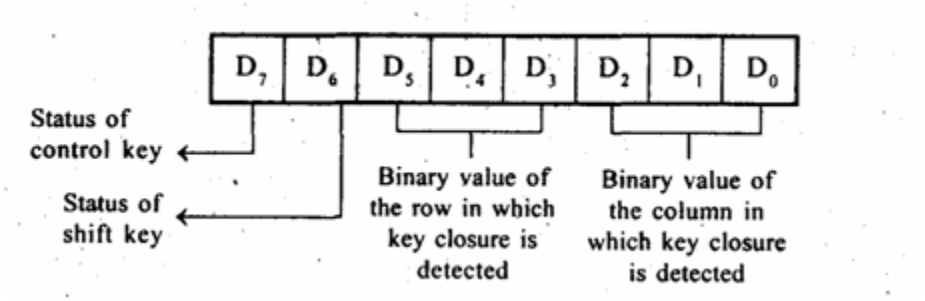
- **FIFO/Sensor RAM and Status Logic**: In keyboard or strobed input mode, this block acts as 8-byte first-in-first-out (FIFO) RAM. Each key code of the pressed key is entered in the order of the entry and in the mean time read by the CPU, till the RAM become empty.
- The status logic generates an interrupt after each FIFO read operation till the

FIFO is empty. In scanned sensor matrix mode, this unit acts as sensor RAM.

Each row of the sensor RAM is loaded with the status of the corresponding row of sensors in the matrix. If a sensor changes its state, the IRQ line goes high to

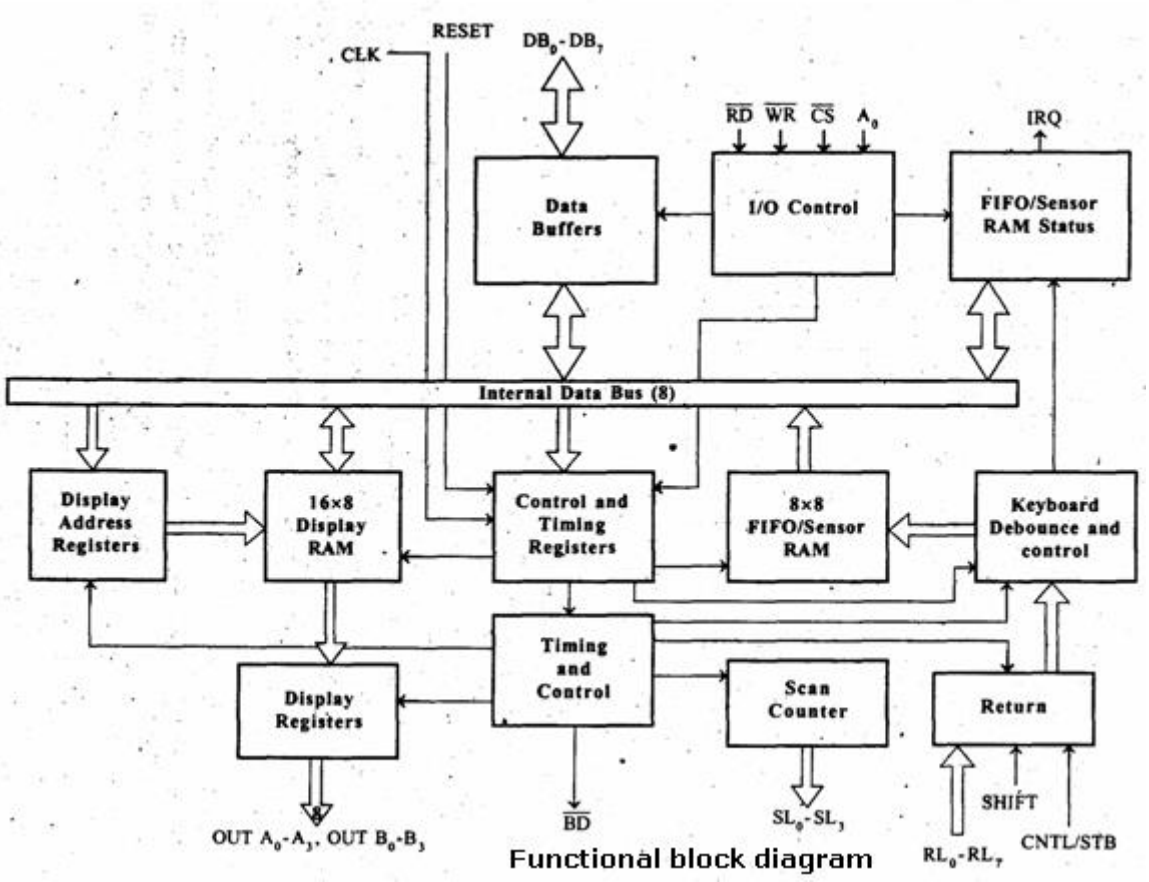
interrupt the CPU.

The FIFO can store eight key codes in the scan keyboard mode. The status of the shift key and control key are also stored along with key code. The 8279 generate an interrupt signal when there is an entry in FIFO. The format of key code entry in FIFO for scan keyboard mode is,



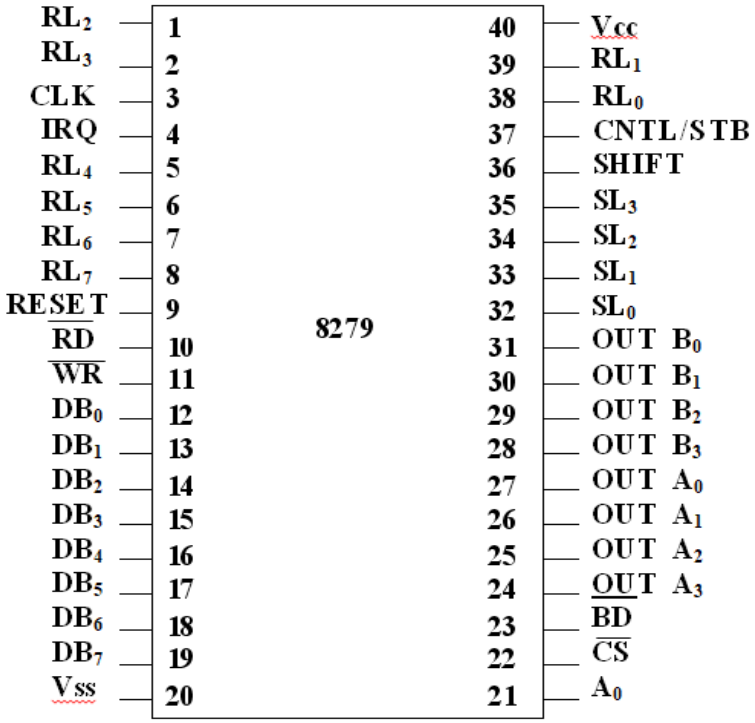
In sensor matrix mode the condition (i.e., open/close status) of 64 switches is stored in FIFO RAM. If the condition of any of the switches changes then the 8279 asserts IRQ as high to interrupt the processor.

• **Display Address Registers and Display RAM** : The display address register holds the address of the word currently being written or read by the CPU to or from the display RAM. The contents of the registers are automatically updated by 8279 to accept the next data entry by CPU.

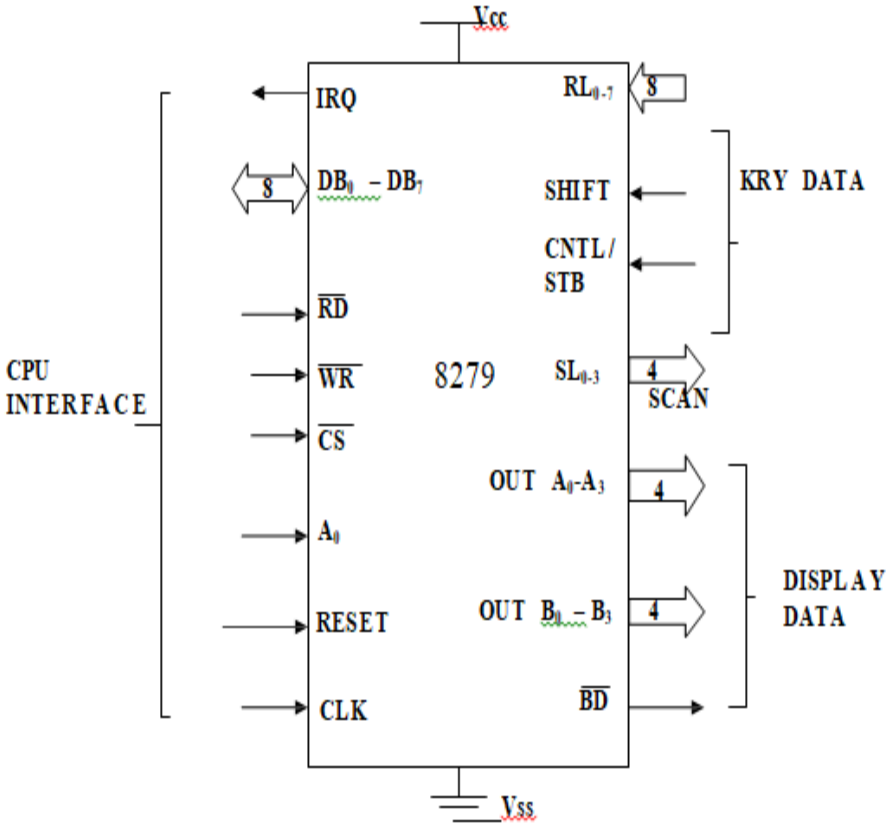


Functional block diagram

Pin diagram of 8279



8279 Pin Configuration



- The signal discription of each of the pins of 8279 as follows :
- **DB0-DB7** : These are bidirectional data bus lines. The data and command words to and from the CPU are transferred on these lines.
- **CLK** : This is a clock input used to generate internal timing required by 8279.
- **RESET** : This pin is used to reset 8279. A high on this line reset 8279. After resetting 8279, its in sixteen 8-bit display, left entry encoded scan, 2-key lock out mode. The clock prescaler is set to 31.
- **CS** : Chip Select – A low on this line enables 8279 for normal read or write operations. Other wise, this pin should remain high.
- **A_0** : A high on this line indicates the transfer of a command or status information. A low on this line indicates the transfer of data. This is used to select one of the internal registers of 8279.
- **RD, WR (Input/Output) READ/WRITE** – These input pins enable the data buffers to receive or send data over the data bus.
- **IRQ** : This interrupt output lines goes high when there is a data in the FIFO sensor RAM. The interrupt lines goes low with each FIFO RAM read operation but if the FIFO RAM further contains any key-code entry to be read by the CPU, this pin again goes high to generate an interrupt to the CPU.
- **Vss, Vcc** : These are the ground and power supply lines for the circuit.
- **SL0-SL3-Scan Lines** : These lines are used to scan the key board matrix and display digits. These lines can be programmed as encoded or decoded, using the mode control register.
- **RL_0 - RL_7 - Return Lines** : These are the input lines which are connected to one terminal of keys, while the other terminal of the keys are connected to the decoded scan lines. These are normally high, but pulled low when a key is pressed.
- **SHIFT** : The status of the shift input lines is stored along with each key code in FIFO, in scanned keyboard mode. It is pulled up internally to keep it high, till it is pulled low with a key closure.
- **BD – Blank Display** : This output pin is used to blank the display during digit switching or by a blanking closure.
- **OUT A_0 – OUT A_3 and OUT B_0 – OUT B_3** – These are the output ports for two

16*4 or 16*8 internal display refresh registers. The data from these lines is synchronized with the scan lines to scan the display and keyboard. The two 4-bit ports may also as one 8-bit port.

- **CNTL/STB- CONTROL/STROBED I/P Mode** : In keyboard mode, this lines is used as a control input and stored in FIFO on a key closure. The line is a strobed lines that enters the data into FIFO RAM, in strobed input mode. It has an interrupt pull up. The lines is pulled down with a key closer.

Modes of Operation of 8279

- The modes of operation of 8279 are as follows :

1. Input (Keyboard) modes.
2. Output (Display) modes.

- **Input (Keyboard) Modes :** 8279 provides three input modes. These modes are as follows:

1. **Scanned Keyboard Mode** : This mode allows a key matrix to be interfaced using either encoded or decoded scans. In encoded scan, an 8*8 keyboard or in decoded scan, a 4*8 keyboard can be interfaced. The code of key pressed with SHIFT and CONTROL status is stored into the FIFO RAM.
2. **Scanned Sensor Matrix** : In this mode, a sensor array can be interfaced with 8279 using either encoded or decoded scans. With encoded scan 8*8 sensor matrix or with decoded scan 4*8 sensor matrix can be interfaced. The sensor codes are stored in the CPU addressable sensor RAM.
3. **Strobed input**: In this mode, if the control lines goes low, the data on return lines, is stored in the FIFO byte by byte.

- **Output (Display) Modes** : 8279 provides two output modes for selecting the display options. These are discussed briefly.

1. **Display Scan** : In this mode 8279 provides 8 or 16 character multiplexed displays those can be organized as dual 4- bit or single 8-bit display units.
2. **Display Entry** : (right entry or left entry mode) 8279 allows options for data entry on the displays. The display data is entered for display either from the right side or from the left side.

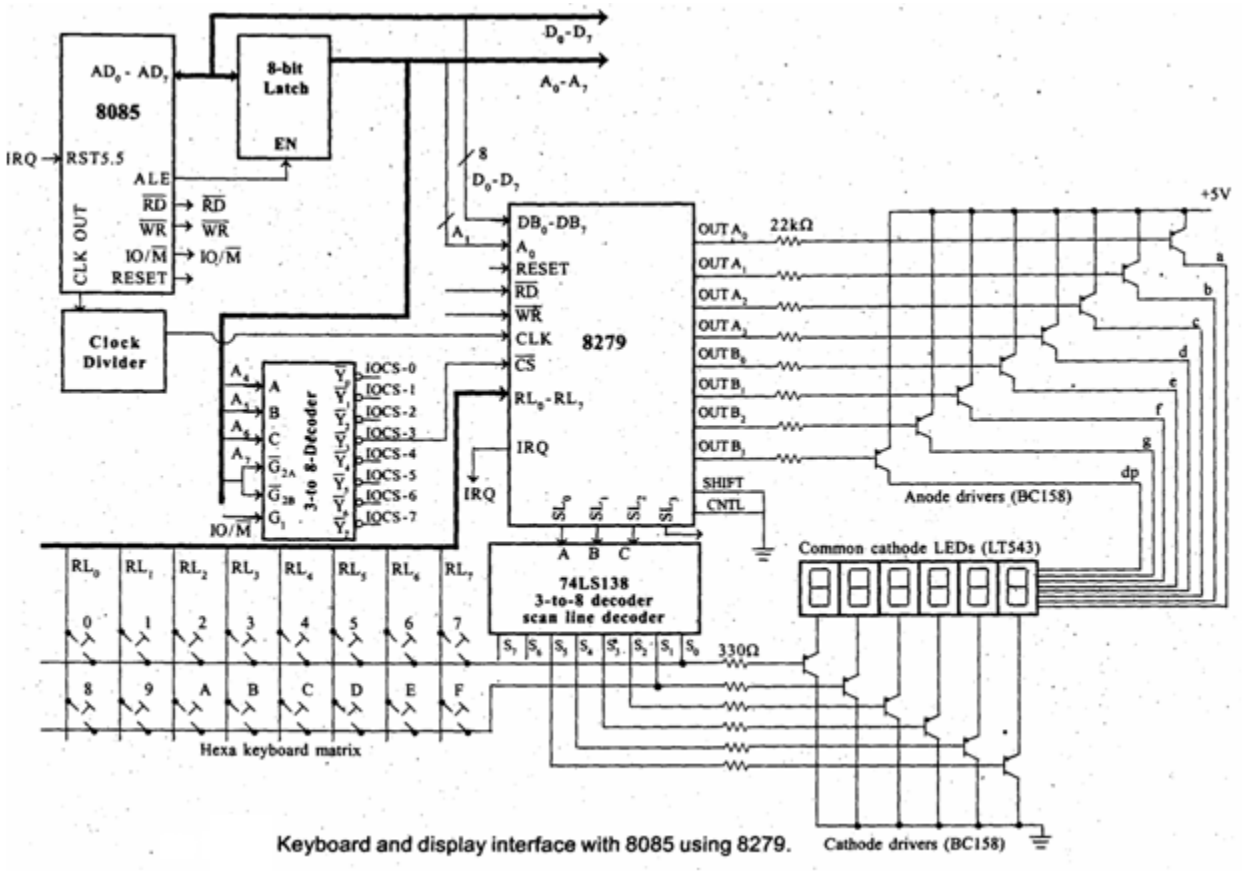
INTERFACING OF 8279 WITH 8085

In a microprocessor b system, when keyboard and 7-segment LED display is interfaced using ports or latches then the processor has to carry the following task.

Keyboard scanning
Key debouncing
Key code generation
Sending display code to LED
Display refreshing

Interfacing 8279 with 8085 processor:

A typical Hexa keyboard and 7-segment LED display interfacing circuit using 8279 is shown.



- The circuit can be used in 8085 microprocessor system and consist of 16 numbers of hexa-keys and 6 numbers of 7-segment LEDs.
- The 7-segment LEDs can be used to display six digit alphanumeric character.

- The 8279 can be either memory mapped or I/O mapped in the system. In the circuit shown is the 8279 is I/O mapped.
- The address line A0 of the system is used as A0 of 8279.
- The clock signal for 8279 is obtained by dividing the output clock signal of 8085 by a clock divider circuit.
- The chip select signal is obtained from the I/O address decoder of the 8085 system. The chip select signals for I/O mapped devices are generated by using a 3-to-8 decoder.
- The address lines A4, A5 and A6 are used as input to decoder.
- The address line A7 and the control signal IO/M (low) are used as enable for decoder.
- The chip select signal IOCS-3 is used to select 8279.
- The I/O address of the internal devices of 8279 are shown in table.

Internal Device	Binary Address								Hexa Address
	Decoder input and enable				Input to address line of 8279				
	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
Data register	0	0	1	1	x	x	x	0	30
Control register	0	0	1	1	x	x	x	1	31

- The circuit has 6 numbers of 7-segment LEDs and so the 8279 has to be programmed in encoded scan. (Because in decoded scan, only 4 numbers of 7-segment LEDs can be interfaced)
- In encoded scan the output of scan lines will be binary count. Therefore an external, 3-to-8 decoder is used to decode the scan lines SL0, SL1 and SL2 of 8279 to produce eight scan lines S0 to S7.
- The decoded scan lines S0 and S1 are common for keyboard and display.
- The decoded scan lines S2 to S5 are used only for display and the decoded scan lines S6 and S7 are not used in the system.
- Anode and Cathode drivers are provided to take care of the current requirement of LEDs.
- The pnp transistors, BC 158 are used as driver transistors.
- The anode drivers are called segment drivers and cathode drivers are called digit drivers.
- The 8279 output the display code for one digit through its output lines (OUT A0 to OUT A3 and OUT B0 to OUT B3) and send a scan code through, SL0- SL3.

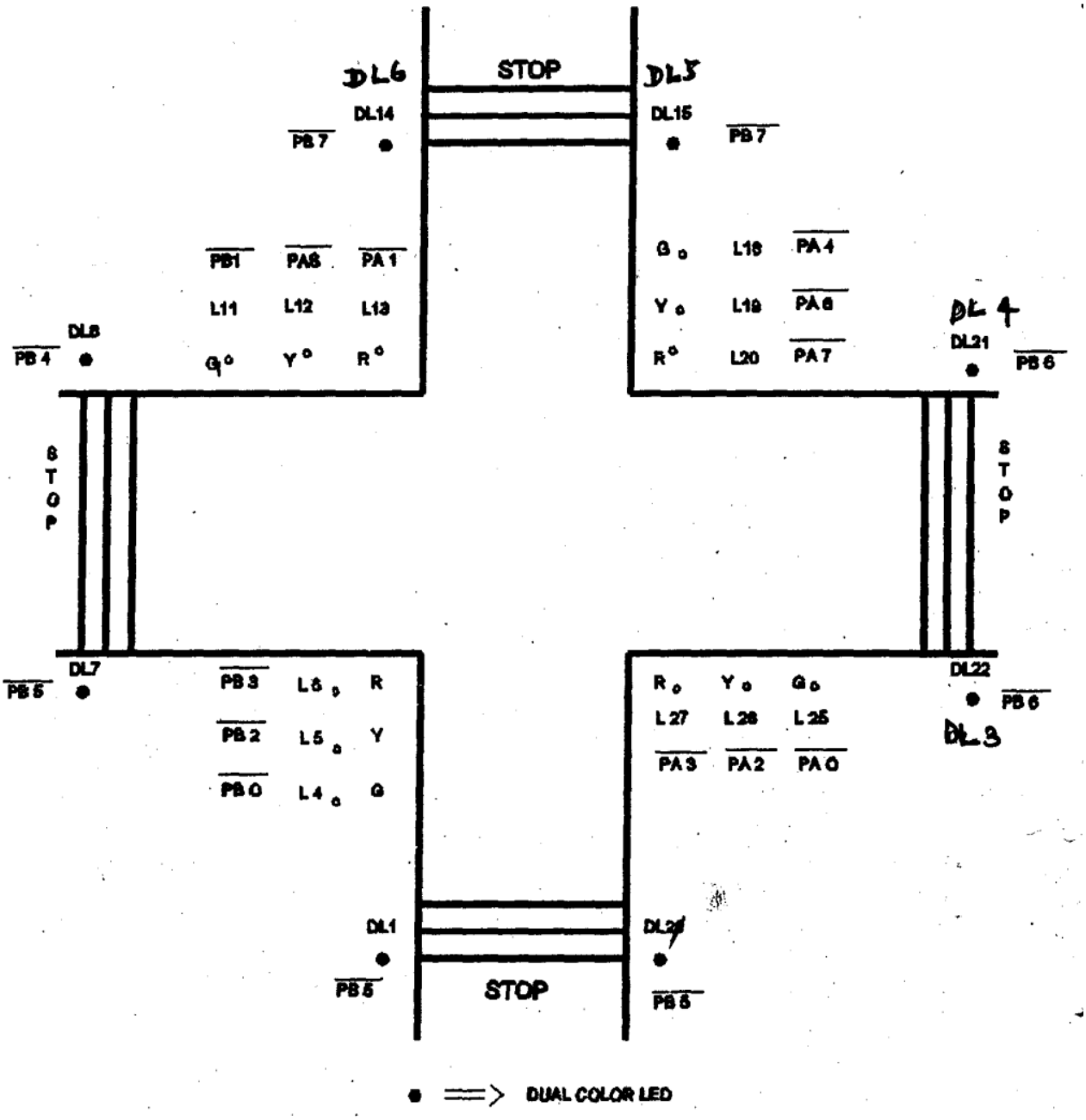
- The display code is inverted by segment drivers and sent to segment bus.
- The scan code is decoded by the decoder and turns ON the corresponding digit driver. Now one digit of the display character is displayed. After a small interval (10 milli-second, typical), the display is turned OFF (i.e., display is blanked) and the above process is repeated for next digit. Thus multiplexed display is performed by 8279.
- The keyboard matrix is- formed using the return lines, RL0 to RL3 of 8279 as columns and decoded scan lines S0 and S1 as rows.
- A hexa key is placed at the crossing point of each row and column. A key press will short the row and column. Normally the column and row line will be high.
- During scanning the 8279 will output binary count on SL0 to SL3, which is decoded by decoder to make a row as zero. When a row is zero the 8279 reads the columns. If there is a key press then the corresponding column will be zero.
- If 8279 detects a key press then it wait for debounce time and again read the columns to generate key code.
- In encoded scan keyboard mode, the 8279 stores an 8-bit code for each valid key press. The keycode consist of the binary value of the column and row in which the key is found and the status of shift and control key.
- After a scan time, the next row is made zero and the above process is repeated and so on. Thus 8279 continuously scan the keyboard.

Device	Function	Internal addresses
INTEL 8279	Keyboard/display controller. Used for keyboard scanning and display refreshing.	**Two-internal addresses** $A_0 = 0 \rightarrow$ Data register $A_0 = 1 \rightarrow$ Control register
INTEL 8257 or INTEL 8237	DMA controller. Used for supporting DMA access to I/O device. It acts as a master during DMA mode. It is a slave device during programming mode.	**Sixteen-internal addresses** A_3 A_2 A_1 A_0 0 0 0 0 0 0 0 1 . . . . 1 1 1 1
INTEL 8259	Interrupt controller. Used to expand the hardware interrupt INTR to eight interrupts in 8085 based system and 256 interrupts in 8086 based system.	**Two-internal addresses** $A_0 = 0$ $A_0 = 1$
INTEL 8253/ 8254	Programmable Timer. Used in the system to produce various timing signals. It has three independent counters and can be programmed in six operating modes.	**Four-internal addresses** A_1 A_0 Counter-0 0 0 Counter-1 0 1 Counter-2 1 0 Control Register 1 1
INTEL 8251 USART	Universal Synchronous/Asynchronous Receiver Transmitter. Used for serial data communication.	**Two-internal addresses** $C/\overline{D} = 0 \rightarrow$ Data register $C/\overline{D} = 1 \rightarrow$ Control register

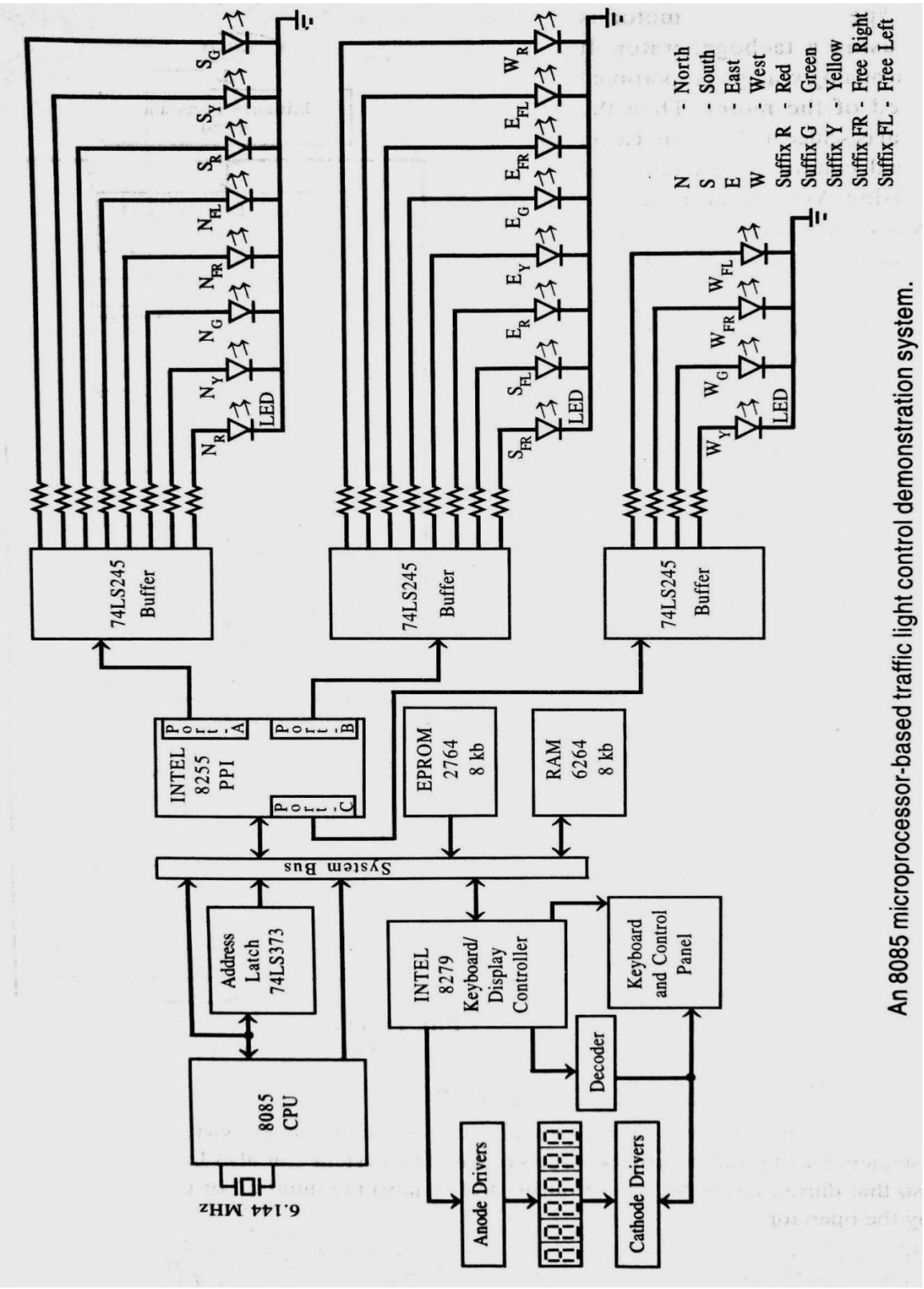
Traffic light control system using 8085 microprocessor through 8255 programmable interface :

Working principle:

- The system can also have a manual control option, so that during heavy traffic(or during traffic jam) the duration of on/off time can be varied by the operator
- For manual control to a keyboard have been provided
- It will be helpful for the operator if the direction of the traffic flow is displayed during manual control
- Hence 7-segment LEDs are interfaced to display the direction of traffic flow both during manual and automatic mode
- The primary function of the microprocessor in the system is to switch on/off the red /yellow/green lights in the specified sequence
- The LEDs are interfaced to the system through buffer(74LS245) and ports of 8255
- In the practical implementation scheme the lights can be turned on/off using driver transistor and relays
- In practical implementation the output of buffer (74LS245) can be connected to the driver transistor .
- A relay placed at the collector of the transistor can be used to switch on/off the lights
- A reverse biased diode is connected across relay coil to prevent relay chattering (for free-wheeling action)
- The microprocessor send high through a port line to switch on the light and low to switch off the light
- In this switching sequence it is assumed that the traffic is allowed only in one direction at a time



- The 1's and 0's that represent on and off condition can be directly output to 8255 port to switch ON/OFF the light
- The processor can output the codes for switching the lights for schedule –i and then waits.
- After a specified time delay the processor outputs the codes for schedule-ii and so on.
- For each schedule the processor can wait for a specified time.
- After schedule-xii, the processor can again return to schedule-i.
- On observing the schedules we can conclude that three different delay routines are sufficient for implementing the twelve switching schedules



Program:

Program for microprocessor based traffic light system using micro-85 EB

To write the program for the TLC

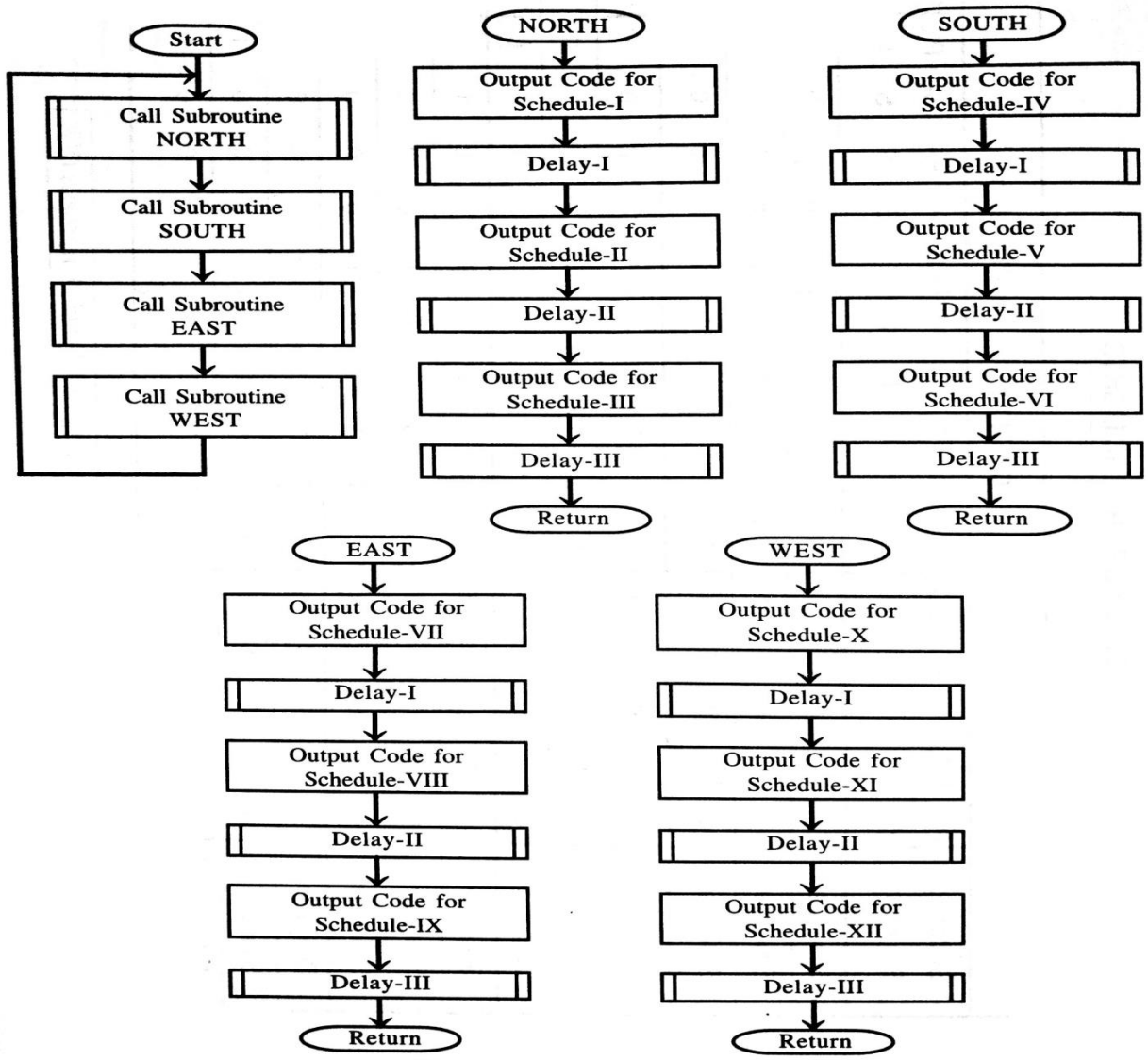
4100		ORG	4100
000F	CONTROL	EQU	0FH
000C	PORTA	EQU	0CH
000D	PORTB	EQU	0DH
000E	PORTC	EQU	0EH

Switching Schedule for traffic:

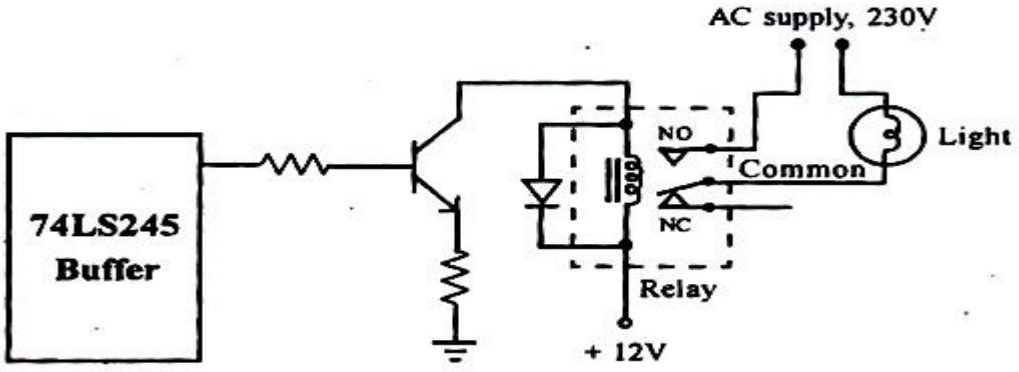
: SWITCHING SCHEDULE FOR TRAFFIC LIGHTS

Switching Schedule	ON/OFF status of traffic lights															
	PC_3 W_{RL}	PC_2 W_{FR}	PC_1 W_G	PC_0 W_Y	PB_7 W_R	PB_6 E_{FL}	PB_5 E_{FR}	PB_4 E_G	PB_3 E_Y	PB_2 E_R	PB_1 S_{FL}	PB_0 S_{FR}	PA_7 S_G	PA_6 S_Y	PA_5 S_R	PA_4 N_{FL}	PA_3 N_{FR}	PA_2 N_G	PA_1 N_Y	PA_0 N_R
Schedule I	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1
Schedule II	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0
Schedule III	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	1	1	1	0	0
Schedule IV	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0
Schedule V	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1
Schedule VI	0	0	0	0	1	0	0	0	0	1	1	1	1	0	0	0	0	0	0	1
Schedule VII	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1
Schedule VIII	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1
Schedule IX	0	0	0	0	1	1	1	1	0	0	0	0	0	0	1	0	0	0	0	1
Schedule X	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1
Schedule XI	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1
Schedule XII	1	1	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1

Note : "1" represents ON and "0" represents OFF.



Flowchart for traffic light control program.



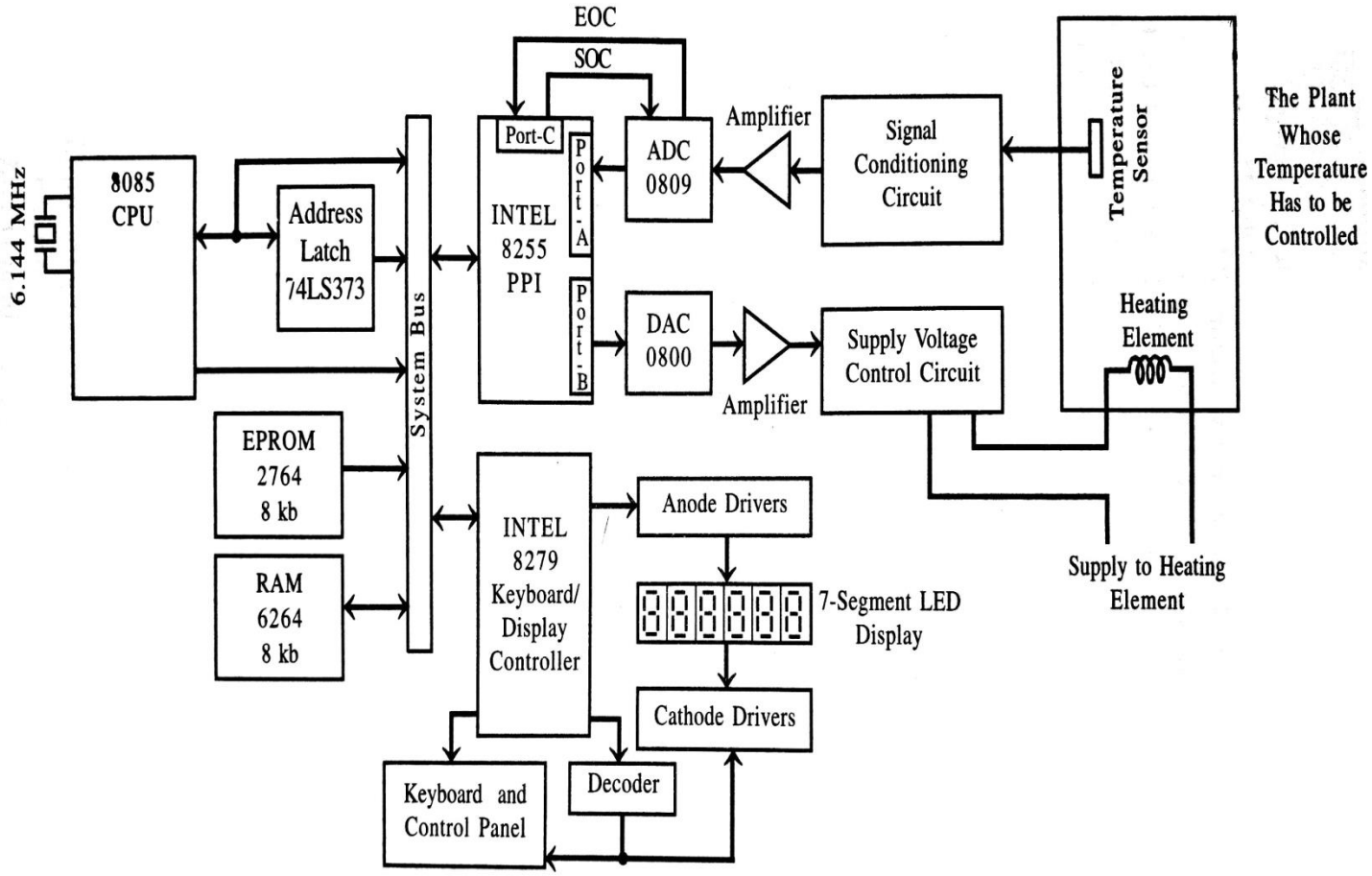
Switching circuit for traffic light.

8085 application in monitoring and control of temperature.

Working principle:

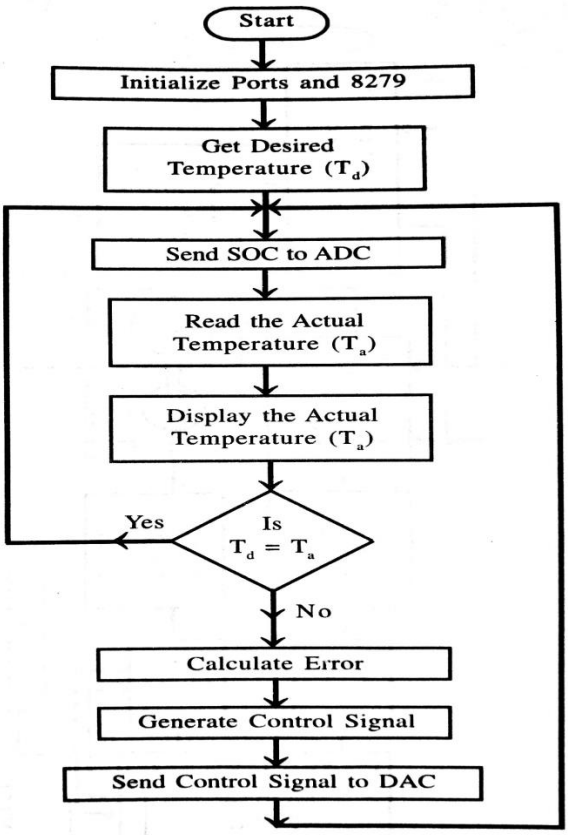
- The microprocessor based temperature control system can be used for automatic control of temperature of a body
- A simplified block diagram of 8085 microprocessor based temperature control system is shown in fig
- In this system the temperature is controlled by controlling the power input to the heating element
- The EPROM memory is provided for storing system program, and RAM memory for temporary data storage and stack operation
- Using INTEL 8279, a keyboard and six number of 7-segment LEDs are interfaced to the system
- The system have been designed to accept the desired temperature and various control commands through keyboard
- The 7-segment display have been provided to display the temperature of the body at any time instant
- The temperature of the body is measured using a temperature sensor
- The different types of temperature sensor that can be used for temperature measurement are thermo-couples, thermistors, PN-junctions, IC sensors like AD590, etc.
- These sensors will convert the input temperature to proportional analog voltage or current.
- The output of the sensor will be a weak signal and so it has to be amplified using high input impedance operational amplifier
- Then the analog signal is scaled to suitable level by the signal conditioning circuit
- The microprocessor can process only digital signals and so the analog signal form signal conditioning circuit cannot be read by the processor directly
- The system has an analog-to-digital convertor(ADC) to convert the analog to proportional digital signal
- In this system the ADC is interfaced to 8085 processor through port-A and port-c of 8233.
- The 8085 processor send signal to ADC to port-C to start conversion and at the end of conversion it read the digital data from the port-A of 8255
- The 8085 processor calculate the actual temperature using the input data and display it on the 7-segment LED
- Also the processor compare the desired temperature with actual temperature (the operator can enter the desired temperature through keyboard)and calculate the error (the difference between actual temperature and desired temperature)
- The error is used to compute a digital control signal, which is converted to analog control signal by DAC.
- The DAC is interfaced to the system through port-B of 8255.
- The analog control signal produced by DAC is used to control the power supply of the heating element of the body

- The digital control signal can be computed by the 8085 processor using different digital control algorithm (P/PI/PID/FUZZY logic control algorithm)
- The control circuit for power supply can be thyristor based circuit or relay
- In case of thyristor control circuit the firing angle can be varied by the control signal to control the power input to the heater
- In case of relay the control signal can switch ON/OFF the relay to control the power input to the heater



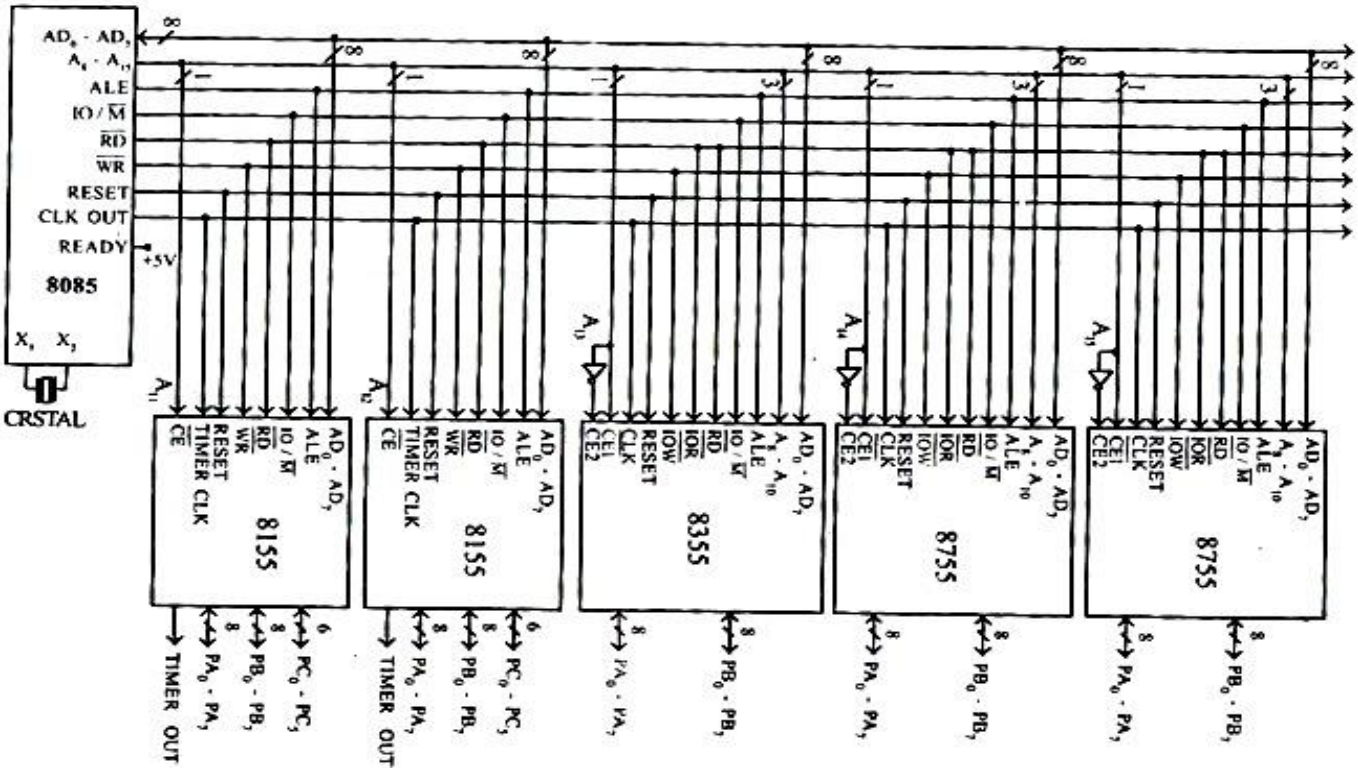
An 8085 microprocessor-based temperature control system.

Temperature monitoring and Control circuit



Flowchart for temperature control system.

Interfacing arrangement

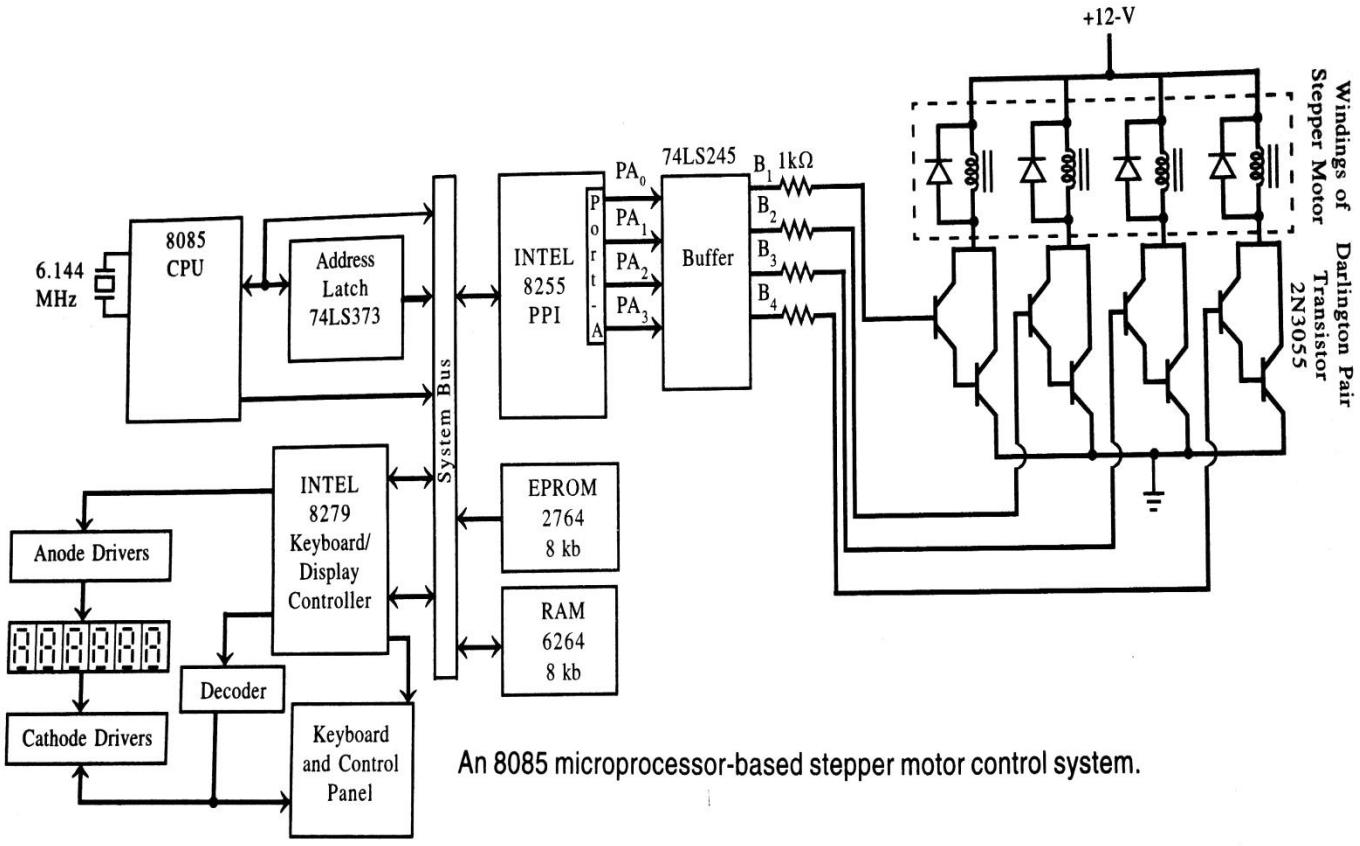


Stepper motor is interfaced with 8085 microprocessor system.

The stepper motors are properly used in computer peripherals, plotters, robots and machine tools for precise incremental rotation. In stepper motor, the stator windings are excited by electrical pulses and for each pulse the motor shaft advances by one angular step. (Since the stepper motor can be driven by digital pulses, it is also digital motor). The step size in the motor is determined by the number of poles in the rotor and the number of pairs od\f stator windings (one pair of stator winding is called one phase). The stator windings are also called control windings.

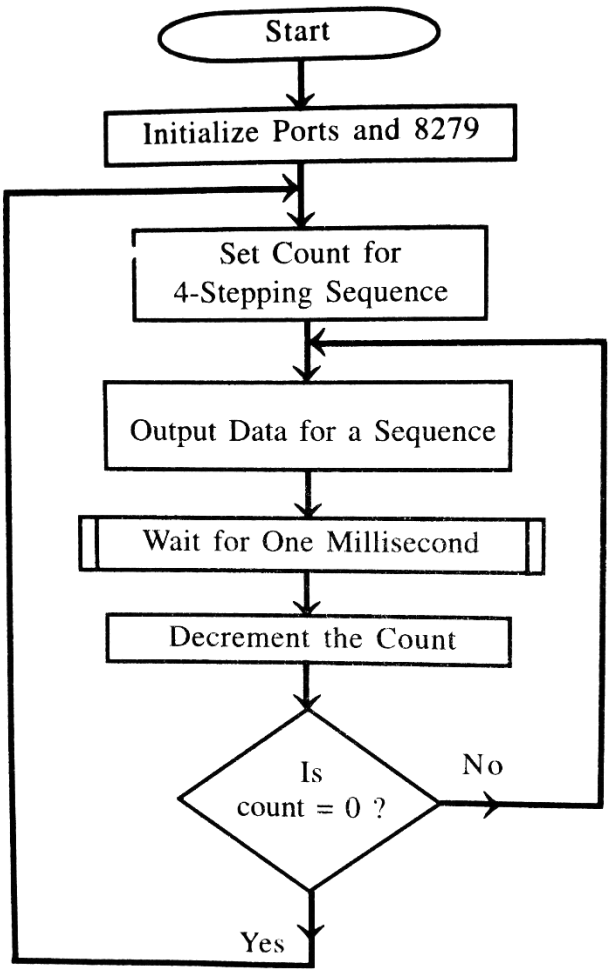
The motor is controlled by switching ON/OFF the control winding. The popular stepper motor used for demonstration in laboratories has a step size of 1.8° (i.e. 200 steps per revolution). This motor consists of four stator winding and require four switching sequence as shown in table. The basic step size of the motor is called full-step. By altering the switching sequence, the motor can be made to run with incremental motion of half the full step value. The switching sequence for half step rotation is shown in table.

A typical stepper motor control system is shown in fig. a two-phase or four winding stepper motor is shown in fig. The system consists of 8085 microprocessor as CPU, EPROM and RAM memory for program and data storage and for stack. Using INTEL 8279, a keyboard and six numbers of 7-segment LED display have been interfaced in the system. Through the keyboard the operator can issue commands to control the system. The LED displays have been provided to display messages to the operator.



An 8085 microprocessor-based stepper motor control system.

The windings of stepper motor are connected to the collector of Darlington pair transistors. The transistors are switched ON/OFF by the microprocessor through the ports of 8255 and buffer (74LS245). A free-wheeling diode is connected across each winding for fast switching. The flowchart for the operational flow of the stepper motor control system is shown in the fig. The processor has to output a switching sequence and wait for 1 to 5 msec before sending next switching sequence (The delay is necessary to allow the motor transients to die-out).



Flowchart for stepper motor control program.

SWITCHING SEQUENCE FOR FULL-STEP ROTATION

Switching sequence	Clockwise rotation				Anticlockwise rotation			
	PA_3	PA_2	PA_1	PA_0	PA_3	PA_2	PA_1	PA_0
Sequence-1	1	1	0	0	0	0	1	1
Sequence-2	0	1	1	0	0	1	1	0
Sequence-3	0	0	1	1	1	1	0	0
Sequence-4	1	0	0	1	1	0	0	1

SWITCHING SEQUENCE FOR HALF-STEP ROTATION

Clockwise rotation				Anticlockwise rotation			
PA_3	PA_2	PA_1	PA_0	PA_3	PA_2	PA_1	PA_0
1	1	0	0	0	0	1	1
0	1	0	0	0	0	1	0
0	1	1	0	0	1	1	0
0	0	1	0	0	1	0	0
0	0	1	1	1	1	0	0
0	0	0	1	1	0	0	0
1	0	0	1	1	0	0	1
1	0	0	0	0	0	0	1

2 marks

1. **What is key bouncing?(Apr/2016)**
   When a key is pressed it bounces for a short time. If a key code is generated immediately after a key actuation, then the processor will generate the same keycode a number of times.(A key typically bounce for 10 to 20 milliseconds). Hence the processor has to wait for the key bounces to settle down before reading the keycode. This process is called keyboard debouncing.
2. **How the RS-232C serial bus is interfaced to TTL logic device?(Apr/2016)**
   The Rs-232C signal voltage level are not compatible with TTL logic levels. Hence for interfacing TTL devices to RS-232C serial bus, level converters are used, the popularly used level converters are MC 1488 and MC 148 or MAX 232.
3. **List the operating modes of 8255A programmable peripheral interface ?(Nov/2015)**

There are two basic modes of operation of 8255, They are:

a)I/O mode.

b)BSR mode.(Bit set-reset)

In I/O mode, the 8255 ports work as programmable I/O ports,

In BSR mode only port C (PC0-PC7) can be used to set or reset its individual port bits.

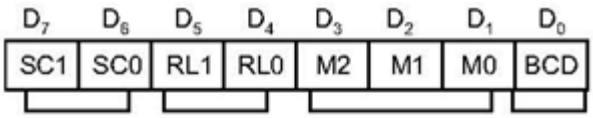
Under the IO mode of operation, further there are three modes of operation of 8255,

So as to support different types of applications, viz. mode 0, mode 1 and mode 2.

1) Mode 0 - Basic I/O mode
2) Mode 1 - Strobed I/O mode
3) Mode 2 - Strobed bi-directional I/O

4. **What do you mean by timing diagram?(Nov/2014)**
   It is the graphical representation of process in steps with respect to time. The timing diagram represents the clock cycle and duration, delay, content of address bus and data bus, type of operation ie. Read/write/status signals.

5.What is the control word for Square Wave rate generator mode of operation of 8253 timer?(Apr/2016)



Definition of Control:

SC-Select Counter:

SC1	SC0	
0	0	Select counter 0
0	1	Select counter 1
1	0	Select counter 2
1	1	Illegal

M-Mode

0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

RL-Read/Load:

RL1	RL0	
0	0	Counter latching operation (see read/write procedure section)
1	0	Read/Load most significant byte only
0	1	Read/Load Least significant byte only
1	1	Read/Load Least significant byte first, then most significant byte

BCD:

0	Binary counter 16-bits
1	Binary coded decimal (BCD) counter (4 decades)

D3 M2	D2 M1	D1 M0	Mode value
0	0	0	mode 0: interrupt on terminal count
0	0	1	mode 1: programmable one-shot
x	1	0	mode 2: rate generator
x	**1**	**1**	**mode 3: square wave generator**
1	0	0	mode 4: software triggered strobe
1	0	1	mode 5: hardware triggered strobe

6.List the various mode of operation of 8259.(Apr/2016)

- Many types of priority modes are available under software control in the 8259.
- They can be changed dynamically during the program by writing appropriate command words.

Commonly used priority modes are given below:

a) Fully nested mode :

b) Automatic rotation mode:

c) Specific rotation mode:

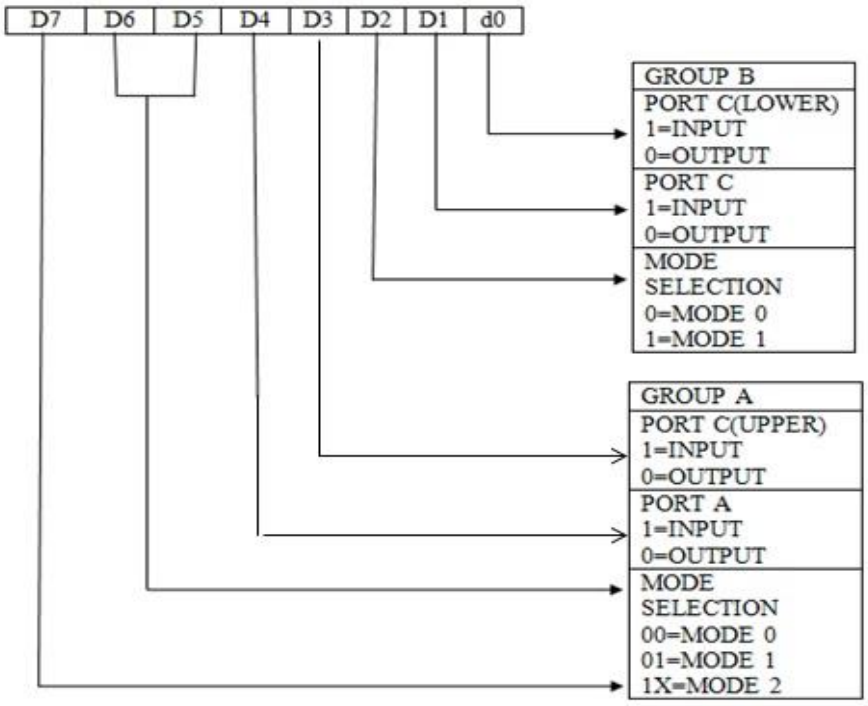
d) End of interrupt

i) Non Specific EOI command

ii) Specific EOI command

e) Automatic EOI

7.Draw the control word format of 8255 for I/O mode. (April / May 2014)



8.What are the hardware components required in designing control circuit to control the temperature. (April /May 2014)

- Keyboard / Display controller
- Signal conditioning circuit
- 8255 Programmable Peripheral Interface
- ADC & DAC
- Temperature sensor

9.Explain the functions of hand shake signals. (April /May 2012,Nov/2015)

In handshake port, signals are exchanged between I/O device and port or between port and processor for checking / informing various condition of the device.

10. What are the input signals available in 8251A programmable communication interface?(Nov-2014)

- Command / Status or data
- Control signal
- Read and Write signal
- DSR and CTS (active low signal)

- RXD

11. List the various modes of operation of 8259.(April 2013)

(a)Fully nested mode
(b)End of Interrupt (EOI)
(c)Automatic Rotation
(d)Automatic EOI Mode
(e)Specific Rotation
(f)Special Mask Mode
(g)Edge and level Triggered Mode
(h) Reading 8259 Status
(i)Poll command
(j)Special Fully Nested Mode
(k)Buffered mode
(l) Cascade mode

12. What are the internal devices of 8255? (November 2013)

The internal devices of 8255 are port-A, port-B and port-C. The ports can be programmed for either input or output function in different operating modes.

13. What is USART? (November 2013)

The device which can be programmed to perform Synchronous or Asynchronous serial communication is called USART (Universal Synchronous Asynchronous Receiver Transmitter). The INTEL 8251A is an example of USART.

14.List the major components of the 8279 keyboard/display interface. (November 2012)

- Display section with its own display RAM
- Keyboard scan section with FIFO registers
- Control logic with signals for interfacing with the processor

15. List the operating models of the 8255A programmable peripheral interface. (Nov. 2012)

Operating modes	
I/O mode (D7 bit=1)	BSR mode (D7 bit=0)
Mode 0: Basic I/O (Applicable to ports A, B, C	
Mode 1: Handshake I/O (Applicable to groups A and B)	
Mode 2: Bidirectional I/O	

(Applicable to group A only)

16. What is Programmable Peripheral device?

If the functions performed by a peripheral device can be altered or changed by a program instruction then the peripherals device is called programmable device. Usually the programmable devices will have control registers. The device can be programmed by sending control word in the prescribed format to the control register.

17.How DMA is initiated.

When the I/O device needs a DMA transfer, it will send a DMA request signal to DMA request signal to DMA controller. The DMA controller in-turnsends a HOLD request to the processor. When the processor receives a HOLD request, it will drive its tri-stated pins to high impedance state at the end of current instruction execution and send an acknowledge signal to DMA controller. Now the DMA controller will perform DMA transfer.

18.What is baud rate?

The baud rate is the rate at which the serial data are transmitted. Baud rate is defined as (1 / the time for a bit cell). In some systems one bit cell has one data bit, then the baud rate and bits per second are same.

19. How does port C of 8255 differ from port A and B?(April 2015)

- The port-C pins are used for handshake signals.
- Port C can be used as an 8 bit parallel I/O port mode-0
- It can be used as two numbers of 4 bit parallel port in mode-0
- The individual pins of port-C can be set or reset for various control applications

20. What are the various programmable data transfers? (April 2015)

- Synchronous data transfer
- Asynchronous data transfer
- Interrupt Driven data transfer

21. List the features of 8079 keyboard interface. (April /May 2014)

- Keyboard scanning
- Key de-bouncing
- Key-code generation
- Informing the key entry to CPU
- Storing display codes
- Output display codes to LEDs

22.What are the hardware components required in designing control circuit to control the temperature? (April /May 2014)

- Keyboard / Display controller

- Signal conditioning circuit
- 8255 Programmable Peripheral Interface
- ADC & DAC
- Temperature sensor

23. What are the internal devices of 8255?

The internal devices of 8255 are port-A, port-B and port-C. The ports can be programmed for either input or output function in different operating modes

24. Write the significance of DMA in microprocessors operation.(Nov. 2012, Nov.2015)

When the I/O device needs a DMA transfer, it will send a DMA request signal to DMA request signal to DMA controller. The DMA controller in-turn sends a HOLD request to the processor. When the processor receives a HOLD request, it will drive its tri-stated pins to high impedance state at the end of current instruction execution and send an acknowledge signal to DMA controller. Now the DMA controller will perform DMA transfer.

25. Write the principle of frequency measurement using microprocessor. (Nov. 2012)

The measurement of frequency requires:

- Measurement of sample time
- Counting of pulses during the measured sample time

Pondicherry University Questions:

2 marks

1. What is key bouncing?(Apr/2016)
2. How the RS-232C serial bus is interfaced to TTL logic device?(Apr/2016)
3. What is the control word for Square Wave rate generator mode of operation of 8253 timer?(Apr/2016)
4. List the various mode of operation of 8259.(Apr/2016)
5. How does port C of 8255 differ from port A and B? (April 2015)
6. What are the various programmable data transfers? (April 2015)
7. List the operating modes of 8255A programmable peripheral interface ?(Nov/2015)
8. What is the significance of HOLD signal available 8085 microprocessor?(Nov/2015)
9. Explain the functions of hand shake signals ?(Nov/2015)
10. What are the input signals available in 8251 A programmable communication interfaces? (Nov/2014)
11. Which is the flag unavailable for the programmer in 8085 µP with neat diagram ?(Nov/2014)
12. What do you mean by timing diagram?(Nov/2014)
13. Draw the control word format of 8255 for I/O mode (April / May 2014)

14. List the features of 8079 keyboard interface. (April / May 2014)
15. What are the hardware components required in designing control circuit to control the temperature? (April / May 2014)
16. List the various modes of operation of 8259. (April 2013)
17. What are the internal devices of 8255? (November 2013)
18. What is USART? (November 2013)
19. Write the significance of DMA in microprocessors operation. (Nov2012)
20. List the major components of the 8279 keyboard / display interface. (Nov-2012)
21. List the operating modes of the 8255A programmable peripheral interface. (Nov-2012)

11 marks

1. With the neat block diagram explain the functions of 8251?(11m-april 2016)
2. Design an interface circuit needed to connect DIP switch as an input device and display the value of the key pressed using a 7 segment LED display. Using 8085 system, write a program to implement the same?(11m-april 2016)
3. List out the salient features of 8279 keyboard interface. Interface an8*8 matrix keyboard to the microprocessor using 8279 IC. Discuss the hardware?(11m-may 2016)
4. Explain the principle operation of stepper motor with the winding excitation sequence and analysis the microprocessor based control of stepper motor with programming using 8085 microprocessor?(11m-may 2016)
5. Explain the mode 1 I/P and O/P configuration of 8255 PPI?(11m-april 2015)
6. Describe the architecture working of 8253 timer?(6m-april 2015)
7. Describe the working of 8259 interrupt controller?(5m-april 2015)
8. List the components of the 8259A interrupt controller and explain their functions?(11m-nov 2015)
9. Explain BSR operating mode of 8255A programmable peripheral interface?(11m-nov 2015)
10. Explain the programmable interrupt controller of 8259?(11m-dec 2014)
11. Explain the social communication interface 8251A USART?(11m-dec 2014)
12. Discuss about the various addressing modes in 8086 microprocessor with examples?(11m-nov 2014)
13. Draw the internal block diagram of 8086 microprocessor and discuss about the various blocks?(11m-nov 2014)
14. Draw the control word format of 8253 and explain various modes of operation with relevant waveform?(11m-may 2014)
15. Draw the block diagram of 8251 USART and explain its salient features?(11m-may 2014)
16. Explain in detail the programmable peripheral interface 8255. (April 2010) (Nov. 2013)
17. Explain in detail the 8251 USART for serial communication. (April 2010), (Nov. 2011)
18. Explain with a neat block diagram the temperature monitoring system using 8085. (Nov. 2011)
19. Discuss the traffic control system in detail. (Nov. 2012)

20. Explain interfacing a stepper motor to the 8085 microprocessor system and write an 8085 assembly language program to control the stepper motor. (April 2013)
21. Draw the block diagram of 8279 and explain the significance of each block (April 2013)

References:

1. Ramesh Gaonkar, "Microprocessor Architecture, Programming and Applications with the 8085", Fifth Edition, PenramInternational Publishing(India) Pvt.Ltd., 2011.
2. M.Senthilkumar,M.Saravanan and S.Jeevanandhan,"Microprocessorsand Microcontrollers", Oxford University Press, 2010.

---

srivenkateshwaraa College of Engineering & Technology
(Approved by AICTE, New Delhi & Affiliated to Pondicherry University, Puducherry)
13-A, Pondy - Villupuram Main Road, Ariyur, Puducherry - 605 102.
ASPIRE TO EXCEL
ISO 21001

Department of Electrical and Electronics Engineering

Subject Name: **Microprocessors and Microcontrollers** Subject Code: **EE T63**

Prepared by:

Mrs.S.Punitha, Assistant Professor/EEE
Mr.V.Malarselvam, Assistant Professor/EEE

Verified by: **Approved by:**

Unit-V

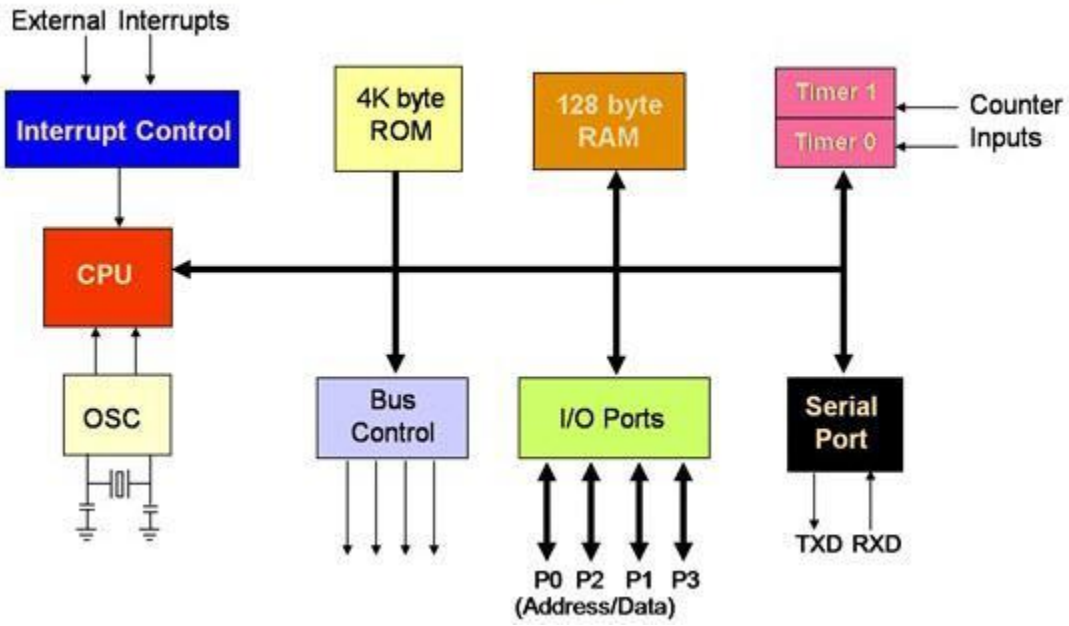
8051 MICROCONTROLLER: Introduction to Microcontrollers– 8051– Architecture –programming - hardware -Input/Output portsand circuits-Memory -Counter and Timers- Serial dataInput/Output-Interrupts-interfacing-keyboard, LCD, ADCand DAC.

Introduction to Microcontrollers– 8051

Brief History of 8051

The first microprocessor 4004 was invented by Intel Corporation. 8085 and 8086 microprocessors were also invented by Intel. In 1981, Intel introduced an 8-bit microcontroller called the 8051. It was referred as system on a chip because it had 128 bytes of RAM, 4K byte of on-chip ROM, two timers, one serial port, and 4 ports (8-bit wide), all on a single chip. When it became widely popular, Intel allowed other manufacturers to make and market different flavors of 8051 with its code compatible with 8051. It means that if you write your program for one flavor of 8051, it will run on other flavors too, regardless of the manufacturer. This has led to several versions with different speeds and amounts of on-chip RAM.

Block Diagram of 8051 Microcontroller
The following illustration shows the block diagram of an 8051 microcontroller −

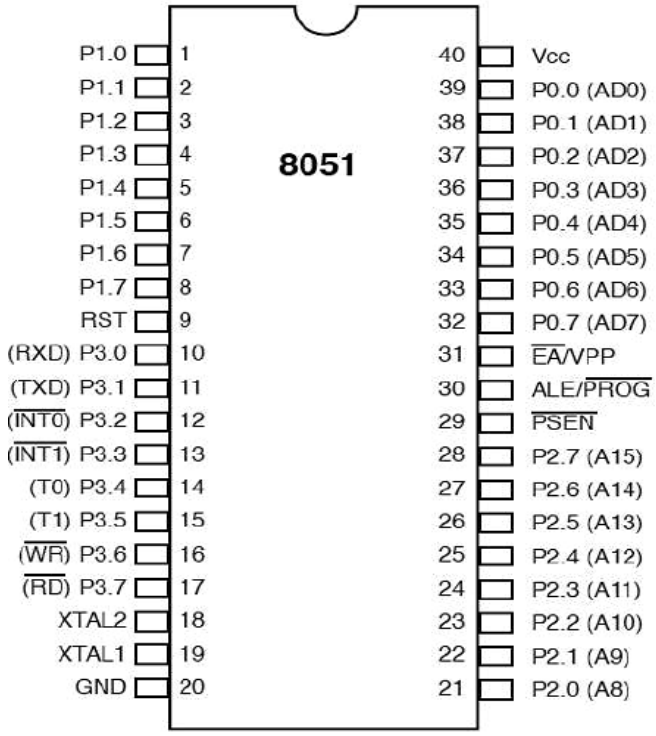


Features of 8051 microcontroller

The 8051 microcontroller is a very popular 8-bit microcontroller introduced by Intel in the year 1981 and it has become almost the academic standard now a days.
The salient features of 8051 Microcontroller:

- 4 Kbytes on chip program memory (ROM or EPROM)).
- 128 bytes on chip data memory(RAM).
- 8-bit data bus
- 16-bit address bus
- 32 general purpose 8-bit registers
- Two -16 bit timers T_0 and T_1
- Five Interrupts (3 internal and 2 external).
- Four Parallel ports each of 8-bits (PORT0, PORT1, PORT2, PORT3) with a total of 32 I/O lines.
- One 16-bit program counter and One 16-bit DPTR (data pointer)
- One 8-bit stack pointer
- One Microsecond instruction cycle with 12 MHz Crystal.
- One full duplex serial communication port.

8051 PIN DESCRIPTION



Pindiagram and Architecture of 8051

Pin Description of 8051

Pin Number	Description
1	Port -1.0
2	Port -1.1
3	Port -1.2
4	Port -1.3
5	Port -1.4
6	Port -1.5
7	Port -1.6
8	Port -1.7
9	RST- Restart*(The RESET pin is an input pin and it is an active high pin. When a high pulse is applied to this pin the microcontroller will reset and terminate all activities. Upon reset all the registers except PC will reset to 0000 Value and PC register will reset to 0007 value.)*
10	Multiplexed pin: port 3.0/ RxD(*Receive serial input data*)
11	Multiplexed pin: port 3.1/TxD (*Transmit serial out data*)
12	Multiplexed pin: port 3.2/ INT0 (*External interrupt 0*)

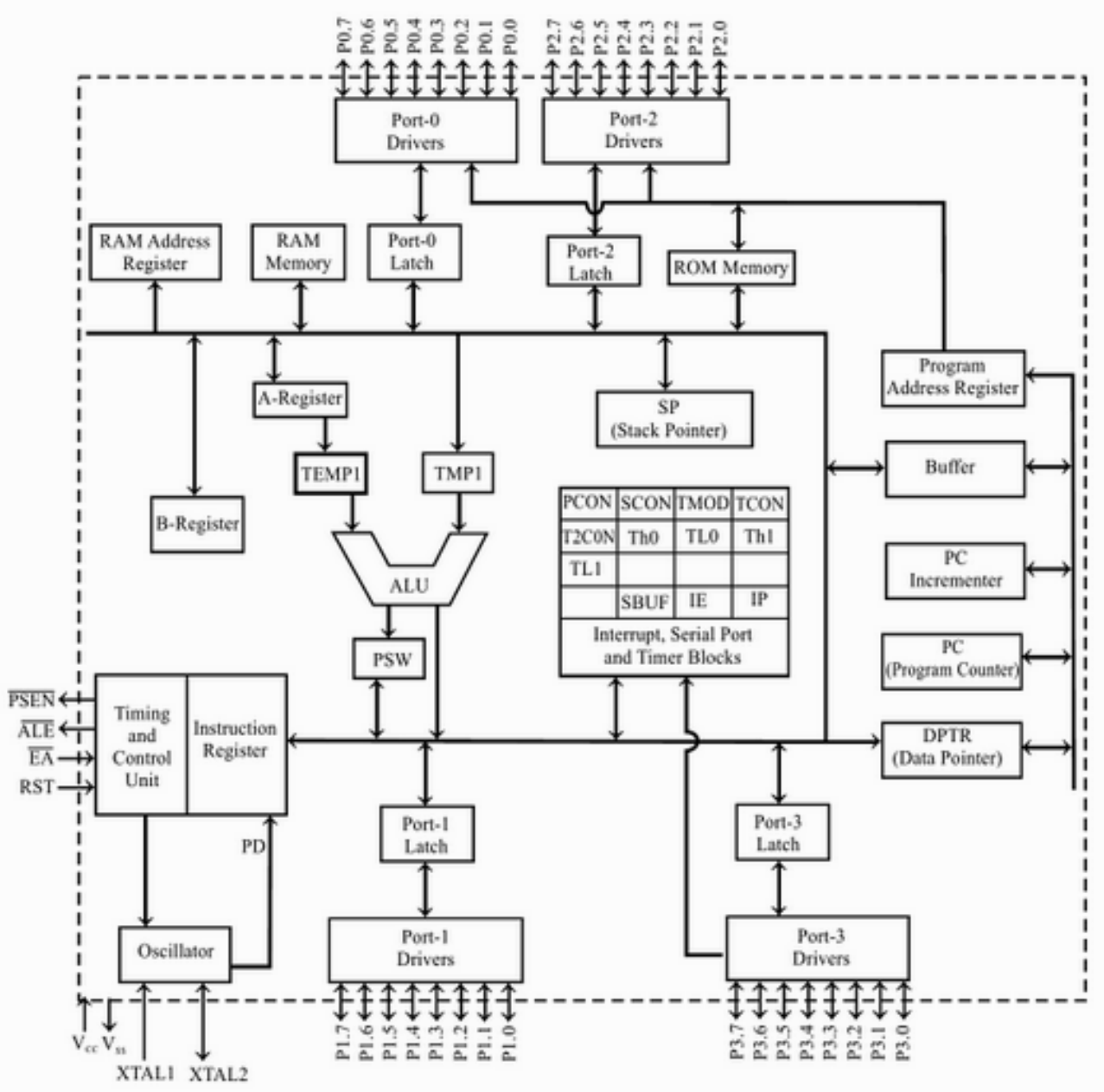
13	Multiplexed pin: port 3.3/ INT1 (*External interrupt 1*)
14	Multiplexed pin: port 3.4/ T0(*Clock to timer 0*)
15	Multiplexed pin: port 3.5/ T1(*Clock to timer 1*)
16	Multiplexed pin: port 3.6/ WR (*write control signal*)
17	Multiplexed pin: port 3.7/RD (*Read control signal*)
18	XTAL2 (*pin to connect crystal oscillator*)
19	XTAL1 (*pin to connect crystal oscillator*)
20	GND (*Ground*)
21	Multiplexed pin: port 2.0/A8
22	Multiplexed pin: port 2.1/A9
23	Multiplexed pin: port 2.2/A10
24	Multiplexed pin: port 2.3/A11
25	Multiplexed pin: port 2.4/A12
26	Multiplexed pin: port 2.5/A13
27	Multiplexed pin: port 2.6/A14
28	Multiplexed pin: port 2.7/A15
29	PSEN' - Program Stobe enable*(This outpin goes low when microcontroller accessing program code from external memeory, this pin is connected to output enable pin of the ROM.)*
30	ALE/PROG' – Address Latch Enable *(Used to demultiplex address and data bus)*
31	EA'- External Access (*This pin isan active low pin. This pin is connected to ground when microcontroller is accessing the program code stored in the external memory and connected to Vcc when it is accessing the program code in the on chip memory. This pin should not be left unconnected*.)
32	Multiplexed pin: Port 0.0/ AD0
33	Multiplexed pin: Port 0.1/ AD1
34	Multiplexed pin: Port 0.2/ AD2
35	Multiplexed pin: Port 0.0/ AD3
36	Multiplexed pin: Port 0.0/ AD4
37	Multiplexed pin: Port 0.0/ AD5
38	Multiplexed pin: Port 0.0/ AD6
39	Multiplexed pin: Port 0.0/ AD7
40	Vcc-Power supply

Architecture of 8051 Microcontroller

- The 8051 is based on an 8-bit CISC core with Harvard architecture. Its 8-bit architecture is optimized for control applications with extensive Boolean processing.
- 8051 is a 8-bit microcontroller, means MC 8051 can Read, Write and Process 8 bit data at a time.

Main blocks in the architecture of 8051 are as follows:

a) Internal Memory
   a. Internal RAM
   b. Internal ROM
b) The Stack and Stack pointer
c) Registers
d) ALU, A and B-Register
e) I/O Port pins, Ports and Circuits
f) Timers / Counters
g) Serial Data Communication
h) PC-Program Counter
i) Flag and PSW Register
j) Data pointer (DPTR)
k) Data and Address Bus



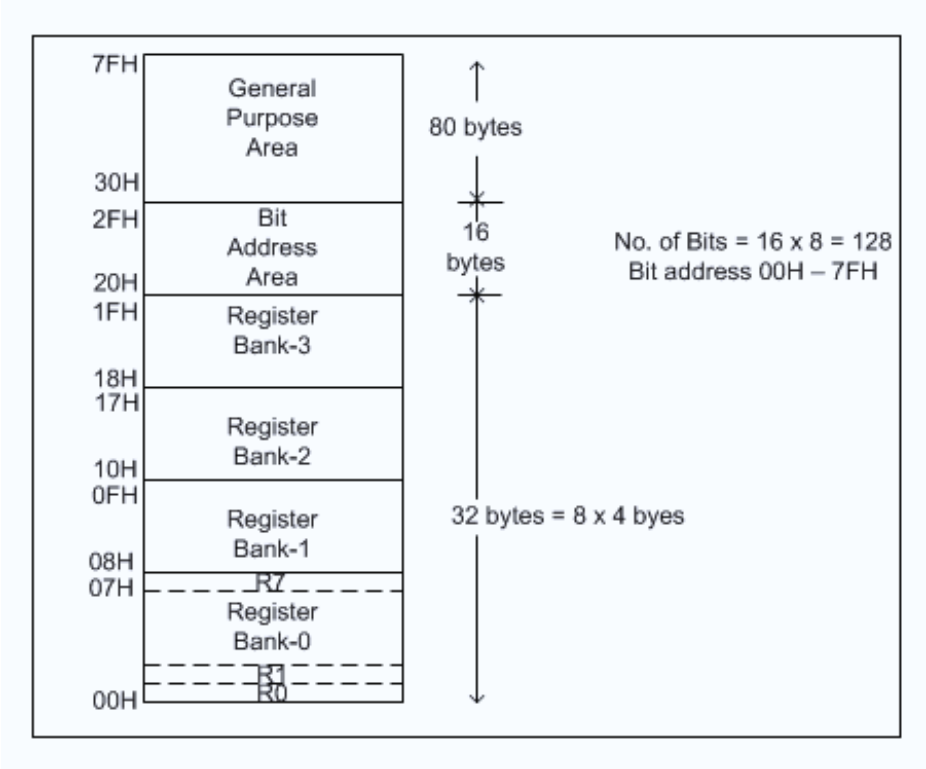
a)Internal Memory

- A functioning computer memory for program code bytes, commonly in ROM and RAM memory for variable data that can be altered as the program runs. Additional memory can be added externally using suitable circuits.
- 8051 has Harvard architecture which uses the same address in different memories for code and data The internal circuitry accesses the current memory based on the nature of operation in the program.

Internal RAM: The 128 bytes internal RAM is organized into 3 distinct areas.

- 32 bytes from address 00h to 1FH that make up 32 working registers organized as 4 memory banks of 8 registers each. The 4 register banks are numbered 0 to 3 and are made up of 8 registers named R0 to R7. Each register can be addressed by name or by its RAM addresses. Thus R0 of bank3 is R0 (if bank3 is selected) or address 18H (where bank3 is selected). Bits RS0 and RS1 in the PSW determine which bank of registers is currently in use at any time when program is running. Register banks not selected can be used as

general purpose RAM. Bank0 is selected by default on reset.



- A bit addressable area of 16 bytes occupies RAM byte addresses 20H to 2F H, forming total of 128 bits. An addressable bit may be specified by its bit address of 00h to 7F H or 8 bits may form any byte address from 20 H to 2F H. For example bit address 4F H is also bit 7 of byte address 29 H. Addressable bits are useful when the program need only remember a binary event.
- A general purpose RAM area above the bit area from 30H to 7F H, addressable as byte.

Internal ROM

- A code of 4K memory is incorporated as on-chip ROM in 8051. The 8051 ROM is a non-volatile memory meaning that its contents cannot be altered and hence has a similar range of data and program memory, i.e, they can address program memory as well as a 64K separate block of data memory.
- 8051 is organized so that data memory and program code memory can be two entirely different physical memory entities. Each has the same address ranges. The internal program ROM occupies code address space 0000H to 0FFFH. The PC is normally used to address program code bytes from address 0000H to FFFFH. Program addresses higher than 0FFFH which exceed the internal ROM capacity will cause the 8051 to automatically fetch code bytes from external memory, addresses 00H to FFFFH by connecting the external access pin (EA) to ground.

b)The Stack and Stack pointer:

- The stack refers to an area of internal RAM that is used in conjunction with certain opcodes to store and retrieve data quickly. The 8 bit Stack Pointer (SP) register is used by the 8051 to hold internal RAM address that is called the top of the stack. The address in SP register is the location in internal RAM where the last byte of the data was stored by stack operation.
- When data is to be placed on the stack , the SP increments before storing data on the stack so that the stack grows up as data is stored. Whenever data is retrieved from the stack, the

byte is read from the stack and then the SP decrements to point to the next available byte of stored data.

c) Register :

Registers are usually known as data storage devices. 8051 microcontroller has 2 registers, namely Register A and Register B. Register A serves as an accumulator while Register B functions as a general purpose register. These registers are used to store the output of mathematical and logical instructions.

The operations of addition, subtraction, multiplication and division are carried out by Register A. Register B is usually unused and comes into picture only when multiplication and division functions are carried out by Register A. Register A also involved in data transfers between the microcontroller and external memory.

8051 microcontroller also has 7 Special Function Registers (SFRs). They are:

1. Serial Port Data Buffer (SBUF)
2. Timer/Counter Control (TCON)
3. Timer/Counter Mode Control (TMOD)
4. Serial Port Control (SCON)
5. Power Control (PCON)
6. Interrupt Priority (IP)
7. Interrupt Enable Control (IE)

d)ALU — Arithmetic Logical Unit, A-Accumulator and B-register:

- ALU unit is used for the arithmetic and logical calculations and it can generally process 8-bit data. A- register is used for arithmetic operations. This is also bit addressable and 8 bitregister. B- register is used in only two instructions MUL AB and DIV AB. This is alsobit addressable and 8 bit register.

e)I/O Port pins, Ports and Circuits

PORT P0: When there is no external memory present, this port acts as a general purpose input/output port. In the presence of external memory, it functions as a multiplexed address and data bus. It performs a dual role.

PORT P1: This port is used for various interfacing activities. This 8-bit port is a normal I/O port i.e. it does not perform dual functions.

PORT P2: Similar to PORT P0, this port can be used as a general purpose port when there is no external memory but when external memory is present it works in conjunction with PORT PO as an address bus. This is an 8-bit port and performs dual functions.

PORT P3: PORT P3 behaves as a dedicated I/O port

f)Timers / Counters :

8051 has two 16-bit programmable UP timers/counters. They can be configured to operate either as timers or as event counters. The names of the two counters are T0 and T1 respectively

Synchronization among internal operations can be achieved with the help of clock circuits which are responsible for generating clock pulses. During each clock pulse a particular operation will be carried out, thereby, assuring synchronization among operations. For the

formation of an oscillator, we are provided with two pins XTAL1 and XTAL2 which are used for connecting a resonant network in 8051 microcontroller device. In addition to this, circuit also consists of four more pins. They are,

Internal operations can be synchronized using clock circuits which produce clock pulses. With each clock pulse, a particular function will be accomplished and hence synchronization is achieved. There are two pins XTAL1 and XTAL2 which form an oscillator circuit which connect to a resonant network in the microcontroller. The circuit also has 4 additional pins -

1. EA: External enable
2. ALE: Address latch enable
3. PSEN: Program store enable and
4. RST: Reset.

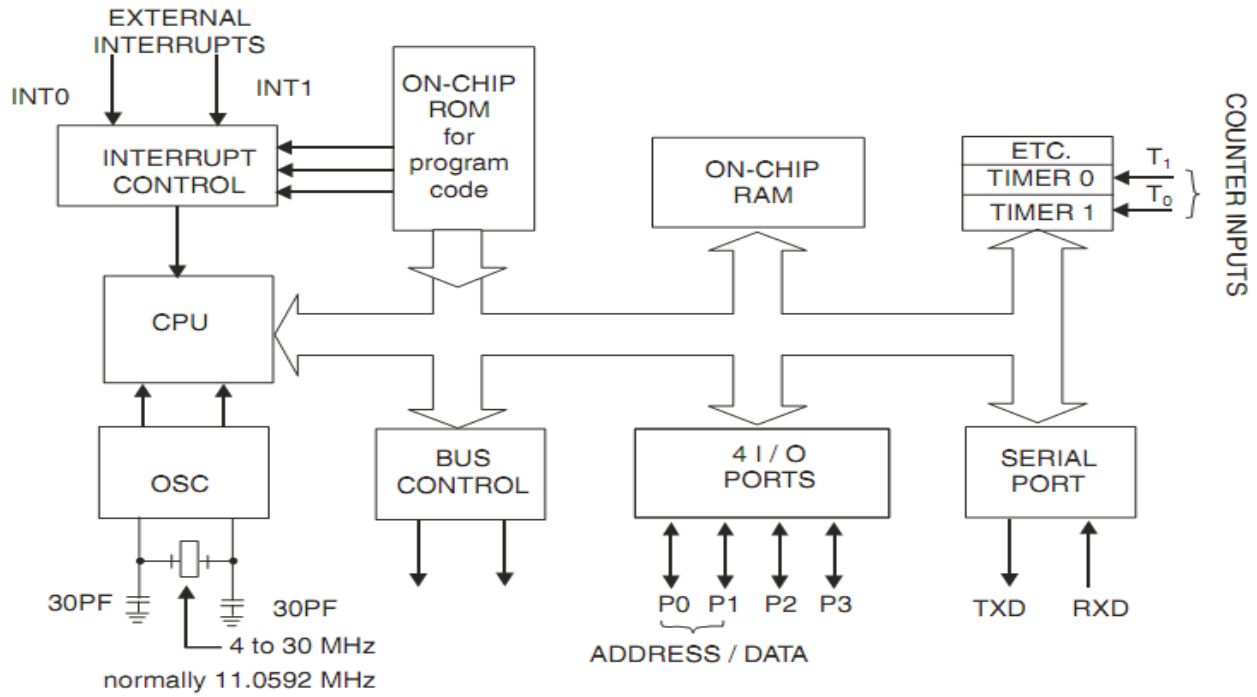
g)Serial Data Communication

A method of establishing communication among computers is by transmitting and receiving data bits is a serial connection network. In 8051, the SBUF (Serial Port Data Buffer) register holds the data; the SCON (Serial Control) register manages the data communication and the PCON (Power Control) register manages the data transfer rates. Further, two pins - RXD and TXD, establish the serial network.

The SBUF register has 2 parts – one for storing the data to be transmitted and another for receiving data from outer sources. The first function is done using TXD pin and the second function is done using RXD pin.

There are 4 programmable modes in serial data communication. They are:

1. Serial Data mode 0 (shift register mode)
2. Serial Data mode 1 (standard UART)
3. Serial Data mode 2 (multiprocessor mode)
4. Serial Data mode 3



Architecture of 8051 microcontroller

h)PC-Program Counter

- Points to the address of next instruction to be executed from ROM
- It is 16 bit register means the 8051 can access program address from0000H to FFFFH. A total of 64KB of code.
- Initially PC has 0000H.ORG instruction is used to initialize the PC ORG 0000H means PCinitialize by 0000H
- The basic function of program counter is to fetch from memory the address of the next instruction to be executed. The PC holds the address of the next instruction residing in memory and when a command is encountered, it produces that instruction. This way the PC increments automatically, holding the address of the next instruction.

i) Flag Bits and PSW Register

- Flag register in 8051 is called as program status word (PSW). Used to indicate the Arithmetic condition of ACC.This special function register PSW is also bit addressable and 8 bit wide means each bit can be set or reset independently.

PSW0.7	PSW0.6	PSW0.5	PSW0.4	PSW0.3	PSW0.2	PSW0.1	PSW0.0
CY	AC	F0	RS1	RS0	OV	—	P

FLAG Register

The bits PSW3 and PSW4 are denoted as RS0 and RS1 and these bits are used to select the bank registers of the RAM location. The meaning of various bits of PSW register is shown below.

CY	PSW0.7	Carry Flag
AC	PSW0.6	Auxiliary Carry Flag
F0	PSW0.5	Flag 0 available for general purpose.

RS1	PSW0.4	Register Bank select bit 1
RS0	PSW0.3	Register bank select bit 0
OV	PSW0.2	Overflow flag
---	PSW0.1	User definable flag
P	PSW0.0	Parity flag .set/cleared by hardware.

j)Data Pointer (DPTR) in 8051 :

- 16 bit register, it is divided into two parts DPH and DPL,
- DPH for Higher order 8 bits, DPL for lower order 8-bits
- DPTR, DPH, DPL these all are SFRs in 8051.

K)Data and Address Bus

A bus is group of wires using which data transfer takes place from one location to another within a system. Buses reduce the number of paths or cables needed to set up connection between components.

There are mainly two kinds of buses - Data Bus and Address Bus

Data Bus: The purpose of data bus is to transfer data. It acts as an electronic channel using which data travels. Wider the width of the bus, greater will be the transmission of data.

Address Bus: The purpose of address bus is to transfer information but not data. The information tells from where within the components, the data should be sent to or received from. The capacity or memory of the address bus depends on the number of wires that transmit a single address bit.

XTAL1,XTAL2:

- These two pins are connected to Quartz crystal oscillator which runs the on-chip oscillator. The quartz crystal oscillator is connected to the two pins along with a capacitor of 30pF as shown in the circuit. If we use a source other than the crystal oscillator, it will be connected to XTAL1 and XTAL2 is left unconnected.

- The crystal of frequency varies from 4MHz to 30 MHz can be used, normally weuse 11.0592 MHz frequency.

ADDRESSING MODES OF 8051

The way in which the data operands are accessed by different instructions is known as the addressing modes. There are various methods of denoting the data operands in the instruction. The 8051 microcontroller supports mainly 5 addressing modes.

They are

a) Immediate addressing mode
b) Direct Addressing mode
c) Register addressing mode
d) Register Indirect addressing mode
e) Indexed addressing mode

a)Immediate addressing mode:

The addressing mode in which the data operand is a constant and it is a part of the instruction itself is known as Immediate addressing mode. Normally the data must be preceded by a

sign. This addressing mode can be used to transfer the data into any of the registers including DPTR.

Ex: MOV A, # 27 H : The data (constant) 27 is moved to the accumulator register
ADD R1, #45 H: Add the constant 45 to the contents of the accumulator
MOV DPTR, # 8245H: Move the data 8245 into the data pointer register.
MOV P1, #21 H

b)Direct addressing mode:

The addressing mode in which the data operand is in the RAM location (00 -7FH) and the address of the data operand is given in the instruction is known as Direct addressing mode. The direct addressing mode uses the lower 128 bytes of Internal RAM and the SFRs

Ex: MOV R1, 42H : Move the contents of RAM location 42 into R1 register
MOV 49H, A: Move the contents of the accumulator into the RAM location 49.
ADD A, 56H: Add the contents of the RAM location 56 to the accumulator

c)Register addressing mode:

The addressing mode in which the data operand to be manipulated lies in one of the registers is known as register addressing mode.

Ex: MOV A, R0: Move the contents of the register R0 to the accumulator
ADD A, R6: Add the contents of R6 register to the accumulator
MOV P1, R2: Move the contents of the R2 register into port 1
MOV R5, R2: This is invalid .The data transfer between the registers is not allowed.

d)Register indirect addressing mode:

The addressing mode in which a register is used as a pointer to the data memory block is known as Register indirect addressing mode.

Ex: MOV A, @ R0: Move the contents of RAM location whose address is in R0 into **A** (accumulator)
MOV @ R1, B: Move the contents of B into RAM location whose address is held by R1 When R0 and R1 are used as pointers, they must be preceded by @ sign

One of the advantages of register indirect addressing mode is that it makes accessing the data more dynamic than static as in the case of direct addressing mode.

e)Indexed addressing mode:

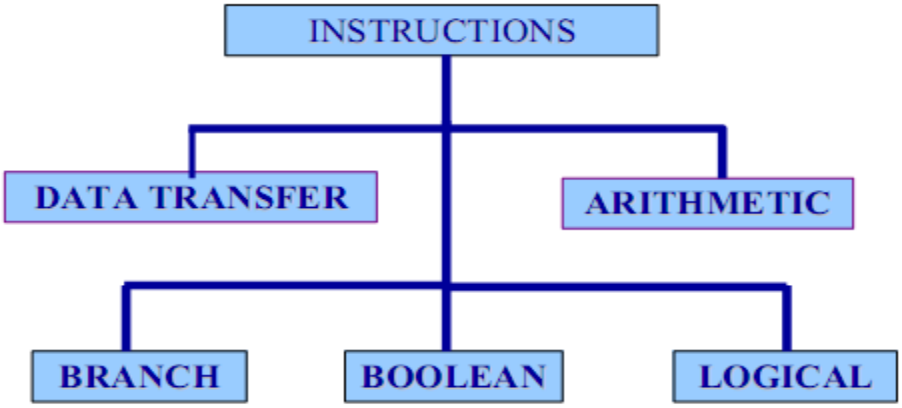
This addressing mode is usedin accessing the data elements of lookup table entries located in program ROM space of 8051.

Ex: 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. Here C denotes code .In this instruction the contents of An are added to the 16-bit DPTR register to form the 16-bit address of the data operand.

INSTRUCTION SET OF 8051

Classification of Instructions



Data transfer Instructions

- Mov A, Rn
- Mov A, Direct
- Mov A, @Ri
- Mov A, #Data8
- Mov Dptr, #Data16
- Mov Rn, A
- Mov Rn, Direct
- Mov Rn, #Data**8**
- Mov Direct, A
- Mov Direct, Rn
- Mov Direct, #Data8
- Mov Direct, Direct
- Mov Direct, @Ri
- Mov Direct, # Data**8**
- Mov @Ri, A
- Mov @Ri, Direct
- Mov @Ri, #Data**8**
- Movx A, @Ri
- Movx A, @Dptr
- Movx @Ri, A
- Movx @dptr, A

- Movc A, @A+Dptr
- Movc A, @A+Pc
- Push Direct
- Pop Direct
- Xch A, Rn
- Xch A, Direct
- Xch A, @Ri
- Xchd A, @Ri

Boolean Instructions

- Clr C
- Clr Bit
- Setb C
- Setb Bit
- Cpl C
- Cpl Bit
- Anl C, Bit
- Anl C, /Bit
- Orl C, Bit
- Orl C, /Bit
- Mov C, Bit
- Mov Bit, C

Branch Instructions

- Jc Reladdr
- Jnc Reladdr
- Jb Bit, Reladdr
- Jnb Bit, Reladdr
- Jbc Bit, Reladdr

Arithmetic Instructions

- Add A, Rn
- Add A, Direct
- Add A, @Ri
- Add A, #Data**8**

- Addc A, Rn
- Addc A, Direct
- Addc A, @Ri
- Addc A, #Data**8**
- Subb A, Rn
- Subb A, Direct
- Subb A, @Ri
- Subb A, #Data**8**
- Inc A
- Inc Rn
- Inc Direct
- Inc @Ri
- Inc Dptr
- Dec A
- Dec Rn
- Dec Direct
- Dec @Ri
- Mul AB
- Div AB
- DA A

Logical Instructions

- Anl A, Rn
- Anl A, Direct
- Anl A, @Ri
- Anl A, #Data**8**
- Anl Direct, A
- Anl Direct, #Data**8**
- Orl A, Rn
- Orl A, Direct
- Orl A, @Ri
- Orl A, #Data**8**
- Orl Direct, A

- Orl Direct, #Data**8**
- Xrl A, Rn
- Xrl A, Direct
- Xrl A, @Ri
- Xrl A, #Data**8**
- Xrl Direct, A
- Xrl Direct, #Data**8**
- Clr A
- Cpl A
- Rl A
- Rlc A
- Rr A
- Rrc A
- Swap A

Branch Instructions

- Acall Addr11
- Lcall Addr16
- Ret
- Reti
- Ajmp Addr11
- Ljmp Addr16
- Sjmp Reladdr
- Jmp @A+Dptr
- Jz Reladdr
- Jnz Reladdr
- Cjne Rn, #Data, Reladdr
- Cjne @Ri, #Data, Reladdr
- Cjne A, #Data, Reladdr
- Cjne A, Direct, Reladdr
- Djnz Rn, Reladdr
- Djnz Direct, Reladdr
- Nop

8051 ASSEMBLY LANGUAGE PROGRAMMING:

a) A program to find sum of N natural numbers and store the sum

Program description:- The number "N" is stored in location 35H. Natural numbers generated from 0 to N must be stored from location 55H. The sum of natural numbers must be stored in location 36H.

Analyzing the program description, we need 3 registers. R0 to store the value of "N" (given in location 35H)and to act as a counter for generating natural numbers up to N. R5 is used to save the value of first storage location of natural numbers and then R5 is incremented by one each to store each newly generated natural number. R7 is initiated as 0 and is incremented by 1 to generate natural numbers.

The Program:-

MOV PSW, #00H / Register bank '0' is selected by executing this instruction.

MOV R0, 35H // the value of 'N' stored in location 35H is transferred to R0.

MOV R5, #55H // The starting location for storing natural numbers '#55H' is transferred to R5

MOV A, #00H // Accumulator is initiated with value 0 for adding natural numbers cumulatively.

MOV R7, #00H // R7 is initialized to '0' to generate natural numbers. Note: '0' is not a natural number.

LOOP: INC R7 // R7 is incremented by 1 to generate next natural number.

MOV @R5, 07H// this is indirect addressing mode used here.It is not possible to transfer data from one register to another register directly. So an instruction like MOV R5, R7 is invalid.Instead we use the direct address (07) of register R7 of register bank #00 to transfer the generated natural number to it's storage location in register R5.Indirect addressing is used as we need to save the generated natural number directly to memory address. R5 holds the starting location address (of the storage area) as its value i.e #55H.By indirectly addressing, we can save what ever value in R7 directly to location #55H.

INC R5 // The storage location is incremented by 1 from #55H to #56H to store the next generated natural number

ADD A, R7 // the generated natural number is added to contents in accumulator.

DJNZ R0, LOOP // the value of register Ro (value of 'N') is decremented by 1. It is checked against stopping condition zero. If it's R0 is not equal to zero, the program

control will move to label LOOP again and the steps from INC R7 will be executed again until R0 is equal to zero. When R0 is equal to zero, program control will exit the loop and move to next instruction given below.

MOV 36H,A // the sum of natural numbers in accumulator is moved to storage location 36H.

STOP: SJMP STOP // an infinite loop written at the end of the program. When this instruction is reached program control will get stuck at this instruction as it's an infinite loop.To get out of this infinite loop system reset must be applied.

b) A simple program to copy a block of data from one location to another

Program Description:- 10 bytes of data stored from 30H is to be copied to another location staring from 50H.

Analyzing the program, two registers are needed to store starting locations of source and destination. Let's say we take R0 as the source and R2 as the destination registers. Now we need a counter to count 10 bytes of data transferred from source to destination. Let's take R3 for that. It is not possible to transfer data from one register to another register directly by using any kind of addressing mode. So we need accumulator to stand in between as a temporary register. So here is it:

The Program:

MOV R0,#30H // Address of the starting location of source data is moved to R0.

MOV R1,#50H // Address of the starting location of destination is moved to R1

MOV R3,#OAH // Set the counter R3 with 10. You can also use decimal number as MOV R3,#10d.

LOOP: MOV A,@R0// Indirect addressing mode is used. Contents at the location of Ro (30H) are copied to accumulator.

MOV @R1, A // Contents in accumulator is copied to location pointed by Ra (that is 50H).

INC R0 // Ro is incremented by 1 to point to next location.

INC R1 // R1 is incremented by 1 to point to next location.

DJNZ R3, LOOP // Counter register R3 is decremented by 1 and checked against zero. See the explanation DJNZ in the first program "sum of natural numbers"

STOP: SJMP STOP // Infinite loop to terminate program

c) To generate a Fibonacci series

Program description:-A Fibonacci series is an infinite series in mathematics that goes like 0,1,1,2,3,5,8,13,21….

Write a program which generates Fibonacci series up to N terms. The value of N is available in location 30H and to save the series from location 40H.

The Program:-

```
BEGIN: MOV R1,30H // Getting the value of "N"

MOV R7,#40H          // The first number '0' of series is stored here.

MOV @R7,#00H         // Loading value 'o' to address 40H using indirect addressing

INC R7               // Incrementing value of R7 from 40H to 41H to store next number '1'

MOV @R7, #01H // Storing value '1' to location 41H. Note that 'o' and '1' are seed values of a
                 Fibonacci series and has to be generated manually.

MOV R5,#42H // New register R5 is loaded with location address 42H to store next values of series,
               generated by adding the 2 previously generated numbers.

DEC R1               // the count value "N" is decremented by 2, as we have already generated and
                        stored 1st two numbers.

DEC R7               // R7 is now reduced from 41H to 40H. We need to add contents of 40H and
                        41 H to get the number that is to be stored in 42H.

LOOP: MOV A, @R7 // Contents in R7 is moved to accumulator.

INC R7               // R7 is incremented to get the next value.

ADD A,@R7            // The two values are added and stored in Acc.

MOV @R5, A          //The newly generated value of the series is stored in the address held by R5.

INC R5               // R5 is incremented to store next value.

DJNZ R1,LOOP         // The count "N" is checked to zero (to know if all the numbers up to N are
                        generated).

STOP: SJMP STOP      // Run infinitely here or end of program execution.
```

Other Programs in 8051

Write a program to −

- Load the accumulator with the value 55H.
- Complement the ACC 700 times.

Solution − Since 700 is greater than 255 (the maximum capacity of any register), two registers are used to hold the count. The following code shows how to use two registers, R2 and R3, for the count.

```
   MOV A,#55H            ;A = 55H
NEXT: MOV R3,#10         ;R3 the outer loop counter

AGAIN:MOV R2,#70         ;R2 the inner loop counter

   CPL A                 ;complement
```

I/O ports in 8051 microcontroller

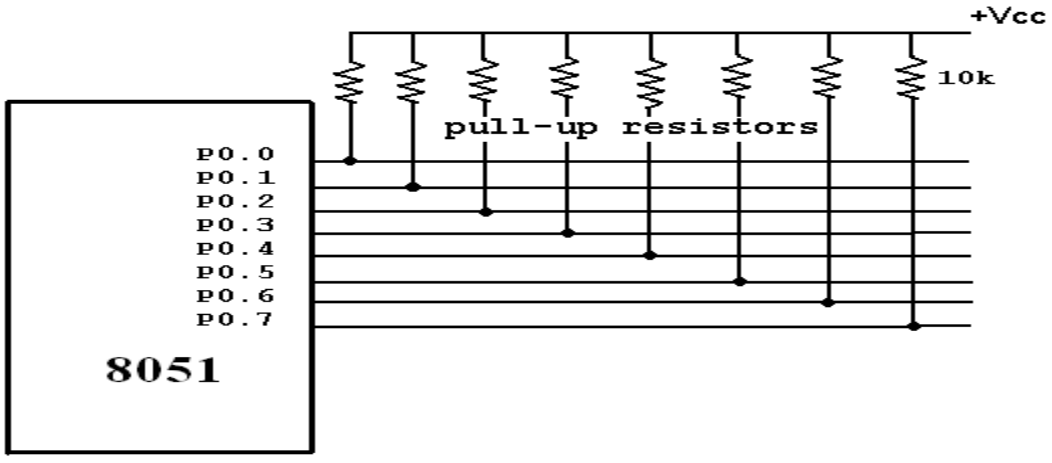
PARALLEL I /O PORTS :

The 8051 microcontroller has four parallel I/O ports , each of 8-bits .So, it provides the user 32 I/O lines for connecting the microcontroller to the peripherals. The four ports are P0 (Port 0), P1(Port1) ,P2(Port 2) and P3 (Port3). Upon reset all the ports are output ports. In order to make them input, all the ports must be set i.e a high bits must be sent to all the port pins. This is normally done by the instruction "SETB".

Ex: MOV A,#0FFH ; A = FF
MOV P0,A ; make P0 an input port

PORT 0:

Port 0 is an 8-bit I/O port with dual purpose.If external memory is used, these port pins are used for the lower address byte address/data (AD_0-AD_7), otherwise all bits of the port are either input or output. Unlike other ports, Port 0 is not provided with pull-up resistors internally ,so for PORT-0 pull-up resistors of nearly 10k are to be connected externally as shown in the Fig.



Port-0 with pull-up resistors

Port 0 can also beused as address/data bus (AD0-AD7), allowing it to be used for both address and data. When connecting the 8051 to an external memory, port 0 provides both address and data. The 8051 multiplexes address and data through port 0 to save the pins. ALE indicates whether P0 has address or data. When ALE = 0, it provides data D0-D7, and when ALE =1 it provides address and data with the help of a 74LS373 latch.

Port 1:

Port 1 occupies a total of 8 pins (pins 1 through 8). It has no dual application and acts only as input or output port. In contrast to port 0, this port does not need any pull-up resistors since pull-up resistors connected internally. Upon reset, Port 1 is configured as an output port. To configure it as an input port, port bits must be set i.e a high bit must be sent to all the port pins. This is normally done by the instruction "SETB".

For Ex:

```
MOV A,#0FFH   ;A=FF HEX
 MOV P1,A     ;make P1 an input port by writing 1's to all of its pins
```

Port 2 :

Port 2 is also an eight bit parallel port. (pins 21- 28). It can be used as input or output port.As this port is provided with internal pull-up resistors it does not need any external pull-up resistors. Upon reset,Port 2 is configured as an output port. If the port is to be used as input port, all the port bits must be made high by sending FF to the port. For ex,

```
MOV A,#0FFH        ;A=FF hex
 MOV P2,A          ;make P2 an input port by writing all 1's to it
```

Port2 lines are also associated with the higher order address lines A8-A15. In systems based on the 8751, 8951, and DS5000, Port2 is used as simple I/O port.. But, in 8031-based systems, port 2 is used along with P0 to provide the 16-bit address for the external memory.Since an 8031 is capable of accessing 64K bytes of external memory, it needs a path for the 16 bits of the address. While P0 provides the lower 8 bits via A0-A7, it is the job of P2 to provide bits A8-A15 of the address. In other words, when 8031 is connected to external memory, Port 2 is used for the upper 8 bits of the 16 bit address, and it cannot be used for I/O operations.

PORT 3:

Port3 is also an 8-bit parallel port with dual function.(pins 10 to 17). The port pins can be used for I/O operations as well as for control operations.The details of these additional operations are given below in the table. Port 3 also do not need any external pull-up resistors as they are provided internally similar to the case of Port2 & Port 1. Upon reset port 3 is configured as an output port . If the port is to be used as input port,all the port bits must be made high by sending FF to the port.

For EX,

```
MOV A,#0FFH        ;A=FF hex
 MOV P3,A          ;make P3 an input port by writing all 1's to it
```

Alternate Functions of Port3:

P3.0 and P3.1 are used for the RxD (Receive Data) and TxD (Transmit Data) serial communications signals. Bits P3.2 and P3.3 are meant for external interrupts. Bits P3.4 and P3.5 are

used for Timers 0 and 1 and P3.6 and P3.7 are used to provide the write and read signals of external memories connected in 8031 based systems. Table 3.3 show alternate function of port-3.

Alternate Functions of port-3

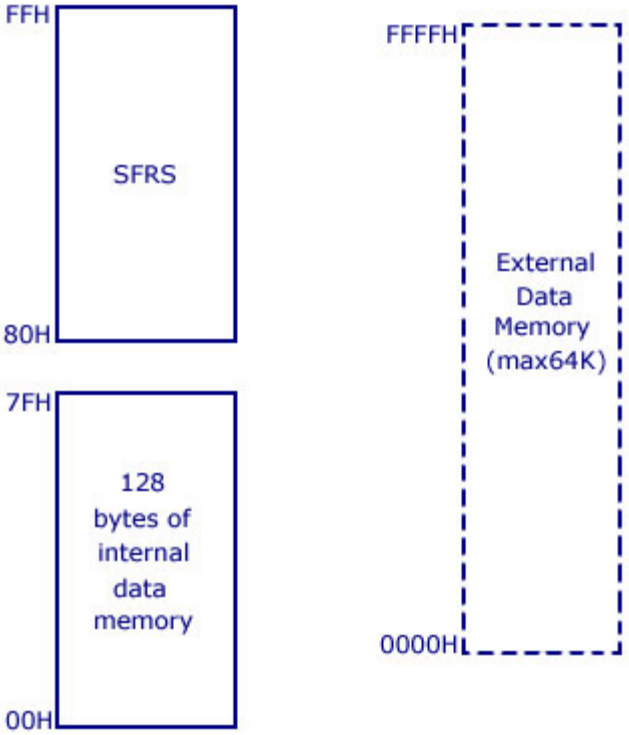
S.No	Port 3 bit	Pin No	Function
1	P3.0	10	RxD
2	P3.1	11	TxD
3	P3.2	12	$\overline{\text{INTO}}$
4	P3.3	13	$\overline{\text{INT1}}$
5	P3.4	14	T0
6	P3.5	15	T1
7	P3.6	16	$\overline{\text{WR}}$
8	P3.7	17	$\overline{\text{RD}}$

MEMORY ORGANIZATION in 8051.

- The 8051 microcontroller has 128 bytes of Internal RAM and 4kB of on chip ROM .
- The RAM is also known as Data memory and the ROM is known as program memory. The program memory is also known as Code memory.
- This Code memory holds the actual 8051 program that is to be executed. In 8051 this memory is limited to 64K .Code memory may be found on-chip, as ROM or EPROM.
- It may also be stored completely off-chip in an external ROM or, more commonly, an external EPROM.
- The 8051 has only 128 bytes of Internal RAM but it supports 64kB of external RAM. As the name suggests, external RAM is any random access memory which is off-chip.
- The memory is off-chip it is not as flexible in terms of accessing, and is also slower.

For example, to increment an Internal RAM location by 1,it requires only 1 instruction and 1 instruction cycle but to increment a 1-byte value stored in External RAM requires 4 instructions and 7 instruction cycles. So, here the external memory is 7 times slower.

Internal and External Data Memory of 8051

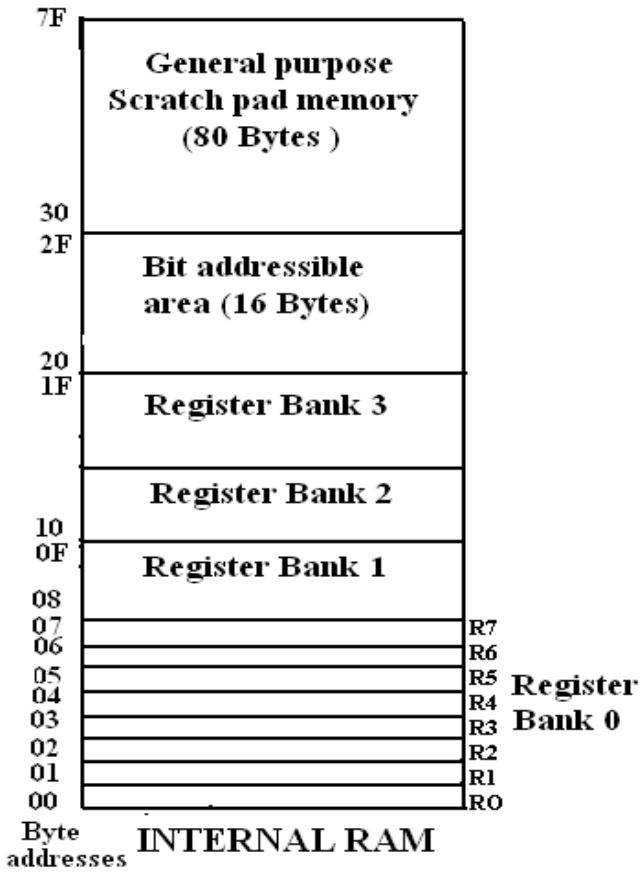


Internal RAM OF 8051:

This Internal RAM is found on-chip on the 8051 .So it is the fastest RAM available, and it is also the most flexible in terms of reading, writing, and modifying it's contents. Internal RAM is volatile, so when the 8051 is reset this memory is cleared. The 128 bytes of internal RAM is organization is shown below

i. Four register banks (Bank0, Bank1, Bank2 and Bank3) each of 8-bits (total 32 bytes). The default bank register is Bank0. The remaining Banks are selected with the help of RS0 and RS1 bits of PSW Register.
ii. 16 bytes of bit addressable area and
iii. 80 bytes of general purpose area (Scratch pad memory) as shown in the diagram below. This area is also utilized by the microcontroller as a storage area for the operating stack.
iv. The 32 bytes of RAM from address 00 H to 1FH are used as working registers organized as four banks of eight registers each.The registers are named as R0-R7 .Each register can be addressed by its name or by its RAM address.

For EX: MOV A,R7 or MOV R7,#05H



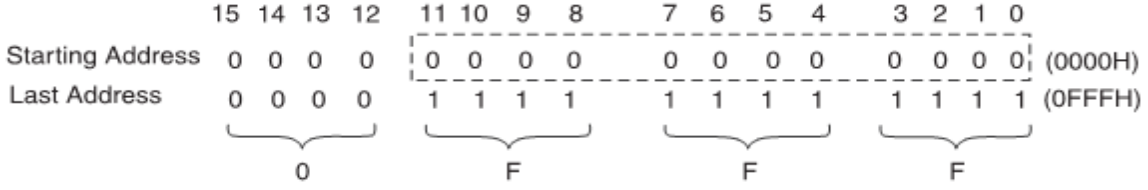
Internal RAM organization of 8051

On chip ROM

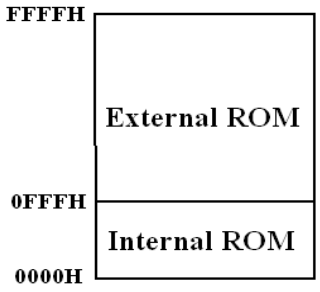
In 8051, 4KB read only memory (ROM) is available for program storage.This is used for permanent data storage or the data which is not changedduring the processing like the program or algorithm for specific applications.

- This is volatile memory; the data saved in this memory does not disappearafter power failure.
- We can interface up to 64KB ROM memory externally if the application islarge. These sizes are specified different by their companies.
- Address Range of PC: Address range of PC means program counter (whichpoints the next instruction to be executing) can be moved between theselocations or we can save the program from this location to this location.The address range can be calculated in the same way just like the RAMwhich is discussed in previous section

$$4\,\text{KB} = 2^2 \cdot 2^{10}\,\text{B}\,(\text{since } 1\text{KB} = 2^{10}\,\text{B})$$
$$= 2^{12}\,\text{Byte}$$



Address range of PC is 0000H to 0FFFH means total 4KB locations are availablefrom 0000H to 0FFFH.



SPECIAL FUNCTION REGISTERS (SFRs):

In 8051 microcontroller there certainregisters which uses the RAM addresses from 80H to FFH and they are meant for certain specific operations .These registers are called Special function registers (SFRs).Some of these registers are bit addressable also.

The list of SFRs and their functional names are given below.In these SFRs some of them are related to I/O ports (P0,P1,P2 and P3) and some of them are meant for control operations (TCON,SCON, PCON..) and remaining are the auxiliary SFRs, in the sense that they don't directly configure the 8051.

SFR and its Location

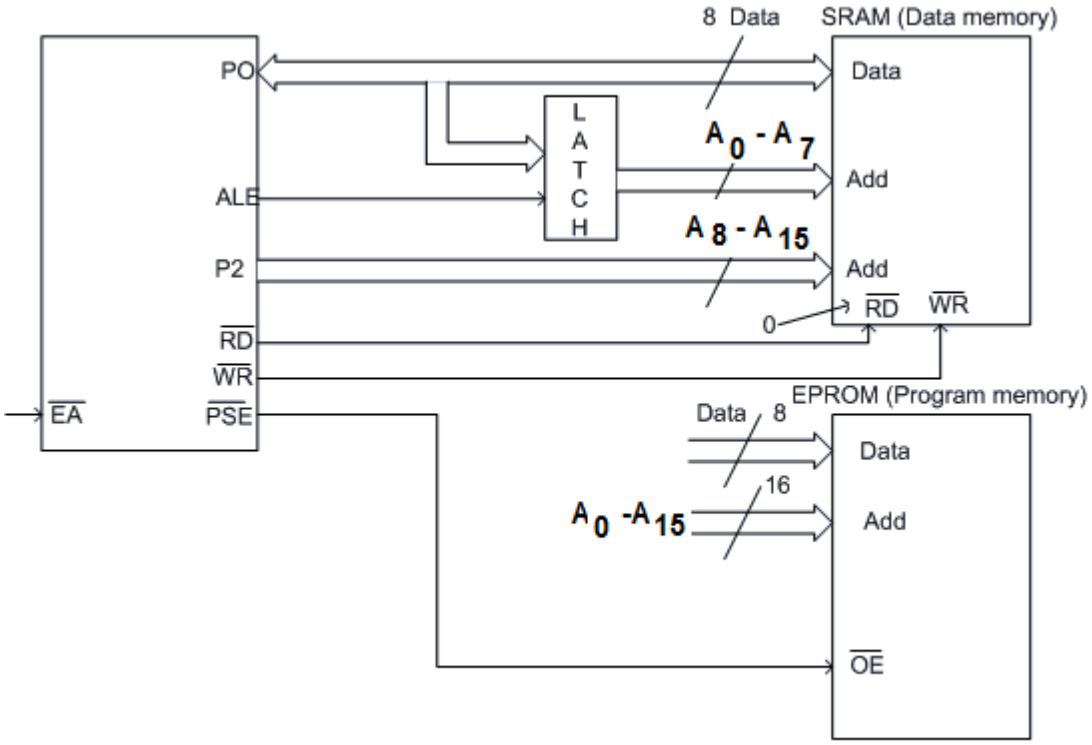
S.No	Symbol		Name of SFR
1	ACC*		Accumulator
2	B*		B-Register
3	PSW*		Program Status word register
4	SP		Stack Pointer Register
5	DPTR	DPL	Data pointer low byte
		DPH	Data pointer high byte
6	P0*		Port 0
	P1*		Port 1
8	P2*		Port 2
9	P3*		Port 3
10	IP*		Interrupt Priority control
11	IE*		Interrupt Enable control
12	TMOD		Tmier mode register
13	TCON*		Timer control register
14	TH0		Timer 0 Higher byte
15	TL0		Timer 0 Lower byte
16	TH1		Timer 1Higher byte
17	TL1		Timer 1 lower byte
18	SCON*		Serial control register
19	SBUF		Serial buffer register
20	PCON		Power control register

The * indicates the bit addressable SFRs

External memory and its interfacing with 8051

Interfacing External Memory

External program/data memory are to be interfaced, they are interfaced in the following way.



Circuit Diagram for Interfacing of External Memory

Since the PC (program counter) of the 8031/51 is 16-bit, it is capable of accessing up to 64K bytes of program code. In the 8031/51, port 0 and port 2 provide the 16-bit address to access external memory. Of these two ports, PO (Port 0) provides the lower 8 bit addresses AO – A7, and P2 provides the upper 8 bit addresses A8 – A15. More importantly, PO is also used to provide the 8-bit data bus DO – D7. In other words, pins PO.O – P0.7 are used for both the address and data paths. This is called address/data multiplexing in chip design. Of course the reason Intel used address/data multiplexing in the 8031/51 is to save pins.

ALE is an output pin for the 8051 microcontroller.

- when ALE = 0 the 8031 uses PO for the data path
- when ALE = 1, it uses it for the address path

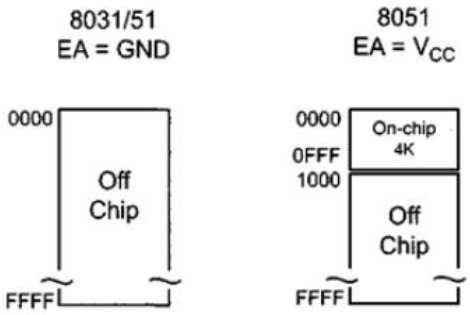
PSEN(PSE) is an output signal for the 8051 microcontroller and must be connected to the OE pin (output enable)of a ROM containing the program code. In other words, to access external ROM containing program code, the 8051 uses the PSEN signal. It is important to emphasize the role of EA and PSEN

.

External program memory is fetched by either of the following two conditions are satisfied.

1. EA(Enable Address) is low. The microcontroller by default starts searching for program from external program memory.
2. PC is higher than FFFH for 8051

PSEN(program store enable) tells the outside world whether the external memory fetched is program memory or data memory. EA is user configurable. PSEN is processor controlled.



- EA pin to V_{cc} to indicate that the program code is stored in the microcontroller's on-chip ROM.
- To indicate that the program code is stored in external ROM, this pin must be connected to GND

COUNTERS AND TIMERS of 8051 Microcontroller

The 8051 microcontroller has two 16-bit timers Timer 0 (T0) and Timer 1(T1) which can be used either to generate accurate time delays or as event counters. These timers are accessed as two 8-bit registers TLO, THO & TL1 ,TH1 because the 8051 microcontroller has 8-bit architecture.

Counting/timing device as timer:

A device for timing when the inputs to counting are given by a clock. The clock pulses are internally given at the specific time intervals in case of functioning as timer.

Counting/timing device as counter:

A device for counting when the inputs to count are given externally. Counter is given the input to count from external input pin.

Counting/timing device External controls for activation or deactivation of running:

When timing or counting devices is externally controlled by the gate input, when GT0 or GT1 is externally activated the device can function else it deactivates in gate input mode. GT0 or GT1 signals are given at P3.2 and P3.3.

Counting/timing device External count inputs in counter mode:

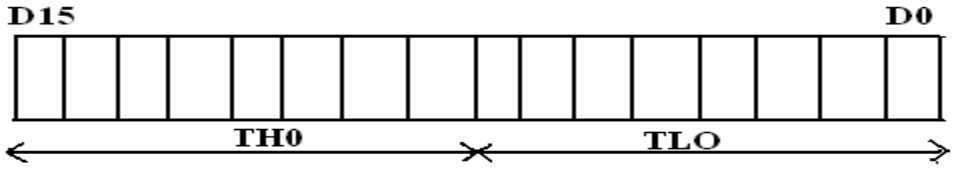
T0 counts the T0 is given the input to count from external input pin T0at P3.4.T1 counts when T1 is given the input to count from external input pin T1at P3.5.

Four Port P3 Pin functions:

P3.2, P3.3, P3.4 and P3.5 When TMOD SFR bits 3, 7, 2 and 6 set = 1, function asGT0 (gate for starting/stopping T1), GT1 (gate for starting/stopping T1), T0 (count input to T0) and T1 (count input to T1) inputs, respectively

TIMER 0:

The Timer 0 is a 16-bit register and can be treated as two 8-bit registers (TL0 & TH0) and these registers can be accessed similar to any other registers like A,B or R1,R2,R3 etc…
Ex: The instruction Mov TL0, #07 moves the value 07 into lower byte of Timer0.
Similarly Mov R5, TH0 saves the contents of TH0 in the R5 register.

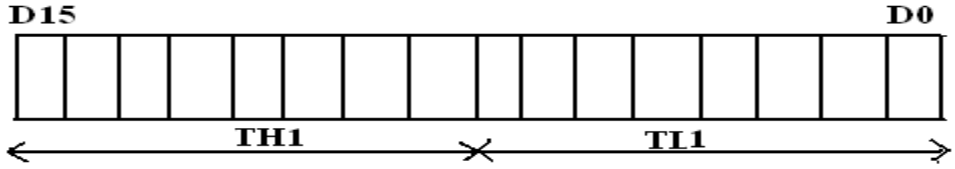


TIMER 1:

The Timer 1 is also a 16-bit register and can be treated as two 8-bit registers (TL1 & TH1) and these registers can be accessed similar to any other registers like A,B or R1,R2,R3 etc…

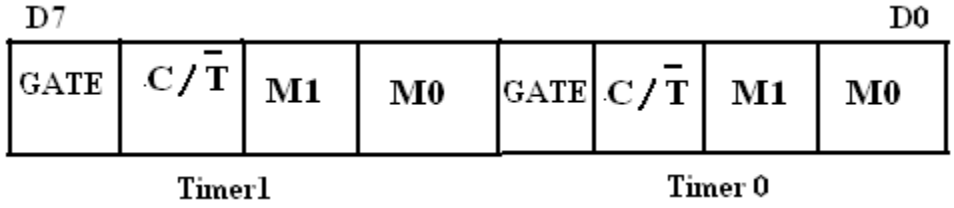
Ex: The instruction MOV TL1, #05 moves the value 05 into lower byte of Timer1.

Similarly MOV R0, TH1 saves the contents of TH1 in the R0 register



TMOD Register:

The various operating modes of both the timers T0 and T1 are set by an 8-bit register called TMOD register**.** In this TMOD register the lower 4-bits are meant for Timer 0 and the higher 4-bits are meant for Timer1.



GATE:

This bit is used to start or stop the timers by hardware .When GATE= 1, the timers can be started / stopped by the external sources. When GATE= 0, the timers can be started or stopped by software instructions like SETB TR0 or SETB TR1

C/T (clock/Timer):

This bit decides whether the timer is used as delay generator or event counter. When **C/T = 01,** the Timer is used as delay generator and if C/T=1 the timer is used as an event counter. The clock source for the time delay is the crystal frequency of 8051.

M1, M0 (Mode):

These two bits are thetimer mode bits. The timers of the 8051 can be configured in three modes.Mode0, Mode1 and Mode2.The selection and operation of the modes is shown table

Table: Mode selection of Timer

S.No	M0	M1	Mode	Operation
1	0	0	0	13-bit Timer mode 8-bit Timer/counter. THx with TLx as 5-bit pre-scalar

2	0	1	1	16-bit Timer mode.16-bit timer /counter without pre-scalar
3	1	0	2	8-bit auto reload. THx contains a value that is to be loaded into TLx each time it overflows
4	1	1	3	Split timer mode

Serial communication modes of operation in 8051 microcontroller.

SERIAL INTERFACE

The serial port of 8051 is full duplex, i.e., it can transmit and receive simultaneously. The register SBUF is used to hold the data. The special function register SBUF is physically two registers. One is, write-only and is used to hold data to be transmitted out of the 8051 via TXD. The other is, read-only and holds the received data from external sources via RXD. Both mutually exclusive registers have the same address 099H.

Power Mode control Register

Register PCON controls processor power down, sleep modes and serial data bandrate. Only one bit of PCON is used with respect to serial communication. The seventh bit (b7)(SMOD) is used to generate the baud rate of serial communication.

SMOD	–	–	–	GF1	GF0	PD	IDL

- SMOD: Serial baud rate modify bit
- GF1: General purpose user flag bit 1
- GF0: General purpose user flag bit 0
- PD: Power down bit
- IDL: Idle mode bit

Data Transmission: Transmission of serial data begins at any time when data is written to SBUF. Pin P3.1 (Alternate function bit TXD) is used to transmit data to the serial data network. TI is set to 1 when data has been transmitted. This signifies that SBUF is empty so that another byte can be sent

Data Reception: Reception of serial data begins if the receive enable bit is set to 1 for all modes. Pin P3.0 (Alternate function bit RXD) is used to receive data from the serial data network. Receive interrupt flag, RI, is set after the data has been received in all modes. The data gets stored in SBUF register from where it can be read.

SCON : Serial Port Control Register (Bit Addressable)

SM0	SM1	SM2	REN	TN8	RB8	TI	RI

SM0	SCON.7	Serial Port mode specifier (NOTE 1).
SM1	SCON.6	Serial Port mode specifier (NOTE 1).
SM2	SCON.5	Enables the multiprocessor communication feature in mode 2 & 3. In mode 2 or 3, if SM2 is set to 1 then RI will not be activated if the received 9th data bit (RB8) is 0. In mode 1, if SM2 = 1 then RI will not be activated if a valid stop bit was not received. In mode 0, SM2 should be 0 (See table 9).
REN	SCON.4	Set/Cleared by software to Enable/Disable reception.
TB8	SCON.3	The 9th bit that will be transmitted in modes 2 & 3. Set/Cleared by software.
RB8	SCON.2	In modes 2 & 3, is the 9th data bit that was received. In mode 1, if SM2 = 0, RB8 is the stop bit that was received. In mode 0, RB8 is not used.
TI	SCON.1	Transmit interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or at the beginning of the stop bit in the other modes. Must be cleared by software.
RI	SCON.0	Receive interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or half way through the stop bit time in the other modes (except see SM2). Must be cleared by software.

Note 1 :

SM0	SM1	MODE	DESCRIPTION	BAUD RATE
0	0	0	SHIFT REGISTER	Fosc./12
0	1	1	8 bit UART	Variable
1	0	2	8 bit UART	Fosc./64 OR Fosc./32
1	1	3	8 bit UART	Variable

Serial Data Modes

8051 micro controller communicate with another peripheral device through RXD and TXD pin of port3.controller have four mode of serial communication. This four mode of serial communication are below.

a. Serial data mode 0-fixed buad Rate.
b. Serial data mode 1-variable baud rate.
c. Serial data mode 2 -fixed baud Rate.
d. Serial Data mode 3 -variable baud rate.

a) Serial Data Mode-0 (Baud Rate Fixed)

In this mode, the serial port works like a shift register and the data transmission works synchronously with a clock frequency of fosc /12.

Serial data is received and transmitted through RXD. 8 bits are transmitted/ received aty a time.

Pin TXD outputs the shift clock pulses of frequency fosc /12, which is connected to the external circuitry for synchronization.

The shift frequency or baud rate is always 1/12 of the oscillator frequency.

b) Serial Data Mode-1 (standard UART mode)b(baud rate is variable)

In mode-1, the serial port functions as a standard Universal Asynchronous Receiver Transmitter (UART) mode. 10 bits are transmitted through TXD or received through RXD.

The 10 bits consist of one start bit (which is usually '0'), 8 data bits (LSB is sent first/received first), and a stop bit (which is usually '1'). Once received, the stop bit goes into RB8 in the special function register SCON. The baud rate is variable.

C)Serial Data Mode-2 Multiprocessor (baud rate is fixed)

- In this mode 11 bits are transmitted through TXD or received through RXD.
- The various bits are as follows: a start bit (usually '0'), 8 data bits (LSB first), a programmable 9 th (TB8 or RB8)bit and a stop bit (usually '1').
- While transmitting, the 9 th data bit (TB8 in SCON) can be assigned the value '0' or '1'. For example, if the information of parity is to be transmitted, the parity bit (P) in PSW could be moved into TB8.
- On reception of the data, the 9 th bit goes into RB8 in 'SCON', while the stop bit is ignored.
- The baud rate is programmable to either 1/32 or 1/64 of the oscillator frequency.

$$f_{baud} = (2^{SMOD}/64)\, f_{osc}$$

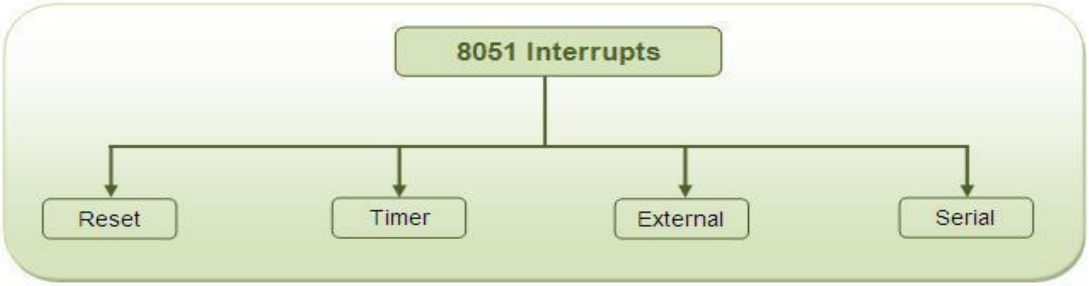
d)Serial Data Mode-3 - Multi processor mode(Variable baud rate)

- In this mode 11 bits are transmitted through TXD or received through RXD.
- The various bits are: a start bit (usually '0'), 8 data bits (LSB first), a programmable 9 th bit and a stop bit (usually '1').
- Mode-3 is same as mode-2, except the fact that the baud rate in mode-3 is variable (i.e., just as in mode-1).

$$f_{baud} = (2^{SMOD}/32) * (f_{osc} / 12\,(256\text{-}TH1))$$

Interrupts used in 8051 microcontroller.

An interrupt is an external or internal event that disturbs the microcontroller to inform it that a device needs its service. The program which is associated with the interrupt is called the **interrupt service routine** (ISR) or **interrupt handler**. Upon receiving the interrupt signal the Microcontroller, finishcurrent instruction and saves the PC on stack.Jumps to a fixed location in memory depending on type of interruptStarts to execute the interrupt service routine until RETI (return from interrupt)upon executing the RETI the microcontroller returns to the place where it was interrupted. Getpop PC from stack



Types of Interrupts in 8051 Microcontroller

The 8051 microcontroller can recognize five different events that cause the main program to interrupt from the normal execution. These five sources of interrupts in 8051are:

- Timer 0 overflow interrupt- TF0
- Timer 1 overflow interrupt- TF1
- External hardware interrupt- INT0
- External hardware interrupt- INT1
- Serial communication interrupt- RI/TI

The Timer and Serial interrupts are internally generated by the microcontroller, whereas the external interrupts are generated by additional interfacing devices or switches that are externally connected to the microcontroller. These external interrupts can be edge triggered or level triggered. When an interrupt occurs, the microcontroller executes the interrupt service routine so that memory location corresponds to the interrupt that enables it.

Each interrupt has a specific place in code memory where program execution (interrupt service routine) begins.

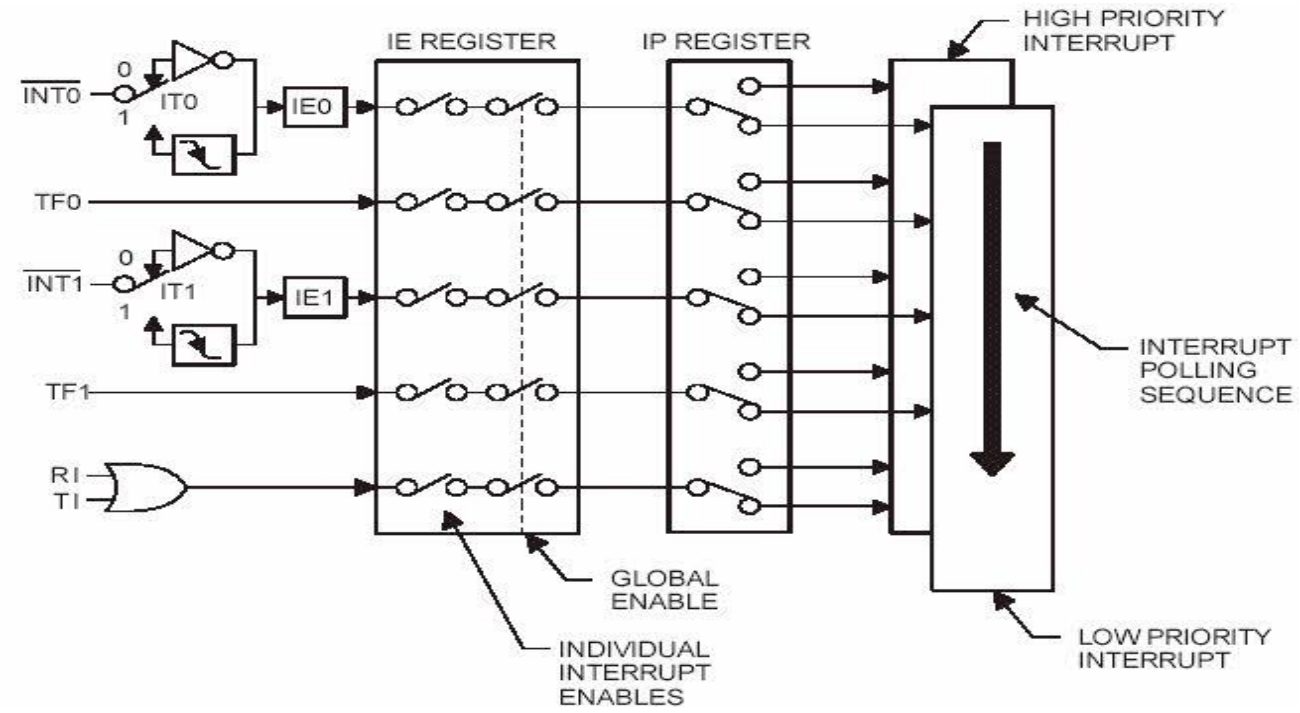
- External Interrupt 0: 0003 H
- Timer 0 overflow: 000B H
- External Interrupt 1: 0013 H
- Timer 1 overflow: 001B H
- Serial Interrupt : 0023 H

Upon reset all Interrupts are disabled & do not respond to the Microcontroller. These interrupts must be enabled by software in order for the Microcontroller to respond to them.This is done by an 8-bit register called Interrupt Enable Register (IE).

Interrupt Structure of 8051 Microcontroller

Upon 'RESET' all the interrupts get disabled, and therefore, all these interrupts must be enabled by software. In all these five interrupts, if anyone or all are activated, this sets the corresponding interrupt flags as shown in the figure. All these interrupts can be set or cleared by bit

in some special function register that is Interrupt Enabled (IE), and this in turn depends on the priority, which is executed by IP interrupt priority register.



Interrupt Enable Register:

EA	—	ET2	ES	ET1	EX1	ET0	EX0

- EA : Global enable/disable. To enable the interrupts this bit must be set High.
- --- : Undefined-reserved for future use.
- ET2: Enable /disable Timer 2 overflow interrupt.
- ES : Enable/disable Serial port interrupts.
- ET1: Enable /disable Timer 1 overflow interrupt.
- EX1:Enable/disable External interrupt1.
- ET0: Enable /disable Timer 0 overflow interrupt.
- EX0 : Enable/disable External interrupt0.

Upon reset the interrupts have the following priority.(Top to down).The interrupt with the highest PRIORITY gets serviced first.

1. External interrupt 0 (INT0)
2. Timer interrupt0 (TF0)
3. External interrupt 1 (INT1)
4. Timer interrupt1 (TF1)
5. Serial communication (RI+TI)

- **Interrupt priority register** -Priority can also be set to “high” or “low” by 8-bit IP register.

—	—	PT2	PS	PT1	PX1	PT0	PX0

IP.7: reserved
IP.6: reserved
IP.5: Timer 2 interrupt priority bit (8052 only)
IP.4: Serial port interrupt priority bit
IP.3: Timer 1 interrupt priority bit
IP.2: External interrupt 1 priority bit
IP.1: Timer 0 interrupt priority bit
IP.0: External interrupt 0 priority bit

Interfacing ADC chip with Microcontroller system.

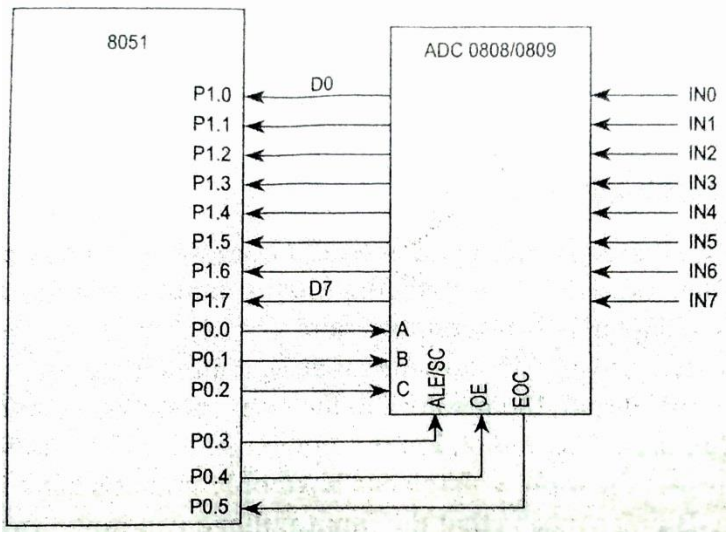
The analog-to-digital converter is an essential component in any microprocessor or microcontroller system, for interfacing analog inputs.

The specifications of an ADC are the range of analog input voltage, the number of digital bits at the output, the resolution, the conversion time, and the number of analog input channels. The analog input voltage can be either unipolar or bipolar. Unipolar means that the input voltage can have only one polarity, e.g.,(0 to +5V) to (0 to +10V). Bipolar means that the input voltage can range from one polarity to the other, e.g.,(-5V to +5V) or (-10V to +10V). Most commercially available ADC chips come with the option of selecting one of these voltage ranges using the Vref input pin. ADC chips are available with different number of output binary bits – 8,10,12, or 16 bits. The number of bits will decide the number of voltage levels sensed. For example, an 8 bit ADC will have 2^8 possible levels, i.e., 256 levels. The number of bits and the input voltage range will decide the resolution. The resolution of an ADC is defined as the smallest change in input voltage that can be sensed or detected at the output. It can be mathematically defined as the range of input voltage range of (5/256). i.e., approximately 19.5mV. The conversion time of ADCs is decided by the type of the ADC and the clock frequency used in the converter circuits.

Some ADC chips come with the option of having more than one analog input. One of these analog input channels is selected using select lines and an analog multiplexer circuit. ADC chips also have a sample-and-hold circuit. The sample-and-hold circuit is used to maintain the analog input voltage constant when the conversion is in progress.

There are two possible methods for interfacing ADC chips with the 8051. In the first method, one can directly interface the ADC chip with the 8051 parallel ports. In the second method, especially in the case of complex systems(involving many external chips including the memory), an 8255 PPI chip may be interfaced with the 8051 and the ADC chip can be interfaced with the 8051 through the 8255. Figure shows the direct interfacing of ADC 0808/0809 with the 8051.

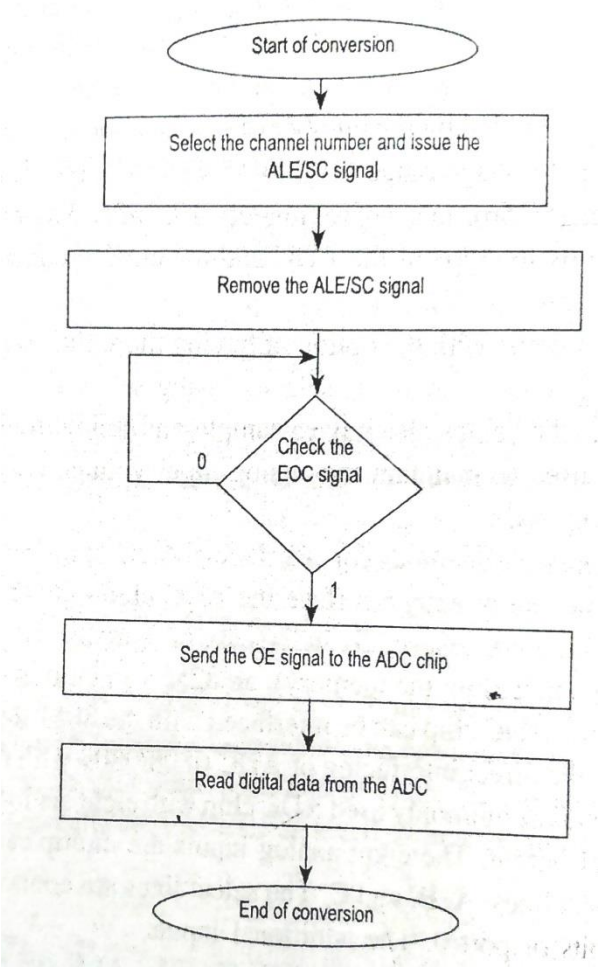
ADC 0808/0809 is a commonly used ADC chip with eight analog input channels and an 8 bit digital output. The eight analog inputs are multiplexed and selected using the three select lines A, B and C. The select lines are connected to the least significant three bits of port 0. The additional inputs +Vref, -Vref, and CLK and the supply inputs must be given to the ADC chip. The ALE and SC pins are tied together and connected to the port pin PO.3. The OE pin is connected to another port 0 pin, PO.4 and the end of conversion signal(EOC) is sensed through PO.5.



The software part follows the flowchart shown in the figure. The channel selection signal is issued to the ADC chip, followed by the ALE/SC signal. Then, the conversion starts within the chip and the EOC signal is received at the end of the conversion. When the EOC signal is received, the program issues the OE signal and then reads the data from the data lines connected to port 1.

The program for analog-to-digital conversion for the interface shown in figure is as follows.

ADCONVERT:	MOV P1,#0FFH	; Make port 1 as the input port by writing 1s
	MOV PO.#CH_NUM	; Select the channel number on port 0 LSB 3 lines
	SETB PO.3	; Issue ALE/SC signal on PO.3 line.
	NOP	; Wait
	NOP	; Wait
	CLR PO.3	; Remove the ALE/SC signal by making it 0
CHECK:	JNB PO.5, CHECK	; Read PO.5 to check EOC until it becomes 1
	SETB PO.4	; Make OE signal high on PO.4
	MOV A,P1	; Reade the result data from port 1
	MOV ADC_RES,A	; Store it in a memory location ADC_res
	RET	; End of subroutine program



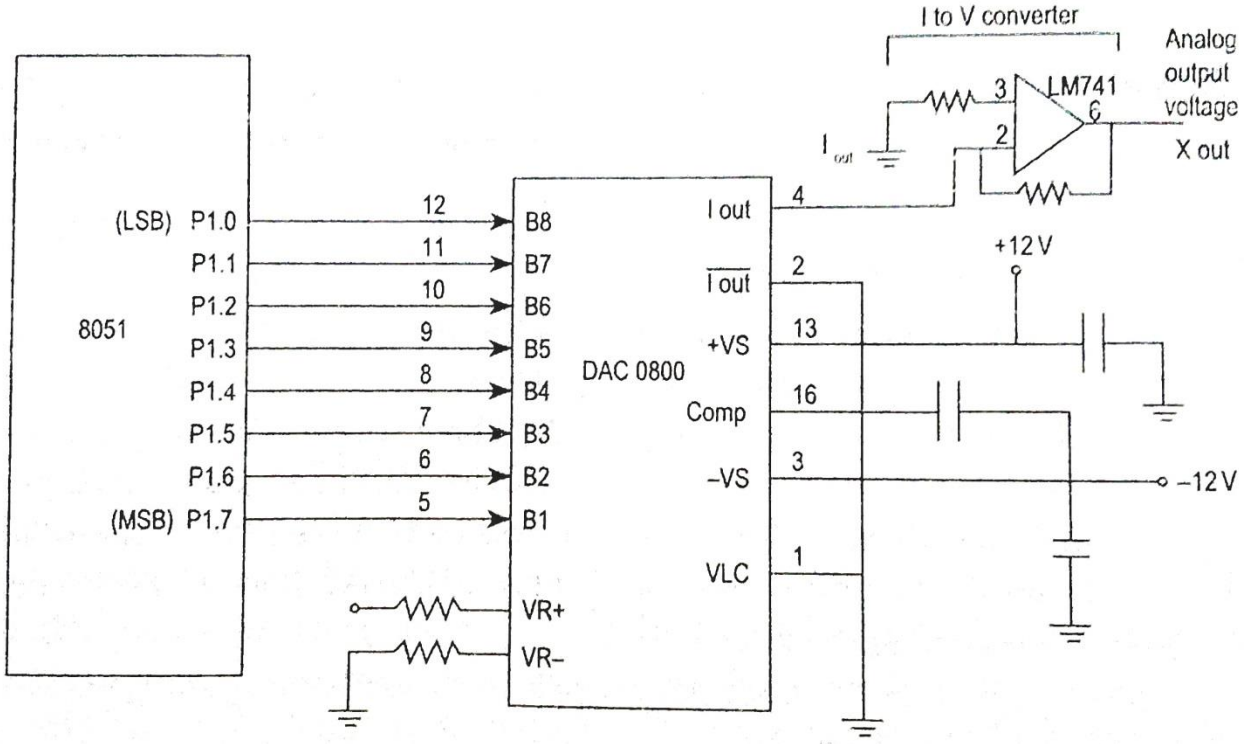
Flowchart for ADC conversion software

INTERFACING DAC CHIP with 8051 Microcontroller system.

Digital-to-analog converters are used to get a proportional analog voltage or current for the digital data sent by the microprocessor.

Let us consider the common digital-to-analog converter chip DAC 0800 in this interface example. This section will describe the interfacing of the DAC 0800 with the 8051 microcontroller. Any one port is enough to interface an 8-bit DAC with the 8051. The interface diagram is shown in the figure. The data lines of the DAC chip are connected to the port 1 lines of the microcontroller. The port 1lines should be made output lines, to send digital output to the DAC. The output from the DAC chip is a variable current and this is converted into a variable voltage using a current-to-voltage converter.

The following section explains the programming for applications involving the DAC. Four common applications – square wave generation, ramp wave generation, staircase wave generation, and sine wave generation - are discussed.



Interfacing DAC 0800 with 8051

Square wave generation:

Write a program to generate a square waveform using the DAC chip.

The DAC chip interfaced with port 1 of the 8051, as shown in figure is considered. In this example, a square waveform is generated using the DAC chip. The square waveform has 0V output for half a period and a voltage of amplitude V_1 for another half.

The data for any particular voltage V_1 is calculated using the following formula: Equivalent value for voltage output output $V_1 = [(2^n - 1)/$ maximum output voltage$] * V_1$, where n is the number of binary bits in the input to the DAC.

For example, the equivalent value for a 3V ouput in a DAC with an 8-bit binary input and a maximum output of 5V, is $[(2^8 – 1)/5]*3=5]*3=153$(in decimal form). The same value (in hexadecimal form) will be 99H.

The following program can generate a square wave of amplitudes 3V and 0V, with a predefined delay and hence a predefined frequency, according to the count value.

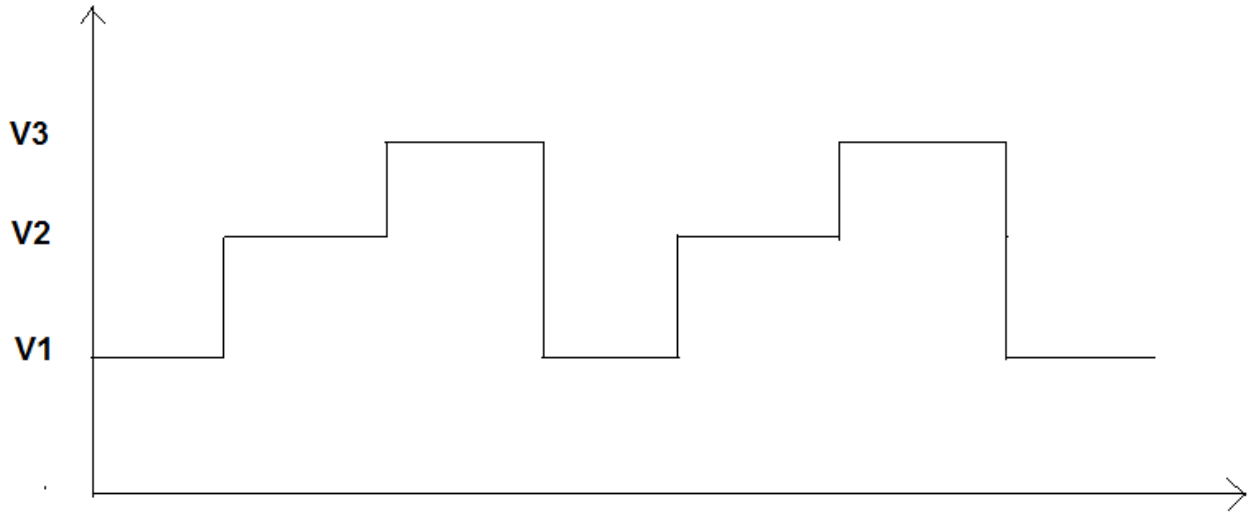
START:	MOV P1, #00H	; Load the hexadecimal equivalent of 0V in port 1
	LOCAL DELAY	; Call the subroutine DELAY
	MOV P1,#99H	; Load the hexadecimal equivalent of 3V in port 1
	LCALL DELAY	; Call the subroutine DELAY
	SJMP START	; Loop again to get a continuous waveform
		; Load a register with a count value
		Decrement it and loop
DELAY:	MOVR0, #COUNT	If the value of count becomes zero, return from
RPT:	NOP	subroutine
	DJNR R0, RPT	
	RET	

Staircase wave generation:

Write a program to generate a staircase waveform using the DAC chip.

The waveform to be generated is given in figure and the hardware is assumed to be the same as in figure. Three levels – V_1 , V_2 and V_3 are assumed in the output voltage waveform. The hexadecimal output to be given to the port is calculated using the formula. The following program assumes these three values as DATA1, DATA2 and DATA3. The fixed time delay is used in all the three levels.

START: MOV P1,#DATA1		; Load the hexadecimal equivalent of V_1 in port 1
CALL DELAY		; Call the subroutine DELAY
MOV P1, #DATA2		; Load the hexadecimal equivalent of V_2 in port 2
CALL DELAY		; Call the subroutine DELAY
MOV P1, #DATA3		; Load the hexadecimal value of V_3 inport 3
CALL DELAY		; Call the subroutine DELAY
SJMP START		; Loop again to get a continuous waveform
DELAY: MOV R0, #COUNT		; Load a register with a count value
RPT: DJNZ R0, RPT		; Decrement it and loop
RET		; If the value of count becomes zero, return from subroutine



Staircase waveform generation

Ramp wave generation:

Write a program to generate a ramp waveform using a DAC chip.

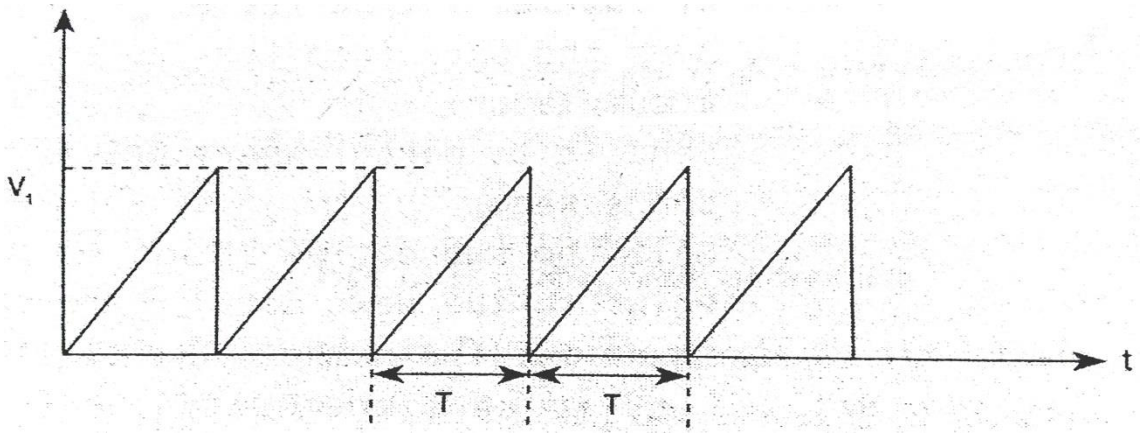
Figure shows the ramp waveform to be generated. Port 1 is given gradually increasing values starting from 0. The hardware details are assumed to be the same as in figure.

The following program generates the ramp waveform shown in figure, with V_1 = 5V. The delay calculation is slightly different from the previous examples. Here, the voltage levels are increased from 00H to FFH, i.e., 0 to 255 in decimal form. So within T seconds, there are 255 levels. Therefore, the delay for each level will be T/255. The delay routine is then written to produce this delay of T/255 seconds. The delay time should be as small as possible for ramp generation. Otherwise, the waveform will look like a staircase waveform with 255 levels.

The following program involves incrementing the value given to port A from 00H to FFH, with a time delay routine called at each level.

START: MOV	; Move the hexadecimal equivalent of 0V to register 1.

R1,#00H	; Move the data in the register R1 to port 1, where the DAC is connected
LOOP: MOV P1, R1	; Call the subroutine DELAY
LCALL DELAY	; Increment register R1
INC R1	; Loop again to send the data to the DAC
SJMP LOOP	; Load a register with a count value
DELAY: MOV R0, #COUNT	; Decrement it and loop
RPT: DJNZ R0, RPT	; If the value of count becomes zero, return from subroutine
RET	



Ramp waveform genearted

Sine wave generation:

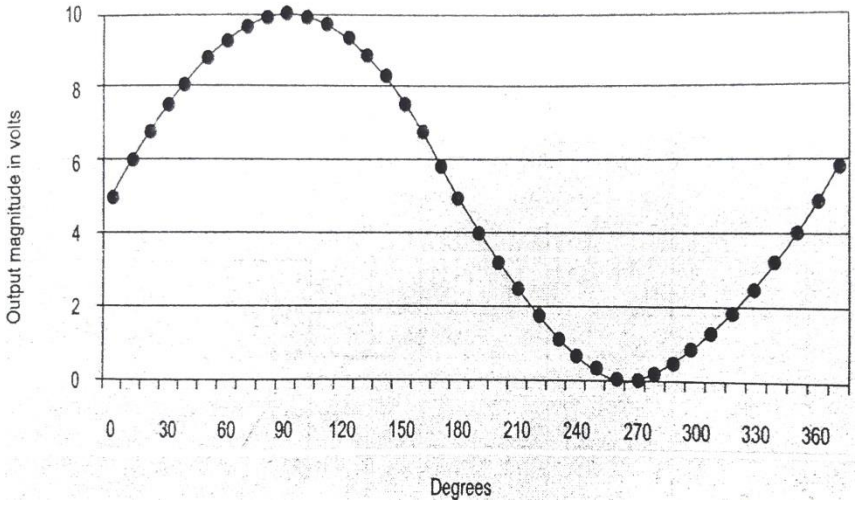
Write a program to generate a sine wave using a DAC chip connected to the 8051 controller.

In this example, the voltage data to be given to port 1 are stored in the memory locations 9000H-9024H. These data can correspond to any predefined waveform, including a sine wave. In such cases, the entire wave is divided into many levels; each level is converted into the corresponding digital value and stored in the memory location. The time delay between the levels is fixed and is generated using the delay routines.

START: MOV DPTR, #9000H	; Initialize the data pointer with the starting address of the data table.
MOV R1, #24H	; Initialize a counter with the number of entries in the table
LOOP: MOV A, @DPTR	; Move the data from the table to the accumulator
MOV P1, A	; Send it t port 1 where DAC is connected
LCALL DELAY	; Call the subroutine DELAY
INC DPTR	; Point to the next data in the table
DJNZ R1, LOOP	; Check for the number of entries and loop to send the next data to port 1
SJMP START	; If all entries in table have been sent, start again from the first entry
DELAY: MOV R0, #COUNT	; Load a register with a count value
RPT: DJNZ R0, RPT	; Decrement it and loop
RET	; If the value of count becomes zero, return from subroutine
9000H: DB 128, 150,	; Table for decimal sine values in steps of 10^0 for 0V to 10V
172, 192, 210, 226,	
239, 248, 254, 256,	
254, 248, 239, 226,	
210, 192, 172, 150,	

128, 106, 84, 64, 46, 30, 17, 8, 2, 0, 2, 8, 17, 30, 45, 64, 84, 105	

For sine wave generation, the sine wave is first converted into the corresponding digital values at the sample intervals. To generate a sine wave, it is assumed that the DAC gives an output voltage of 0V-10V. This voltage rage can be easily adjusted by changing the gain of the I-to-V converter at the output of the DAC network. So the equation (5 + 5 * sin Θ) is used to calculate the voltage required at the output of the DAC network. This voltage waveform is shown in figure. To find the digital values to be given at the input of the DAC network, we should multiply the voltage value by 25.6. This is because for an 8 bit DAC, there are 256 levels with a full-scale output voltage of 10V and so the per step value is256/10. For reducing the number of levels, the sine wave is generated with v,alues for every 10^0 step. The decimal values to be given to the DAC network are given in the program as the look-up table starting at 9000H. To get a more accurate waveform, the table can be constructed with 1^0 steps.



Sine wave generation

In the program given, the delay corresponds to 10^0 steps in the waveform, as the sine values are stored in the look-up table for every 10^0. For example, to generate a 50 Hz sine wave, the delay for 10^0 should be (1/(50×36)), i.e.,0.55 ms. If the count in the delay routine corresponds to 0.55ms, the waveform generated will be at 50Hz.

Interface Matrix key pad with 8051.

Interfacing matrix keypad:

The matrix keypad is organized as a matrix connection of switches. Mechanical switches have a problem called contact bounce because of their construction. The pressing of a mechanical switch must produce a single pulse output. Practically, instead of producing a single clean pulse output, the switches generate a series of pulses because the switch contacts do not come to rest immediately. As the microprocessor is faster than a manual key press, the single key press will be registered as multiple key presses. This is the main consequence of key bouncing. The signal from the key falls and rises a few times within a period of about 5 ms, as the contact bounces. So the

signal from the key must be made free from key bouncing transients. This technique is called key de-bouncing.

Key de-bouncing can be accomplished using hardware or software. The bouncing of key signals occurs within 5 ms. A human cannot press and release a switch in less than 20ms. A de-bouncing logic will check the signal after 20ms and then recognize whether a key is pressed or not. This can be implemented both in hardware and in software. The hardware techniques employ set-reset flip-flops, non-inverting CMOS gates, or integrating de-bouncers. The software technique uses the wait-and-see method. When a signal is sensed from the switch still indicates the key press, the program decides that the user has pressed matrix keyboard interfaced with the 8051.

In the software, a column scan is done by giving an output of 0 on the lines P1.0-P1.3 of port 1. A key press is detected by checking for the row (P2.0-P2.3 of port 2)in which the zero level appears. The scanning of a row is achieved by applying 0V to one port 1 pin and 5V to the other three port 1 pins, then scanning each individual port 2 pin to see if one of them is low. If it is, the key at the junction between the row and column being scanned is the pressed key. As ASCII code (0,1,2,3,……,9.A,B,C,D,E, and F) can be assigned to each key. The scan program uses the subroutine get_code, which in turn uses the look-up table concept to find the ASCII value for the key pressed. It returns the code to register R2. The routine includes a software delay routine of 20ms to solve the keyboard de-bouncing problem.

In the software, a column scan is done by giving an output of 0 on the lines P1.0-P1.3 of port 1. A key press is detected by checking for the row (P2.0-P2.3 of port 2)in which the zero level appears. The scanning of a row is achieved by applying 0V to one port 1 pin and 5V to the other three port 1 pins, then scanning each individual port 2 pin to see if one of them is low. If it is, the key at the junction between the row and column being scanned is the pressed key. As ASCII code (0,1,2,3,……,9.A,B,C,D,E, and F) can be assigned to each key. The scan program uses the subroutine get_code, which in turn uses the look-up table concept to find the ASCII value for the key pressed. It returns the code to register R2. The routine includes a software delay routine of 20ms to solve the keyboard de-bouncing problem.

The procedure for finding a key press involves the following steps:

(i) Clear P1.0, set the other three bits
   a. Scan P2.0
   b. Scan P2.1
   c. Scan P2.2
   d. Scan P2.3

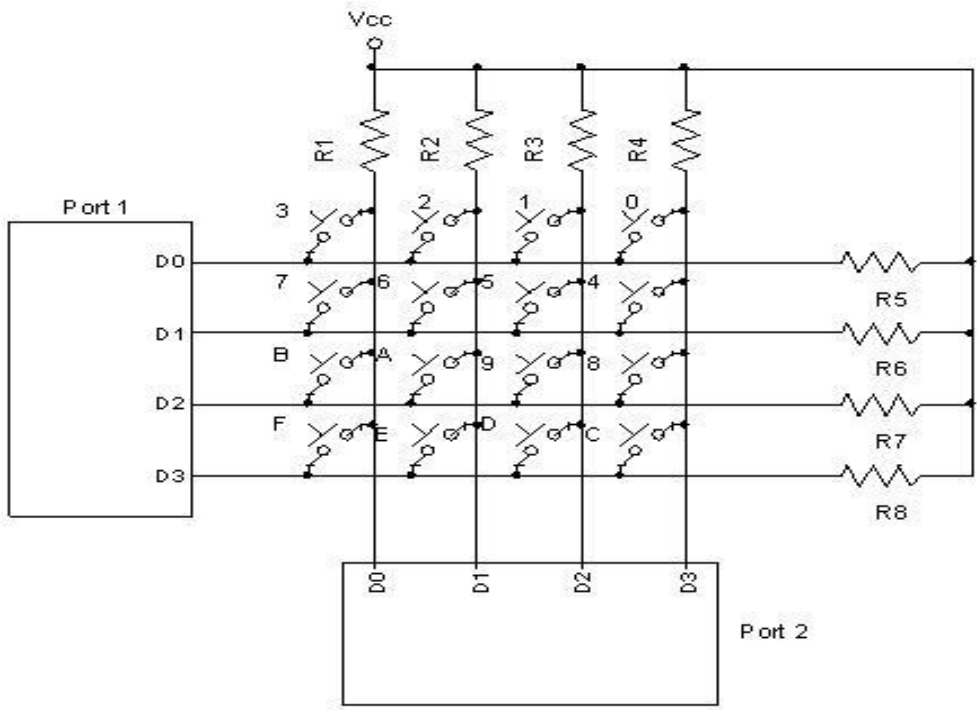
(ii) Clear P1.1, set the other three bits
   a. Scan P2.0
   b. Scan P2.1
   c. Scan P2.2
   d. Scan P2.3

(iii)Clear P1.2, set the other three bits

a. Scan P2.0
b. Scan P2.1
c. Scan P2.2
d. Scan P2.3

(iv) Clear P1.3. set the other three bits

a. Scan P2.0
b. Scan P2.1
c. Scan P2.2
d. Scan P2.3



Matrix keyboard interfaced with 8051

The routine for the scan program is as follows:

SCAN: MOV P1, #11111110B	; Clear P1.0 and set the other three bits
JNB P2.0, key 0	; Check P2.0; If it is 0, go to the subroutine for de-bouncing and key reading
scan1: JNB P2.1, key1	; Check P2.1; If it is 0, go to the subroutine for de-bouncing and key reading
scan2: JNB P2.2, key2	; Check P2.2; If it is 0, go to the subroutine for de-bouncing and key reading
scan3:JNB P2.3, key 3	; Check P2.3; If it is 0, go to the subroutine for de-bouncing and key reading
	; Clear P1.1 and set the other three bits

scan4:MOV P1, #11111101B	; Check P2.0; If it is 0, go to the subroutine for de-bouncing and key reading
JNB P2.0, key 4	; Check P2.1; If it is 0, go to the subroutine for de-bouncing and key reading
scan5: JNB P2.1,key5	; Check P2.2; If it is 0, go to the subroutine for de-bouncing and key reading
scan6: JNB P2.2, key6	; Check P2.3; If it is 0, go to the subroutine for de-bouncing and key reading
scan7: JNB P2.3, key7	; Clear P1.2 and set the other three bits
scan8: MOV P1, #11111011B	
-(continue scanning with other bits)	; Call the subroutine DELAY for key de-bouncing ; If the port 1.0 value is not 0, return to scan1 ; if it is 0, store the key value in reg A
Key0: LCALL DELAY_20ms	; get the corresponding ACSII key code ; call the subroutine DELAY for key de-bouncing
JB P1.0, scan1	; if the port 1.1 value is not zero, return to scan2
MOV A,#0	; if it is 0, store the key value in register A
LJMP get_code	; get the corresponding ASCII key code
Key1: LCALL DELAY_20ms	
JB P1.1, scan2	; use the data pointer to point to the ASCII code table
MOV A, #1	; get the ASCII value using indexed addressing
LJMP get_code	; save the ASCII value in reg R2
…(continue for other keys)	; return from subroutine
get_code: MOV DPTR, #key_table	; table for the ASCII values of keys 0 to F
MOVC A,@A+DPTR	Terminate program execution
MOV R2, A	
RET	
Key_table: BD’0123456789ABCDEF’	
END	

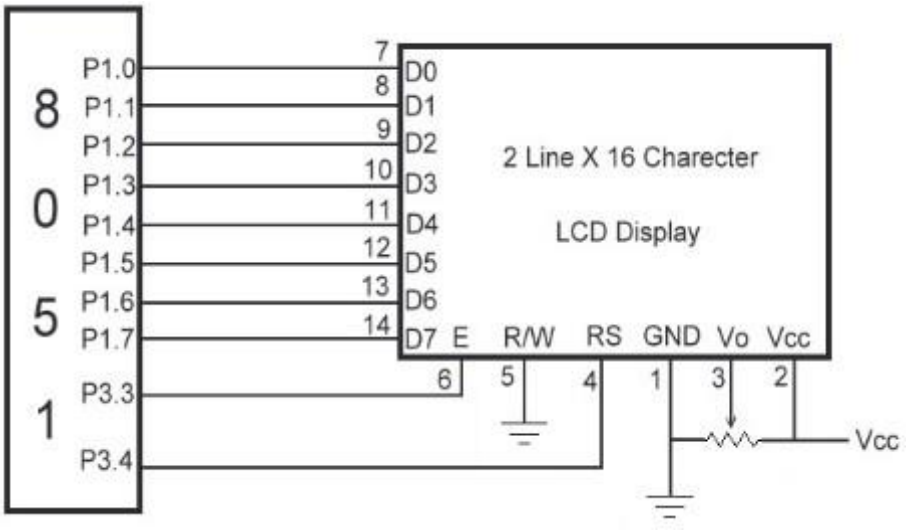
This software can also be modified and written using the ‘SETB bit’ and the ‘CLR bit’ instructions. A set of 16 flag bits in the bit-addressable internal RAM region can be used to store bit information about which key was pressed, and this can be used by the main software routine.

Interfacing LCD with 8051 microcontroller system

Modern LCD devices are becoming popular and are finding an increasing number of applications. The main advantage of LCDs over seven-segment displays is their ability to display numbers, characters, and graphics; the latter can display only numbers and a limited set of characters. LCDs come with an internal controller and refreshing circuit. So the CPU is relieved of the task of refreshing the display. The programming of LCD devices is easier, since they have predefined control words and addresses. The cost of LCDs is also declining.

General LCD devices have 14 pins, which are listed in table.

Pin number	Symbol	Level	I/O	Function
1	V_{ss}	-	-	Power supply(GND)
2	V_{cc}	-	-	Power supply (+5 V)
3	V_{EE}	-	-	Contrast adjust (V_0)
4	RS	0/1 0/1	I	Command or data register select line 0=instruction input 1=data input
5	R/W		I	0=write to LCD module 1=read from LCD module
6	E	1 to 0	I	Enable signal
7	DB0	0/1	I/O	Data bus line 0 (LSB)
8	DB1	0/1	I/O	Data bus line 1
9	DB2	0/1	I/O	Data bus line 2
10	DB3	0/1	I/O	Data bus line 3
11	DB4	0/1	I/O	Data bus line 4
12	DB5	0/1	I/O	Data bus line 5
13	DB6	0/1	I/O	Data bus line 6
14	DB7	0/1	I/O	Data bus line 7(MSB)



LCD interface with 8051

Eight data lines are used for interfacing the LCD with the processor. The three control signals are RS, R/$\overline{W}$, and E.

1. RS is used to select a control register or a data register
2. R/$\overline{W}$ indicates the direction of data flow between the processor and the display
3. The E signal is used to enable data transfer
   The three control signals and eight data lines are interfaced with the two ports of the 8051. In the interface diagram, the port 1 lines are used to transfer data and the port 2 lines are used to issue the control signals from the microcontroller to the LCD.

The LCD interface accepts a number of commands/instructions. A select list of these commands is given in table.

Hex code	Instruction
01	Clear display and return cursor to home position
02	Return cursor to home position
04	Shift cursor left(decrement)
05	Shift cursor right(increment)
06	Shift display right
07	Shift display left
08	Display off and cursor off
0A	Display off and cursor on
0C	Display on and cursor off
0E	Display on and cursor character not blinking
0F	Display on and cursor character blinking

An exhausting list of LCD command words is given in table.

Instruction	Code										Description
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
Clear display	0	0	0	0	0	0	0	0	0	1	Clears display and returns cursor to the home position(address 0)
Cursor home	0	0	0	0	0	0	0	0	1	*	Returns cursor to home postion (address 0). Returns the rotating display to the original position. DDRAM contents remain unchanged

The programming of the LCD is done with the appropriate initialization word and command words. Every time a control or command word is written into the LCD, the RS signal input must be made 0; R/W must be made 0 for the write operation. After setting these two signals, the enable signal E must be applied as a pulse for a duration of not less than 450ns. Similarly, when data is written into the LCD device, the RS signal must be made 1 and R/$\bar{W}$ must be made 0. To display characters and numbers the corresponding ASCII code is sent to the data registers of the LCD display.

The following program is used to display the 15-character data available in the series of external memory locations starting at 9000H in the LCD.

This program calls a delay subroutine to allow the LCD to take the command or data into its registers. The delay is written with two registers. Instead of the delay routine, the busy flag of the LCD can be checked to see whether the data or command has been written into the LCD registers. The busy flag is available in the D7 data line of the LCD. The following subroutine, READY, can be used instead of the DELAY subroutine. This subroutine checks the busy flag from the LCD and returns from the subroutine when the busy signal is removed by the LCD.

OTHER APPLICATION FOR 8051 MICROCONTROLLER

2 marks

1. What is the significance of watch dog timer?(April 2016)

A watchdog timer (sometimes called a computer operating properly or COP timer, or simply a watchdog) is an electronic timer that is used to detect and recover from computer malfunctions. During normal operation, the computer regularly resets the watchdog timer to prevent it from elapsing, or "timing out".

2. What are the different timer mode operations of 8051?(may 2016)

This is an 8-bit register which is used by both timers 0 and 1 to set the various timer modes. In this TMOD register, lower 4 bits are set aside for timer0 and the upper 4 bits are set aside for timer1. In each case, the lower 2 bits are used to set the timer mode and upper 2 bits to specify the operation

Fig. TMOD Register

These two bits are thetimer mode bits. The timers of the 8051 can be configured in three modes.The selection and operation of the modes is shown in below table

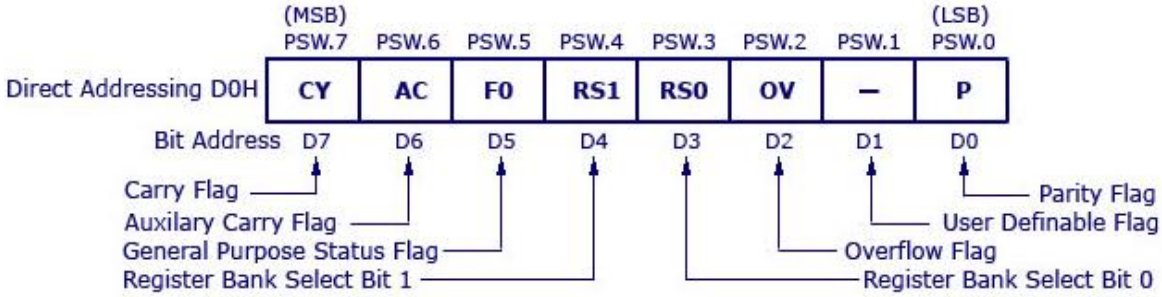
Table: Mode selection of Timer

S.No	M0	M1	Mode	Operation
1	0	0	0	13-bit Timer mode 8-bit Timer/counter. THx with TLx as 5-bit pre-scalar
2	0	1	1	16-bit Timer mode.16-bit timer /counter without pre-scalar
3	1	0	2	8-bit auto reload. THx contains a value that is to be loaded into TLx each time it overflows

4	1	1	3	Split timer mode

3. What is PSW register?

It is also referred to as the flag register. The Program Status Word (PSW) contains status bits that reflect the current CPU state. The 8051 variants provide one special function register called PSW with this status information.



Symbol	Function
CY	Carry Flag
AC	Auxiliary Carry Flag (For BCD Operations)
F0	Flag 0 (Available to the user for General Purpose)
RS1 RS0	Register bank select bit 1 Register bank select bit 0 RS1 RS0 Working Register Bank Address 0 0 Bank 0 (00:00H-00:07H) 0 1 Bank 1 (00:08H-00:0FH) 1 0 Bank 2 (00:l0H-00:17H) 1 1 Bank 3 (00:18H-00:1FH)
OV	Overflow flag
UD	User definable flag
P	Parity Flag

4. Show the contents of the PSW register after the execution of the following 8051 instructions:

MOV A,0BFH

ADD A,#1BH **(May 2016)**

Solution :

BF --→1011 1111

+1B --→0001 1011

DA --→ 1101 1010

The bits of the PSW are now as follow

PSW.7- CY= 0 since there is no carry beyond the D7 bit

PSW.6-AC=1 since there is a carry from D3 to the D4 bit

PSW.5-F0: unused, hence 0.
PSW.4-Register bank selector bit RS1=0 since by default, Bank 0 is selected
PSW.3- Register bank selector bit RS0=0 since by default, Bank 0 is selected
PSW.2-OV= 0 since there is no carry from D6 to D7(the signed bit in signed operations)
PSW.1-Not used, hence 0
PSW.0-P=1 sice there is an odd number of '1's in the accumulator.
The content of the PSW is thus 01000001.ie. 41h.

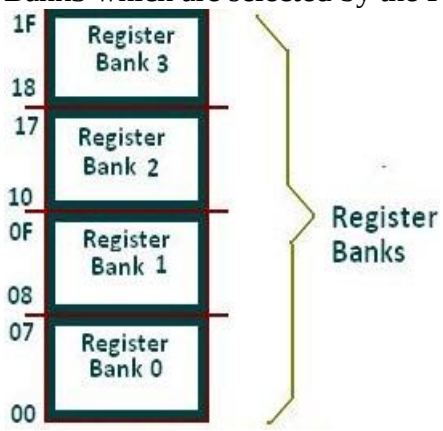
5. List the interrupts of 8051 micro controller? (April 2015)

The 8051 microcontroller can recognize five different events that cause the main program to interrupt from the normal execution. These five sources of interrupts in 8051are:

- Timer 0 overflows interrupt- TF0
- Timer 1 overflow interrupt- TF1
- External hardware interrupt- INT0
- External hardware interrupt- INT1
- Serial communication interrupt- RI/TI

6. What are regular banks in 8051 micro controller? (April 2015)

The 8051 microcontroller consists of four register banks, such as Bank0, Bank1, Bank2, Bank3 which are selected by the PSW (Program Status Word) register.



These register banks are present in the internal RAM memory of the 8051 microcontroller, and are used to process the data when the microcontroller is programmed.

7. What is micro controller? (Nov 2015,Dec 2014)

A microcontroller is a computer present in a single integrated circuit which is dedicated to perform one task and execute one specific application.
It contains memory, programmable input/output peripherals as well a processor. Microcontrollers are mostly designed for embedded applications and are heavily used in automatically controlled electronic devices such as cellphones, cameras, microwave ovens, washing machines, etc

8. What is the difference between timer and counter in 8051 micro controller? (Nov 2015)

- The 8051 has two counters/timers which can be used either as timer to generate a time delay or as counter to count events happening outside the microcontroller.

- The 8051 has two timers: timer0 and timer1. They can be used either as timers or as counters. Both timers are 16 bits wide. Since the 8051 has an 8-bit architecture, each 16-bit is accessed as two separate registers of low byte and high byte.

9. What are the hardware components required in designing control circuit to control the temperature? (May 2014)

- Temperature Sensor LM35
- Op-Amp IC (LM324/741)
- Potentiometer-10k
- 7805 Voltage Regulator
- 100uF Capacitor

10. Give the bit pattern of 8051 flags. (April / May 2014)

The flag register of 8051 is called PSW (Program Status Word). The PSW consists of four math flags and two register bank select bits. The math flags are Carry, Auxiliary carry, Overflow and Parity flag. The register bank select bits are RS0 and RS1.

11. List the features of 8051 microcontroller. (November 2013)

- 8 bit controller operating on bit and byte operand
- Provide separate code and data memory address base
- 856 bytes internal RAM and 4kb internal ROM
- 64kb external data memory address space
- One number of programmable serial ports
- Two numbers of Programmable timers
- 5 numbers of vectored interrupts

12. Write any five functional blocks of 8051.(November 2012)

- 128 bytes of on-chip RAM
- Four ports of eight bits each
- Timing and Control block
- 8 bit CPU
- Full-duplex serial port

13. Compare Microprocessorand Microcontroller. (November 2012) or Name any four additional features in micro controller when compared to microprocessors? (Dec- 2014)

S.No	**Microprocessor**	**Microcontroller.**
1.	Microprocessor contains ALU, General Purpose register, Stack Pointer, Program Counter, timing and control circuit.	Microcontroller contains the circuitry of microprocessor and in addition it has built-in ROM, RAM,I/O devices, timers and counters.
2.	It has many instructions to move data between memory and CPU.	It has one or two instructions to move data between memory and CPU.
3.	It has one or two bit handling instructions.	It has many bit handling instructions
4.	Access times for memory and I/O devices are more.	Less access times for built-in memory and I/O devices.
5.	Microprocessor based system requires	Microcontroller based system requires less

	more hardware.	hardware reducing PCB size and increasing the reliability.
6.	Microprocessor based system is more flexible in design point of view.	Less flexible in design point of view.
7.	It has single memory map for data and code.	It has separate memory map for data and code.
8.	Less number of pins are multi-functioned.	More number of pins are multi-functioned.

14. List the various machine cycles of 8051 microcontroller.

- External program memory fetch cycle
- External data memory read cycle
- External data memory write cycle
- Port operation cycle

15.What are the addressing modes available in 8051 controller?

- Immediate addressing
- Register indirect addressing
- Direct addressing
- Implied addressing
- Register addressing
- Relative addressing

16. How to estimate the time taken to execute an instruction in 8051 microcontroller.

The time taken to execute an instruction by 8051 controller is obtained by multiplying the time to execute a machine cycle by the number of machine cycles of the instruction. The time to execute a machine cycle is 12 clock periods

Therefore, Time taken to execute an instruction = C x 12 x T = C x 12 x (1 / f)

Where, C=No. of machine cycles of an instruction
T = Time period of crystal frequency in seconds
f = Crystal frequency in Hz

Pondicherry University Questions:

2 marks

1. What is the significance of watch dog timer?(April 2016)
2. What are the different timer mode operations of 8051?(may 2016)
3. Show the contents of the PSW register after the execution of the following 8051 instructions:
MOV A,0BFH
ADD A,#1BH(May 2016)
4. List the interrupts of 8051 micro controller?(April 2015)
5. What are regular banks in 8051 micro controller?(April 2015)
6. What is micro controller?(Nov 2015,Dec 2014)
7. What is the difference between timer and counter in 8051 micro controller?(Nov 2015)
8. Name any four additional features in micro controller when compared to microprocessors?(Dec 2014)
9. What is the use of a DMA controller?(Nov 2014)
10. What are the hardware components required in designing control circuit to control the temperature? (May 2014)

11. Give the bit pattern of 8051 flags. (April / May 2014)
12. List the features of 8051 microcontroller. (November 2013)
13. Write any five functional blocks of 8051. (November 2012)
14. Compare microcontroller and microprocessor. (November 2012)

11 marks

1. Explain the relevant hardware diagram, how an ADC is interfaced with 8051 microcontroller?(11m-may16)
2. Explain serial data i/o of 8051 micro controller?(6m-april 2016)
3. Describe the external memory and its interfacing with 8051?(5m-april 2016)
4. Explain the architecture of 8051 microcontroller. (April 2015)
5. Explain the different serial communication modes of operation in 8085 micro controller?(5m-april 2015)
6. Write an assembly language program to generate the square and triangle waveform using 8051. (April 2015)
7. Explain the different modes of operation of micro controller 8051? (6m-April 2015)
8. With suitable diagram explain how a microcontroller can be used for temperature monitoring and control?(11m-nov 2015, 11m-dec 2014)
9. Explain the various functional blocks of 8051?(11m-dec 2014)
10. Draw the simplified block diagram of 8051 micro controller and discuss the various i/o ports Available?(11 m-Mays 2016, 11m-may 2014)

References:

M. Senthilkumar, M. Saravanan and S. Jeevanandhan,"Microprocessors and Microcontrollers", Oxford University Press, 2010.