MICROPROCESSOR AND INTERFACES

IV C.S.

1 INTRODUCTION TO MICROPROCESSORS

CHAPTER IN A NUTSHELL

❑ Microprocessor

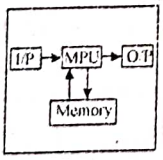
A microprocessor is a multipurpose, programmable logic device that reads binary instructions from a storage device called memory, accepts binary data as input and processes data according to those instructions and provides results as output.

The functions of various components of a microprocessor based system can be summarized as follows:

1. **Memory**
   - Stores binary instructions, called programs and data.
   - Provides the instructions and data to the microprocessor on request.
   - Stores results and data for the microprocessor.
2. **Input Device**
   Enters data and instructions under the control of program such as a monitor program.
3. **Output Device**
   Accepts data from the microprocessor as specified in a program.
4. **Bus**
   Carries bits between the microprocessor and memory and I/Os.

❑ Microcontroller

In microprocessor, peripheral devices (I/O devices + memory) are fabricated on a single chip then it is called microcontroller which is shown in figure.



❑ Different types of Computers

1. Microcomputers : As the name implies, microcomputers are small computers. They are low cost, small digital computers. Portable computers, personal computers (PCs) (single user desktop computers), computers for dedicated applications such as–industrial control, instrumentation appliances control etc. come under the category of microcomputers. A computer with its microprocessor as its CPU is called microcomputers. Single chip microcontrollers are known as microcomputers. Present day microcomputers can be classified in four groups: Personal (or business) computer (PC), work stations, single board and single chip computers.

2. Mini Computers : Mini computers are more powerful than micro computers. They are multi- user system. A mini computer runs more slowly. It works directly with smaller data words (often 32 bit words).They are used for single scientific calculation and industrial control.

3. Mainframe Computers : These are high speed computers. The data storage capacity of these computers are very large. The word length varies normally from 32 bit to 64 bits. They are multi-user system. They are used for scientific calculation.

4. Super Computers : These are more powerful, much faster and costlier than mainframe computers. They are used for complex analysis and computation.

Features of 8085 μP

(1) It is an 8 bit microprocessor *i.e.* it can accept, process or provide 8 bit data simultaneously.
(2) It is a single chip NMOS device implemented with 6500 transistors.
(3) It requires a single + 5V power supply
(4) The maximum clock frequency is 3.07 MHz and minimum clock frequency is 500KHz.
(5) It requires two phases, 50% duty cycle, TTL clock. These clock signals are generated by an internal clock generator.
(6) It provides 74 instructions with following addressing modes register, direct, immediate, indirect and implied addressing modes.
(7) The data bus is multiplexed with lower address bus, hence it requires external hardware to separate data lines from address lines.

(8) It provides 16 address lines, hence it can access $2^{16} = 64$ K bytes of memory.

(9) It generates 8 bit I/O address, hence it can access $2^8 = 256$ input ports and 256 output ports.

(10) It provides 5 hardware interrupts : TRAP, RST5.5, RST6.5, RST7.5, INTR.

(11) By providing external hardware (8259 chip) one can increase the interrupt capability of it.

(12) It provides status and control signals.

PREVIOUS YEARS QUESTIONS

PART-A

Prob.1 ***What do you mean by input port and output port?*** [R.T.U. 2019]

Sol. In computer hardware, a port serves as an interface between the computer and other computers or peripheral devices. In computer terms, a port generally refers to the part of a computing device available for connection to peripherals such as input and output devices.

Input/Output Port

1. An I/O port is a socket on a computer that a cable is plugged into. The port connects the CPU to a peripheral device via a hardware interface or to the network via a network interface.
2. In a PC, an I/O port is an address used to transfer data.

Prob.2 ***What is the use of ALE signal?*** [R.T.U. 2019]

Sol. ALE : This is a positive going pulse generated every time the 8085 begins an operation; when it is high, it indicates that the bits on $AD_7 - AD_0$ are address bits and when it is low it indicates that bits on $AD_7 - AD_0$ are data bits.

Prob.3 ***Why AD_0-AD_7 lines are multiplexed?*** [R.T.U. 2019]

Sol. Microprocessor is responsible for both data transfer and memory access. 8085 microprocessor has 8 data lines and 16 address lines – lower order eight address lines ($A_7 - A_0$) and eight data lines ($D_7 - D_0$) are multiplexed with each other to form $AD_7 - AD_0$. If we do not multiplex these lines then we have to add more pins on microprocessor IC. To reduce complications on microprocessor IC, we need multiplexing.

Prob.4 ***Define general purpose registers and their use.*** [R.T.U. 2019]

Sol. General Purpose Registers : 8085 has six 8–bit general–purpose registers to store 8–bit data named as B,C,D,E,H and L. They can be combined to form register pairs – BC, DE and HL to store 16–bits of data. These registers are programmable by user to store/copy any data and used to perform different operations and hence also known as **scratched registers**. The HL pair is used to address members.

Prob.5 ***What is microprocessor?*** [R.T.U. 2019]

OR

Explain microprocessor. [R.T.U. 2017]

Sol. Microprocessor : It is a multipurpose, programmable, clock driven, register based electronic device that reads binary instruction from memory, accepts binary data as input and processes them.

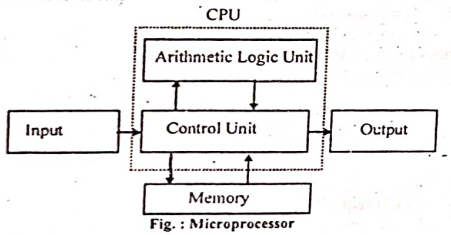


Fig. : Microprocessor

Prob.6 ***Write short note on Multiplexer.*** [R.T.U. 2017]

Sol. Multiplexer : Multiplexer means many into one. A multiplexer is a circuit used to select and route any one of the several input signals to a signal output. A simple example of a non electronic circuit of a multiplexer is a single pole multiposition switch.

Multiposition switches are widely used in many electronics circuits. However circuits that operate at high speed require the multiplexer to be automatically selected. A mechanical switch cannot perform this task satisfactorily. Therefore, multiplexer used to perform high speed switching are constructed of electronic components.

Multiplexer handle two type of data that is analog and digital. For analog application, multiplexer are built of relays and transistor switches. For digital application, they are built from standard logic gates.

The multiplexer used for digital applications, also called digital multiplexer, is a circuit with many input but only one output. By applying control signals, we can steer any input to the output. Few types of multiplexer are 2-to-1, 4-to-1, 8-to-1 and 16-to-1 multiplexer.

Following figure shows the general idea of a multiplexer with n input signal, m control signals and one output signal.

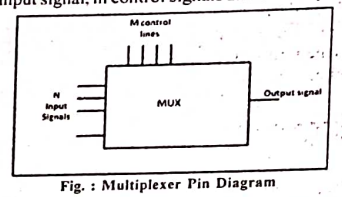


Fig. : Multiplexer Pin Diagram

Prob.7 ***Write short note on De-multiplexer.*** [R.T.U. 2017]

Sol. De-multiplexer : De-multiplexer means one to many. A de-multiplexer is a circuit with one input and many outputs. By applying control signal, we can steer any input to the output. Few types of de-multiplexer are 1-to-2, 1-to-4, 1-to-8 and 1-to-16 De-multiplexer.

Following figure illustrate the general idea of a de-multiplexer with 1 input signal, m control signals, and n output signals.

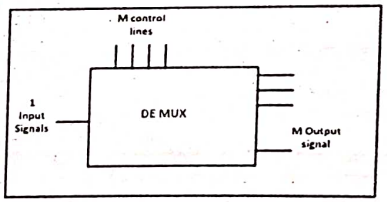


Fig. : De-multiplexer Pin Diagram

Prob.8 ***Write short note on Encoder.*** [R.T.U. 2017]

Sol. Encoder : "An encoder is a combinational logic circuit that essentially performs a 'reverse' decoder function. It accepts an active level on one of its inputs representing a digit, such as a decimal or octal digit, and converts it to a coded output such as binary or BCD."

Encoders can also be devised to encode various symbols and alphabetic characters. This process of converting from familiar symbols or numbers to a coded format is called Encoding. Encoder has n input lines, only one of which is active at any time and m output lines. In an encoder the number of output is less than the number of input.

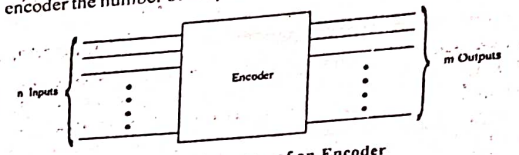


Fig. : Block Diagram of an Encoder

An 8 to 3 encoder has eight active low inputs and three output lines. When the input line 0 goes low, the output is 000, when the input line 5 goes low, the output is 101. For simultaneous inputs, priority encoders are very successful. 74LS148 is an example of 8 to 3 priority encoder.

Encoders are commonly used with keyboards. For each key pressed, the corresponding binary code is placed on the data bus.

Prob.9 ***Write short note on Decoder.*** [R.T.U. 2017]

Sol. Decoder : "Decoder is used to detect the presence of a specified combination of bits (code) on its inputs and to indicate that presence by a specified output level."

When generalized, a decoder has n input lines to handle n bits and from one to 2 output line to indicate the presence of one or more n-bit combinations. Decoding is necessary in applications such as data multiplexing, digital display, digital to analog converters and memory addressing.

For example, if the input to a decoder has 2 binary lines, the decoder will have 4 output lines. The 2 lines can assume four combinations of input signals 00, 01, 10, 11 with each combination identified by the output lines 0 to 3. If the input is $(10)_2$, the output line 2 will be at logic 1, and others will remain at logic 0. This is called Decoding.

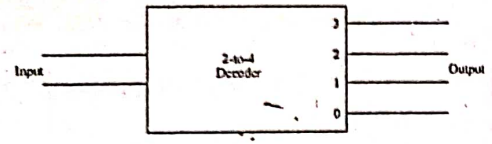


Fig. 1 : 2 to 4 Decoder Logic Symbol

Example of the decoder can be 3 to 8, 4 to 16, 4 to 10 decoders. Decoders have active low enable lines, which decide when to turn it on.

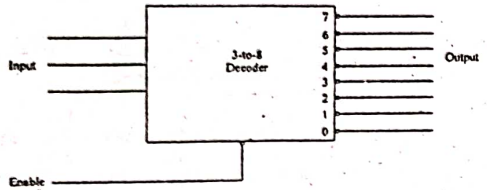


Fig. 2 : 3 to 8 Decoder Logic Symbol with Enable

A decoder is commonly used device in interfacing I/O peripherals and memory. Decoders are also built internal to a memory chip to identity individual memory register.

Basic Binary Decoder : Suppose we wish to determine when a binary 1001 occurs on the inputs of a digital circuit. An AND gate can be used as the basic decoding element because it produces a HIGH output only when all of its inputs are HIGH. Therefore, we must make sure that all of the inputs to the AND gate are HIGH when the binary number 1001 occurs, this can be done by inverting the 2 middle bits (the 0s).

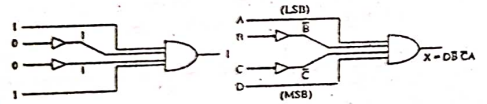


Fig. 3 : Decoding logic for 1001 with an Active-HIGH Output

A is the LSB and D is the MSB. In the representation of a binary number or other weighted code, LSB is always the right most bit in a horizontal arrangement and the top most bit in a vertical arrangement, unless specified otherwise.

Prob.10 *Explain microcontroller.* [R.T.U. 2017]

Sol. Microcontroller : It is a device that includes microprocessor memory and I/O signal lines on a single chip, fabricated using VLSI technology.

(a) Reprogrammable system.

(b) Embedded system.

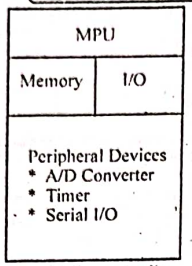


Fig. : Microcontroller

Prob.11 *Explain address bus.* [R.T.U. 2017]

Sol. Address Bus : The address bus is a group of 16 lines generally identified as A_0 to A_{15}. The address bus is unidirectional : bits flow in one direction - from the MPU to peripheral devices. The MPU uses the address bus to perform the first function : identifying a peripheral or a memory location.

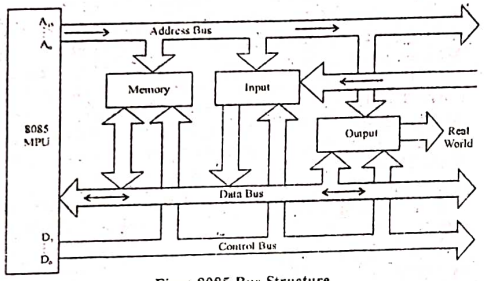


Fig. : 8085 Bus Structure

In a computer system, each peripheral or memory location is identified by a binary number, called an address and the address bus is used to carry a 16-bit address. This is similar to the postal address of a house. A house can be identified by various number schemes. For example, the forty-fifth house in a lane can be identified by the two digit number 45 or by the four digit number 0045. The two digit number scheme can identify only a hundred houses, from 00 to 99. Similarly, the number of address lines of the MPU determines its capacity to identify different memory locations (or peripherals). The 8085 MPU with its 16 address lines is capable of addressing $2^{16} = 65,536$ (generally known as 64K) memory locations. As 1K memory is determined by rounding off 1024 to the nearest thousand, similarly, 65,536 is rounded off to 64,000 as a multiple of 1K.

The number of address lines is arbitary, it is determined by the designer of a microprocessor based on such considerations as availability of pins and intended application of the processor. For example, the Intel 8088 processor has 20 and pentium processor has 32 address lines.

Prob.12 *Explain data bus.* [R.T.U. 2017]

Sol. Data Bus : The data bus is a group of eight lines used for data flow. These lines are bi-directional, data flows in both directions between the MPU and memory and peripheral devices. The MPU uses the data bus to perform the second function: **transferring binary information.**

The eight data lines enable the MPU to manipulate 8-bit data ranging from 99 to FF ($2^8 = 256$ number). The largest number that can appear on the data bus is 11111111 $(255)_{10}$ the 8085 is known as an 8-bit microprocessor. Microprocessors such as the Intel 8086, Zilog Z8000 and Motorola 68000 have 16 data lines; thus they are known as 16-bit microprocessors. The Intel 80386/486 have 32 data lines; thus they are classified as 32-bit microprocessors.

PART-B

Prob.13 *Explain role of flag register in Assembly Language Programming also describe various flags available in 8085.* [R.T.U. 2019, 09]

Sol. Flag Register : Flag register is a 8-bit special purpose register. The role of the flag register in programming is that all the conditional branching operations are based on the four flags of the flag register as an example INC "addr" is based on the carry flag. If the carry flag value is CY = 0 then only the program control will transfer to the given address otherwise the next instruction will be executed. The flag register and five bits set and reset condition is shown below :

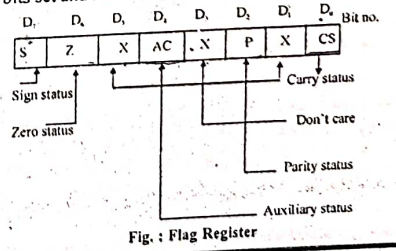


Fig. : Flag Register

1. **Carry Flag (CY):** If an arithmetic operation results in carry, a carry flag is set (1). Otherwise it is reset (0). The carry flag also serves as a borrow flag for the substraction. The carry flag solves the dual purpose of showing carry borrow in case of addition and substraction respectively. It is represented by D_0 bit of flag register.

2. **Parity Flag (P):** This flag bit (D_2) shows the result of the arithmetic operation is having even number of ones (or) add. Since 8085 microprocessor follows even parity system so if the number of ones in accumulation after arithmetic operation is even. This flag is set (1) other wise it is cleared (0).

3. **Auxiliary Carry Flag (AC):** In an arithmetic operation when a carry is generated by lower nibble and passes to upper nibble, the auxiliary carry flag is set. This flag is used internally for BCD (Binary Coded Decimal) operations D_4 bit of flag represents the AC flag.

4. **Zero Flag (Z):** The zero flag shows that the result of any arithmetic or logical operation is zero or non zero. The zero flag is set (1) when the result is zero otherwise it is reset (0). Zero flag is represented by D_6 bit of flag register.

5. **Sign Flag (S):** The sign flag is replica of the D_7 bit of the result. The sign flag is set (1) if bit 7 of the reset is (1), otherwise it is reset (0). Sign flag is represented by D_7 bit of flag register.

Prob.14 *Explain the role of following in 8085-*

(a) Program Counter

(b) Stack Pointer [R.T.U. 2019]

Sol.(a) Program Counter : Program is a sequence of instructions, microprocessor fetches these instructions from the memory and executes them sequentially. The program counter is a special purpose register which, at a given time, stores the address of the next instruction to be fetched. Program Counter acts as a pointer to the next instruction. How processor increments program counter depends on the nature of the instruction; for one byte instruction it increments program counter by one, for two byte instruction it increments program counter by two and for three byte instruction it increments program counter by three such that program counter always points to the address of the next instruction.

In case of JUMP and CALL instruction, address followed by JUMP and CALL instructions is placed in the program counter. The processor then fetches the next instruction from the new address specified by JUMP or CALL instruction. In conditional JUMP and conditional CALL instructions, if the condition is not satisfied, the processor increments program counter by three so that it points the

instruction followed by conditional JUMP or CALL instruction; otherwise processor fetches the next instruction from the new address specified by JUMP or CALL instruction.

(b) Stack Pointer : It is a register that points to the top of the stack. Stack is a memory area meant for specific purpose. This area should not be used for any other purpose. The stack works on Last in First Out basis. The stack pointer will be pointing to the bottom of the stack. The stack pointer will be decremented when data or address is stored on the top of the stack. The stack pointer will be incremented whenever the data or address is retrieved from the stack memory. The system makes use of the stack whenever there is a CALL instruction executed RET instruction executed, service interrupt and returns from the interrupt service routine.

Prob.15 *Explain the address BUS, data BUS and control BUS of 8085 microprocessor. Also draw the flag register.* [R.T.U. 2015]

OR

Define and explain the Bus system

[Raj. Univ. 2003, 2001, MREC 1996]

OR

Explain the function of various types of buses in 8085 microprocessor in brief. [R.T.U. 2013]

Sol. Microprocessor-Initiated Operations and 8085 Bus Organization :

The MPU performs primarily four operations :

1. **Memory read :** Reads data (or instructions) for memory.
2. **Memory Write :** Writes data (or instructions) into memory.
3. **I/O Read :** Accepts data from input devices.
4. **I/O Write :** Sends data to output devices.

All these operations are part of the communication process between the MPU and peripheral devices (including memory). To communicate with a peripheral (or a memory location), the MPU needs to perform the following steps.

Step 1 : Identify the peripheral or the memory location (with its address).

Step 2 : Transfer binary information (data and instruction).

Step 3 : Provide timing or synchronization signals.

The 8085 MPU performs these functions using three sets of communication lines called buses : the *address bus*, the *data bus* and the *control bus*. The buses shown as one group called the system bus. This is called **3 lines concept.**

(1) Address Bus : *Refer to Prob.11.*

(2) Data Bus : *Refer to Prob.12.*

(3) Control Bus : The control bus is comprised of various signal lines that carry synchronization signals. The MPU uses such lines to perform the third function : **providing timing signals.**

The term bus, in relation to the control signals, is somewhat confusing. There is no group of lines like address or data buses, but individual lines that provide a pulse to indicate an MPU operation. The MPU generates specific control signals for every operation (such as Memory Read or I/O Write) it performs. These signals are used to identify a device type with which the MPU intends to communicate.

To communicate with a memory for example, to read an instruction from a memory location the MPU places the 16-bit address on the address bus. The address on the bus is decoded by an external logic circuit, and the memory location is identified. The MPU sends a pulse called memory read as the control signal. The pulse activates, the memory chip and the contents of the memory location (8-bit data) are placed on the data bus and brought inside the microprocessor.

What happens to the data byte brought into the MPU depends on the internal architecture of the microprocessor.

Flag Register : *Refer to Prob.13.*

Prob.16 *State the differences between static and dynamic RAM.* [R.T.U. 2014]

Sol. The two main forms of RAM are static RAM and dynamic RAM. SRAM stores data using the state of a six transistor memory cell. In DRAM, a bit is stored as charge on the capacitor in the transistor-capacitor pair, which together form a DRAM memory cell.

SRAM is more expensive to produce but requires lower power than DRAM. SRAM is generally faster than DRAM.

DRAM must be periodically refreshed as the capacitors slowly discharge due to small leakage current from even "non-conducting" transistors. SRAM exhibits data remanence ,i.e., it need not be refreshed periodically.

The bistable latching circuitry to store each in the SRAM is shown in fig.1.

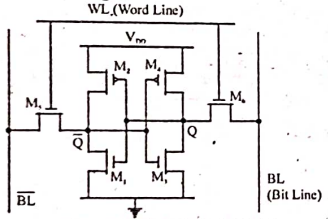


Fig.1 : Bistable latching circuitry (A Six transistor CMOS SRAM cell)

A DRAM cell is shown in Fig.2

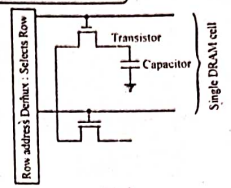


Fig.2

DRAM is the main memory (the "RAM") in personal computers, workstations and video game consoles. SRAM exists as the fast "cache" memory in micro-controllers, microprocessors and to store the registers and parts of state-machines used in some microprocessor.

Prob.17 *How will you demultiplex the address and databus? How can you interface 2048 KB RAM with 8085 microprocessor? Explain.* [R.T.U. 2014]

Sol. On the 8085, pins 19-12($AD_7 - AD_0$) constitute the multiplexed address-data bus. Low order 8 bits of the memory (or I/O) address appear on the AD bus during the first clock cycle (T_1 state of a machine cycle). During T_2 and T_3, the AD bus becomes the data bus. The signal ALE (Address Latch Enable) is high during T_1 and is used to latch these lower 8 address bits, using the on-chip latch of certain peripherals like the 8155. IC chips like the 8155 internally demultiplex the AD bus using the ALE signal. The ALE output of the 8085 is connect to the ALE input of 8155.

Since a majority of peripheral devices do not have the internal multiplexing facility, there is external hardware necessary for it. Fig. 1 shows a schematic that uses a latch and the ALE signal to demultiplex the AD bus. AD_7–AD_0 is connected as input to the latch 74LS373. The ALE signal is connected to the enable (G) pin of the latch, and the Output Control (OC) signal of the latch is grounded. When ALE goes high during the T_1 state of a machine cycle, the latch is transparent in that the output of the latch changes according to the input. The CPU is putting lower-order bits of address during this time. When the ALE goes low, the address bits gets latched on the output and remain so until the next ALE signal.

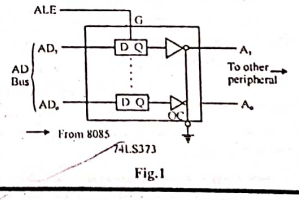


Fig.1

The interfacing of 8085 with 2048 KB RAM is illustrated in Fig. 2.

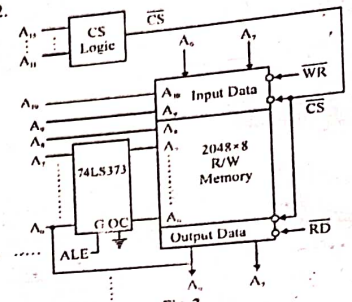


Fig. 2

$A_{15} - A_0$, $\overline{WR}$, $\overline{RD}$, ALE signals are available from 8085. $A_{15} - A_{11}$ may/may not be used to form $\overline{CS}$ signal to select a particular 2048 KB memory. The 74LS373 is used to latch the lower-order address ($A_7 - A_0$) during T_1. The output data/input data to/from the 8085 is connected to $A_0 - A_7$ (signals before the 74LS373 latch).

Prob.18 *Explain through block diagrams the similarities and difference in microprocessor, microcontroller and microcomputer devices.*

OR

Specify the terms microprocessor, microcomputer and microcontroller. [R.T.U. 2008, Raj. Univ. 2003]

OR

Is there any difference among microprocessor, microcontroller and microcomputer? Explain. [R.T.U. 2014]

Sol. Microprocessor : *Refer to Prob.5.*

Microcomputer : A computer that is designed using a microprocessor as its CPU. It includes microcontroller memory and input.

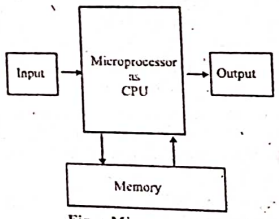


Fig. : Microcomputer

Microcontroller : *Refer to Prob.10.*

Prob.19 *Explain the function of various control and status signals available on 8085 microprocessor.* [R.T.U. 2013]

Sol. Control & Status Signals

There are 2 control signals ($\overline{RD}$ and $\overline{WR}$) and three status signals (IO/$\overline{M}$, S_1 and S_0) to identify the nature of operation and one special signal. ALE for distinguishing the higher order and lower order addresses.

(a) **ALE (Address Latch Enable) :** This signal is generated in starting of the 8085 operation and is used to latch the lower order address from the multiplexed bus and generate a separate set of eight address lines $A_7 - A_0$. If this is one, address will be passed otherwise data will be passed through $AD_7 - AD_0$.

(b) **$\overline{RD}$ (Read) :** This is an active low signal and is used to read memory or I/O device and data will be available on data bus. This signal should be low to perform the read operation.

(c) **$\overline{WR}$ (Write) :** A low on $\overline{WR}$ indicates that data on the data bus which has been placed on it by the processor is to be written into the selected memory or I/O operation.

(d) **IO/$\overline{M}$:** This is a status signal used to differentiate between I/O and memory operation. If it is high, I/O operation and if low, memory operation. It is combined with $\overline{RD}$ and $\overline{WR}$ to generate I/O and memory control signals.

(e) **S_1 and S_0 :** These are also status signals and used to identify various operations, according to the following table :

Table : 8085 Machine Cycle, Status & Control Signal

Machine Cycle	Status			Control Signals
	IO/$\overline{M}$	S_1	S_0	
Opcode Fetch	0	1	1	$\overline{RD}$ = 0
Memory Read	0	1	0	$\overline{RD}$ = 0
Memory Write	0	0	1	$\overline{WR}$ = 0
I/O Read	1	1	0	$\overline{RD}$ = 0
I/O Write	1	0	1	$\overline{WR}$ = 0
Interrupt Acknowledge	1	1	1	$\overline{INTA}$ = 0
Halt	Z	0	0	$\overline{RD}$, $\overline{WR}$ = Z and $\overline{INTA}$ = 1
Hold	Z	X	X	
Reset	Z	X	X	

Here Z=Tri-state/high Impedance ; X=Unspecified

Prob.20 *Explain the difference between the following*
(i) Microprocessor and Microcontroller
(ii) Assembler and Compiler
(iii) High level and low level language
(iv) RAM and ROM [R.T.U. 2012]

Sol. (i) Difference between Microprocessor and Microcontroller :

Table : Comparison Between Microprocessor and Microcontroller

S. No.	Microprocessor	Microcontroller
1.	Microprocessor contains ALU, general purpose register, stack pointer, program counter, clock timing circuit and interrupt 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 more hardware.	Microcontroller based system requires less 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 pins are multifunctioned.
9.	Few bit hiding information.	Many bit hiding information.
10.	High performance pipeline parallel CPU architecture.	Low performance simple CPU Architecture.

(ii) Assembler and Compiler

S. No.	Assembler	Compiler
1.	An assembler can be considered as a special type of compiler which translates only assembly language to machine code.	Compiler is a computer program that reads a program written in a language and translates it in to another language.
2.	It produces an object code which might have to be linked using linker program in order to run on machine.	Compiler usually produces the machine executable code, directly from a higher level language.

(iii) High Level Language and Low Level Language

S. No.	High level language	Low level language
1.	High level languages are easy to learn.	Low level languages are difficult to learn.
2.	These are near to human language.	These are far from human language.
3.	These are easy to modify.	Program in this language are difficult to modify.
4.	Execution of program is slow.	Execution of program is fast.
5.	These languages are normally used to write application program.	These languages are normally used to write hardware program.

(iv) RAM and ROM

S. No.	Features	RAM	ROM
1.	Elaboration	Random Access Memory	Read Only Memory.
2.	Memory type	Internal Memory	External Memory.
3.	Working type	Both Read and Write operation can be performed over the information stored in RAM.	The ROM only allows the user to read the information. User cannot make changes to the information
4.	Type	1. Static RAM (SRAM) 2. Dynamic RAM (DRAM)	1. PROM (Programmable ROM) 2. EPROM (Erasable Programmable ROM) 3. EEPROM (Electrically Erasable Programmable ROM)

Prob.21 *There are two types of read and write in microprocessor interfaces, such as Read and Write from/to input device and output device as well as Read and Write from/to memory device. Explain how they are realized by only three pins in 8085.* [R.T.U. 2009]

Sol.

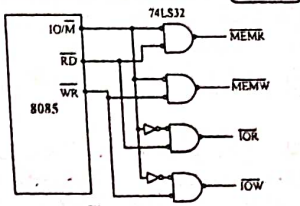


Fig.

As we know that $\overline{RD}$ is a control signal because this signal is used both for reading memory and for reading as input devices. It is necessary to generate two different read signals. One for memory read and another for input. Similarly, two separate write signals must be generated.

The diagram shows that four different control signals are generated by combining the signals $\overline{RD}$, $\overline{WR}$ and IO/$\overline{M}$. The IO/$\overline{M}$ goes low for the memory operation. This signal is ANDed with $\overline{RD}$ and $\overline{WR}$ signals by using 74LS32 quadruple the input or gate. The OR gates are functionally connected as negative NAND gates. When both the input signal goes low the output of the gates low and generate $\overline{MEMR}$ (Memory Read) and $\overline{MEMW}$ (Memory Write) control signals. When the input IO/$\overline{M}$ goes high, it indicates that peripheral I/O operation. Figure shows that this signal is complemented using hex inverter and ANDed with the $\overline{RD}$ & $\overline{WR}$ signals to generate $\overline{IOR}$ (I/O Read) and $\overline{IOW}$ (I/O Write) control signals.

PART-C

Prob.22 *Explain the concept of multiplexing and De-multiplexing of buses.* [R.T.U. 2019]

Sol. The data bus and the low order address bus on the 8085 microprocessor are multiplexed with each other. This allows 8 pins to be used where 16 would normally be required. The hardware interface is required to de-multiplex the bus by latching the low order address in the first T cycle, on the falling edge of ALE.

The address bus is multiplexed in 8085. The multiplexing is done with the help of ALE signal. ALE stands for address latch enable. When ALE is High (Logic 1) : upper address

lines (line 15-8) and lower address lines (line 7-0) combinely holds the 16 bits of the address. When ALE is low (Logic 0): Upper address lines (line 15-8) holds the upper 8 bit address and Lower address lines (line 7-0) holds the "8 bit DATA".

Multiplexing is used to reduce the number of pins of 8085, which otherwise would have been a 48 pin chip. But because of multiplexing, external hardware is required to de-multiplex the lower byte address cum data bus.

The Pin no= 30 of 8085 is the ALE pin which stands for 'Address Latch Enable'. ALE signal is used to de-multiplex the lower order address bus (AD_0 - AD_7).

Pins 12 to 19 of 8085 are AD_0 - AD_7 which is the multiplexed address-data bus. Multiplexing is done to reduce the number of pins of 8085.

Lower byte of address (A_0 - A_7) are available from AD_0 - AD_7 (pins 12 to 19) during T_1 of machine cycle. But the lower byte of address (A_0 - A_7), along with the upper byte A_8 - A_{15} (pins 21 to 28) must be available during T_2 and rest of the machine cycle to access memory location or I/O ports.

Now ALE signal goes high at the beginning of T_1 of each machine cycle and goes low at the end of T_1 and remains low during the rest of the machine cycle. This high to low transition of ALE signal at the end of T_1 is used to latch the lower order address byte (A_0 - A_7) by the latch IC 74LS373, so that the lower byte A_0 - A_7 is continued to be available till the end of the machine cycle. The situation is explained in the following figure :

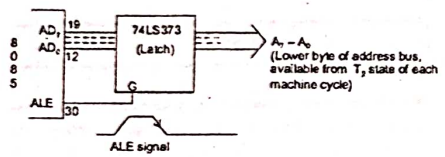


Fig. : Lower byte of address latching achieved by the H to L transition of ALE signal, which occurs at the end of T_1 of each machine cycle

Prob.23 ***Draw and explain the functional block diagram of 8085 microprocessor along with the features in detail.***

[R.T.U. 2019]

OR

Draw the architecture diagram of 8085 microprocessor and explain functions of various registers.

[R.T.U. 2015]

OR

Draw and explain the functional block diagram of 8085 microprocessor.

[Raj. Univ. 2006, 2004, 2002, 2001]

OR

Draw the diagram of 8085 microprocessor architecture. Explain the components.

[R.T.U. 2014]

Sol. Intel 8085 is an 8 bit, NMOS microprocessor. It is a 40 pin IC package fabricated on a single LSI chip.

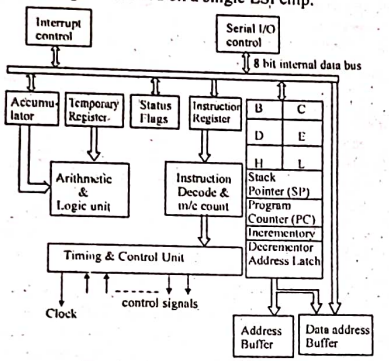


Fig. : Block diagram of 8085 MPU

The intel 8085 uses a single +5V dc supply for its operation. Its clock speed is about 3 MHz. The clock cycle is of 320 ns. The time for the clock cycle of the intel 8085 is 200 ns. It has 80 basic instructions and 246 opcodes.

Fig. shows the block diagram of intel 8085. It consists of three main sections, an arithmetic and logic unit, a timing and control unit and several registers. These important sections are described below:

1. ALU

The arithmetic and logic unit (ALU), performs the following arithmetic and logic operations.

1. Addition
2. Subtraction
3. Logical AND
4. Logical OR
5. Logical Exclusive OR
6. Complement (Logical NOT)
7. Increment (add 1)
8. Decrement (subtract 1)
9. Left shift (add input to itself)
10. Clear

2. Register Array

Register is a group of predefined memory location which is used to store data and result. 8085 has got three types of registers:

(a) General Purpose Registers (GPR) : *Refer to Prob.4.*

(b) Temporary Registers : W and Z are temporary registers which are not available for user. To perform arithmetic and logical operations, microprocessor assumes one data is available in accumulator and takes another and then performs operation on the 2 data bytes. Temporary registers are used by microprocessor for internal operations to store operand, immediate operand or address of memory.

(c) Special Purpose Registers : 8085 microprocessor contains three special purpose registers which are Program Counter (PC), Stack Pointer (SP) and Increment/ Decrement Latch.

(i) Program Counter (PC) : *Refer to Prob.14(a).*

(ii) Stack Pointer (SP) : *Refer to Prob.14(b).*

(iii) Increment/Decrement Latch: This is also a 16–bit register used to increment or decrement the contents of SP and PC registers. Address buffer and address/data buffers are used in co–ordination with these registers. Address buffer is an 8 bit unidirectional buffer used for A_{15} – A_8 address lines. When they are not used, buffer is used to tri–state these lines. Address/Data buffer is an 8–bit bidirectional buffer for address/data lines AD_0 – AD_7 under certain conditions (reset, hold, halt).

3. Interrupt Control

This block is responsible for accepting different interrupt request inputs such as TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR. Interrupts are generated to interrupt the execution of program. When a valid interrupt request is present, it informs control logic to take action in response to each signal.

4. Serial I/O Control Groups

The data transferred via D_0 to D_7 lines is parallel data, but under certain condition it is advantageous to use serial data transfer (bit by bit transmission). Microprocessor 8085 uses SID and SOD signals to transmit data serially. The data on these lines is accepted or transferred under software control by serial I/O control blocks.

5. Instruction Register, Decoder and Control Group

Instruction Register : When an instruction is fetched from memory, it is loaded in instruction register and that instruction is provided to decoder for decoding. This register is only activated when an instruction code or OPCODE is available on internal data bus. It is a non–programmable register.

Instruction Decoder : This accepts a bit pattern from instruction register, decodes it and gives the decoded information to control logic. The information includes what operation is to be performed, who is going to perform it, how many operand bytes the instruction contains etc.

Timing and Control Unit : It contains an oscillator and a controller sequencer. Oscillator is responsible for generating clock signals to synchronize all registers and peripherals for communication to the microprocessor. Controller sequencer accepts information from instruction decoder and generate control signals for internal or external control. $\overline{RD}$ and $\overline{WR}$ are sync pulses indicating the availability of data on the data bus.

Prob.24 ***Draw pin diagram of 8085 and explain each pin.***

[R.T.U. 2017]

OR

Explain the following pins in relations to 8085 microprocessor.

(i) ALE

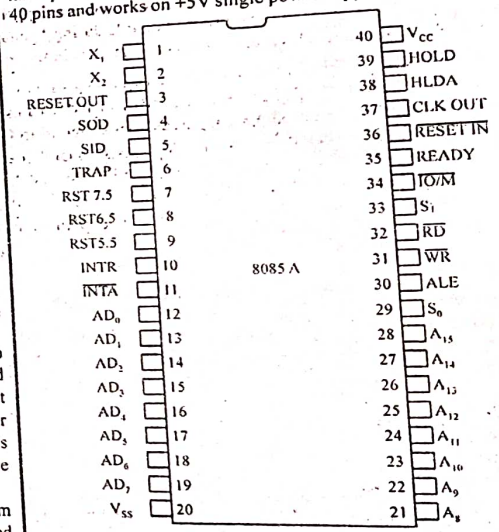
(ii) READY

(iii) S_1 and S_0

(iv) IO/$\overline{M}$

[R.T.U. 2015]

Sol. PIN Diagram of 8085 Microprocessor : The 8085 microprocessor is a 8 bit general purpose microprocessor having 40 pins and works on +5V single power supply.



8085 PIN out

Fig. : PIN Diagram of 8085

Functions of Various Pins

1. Pin No. 01, 02 (X_1, X_2) : A crystal either RC or an LC network is connected at these two pins. The frequency is internally divided by two. Hence to operate the system at 3MHz, the crystal should have a frequency of 6MHz.

2. Pin No. 03 (RESET OUT) : This signal indicates that the microprocessor unit is being reset. This signal can be used to reset other devices.

3. Pin No. 04, 05 (SOD, SID) : These are the serial input / output ports. The 8085 has two signals to implement the serial transmission SID (Serial Input Data) and SOD (Serial Output Data). In serial transmission data bits are sent over a single line, one bit at a time such as the transmission over the telephone lines.

4. Pin No. 07, 08, 09 (RST 7.5, RST 6.5, RST 5.5) : These are restart interrupts. These are vectored interrupts that transfers the program control to specific locations. They have higher priorities than the INTR signal.

5. Interrupt : Interrupt is an external signal which is generated by any peripheral. The 8085 microprocessor has five interrupt signals that can be used to interrupt a program execution. The various interrupts are as follows :

(i) Pin No. 06 (TRAP) : This active high i/p signal TRAP is non maskable interrupt and has highest priority. When pin goes high MPU goes to interruption state (stores the execution instruction.

- TRAP is atomatically vectored transferred.
- Program counter will be transferred to a specific memory call location.
- Call location of TRAP interrupt is 0024 H.
- It also is a active high edge triggered interrupt. Which means I/P should go high and stay high to be acknowledge.

(ii) Pin No. 10 (INTR) Interrupt request : This is used as a general purpose interrupt.

(iii) Pin No. 11 (INTA) : This is an output signal and is generated by microprocessor to acknowledge and interrupt. This is known as interrupt acknowledge pin.

6. Pin No. 12 to 19 (Multiplexed Address/Data bus) and Pin No. 21 - 28 $A_8 - A_{15}$ (Address Bus) : The 8085 microprocessor has 16 signal lines are used as address but; however, these lines are split into segments $A_{15} - A_8$ and unidirectional and used for the most significant bits called higher order address of a 16 bit address.

The signal lines $AD_7 - AD_0$ are used for a dual purpose *i.e.* they are used as the lower order address bus as well as the data bus. During the later part of the cycle, they are used as data bus (this is known as multiplexing the bus). However, the low-order address bus can be separated from these signals by using a latch.

Pin No. 20 (V_{ss}) : It is ground reference for 8085 microprocessor.

Pin No. 31 ($\overline{WR}$ – Write) : This is a write control signal and is active low. This signal in active condition indicates that the data on data bus are to be written into a selected memory or input output location.

Pin No. 32 ($\overline{RD}$ – Read) : This is a read control signal and is active low. This signal in active condition indicates that the selected input/output or memory device is to be read and data are available on the data bus.

Pin No. 36 ($\overline{RESET\ IN}$) : When the signal on this pin goes low, program counter is set to zero, the buses are tristate and the microprocessor unit is reset.

Pin No. 37 (CLK OUT : Clock Output) : This signal can be used as the system clock for another devices.

Pin No. 38 (HLDA - Hold Acknowledge) : To respond the HOLD request on PIN 38, the 8085 µP has one signal called HLDA. This active high O/P signal when MPU receives hold signal from external device. It will send a request to device *i.e.* it receives the request and gives the control of buses to external device.

This pin is used in data transfer through a direct memory access.

Pin No. 39 (HOLD) : When the HOLD pin is activated by an external signal generated by peripheral, the microprocessor relinquishes the control of address and data buses and allows the requesting peripheral to use them.

Pin No. 40 (V_{cc}) : It is a 5V power supply for 8085 microprocessor.

Pins in Relations to 8085 Microprocessors :

(i) ALE : *Refer to Prob.2.*

(ii) Ready : The 8085 microprocessor has a pin called READY. If the signal at this READY pin is low, the microprocessor enters into the wait state. The signal is used primarily to synchronize slower peripherals with the microprocessor. This signal actually delays the microprocessor Read or Write cycles until a slow responding peripheral is ready to see or accept data.

(iii) S_1 and S_0 : These are status signals. These can identify various operations, but they are rarely used in small systems.

(iv) IO / $\overline{M}$: This is a status signal and is used to differentiate between input/output and memory operations. This signal is combined with $\overline{RD}$ (read) and $\overline{WR}$ (write) to generate input / output and memory control signal.

Prob.25 *Define following in 8085 Microprocessor :*

(a) Program counter
(b) Stack pointer
(c) General purpose programmable registers
(d) Instruction register
(e) HOLD and HLDA pin
(f) Control and status pins
(g) Flags
(h) PSW **[R.T.U. 2016]**

Sol.(a) Program Counter : *Refer to Prob.14(a).*

(b) Stack Pointer : *Refer to Prob.14(b).*

(c) General Purpose Programmable Registers : *Refer to Prob.4.*

(d) Instruction Register : *Refer to Prob.23.*

(e) HLDA Pin : *Refer to Prob.24.*

HOLD Pin : *Refer to Prob.24.*

(f) Control and Status Pins (Bus)

1. $IO/\bar{M} = \begin{cases} 1 = \text{data transfer from I/O.} \\ 0 = \text{data transfer from memory.} \end{cases}$
2. $S_1, S_0 \Rightarrow$ by measuring logic one that we can find out which machine cycle going on inside the microprocessor.
3. $\overline{WR} = 0 =$ Write operation can be performed.
4. $\overline{RD} = 0 =$ read operation can be performed.
5. $\overline{WR}$ and $\overline{RD}$ does not operate simultaneously,

   $\left.\begin{matrix} \overline{WR} = 0 \\ \overline{RD} = 0 \end{matrix}\right\}$ Wrong

   $\left.\begin{matrix} \overline{WR} = 1 \\ \overline{RD} = 1 \end{matrix}\right\}$ Right
6. Mixed directional
   ALE (Address Latch Enable)
   $= 1 \Rightarrow A_{D0} - A_{D7}$ as address pins
   $= 0 \Rightarrow A_{D0} - A_{D7}$ as data pins
   Latch 74LS373
   D-flip flop buffer
   $ALE = 1 \Rightarrow A_{D0} - A_{D7}$ as address line
   $ALE = 0 \Rightarrow A_{D0} - A_{D7}$ as data
   Lower order eight bins of address pins work also as data pins.

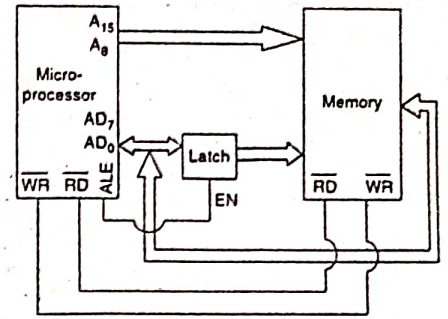


Fig. : Block diagram showing ALE Action

Multiplexing and demultiplexing of lower order eight pins of address pins as data pins is time diversion multiplexing.

(g) Flags : *Refer to Prob.13.*

(h) PSW : The program status word consists of five status flags. These are described below :

(i) The carry flag (Cy) reflects the final carry out of the most significant bit of any arithmetic operation. Any logic instruction resets or clears the carry flag. This flag is also used by the shift and rotate instructions. The 8085 does not have any CLEAR CARRY instruction. One way of clearing the carry will be by ORing or ANDing the accumulator with itself.

(ii) The parity status flag (P) is set to 1 if arithmetic or logic instruction generates an answer with even parity, that is, containing an even number of 1 bits. This flag is 0 if the arithmetic or logic instruction generates an answer with odd parity, that is, containing an odd number of 1s.

(iii) The auxiliary carry flag (Ac) reflects any carry from bit 3 to bit 4 (assuming 8-bit data with bit 0 as the least significant bit and bit 7 as the most significant bit) due to an arithmetic. The auxiliary carry flag (Ac) reflects any carry from bit 3 to bit 4 due to an airthmatic operation. This flag is useful for BCD operations.

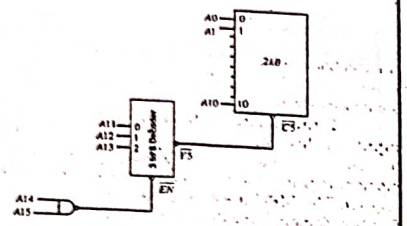
(iv) The zero flag (Z) is set to 1 whenever an arithmetic or logic operation produces a result of 0. The zero flag is cleared to zero for a non zero result due to arithmetic or logic operation.

(v) The sign status flag (S) is set to the value of the most significant bit of the result in the accumulator after an arithmetic or logic operation. This provides a range of -128_{10} to $+127_{10}$ (with 0 being considered positive) as the 8085's data-handling capacity.

The 8085 does not have an overflow flag. Note that execution of arithmetic or logic instructions in the 8085 affects the flags. All conditional instructions in the 8085 instruction set use one of the status flags as the required condition.

Prob.26 *(a) Define multiplexed pins of 8085 Microprocessor. Explain with block diagram Demultiplexing of AD0-AD7 pins of 8085 using tri state gates.*

(b) Find memory map of the following 2 KB Memory chip.



[R.T.U. 2016]

Sol.(a) PIN Diagram of 8085 Microprocessor : *Refer to Prob.24.*

Sol.(b) The 2K memories IC have 11 address lines A_{10}-A_0, which are used to locate the memory location where data will be stored or read. The other address lines A_{11}-A_{15} of the microprocessor can be used for the chip select signal. The memory map of 2K memories is given below:

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	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	=0000H
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	=07FFH

Prob.27 *Explain the need to demultiplex the bus AD_0-AD_7.* [R.T.U. 2015]

OR

Define and explain the demultiplexing of address-data bus of 8085. [R.T.U. 2010]

OR

Why are $AD_7 - AD_0$ lines multiplexed? With the help of latching circuit, explain how these lines are demultiplexed. [R.T.U. 2013, 09, 2008, Raj. Univ. 2005, 2002]

Sol. Need of Multiplexing : *Refer to Prob.3.*

Demultiplexing the Buses : $AD_0 - AD_7$ are used in multiplexed mode, they carry both data and address. Within a fetch cycle the address is sent during the first state of the cycle. During the second state the content of the specified memory location is read and during the third state the content is transferred into the microprocessor through the same $AD_0 - AD_7$ lines/pins. The address/data bus, $AD_0 - AD_7$, is bi-directional where as the address bus $A_8 - A_{15}$ is unidirectional.

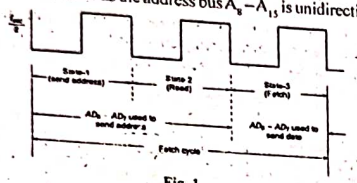


Fig. 1

In 8085 the higher order address lines are dedicated to carry the higher order address but the lower order address lines are multiplexed with data bus. Thus, we can say that the need to demultiplex the bus $AD_0 - AD_7$ arises due to following reasons :

1. $AD_0 - AD_7$ bus are demultiplexed in 8085 microprocessor to provide individual 16-bit address lines and 8-bit data lines. As these lines are multiplexed, number of pins required are less so hardware cost is less.

2. Timing : Transfer of byte from memory to MPU diagram shows that address on the high order bus (20H) remains on the bus for three clock periods. However the low order bus (50 H) is lost after the first clock period. This address needs to be latched and used for identifying the memory address. If the bus $AD_0 - AD_7$ is used to identify the memory location (2005 H), the address will change to 204FH after the first clock period.

3. The need for demultiplexing the bus $AD_0 - AD_7$ becomes easier to understand after examining.

To do this demultiplexing first we see how long multiplexed address/data is available on AD_0 - AD_7. During T_1 state of every machine cycle AD_0-AD_7 lines carry the address part. In this T_1-state ALE signal is also high to indicate address on these lines. After T_1-state microprocessor 8085 will remove the address contents from AD_0-AD_7 lines and use these lines as data lines for next clock cycle onwards.

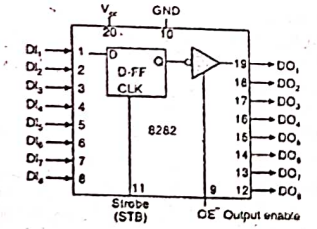


Fig. 2 : Pin diagram of D-latch 8282

The demultiplexing of these lines is done by using an 8-bit D-latch IC (shown in Fig. 2) along with the ALE signal as shown in Fig. 3. When ALE is high the address signals will get latched in 8-bit latch and the output of latch will provide A_0-A_7 address contents. When ALE is low for data, it will disable the latch from accepting contents from AD_0-AD_7 lines and the same lines can be used as data lines.

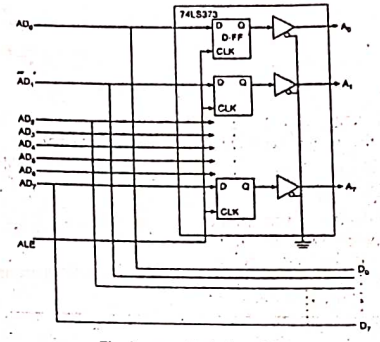


Fig. 3 : Demultiplexing of AD_0-AD_7

Here, in the circuit the D-FF(Flip Flop) are triggered by the ALE signal. When these FFs are triggered their output becomes equal to the input which is nothing but address at this moment. Now when even if the ALE signal goes down these FF retain the address component at their output for peripheral addressing. The tri-stated buffers are used just to increase the driving capabilities of the buses.

Prob.28 *Write comparison between memory mapped I/O and peripheral mapped I/O.* [R.T.U. 2012]

OR

Explain the difference between the peripheral I/O and the memory mapped I/O. Define logic and explain the functions of following devices :

(a) Buffer (b) Decoder
(c) Encoder (d) Latch

[R.T.U. 2008, Raj. Univ. 2002]

OR

What is Peripheral I/O and Memory-Mapped I/O? Differentiate between them. [Raj. Univ.2007, 2004]

Sol. Peripheral I/O and memory mapped I/O : The input techniques are applied on input devices. The input devices are the devices through which the MPU communicates with the user or the outside world. There are two different methods by which input devices can be identified.

1. Inputs with 8-bit addresses or peripheral mapped input or input mapped input : In this the MPU uses eight address lines to identify an input or an output device. This is an 8-bit numbering system for inputs used in conjunction with input and output instructions. This is also known as input space separate from memory space, which is a 18-bit numbering system. The eight address lines can have 256 (2^8 combinations) address, thus, the MPU can identify 256 input devices and 256 output devices with address ranging from 00 to FFH. The input and output devices are differentiated by control signals, the MPU uses the input read control signal for input devices and the output write control signal for output devices.

2. Input with 16-bit address or memory mapped input : In this type of input, the MPU uses 16 address lines to identify an input device. An input is connected as if it is memory register. The MPU is connected as if it is a memory register. The MPU uses the same control signal (Memory Read or Memory write) and instructions as those of memory. In memory mapped input, the MPU follows the same steps as if it is accessing a memory register. Sequentially from floor to floor instead of moving from floor to floor as requested, the head is moved to access files in the sequence they are located on disk tracks rather than strictly in the sequence they were requested.

(a) Buffer : When the output current of a digital device is insufficient to drive another device, which is to be connected to the output terminal of the digital devices, a buffer is used to increase the current rating. A buffer increases the output current. A buffer is represented by a triangle. For an inverting buffer a bubble is put at the output point of the triangle. Sometimes, a change in voltage level is required to interface lamps, relays etc. This is also achieved by suitable buffers/drivers. Thus we see that the function of buffer/driver is to change the current/voltage levels. Following table shows important buffer/drivers.

Buffer/Driver ICs

IC No.	Description
7406, 7416	Hex Inverter Buffer/Drivers with open collector outputs.
7407, 7417	Hex Buffer/Drivers with open collector outputs.
74125	Quad Bus Buffers Gates with 3-state outputs, output is enabled when C is low.
74126	Quad Bus Buffers Gates with 3-state outputs, outputs is enabled when C is high.
74240	Octal buffers/Line Drivers/Line Receivers, inverted 3-state outputs.
74241, 74244	Octal Buffers/Line Drivers/Line Receivers, Non-inverted 3-state outputs.
7426, 7437	Quad 2-input NAND Buffers.

7438	Quad 2-input NAND Buffers with open collector outputs.
7440	Dual 4-input NAND Buffers.
7428	Quad 2-input NOR Buffer.
7433	Quad 2-input NOR Buffer with open collector outputs.
74128	Quad 2-input NOR Line Drivers.

(b) and (c) Decoders/Encoders : A digital computer, microprocessor based system or any other digital system uses binary numbers for its operation. They understand information composed of only 0's and 1's. The user is allowed to use decimal numbers and alphabetic characters while working on a digital system for his convenience. A suitable interfacing circuit is used to convert decimal numbers alphabetic character to proper binary forms required by the digital system. Various kinds of binary codes are used by digital systems such as BCD (Binary coded decimal), Hexadecimal, Octal, Excess-3, Gray code, ASCII, EBCIDC, etc. The function of an encoder is to generate a binary code, the process of generating binary codes is known as encoding. The decoding is the reverse process of encoding.

IC No.	Description
7441, 74141	BCD to Decimal Decoders/ Drivers, open collector outputs, Nixie tube driver.
7442	BCD to Decimal Decoders/ Drivers, totem pole outputs, LED driver.
7445, 74145, 74445	BCD to Decimal Decoders/ Drivers, Open collector outputs, Indicator/Relay driver.
7446, 7426	BCD to 7-Segment Decoders/ Drivers, active low, open collector outputs, 30V outputs.
7447, 74247	BCD to 7-Segment Decoders/ Drivers, active low, open collector outputs, 15V outputs.
7448, 74248	BCD to 7-Segment Decoders/ Drivers, internal pull-up outputs, active high.
7449, 74249	BCD to 7-Segment Decoders/ Drivers, open collector outputs, active high.

The examples of encoder ICs are :

74147 Decimal to BCD encoders.

74148 and 74348 Octal to Binary encoders.

The hexadecimal to binary encoder is realized using two 74148 and a data selector. Examples of decoders are : BCD to decimal decoder/driver, BCD to 7-segment decoder.

(d) Latch : A flip-flop in its simplest form is called a *latch*. A latch stores a binary bit 1 or 0, the unclocked simple flip-flop and level-triggered R-S and D flip-flop fall under the catego of latches. An n-bit latch is composed of n number of 1-b latches. The n bits of a binary word are transferred to tl latch simultaneously in parallel. There is no facility to read i contents. The latches are used as temporary storage device They are ideally suited for storing information betwe processing units and I/O devices or indicating devices.

Difference between peripheral I/O and mapped I/O :

Characteristics	Memory Mapped I/O	Peripheral Mapped I/O
1. Device Address	16 bit	8 bit
2. Control Signal for Input / Output	$\overline{MEMR}$ / $\overline{MEMW}$	$\overline{IOR}$ / $\overline{IOW}$
3. Instruction Available	Memory-related instructions such as STAX; MOV M, R; ADD M; SUB M; ANA M etc.	IN and OUT
4. Data Transfer	Between any register and I/O	Only between I/O and the accumulator.
5. Maximum no. of I/Os possible	The memory map (64K) is shared between I/Os and system memory.	The I/O map is independent of the memory map; 256 input devices and 256 devices output can be connected.
6. Execution Speed	13 T - states STA, LDA of T-states (MOV M, R)	10 T-states
7. Hardware Requirements	More hardware is needed to decode 16-bit address.	Less hardware is needed to decode 8 bit address.
8. Other Features	Arithmetic or logical operation can be directly performed with I/O data	Not available.

Prob.29 *(a) Why a microprocessor needs ROM an RAM? Write the procedure to communicat with memory by a microprocessor.* [R.T.U. 2011

(b) Explain the function of flag register in M 8085. What is the use of the general purpos registers CY and Z? [R.T.U. 2011

Sol.(a) Memory Organization of 8085 : Microprocesso is an useless chip without peripheral and memory unit Memory unit plays an important part in the operation of th microprocessor. Memory refers to any device that store information for later use.

The memory of a computer can be divided into tw categories, they are :

1. Primary or Main Memory
2. Secondary or Mass Storage Device

Primary memories are semiconductor memories Commonly used semiconductor memories are :

1. Read Only Memory (ROM)
2. Programmable Read Only Memory (PROM)
3. Electrically Alterable ROM (EAROM)
4. Random Access Memory (RAM) or Read/Write Memory (RWM)

All the above memories are random access memories. Any bit stored in these memories may be accessed directly. Memories, in general, are classified on the basis of several attributes.

ROM is the acronym for 'Read Only Memory' : ROM is an internal memory in the computer which is used to store critical information required by the computer like booting programs. They are prerecorded and are read only.

RAM is the acronym for 'Random Access Memory' : RAM is another form of internal memory. RAMs are used to store temporary programs that are currently being executed. They come in large sizes of 4GBs and higher. They are accessible and can be changed.

Procedure to Communicate with Memory by a Micro-processor :

Generally, a microprocessor performs four different operations : memory read, memory write, input/output read and input/output write. In the memory read operation, data will be read from memory and in the memory write operation, data will be written in the memory. Data input from input devices are I/O read and data output to output devices are I/O write operations.

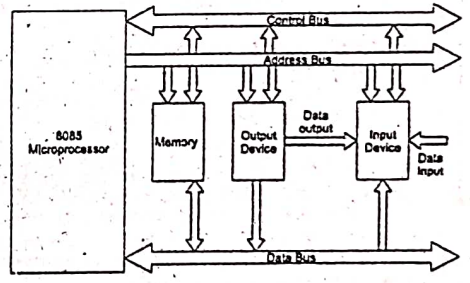


Fig. 1: Bus structure of 8085 microprocessor

The memory read/write and Input/Output read and write operations are performed as part of communication between the microprocessor and memory or Input/Output devices. Microprocessors communicate with the memory and I/O devices through address bus, data bus and control bus as depicted in fig.1. For this communication, firstly the microprocessor identifies the peripheral devices by proper addressing. Then it sends data and provides control signal for synchronization.

Fig. 2 shows the memory read operation. Initially, the microprocessor places a 16-bit address on the address bus. Then the external decoder logic circuit decodes the 16-bit address on the address bus and the memory location is identified. Thereafter, the microprocessor sends $\overline{MEMR}$ control signal which enables the memory IC. After that, the content of the memory location is placed on the data bus and also sent to the microprocessor. Fig. 3 shows the data flow diagram for data transfer from the memory to microprocessor. The step-by-step procedure of data flow is given below :

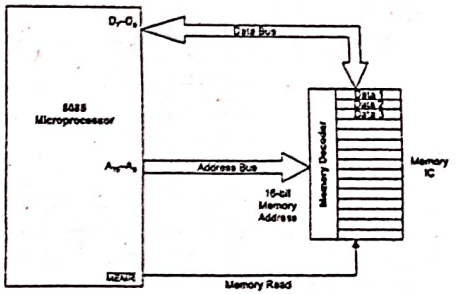


Fig. 2: Memory read operation

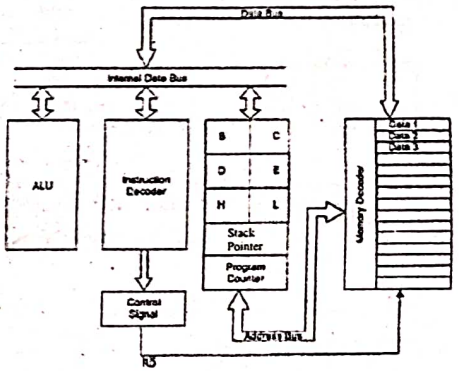


Fig. 3: Data flow from memory to microprocessor

- The 16-bit memory address is stored in the program counter. Therefore, the program counter sends the 16-bit address on the address bus. The memory address decoder is decoded and identifies the specified memory location.
- The control unit sends the control signal $\overline{RD}$ in the next clock cycle and the memory IC is enabled. $\overline{RD}$ is active for two clock periods.

- When the memory IC is enabled, the byte from the memory location is placed on the data bus AD_7–AD_0. After that data is transferred to the microprocessor.

Sol.(b) Flag Register : *Refer to Prob.13.*

Prob.30 ***Define and explain the following :***
(a) Microprocessor and Microcontroller
(b) Machine and Assembly Language **[R.T.U. 2010]**

Sol.(a) A semiconductor device (integrated circuit) manufactured by using the LSI techniques. It includes the ALU, register arrays and control circuits on a single chip. The term MPU is also synonymous with the microprocessor.

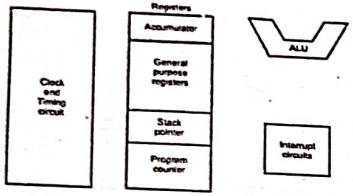


Fig. 1: Block diagram of a microprocessor

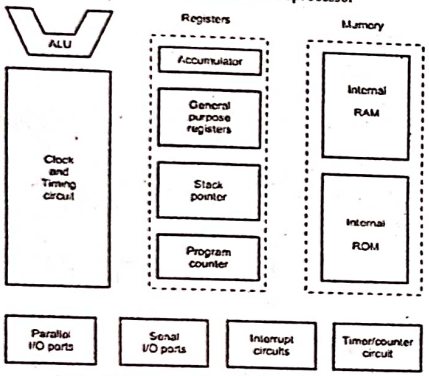


Fig. 2: Block diagram of a microcontroller

Microcontroller is a single chip microcomputer. The word lengths of a micro computer may be 8 bit, 16 bit, 32 bit, or 64 bit. It is generally used in traffic lights, aeroplanes, and radar etc.

A microcontroller differs from a microprocessor in several important ways. Early name for a microcontroller was micro computer. The main difference between a microprocerssor and microcontroller is the completeness of the machine each represents. In order to put a microprocessor into use the designer required memory peripheral chips and serial and parallel ports to make completely funtional computer. On the other hand a complete computer based system could be build using a single chip microcontrollers, with a minimum of external components, Fig.1 shows simplified block diagram of a microprocessor, which consists of an arithmetic and logic unit (ALU), general purpose registers, stack pointer (SP), program counter (PC), clock timing circuit and interrupt circuit.

To make a complete microcomputer system only microprocessor is not sufficient but it is required to add other peripherals such as read only memory (ROM), read/write memory (RAM), decoders, drivers, number of input/output devices to make a complete microcomputer system. In addition, special purpose devices, such as interrupt controller, programmable timers, programmable I/O devices, DMA controllers may be added to improve the capacity and performance and flexibility of a microcomputer system.

The microcontroller incorporates all the features that found in microprocessor, apart it has also added features to make a complete microcomputer system on its own. The microcontroller has built-in ROM, RAM, parallel I/O, serial I/O, counters and a clock circuit. Fig.2 shows the simplified block diagram of a microcontroller.

Table : Comparisons between Microprocessor and Microcontroller : *Refer to Prob.20(i).*

Sol.(b) Machine and Assembly Language

Machine language

All digital circuits can identify only two possible states and which can be specified by 0's and 1's. Hence, microprocessor also can be instructed only using combination of 0's and 1's. Such a language, which comprises of 0's and 1's is called machine language.

The speed of execution of machine language is fastest as it is directly understood by the computer. Programming is difficult and time consuming. Programmer has to search for the binary code for any task for perform.

Assembly Language

To overcome the difficulty of machine language, assembly language was designed, in which each statement of machine language was written in some set of characters which are easier to understand called mnemonics. The program which converts assembly language to machine language is called assembler.

Prob.31 ***What are ASCII code, BCD code, signed and unsigned integer ? Clearly specify how do you work with these numbers in microprocessors ?*** **[R.T.U. 2008]**

Sol. ASCII and Unicode Data : ASCII (American Standard Code for Information Interchange) data represents alphanumeric characters in the memory of a computer system (see Table 1). The standard ASCII code is a 7-bit code, with the eighth and most significant bit used to hold parity in some antiquated systems. If ASCII data are used with a printer, the most significant bits are a 0 for alphanumeric printing and 1 for graphics printing. In the personal computer, an extended ASCII character set is selected by placing a 1 in the leftmost bit. Table 2 shows the extended ASCII character set, using code 80H-FFH. The extended ASCII characters store some foreign letters and punctutation, Greek characters, mathematical characters, box-drawing characters and other special characters.

Table 1 : ASCII code

First \ Second	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
0X	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1X	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2X	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3X	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4X	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5X	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6X	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7X	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

Table 2 : Extended ASCII code, as printed by the IBM proprinter

First \ Second	X0	X1	X2	X3	X4	X5	X6	X7	X8	X9	XA	XB	XC	XD	XE	XF
0X		☺	☻	♥	♦	♣	♠	•	◘	○	◙	♂	♀	♪	♫	☼
1X	►	◄	↕	‼	¶	§	▬	↨	↑	↓	→	←	∟	↔	▲	▼
8X	Ç	ü	é	â	ä	à	å	ç	ê	ë	è	ï	î	ì	Ä	Å
9X	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü	¢	£	¥	₧	ƒ
AX	á	í	ó	ú	ñ	Ñ	ª	º	¿	⌐	¬	½	¼	¡	«	»
BX	░	▒	▓	│	┤	╡	╢	╖	╕	╣	║	╗	╝	╜	╛	┐
CX	└	┴	┬	├	─	┼	╞	╟	╚	╔	╩	╦	╠	═	╬	╧
DX	╨	╤	╥	╙	╘	╒	╓	╫	╪	┘	┌	▒	▒	▒	▒	▒
EX	α	ß	Γ	π	Σ	σ	µ	τ	Φ	Θ	Ω	δ	∞	φ	ε	∩
FX	≡	±	≥	≤	⌠	⌡	÷	≈	°	∙	·	√	ⁿ	²	■	

Note that extended characters can vary from one printer to another. The list provided is designed to be used with the IBM Proprinter, which also matches the special character set found with most word processors.

The ASCII control characters, also listed in Table 1 perform control functions in a computer system, including clear screen, backspace, line feed, and so on. To enter the control codes through the computer keyboard, hold down the Control key while typing a letter. To obtain the control code 01H, type a Control-A; a 02H is obtained by a Control-B and so on. Note that the control codes appear on the screen, from the DOS prompt, as ^A for Control-A, ^B for control-B and so forth. Also note that the carriage return code (CR) is the Enter key on most modern key-boards. The purpose of CR is to return the cursor or print head to the left margin. Another code that appears in many programs is the line feed code (LF), which moves the cursor down one line.

To use Table 1 or 2 for converting alphanumeric or control characters into ASCII characters, first locate the alphanumeric code for conversion. Next, find the first digit of the hexadecimal ASCII code. Then find the second digit. For example, the capital letter "A" is ASCII code 41H and the lowercase letter "a" is ASCII code 61H. Many windows-based applications, since Windows 95, use the Unicode system to store alphanumeric data. This system stores each character as 16-bit data. The codes 0000H-00FFH are the same as standard ASCII code. The remaining codes, 01002H-FFFFH, are used to store all special characters from many world wide character sets. This allows software written for the Windows environment to be used in many countries around the world.

BCD (Binary-Coded Decimal) Data

Binary-coded decimal (BCD) information is stored in either packed or unpacked forms. Packed BCD data are stored as two digits per byte and unpacked BCD data are stored as one digit per byte. The range of a BCD digit extends from 0000_2 to 1001_2 or 0-9 decimal. Unpacked BCD data are returned from a keypad or keyboard. Packed BCD data are used for some of the instructions included for BCD addition and subtraction in the instruction set of the microprocessor.

Table 3 shows some decimal numbers converted to both the packed and unpacked BCD forms. Applications that require BCD data are point-of-sales terminals and almost any device that performs a minimal amount of simple arithmetic. If a system requires complex arithmetic, BCD data are seldom used because there is no simple and efficient method of performing complex BCD arithmetic.

Table 3 : Packed and unpacked BCD data

Decimal	Packed		Unpacked		
12	0001 0010		0000 0001	0000 0010	
623	0000 0110	0010 0011	0000 0110	0000 0010	0000 0011
910	0000 1001	0001 0000	0000 1001	0000 0001	0000 0000

Byte-Sized Data

Byte-sized data are stored as unsigned and signed integers. Figure illustrates both the unsigned and signed forms of the byte-sized integer. The difference in these forms is the weight of the leftmost bit position. Its value is 128 for the unsigned integer and minus 128 for the signed integer. In the signed integer format, the leftmost bit represents the sign bit of the number, as well as a weight of minus 128, For example, 80H represents a value of 128 as an unsigned number; as a signed number, it represents a value of minus 128. Unsigned integers range in value from 00H to FFH (0-255). Signed integers range in value from –128 to 0 to + 127.

Although negative signed numbers are represented in this way, they are stored in the 2's complement form. The method of evaluating a signed number by using the weights

of each bit position is much easier than the act of 2's complementing a number to find its value. This is especially true in the world of calculators designed for programmers.

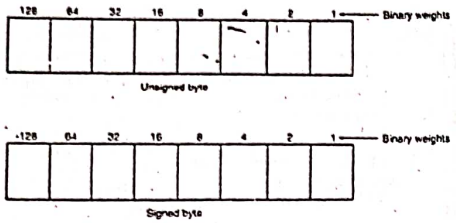


Fig. : The unsigned and signed bytes illustrating the weights of each binary-bit position

Whenever a number is 2's complemented, its sign changes from negative to positive or positve to negative. For example, the number 00001000 is a +8. Its negative value (–8) is found by two's complementing the +8. To form a two's complement, first one's complement the number. To one's complement a number, invert each bit of a number from zero to one or from one to zero. Once the one's complement is formed, the two's complement is found by adding a one the one's complement. Example 1 shows how numbers 2's complemented using this technique.

Example 1

+ 8 = 00001000
11110111 (1's complement)
\+ 1
–8 = 11111000 (2's complement)

Another, and probably simpler, technique for 2 complementing a number starts with the rightmost digit. St by writing down the number from right to left. Write t number exactly as it appears until the first one. Write dov the first one and then invert all bits to its left. Example shows this technique with the same number as in example

Example 2

+8 = 00001000
1000 (write number to first 1)
1111 (invert the remaining bits)
–8 = 11111000

□□

ASSEMBLY LANGUAGE PROGRAMMING 2

CHAPTER IN A NUTSHELL

❑ **Assembly Language**
A medium of communication with a computer in which programs are written in mnemonics. An assembly language is specific to a given computer.

❑ **Instruction Set :** All the instructions are described fully in terms of its operation and the operand, including details such as number of bytes, machine cycle, T-states hex code and affected flags.

The various abbreviation used- Reg. 8080A/ 8085 Register.

Mem = Memory Location
R = Register
R_s = Register Source
R_d = Register Destination
M = Memory
( ) = Contents of
XX = Random Information

❑ **Flags :** All flags are modified to reflect the result of addition.

S = Sign
Z = Zero
AC = Auxiliary Carry
P = Parity
CY = Carry

Example :

Opcode	Operand	Byte	M-cycle	T-States	Hex code
ACI	8–bit data	2	2	7	CE

Description : The 8–bit data (operand) and the carry flag are added to the contents of the accumulator and the result is stored in accumulator.

❑ **Program Structure :** These are of 3 types.

(i) **Sequential :** A program with top location approach, without much complexity.
(ii) **Conditional :** Conditions are applied in the form of conditional loops and counters.
(iii) **Iterative/ Recursive :** Some execution of a program counter for a finite number of times.

PREVIOUS YEARS QUESTIONS

PART-A

Prob.1 Define instruction & instruction set. **[R.T.U. 2019]**

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

Instruction Set : All the instructions are described fully in terms of its operation and the operand, including details such as number of bytes, machine cycle, T-states hex code and affected flags.

The various abbreviation used- Reg. 8080A/ 8085 Register.

Mem = Memory Location
R = Register
R_s = Register Source

R_d = Register Destination
M = Memory
( ) = Contents of
X X = Random Information

Prob.2 *What is the role of accumulator in Microprocessor?*

[R.T.U. 2019]

Sol. Accumulator : The accumulator is a 8 bit general purpose register connected to internal data bus and to ALU. It is also called as a register. It is used in most of the arithmetic and logical operations. After performing an operation, the ALU places its result on internal data bus, from there it is generally stored in accumulator.

Prob.3 *Find the content of 'C' register after execution of the following assembly language program.*

```
        MVI A,   17 H
Loop:   RLC
        JNC      Loop :
        MOVC,    A
        HLT
```

[R.T.U. 2016]

Sol.

17-000101110

RLC: Each binary bit of the accumulator is rotated left by one position. Bit D7 is placed in the position of D0 as well as in the Carry flag. CY is modified according to bit D7. Any other bit is not affected.

The loop moves 4 times

Thus the 'RLC' is called 4 times

Thus C- 10111000

Prob.4 *Find content of register 'C' after execution of the following assembly language program.*

```
MVI A,   67 H
MVI B,   24H
ADDB
DAA
MOV C, A
HLT
```

[R.T.U. 2016]

Sol.

67	→ A	01100111
24	→ B	00100100
		10001011 [8B]

ADD B		10001011
8B	→ A	00000110
DAA		10010001 [91] A

MOVC, A

91 → C

Prob.5 *Compare the function of the RST and RET.*

[R.T.U. 201]

Sol. Comparison between RST and RET instructions

S. No.	RET	RST
1.	This instruction is used to Return from subroutine to the main program unconditionally.	This instruction is used to transfer the control from the program execution to a specific location.
2.	It has following addressing : [PCL] ← [SP] [PCH] ← [[SP]+1] [SP] ← [SP]+2	It has following addressing : [[SP]–1] ← [PCH] [[SP]–2] ← [PCL] [SP] ← [SP]–2 [PC] ← 8 × n
3.	It is 1-byte only opcode instruction.	It is also 1-byte opcode instruction.
4.	It has 3 machine cycles as : (a) Opcode fetch-4T states (b) memory read-3T sates (c) memory read-3T states	It has 3 machine cyles as : (a) Opcode fetch-6T states (b) memory read-3T state [PCH] (c) memory read-3T state [PCL]
5.	RET saves the address back from the stack pointers to program counter	RST does not store previous address.
6.	It is used where subpart of program is used.	It is the type of interrupts. It is used where software interrupt is required.

Prob.6 *Compare the function of the JMP and CALL.*

[R.T.U. 2013]

Sol. Comparison between JMP and CALL instructions

S. No.	JMP	CALL
1.	It is used to Jump the execution of program at any specified address unconditionally.	It is used to execute the subroutine programs instruction unconditionally.
2.	It is 16-bit address.	It is also 16-bit address.
3.	It has 3 machine cycles and 10-T states.	It has 5 machine cycles and 18-T states.
4.	It has Hex code C3H	It has Hex Code CDH.

Prob.7 *Compare the function of the XTHL and XCHG*

[R.T.U. 2013]

Sol. Comparison of XTHL and XCHG instructions

S. No.	XTHL	XCHG
1.	This instruction is used to exchange the contents of HL pair to stack pointer's top.	This instruction is used to exchange the contents of HL pair with DE pair.
2.	Syntex : [L] ← [SP] [H] ← [SP + 1]	Syntex : [HL] ↔ [DE]
3.	It is 1-byte instruction.	It is also 1-byte instruction.
4.	It has 5 machine cyles and 16-T states as follows : (a) opcode fetch-4T states (b) memory read-3T states (c) memory read-3T states (d) memory read-3T states (e) memory read-3T states	It has 1 machine cycles and 4-T states as follows. Opcode fetch-4T states.
5.	Hex code for XTHL instruction is E3H.	Hex code for XCHG instruction is EBH.

Prob.8 *Compare the function of the STAX and LDAX.*

[R.T.U. 2013]

Sol. Comparison between STAX and LDAX

S. No.	STAX	LDAX
1.	Store the content of accumulator in memory location whose address is specified in the register pair.	Load the contents of memory location whose address is specified in the register pair to accumulator.
2.	It uses BA and DE pair.	It is also uses BC and DE pairs.
3.	Syntex : [A] ← [r_p]	Syntex : [r_p] ← [A]
4.	It has 2 machine cycle and 7-T state	It has 2 machine cycle and 7T state.

Prob.9 *Write a program to store hexadecimal data AB in memory location 2300, split the data in the from of OA and OB, and store them in memory location 2501 and 2502.*

[R.T.U. 2013]

Sol.

```
LXI   H, 2300    Load HL with 2300 H
MVI   M, AB H    Store AB H in memory location
                 pointed by HL register pair
                 (2300H)
MOV   B, M       Get the data in B-reg
MOV   A, B       Copy the data to A-reg
ANI   OF H       Mask the upper nibble
MOV   M,A
STA   2501       Store the lower nibble in memory
MOV   A, B       Get the data in A-reg
ANI   FO H       Bring the upper nibble to lower
                 nibble position
RRC
RRC
RRC
RRC
MOV   M, A
STA   2502       Store the upper nibble in memory
HLT              Terminate the program
```

Prob.10 *Write contents of all registers after every instruction given program segment.*

```
MVI C, 10 H
LXI H, 2370 H
LXI D, 7870 H
MOV A, M
STAX D
MOV M, C
LDAX D
HLT
```

[R.T.U. 2009]

Sol. MVI C, 10 H

[CReg] ← [10] ⇒ C=10

LXI H, 2370 H

[HL Reg] ← [2370] ⇒ HL=2370 H

⇒ H = 23 L = 70

LXI D, 7870 DE = 7870

D = 78 E = 70

MOV A,M A ← (2370) load 2370 data into accumulator

STAX D A → (7870) store accumulator data into 7870 memory location

OR (2370) → (7870)

MOV M, C M = (2370) = C = 10

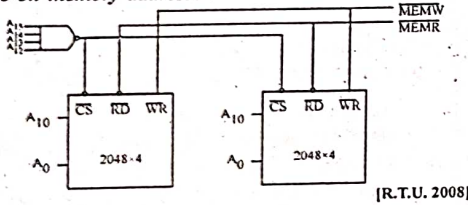
LDAX D

A ← (7870) load 7870 memory location data into A.

HLT

Stop program

Prob.11 *For the figure below, specify the entire memory map and explain the significance of the don't care address line on memory addresses.*



[R.T.U. 2008]

Sol. Memory mapping

A_{15} A_{14} A_{13} A_{12}	A_{11}	A_{10} ..... A_0	Hex. Address	
1 1 1 1	×	0 0 0	F000 or F800 minimum value of the Address	if × 0 → if × 1 →
1 1 1 1	×	1 1 1	FFFF or F7FF maximum value of the Address	if × 1 → if × 0 →

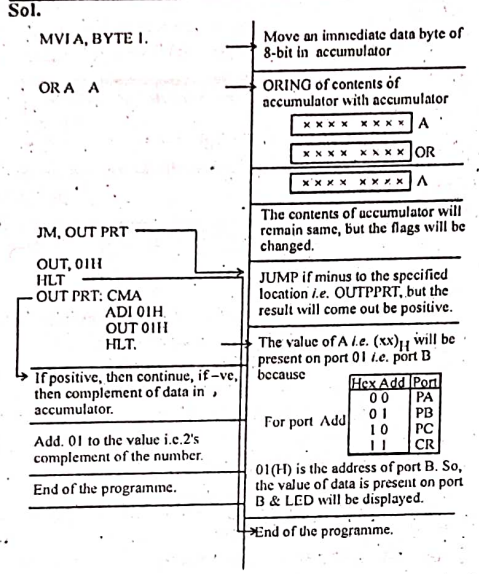
The A_{11} is not used in the NAND gate logic and not in the add range. ∴ A_{11} can be used as (X) don't care, because it is not taking part in the add generator. Therefore, the (X) is used for address variable.

Prob.12 *Explain the function of following program.:*

```
MVI A, Byte 1
ORA A
JM OUT PRT
OUT 01H
HLT
OUTPRT : CMA
         ADI 01H
         OUT 01H
         HLT
```

[R.T.U. 2008]

Sol.



Prob.13 *If the clock frequency is 5MHz, how much time is required to execute an instruction of 18 T-States.*

[R.T.U. 2008]

Sol. Frequency = 5MHz

$$\text{Time} = \frac{1}{5}\times10^{-6} = 0.2\times10^{-6} = 0.2\,\mu\text{sec.}$$

No. of T-States = 18

Time required to execute the instruction $= 0.2\times10^{-6}\times18$

$$= \frac{2}{10}\times10^{-6}\times18 = 3.6\times10^{-6} = 3.6\,\mu\text{sec.}$$

∴ Time = 3.6 μsec.

Prob.14 *Draw the timing diagram of the OUT 80H. Also explain the same.*

[R.T.U. 2008]

Sol. OUT : Output data from accumulator to a port with bit address :

Opcode	Operand	Bytes	T-Cycle	T-States	T-Flags
OUT	8 bit port address	2	3	10	No flag effected

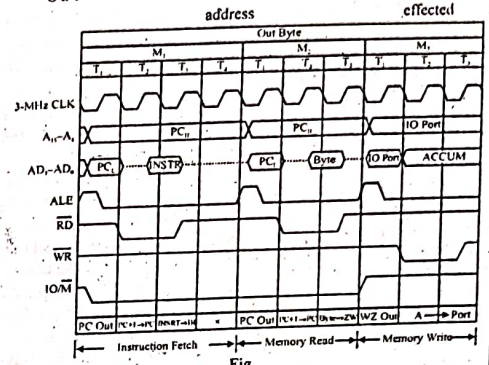


Fig.

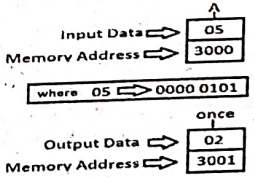
The operand is an 8-bit address, therefore, port addresses can range from 00H to FFH. While executing the instruction, a port address is placed on the low order address bus (A_7-A_0) as well as high order address bus (A_{15}-A_8). Any of the sets to address lines can be decoded to enable the output port.

PART-B

Prob.15 *Write a program in Assembly Language to find number of 1's in the given bit pattern of 8 bits.*

[R.T.U. 2019]

Sol.



Algorithm -

1. Move 00 to register B immediately for count
2. Move 08 to register C immediately for shifting
3. Load the data of memory [3000] into accumulator
4. Rotate 'A' right with carry
5. Jump if no carry to step-7
6. Otherwise increase register B by 1
7. Decrease register C by 1
8. Jump if not zero to step-4
9. Move content of register B into accumulator
10. Store content of accumulator into memory [3001] (number of count)
11. Stop

Program

Memory	Mnemonics	Operands	Comment
2000	MVI	B, 00	[B] <- 00
2002	MVI	C, 08	[C] <- 08
2004	LDA	[3000]	[A] <- [3000]
2007	RAR		rotate 'A' right with carry
2008	JNC	200C	jump if no carry
200B	INR	B	[B] <- [B] + 1
200C	DCR	C	[C] <- [C] - 1
200D	JNZ	2007	jump if not zero
2010	MOV	A, B	[A] <- [B]
2011	STA	[3001]	number of ones
2014	HLT		Stop

Explanation - Registers A, B and C are used for general purpose.

1. **MVI** is used to load an 8-bit given register immediately (2 Byte instruction)
2. **LDA** is used to load accumulator direct using 16-bit address (3 Byte instruction)
3. **MOV** is used to transfer the data from accumulator to register(any) or register(any) to accumulator (1 Byte)
4. **RAR** is used to shift 'A' right with carry (1 Byte instruction)
5. **STA** is used to store data from accumulator into memory direct using 16-bit address (3 Byte instruction)
6. **INR** is used to increase given register by 1 (1 Byte instruction)

7. JNC is used to jump to the given step if their is no carry (3 Byte instruction)
8. JNZ is used to jump to the given step if their is not zero (3 Byte instruction)
9. DCR is used to decrease given register by 1 (1 Byte instruction)
10. HLT is used to halt the program

Prob.16 ***Define direct and indirect addressing modes with appropriate examples.*** **[R.T.U. 2016]**

OR

Describe various addressing modes available in 8085 microprocessor with two example of each. **[R.T.U. 2013]**

OR

Explain various addressing modes with example in 8085 assembly language programming. **[R.T.U. 2013, 10, Raj. Univ. 2001, 2000]**

OR

Explain direct and indirect addressing with suitable examples. **[R.T.U. 2014]**

Sol. Addressing Modes : Each instruction requires certain data on which it has to operate. There are various techniques to specify data for instructions these techniques are called addressing mode. Intel 8085 uses the following addressing modes :

(a) Direct Addressing : In this type of addressing, the address of the operand (data) is given in the instruction itself.

Examples :

(1) STA2400 H : Store the content of the accumulator in the memory location 2400 H.
32, 00 24 : The above instruction in the code form.

In this instruction 2400H is the memory address where data is to be stored. It is given in the instruction itself D. The 2nd and 3rd bytes of the instruction specify the address of the memory location. Here, it is understood that the source of data is accumulator.

(2) IN 02 : Read data from the port C
D B, 02 : Instruction in the code form.

In this instruction 02 is the address of the port C of I/O port from where the data is to be read. Here, it is implied that the destination is the accumulator. The 2nd byte of the instruction specifies the address of the port.

(b) Register Addressing : In register addressing mode, the operands are in the general purpose register. The opcode specifies the address of the registers in addition to the operation to be performed.

Examples :

(1) MOV A, B : Move the content of register B to register A.
78 H : The instruction in the code form.
(2) ADD B : Add the content of register B to the content of register A.
80 H : The instruction in the code form.

In example (1) the opcode for MOV A, B is 78H Besides the operation to be performed, the opcode also specifies the registers, which contain data. The opcode 78H can be written in binary form as 01111000. The first two bytes, *i.e.* 01 are for MOV operation, the next three bits 111 are the binary code for register A and the last three bits 000 are the binary code for register B.

In example (2) the opcode for ADD B is 80H. In this instruction one of the operands is register B (its content is one of the data) which is indicated in the instruction itself. In this type of instruction (arithmetic group), it is understood that the other operand is in the accumulator. The opcode 80H in the binary or is 10000000. The first five bits *i.e.* 10000 specify the operation to be performed, *i.e.* ADD. The last three bits 000 are the binary code for register B for 8085 microprocessor.

(c) Register Indirect Addressing : In this mode of addressing the address of the operand is specified by a register pair.

Examples :

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

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

(2)LXI H, 2500 H : load the HL pair with 2500 H.
ADD M : Add the content of the memory location, whose address is in HL pair (i.e. 2500 H), to the content of the accumulator.
HLT : Halt

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

(d) Immediate Addressing : In immediate addressing mode, the operand is specified with the instruction itself.

Examples :

(1) MVI A, 05 : Move 05 in register A.
3 E, 05 The instruction in the code form.

(2) ADI 06 : Add 06 to the content of the accumulator.
C6, 06 : The instruction in the code form.

In these instructions the 2nd byte specifies data.

LXI H, 2500 H is an example of immediate addressing 2500 is 16-bit data.

Which is given in the instruction itself. It is to be loaded into HL pair.

(e) Implicit Addressing : There are certain instructions, which operate on the content of the accumulator. Such instructions do not require the address of the operand.

Examples : CMA, RAL, RAR etc.

Prob.17 ***Write a short note on rotate instructions of 8085 Microprocessor.*** **[R.T.U. 2016]**

OR

Explain the all type of Rotate-instructions with the help of suitable examples. **[R.T.U. 2013]**

Sol. 8085 Microprocessor has four types of rotational instructions.

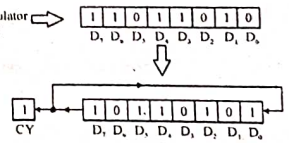
(i) RLC (Rotate Accumulator Left)
(ii) RAL (Rotate Accumulator Left Through Carry)
(iii) RRC (Rotate Accumulator Right)
(iv) RAR (Rotate Accumulator Right Through Carry)

(i) RLC (Rotate Accumulator Left) : Rotate or shift the content of accumulator left by one bit without carry. Seventh bit (D_7) of the accumulator is moved to carry bit (cy) as well as to zero bit (D_0).

$D_{n+1} \leftarrow D_n$, $D_0 \leftarrow D_7$, $CY \leftarrow D_7$

(a) It is 1 byte opcode.
(b) Only carry flag is affected.
(c) Implicit addressing mode (no oprand is there, works on accumulator.)
(d) It has 1 machine cycle and 4-T states.

Example : If accumulator contain 8-bit data 11011010. Than applying RLC instruction on it.



Accumulator after applying RLC Instruction.

Use of RLC

1. To multiply the number by two.
2. To check whether bit has carry or no carry.

(ii) RAL (Rotate Accumulator Left Through Carry): This instruction rotate the contents of accumulator left by one bit with carry.

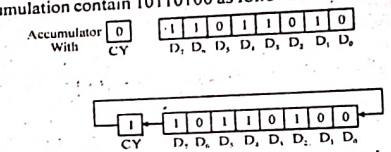
The seventh bit of the accumulator shift to carry bit and carry bit is shift to zero bit of the accumulator.

Syntex :

$D_{n+1} \leftarrow D_n$, $CS \leftarrow D_7$, $D_0 \leftarrow CY$

(a) It is 1 byte opcode.
(b) Only carry flag is modifed.
(c) Implicit addressing mode.
(d) It has same 1 machine cycle and 4-T states.

Example : If accumulator contain 8-bit data 11011010 and carry bit contain 0 than after instruction execution accumulation contain 10110100 as follows.



Accumulator and carry bit after RAL execution.

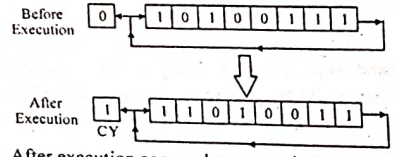
(iii) RRC (Rotate Accumulator Right) : Rotate or shift the contents of accumulator right by one bit without carry. Zero bit of accumulator shift to seventh bit and carry bit.

Syntex :

$D_7 \leftarrow D_0$, $CS \leftarrow D_0$, $D_n \leftarrow D_{n+1}$

(a) It is 1 byte opcode instruction.
(b) Only carry flag is affected.
(c) Implicit address mode.
(d) It has 1 machine cycle and 4-T state.

Example : Accumulator contain A7H and carry flag is reset to 0.



After execution accumulator contain DEH value.

Use

(1) To divide the numbers by two.

(2) To check whether the bit has carry or no carry.

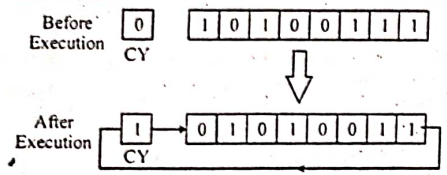
(iv) RAR (Rotate Accumulator Right Through Carry) : Rotae of shift the contents of accumulator right by one bit with carry. Zero bit of Accumulator is shift to carry bit and carry bit is shift to seventh bit.

Syntex :

$D_n \leftarrow D_{n+1}$, $CY \leftarrow D_0$, $D_7 \leftarrow CY$

It has same features like RRC instructions.

Example : If accumulator contain A7H and carry flag contain 0 bit.



Prob.18 *Explain instruction cycle of an instruction MVI A, 05 H using timing diagram.* [R.T.U. 2016]

Sol. Let the actual physical memory location of register A be 1000H and the machine code for the same be 06H. The data is given to be in 05H. Let the actual physical memory location of this data be 1001 H.

Mnemonics	Machine Code	Memory Locations
MVIA, 05H	06H	1000H
	05H	1001H

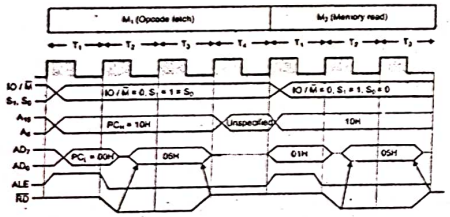


Fig. : Timing diagram for MVI A, 05H

The MVI A, 05H instruction requires 2-machine cycles (M_1 and M_2). M_1 requires 4-states and M_2 requires 3-states, total of 7-states as shown in figure. Status signals IO/M, S_1 and S_0 specifies the 1st machine cycle as the opcode fetch.

In T_1-state, the high order address {10H} is placed on the bus $A_{15} \Leftrightarrow A_8$ and low-order address {00H} on the bus $AD_7 \Leftrightarrow AD_0$ and ALE = 1. In T_2 -state, the RD line goes low and the data 06 H from memory location 1000H are placed on the data bus. The fetch cycle becomes complete in T_3-state. The instruction is decoded in the T_4-state. During T_4-state, the contents of the bus are unknown. With the change in the status signal, IO/ M = 0, S_1 = 1 and S_0 = 0, the 2nd machine cycle is identified as the memory read. The address is 1001H and the data byte [05H] is fetched via the data bus. Both M_1 and M_2 perform memory read operation, but the M_1 is called opcode fetch i.e., the 1st machine cycle of each instruction is identified as the opcode fetch cycle. Execution time for MVI A, 05H i.e., memory read machine cycle and instruction cycle is

Mnemonic	Instruction Byte	Machine Cycle	T-states
MVI A, 05H	Opcode	Opcode Fetch	4
	Immediate Data	Read Immediate Data	3/7

Clock frequency of 8085 = 3.125 MHz
Time (T) for one clock = 1/3.125 MHz = 0.32 μS
Time for Memory Read = 3T = 3*0.320 μS = 0.96 μS
Total Execution time for Instruction = 7T = 7*0.320 μS = 2.24 μS

Prob.19 *Explain LDAX with an example. Write no. of T states and machine cycles involved in it.* [R.T.U. 2014]

Sol. LDAX : Load Accumulator Indirect. This instruction is used to load the contents of a memory location (location indicated by either register pair BC or DE) into the accumulator.

Example: Assume B = 20H, C = 50H and the memory location 2050H = 71H. The instruction 'LDAX B' transfers the contents of 2050 H (i.e., 71H) to the accumulator. The complete description of LDAX is :

Opcode	Operand	Bytes	M-cycles	T-states	Hex code
LDAX	BC	1	2	7	0A
	DE				1A

No flags are affected by this instruction. The timing diagram for LDAX is shown in Fig. (assume that this opcode 'LDAX B' is stored in the memory location 103EH)

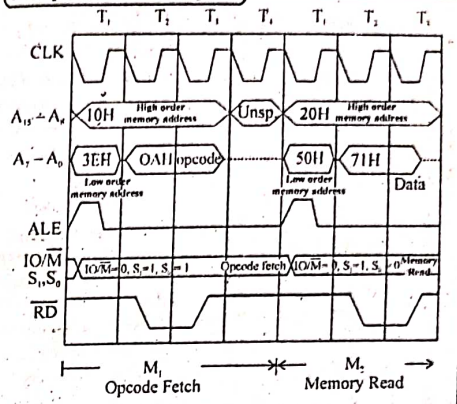


Fig. : Timing diagram of LDAX B

Prob.20 *Write an assembly language delay subroutine to provide a time delay of 0.5ms for an 8085 microprocessor operating at 2 MHz frequency.* [R.T.U. 2012]

Sol. The delay required is 0.5 millisecond hence an 8-bit register of an 8085 can be used to store a count value. The count is decremented by one and the zero flag is verified. If zero flag is set, then the decrement operation is terminated. The delay routine is written as a subroutine, as shown below.

Delay routine

MVI D, N ; Load the count value, N in D-register.

Loop : DCR D ; Decrement to count.

JNZ Loop ; If count is not zero go to Loop.

RET ; If count is zero return to main program.

The following table shows the T-state required for execution of the instructions in the subroutine.

Table

Instruction	T-state required for execution of an instruction	Number of times the instruction is executed	Total T-states
CALL addr 16	18	1	18×1 = 18
MVI D, N	7	1	7 × 1 = 7
DCR D	4	N times	4 × N = 4N
JNZ Loop	10	(N-1) times	10 × (N-1) = 10N-10
	or 7	1	7×1 = 7
RET	10	1	10 × 1 = 10
		Total T-state required for subroutine = 14N + 32	

Calculation to find the count value, N

External clock frequency = 2 MHz

Internal clock frequency = $\frac{\text{External clock}}{2} = \frac{2}{2} = 1\text{MHz}$

Time period of one T state = $\frac{1}{\text{Internal clock frequency}}$

$= \frac{1}{1\times10^6} = 10^{-6}$

Number of T states required for 0.5 ms = $\frac{\text{Required time delay}}{\text{Time for one T-state}}$

$= \frac{0.5\times10^{-3}}{10^{-6}} = 500_{10}$

On equating the total T states required for subroutine and number of T states for the required time delay the count value, N can be calculated.

$\therefore\ 14N + 32 = 500_{10}$

$$N = \frac{500-32}{14} = 33.428_{10} \approx 34_{10} = 22_H$$

$\therefore$ Count value, $N = 22_H$

If the above delay routine is called by a program and executed with the count value of 22_H then the delay produced will be 0.5 millisecond.

Prob.21 *(a) Write a program to transfer sixteen bytes of data stored in memory location at C250H to C25FH to new memory locating starting at C270 H.*

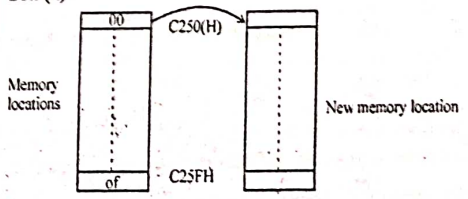
(b) Specify the register contents and the flag status as the following instructions are executed.

A	B	S	Z	CY
XX	XX	X	X	X

SUB A
MOV B, A
DCR B
INR B
SUI 01 H
HLT

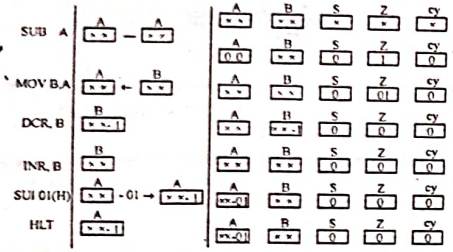
[R.T.U. 2008]

Sol. (a)



```
LXI H, C250H          H-L
LXI D, C270(H)        C250
MVI C, OF (H)         D-E
Loop: MOV A, M.       C270
XCHG                  C
MOV B,M               OF
MOV M, A.             A
XCHG                  XX1
MOV M,B               H-L
INX H.                C270          D-E
INX D.                B             C250
DCR.C.                XX2           D-E
JNZ : Loop            H-L           C270
                      C250
```

Sol. (b)



***Prob.22** Write the assembly Language program for the following problems :*

(a) Addition of the two Hexadecimal numbers 3A968A67 and 9B476C8B [Raj. Univ. 2004]

(b) Exchange of the 10 data contained from the memory location 2450 onwards (1st Block) with memory location 2480 onwards (2nd Block). [Raj. Univ. 2007, 2001]

Sol. (a) Addition of two 32–bit hexadecimal numbers 3A968A67 and 9B476C8B

Mnemonics		Comments
MVI	A, 67 H	Move 67 H *i.e.* 8LSB of first number in the accumulator.
MVI	L, 8BH	Move 8B *i.e.* LSB of second number in the L register.
ADD	L	L register A ← A + L
MOV	CA	C ← A
MVI	A, 8A H	Move next 8 LSB of first number in the accumulator.
MVI	I, 6C H	Move next 8 LSB of second number in the accumulator.
ADD	L	A ← A + L (carry from previous sum)
MOV	B, A	B ← A
MVI	A 96 H	Move 96 H to acc.
MVI	L, 47 H	Move 47 H to reg.
ADC	L	A ← A + L + carry of previous sum
MOV	E, A	E ← A
MVI	A, 3AH	Move 3A to accumulator
MVI	L, 9 BH	Move 9 B to reg L.
ADC	L	A ← A + L + carry
MOV	D, A	D ← A
ACI	00H	A ← 00H + carry + A'
HLT		Stop

Sol. (b)

Mnemonics		Comments
MVI	C, A'H	Move counter to reg C.
LXI	H, 2450 H	Starting address of the 1st block is stored in the HL pair.
LXI	D, 2480 H	Starting address of the second block is stored in the DE pair
Label MOV		A, M Move the data stored in the address, stored in HL pair to accumulator
MOV	B, A	B ← A
LDA	X, D	Load the accumulator with the data that is stored in the address, stored in DE pair.
Move	M, A	Move the data stored in accumulator and the address stored in HL pair.
Move	A, B	A ← B
STAX	D	Stored the data in the accumulator in the address that is stored in the DE pair
INX	H	Increment HL Pair
INX	D	Increment DE pair
DC	RC	Decrement counter
JNZ	Label	Check if the counter is equal to zero if not, then jump to label.
HLT		Stop.

PART-C

***Prob.23** Explain the instruction format of 8085. Draw the timing diagram of instruction MOV A, M and explain it.* [R.T.U. 2017]

Sol. Instruction Format of 8085 : *Refer to Prob.1.*

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

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

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

1. One-Byte Instructions : A 1-byte instruction includes the opcode and operand in the same byte. Operand(s) are internal register and are coded into the instruction

Table 1 : Example for 1 Byte Instruction

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

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

MOV rd, rs

rd ß rs copies contents of rs into rd.

Coded as 01 dddsss

where ddd is a code for one of the 7 general registers which is the destination of the data, sss is the code of the source register.

Example : MOV A,B

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

ADD r

A ß A + r

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

Table 2 : Example for 2 Byte Instruction

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

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

MVI r,data

r ß data

Example: MVI A,30H coded as 3EH 30H as two contiguous bytes.

This is an example of immediate addressing.

ADI data

A ß A + data

OUT port

0011 1110

DATA

Where port is an 8-bit device address.(Port) ß A.

Since the byte is not the data but points directly to where it is located this is called direct addressing.

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

opcode + data byte + data byte

Table 3 : Example for 3 Byte Instruction

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

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

Three byte instructions - opcode + data byte + data byte

LXI rp, data16

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

rp ß data16

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

LDA addr

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

Example: LDA 2134H coded as 3AH 34H 21H. This is also an example of direct addressing.

Timing Diagram for MOV A,B

The instruction MOV A,B is a 1-byte instruction. Microprocessor takes only one machine cycle (op-code fetch) to complete instruction. Hence, hex code for MOV A,B is passed to the microprocessor.

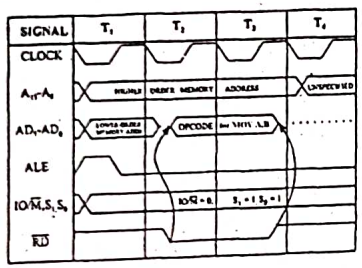
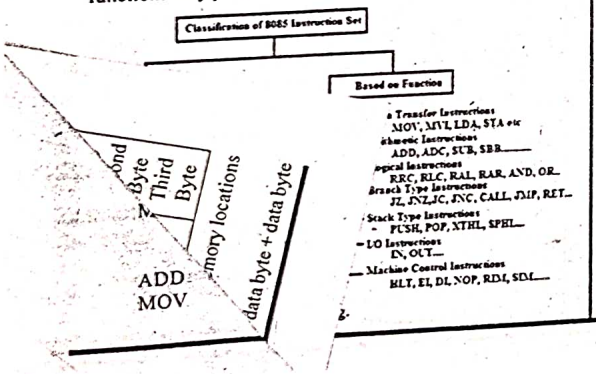


Fig.

Prob.24 *Classify the instructions set of 8085 and explain them.* [R.T.U. 2017]

Sol. Instruction Set of 8085 : 8085 Instructions can be classified based on the size they occupy in memory or by the functions they perform.



Classification of Instruction Set of 8085

Instructions Based on Size : Based on the size, tl instructions can be classified are as follows:

1. **One byte Instructions:** These instructions are of o byte in size and hence occupy one memory location RAM. Examples are CMA, RLC, RRC, RAL, RA STC, CMC etc. These instructions do not require a operand to be specified with the instructions, inste the operand is implied in the instructions.
2. **Two Byte Instruction:** These instructions of tv byte(16-bits) in size and hence will occupy two memc locations in RAM. Examples of such instructions a MVI C,0A;
3. **Three Byte Instructions:** These are of three by in size and hence occupy three locations memory(RAM). Examples of such instructions a CALL, JMP etc.

Instruction Set Based on Function

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

1. Data Transfer Group : The data transfer instructic move data between registers or between memory a registers.

MOV	Move
MVI	Move Immediate
LDA	Load Accumulator Directly from Memor
STA	Store Accumulator Directly in Memory
LHLD	Load H & L Registers Directly frc Memory
SHLD	Store H & L Registers Directly in Memc

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

LXI	Load Register Pair with Immediate data
LDAX	Load Accumulator from Address in Regis Pair
STAX	Store Accumulator in Address in Regis Pair
XCHG	Exchange H & L with D & E
XTHL	Exchange Top of Stack with H & L

2. Arithmetic Group : The arithmetic instructions a subtract, increment, or decrement data in registers or memc

ADD	Add to Accumulator
ADI	Add Immediate Data to Accumulator
ADC	Add to Accumulator Using Carry Flag
ACI	Add immediate data to Accumulator Usi Carry
SUB	Subtract from Accumulator
SUI	Subtract Immediate Data from Accumula

SBB	Subtract from Accumulator Using Borrow (Carry) Flag
SBI	Subtract Immediate from Accumulator Using Borrow (Carry) Flag
INR	Increment Specified Byte by One
DCR	Decrement Specified Byte by One
INX	Increment Register Pair by One
DCX	Decrement Register Pair by One
DAD	Double Register Add; Add Content of Register Pair to H & L Register Pair

3. Logical Group : This group performs logical (Boolean) operations on data in registers and memory and on condition flags. The logical AND, OR, and Exclusive OR instructions enable you to set specific bits in the accumulator ON or OFF.

ANA	Logical AND with Accumulator
ANI	Logical AND with Accumulator Using Immediate Data
ORA	Logical OR with Accumulator
OR	Logical OR with Accumulator Using Immediate Data
XRA	Exclusive Logical OR with Accumulator
XRI	Exclusive OR Using Immediate Data
	The Compare instructions compare the content of an 8-bit value with the contents of the accumulator;
CMP	Compare
CPI	Compare Using Immediate Data

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

RLC	Rotate Accumulator Left
RRC	Rotate Accumulator Right
RAL	Rotate Left Through Carry
RAR	Rotate Right Through Carry

Complement and carry flag instructions:

CMA	Complement Accumulator
CMC	Complement Carry Flag
STC	Set Carry Flag

4. Branch Group : The branching instructions alter normal sequential program flow, either unconditionally or conditionally. The unconditional branching instructions are as follows:

JMP	Jump
CALL	Call
RET	Return

Conditional branching instructions examine the status of one of four condition flags to determine whether the specified branch is to be executed. The conditions that may be specified are as follows:

NZ	Not Zero (Z = 0)
Z	Zero (Z = 1)
NC	No Carry (C = 0)
C	Carry (C = 1)
PO	Parity Odd (P = 0)
PE	Parity Even (P = 1)
P	Plus (S = 0)
M	Minus (S = 1)

Thus, the conditional branching instructions are specified as follows:

Jumps	Calls	Returns	
INC	CNC	RNC	(No Carry)
JNZ	CNZ	RNZ	(Not Zero)
JM	CM	RM	(Minus)
JP0	CPO	RPO	(Parity Odd)
JM	CM	RM	(Minus)
JPE	CPE	RPE	(Parity Even)
JP0	CPO	RPO	(Parity Odd)

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

PCHL	Move H & L to Program Counter
RST	Special Restart Instruction Used with Interrupts

5. Stack Instructions : The following instructions affect the Stack and/or Stack Pointer

PUSH	Push Two bytes of Data onto the Stack
POP	Pop Two Bytes of Data off the Stack
XTHL	Exchange Top of Stack with H & L
SPHL	Move content of H & L to Stack Pointer

6. I/O Instructions

IN	Initiate Input Operation
OUT	Initiate Output Operation

7. Machine Control Instructions

EI	Enable Interrupt System
DI	Disable Interrupt System
HLT	Halt
NOP	No Operation

Prob.25 *Define instruction cycle, machine cycle and T-state.* [R.T.U. 2016]

OR

Define the following terms :

(i) Instruction Cycle

(ii) Machine Cycle

(iii) T-State. [Raj. Univ. 2007, 2001, 2000]

Sol. Instruction Cycle : An instruction cycle is defined as the sequence of operations that are required by the central processing unit to fetch an instruction and data from the memory and to execute it. The instruction cycle consists of a fetch cycle and a execute cycle. In a fetch cycle the CPU fetches the opcode (the machine code of an instruction) from the memory. The sequence of operations which are required to fetch an opcode from the memory constitute a fetch cycle.

The necessary steps which are required to get data from the memory and to perform the operation specified by an instruction constitute an execute cycle.

Instruction Cycle = Fetch Cycle + Execute Cycle.

Fetch Operation : The instruction may be one byte, two byte or three byte long. The first byte of any type of instruction is the opcode (machine code of the instruction). The second or third byte of an instruction may be the data or the address.

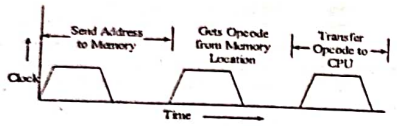


Fig. 1

The program counter keeps the address of the next instruction to be executed. In the beginning of a fetch operation, the contents of the program counter which is the address of the memory location where opcode of the instruction is stored is sent to the memory. The memory places opcode of the instruction on the data bus so as to transfer to the CPU. This is called the fetch operation. To fetch an opcode of an instruction, three clock cycles are required. A slow memory may take more time. In the case of slow memory, the CPU has to wait till the memory send the opcode. The clock cycle for which the CPU waits is called the wait cycle. Fetch cycle is shown in fig. 1.

Execute Operation : After the opcode of an instruction fetched from the memory, the execution begins. The opcode fetched goes to the data register and then it goes to instruction register. From the instruction register it goes to instruction decoder which decodes the instruction and after the decoding of the instruction, the task specified in the instruction is carried out. This is called the execute operation. In one byte instruction, the operand is a general purpose register and opcode and operand are specified by one byte. The execution of one by one instruction is immediately performed.

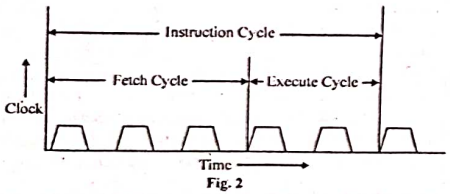


Fig. 2

The time taken in execution is one clock cycle. In the two byte or three byte instruction which contains data or address which is still laying in the memory, the CPU performs the read operation to get the desired data. After receiving the data, it performs the execute operation. Read operation is similar to the fetch operation. In the read operation, the quantity received from the memory, are data or operand address where as in the case of fetch operation, it is an opcode. In some instructions, the write operation is performed. In the write operation, data are sent from the CPU to the memory or to an output device. So execute operation may also consists of one or more read or write cycles. A fig. 2 shows the instruction cycle, which contains fetch cycle and execute cycle.

In the fig. 3 M_1 and M_2 are the machine cycles, T_1, T_2 etc. are T-states. In this diagram instruction cycle consists of two machine cycles.

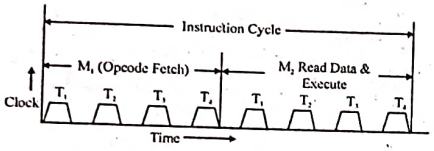


Fig. 3

Machine cycle : Machine cycle is defined as time required to complete one operation to accessing memory, I/O or acknowledge an external request. This cycle may consist three to six T – states.

T-state is defined as one subdivision of operation performed in one clock period, these subdivision are internal state synchronization with system clock and each T – state is precisely equal to one clock period. The term T – state clock period are often used synchronously.

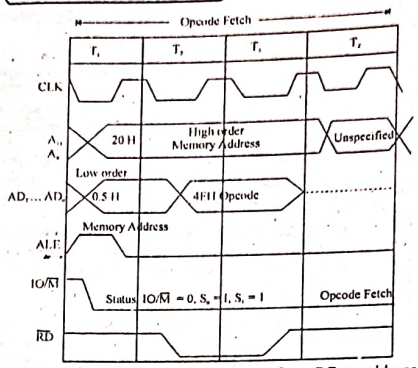
There are following types of machine cycles related with 8085 MPU.

1. Opcode fetch
2. Memory read
3. Memory write
4. I/O read
5. I/O write
6. Interrupt acknowledge
7. Halt
8. Hold
9. Reset

Opcode Fetch Machine

To illustrate operation opcode fetch, we are using data flow to instruction code MOV C, A (4 F H) stored at location 2005 H is fetched.

To fetch the bytes (4F H), the MPU needs to identify the memory location 2005 H and enable the data flow from memory. This is called fetch cycle. Fig. shows that data flow is transferred from memory to MPU. It shows five groups of signals in relation to system clock.



Step-1 : The MPU places 16 bit address from PC on address bus in figure at T1 the higher order memory address 20H is placed on address line $A_{15} - A_8$ and lower order address 05H is placed on bus $A_7 - A_0$ and ALU signal goes high similarly status signal IO/$\overline{M}$ goes low, indicating that this is memory related operation.

Step-2 : The control unit sends control signals RD to enable memory chip.

The control signal RD is sent out during clock period T_2 thus enabling memory chip, RD signal actives two clock period.

Step-3 : The bytes from memory location is placed on data bus.

When memory is enabled the instruction byte (4F– H) is placed on bus $AD_7 - AD_0$ and transferred to microprocessor. RD signal causes 4FH to be placed on bus $AD_7 - AD_0$ (shown by arrow) and when RD goes high, causes the bus go to high impedance stage.

Step-4 : The bytes placed in instruction decoder of MPU and result is carried out according to instructions.

The machine code is decoded by instruction decoder and the contents of accumulator are copied into register C this state performed during T_4 states(clock period).

Prob.26 *(a) What is assembly language? Also explain its advantages and disadvantages.*

(b) Draw and explain the timing diagram for the instruction MVIM, 7AH. [R.T.U. 2015]

Sol.(a) An assembly language is a low level programming language designed for a specific family of processors to represent the obscure and complex machine language instructions in symbolic and more understandable form. The processor only understand machine language instructions, so the assembly language code is translated to machine language code using a program called 'Assembler'.

Advantages of Assembly Language :

- It is easy to write, modify and debug.
- It has the same efficiency as machine level language but is more readable.
- The syntax and symbolic programming in assembly language is easier to understand and saves time and effort of programmer.
- It is faster than high level programming languages, requires less memory and execution time.
- It allows complex hardware specific jobs in an easier way.
- It is the best suited for writing input or output drivers of an operating system.

Disadvantages of Assembly Language :

- It is machine and processor dependent. A program written for one computer might not run on computers with different hardware configuration.
- The programmer has to know the hardware details and addressing details.
- The code written in assembly language is longer than that written in a high level programming language,

Example: If we need to add two integers 'a' and 'b', the code in C and x86 assembly language is given below :

```
C:   c = a + b;
x86: movl [a], %eax
     addl [b], %eax
     movl %eax, [c]
```

Sol.(b) In the Microprocessor 8085, MVI means Move Immediate 8-bit. The syntax is MVI <Register>, <data>

On execution of this instruction, the <data> which is of 8-bits is stored in the <Register>.

Let the actual physical memory location of register M be 1000H and the machine code for the same be 06H. The data is given to be in 7AH. Let the actual physical memory location of this data be 1001 H.

The MVI M, 7AH instruction requires 2-machine cycles (M_1 and M_2). M_1 requires 4-states and M_2 requires 3-states, total of 7-states as shown in fig. Status signals IO/M, S_1 and S_0 specifies the 1st machine cycle as the op-code fetch. In T_1-state, the high order address {10H} is placed on the bus $A_{15} - A_8$ and low-order address {00H) on the bus $AD_7 - AD_0$

and ALE = 1. In T_2-state, the RD line goes low, and the data 06H from memory location 1000H are placed on the data bus. The fetch cycle becomes complete in T_3-state. The instruction is decoded in the T_4-state. During T_4-state, the contents of the bus are unknown. With the change in the status signal, IO/ M = 0, S_1 = 1 and S_0 = 0, the 2nd machine cycle is identified as the memory read. The address is 1001H and the data byte [7AH] is fetched via the data bus. Both M_1 and M_2 perform memory read operation, but the M_1 is called op-code fetch i.e., the 1st machine cycle of each instruction is identified as the opcode fetch cycle. Execution time for MVI M, 7AH i.e., memory read machine cycle and instruction cycle is :

Clock frequency of 8085 = 3.125 MHz
Time (T) for one clock = 1/3.125 MHz = 0.32 μs
Time for Memory Read = 3T = 3*0.320 μs = 0.96 μs
Total Execution time for Instruction = 7T = 7*0.320 μs = 2.24 μs

The Timing Diagram is

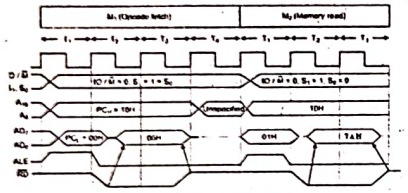


Fig. : Timing Diagram

Prob.27(a) ***What are peripheral mapped I/O memory and memory mapped I/O? Differentiate b/w them.*** [R.T.U. 2015]

OR

Explain memory mapped and I/O Mapped Techniques. [R.T.U. 2010]

(b) Write the uses of buffers, decoders, encoders and D-flip/flop in Microprocessor? [R.T.U. 2015]

Sol.(a) I/O devices (peripherals) like keyboard, switches, converters, CRT, printers, LEDs etc. are communicated with microprocessor using interfacing methods. In the 8085 microprocessor based system, I/O devices can be interfaced using both techniques : **memory-mapped I/O and peripheral-mapped I/O.** The process of data transfer is identical in both.

Peripheral-Mapped I/O

In this method, mostly lower address lines ($A_0 - A_7$) are used to get an address of I/O ports and this method is identified by 8-bit address. Following steps are used in this method :

1. Address lines ($A_0 - A_7$ or $A_8 - A_{15}$) are decoded generate a unique pulse corresponding to the I/ address on the bus and this is called device or I/ address pulse.
2. Device address pulse with the control signal ($\overline{IO}$ or $\overline{IOW}$) generates a device select (I/O Selec pulse that is generated only when both signals a asserted.
3. Use the I/O pulse to activate the interfacing devi or I/O port.

Memory-Mapped I/O

I/O devices are treated as memory locations. Ea I/O device will have 16-bit address, ranges from 0000H FFFFH. $\overline{MEMR}$ and $\overline{MEMW}$ are the control signals used this technique. The microprocessor communicates with I/O device as if it were one of the memory locations. Ste involved are :

1. Decode the address bus to generate the devi address pulse.
2. AND the control signal with the device addre pulse to generate the device select pulse.
3. Use the device select pulse to enable the I/O por 16-bit instructions like LDA, STA are used here.

Comparison of Memory-Mapped I/O and Peripher Mapped I/O

Characteristics	Memory-Mapped I/O	Peripheral-Mapped I/O
1. Device address	16-bit	8-bit
2. Control signals for I/O	MEMR/MEMW	IOR/IOW
3. Instructions available	STA, LDA, LDAX, STAX, MOV M, R, ADD M, SUB M, ANA M etc	IN and OUT
4. Data transfer	Between any register and I/O	Only between I/O and the accumulator
5. Maximum number of I/Os possible	The memory map (64K) is shared between I/Os and system memory	The I/O map is independent of the memory map. 256 input and output devices are connected.
6. Execution speed	13 T-states (STA, LDA) 7 T-states (MOV M,R)	10 T-states
7. Hardware requirements	More hardware is needed to decode 16-bit address	Less hardware is needed to decode 8 bit address
8. Other Features	Arithmetic or logical operations can be directly performed with I/O devices.	Not available

Sol.(b) Buffer : A data buffer (or just buffer) is a region a physical memory storage used to temporarily store da while it is being moved from one place to another.

Buffers can increase application performance allowing synchronous operations such as file reads or writ to complete quickly instead of blocking while waiting f hardware interrupts to access a physical disk subsyste instead, an operating system can immediately return successful result from an API call, allowing an application continue processing while the kernel completes the di operation in the background. Further benefits can be achiev if the application is reading or writing small blocks of da that do not correspond to the block size of the disk subsyste

allowing a buffer to be used to aggregate many smaller read or write operations into block sizes that are more efficient for the disk subsystem, or in the case of a read, sometimes to completely avoid having to physically access a disk.

Encoders and Decoders : In digital domain, for ease of transmission of data, the data is often encrypted or placed within codes and then this secured code is transmitted. At the receiver, the coded data is decrypted or gathered from the code and is processed to be displayed or given to the load accordingly.

This task of encrypting the data and decrypting the data is done by Encoders and Decoders.

Encoders : Encoders are digital ICs used for encoding. By encoding, we mean generating a digital binary code for every input. An Encoder IC generally consists of an Enable pin which is usually set high to indicate the working. It consists of 2^n input lines and n output lines with each input line being represented by a code of zeros and ones which is reflected at the output lines.

Decoders : Decoders are digital ICs which are used for decoding. In other words the decoders decrypt or obtain the actual data from the received code, i.e. convert the binary input at its input to a form, which is reflected at its output. It consists of n input lines and 2^n output lines. A decoder can be used to obtain the required data from the code or can also be used for obtaining the parallel data from the serial data received.

D Flip-Flops : They provide a separation of state (before and after the clock signal) and therefore help control things that might otherwise run away if left to propagate on their own.

A D flip-flop allows you to know in your design exactly when you accepted data and froze it. Because of that property, you can then design orderly transitions from state to state, which is very useful in microprocessors which are state machines. State machines proceed in an orderly way from condition to condition, unlike analog systems which constantly change their state. The property also allows you make sure operations are synchronized across many devices.

Prob.28(a) ***Explain the difference between 1 byte, 2 byte and 3 byte instructions. Quote suitable examples.***

(b) Explain STA with an example. Write the no. of T states and machine cycles involved in it. [R.T.U. 2014]

Sol. (a) 1 byte, 2 byte and 3 byte instructions are respectively 1,2 and 3 bytes long, i.e., they require 1,2 and 3 bytes respectively for storage in memory. Straight forward instructions like CMA (complement the contents of the accumulator) are 1 byte long. Instructions that require internal register contents and not specific data are 1 byte long. eg. : ADC (Add register contents to accumulator with carry).

Instructions that require values of data may be 2 bytes long. eg. : ADI (Add an immediate value to accumulator content). In this case, the immediate value that is to be added is the 2nd byte of the instruction. CALL is a 3 byte instruction as it contains a 16-bit address (2 bytes) where the program has to jump to.

More examples are given below :

1 byte : CMC (complement carry),
CMP (compare with accumulator)
DCR (decrement source by 1),
HLT (halt and enter wait state).

2 bytes : IN (input data to accumulator from an 8-bit port address),
MVI (move immediate 8-bit),
CPI (compare immediate with accumulator).

3 bytes : CALL (unconditional subroutine call),
JMP (jump unconditionally),
LXI (load register pair immediate).

Note : The designer of the microprocessor would want all instructions to be as small as possible to save program memory.

Sol. (b) STA : Store Accumulator Direct. This instruction is used to load the contents of the accumulator into the memory location specified by the second and third bytes of the instruction itself. This is a 3 byte instruction. The second byte gives the lower order memory address and the third byte gives the higher order address. Example - Assume that the accumulator contains 6CH, which is to be written at the memory location 31C7H. Then, the complete description of STA is :

Opcode	Operand	Bytes	M-cycles	T-states	Hex code
STA	16 bit	3	4	13	32C731

No flags are affected by this instruction. The timing diagram for STA is shown in Fig. (assume that the opcode STA (32C731H) is stored in the three consecutive memory locations 4700H, 4701H and 4702H)

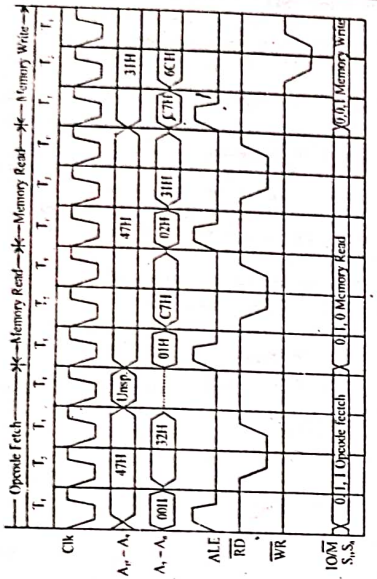


Fig.

Prob.29 ***(a) Draw and explain the timing diagram of fetch operation and memory read operation.***

(b) Explain the following 8085 instructions giving suitable examples. Also indicate the flags affected by these if any :

(i) ACI (ii) DAD (iii) CALL (iv) POP **[R.T.U. 2012]**

Sol. (a) Timing Diagram of Fetch Operation

In a fetch cycle the microprocessor fetches the opcode of an instruction from the memory. Fig. (1) shows the timing diagram for an opcode fetch cycle T_1, T_2, T_3 and T_4 are consecutive 4 clock cycles. The microprocessor issues a low IO/$\overline{M}$ signal to indicate that it wants to make communication with the memory. High S_0 and S_1 signals to indicate that it is going to perform fetch operation.

During the first clock cycle, T_1 the microprocessor sends out the address of the memory location. The 16 bit memory address is sent through the address bus ($A_8 - A_{15}$) and address/data bus ($AD_0 - AD_7$). The MSBs of the memory address are sent over the A bus, and 8 LSBs of the memory address over AD bus.

The AD bus is needed to transfer data durin subsequent clock cycles, it is used in time multiplexed mod Therefore it has to be made available to carry data during T and T_3. To accomplish this the microprocessor sends a address latch enable signal ALE to latch the 8LSBs of memo address either in memory or an external latch.

16 bit memory address is needed by the memory to obtai the opcode from the given memory address.

During T_2 AD bus becomes ready to carry data. In T the microprocessor makes $\overline{RD}$ low. Now memory gets th opcode from the specified memory location and places it o the data bus.

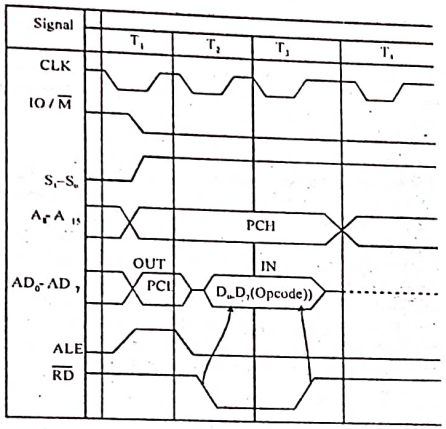


Fig. 1: Timing Diagram for Opcode Fetch Operation

During T_3, the opcode is placed in the instruction register IR which is within the microprocessor. The memory is disabled when $\overline{RD}$ goes high during T_3. The fetch cycle is completed by T_3. The opcode is decoded in T_4. The microprocessor examines READY signal in T_2. If it is high, the microprocessor enters into T_3 state. If it is low the microprocessor inserts a wait state in between T_2 and T_3.

Example : 1 byte instruction are MOV, SUB, ADD etc. As the operand are in the general purpose register, the decoding of the operation code and its execution takes one clock cycle T_4. The 1st machine cycle (M_1) is for fetching the opcode from the memory.

Timing Diagram of Memory Read Operations : It consists following operation :

(i) To fetch the operand of an instruction if it is a multi byte illustration.

(ii) To read data from the memory.

(iii) In memory read the address is given by instruction.

All of these operations require 3-T state *i.e.* T_1 to T_3. Also known as Operand Fetch Cycle.

Operations of operand fetch of memory read cycles:

Step 1 (State T_1) : In T_1 state, 8085 sends appropriate status signals IO/$\overline{M}$ S_0 and S_1.

IO/$\overline{M}$ = 0, S_0 = 0, S_1 = 1 (Memory read).

The 16-bit address is transferred on $A_8 - A_{15}$ and $AD_0 - AD_7$. The address is depended on the operation.

In operand fetch cycle, address is given by program counter and then PC is incremented by 1. Address may be from SP or general purpose register pair. ALE is generated by 8085 to indicate the availability of $A_0 - A_7$ on $AD_0 - AD_7$. The ALE is used to latch A_0 to A_7.

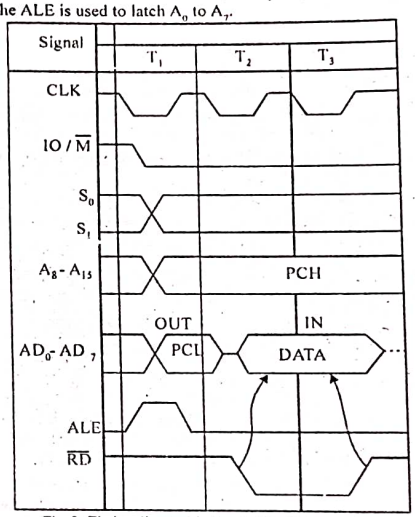


Fig. 2: Timing diagram for Memory Read Operation.

Step 2 (State T_2) : In T_2 state, the line $AD_0 - AD_7$ will be used as $D_0 - D_7$ to transfer data. The control signal $\overline{RD}$ is made low which will enable the selected memory. The memory will send data on data bus.

Step 3 (State T_3) : In T_3 state, data from memory is transferred to 8085. It accepts this data and transfers on internal data bus and $\overline{RD}$ is made high.

Sol. (b) An instruction is a command to the microprocessor to perform a given task on specified data.

The 8085 instruction set is classified into following 3 groups according to word size or byte size :

1. 1-byte instructions
2. 2-byte instructions
3. 3-byte instructions

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

For Example

Task	Opcode	Operand	Binary Code	Hex Code
Copy the content of the accumulator in register C	MOV	C, A	01001111	4F H
Add the contents of register B to the contents of the accumulator.	ADD	B	10000000	80 H
Invert each bit in the accumulator.	CMA		00101111	2F H

2. 2-byte instruction : In a 2-byte instruction, the 1st byte specifies the operation code and the 2nd byte specifies the operand.

For Example

Task	Opcode	Operand	Binary Code	Hex Code
Load an 8-bit data byte in the accumulator	MVI	A, 32 H	00111110 00110010	3F; 1st byte 32;2nd byte
Load an 8 bit data byte in register B	MVI	B, F2 H	00000110 11110010	06; 1st byte F2; 2nd byte

3. 3-byte instruction : In this instruction, the 1st byte specifies the opcode and following 2 bytes specify the 16 bit address.

For Example

Task	Opcode	Operand	Binary Code	Hex Code
Load Content of memory 2050H into A	LDA	2050H	00111010 01010000 00100000	3A; 1st byte 50;2nd byte 20;3rd byte
Transfer the program sequence to memory location 2085H	JMP	2085H	11000011 10000101 00100000	C3; 1st byte 85;2nd byte 20;3rd byte

(i) ACI : Add immediate to accumulator with carry.

Syntax : ACI 8 bit data

The 8 bit data (operand) and carry flag are added to the content of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the addition.

Ex : ACI 45 H

(ii) DAD : Add register pair to H and L registers.

Syntax : DAD register - pair

The 16-bit contents of the specified register pair are added the contents of HL register and sum is stored in the HL register. If the result is larger than 16-bits, the CY flag is set.

No other flags are affected.

Ex : DAD H

(iii) CALL : Unconditional Subroutine Call.

Syntax : CALL 16 bit address

The program sequence is transferred to memory location specified the 16-bit address given in operand. Before the transfer, the address of next instruction after CALL is pushed onto stack.

Ex : CALL 2034 H or CALL xyz.

(iv) POP : POP off stack to register pair.

Syntax : POP Reg pair

The contents of the memory location pointed out by the stack pointer register are copied to the low-order register (C, E, L status flag) of the operand. The stack pointer is incremented by 1 and the contents of that memory location are copied to high-order register (B, D, H, A) of the operand.

Ex : POP H or POP A

Prob.30 ***(a) Explain the following instructions indicating their addressing modes, flags affected, number and names of machine cycles on the execution of each : XRA , CALL, XTHL, LHLD.***

(b) Write a program to count number of 1's and 0 bits in a register. Assume data is in B register and store number of 1 in H register, 0 in L register. [R.T.U. 2011]

Sol.(a) XRA

XRA r : It is a mnemonic for Exclusive OR register. It means logical XORing the contents of accumulator with the contents of register mentioned and store the result back into the accumulator. It is a single byte instruction. It takes one machine cycle of four states. It is register addressing mode. The CY and AC is cleared. Other flags are affected. The opcode is :

10101 SSS

Macro RTL : (A) ← (A) ⊕ (r)

Micro RTL : XRA r

OPFMC : IO/ $\overline{M}$ = 0, S_1 = 1, S_0 = 1

T_1 : $AD_0 - AD_7 \leftarrow PC_0 - PC_7$; $A_8 - A_{15} \leftarrow PC_8 - PC_{15}$; ALE

T_2 : PC = PC + 1; $\overline{RD}$ = 0; BDB ← M (AB)

T_3 : $\overline{RD}$ = 1; IR ← BDB

T4 : (A) ← ⊕ (r)

XRA M : It is a mnemonic for Ex-OR contents of memory with the accumulator. It means logical Ex-OR the content of accumulator with the contents of memory whose address is available in (H, L) register pair and store the result into the accumulator. It is a single byte instruction. It takes two machine cycles of 7 states. It is register indirect addressing mode. The CY and AC flags are cleared. All other flags are affected. The opcode is :

10101110

Macro RTL : (A) ← (A) ⊕ M (H, L)

Micro RTL : XRA M

OPFMC : IO/ $\overline{M}$ = 0, S_1 = 1, S_0 = 1

T_1 : $AD_0 - AD_7 \leftarrow PC_0 - PC_7$; $A_8 - A_{15} \leftarrow PC_8 - PC_{15}$; ALE

T_2 : PC = PC + 1; $\overline{RD}$ = 0; BDB ← M (AB)

T_3 : $\overline{RD}$ = 1; IR ← BDB

T_4 : AND [(A) ⊕ M (H, L)] FF is set

MRDMC : IO/ $\overline{M}$ = 0, S_1 = 1, S_0 = 1

T_1 : $AD_0 - AD_7 \leftarrow$ L; $A_8 - A_{15} \leftarrow$ H; ALE

T_2 : $\overline{RD}$ = 0; BDB ← M (AB)

T_3 : $\overline{RD}$ = 1; (A) ← (A) ⊕ M (H; L)

CALL : CALL addr

Call unconditionally to the subroutine outside the main program specified by the address in the operand of the instruction

(i) [[SP – 1] – 1] ← [PCH], [[SP – 2]–2] ← [PCL], [SP] ← [SP]–2 and [PC] ← addr

(ii) 3 bytes
- 1st byte (opcode)
- 2nd byte (LOA)
- 3rd byte (HOA)

(iii) No flags (no operation performed only sequence is changed).

(iv) Immediate Addressing Mode (immediately jumping).

(v) 5 machine cycles
- opcode fetch – 6 T–states
- memory read – 3 T–states
- memory read – 3 T–states
- memory write – 3 T–states
- memory write – 3 T–states

(vi) 18 T–states
- 6 T–states opcode fetch.
- Memory read lower order address 3 T–states (8–LSB).
- Read higher order address 3 T–states (8–MSB).
- Memory write contents of stack pointer register [SP – 1] are placed on the address bus. The higher order byte of the (PCH) is placed on the data bus and stored in the location [SP–1]. At this instant stack pointer register becomes [SP–2]. (3 T–states).
- Memory write the contents of stack pointer register [SP–2] on the address bus. The lower order byte of the program counter (PCL) is placed on the data bus and stored at location [SP–2], (3 T–states).

XTHL

This instruction exchanges the contents of register L with the data of top of the stack. Also the contents of the register H are unchanged. With the data below the stack top.

(i) [L] ↔ [SP], [H] ↔ [SP+1]

(ii) 1 byte – opcode

(iii) No flags

(iv) Register Indirect Addressing Mode

(v) 5 machine cycles
- opcode fetch – 4 T–states
- memory read – 3 T–states
- memory read – 3 T–states
- memory read – 3 T–states
- memory read – 3 T–states

LHLD Address

Load register pair HL directly with the contents of two consecutive memory locations

(address) and (address+1)

(i) [L] ← [addr], [H] ← [addr+1]

(ii) 3 bytes – 1st byte is the opcode
- 2nd byte is the lower order address
- 3rd byte is the higher order address

(iii) No flags are affected as data transfer effects no flags

(iv) Direct Addressing Mode (address directly defined)

(v) 5 machine cycles
- opcode fetch – 4 T–states
- memory read – 3 T–states
- memory read – 3 T–states
- memory read – 3 T–states
- memory read – 3 T–states

(vi) 16 T–states
- 4 T–states – opcode fetch
- 3 T–states –(LOB) 8–bit
- 3 T–states – (HOB) 8–bit
- 3 T–states – read data at address in L
- 3 T–states – read data at (address + 1) in H

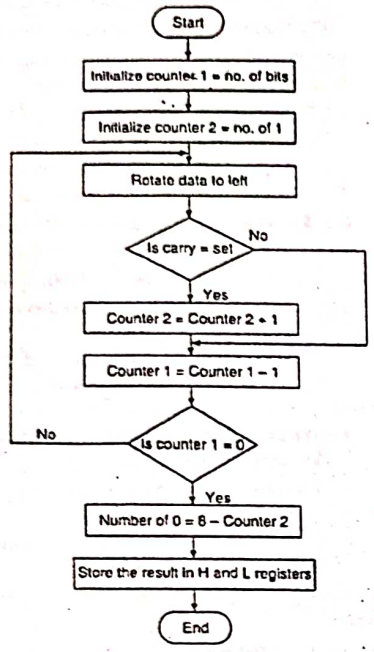
Sol.(b)

1. The data is stored in B register; it is an 8-bit register so number of bits to be checked is 8.
2. Take data in accumulator, rotate the data the bit will come in carry flag.
3. Check the carry and go on counting after each rotate for number of 1.
4. Number of 0 = 8 – Number of 1.

Program

	Instruction		Comments
	MVI C, 08H	:	Set counter to eight
	MVI H, 00H	:	Clear H to store 1s
	MOV A, B	:	Load data to A
UP :	RAL	:	Rotate A
	JNC NEXT	:	Jump if no carry
	INR H	:	Increment H if there is a carry
NEXT :	DCR C	:	Decrement counter
	JNZ UP	:	Jump to UP if counter is not 0
	MVI A, 08H	:	Move total number of bits to accumulator
	SUB H	:	Subtract total number of bits to accumulator
	MOV L, A	:	Move A to L
	HLT		

Flowchart



Prob.31 (a) ***Write a program to compare two strings of ASCII characters to see if they are same. The length of the strings is in memory location 9041H; one string starts at memory location 9042 and other in memory location 9062. If the two string match, clear the memory location 9040H otherwise set memory location 9040 to FFH.***

OR

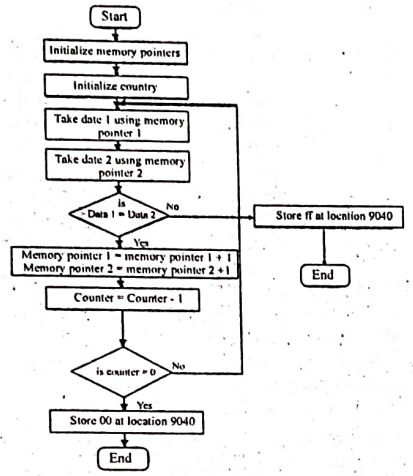
Write a program to compare two strings of ASCII characters to see if they are same. The length of the string is in memory location 90XXH; One string starts in memory location A0XXH and the other in memory location B0XXH. If the two strings match, clear the memory location B0XXH otherwise set memory location B0XXH to FF H.

(b) Explain with example, how flags can be set or reset in 16-bit instructions. **[R.T.U. 2011]**

Sol.(a) Program

1. The ASCII characters are stored in memory, so u memory pointers to access the data.
2. Number of data bytes to be checked is specified the memory location 9041, so read the number a use it as a counter.
3. Two memory pointers are required to access t two strings
4. Take the data using the memory pointer and che if they are equal or not.
5. Check for zero flag, if Z= 1 it means string is equ

Flowchart



Program

	Instruction		Comments
	LXI H, 9041H	:	Initialize the memory pointer
	MOV C, M	:	Copy data from memory to C
	INX H	:	Increment memory pointer
	LXI D, 9062H	:	Initialize another memory location by DE
UP :	LDAX D	:	Load accumulator from second memory
	CMP M	:	Compare with content of the firs memory
	JNZ NEXT	:	Go to NEXT if no match
	INX H	:	Increment first memory pointer
	INX D	:	Increment second memory pointer
	DCR D	:	Decrement counter

	JNZ UP	:	Jump to UP if counter is not 0
	MVI A, 00H	:	Load 00 to A
	STA 9040H	:	Store 00H to memory
	HLT		
NEXT :	MVI A, FFH	:	Load FFH to A
	STA 9040H	:	Store FFH to memory
	HLT		

Sol.(b) 16-bit Operations and Flags in 8085

Although the 8085 is an 8-bit processor, it also has some 16-bit operations. Any of the three 16-bit register pairs (BC, DE, HL) or SP could be loaded with an immediate 16-bit value (using LXI), incremented or decremented (using INX and DCX), or added to HL (using DAD). LHLD loaded HL from directly-addressed memory and SHLD stored HL likewise. The XCHG operation exchanges the values of HL and DE. Adding HL to itself performs a 16-bit arithmetical left shift with one instruction. The only 16 bit instruction that affects any flag was DAD (adding HL to BC, DE, HL or SP), which updates the carry flag to facilitate 24-bit or larger additions and left shifts (for a floating point mantissa for instance). Adding the stack pointer to HL is useful for indexing variables in (recursive) stack frames. A stack frame can be allocated using DAD SP and SPHL, and a branch to a computed pointer can be done with PCHL. These abilities make it feasible to compile languages such as PL/M, Pascal, or C with 16-bit variables and produce 8085 machine code.

Subtraction and bitwise logical operations on 16 bits is done in 8-bit steps. Operations that have to be implemented by program code (subroutine libraries) included comparisons of signed integers as well as multiply and divide.

Z (Zero) Flag : This flag indicates whether the result of a mathematical or logical operation is zero or not. If the result of the current operation is zero, then this flag will be set, otherwise reset.

CY (Carry) Flag : This flag indicates whether, during and addition or subtraction operation, carry or borrow is generated or not, if generated then this flag bit will be set. (This flag may also be set before a mathematical operation as an extra operand to certain instruction.)

AC (Auxiliary Carry) Flag : It shows carry propagation from D_3 position to D_4 position. To understand it better. Consider fig.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	0	0	0	0
0	0	1	0	1	0	1	1
1	0	1	1	1	1	1	1

Fig. : Auxiliary Carry Flag.

Here, a carry generators from D_3 bit position and propagates to the D_4 position. This carry is called auxiliary carry. This flag is never used for setting or testing a condition.

S (Sign Flag) : Sign flag indicates whether the result of a mathematical operation is negative or positive. If the result is positive, then this flag will reset and if the result is negative this flag will be set. This bit, infact, is a replica of the D_7 bit.

P (Parity) Flag : This flag indicates whether the current result is of even parity (1) or of odd parity (0).

Prob.32 ***(a) Write an assembly program to move a block of 10 data bytes from one memory location to another in reverse order.***

(b) Find the content of register C after following programs:

(i)
```
MVI A, 79 H
MVI B, 09 H
ADD B
DAA
MOV C, A
HLT
```

(ii)
```
MVI A, 25 H
Loop: NOP
MOV C, A
RLC
JNC Loop
MOV C, A
HLT
```

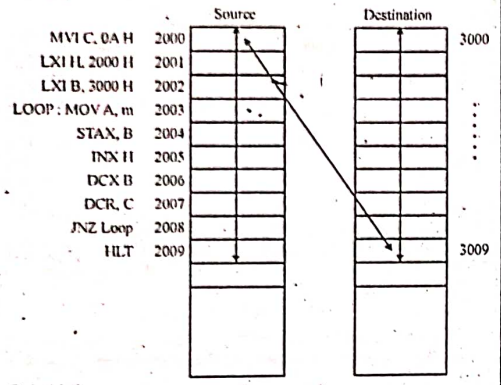
(c) What are the contents of PC and SP after:

(i) CALL instruction (ii) RET instruction in following program:

Memory Address	Instruction
1000	LXI SP, 3999 H
	LXI B, 2015 H
	CALL 2050 H
	↓
2050	MOV A, B
	ADD B
	RET

[R.T.U. 2010]

Sol.(a)



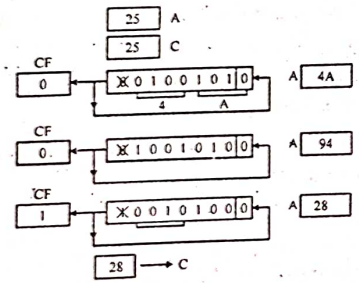
Sol.(b)(i)

79 → A 0111 1001
09 → B 0000 1001
1000 0010 – 82

ADD B
82 → A
DAA
1000 0010
0000 0110
1000 1000 – 88 A

MOV C, A
88 → C

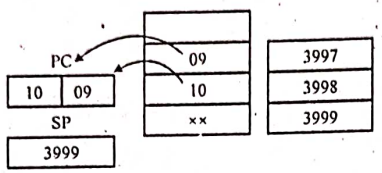
(ii)



Sol.(c)

```
1000, 01, 02   LXI SP, 3999 H
1003, 04, 05   LXI B, 2015 H
1006, 07, 08   CALL 2050 H
                  ↓
2050           MOV A, B
2051           ADD B
2052           RET
```

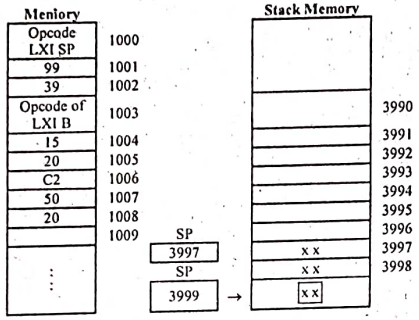
After RET



SP PC
3999H ← 3999H 1000

B – C
2015 ← 2015H

After CALL



(i) As the CALL instruction is executed the PC is pointing the address of next location *i.e.* PC 1009, but as two instruction CALL, 2050, takes the control, of upto location 2050. Therefore, the new value of program counter is loaded *i.e.* PC 2050, but the previous value of PC is stored in stack automatically. Therefore the stack pointer will be decremented by 2 *i.e.* SP 3997

∴ After CALL New SP 3997 and PC 2050

(ii) As the RET is executed μp takes back the add from stack, back to the PC. Therefore SP will be incremented by (SP + 2) from previous value. *i.e.*

PC SP
1009 and 3999

Prob.33 ***(a) What do you mean by 'Programming model' of 8085 MPU? Explain with suitable diagram.***

[R.T.U. 2010; Raj. Univ. 2007]

(b) Is it possible that an output and an input port have the same 8 bit address? If yes, how does the 8085 microprocessor differentiate between the ports? If no, why? **[Raj. Univ. 2007]**

Sol. (a) The microprocessor can be represented in terms of its :

(1) Hardware model (physical electronic components)

(2) Software model (programming model)

Hardware Model

Hardware model of a 8085 microprocessor includes basic three components : ALU, register array and control unit. These are communicated to each other by various internal connections called an internal bus.

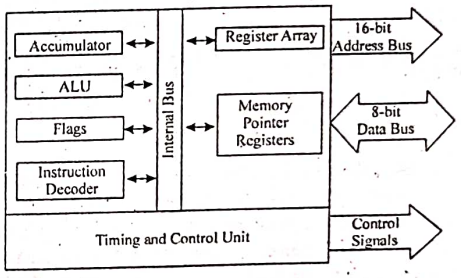


Fig. 1: Hardware model

ALU : The ALU performs arithmetic operations such as addition, subtraction and logical operations such as AND, OR, EXOR etc. It is group of logic circuits. The size of ALU will be size of microprocessor (8 bit for 8085 MP).

Accumulator : *Refer to Prob.2.*

Software Model (Programming Model)

Software model of a processor means the available software resources of the microprocessor. The programming model of microprocessor is the layout of the internal register and flags within the microprocessor. The programming model of 8085 microprocessor has been shown in fig. 2. It contains ten registers, eight registers of 8 bit each (A,F,B,C,D,E,H,L) and two registers of 16-bit (PC : program counter, SP: Stack pointer).

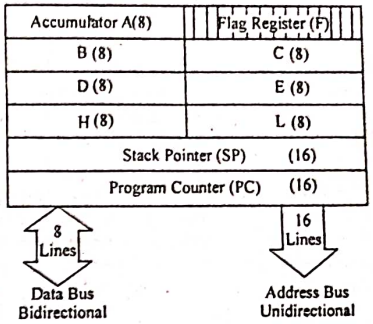


Fig. 2: Programming model

Flag Register : The flag register is nothing but a group of flip-flop used to give status of different operations results. The flag register is connected to ALU, when an operation is performed by ALU the result is transferred to an internal data bus and status of result will be stored in flip-flops. It only give status if an operation is performed in ALU.

The different flags and their position in flag register are shown in fig.

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

Fig. 3: Flag Register

(i) CY–Carry: If an operation performed in ALU, generated a carry from D_7 bit, the carry flag is set, otherwise it is reset. The carry flag also serves as a borrow flag for subtraction.

(ii) AC – Auxiliary carry flag: If an operation performed in ALU generates a carry from lower nibble (*i.e.* D_0 to D_3) to upper nibble (i.e D_4 to D_7). The AC flag is set *i.e.*, a carry given by D_3 bit to D_4 is a AC flag.

This is not a general purpose flag, it is only used internally by microprocessor to perform binary to BCD conversion. It is not available for programmer for any decision making.

(iii) Z-Zero flag : If an operation in ALU result is zero, the zero flag is set. If the result is not zero, the zero flag is reset.

(iv) S-Sign flag : In Sign flag magnitude format, the sign of a number is indicated by MSB bit. If MSB bit =0, the number is positive. In 8085 microprocessor, MSB bit is D_7 bit. The sign flag is exact replica of D_7 bit of the result. If D_7=1 : the sign flag is set and if D_7=0, the flag is reset.

(v) P – Parity flag : This bit is used to indicate the parity of the result. If the result contains even numbers of 1's, this flag is set. If the result contains odd numbers of 1's, this flag is reset.

Program Counter (PC): This is a 16-bit register deals with the fourth operation, sequencing the execution of instructions. This register is a memory pointer, always points to address of memory from where the next instruction is to be fetched and executed, that's why this is a 16-bit register. The microprocessor uses this register to sequence the execution of instructions. When microprocessor executes one fetching instruction, the PC contents are automatically incremented by one point to the next memory location.

Stack Pointer: This is a 16-bit register used as a memory pointer to define the stack starting address. It points load to memory location in R/W memory called the stack. Stack is reserved portion of memory where register pair information can be stored or taken back under software control.

Sol. (b) Yes, the same address can be provided to I/P memory, because in 8085, there is one signal called IO/$\overline{M}$ which differentiates the address. If IO/$\overline{M}$ = 0, then the address on address bus is for memory device.

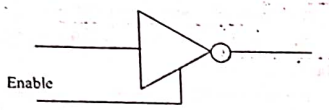
If IO/$\overline{M}$ =1, the address on address bus is for I/O provided, the same address is provided to I/O and memory.

Prob.34 ***(a) What do you mean by Tri- state logic? Explain its necessity in microcomputer system.***

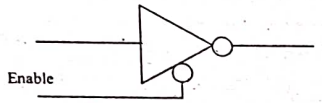
(b) With help of suitable diagram and truth table explain the working of an 8 to 3 priority encoder.

[Raj. Univ. 2007, 2002]

Sol.(a) Tri-state logic devices have three states : logic 1, logic 0 and high impedance. The term TRI-STATE is a trademark of national semiconductor and is used to represent three logic states. A tri-state logic device has a third line called enable, as shown in fig. When this line is activated, the tri-state device functions the same way as ordinary logic devices. When the third line is disabled, the logic device goes into the high impedance state as i it were disconnected from the system. Ordinarily, curren is required to derive a device in logic 0 and logic 1 states In the high impedance state, practically no current is drawn from the system. In micro computer system, peripheral are connected in parallel between the address bus an data bus. However, because of the tri-state interfacin devices, peripherals do not load the system buses. Th microprocessor communicates with one device at a tim by enabling the tri-state line of the interfacing device. Tri state logic is essential to proper functioning of th microcomputer.



(a) Active High

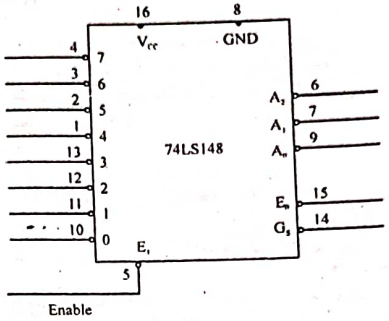


(b) Active low

Fig.

Sol. (b) The encoder is a logic circuit that provides th appropriate code (binary, BCD) as output for each inpu signal. The process is the reverse of decoding. Figure (a shows an 8 to 3 encoder. It has eight active low inputs an three output lines. When the input line 0 goes low, the outpu is 000, when the input line 5 goes low, the output is 10 However, this encoder is unable to provide an appropriat output code, if two or more input lines are activate simultaneously. Encoders called priority encoders ca resolve the problem of simultaneous inputs.

Figure (b) shows the logic symbol of the 74LS148, a 8 to 3 priority encoder. It has eight inputs and one activ low enable signal. It has five output signals. Three ar encoding lines and two are output – enable indicators. Th output lines G_S and E_0 can be used to encode more tha eight inputs by cascading these devices. When th encoder is enabled and two or more input signals ar activated simultaneously, it ignores the low priority inpu and encodes the highest priority input.



(a)

74 LS148 Function Table

Input								Output				
E_1	0	1	2	3	4	5	6	7	A_2	A_1	A_0	G_S	E_0
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H	X	X	X	X	X	X	X	H	H	L	H	L
L	L	H	X	X	X	X	X	X	H	L	H	L	H
L	L	L	H	X	X	X	X	X	H	L	L	L	H
L	L	L	L	H	X	X	X	X	L	H	H	H	L
L	L	L	L	L	H	X	X	X	L	H	L	H	H
L	L	L	L	L	L	H	X	X	L	L	H	H	L
L	L	L	L	L	L	L	H	X	L	L	L	H	H
L	L	L	L	L	L	L	L	H	H	H	H	H	H

(b)

Fig.

Similarly, the output of this is decoded as active low, when input signal 7 is active, the output is 000 rather than 111. □□□

ADVANCED ASSEMBLY LANGUAGE PROGRAMMING 3

CHAPTER IN A NUTSHELL

❑ **Interrupt Controller 8259A**

It is a programmable interrupt controller it can :

1. Manage 8 interrupts.
2. Vector on interrupt request.
3. Resolve eight levels of interrupt priorities in a variety of modem.
4. Mask each interrupt request individually.
5. Read the status of pending, in source and masked interrupts.
6. To set up to accept either the level triggered or the edge triggered interrupt request.
7. Be expanded to 64 priority level by cascading additional 8259 As

USART 8251
PPI 8255
TIMER 8254

It generates accurate time delays and can be used for applications such as a real time clock, an equal counter, a digital one short, a square wave generator, and a complex wave from generator.

It can initialize a counter.

In even counters, it can real the value of counter in progress through programming.

❑ **Macros and Subroutine :** A subroutine is a group of instruction written separately from the main program to perform a function that occurs repeatedly in the main program.

The 8085 µP has 2 instructions to implement subroutine : CALL and RET (return to macro program from a subroutine)

❑ **Stack :** The stack is shared by the programmer and the µP, the programmer can retrieve and store the contents of a regular pair by POP and PUSH instructions.

Similarly, the µP automatically stores the content of the program counters when a subroutine is called.

❑ **Counters :** A counter is designed simply by loadin appropriate number into one of the register using the (increment by one) or the DCR (decrement by one) instruc A loop is established to update the count and each cou checked to determine whether it has reached the final num if not, the loop is repeated.

❑ **Time Delay :** A register is loaded with a num depending upon the time delay required and then the reg is decremented until it reaches zero by setting up a loop a conditional jump instruction.

The loop causes the delay depending upon the c period of the system.

❑ **Interrupt Structure and its Programming**

Interrupt is an asynchronous process of communic with the µP, initiated by an external peripheral.

Interrupts	Type	Instructions	Hardwar
TRAP	Nonmaskable	Independent of EI and DI	No extern
RST 7.5	Maskable	Controlled by Unmasked by SIM	No extern
RST 6.5	Maskable	Unmasked by SIM	No extern
RST 5.5	Maskable	Unmasked by SIM	No extern
INTR	Maskable		Controlle RST cod from
8085		EI and DI	external hardware

A programmable interrupt controller such as the 82 used commonly to implement and extend the capabili the 8085 interrupt.

The direct memory access (DMA) is a process of speed data transfer under the control and external d such as a DMA controller.

PREVIOUS YEARS QUESTIONS

PART-A

Prob.1 *What are interrupts?* [R.T.U. 2019]

Sol. Interrupts : Interrupt is a process by which some external device or peripheral informs microprocessor to become ready for data communication with the help of accepting the request made. Interrupt is a process in which the control of the program transfers from the main program to the starting location defined by the interrupt.

Prob.2 *Differentiate between Synchronous and Asynchronous data transfer.* [R.T.U. 2019]

OR

What do you mean by synchronous and asynchronous data transfer? [R.T.U. 2010]

Sol. In **synchronous transmission,** a receiver and a transmitter are synchronized. In this method, a block of characters is transmitted along with synchronization information. See fig. (a) such type of transmission is used for high speed data transfer (more than 20 k bits / second).

In **asynchronous transmission,** each character is transmitted with start and stop bits. Transmission begins with one start bit (low), followed by 8 bit code of the character and one or two stops bits (high) at the end. See fig. (b). The asynchronous transmission is used in low-speed data transfer (less than 20k bits / second).

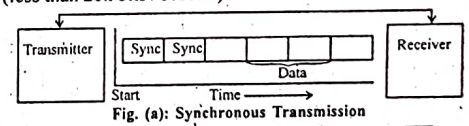


Fig. (a): Synchronous Transmission

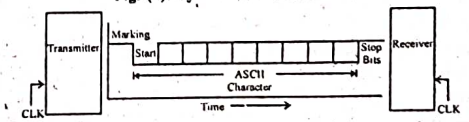


Fig. (b): Asynchronous Transmission

Prob.3 *Differentiate vectored and non vectored interrupts of 8085.* [R.T.U. 2016]

Sol. Difference between vectored and non vectored interrupts :

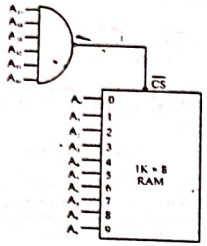
S. No.	Vectored Interrupt	Non-vectored Interrupt
1.	Interrupts which are automatically transformed (or vectored) to specific locations on memory page 00H without any external hardware are called vectored interrupts.	Interrupts which are not automatically vectored to specific locations on memory page 00H are called non-vectored interrupts.
2.	Vectored interrupts do not require the $\overline{\text{INTA}}$ signal or an input port.	Non-vectored interrupts require external hardware to transfer it to a particular location.
3.	These interrupts and their call location are as follows : Interrupts / Call Location: (i) TRAP 0024 H; (ii) RST 7.5 003C H; (iii) RST 6.5 0034 H; (iv) RST 5.5 002C H	Only INTR interrupt is non-vectored interrupt. INTR requires RSTN instruction for its working.

Prob.4 *Write an assembly language program to add a memory block of 10 bytes starting from 2000 H and strore the sum in the memory at 200 F H location. If carry generated, store the carry at 2010 H location.* [R.T.U. 2012]

Sol.

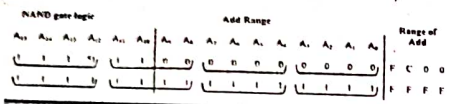
```
        LXI H, 2000 H
        MVI C, OA H
        XRAA
Loop :  ADD M
        JC END
        INX H
        DCR C
        JNZ Loop
        STA 200 F H
        JMP SKIP
        MVI A, 01 H
        STA 2010 H
SKIP :  HLT
```

Prob.5 ***Find the address range of following memory device***



[R.T.U. 2010]

Sol.



Prob.6 ***What do you mean by analog and digital data?***

[R.T.U. 2010]

Sol. The real word deals with physical quantities, such as temperature, pressure etc. which are continuous. These physical quantities can be converted into electrical quantities such as voltage or current, known as analog data. But microprocessor based system deals with digital data (0, 1) which are discontinuous.

PART-B

Prob.7 ***What is Subroutine? Explain the use of stack in CALL and RETURN instructions.*** [R.T.U. 2019]

Sol. Subroutine : Subroutine is a group of instructions written separately from the main program, to perform that functions which occurs repeatedly in the main program. While writing a program, certain operation may repeat several times. The program for these operations are written separately as a small program and are stored in the memory locations. These small program stored in the memory which are used frequently for repeated task in the main program, are called **subroutines**. The small programs to perform various functions like multiplication, sine, cosine, logarithm, square, square root are written and are stored in different memory locatio Whenever these operations are required, these subrouti are called from the specified memory locations and are u in the main program.

The 8085 microprocessor has two instructions to these subroutines. The first instruction is CALL 16–bit mem address. This instruction will call subroutine specified b 16–bit memory address, wherever required, these subrouti are called from the specified memory locations and are u in the main program, the second program, the instruction RI This instruction will return from the subroutine to the m program. The instruction RET issued at the end of subroutine. RET instruction shows that the operation specif by the subroutine is complete and the program will return the main program.

When a subroutine is called using CALL instructi the contents of program counter, which is the address of instruction following the CALL instruction is stored on stack and the program execution is transferred from the m program to the subroutine address specified by CAL 16–bit memory address instruction. When the RET instructi is executed at end of the subroutine, the program is transfer to the address stored on stack and it is returned to the m program.

The CALL instruction is used in the main progr whereas the RET instruction is used at the subroutine. T subroutine may be used several times in a program. The may be several subroutines (for different operations) used a program. Subroutines are used to save memory locatic Some important subroutine may be written in the ROM the manufacture.

The subroutine operation is described below with t help of example.

Memory Address	Memory Address
2000	-
2001	-
:	-
2019	-
2020	CALL
2021	50
2022	30
2023	-
:	:
2050	-

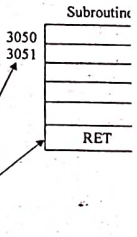


Fig.

Instruction of Subroutine :

	Opcode	Operand	Call subroutine unconditionally
(A)	CALL	16 bit memory subroutine	• This is a 3 byte instruction that transfers the address of a progr sequence to subroutine address. • Saves the content of program coun on stack.

		• Decrement the stack pointer register by two • Jump unconditionally to the memory location specified by the 2nd and 3rd bytes
(B)	RET	Return from subroutine unconditionally • This is a 1 byte instruction • Insert the two bytes from the top of the stack into the program counter and increment the stack pointer register

Example: Memory Address

2000	LXI SP, 2400H	; Initialize the stack pointer of 2400H.
↓	↓	
2040	CALL 2070 H	; Call the subroutine located at 2070 H.
2041		
2042		
2043	NEXT INSTRUCTION	; The address of the next instruction following the CALL instruction.
↓	↓	
205F	HLT	; End of the main program.
2070	First subroutine Instruction	; Begining of subroutine.
↓	↓	
207F	RET	; End of subroutine
2080	↓	
	Other subroutine	
	↓	
2398	Empty Space	
23FF	↓	
2400		; The stack is initialized at 2400 H.

Prob.8 ***Explain, how subroutines are implemented and executed. Explain call by value and call by reference.*** [R.T.U. 2017]

Sol. Subroutine Implemented : In computer programming, a **subroutine** is a sequence of program instructions that perform a specific task, packaged as a unit. This unit can then be used in programs wherever that particular task should be performed.

Subprograms may be defined within programs, or separately in libraries that can be used by multiple programs. In different programming languages, a subroutine may be called a **procedure**, a **function**, a **routine**, a method, or a **subprogram**. The generic term **callable unit** is sometimes used.

The name *subprogram* suggests a subroutine behaves in much the same way as a computer program that is used as one step in a larger program or another subprogram. A subroutine is often coded so that it can be started (called) several times and from several places during one execution of the program, including from other subroutines, and then branch back (*return*) to the next instruction after the *call*, once the subroutine's task is done. Maurice Wilkes, David Wheeler, and Stanley Gill are credited with the invention of this concept, which they termed a *closed subroutine*, contrasted with an *open subroutine* or macro.

Subroutines are a powerful programming tool, and the syntax of many programming languages includes support for writing and using them. Judicious use of subroutines (for example, through the structured programming approach) will often substantially reduce the cost of developing and maintaining a large program, while increasing its quality and reliability.Subroutines, often collected into libraries, are an important mechanism for sharing and trading software. The discipline of object-oriented programming is based on objects and methods (which are subroutines attached to these objects or object classes).

The content of a subroutine is its body, which is the piece of program code that is executed when the subroutine is called or invoked.

Subroutine Execution : A subroutine may be written so that it expects to obtain one or more data values from the calling program (to replace its parameters or formal parameters). The calling program provides actual values for these parameters, called arguments. Different programming languages may use different conventions for passing arguments :

Convention	Description	Common use
Call by value	Argument is evaluated and copy of value is passed to subroutine	Default in most Algol-like languages after Algol 60, such as Pascal, Delphi, Simula, CPL, PL/M, Modula, Oberon, Ada, and many others. C, C++, Java (References to objects and arrays are also passed by value)
Call by reference	Reference to argument, typically its address is passed	Selectable in most Algol-like languages after Algol 60, such as Algol 68, Pascal, Delphi, Simula, CPL, PL/M, Modula, Oberon, Ada, and many others. C++, Fortran, PL/I

Prob.9 ***What is the use of "Stack"? Illustrate the PUSH and POP operations with help of suitable exmaple.*** [R.T.U. 2013]

OR

Explain use of stack during interrupt processing. [R.T.U. 2017]

OR

Explain the Stack with examples. [R.T.U. 2012]

Sol. Stack : "The stack is a group of memory location in R/W memory that is used for temporary storage of binary information during the execution of program".

The stack can be defined as a set of memory locations in the R/W memory, specified by a programmer in the main program. This set of memory location can be used to store the data temporarily during the execution of a program. Sometimes, it becomes necessary during the execution of program to store the contents of certain registers because these registers are required to store data for some other operations in the subsequent steps of a program. The contents of these registers are moved to certain memory location by PUSH instruction. Then these registers can be used for some other operations. After completing these operations, those contents of registers, which were saved in the memory, are transferred back to these registers by POP instructions. Memory locations for this purpose are specified by a programmer in the beginning of a program. These memory locations are called the stack top. A special 16–bit register called the stack pointer register (SP) holds the address of the stack top.

The stack operation can be described in the following way :

(i) In the beginning, the stack pointer (SP) is loaded by 16–bit memory address using the instruction LXI SP, 16–bit address. This instruction will load the 16–bit memory address in the stack pointer register of the microprocessor. Once the stack location is defined, the storing of data begins at the memory location that is one less than the address in the stack pointer register. For example, the instruction LXI SP 3099H will load memory address 3099 in the stack pointer register. Now the storing of data begins at memory location 3098 and continues in the decreasing memory address like 3097, 3096 and so on.

(ii) Now the data bytes in the register pairs can be stored on the stack. The instruction for example PUSH B will copy the contents of register pair BC on the stack, the contents of B register are copied in the memory location 3098 and contents of C register are copied in the memory location 3097. After this Push B instruction, the stack pointer register is also decremented by 2 *i.e.* it contains the address 3097.

(iii) The register pair BC now can be used to store any other data in subsequent step of a program. When this operation is finished, the contents of register pair BC which were stored on the stack can be transferred back to this register pair by using the instruction POP B. This instruction will copy the contents of top of the stack memory locat 3097 in the register C and 3098 in the register B and stack pointer register again contains the address 30 (incremented by two).

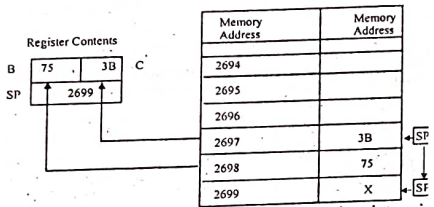
Example :

The following example also explains the stack operati Consider the contents of register B are 75 and contents register C are 3B. The stack is initialized with instruct LXI SP, 2699.

Now the instruction PUSH B will copy the conte of register B in the memory location 2698 and the conte of register C in the memory location 2697. After this PU B instruction, the stack pointer register is decremented b and now, it contains the memory address 2697 as shown in

After Push B instruction, the stack pointer regi contains the memory address 2697. The contents of B regi and C register are not destroyed, as these registers can n be used to store any other data.

Now the instruction POP B will transfer the conte of 2697 to register 3B and contents of 2698 to register E shown in fig.



After the POP B instruction, the stack point regis contains the memory address 2699.

Prob.10 ***Explain the addressing modes for 8(microprocessor by giving examples of each.*** [R.T.U. 20

OR

Explain addressing modes of 8085. [R.T.U. 20

Sol. There are several different addressing modes for 8(microprocessor. Some of them are given below :

- **Inherent/Implied Addressing Mode :** belongs to the 0-Address class. It is essentially not addressing mode as no addressing mode is used to refer any operand. Only the opcode for the instruction is provid There is at most one operand required in some instructic

and that operand is implicitly used from the accumulator register. Like in case of Not instruction, the operand is fetched from the accumulator and the result (the complement) is also stored in the same register. Halt is an instruction which doesn't require any operand and hence it uses Inherent Addressing Mode.

- **Immediate Addressing Mode:** In this addressing mode, the operand itself is provided along with the opcode of the instruction. There is no memory address required for the operand to be fetched. It is commonly used to store constant in a register. Like storing the value of π (pi) in the register B. It can also be used in a loop. We can initialize the register with the value for which we want to use the loop. For example- MVI B,40H (40H is copied into the register B).
- **Direct/Absolute Addressing Mode:** We provide the addresses of the operands along with the opcode for the instruction. The size of the address can be different depending on what type of address we are referring, like I/O address is of one byte whereas memory address is 2 bytes or 16 bits, This address will point to the memory location where the operand is stored. It is the most natural and simple way for providing operands for the instruction and takes three machine cycles to read the address (which can be 2 bytes) and fetch the operand. For example - LDA 3000H (The content at the location 3000H is copied to the register A).
- **Register Addressing Mode:** The operands are stored in the registers. Therefore, we only need to provide the address of the register along with the opcode of the instruction. This results in a shorter instruction which takes less time to read. The operand fetch is also fast as we just have to look into the corresponding register and not fetch it from slow main memory. For example - MOV A, C (the content of C is copied into the register A).
- **Register Indirect Addressing Mode:** In this addressing mode, we provide along with the instruction opcode, an address is stored in a register pair (memory address is 16 bits) which points to memory location that stores the address of the operand. In this case, the effective address is essentially the content of the memory location in the register provided with the opcode. This is slower in comparison to direct mode of addressing as it involves reading from a register along with a memory fetch. But it is a very flexible way of providing the address as we don't have to change the program or the instruction to get a different operand as we can just change the address stored in the register provided as part of the instruction. For example - MOV A, M (data is transferred from the memory location pointed by the register to the accumulator).

Prob.11 ***(a) Compute the time delay introduced by following routine :***

		T-States
	MVI A, 0AH	*7*
Loop :	*NOP*	*4*
	NOP	
	MOV B, A	*4*
	DCRA	*4*
	JNC Loop:	*10/7*
	RET	*10*

Assume : Clock frequency of Microprocessor is 3 MHz

(b) Write an assembly program to implement 16 bit counter.

(c) Differentiate maskable and Nonmaskable interrupts of 8085. [R.T.U. 2016]

Sol.(a) Given :
Value of T= 3MHz= 1/3*10^6
T-States of MVI = 7T
In Loop
Total T-States = 26T*10
T-States of RET=10T
Therefore
Time Delay = 10T+7T+(26T*10)
Value of T=1/3*10^6
Time Delay =40/(3*10^6)

Sol.(b) 16 bit assembly program to convert a number to string, also note that you'll have to $-terminate the string instead of NUL-terminate.

```
[org 0x100]                 ; DOS .COM files are loaded at CS:0100h
[bits 16]                   ; And are 16-bits
start:
mov ax, 0                   ; The current count
printloop:
push ax                     ; Save current number
Call myconvertfunc          ; Some function to convert a number in ax and return a '$'-terminated string in dx

mov ah, 0x09
int 0x21
mov dx, newline             ; Point dx to newline string
mov ah, 0x09                ; Print $-terminated string in dx
int 0x21
pop ax                      ; Restore current number
inc ax                      ; Next number
cmp ax, 50000               ; Compare number to the maximum number
jbe printloop               ; Notice branching based on unsigned comparison is needed
mov ax, 0x4c00              ; Return 0. AH=4Ch AL=Return value
int 0x21
newline: db 13, 10, '$'     ; String containing "\r\n$"
```

Sol.(c) Difference between Maskable and Non-Maskable Interrupts :

S. No.	Maskable Interrupt	Non-Maskable Interrupt
1.	It can be masked off or made pending.	It cannot be masked off or made pending.
2.	This interrupt does not disable non-maskable interrupts	This interrupt disables all maskable interrupts.
3.	It is used to interface peripherals.	It is used for emergency purpose like power failure, smoke detector etc.
4.	Lower priority	Higher priority.
5.	Vectored or non-vectored.	Vectored
6.	Response time is high.	Response time is low.

Prob.12 *(a) Write a program to transfer a block of 10 data elements from memory location 5000H to 6000H.* [R.T.U. 2015]

(b) Explain the following Instructions:-
(i) CMA
(ii) CMP
(iii) LDAX
(iv) LXI
(v) XCHG
(vi) IN [R.T.U. 2015]

Sol. (a) The 8085 program to transfer a block of 10 data elements from memory location 5000H to 6000H is given below :

```
LXI D,5000   : "Load destination address in DE pair"
MOV C,0a     : "Store the count in register C"
LXI H,5000   : "Load H-L pair with the memory address 4000H"
LOOP: INX H  : "Increment memory address pointer"
MOV A,M      : "Copy element to Accumulator"
STAX D       : "Store the element to the address in the DE pair"
INX D        : "Increment destination address"
DCR C        : "Decrement count"
JNZ LOOP     : "Jump on non-zero to the label LOOP"
HLT          : "Stop the program"
```

Sol. (b) (i) CMA : Use : CMA

Also known as Complement accumulator. In this instruction, the contents of accumulator register are complemented. There are no changes in any of the flags. It requires no operands and only has opcode in its machine code. As it only contain the opcode, it only takes one byte of memory.

(ii) CMP : Use : CMP B or CMP M.

It compares the contents of a register or a memory location with that of the accumulator. The contents of both the registers or memory and the accumulator are preserved during this instruction. No alterations in the value of the register or the accumulator. The results of the comparison are shown as setting different flags :

if (A) < (reg/mem): carry flag is set, s = 1 if (A) = (reg/mem): zero flag is set, s = 0

if (A) > (reg/mem): carry and zero flags are reset, s = 0

(iii) LDAX :Use : LDAX B or LDAX D (regs. pair)

Also known as Load accumulator indirect. This is used to load the contents into the accumulator using the Indirect Addressing Mode. The memory location pointed by the register pair stores the address to another memory location. This instruction copies the contents from that memory location into the accumulator. The contents of neither the register pair and nor the memory location pointed by them are altered.

(iv) LXI :Use : LXI <Reg. Pair>, 16 bit data (Immediate)

This instruction loads the 16 bit data provided (Immediate) into the register pair designated in the operand. Example - LXI B, 4052.

(v) XCHG : Use : XCHG

In this instruction, the contents of register pair H and are interchanged with that of register pair D and E, i.e. contents of register H are exchanged with the contents of register D and the contents of register L are exchanged with the contents of register E.

(vi) IN : Use : IN 8 bit port address

Using this instruction, one can load the contents in the input port designated in the operand into the accumulator. Example - IN 82.

Prob.13 *Write a program to sort the 10 data elements in descending order assume that the data are stored at 4000H to 5000H.* [R.T.U. 2015]

Sol. The 8085 program for sorting 10 data elements stored in memory address 4000H is given below :

```
MVI B,0aH          : "Initialize counter 1 to 10"
REPEAT: MVI C,0aH  : "Initialize counter 2 to 10"
LXI H,4000H        : "Load H-L pair with the memory address 4000H"
BACK: MOV A,M      : "Move the number from memory to register A"
INX H              : "Increment the memory pointer"
CMP M              : "Compare the number with the next number"
JNC NOTCHANGE      : "Don't interchange if current no > next number"
JZ NOTCHANGE       : "Don't interchange if current no = next number" :"Swap the numbers if current no < next number"
MOV D,M
MOV M,A
DCX H              : "Decrementing H-L pair"
MOV M,D            : "Move next number from D to memory"
INX H              : "Incrementing H-L pair"
NOTCHANGE: DCR C   : "Decrement counter 2"
JNZ BACK           : "If counter 2 is non-zero then go back"
DCR B              : "Decrement counter 1"
JNZ REPEAT         : "If counter 1 is non-zero then repeat the steps again"
HLT                : "Stop the program"
```


Prob.14 *(a) Is there any difference between vectored and non vectored interrupts? Explain with suitable examples.*

(b) Explain the control word of 8259. [R.T.U. 2014]

Sol. (a) Difference between vectored and non vectored interrupts : *Refer to Prob.3.*

Example : A vectored interrupt, on calling, the program control is automatically vectored (transferred) to specific addresses in memory. For a non-vectored interrupt, the program control location is decided by external hardware. For example , for the 8085, TRAP, RST 7.5, RST 6.5 and RST 5.5 are vectored interrupts for which the call locations are 0024H, 003CH, 0034H and 002CH respectively. INTR is a non vectored interrupt.

Sol. (b) The 8259 is a programmable interrupt controller used for the 8085 and 8086 . It is a 28-pin IC. The pins A_0 and D_7-D_0 together make a 9-bit control word :-

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

There are 7 Control words. The 4 control words ICW1, ICW2,ICW3 and ICW4 are Initialisation Control Words, used for the initialisation sequence of the 8259. The remaining 3 control words OCW1, OCW2 and OCW3 are operational control words. During intialisation, the ICWs specify the ISR address, whether edge triggered or level triggered mode is used, whether ISRs are 4 or 8 bytes apart, whether the 8259 is in master or slave mode etc.

The OCWs specify interrupt masks, end of interrupt signal, rotation of priorities mode etc.

Prob.15 *Explain the RIM and SIM instruction briefly.* [R.T.U. 2013]

OR

Explain the format of SIM. [R.T.U. 2014]

OR

Explain the instructions SIM and RIM and illustrate how to use them for 8085 interrupts. [Raj. Univ. 2006]

Sol. SIM : Set Interrupt Mask

Opcode	operand	bytes	m cycles	T states
SIM	None	1	1	4

Description : This instruction is used to implement 8085 interrupts (RST 7.5, 6.5, 5.5) and serial data output. The instruction interprets accumulator contents as follows :

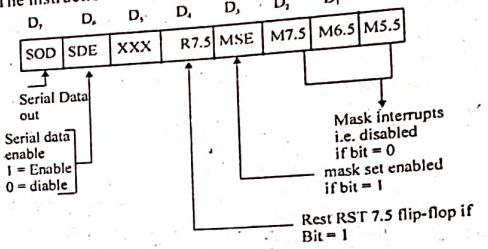


Fig.

SOD – Serial output data : Bit D_7 of accumulator is latched into SOD output time and made available to a serial peripheral if bit $D_6 = 1$.

SDE – Serial data enable : If this bit is 1 it enables serial output to implement serial output bit need to be enabled.

XXX – Don't care (None)

MSE – Mask set enable : If this bit is high it enables function of bit D_2, D_1 and D_0. This is master control over all interrupt masking bits.

If this bit is low, bit D_2, D_1 and D_0 do not have any effect on masks.

M7.5	$D_2 = 0$	RST 7.5	is enhanced
	1	RST 7.5	is disabled/masked
M6.5	$D_1 = 0$	RST 6.5	is enabled
	1	RST 6.5	is masked
M5.5	$D_0 = 0$	RST 5.5	is enabled
	1	RST 5.5	is masked

RIM : Read interuppt Mask

This instruction is used to read/check the status of interrupt 7.5, 6.5 and 5.5 and read to serial input bit. The instruction loads eight bits in accumulator with following interpretation.

Opcode	operand	bytes	m cycles	T-states
RIM	None	1	1	4

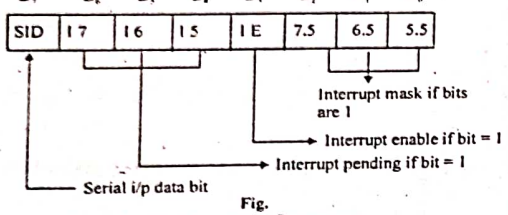


Fig.

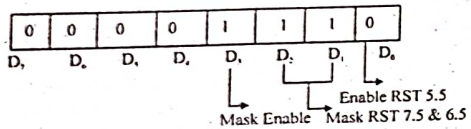
Use of RIM & SIM for 8085 Interrupts

SIM : We want to enable interrupt RST 5.5 and mask other interrupts we can use SIM instructions as follows :

MVI A, OE : This will set D_3 i.e. RST 5.5 at $D_0 = 0$

SIM : Enable RST 5.5.

In this instruction set, accumulator is loaded with OEH.



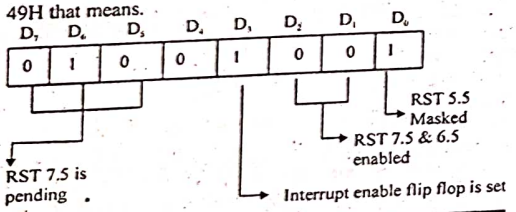
After execution of SIM, RST 5.5 is enable and other interrupts are disabled/ masked.

RIM : If we want to check the status of interrupt either they are enable/disable or pending. This instruction will be best way to do this.

Following instruction interpretation of accumulator as :

Opcode	operand	bytes	m cycles
RIM	None	1	1

After execution of above instruction, accumulator contained 49H that means.



Prob.16 *Explain the implementations of stack in 8085 microprocessor programming.* [R.T.U. 2014]

Sol. The stack is a group of memory locations in the R/W memory that is used for temporary storage during the execution of a program, eg., while calling sub routines. The stack is a LIFO (Last In First Out) structure. The starting location of the stack is defined by loading a 16 bit address into the stack pointer(the stack pointer (SP) points to the top of the stack).

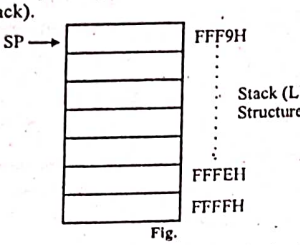


Fig.

In 8085 microprocessor system, the beginning of the stack is defined by using the instruction : LXI SP, 16 bit

This instruction will load the 16-bit address specified into the stack pointer. The 8085 provides two instruction PUSH and POP for storing information on the stack and retrieving it back. As an example, consider the following instruction sequence :

```
2000    LXI   SP, 2099H
2003    LXI   H, 42F2H
2006    PUSH  H
2007    ...
 :
2010    POP   H
```

The instruction 'LXI SP, 2099H' on address 2000 is a 3 byte instruction which loads the stack pointer register with the address 2099H. Then 'LXI H, 42F2H, loads the HL register pair with the value 42F2H. Due to the 'PUSH H' instruction, the SP is decremented to 2098H and then the value 42H is stored at location 2098H, and then the value F2H is stored at 2097 H (after one more decrement of the SP). After some instructions, 'POP H' instruction is present at location 2010. During popping, the stack operates in a 'use then increment' style. The value F2H is loaded into H and then the SP is incremented. Then the value 42H is loaded into L and then SP is incremented.

Prob.17 *Interface two chips of 4K byte RAM and one chip of 2K byte ROM to MP 8085, by absolute decoding using 74 LS 138 decoder. Give the memory map starting at address 8000H for EPROM.* [R.T.U. 2011]

Sol.1. Memory interfacing with foldback address

Address Decoding Table

Chip no.	Binary address																Hexadecimal address
	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	
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	From 8000H
	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	To 8FFFH
2	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	From 9000H
	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	To 9FFFH
3	1	0	1	0	x	0	0	0	0	0	0	0	0	0	0	0	From A000H to A7FFH or
	1	0	1	0	x	1	1	1	1	1	1	1	1	1	1	1	From A800H to AFFFH

Here, in this problem, as shown in the address decoding table the address line A_{11} is not decoded for the 2K ROM chip. This chip is selected by A_{15}, A_{14}, A_{13} and A_{12} address lines. Hence, A_{11} address line may be 0 or 1. One the basis of the line this ROM chip will have two address spaces, *i.e.*, A000H to A7FFH and A800H to AFFFH.

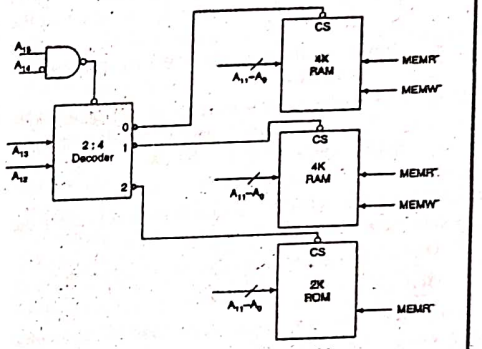


Fig. : Interfacing with foldback address

2. Memory interacing without foldback address:

Address Decoding Table

Chip no.	Binary address																Hexadecimal address
	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	
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	From 8000H
	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	To 8FFFH
2	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	From 9000H
	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	To 9FFFH
3	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	From A000H
	1	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	To A7FFH

Here, in this problem, as shown in the address decoding table the address line A_{11} is also decoded by using an extra NAND gate for the 2K ROM chip. Now this chip is selected by A_{15}, A_{14}, A_{13} and A_{12} and also by address line. This ROM chip now will be selected only when $A_{15} = 1$, $A_{14} = 0$, $A_{13} = 1$, $A_{12} = 0$ and $A_{11} = 0$. One the basis of the line this ROM chip will have two address spaces A000H to A7FFH.

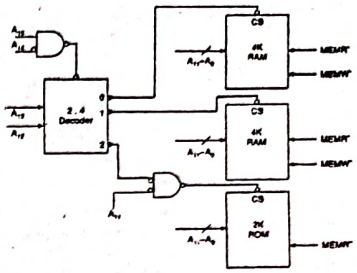


Fig. : Interfacing without foldback address

Prob.18 *Draw the timing diagram for instruction SHLD address. Also state machine cycles for it.*

OR

Draw the timing diagram for the instruction SHLD BABAH. Show of the status of the various signals

$A_8 - A_{15}, AD_7 - AD_0, ALE, \overline{RD}, IO/\overline{M}, S_0, S_1, \overline{WR}$

[R.T.U. 2011]

Sol. Fig. gives the timing diagram of SHLD address instruction. This instruction stores the contents of L register in the memory location given within the instruction and contents of H register at address next to it. It requires the following five machine cycles.

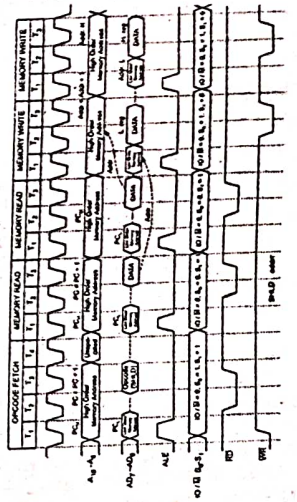


Fig.

1. **Opcode fetch :** Program counter places address on low order and high order address bus. Microprocessor reads the opcode of SHLD from this memory location and decodes it. Program counter is incremented by one.

2. **Memory read :** Program counter gives address on low order and high order address bus. Microprocessor reads data from the addressed memory location. This data is the low order byte of the address specified within the instruction. Program counter is incremented by one.

3. **Memory read :** Program counter gives address on low order and high order address bus. Microprocessor reads data from the addressed memory location. This data is the high order byte of the address specified within the instruction.

4. **Memory write :** The data read in the previous two memory read cycle is placed on the address bus. Microprocessor writes the contents of L register at this memory address. This memory address is incremented by one.

5. **Memory write :** Now the incremented address is present on the address bus. Microprocessor writes the contents of H register at this memory address.

PART-C

Prob.19 Explain the operational difference between following pair of instructions :

(a) SPHL and XTHL

(b) LHLD and SHLD Addr

(c) XRA A and MVI A, OOH

(d) DAD RP and DAA

(e) INR A and ADI01 H **[R.T.U. 2019]**

Sol. (a)SPHL: This instruction copies H and L register to the stack pointer. The instruction loads the contents of the H and L registers into the stack pointer register, the contents of the H register provide the high-order address and the contents of the L register provide the low-order address. The contents of the H and L registers are not altered.

XTHL: This instruction exchanges H and L with top of stack.The contents of the L register are exchanged with the stack location pointed out by the contents of the stack pointer register. The contents of the H register are exchanged with the next stack location (SP+1); however, the contents of the stack pointer register are not altered.

(b) "LHLD" Instruction

- LHLD stands for load H and L registers direct .
- This instruction copies the contents of the memory location pointed by the 16-bit address in register L and copies the contents of the next memory location in register H. The contents of source memory locations are not altered.
- **Flags:** The execution of LHLD instruction can not affect the flags.

"SHLD" Instruction

- SHLD stands for store H and L registers direct.
- The contents of register L are stored in the memory location specified by the 16-bit address in the operand and the contents of H registers are stored in the next memory location by incrementing the operand.
- Remember, the contents of the HL pair register are not altered.
- This is also a 3-byte instruction. The second byte specifies the low-order address and the third byte specifies the high-order address.
- **Flags:** No flags are affected.

(c) XRA A : The XRA A instruction is used to clear the contents of the Accumulator and store the value 00H.

MVI A : Move immediate date to a Accumulator.

OOH : High byte of index address.

(d) DAD RP : DAD is a mnemonic, which stands for Double ADD and also RP stands for any one of the following register pairs as mentioned below :

RP = BC, DE, or HL

As RP can have any of the three values, there are three opcodes for this type of instruction. It occupies only 1-byte in memory.

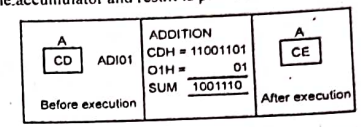
DAA : The DAA (Decimal Adjust after Addition) instruction allows addition of numbers represented in 8-bit packed BCD code. It is used immediately after normal addition instruction operating on BCD codes. This instruction assumes the AL register as the source and the destination and hence it requires no operand.

The effect of DAS (Decimal Adjust after Subtraction) instruction is similar to that of DAA, except that it is used after a subtraction instruction.

(e) INR A : INR A (the content of register A(Accumulator) is incremented by 1.The contents of the designated register

or memory) are incremented by 1 and the result is stored in the same place. If the operand is a memory location, its location is specified by the contents of the HL registers.

ADI01 H : The 2 bytes data O1H is added to the content of the accumulator and result is placed in the accumulator.



Prob.20 ***Explain :***

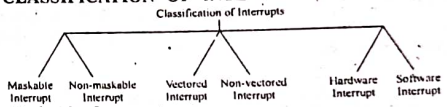
(a) Interrupts of 8085.

(b) RST instructions of 8085.

(c) Maskable and non-maskable interrupts and related data instruction to mask or unmask interrupt. **[R.T.U. 2017]**

Sol.(a) Interrupts of 8085 : *Refer to Prob.1.*

CLASSIFICATION OF INTERRUPTS



Maskable Interrupt

8085 has four maskable interrupts :

(1) RST 7.5
(2) RST 6.5
(3) RST 5.5
(4) INTR

It is the property of the microprocessor that it can provide some delay to a maskable type of interrupt if it is performing some execution or some critical task, but it has to accept the request made by a non maskable interrupt immediately. This interrupt is dependent upon EI and DI.

Non-maskable Interrupt

This interrupt can not be disabled and therefore, need not to be enabled. TRAP is a non-maskable interrupt. It has the highest priority among all the interrupt signals. It is level and edge sensitive to the interrupt signal. It is a vectored interrupt that when this interrupt is triggered, the program execution is transferred to a predefined call location, 0024H. Unlike INTR, the TRAP does not require external hardware.

Vectored Interrupt

Those interrupts whose vector locations can be identified are called vectored interrupts, microprocessor 8085 consists of four vectored interrupts which are :

(1) TRAP (RST 4.5) (2) RST 7.5
(3) RST 6.5 (4) RST 5.5

For the case of vectored interrupts the hardware automatically transfers the control of the program to a specific memory location.

The location of the vectored interrupt can be known by the following procedure :

(1) TRAP (RST 4.5)

$4.5 \times 8 = 36$ Convert into Hexadecimal

$\therefore$ RST $4.5 = (0024)_{16}$

(2) RST 7.5

$7.5 \times 8 = 60$ Convert into Hexadecimal

$\therefore$ RST $7.5 = (003C)_{16}$

(3) RST 6.5

$6.5 \times 8 = 52$ Convert into Hexadecimal

$\therefore$ RST $6.5 = (0034)_{16}$

(4) RST 5.5

$5.5 \times 8 = 44$ Convert into Hexadecimal

$\therefore$ RST $5.5 = (002C)_{16}$

$\therefore$ For all the four vectored interrupts 00 is called the page number

Page number $\rightarrow$ 00 24_{16} $\rightarrow$ Line number

Page number $\rightarrow$ 00 $3C_{16}$ $\rightarrow$ Line number

Page number $\rightarrow$ 00 34_{16} $\rightarrow$ Line number

Page number $\rightarrow$ 00 $2C_{16}$ $\rightarrow$ Line number

All these vectored interrupts do not require any external hardware since their locations can be determined.

Non-Vectored Interrupt

Microprocessor 8085 consists of one non vectored interrupt which is called INTR. Since it is a non vectored interrupt so external hardware is required to generate the RST code.

8085 Hardware Interrupts : The 8085 microprocessor has 5 hardware interrupts as given in table.

Table

Interrupt	Vector address in Hex
TRAP	0024
RST 5.5	002C
RST 6.5	0034
RST 7.5	003C
INTR	

The four interrupts TRAP, RST 5.5, RST 6.5 and RST 7.5 are directly vectored to the address specified in the interrupt vector table. These interrupts are also called RST hardware interrupts. The fifth interrupt INTR (interrupt request), has no specified vector address. When INTR interrupt is activated, the microprocessor issues an interrupt acknowledge signal INTA. The features and operation of hardware interrupts are explained below :

1. **Interrupt TRAP :** The interrupt TRAP has the highest priority. It is non maskable interrupt and need not be enabled. It can not be disabled. It is level and edge sensitive. In this interrupt, the interrupt service routine starts at vector location address 0024. It is used in a situation such as power failure and emergency shut off. The operation of the interrupt TRAP is as follows :

(i) The peripheral device sends an interrupt request signal to the TRAP input of the microprocessor.

(ii) The signal sets TRAP flip-flop and activates TRAP request. The microprocessor samples the TRAP request at the end of the current instruction cycle, when it finds the TRAP request high, it completes the current instruction cycle.

(iii) The microprocessor executes the TRAP instruction internally. It also activates internal TRAP acknowledge signal to reset the TRAP flip flop. It also activates any interrupt acknowledge signal to reset INTE flip-flop so that all maskable interrupts are disabled.

(iy) The microprocessor saves the contents of the program counter on to the stack. It then transfer the control to the TRAP interrupt service routine 0024 H.

2. **Interrupt RST 7.5 :** RST 7.5 interrupt has the second highest priority. It is maskable and positive edge triggered interrupt. The RST 7.5 interrupt request is stored internally by D flip-flop. The flip-flop is reset by either RST 7.5 bit of SIM, RESET IN signal or internal RST 7.5 acknowledge signal. Vector location address of interrupt RST 7.5 is 003C.

The operation of this interrupt is as follows :

(i) The peripheral devices sends an interrupt request signal to the RST 7.5 pin on the microprocessor.

(ii) This signal set RST 7.5 flip and activates internal RST 7.5 signal.

(iii) The microprocessor samples the RST 7.5 request at the end of the current machine cycle. When it finds this request high, it completes the current instruction cycle.

(iv) The microprocessor executes the RST 7.5 instruction to generate string address of the interrupt service routine.

(v) The microprocessor saves the contents of the program counter on the stack.

(vi) The microprocessor transfers the control to the RST 7.5 address which is 003C H.

3. **Interrupt RST 6.5 and RST 5.5 :** These are maskable interrupts. These interrupts has the lower priority than TRAP and RST 7.5. Both these interrupts are level sensitive. They can be enabled by SIM instructions. Vector address for RST 6.5 is 0034 H and RST 5.5 is 002C H. The operation of these interrupts is as follows:

(i) The peripheral device sends an interrupt request signal to the RST 6.5 or RST 5.5 pin of the microprocessor.

(ii) This signal enables the AND gate and activate the RST 6.5 or RST 5.5 request.

(iii) The microprocessor samples the request at the end of current instruction cycle. When it finds the request high, it completes the current instruction cycles.

(iv) The microprocessor executes the RST 6.5 or RST 5.5 instruction internally. It also activates any interrupt acknowledge signal to reset the INTE flip-flop so that all maskable interrupts are disabled.

(v) The microprocessor saves the contents of the program counter on the stack.

(vi) The microprocessor transfer its control to the RST 5.5 with vector address 002CH or RST 6.5 with vector address 0034 H.

4. **Interrupt INTR :** INTR interrupt is maskable and it has lowest priority. During the execution of each instruction, the microprocessor check INTR. If it is high, then the microprocessor completes its current operation and sends an interrupt acknowledge signal $\overline{INTA}$ active low. The operation of the interrupt INTR is as follows :

(i) These peripheral devices send an interrupt request signal to the INTR pin of the microprocessor.

(ii) This signal enables AND gate and activates internal INTR request.

(iii) In response to INTR request, the microprocessor, complete current instruction cycle and executes interrupt acknowledge cycles for RST instruction or CALL instruction.

(iv) During the interrupt acknowledge cycle, the microprocessor activates an interrupt acknowledge signal to reset INTE flip-flop. During this cycle, microprocessor reads either RST N or CALL instruction from the interrupting device.

(v) The microprocessor saves the contents of program counter on the stack.

(vi) The microprocessor transfers its control to the appropriate interrupt service routine (ISR).

Software Interrupt

When the normal operation of microprocessor is interrupted by some special instructions they are called software interrupts. There are 8 RSTn instructions which are used as software instruction for microprocessor 8085.

The RSTn can be used in the program to find out or debug some type of errors by the break point technique method. All the RSTn instructions are 1 byte call instructions given by :

RSTn

RST 0 $0 \times 8 = 0000H$

RST 1 $0001 \times 8 = 0008$ H

RST 2 $0002 \times 8 = 0016$ Convert into Hexadecimal
$= 0010H$

RST 3 $.0003 \times 8 = 0024$ Convert into Hexadecimal
$= 0018_{16}$

RST 4 $0004 \times 8 = 0032$ Convert into Hexadecimal
$= 0020_{16}$

RST 5 $0005 \times 8 = 0040$ Convert into Hexadecimal
$= 0028_{16}$

RST 6 $0006 \times 8 = 0048$ Convert into Hexadecimal
$= 0030_{16}$

RST 7 $0007 \times 8 = 0056$ Convert into Hexadecimal
$= 0038_{16}$

All RSTn instructions of 8085 are used as software instructions which when inserted, the programs executes up to that address. These RSTn instructions can also be used for debugging the program.

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

(a) Software interrupts

(b) Hardware interrupts

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

Table 1: Software interrupts

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

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

Sol.(c) Maskable and Non-maskable Interrupts : *Refer to Prob.20(a).*

Related Data Instruction : Some data instruction to represent mask or unmask interrupt:

Trap : It is non maskable edge and level triggered interrupt. TRAP has the highest priority and vectores interrupt. Edge and level triggered means that the TRAP must go high and remain high until it is acknowledged. In case of sudden power failure, it executes a ISR and send the data from main memory to backup memory.

As we know that TRAP can not be masked but it can be delayed using HOLD signal. This interrupt transfers the microprocessor's control to location 0024H.

RST7.5 : It has the second highest priority. It is maskable and edge level triggered interrupt. The vector address of this interrupt is 003CH. Edge sensitive means input goes high and no need to maintain high state until it is recognized.

RST6.5 and RST5.5 : These are level triggered and maskable interrupts. When RST6.5 pin is at logic 1, INTE flip-flop is set. RST 6.5 has third highest priority and RST 5.5 has fourth highest priority.

INTR : It is level triggered and maskable interrupt. The following sequence of events occurs when INTR signal goes high:

1. The 8085 checks the status of INTR signal during execution of each instruction.
2. If INTR signal is high, then 8085 complete its current instruction and sends active low interrupt acknowledge signal, if the interrupt is enabled.
3. On receiving the instruction, the 8085 save the address of next instruction on stack and execute received instruction.

Example

```
Compel MACRO     Address
LXI              H
MOV              A, M
CMA
ENDM
```

In this above example Compel is name of macro. Macro is written in beginning of the definition as shown in example and address is parameter. ENDM is used to end of the macro if we write compel 2500 H in assembly language program, assembler will replace this macro by following sequence of instruction in program.

```
LXI     H, 2500 H
MOV
CMA
```

This sequence of instruction will take the complement of the contents of memory location 2500 H.

Example

```
Shift MACRO
ADD A
END M
```

Shift is name of macro. If shift is written program, the assembler will replace it by ADD A. This instruction adds the contents of accumulator to itself. This addition result in logical shifting of contents of accumulation left by one bit.

Once a sequence of instruction is written and maci name is assigned to it, the macro name can be frequent used in program. Its name in program is easier to read an understand. Marco and subroutine are similar.

A subroutine requires CALL and RETURN instructic whereas macro do not. Macro execute faster than subroutin Macros are used for short sequence of instruction where subroutine for larger ones, preferably for 10 instructions and mon

Subroutine : *Refer to Prob.7.*

S.No.	Subroutine	Macro
1.	Can be called by using CALL instruction.	Can be called by writing only the name of the Macro in the program as an instruction.
2.	Subroutines physically exist in the memory.	Macros exist only until your code is compiled, after compilation all Macros are replaced with real instructions. If you declared a Macro and never used it in your code compiler will simply ignore it
3.	Program execution is transferred to the memory location specified in the CALL instruction.	Macro name in the program is replaced by instruction used in Macro definition then compiled.
4.	It is a feature of programming.	It is a feature available in any assembler.

□□□

8085 Microprocessor Interfacing 4

Chapter in a Nutshell

□ **8255A Programmable Peripheral Interface**

8255A, also known as PPI is widely used, programmable, parallel I/O device. It is an important general purpose I/O device that can be used with almost any microprocessor. It can be programmed to transfer data under various conditions, from simple I/O to interrupt I/O. It is economical and flexible and can be used when multiple I/O ports are required.

□ **8254 Programmable Interval Timer**

8254 is popular programmable interval timer chip. Intel 8254 contains three identical 16–bit down counters that can operate independently in any one of the six modes. A 16–bit count is loaded in register and control word is loaded in the control word register. Count is decremented until it reaches zero. When count is terminated, a pulse is generated that can be used to interrupt the microprocessor. Counter can operate in either binary or BCD. 8254 includes status read–back command which helps in reading count by microprocessor while the counter is decrementing.

8254 is an upgraded version of 8253, and both of these are pin–compatible.

□ **I/O Interfacing**

I/O devices (peripherals) like keyboard, switches, converters, CRT, printers, LEDs etc. are communicated with microprocessor using interfacing methods. In the 8085 microprocessor based system, I/O devices can be interfaced using both techniques : **memory–mapped I/O and peripheral–mapped I/O.** The process of data transfer is identical in both.

Previous Years Questions

Part-A

Prob.1 *What is the need of DMA in microprocessor?*

[R.T.U. 2019, 10]

Sol. DMA : The bulk data transfer from fast I/O devices to the memory or from the memory to I/O devices through the accumulator, is a time consuming process. For such a situation, the direct memory access (DMA)technique is preferred. In DMA data transfer scheme, data are directly transferred from I /O devices to RAM or from RAM to I/O devices. For DMA data transfer, the data and address buses come under the control of the peripheral device which wants DMA transfer. The microprocessor has to retain the control of the address and data buses for DMA data operation on the request of the I/O device. For DMA data transfer, the I/O device must have its own registers to store by the count and memory address. It must also be able to generate control signal required for DMA data transfer.

Prob.2 *Write a control word of 8255 in IO mode 0, for port A and port B is input and port C is in output port.*

[R.T.U. 2016]

Sol. The ports A, B and C can be configured as simple input or output ports by writing the appropriate control word in the control word register. In the control word, D_7 is set to 1 (to define a mode set operation) and D_6, D_5 and D_2 are all set to 0 to configure all the ports in mode 0 operation. The status of bits D_4, D_3, D_1, and D_0 then determine whether the corresponding ports are to be configured as Input or Output.

In port A and port B are to operate as input ports with port C lower and upper as output, the control word that will have to be loaded into the control register will be as follows

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	1	0

Prob.3 *Find the address of port A, port B, port C and control register of 8255 for following interfacing.*

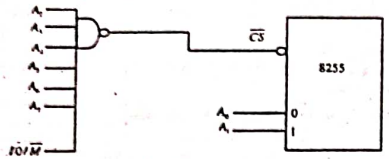


Fig. [R.T.U. 2016]

Sol. I/O Map

Ports/Control register	Address lines								Address
	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
Port A	0	0	0	0	0	0	0	0	00H
Port B	0	0	0	0	0	0	0	1	01H
Port C	0	0	0	0	0	0	1	0	02H
Control register	0	0	0	0	0	0	1	1	03H

Prob.4 *Design a square wave generator with a pulse width of 100 µs by using 8254 timer. Set the timer in mode 3. The clock frequency is 3 MHz.* [R.T.U. 2012, 2008]

Sol. Frequency of the clock = 3MHz

$$\therefore \quad \text{Time} = \frac{1}{3\times10^6} = 0.33\,\mu\text{sec}$$

$$\text{Count} = \frac{100\times10^{-6}}{0.33\times10^{-6}} = \frac{100}{33}\times100 = \frac{10000}{3} = 303.3 = 0130\text{H}$$

MVI A, 01110110. B	Control word for mode 3 and count 2
OUT 83H	Written 8254 control Reg.
MVI A,30 H.	Lower order byte of count.
OUT 80(H)	Load counter 1 with low order byte
MVI A, 01H.	High order byte of the count
OUT 81(H)	Load counter 1 with high order byte
HLT	End of the programme.

PART-B

Prob.5 *Explain various modes supported in 8254 Timer in detail.* [R.T.U. 2019]

Sol. Operation Modes of 8254 : The 8254 can operate in six different modes :

Mode 0: Interrupt on Terminal Count

In this mode, initially the OUT is low. Once a 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 = 5) is temporarily stopped when the gate is disabled (G = 0) and continued again when the gate is at logic1.

Mode 1: Hardware-Retriggerable One-Shot

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 (Figure, Mode 1).

Mode 2: Rate Generator

This mode is used to generate a 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

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 words, 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.

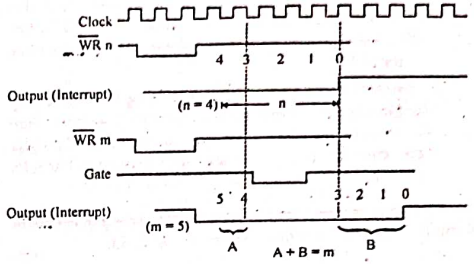
Mode 4 : Software-Triggered Strobe

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.

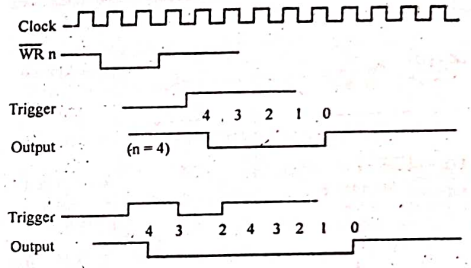
Mode 5 : Hardware-Triggered Strobe

This mode is similar to Mode 4, except that it is triggered by the rising pulse at the gate. Initially, the OUT is low 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.

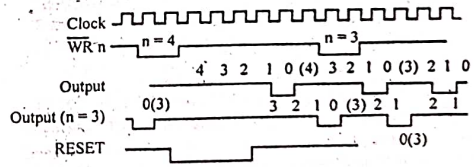
Mode 0 : Interrupt on Terminal Count



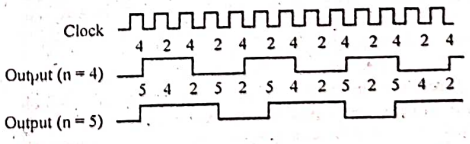
Mode 1 : Retriggerable One-Shot



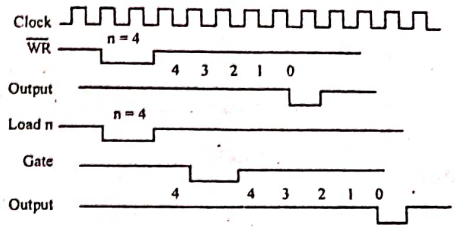
Mode 2 : Rate Generator Clock



Mode 3 : Square Wave Generator



Mode 4 : Software Triggered Strobe



Mode 5 : Hardware Triggered Strobe

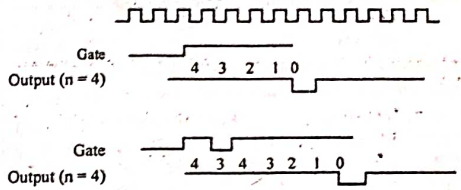


Fig. : Six Modes of 8254

Prob.6 *Write and explain control word in 8254 Timer in detail.* [R.T.U. 2019]

Sol. Control-Word of 8254 PIT : The register selected when line A_0 and A_1 are at logic 1 ($A_0 A_1 = 11$). It is used to write a command word which specifies the counter to be used. Its mode is either a read or a write operation. The control word format and mean of particular bit is shown :

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC_1	SC_0	RW_1	RW_0	M_2	M_1	M_0	BCD

Select Counter (SC)

SC_1	SC_0	
0	0	Select counter 0
0	1	Select counter 1
1	0	Select counter 2
1	1	Read back command

RW (Read Write)

RW_1	RW_0	
0	0	Counter latch command
0	1	Read / Write least significant bit only
1	0	Read / Write most significant bit only
1	1	Read / Write least significant bit first than most significant bit

M–Mode

M_2	M_1	M_0	–	
0	0	0	–	Mode 0
0	0	1	–	Mode 1
×	1	0	–	Mode 2
×	1	1	–	Mode 3
1	0	0	–	Mode 4
1	0	1	–	Mode 5

BCD

0 – Binary counter 16 bit

1 – Binary coded decimal (4 decades)

Prob.7 ***Differentiate memory mapped and IO mapped techniques.*** [R.T.U. 2016]

Sol. Comparison of I/O Mapped and Memory Mapped :

In systems using microprocessors which allow I/O Mapped I/O, the decision to go for one type of I/O addressing or the other is usually a matter of convenience. For example, suppose one is using a decoder like the 8205 to decode chip selects (CS) for some memory modules. If after providing the required chip selects, one discovers that some 8205 decoded outputs are still unused then these will generally be used to select I/O ports. These ports will, therefore, be memory mapped. Apart from such considerations, there are some pros and cons to these I/O addressing techniques. These are described below :

(i) **Using Memory Mapped I/O :** We can typically address more I/O ports than when using I/O Mapped I/O. This is because in the latter case only a limited number of the address lines generally carry useful address information. For example, the 8085A can address 64 K memory locations but can only address 256 I/O ports. (Of course, 256 I/O ports is much more than what is usually needed. This is at the most a very dubious advantage in favour of memory mapped I/O.

(ii) **Using I/O Mapped I/O :** We can distinguish between separate I/O and memory address spaces. Conceptually, that is sometimes an advantage in favour of I/O mapped I/O. On the other hand the decoding used for memory selection (i.e. $\overline{CSs}$) may be such that memory mapped I/O is easier to implement. This advantage was shown by the example given earlier.

(iii) It is easier to implement Direct Memory Access (DMA), an I/O technique described later-when I/O mapped I/O is being used. This feature will be described later.

(iv) The variety of instructions available for I/O mapped I/O is generally very limited (typically just IN for input and OUT for output). On the contrary one can use all memory access instructions available in the microprocessor's instruction repertone' for memory mapped I/O. Generally, a large number of such instruction using different addressing modes are available which apart from simple input and output also allow arithmetic and logical manipulation of I/O port data. This is a very important point in favour of memory mapped I/O.

(v) The instruction controlling I/O mapped I/O generally require fewer program memory bytes to encode than the instructions for memory mapped I/O and usually also execute faster. This makes I/O mapped I/O usually more efficient than memory mapped I/O.

Prob.8 ***Draw and explain diagram of 8253.*** [R.T.U. 2015]

Sol.

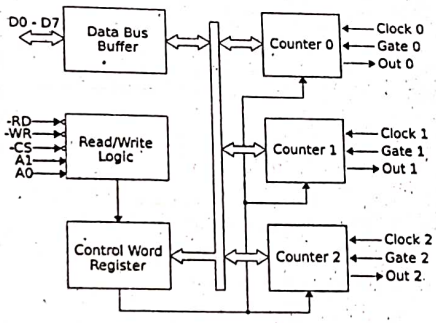


Fig. : Block Diagram of Intel 8253

The Intel 8253 is a Programmable Interval Timer (PIT) chip used for timing and counting functions. It was designed for use in Intel 8080/8085 microprocessors. It is packaged in a 24-pin plastic DIP. It has 6 programmable timer modes which allows it to be used as an event counter, elapsed time indicator, programmable one- shot and in various other application to create different intervals.

It has three counters. We can see in the above block diagram that each counter is connected to 3 lines out of which clock and gate are input lines whereas out is an output line. These lines have functions which depends on the way the chip is programmed. The eight control bits ($D_0 - D_7$) controls

the behavior of the device. Last two MSB bits tells the counter to be used. The bits D_4, D_5 determines how the count value is to be read and write. The bits D_1, D_2, D_3 selects between the 6 count modes. Bit D_0 tells the device the way in which to count, i.e, in binary or BCD.

Modes of Operation of 8253 :

Mode 0 : Interrupt on Terminal Count - In this mode, the output has initial value of zero. The count is loaded and the timer counts down until it reaches 0. When it reaches 0, the output will be set to high or 1 and will remain same until the count is reloaded. This mode is used for the generation of accurate time delay under software control.

Mode 1 : Programmable One Shot - In this mode, Gate input acts as a trigger. The output is initially 1 and it is set to 0 after the gate is triggered. The count will count down to zero and once the count is zero, the output is again set to 1 and will remain as such until gate is triggered again. It can be used as Monostable Multivibrator.

Mode 2 : Rate Generator - In this mode, the device acts as a divide-by-n counter, which is commonly used to generate a real-time clock interrupt. The output is initially set to high after a count is sent. It will count down to zero and the output remains high until count reaches 1 at which point output is set to 0 for one pulse and will remain so until next count is sent.

Mode 3 : Square Wave Generator - This mode is similar to mode 2 except for the output in this case remains low for half of the timer period and high for the other half period.

Mode 4 : Software Triggered Pulse - In this mode, the output will remain high until the timer has counted to zero, at which point the output is set to low for one clock cycle and after that the output will go high again.

Mode 5 : Hardware Triggered Pulse - This mode is similar to mode 4 except for the counting process is triggered by the gate input. The counter only starts counting once the gate input goes high.

Prob.9 ***Design a square wave 'generator' with a pulse width of 150 μs. by using 8254. Set the timer in mode 3. The clock frequency is 2MHz.*** [R.T.U. 2013]

Sol. We assume that 80H is address of counter-0, 81 is address of counter-1, 82 H is address of counter-2 and 83H is address of control word register.

To generate a square wave, 8254 should be in mode 3 and counter-0 will be operating in BCD mode. The count value can be calculated for 150 μsec.

8254 operates at 2MHz. (given)

The time period is $\frac{1}{F} = \frac{1}{2} = 0.5\mu s$

The number of T states required is

$$N = \frac{150\times10^{-6}}{0.5\times10^{-6}} = 300 \text{ states}$$

The control word of 8254 is 37H as shown below :

SC_1	SC_0	RL_1	RL_0	M_2	M_1	M_0	BCD
0	0	1	1	0	1	1	1

The following instruction will be executed to generate a square wave.

Levels	Instructions	Comments
START	MOV A, 37H	Initialize 8254 counter with control word 37 H
	OUT 83	Load control word into control word register.
	MVI A, 00H	counter-0 operates in mode 3. Write 00H in Accumulator
	OUT80	LSB of count value is 00H which is loaded in the counter
	MVI A, 20H	Write 20H in Accumulator
	OUT 80, AL	MSB of count value is 20H which is loaded in the counter
	HLT	Halt

Prob.10 ***An 8255 is interfaced in memory mapped I/O. So that its address range is 8000 H to 8003 H. Frame the control word for the following configuration :***

Port A : Input in Mode 0
Port B : Output in Mode 0
Port Cu : Input
Port Cl : Output

Write instructions to initialize the 8255. [R.T.U. 2012]

Sol. 1. Port Addresses : This is a memory mapped I/O; when the address line A_{15} is high the chip select line is enabled

Port A = 8000 H
Port B = 8001 H
Port C = 8002 H
Control Register = 8003 H

2. Control Word

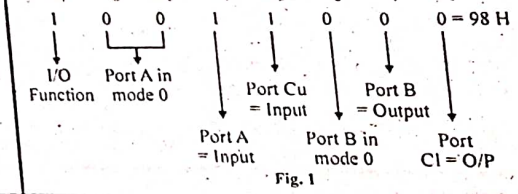


Fig. 1

3. Program

MVI A, 98 H	:	Load accumulator with the control word.
STA 8003 H	:	Write word in control register to initialize the port.
LDA 8000 H	:	Read switches at port A.
STA 8001 H	:	Display the reading at Port B.
LDA 8002 H	:	Read switches at port C.
ANI OF H	:	Mask the upper four bit of port C, these bits are not input data.
RLC	:	Rotate and place data in the upper half of the accumulator.
RLC		
RLC		
RLC		
STA 8002 H	:	Display data of port Cu
HLT		

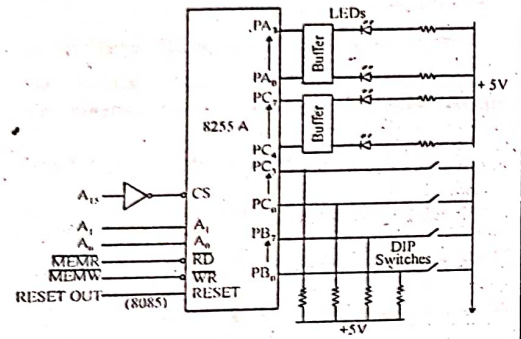


Fig. 2

Prob.11 ***Draw the functional block diagram of 8254 IC and explain how this can be used to obtain a signal having frequency equal to 1/6 of input clock signal.*** [R.T.U. 2011]

Sol.

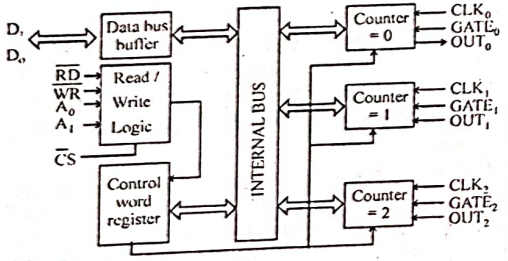


Fig. : Block diagram of 8253/54

To obtain a signal having frequency equal to 1/6 of input clock signal Mode 2 (Rate generator) is used. The counter that is programmed for mode 2 becomes a "divide by n" counter. The out pin of the counter goes to low for one input clock period. The time between the pulse going low is dependent on the present count in the counter's register, *i.e.*, the time of the logical pulse. For example, to get an output frequency of 1000 Hz, the period would be 1/1000 s = 1ms or 1000 µs. If an input clock of 1MHz was applied to the clock input of the counter #0, then the counter #0 would need to be programmed to 1000 µs. This could be done in decimal or in BCD. The formula is

$$N = \frac{F_{clk}}{F_{out}}$$

Where,

N = Value to be loaded

F_{clk} = Input clock frequency

F_{out} = Output frequency

Thus, in our case

$$F_{out} = \frac{1}{6} F_{clk}$$

$$F_{clk} = 6\, F_{out}$$

Putting values, we get

$$N = \frac{6F_{out}}{F_{out}}$$

$$\boxed{N = 6}$$

To get an output frequency equal to $\frac{1}{6}$ of input clock frequency, the period would be 1/(1/6) s or 6 s.

PART-C

Prob.12 ***Explain the organization and architecture of 8255 Programmable Peripheral Interface IC with a function block diagram also draw the interfacing scheme of 8255 and 8085 in memory mapped I/O mode.*** [R.T.U. 2019]

Sol. Block Diagram of 8255 : The 8255 is widely used programmable peripheral interface *i.e.* PPI 8255. It is a general purpose programmable I/O device.

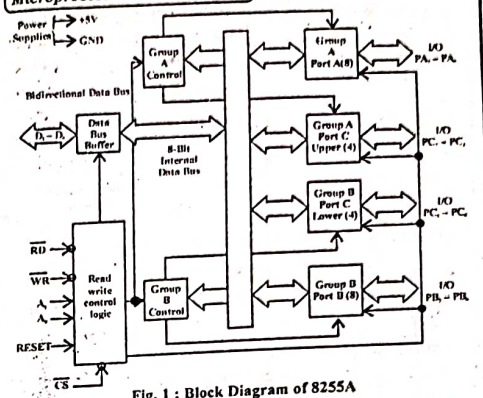


Fig. 1 : Block Diagram of 8255A

The Block diagram of 8255 is shown in above fig.1. It shows:

(i) Data bus buffer

(ii) Read / Write logic control

(iii) Group A and Group B control

(iv) Port A and B

(v) Port C

Control Lines and I/O Port Address of 8255

(i) **$\overline{RD}$:** The $\overline{RD}$ (read) signal enable the read operation. When this signal goes low, the microprocessor reads data from the selected I/O port of 8255.

(ii) **WR :** The WR (write) signal enables the writes operation. When this signal is low, the microprocessor writes into a selected I/O port of control register.

(iii) **RESET :** When the reset signal is high, it clears the control register and sets all ports in the input mode.

(iv) **CS, A_0 and A_1 :** These are the device select signals. CS is the master chip select and A_0 and A_1 specify one of the I/O port or the control register.

Operation mode of 8255 : Control word is written in the control register of 8255. The control word written in the control register, specify the I/O function for each port. The control register can be accessed to write a control word when address lines A_1 and A_0 are at logic 1.

In the control word, bit D_7 is 1, then bit $D_6 - D_0$ determine the input or output function of various ports and the mode of operation. If D_7 bit is 0 then port C operates in Bit Set/Reset mode. In the BSR mode, functions of port A and port B are not affected.

The control word of 8255 is divided into two different control word formats which designate two basic modes :

(1) BSR mode (when $D_7 = 0$)

(2) I/O mode (when $D_7 = 1$)

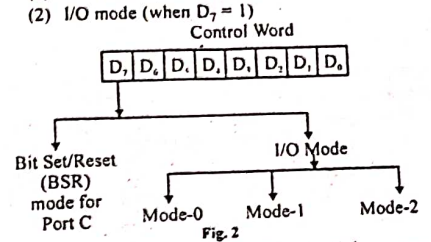


Fig. 2

(1) BSR Mode : The bit Set/Reset BSR is concerned with eight bits of port C only. The individual bit of port C can be set or reset by writing an appropriate control word in the control register. BSR bit does not used after any previously transmitted control word with bit $D_7 = 1$.

In BSR mode, individual bit of port C can be used for applications such as an on/off switch.

Control word format in the BSR mode is given in figure 3.

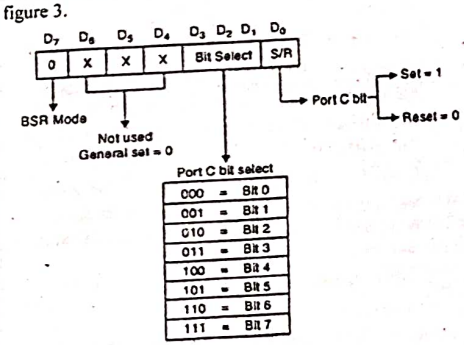


Fig. 3 : Control word format in BSR mode

(2) I/O Mode : A control word with bit $D_7 = 1$ is recognized as a I/O mode, and it is classified into three different I/O modes.

(i) Mode-0 (Simple input or output) : This mode is also called basic I/O mode. Port A, B, C are used as simple I/O ports in this mode.

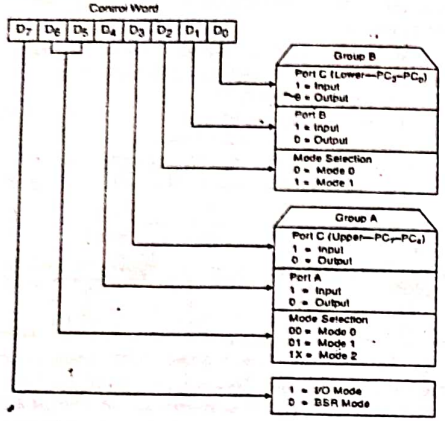


Fig. 4 : 8255 control word format for I/O mode

It has following features:

(a) Two 8-bit ports (Port A and port B) and two 4-bit ports (Port C_{Upper} and C_{Lower}). Port C can be used as 8-bit port.

(b) Any port can be used either as I/O or O/P port.

(c) Output ports are latched and input ports are not latched.

(d) A maximum of four ports available which is to say that overall 16 I/O configurations are possible.

(ii) Mode-1 (Input or Output with Handshake): This mode is also called as strobed I/O mode. In this mode, handshake signals are exchanged between the microprocessor and peripherals prior to data transfer.

It has following two different modes.

(A) Mode-1 (Input Control Signal)

(B) Mode-1 (Output Control Signal)

(A) Mode-1 (Input Control Signal) : When port A and B are configured as input port, the associated control signals used for handshaking are shown in fig.5.

Port A uses the upper three signals: PC_3, PC_4 and PC_5. Port B uses lower three signals : PC_2, PC_1 and PC_0 for handshaking. PC_6 and PC_7 which can be used as input or output as programmed by bit D_3 of the control word. The functions of these control signals are given below:

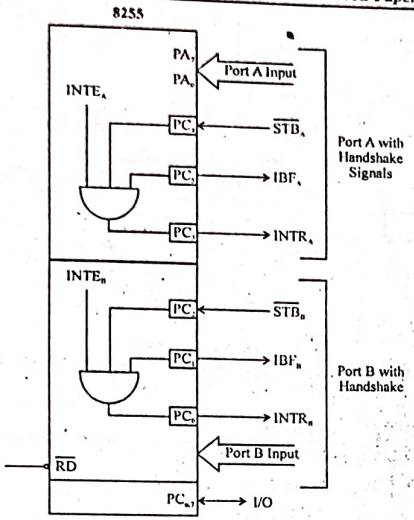


Fig. 5 : 8255 Mode 1 : Input Configuration

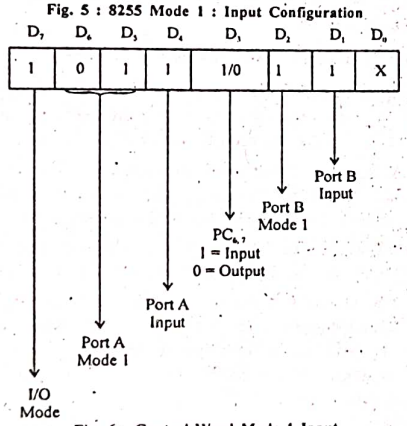


Fig. 6 : Control Word Mode 1 Input

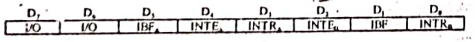


Fig. 7 : Status Word Mode 1 Input

$\overline{STB}$ (Strobe Input) : This is an active low singal. This is generated by peripheral device, when transmitting a data byte.

IBF (Input Buffer Full) : This is active high acknowledgement signal generated by 8255 to indicate that input latch has received the data byte.

INTR (Interrupt Request) : This is active high output signal, used to interrupt microprocessor. This signal is generated if $\overline{STB}$, IBF and INTE are all at logic 1 and this is reset by the falling edge of the $\overline{RD}$ signal.

INTE (Interrupt Enable) : This signal is used to generate INTR signal. Two flip flop $INTE_A$ and $INTE_B$ are used for this purpose.

Control and Status Words : The status word is accessed by reading port C i.e. when $A_1A_0 = 10$ and $\overline{RD}$ and $\overline{CS}$ both are low.

(B) Mode-1 (Output Control Signal) : When the port A and B configured as output port. Port A uses three signals: PC_3, PC_6 and PC_7. Port B uses another three signals: PC_0, PC_1 and PC_2 for handshaking signals. Leaving PC_4 and PC_5 which can be used as input or output as programmed by bit D_3 of control word.

$\overline{OBF}$ (Output Buffer Full) : This is an active low output signal. It goes low when microprocessor write data into output latch of 8255.

$\overline{ACK}$ (Acknowledge) : It is active low signal and indicate that output peripheral has taken the data from output buffer..

INTR (Interrupte Request) : This is active high output signal. INTR is set when $\overline{OBF}$, $\overline{ACK}$ and INTE are all one and reset by falling edge of $\overline{WR}$.

INTE (Interrupt Enable) : This is used for enable or disable the INTR signal.

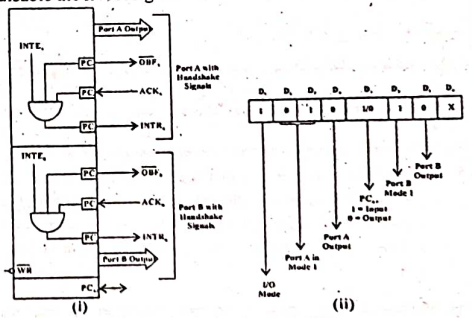


Fig. 8 : 8255 Mode 1 : Output Configuration

Control and Status Words : When Port A and B are configured as output ports in the handshake, mode 1, the control word which will be required to be loaded into the control register.

(iii) Mode-2 (Bidirectional Data Transfer) : This is also called as strobed bidirectional I/O mode. This mode operation is only for port A. In mode-2, port A uses five signals from port C as handshaking signals for data transfer, and the remaining three signals as simple I/O or as handshake for port B.

(A) Mode-2 (Input Control Signals) :

$\overline{STB_A}$ (Strobe) : This is an active low input signal.

IBF_A (Input Buffer Full) : When data is available in input buffer, 8255 will enable IBF_A signal for peripheral devices.

(B) Mode-2 (Output Control Signal) :

$\overline{OBF_A}$: This indicate that data available in output buffer.

$\overline{ACK_A}$: This is active low signal, indicate that peripheral reads data from port A and acknowledges the reading.

INTR (Interrupt request) : This is an output signal given by 8255 to request MPV service.

(C) Mode-2 (Status Word) : The status of the I/O operation in Mode-2 can be varified by reading the contents of port C.

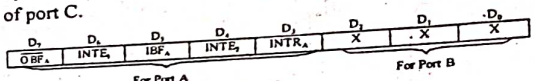


Fig. 9

Interfacing 8255 with 8085

Description

Step 1:

- Lower order of 8-bit address A0-A7 is separated from AD0-AD7 using address latch/buffer (Ex: IC 74373) and ALE signal.
- The separated address lines A_0-A_7 are connected to A_0-A_7 input pins of 8255 and the separated data bus D_0-D_7 are connected to D_0-D_7 pins of 8255.
- Reset out of 8085 is connected to reset pin of 8255.

Step 2:

- 8255 does not have internal (separate) control logic generator, hence the IO/M(bar), RD(bar) and WR(bar) control signals are not connected directly to 8255. These pins are 1st given to decoder and decoded using 3:8 decoder (Ex: IC 74138).
- The generated control signals IOR(bar) and IOW(bar) are connected to RD(bar) and WR(bar) input of 8155.

Step 3:

- An active low signal of chip select logic is obtained decoding remaining address lines of lower order addresses A_2-A_7.
- Chip select logic and IO port address for this interfacing circuit are as:

Chips select address lines	Address lines to select port	HEX address	Selected I/O						
A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0		
1	0	0	0	0	0	0	0	80H	PORT A
1	0	0	0	0	0	0	1	81H	PORT B
1	0	0	0	0	0	1	0	82H	PORT C
1	0	0	0	0	0	1	1	83H	Chip Select Register

Interfacing Diagram

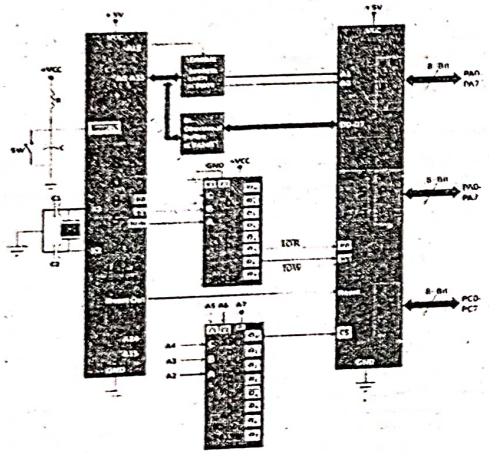


Fig. 10

Memory mapped I/O interfacing with 8085 microprocessor

In memory mapped I/O interfacing with 8085 microprocessor, the I/O devices are not given separate addresses other than treated as a memory location. Whose address range between 0000h to FFFFh (64k). But some part of the space is reserved for I/O devices. The advantage is any instruction that references memory can also transfer data between an I/O device and the microprocessor, as long as the I/O port is assigned to the memory address space rather than to the I/O address space. The register associated with the I/O port is simply treated as memory location register.

This memory mapped I/O interfacing with 8085 microprocessor with an example in which address bit A_{15} designates whether instructions reference memory or an I/O device. If $A_{15} = 0$, a memory register is addressed; If $A_{15} = 1$, than a memory mapped I/O device is address. This assignment elevates the first 32k bytes of memory address space to memory and second 32k to memory mapped I/O devices. External logic generates devices select pulses for memory mapped I/O only when IO/$\overline{M}$ = 0, the appropriate address is on the address low and a $\overline{WR}$ or $\overline{RD}$ strobe occurs.

Input and output transfer using memory mapped I/O are not limited to the accumulator. For example, same of 8085 A instructions that can be used for input from memory mapped I/O ports.

MOV r, m : Move the connects of input port whose address is available in (H,L) reg pair to any internal register.

LDA addr : Load the acc with the content of the input port whose address is available as a second and third byte of the instruction.

Other instructions include, ANA M, ADD M, M provide input data transfer and computation in a single instruction. Same instruction that out the data from memory mapped ports are :

MOV M,r

STA addr

MVI M, data

SHLD addr

LHLD and SHLD carry out 16- bit I/O transfers with single instructions which reduce program executive time considerably. The price paid for this added capability is a reduction in directly addressable main memory and the necessity of decoding a 16- bit rather than an 8-bit address.

When a microprocessor puts out an address and generates a control strobe for a memory read, it has no way of determining whether the device that responds with data is a memory device or an I/O device. It only requires that the devices that respond with in the allowable access time or uses the READY line to request a sufficient number of WAIT states. It supplies an address data and a write strobe and continues its operations, external logic determines whether memory, I/O or anything at all receives the data transferred.

Memory address decoding is nothing but to assign an address for each location in the memory chip. The data stored in the memory is accessed by specifying its address. Memory address can be decoded in two ways:

1. Absolute or fully decoding
2. Linear select or partial decoding

Absolute address decoding has many advantages such as :

1. Each memory location has only one address, there is no duplication in the address.
2. Memory can be placed contiguously in the address space of the microprocessor.
3. Future expansion can be made easily without disturbing the existing circuitry.

There are few disadvantages in this method :

1. Extra decoders are necessary.
2. Some delay will be produced by these extra decoders.

The main advantage of linear select decoding is its simplified decoding circuit. This reduces the hardware design cost. But there are many disadvantages in this decoding.

1. Multiple addresses are provided for the same location.
2. Complete memory space of the microprocessor is not efficiently used.
3. Adding or interfacing ICs with already existing circuitry is difficult.

Absolute Address Decoding: The 8085 microprocessor has 16 address lines. Therefore it can access 2^{16} locations in the physical memory. If all these lines are connected to a single memory device, it will decode these 16 address lines internally and produces 2^{16} different addresses from 0000H to FFFFH so that each location in the memory will have a unique address.

74LS138 address decoder to generate the chip select signals for each memory block. In this decoder when the address lines A_{13}, A_{14} and A_{15} are 000, the output line Y_0 will be activated as shown in Fig. 11. This in turn selects the first memory block. Similarly when these lines are 001 (C=0, B=0 and A=1) Y_1 will be activated and the second memory block will be selected.

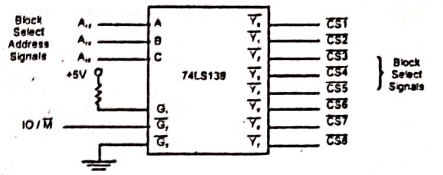


Fig. 11

In this type of memory interfacing, all the address lines (A_0 to A_{15}) have been used. Each location in the memory will have a single address. This type of address decoding is called as absolute or fully decoded addressing.

According to the value of A_0 and A_1, any one register will be selected and to select one memory chip we need one chip select signal, CS signal as shown in the Figure 12.

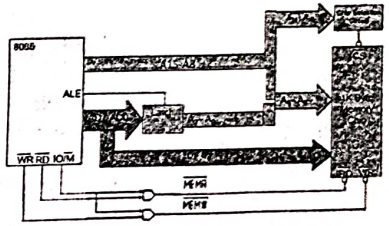


Fig. 12

Prob.13 ***Explain 8279 Keyboard Interface in detail.*** [R.T.U. 2019]

OR

Explain 8279 with block diagram. [R.T.U. 2016]

OR

Draw and explain the block diagram of 8279 keyboard/ display interface. [R.T.U. 2015]

OR

Explain working and modes of 8279 keyboard/display interface. [R.T.U. 2014]

OR

Explain various programming modes of 8279 keyboard and display controller. Also draw a block diagram showing its interfacing with microprocessor 8085. [R.T.U. 2013]

OR

List the major components of 8279 keyboard/display interface and explain their functions. [R.T.U. 2012]

OR

Explain input modes provided by 8279. [R.T.U. 2008, Raj. Univ. 2001]

Sol. Programming Modes of 8279

Keyboard Modes

(i) Scanned Keyboard Mode with 2 Key Lockout : In this mode of operation, when a key is pressed, a debounce logic comes into operation. During the next two scans, other keys are checked for closure and if no other key is pressed the first pressed key is identified. The key code of the identified key is entered into the FIFO with SHIFT and CNTL status, provided the FIFO in not full, i.e. it has at least one byte free. If the FIFO does not have any free byte, naturally the key data will not be entered and the error flag is set. If FIFO has at least one byte free, the above code is entered into it and the 8279 generates an interrupt (on IRQ line) to the CPU to inform about the previous key closures. IF another key released before the first key, the keycode is entered into FIFO. If the first pressed key is released before the others, the first will be ignored. A keycode is entered to FIFO only once for each valid depression, independent of other keys pressed along with it, or released before it. If two keys are pressed within a debounce cycle (simultaneously), no key is recognized till one of them remains closed, and the other is released. The last key, that remains depressed is considered as single valid key depression.

(ii) Scanned Keyboard with N-Key Rollover : In this mode, each key depression is treated independently. When a key is pressed, the debounce circuit waits for 2 keyboard scans and then checks whether the key is still depressed. If it is still depressed, the code is entered in FIFO RAM. Any number of keys can be pressed simultaneously and recognized in the order, the keyboard scan recorded them. All the codes of such keys are entered into FIFO. Note that, in this mode, the first pressed key need not be released before the second is pressed. All the keys are sensed in the order of their depression, rather in the order the keyboard scan senses them, and independent of the order of their release.

(iii) Scanned Keyboard Special Error Mode : This mode is valid only under the N-Key rollover mode. This mode is programmed using end interrupt/error mode set command. If during a single debounce period (two keyboard scans) two keys are found pressed, this is considered a simultaneous depression and an error flag is set. This flag, if set, prevents further writing in FIFO) but allows generation of further interrupts to the CPU for FIFO read. The error flag can be read by reading the FIFO status word. The error flag is set by sending normal clear command with CF = 1.

(iv) Sensor Matrix Mode : In the sensor matrix mode the debounce logic is inhibited. The 8-byte FIFO RAM now acts as 8 × 8-bit memory matrix. The status of the senso switch matrix is fed directly to sensor RAM matrix. Thus th sensor RAM bits contain the row-wise and column-wis status of the sensors in the sensor matrix. The IRQ line goe high, if any change in sensor value is detected at the end of sensor matrix scan or the sensor RAM has a previous entr to be read by the CPU. The IRQ line is reset by the first dat read operation, if AI = 0, otherwise, by issuing the end interrup command. AI is a bit in read sensor RAM word.

Display Modes : There are various options of dat display. For example, the command number of character can be 8 or 16, with each character organized as single 8-bi or dual 4-bit codes. Similarly there are two display format The first one is known as left entry mode or type writer mod since in a type writer the first character typed appears at th left-most position, while the subsequent characters appea successively to the right of the first one. The other displa format is known as right entry mode, or calculator mod since in a calculator the first character entered appears the rightmost position and this character is shifted one positio left when the next character is entered. Thus all the previous entered characters are shifted left by one position when new character is entered.

(i) Left Entry Mode : In the left entry mode, the dat is entered from the left side of the display unit. Address 0 the display RAM contains the leftmost display character an address 15 of the RAM contains the rightmost displa character. It is just like writing in our note books, *i.e.* fro left to write. If the 8279 is in autoincrement mode, the displ RAM address is automatically updated with successive rea or writes. The first entry is displayed on the leftmost displ and the sixteenth entry on the rightmost display. Th seventeenth entry is again displayed at the leftmost displ position.

(ii) Right Entry Mode : In the right entry mode, t first entry to be displayed is entered on the rightmost displa The next entry is also placed in the right most display b after the previous display is shifted left by one display positio The leftmost character is shifted out of that display at t seventeenth entry and is lost, i.e. it is pushed out of the displ RAM.

Interfacing of 8279 with microprocessor 8085:

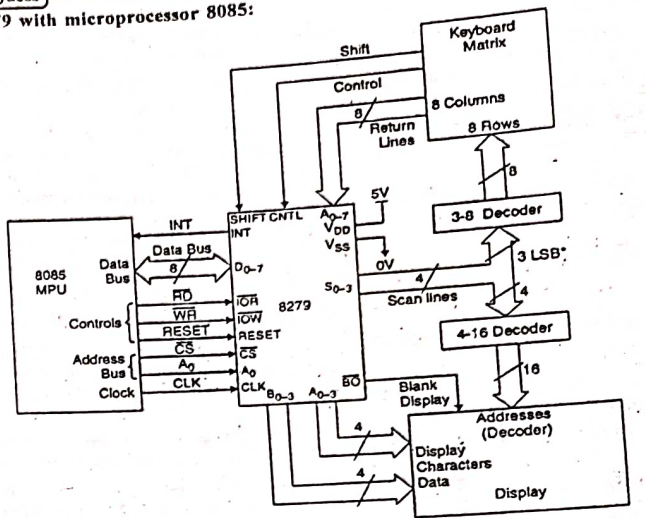


Fig. 1 : Interfacing Connection Block diagram of 8279

8279 Keyboard / Display Interface : 8279 is a general purpose programmable keyboard and display I/O interface device. It can be interfaced with any type of CPU. It relieves CPU scanning keyboard refreshing display, debouncing key closures, checking display etc.

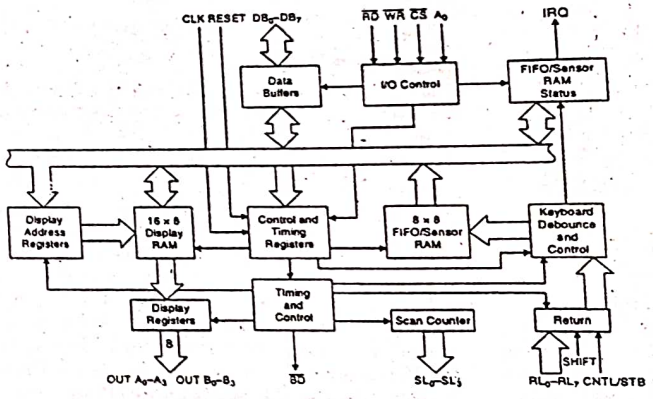


Fig. 2 : Block Diagram of 8279

The component of 8279 keyboard / display interface categorize in different section that are following :

1. Keyboard Section : It consists of :

(i) Keyboard debounce

(ii) FIFO / Sensor-RAM

(iii) Return Buffer

(iv) Status Logic

(i) Keyboard Debounce : This section has 8 lines (RL_0 - RL_7) that can be connected to 8 columns of a keyboard plus two additional lines SHIFT and CNTL / STB (Control/ Strobe). The status of the SHIFT key and the control key can be stored along with a key closure. The keys are automatically debounced and keyboard can operate in two modes (a) Two-key lockout (b) N-key roll over.

(a) Two-key Lockout Mode : In two key lockout, if two keys are pressed almost simultaneously then only 1st key is recognized.

(b) N-key Roll Over Mode : In the N-key roll over mode, simultaneously keys are recognized, their codes are stored in the internal buffer.

(ii) FIFO / Sensor RAM : The keyboard section also includes 8 × 8 FIFO (First in First Out) RAM. Its functions depends upon operating mode of 8279. This FIFO RAM consists of 8 registers that can store 8 keyboard entries; each is then read in the order of entries.

(iii) Return Buffer : The $RL_0 - RL_7$ lines are buffered and latched in this block. It scans $RL_0 - RL_7$ lines during each row scan. In scanned keyboard or sensor matrix mode $RL_0 - RL_7$ lines are not scanned in strobbed mode.

(iv) Status Logic : The status logic keeps track on the number of entries and provides an IRQ (Interrupt Request) signal when the FIFO is not empty.

2. Scan Section : The scan section has a scan counter and four scan lines ($SL_0 - SL_3$). These scan lines can be decoded using a 4 to 16 decoder to generate 16 lines for scanning. These lines can be connected to the rows of a matrix keyboard and the digit drivers of a multiplexed display.

3. Display Section : Display section has 8 output lines divided into two groups $A_0 - A_3$ and $B_0 - B_3$. These lines can be used, either as a group of 8 lines or as two group of four in conjuction with the scan lines for multiplexed display. The display can be blanked by using the $\overline{BD}$ line. This section includes 16 × 8 display RAM. The MPU can read from or write into any of these registers.

(i) Display Address Register : It contains two type of address registers *i.e.* microprocessor controlled address register and timing controlled address register.

(ii) Display Register : These registers convert the codes of character to be displayed. They are divided into two nibbles *i.e.* nibble A and nibble B.

Nibble A and nibble B can be blanked and inhibited individually.

4. MPU Interface Section : It consists of data buffer I/P control, control and timing register and timing and control logic.

(i) Data Buffer : These are bidirectional tristate buffers. It includes 8 bidirectional data bus ($DB_0 - DB_7$).

(ii) I/O Control : It uses $\overline{CS}$, A_0, $\overline{RD}$ and $\overline{WR}$ lines to control data flow to and from the various internal registers and buffers.

(iii) Control and Timing Registers : This block consists of control and timing register. 8279 provides 8 control and timing registers which are distinguished by higher 3 bits of command word.

Input Modes By 8279 : The 8279 is a 40-pin device which functionally consists of (i) a CPU interface, (ii) a set of scan lines, (iii) input lines for key data, and (iv) output lines for display data.

There are three input modes :

(i) Scanned Keyboard

(ii) Scanned Sensor Matrix and

(iii) Strobed input

(i) Scanned Keyboard Mode : This can have either enclosed or decoded scan lines. Keys are automatically debounced. In this scanned keyboard mode, there are two further alternative ways of operation :

(1) 2-key lockout, and

(2) N-key-rollover

In this mode, when a key is pressed, a unique 6-bit data is generated, characteristic of the key position data along with the CNTL and SHIFT states and 8-bit word is formed which is stored in a RAM inside the 8279. As soon as the 8-bit word is stored in this FIFO (First in First out) RAM, the interrupt request IRQ pin of the device goes high. This could be connected to one of the hardware interrupt lines of the CPU, which could be followed by, in the interrupt service routine program, an instruction to read the FIFO RAM. As the FIFO RAM is read the IRQ goes low, but becomes high again, if the FIFO RAM contains further data.

The data format for the scanned keyboard mode, is as follows :

The three scan bits (D_3 , D_4, D_5) are from the scan counter and indicate the row on which the pressed key was located. The three return bits (D_0, D_1, D_2) are from the column counter and indicate the column on which the pressed key was located.

(1) The 2-key lockout option : In this mode when any key is pressed, the debounce logic is set. For the next two scans, it is checked whether other keys are pressed. At this stage several alternatives are possible.

(a) No other keypresses are found. This is then considered a single key depression, the data is taken into FIFO and IRQ is activated. (If the FIFO is full, the key data is not entered, an error flag is set).

(b) If one or more additional key depression is detected, no data entry to FIFO can occur. This status is resolved as follows :

(A) If all the keys are released before the key first pressed, then the first key data is entered into the FIFO.

(B) If the first key is released before others, then the keypress is entirely ignored. Note that the first key is the crucial one (there may be many other key presses in many different orders); it is the first keypress that is either entered or ignored.

If however, two keys are pressed within one debounce cycle, this is considered as simultaneous depression. Neither of them is recognized till one of them is released. The one that still remains pressed, is considered to be equivalent to a single key depression.

(2) The N-Key rollover option : In this mode, when a key is pressed, the debounce circuit waits for two scans and then checks again whether the key is still pressed. If it still is, the key data is then taken into the FIFO. There is no limit to the number of keys that can be pressed. For a simultaneous depression, the key data are entered according to the order of the keypresses.

A special error mode can be set in this option by End Interrupt/ Error Mode set command, if during signal debounce cycle, two keys are found pressed, then it sets an *Error Flag*, which present further data entering into FIFO, and sets an interrupt. The error flag is read from the FIFO STATUS word, and is reset by sending CLEAR command $C_2 = 1$.

(ii) Scanned Sensor Matrix Mode : In this mode, the keys are placed in the form of a matrix (scan lines comprising the columns and the return lines comprising the rows), and the key status (open or closed) stored in a RAM which can be addressed by the CPU. The matrix size is 8 × 8 (with encoded scan lines) or 4 × 8 (with decoded scan lines).

The data on each of the eight return lines, enter directly in eight columns of the sensor RAM, each switch position therefore maps to a specific sensor RAM position. The SHIFT and CNTL status are not taken as inputs in this mode.

In this mode, besides switches, other devices (or logic circuits) may be connected to the return lines. The constraint is that device (or logic circuits), should be triggerable by the scan lines, the output of the circuit (1 or 0) can then enter data into the return lines.

In this mode, debouncing is not provided, however, it has the advantage that the CPU knows, how long the sensor was closed. The IRQ line goes high if a sensor value is found to have changed at the end of a sensor matrix scan. When the auto increment flag is set to zero, the IRQ line is reset by the first data read operation when the auto increment flag is set to one, then by the End Interrupt command.

(iii) Strobed Input Mode : In this mode, the data is accepted from the return lines and they go to the FIFO, they enter at the rising edge of a CNTL/STB line pulse. The data (placed on return lines) can come from any source. (Such as another encoded keyboard or switch matrix). Each, scan line (now corresponding to arrow in the sensor matrix) would lead to an 8-bit word.

Prob.14 ***Explain, 8254, programmable interval timer, explain all the modes of 8254 with example.*** **[R.T.U. 2017]**

Sol. 8254 Programmable Interval Timer

8254 is popular programmable interval timer chip. Intel 8254 contains three identical 16–bit down counters that can operate independently in any one of the six modes. A 16–bit count is loaded in register and control word is loaded in the control word register. Count is decremented until it reaches zero. When count is terminated, a pulse is generated that can be used to interrupt the microprocessor. Counter can operate in either binary or BCD. 8254 includes status read–back command which helps in reading count by microprocessor while the counter is decrementing.

8254 is an upgraded version of 8253, and both of these are pin–compatible.

Control Word of 8254 PIT : *Refer to Prob.6.*

Operation Modes of 8254 : *Refer to Prob.5.*

Prob.15 *Explain 8255, programmable peripheral interface (controller). Explain all modes of 8255 with example.* [R.T.U. 2017]

OR

Draw and explain the block diagram of 8255 PPI. [R.T.U. 2015]

OR

Explain the all operation mode of 8255 in brief. [R.T.U. 2013]

OR

Explain working and control word format of 8255 programmable peripheral interface. [R.T.U. 2014]

OR

Draw block diagram of 8255 and explain its various modes of operations. [R.T.U. 2012]

OR

Draw the block diagram of 8255 PPI. What are different operating modes of 8255 PPI? Discuss how to determine the control word for 8255. [R.T.U. 2011]

Sol. *Refer to Q.12.*

Prob.16 *Draw and explain the pin diagram of 8255 PPI? Explain all of its modes?* [R.T.U. 2015]

Sol. Pin diagram of 8255 PPI

Table : Pin description of 8255

Pin Symbols	Functions
D_0-D_7 (Data Bus)	These bi-directional, tri-state data bus lines are connected to the system data bus. They are used to transfer data and control word from microprocessor (8085) to 8255 or to receive data or status word from 8255 to the 8085.
PA_0-PA_7 (Port A)	These 8-bit bi-directional I/O pins are used to send data to output device and to receive data from input device. If functions as an 8-bit data output latch/buffer, when used in output mode and an 8-bit data input buffer, when used in input mode.
PB_0-PB_7 (Port B)	These 8-bit bi-directional I/O pins are used to send data to output device and to receive data from input device. If functions as an 8-bit data output latch/buffer, when used in output mode and an 8-bit data input buffer, when used in input mode.
PC_0-PC_7	These 8-bit bi-directional I/O pins are divided into two groups PC_L (PC_3-PC_0) and PC_J (PC_7-PC_4). These groups individually can transfer data in or out when programmed for simple I/O and used as handshake signals when programmed for handshake or bi-directional modes.
$\overline{RD}$ (Read)	When this pin is low, the CPU can read the data in the ports or the status word, through the data buffer.
$\overline{WR}$ (Write)	When this input pin is low, the CPU can write data on the ports or in the control register through the data bus offer.
$\overline{CS}$ (Chip Select)	This is an active low input which can be enabled for data transfer operation between the CPU and the 8255.
Reset	This is an active high input used to reset 8255. When RESET input is high, the control register is cleared and all the ports are set to the input mode. Usually, Reset Out signal from 8085 is used to reset 8255.
A_0 and A_1	These input signals along with $\overline{RD}$ and $\overline{WR}$ inputs control the selection of the control/status word registers or one of the three ports. Table below summarizes the status of A_0, A_1, $\overline{CS}$, $\overline{RD}$ and $\overline{WR}$ to access the control word/ports. A_0 and A_1 are generally connected to the A_0. A_1 pins of the address bus; the 8255 therefore occupies four consecutive locations in the I/O space.

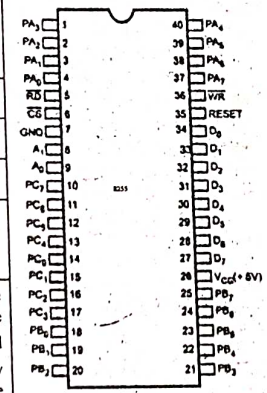


Fig. : Pin diagram of 8255 PPI

Modes of 8255 PPI : *Refer to Prob.12.*

Prob.17 *Interface the keyboard and display controller 8279 with 8085 at address ABCOH. Write an ALP to set up 8279 in scanned keyboard mode with encoded scan, N-key roll-over mode. Use 16-character display in right entry display format.* [R.T.U. 2011]

OR

Interface keyboard and display controller 8279 with 8086 at address 0080H. Write an ALP to set up 8279 in scanned keyboard mode with encoded scan, N-Key rollover mode. Use a 16 character display in right entry display format. Then clear the display RAM with zeros. Read the FIFO for key closure. If any key is closed, store it's code to register CL. Then write the byte 55 to all the displays, and return to DOS. The clock input to 8279 is 2MHz, operate it at 100KHz.

Sol. Interface Keyboard and Display Controller 8279 :

1. The 8279 is interfaced with lower byte of the data bus, *i.e.* D_0-D_7. Hence the A_0 input of 8279 is connected with address line A_1.

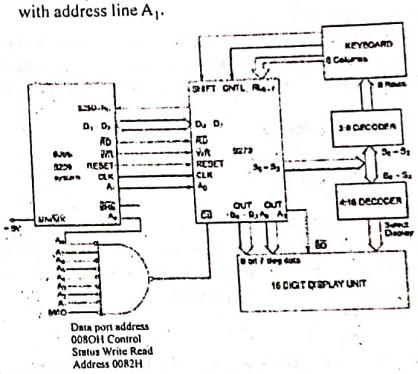


Fig. 1 : Interfacing Schematic

2. The data register of 8279 is to be addressed as 0080H, *i.e.* $A_0 = 0$.
3. For addressing the command or status word A_0 input of 8279 should be 1.
4. The next step is to write all the required command words for this problem.

Keyboard/Display Mode Set CW :

This command byte sets the 8279 in 16-character right entry and encoded scan N-key rollover mode.

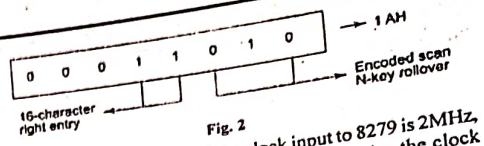


Fig. 2

Program Clock Selection : The clock input to 8279 is 2MHz, but the operating frequency is to be 100KHz, *i.e.* the clock input is to be divided by 20 (10100). Thus the prescalar value is 10100 and the command byte is set as given.

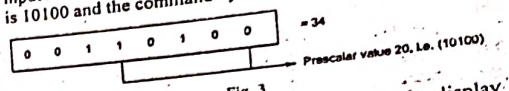


Fig. 3

Clear Display RAM : This command clears the display RAM with the programmable blanking code.

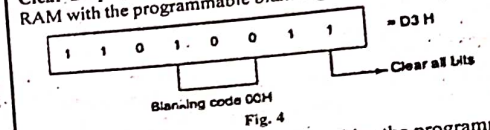


Fig. 4

Read FIFO : This command byte enables the programmer to read a key code from the FIFO RAM.

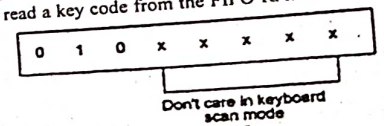


Fig. 5

Write Display RAM : This command enables the programmer to write the addressed display locations of the RAM as presented below.

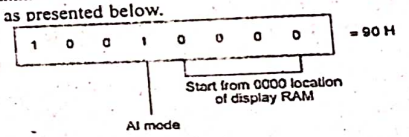


Fig. 6

Program for ALP Required to Initialize the 8279 as Required :

```
        Assume CS      :   Code
        Code Segment
Start:  MOV AL, 1AH    :   Set 8279 in Encoded scan,
        OUT 82H, AL    :   N Key rollover, 16 display, Right entry
                           mode.
        MOV AL, 34H    :   Set clock prescalar to
        OUT 82H, AL    :   100KHz
        MOV AL, 0D3H   :   Clear display ram
```

```
        OUT 82H, AL   :  command
        MOV AL, 40H   :  Read FIFO command
        OUT 82H, AL   :  for checking display RAM
Wait:   IN AL, 82 H   :  Wait for clearing of
        AND AL, 80H   :  Display RAM by reading
        CMP AH, 80H   :  FIFO Du bit of the status word i.e.
        JNZ Wait      :  If Du bit is not set wait, else proceed.
        MOV AH, 40H   :  Read FIFO command
        OUT 82H, AL   :  for check key closure
        IN AL, 82H    :  Read FIFO status
        AND AH, 07H   :  Mask all bits except the
        CMP AH, 00    :  number of characters bits
        JNZ Key code  :  if any key is pressed, take
Warm:   MOV AL, 90H   :  required action, otherwise
        OUT 82H, AL   :  Proceed to write display
        MOV AL, 55H   :  RAM by using write display
        MOV CL, 10H   :  command. Write the byte
Next:   OUT 80H, AL   :  55H to all display RAM
        DEC CL        :  Locations
        JNZ Next      :
        JMP Stop      :
        Key code      :  Call Read code: Call routine to read
                         the key
        JMP Warm      :  Code of the pressed key is assumed
                         available
Stop    MOV AH, 4CH   :  stop
        INT 21H
        Code ENDS
        END START
```

Prob.18 *Write short note on DMA Controller 8257.*

[R.T.U. 2010]

Sol. Direct Memory Access (DMA) Controller-8257

In the software technique of data transfer, the microprocessor is busy in task, which does not require processing.

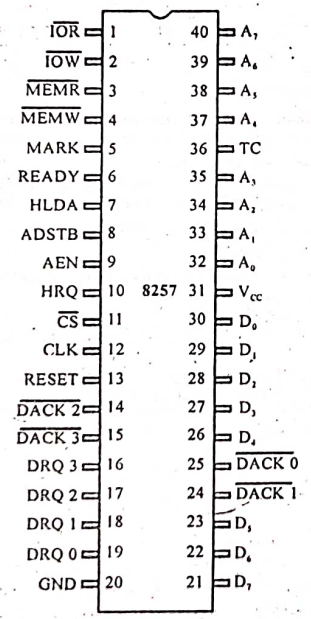
Direct Memory Access (DMA) is an I/O technique commonly used for high speed data transfer technique; for example, data transfer between primary memory and hard-disk. In this technique, microprocessor releases the control of buses to a device, called DMA controller. DMA controller transfers the data between primary memory and various peripherals. It bypasses the microprocessor. The DMA controller works in two modes: master mode and slave mode. The HOLD and HLDA signals of the 8085 microprocessor are used by the DMA controller.

Channels and Registers of 8257 : The 8257 DMA controller has 4-channels, which may be used to connect 4 peripherals. Each channel has two 16-bit registers. One is called the address register, which is used to hold the 16-bit memory address. Another register, called count register, is used to store 14-bit count and 2-bit to specify DMA function: Read, Write or Verify (see figure 2 (a)). Two signals DRQ and $\overline{DACK}$ are also connected to each channel which are explained below.

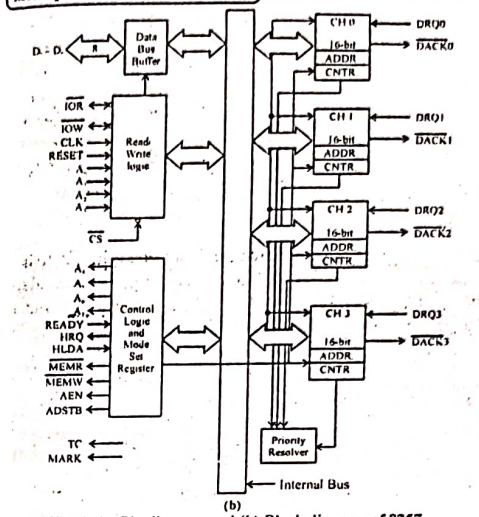
Two additional registers, called Mode Set Register (with control word format) and status register are also shown in fig. 2(b). The Mode Set register is similar to control register used to store the control word. The address of Mode Set register and status register are same. The address refers to Mode Set register for write operation and refers to status register for read operation.

Pin Diagram of 8257

The 8257 has 40 pins. The pin diagram of 8257 is shown in figure 1(a).



(a)



(b)

Fig. 1: (a) Pin diagram and (b) Block diagram of 8257

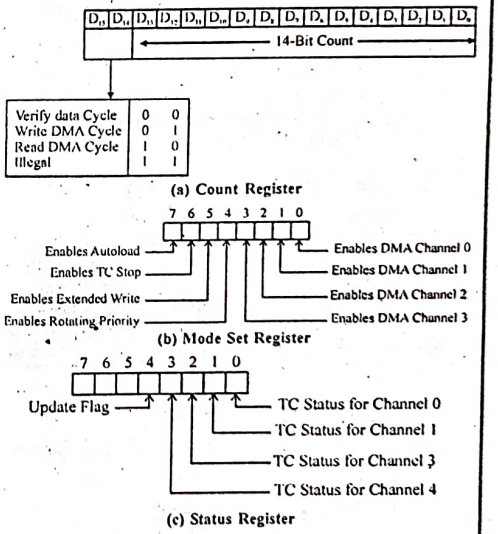


Fig. 2

Working of DMA Controller : To transfer data from primary memory to peripheral and vice-versa, the DMA controller functions in two modes: the Slave Mode and the Master Mode. The tasks performed in these modes are discussed below :

Slave Mode: In slave mode, DMA controller works as a peripheral. In this mode, the address lines $A_0 - A_3$, working as the input, in combination with the Chip Select, are used to select various registers of the DMA controller. The $\overline{IOR}$ and $\overline{IOW}$ pins also work as input, used for read and write operation. Following tasks are performed in this mode :

(I) Microprocessor writes the Control Word in the Mode Set Register.

(II) Microprocessor writes the low-order and then high order byte of the count, including data flow information, into the Count Register of the proper channel.

(III) Writes the low-order and then high order byte of the memory address into the Address Register of the proper channel.

In this mode, the address lines $A_4 - A_7$ and the control signal $\overline{MEMR}$ and $\overline{MEMW}$ are tri-stated. The other signals are not used.

Master Mode : After intialization, in master mode, the DMA controller keeps checking for DRQ (DMA Request) signal. When any peripheral needs data transfer, it sends DRQ signal to the concern channel. After receiving DRQ signal from the peripheral, DMA controller performs following tasks.

(I) DRQ signal enables the channel. The DMA controller generates HRQ (high) signal, which is connected to HOLD pin of the microprocessor.

(II) After receiving HOLD signal, in the next cycle, microprocessor releases the control of buses and generates HLDA (high) signal, which is connected to HLDA pin of the DMA controller.

(III) When DMA controller receives HLDA signal, it generates $\overline{DACK}$ (low) signal, which acknowledge the requesting peripheral.

(IV) At the same time, 8257 generates AEN (high) signal which disables the microprocessor's demultiplexed address bus ($A_7 - A_0$) by disabling the upper 8212, which is active low. AEN also blocks the control signals from the microprocessor, by disabling the 74LS257 chip, which is also active low. Now the control signal will be generate from the DMA controller.

(V) When AEN is high, ADSTB signal also goes high, which in combination with AEN, enables the lower 8212 chip, to generated high order byte of the memory address ($A_{15} - A_8$).

(VI) The low-order memory address is generated by the DMA controller.

(VII) Data transfer continuous until counter reaches zero.

□□□

MICROPROCESSOR APPLICATIONS 5

CHAPTER IN A NUTSHELL

❑ **Microprocessor Application**

Applications include such examples as the scanned LED display, the matrix keyboard and LCD etc.

LCD (Liquid Crystal Display)

An LCD is used in system where low power consumption is necessary. Watches, calculators and consumer electronics displays are its examples.

❑ **Interfacing a Matrix Keyboard**

A matrix keyboard is commonly used as input device when more than eight keys are necessary instead of a linear format row of keys.

❑ **Keyboard Display Controller 8279**

- It is a hardware approach to interface a matrix keyboard and a multiplexed display. The software approach to interface a matrix keyboard and a multiplied display of seven segment LEDs is used.
- 8279 reduces time in checking the keyboard and refreshing the display.

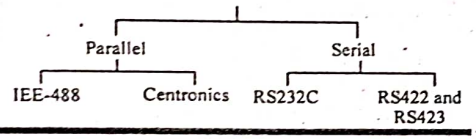
Keyboard entries redelayed and stored in the internal FIFO (first in first out) memory, an interrupt signal is generated with each entry.

- The display segment can provide a 16-character scanned display interval with such device as LEDs.
- This segment has 16 × 8 R/W memory (RAM) which can be used to read/write information for displayed purpose.
- The display can be set up is either right entry or left entry format.
  - Level converters MC 1488 MC 1489
  - Communication buses : Centronics IEEE 488
  - Current loop
  - RS 232C
  - RS 422A and RS423A

❑ **Communication Buses**

Through interfacing these are of two types :

Communication Buses



PREVIOUS YEARS QUESTIONS

PART-A

Prob.1 *List the features of 8251.* [R.T.U. 2019]

Sol. Features of 8251 Microcontroller : The features of 8251 Microcontroller are namely :

1. The Intel 8251A is an universal synchronous and asynchronous communication controller.
2. It supports standard asynchronous protocol with :
   - 5 to 8 bit character format.
   - Odd, even or no parity generation and detection.
   - Baud rate from DC to 19.2 Kbaud.
   - False start bit detection.
   - Automatic break detect and handling.
   - Break character generation.
3. It has built in baud rate generator.
4. 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.
5. It allows full duplex transmission and reception.
6. It provides double buffering of data both in the transmission section and in the receiver section.
7. It provides error detection logic, which detects parity, overrun and framing errors.
8. It has Modern Control Logic, which supports basic data set control signals.
9. It provides separate clock inputs for receiver and transmitter sections, thus providing an option of fixing different baud rates for the transmitter and receiver section.
10. It is compatible with an extended range of Intel microprocessors.
11. It is fabricated in 28 pin DIP package and its all inputs and outputs are TTL compatible.
12. It is available in standard as well as extended temperature range.

Prob.2 *Write difference between serial communication and parallel communication.* [R.T.U. 2017]

Sol. Serial and Parallel Communication : Data can be transmitted between a sender and a receiver in two main ways: serial and parallel.

Serial communication is the method of transferring one bit at a time through a medium.

0	1	0	0	0	0	1	0

Parallel communication is the method of transferring blocks, eg: BYTEs, of data at the same time.

0
1
0
0
0
0
1
0

As parallel communication is faster than the serial communication, therefore it is preferred over serial. For this reason, the internal connections in a computer, i.e. the busses, are linked together to allow parallel communication. However, the use of parallel communication for longer distance data communication is unfeasible for economic and practical reasons, i.e. amount of extra cable required and synchronisation difficulties. Therefore, all long distance data communications takes place over serial connections.

Prob.3 *Draw a schematic diagram to show how RS 232C cable is connected through 8251 to 8085.* [RTU 2011]

Sol. Block Diagram

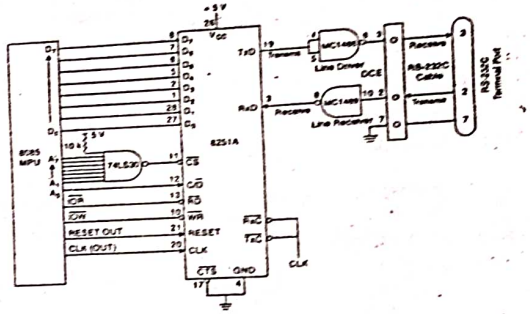


Fig. : Schematic of Interfacing an RS-232 Terminal with an 8085 System using the 8251A

Prob.4 *Explain Bus Interface Unit.*

Sol. Bus Interface Unit (BIU)

This unit handles all transfer of data and address on the buses. *i.e.*, it sends out addresses, fetches instructions from memory, reads data from input port and memory, writes data to output port and memory.

The BIU includes 4 segments registers (each of 16 bits), called Code Segment (CS) Register, Data Segment (DS) Register, Stack Segment (SS) Register and Extra Segment (ES) Register and one offset register called Instruction Pointer (IP). In addition to these registers, the BIU has an array of 6 registers called queue.

PART-B

Prob.5 ***Explain signals of RS-232C in detail.*** **[R.T.U. 2019]**

Sol. RS-232C : Fig. 1(b) shows the RS-232C 25-pin connector and its signals. The signals are divided into four groups : Data signals, Control signals, Timing signals, and Grounds. For data lines, the voltage level +3 V to + 15 V is defined as logic 0; from –3 V to –15 V is defined as logic 1 (normally, voltage levels are ±12V). This is negative true logic. However other signals (control and timing) are compatible with the TTL level. Because of incompatibility of the data lines with the TTL logic, voltage translators, called line drivers and line receivers, are required to interface TTL logic with the RS-232 signals, as shown in fig. 1(a). The line driver, MC1488, converts logic 1 into approximately –9 V and logic 0 into +9 V, as shown in fig. 1(a). Before it is received by the DCE, it is again converted by the line receiver, MC1489, into TTL-compatible logic. This raises the question : if the received signal is to be converted back to the TTL level. What is the reason, in the first place to convert transmitted signal to the higher level? The primary reason is that the stan 'ard was defined before the TTL level came into existence; before 1960, most equipment was designed to handle higher voltage. The other reason is that this standard provides a higher level of noise margin from –3V to +3V.

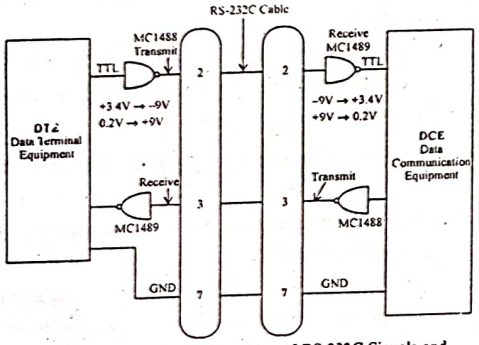


Fig. 1(a) : Minimum Configuration of RS-232C Signals and Voltage Levels

Signals	Pins		Signals
Secondary Transmitted Data	14	1	Protective Ground
Transmission Signal Element Timing (DCE Source)	15	2	Transmitted Data (T_xD) → DCE
Secondary Received Data	16	3	Received Data (R_xD) → DTE
Receiver Signal Element Timing (DCE Source)	17	4	Request to Send (RTS) → DCE
Unassigned	18	5	Clear to Send (CTS) → DTE
Secondary Request to Send	19	6	Data Set Ready (DSR) → DTE
DCE ← Data Terminal Ready (DTR)	20	7	Signal Ground
Signal Quality Detector	21	8	Received Line Signal Detector
Ring Indicator	22	9	(Reserved for Data Set Testing)
Data Signal Rate Selector (DTE/DCE Source)	23	10	(Reserved for Data Set Testing)
Transmit Signal Element Timing (DTE Source)	24	11	Unassigned
Unassigned	25	12	Sec. Read Line Sig. Detector
		13	Sec. Clear to Send

Fig. 1(b) : RS-232C Signal Definitions and Pin Assignments

The minimum interface between a computer and a peripheral, requires three line pins 2, 3, and 7, as shown in fig. 1(a). These lines are defined in relation to the DTE, the terminal transmits on pin 2 and receives on pin 3. On the other hand, the DCE transmits on pin 3 and receives on pin 2. Now the dilemma is: How does a manufacturer define the role of its equipment? For example, the user may connect its microcomputer to a serial printer configured as a DTE. Therefore, to remain compatible with the defined signals of the RS-232C, the RS-232 cable must be reconfigured as shown in fig. 2(b). In figure 2 the microcomputer is defined as DTE, and it can be connected to the modem, defined as a DCE, without any modification in the RS-232 cable, as shown in fig. 2(a). However, when it is connected to the printer, the transmit and the receive lines must be crossed as shown in fig. 2(b), this is known as a null-modem connection.

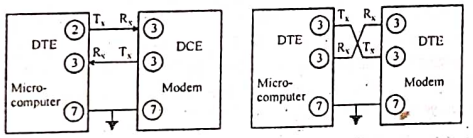


Fig. 2 : (a) RS-232C Connections DTE to DCE and (b) DTE to DTE

Typically, data transmission with a handshake requires eight lines. Listed in Table 1. Specific functions of handshake lines differ in different peripherals and, therefore should be referred to in the manufactures manuals.

For high/speed transmission, the standards RS-422A and RS-423A are used. These standards use differential amplifiers to reject noise levels and can transmit data at higher speed with longer cable. The RS-422A allows a maximum speed of 10 kbaud for a 40-ft distance and 100 kbaud for 4000 ft. The RS-423A is limited to 100 kbaud for a 30-ft distance and 10 kbaud for 300 ft. Table 2 shows comparison of the three standards RS-232C, RS-422A, and RS-423A.

Table 1 : RS-232C Signals used with Handshake Data Communication

Pin No.	Signals	Pin Name	Functions
2	Transmitted Data	T×D	Output: transmits data from DTE to DCE
3	Received Data	R×D	Input: DTE receives data from DCE
4	Request to Send	RTS	General purpose output from DTE
5	Clear to Send	CTS	General-purpose input to DTE; can be used as a handshake signal
6	Data Set Ready	DSR	General-purpose input to DTE; can be used to indicate that DCE is ready
7	Signal Ground	GND	Common reference between DTE and DCE
8	Data Carrier Detect	DCD	Generally used by DTE to disable data reception
20	Data Terminal Ready	DTR	Output: generally used to indicate that DTE is ready

Table 2 : Comparison of Serial I/O Standards

Specifications	RS-232C	RS-422A	RS-423A
Speed	20 kbaud	10 Mbaud at 40 ft	100 kbaud at 30 ft
		100 kbaud at 4000 ft	1 kbaud at 4000 ft
Distance	50 ft	4000 ft	4000 ft
Logic 0	> +3 to + 25 V	B > A	+4 to +6 V
Logic 1	< –3 to –25 V	B < A	–4 to –6 V
Receiver Input			
Voltage	± 15 V	± 7 V	± 12 V

Merits

1. The advantage of current loop method is that signals are noise free.
2. Suitable for transmission over a distance.

Demerits

1. The same standard is used for communication between computers and peripherals and the roles of a data terminal and a modem have become ambiguous.
2. The lines used for transmission and reception will differ, depending on the manufacturer's role definition of its equipment.

Prob.6 ***Write short note on liquid crystal display.*** **[R.T.U. 2016]**

Sol. Liquid Crystal Display: A liquid crystal display (LCD) is an output device. It is a handy method of providing a low-power consumption user interface. LCDs are everywhere. You probably have seen them as displays in watches, calculators, cellular phones, and even as a color television type monitor on a video camera. LCDs are available in monochrome as well as color displays. They are also available in two general categories:

(1) Character display

(2) Graphic dot matrix style display

In general, a LCD display consists of two panes of polarized glass with a liquid suspended between the two panes. The liquid between the two panes of glass may be of either the twisted nematic (TN) or super twisted nematic (STN) type. In general, the TN technology is less expensive, but it has poorer contrast and a narrower viewing angle than the STN technology. Whichever technology is employed, characters and graphics are formed within the display by arranging the liquid crystals into segments. When the individual liquid crystal segments are properly excited, their optical properties may be adjusted to transmit or reflect light such that characters or graphics are formed.

Features available on LCDs include different color panels, backlit displays for easier viewing, many different configurations of characters per line, number of lines, and a number of dot matrix formats. All intelligent LCDs are controlled by a microprocessor that is provided with and affixed to the backside of the LCD display. The microprocessor converts the input provided to the LCD module into the proper excitation to display the input. The microprocessor is equipped with a ROM to store a character library and a RAM that holds character currently being displayed. So when you establish an interface between the 68HC12 and the LCD display, you are actually establishing an interface between two processors.

Prob.7 ***What is the difference between microprocessor and microcontroller.*** **[R.T.U. 2015]**

Sol. A microprocessor is an Integrated Circuit which only has CPU inside it. It only has programming capabilities like Intel Pentium processors. It doesn't have any memory like RAM and ROM and other peripheral on the chip. These are to be provided externally to make the system as a whole to function in certain way. Since memory and I/O components are all external, each instruction will need external operation, hence it is relatively slower. They have less number of registers and more operations are memory based. It finds application in Laptops and Desktop PCs. Microprocessors can be used to perform numerous tasks like developing software, games, photo and video editing. These tasks are not specific in input and output and depends on the usage. These tasks also require more resources like high RAM and various I/O ports and hence can't be embedded on a single chip. For the same reason their cost is very high. The microprocessors operate at a very high clock rate at around 1-2 GHz as they have to perform very complex tasks and perform them quickly.

A micro-controller on the other hand has some fixed amount of RAM, ROM and other peripherals along with the CPU on a single chip. It is also termed as a mini computer. Since components are internal, most of the operations are

internal instruction, hence speed is fast. They also have more number of registers than microprocessors. They perform some specific tasks where the relationship between input and output is defined. Since they have a specific application, they need small resources like RAM and I/O ports and hence can be embedded on a single chip. As they have limited tasks they are far cheaper than microprocessors. Micro-controllers find applications in various places like in cars, bikes, washing machines, mp3, keyboard, mouse, pen-drive and many others. They operate at low frequency of about 50 MHz in comparison to 1 GHz speed of micro-processors.

Prob.8 ***Write short note on 8257.*** **[R.T.U. 2014]**

Sol. The 8257 is a 40-pin Intel IC which is a programmable DMA controller. It is a 4-channel direct memory access(DMA) controller, It is specifically designed to simplify the transfer of data at high speeds for the Intel microcomputer systems. Its primary function is to generate, upon a peripheral request, a sequential memory address which will allow the peripheral to read or write data directly to or from memory. The system bus is acquired via the CPU's hold function. The 8257 has priority logic that resolves the peripheral requests and issues a composite hold request to the CPU. It maintains the DMA cycle count for each channel and outputs a control signal to notify the peripheral that the programmed number of DMA cycles is complete.

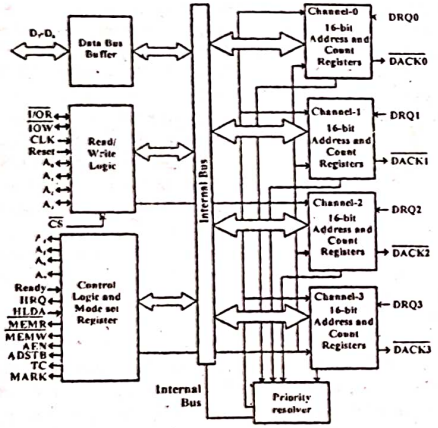


Fig. : 8257 block diagram

Prob.9 ***Explain the interfacing of Liquid Crystal Display with microprocessor 8085.*** **[R.T.U. 2014]**

Sol. An LCD consists of cyrstal material placed between two plates. The crystal material can pass or block the light that passes through ; thus it creates a display.

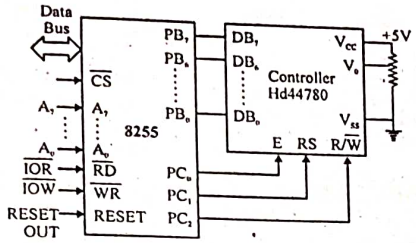


Fig. 1

To display a character, certain segments or dot are driven by a square wave pattern, which is supplied by a built in driver. The driver is interfaced with the processor (8085) using signals such as data lines, enable, chip select and read/ write.

Fig.1 shows 20 character X 2–line dot matrix LCD module using 8255. The built in controller driver (HD 44780) is 14–pin. 8 bits(DB_0–DB_7) are data lines , E, RS, R/$\overline{W}$ are for control signals. V_{cc}, V_o, V_{ss} are for power supply and ground. The controller has two 8-bit control registers -

1. Instruction registers for commands
2. Data registers for data.

Mircoprocessor can write command when RS is low and data when RS is high. When microprocessor writes one of the registers, the controller sets data line DB_7 high as a busy flag. After the controller completes its internal operation, it resets the busy flag. Finally, to display a message, the LCD controller must be initialized by writing into the IR Register.

- Timing diagram(fig. 2) shows that after selection of a register (IR and DR) by RS, the R/$\overline{W}$ signal is asserted and enable signal is pulsed low-high-low.
- At trailing edge of enable signal, the controller sets the flag(data line DB_7) and executes the internal operation of either writing a command or displays a character.

- During the internal operation, the microprocessor should continue to check the busy flag(DB_7) by asserting the enable signal (E) low-high-low until the controller resets the flag.
- When DB_7 is reset, the microprocessor can write the next command or data.

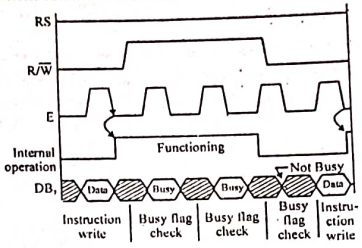


Fig. 2

Prob.10 ***Explain the application of microprocessor in Interfacing scanned multiplexed display and liquid crystal display.*** **[R.T.U. 2013]**

Sol. Interfacing of Scanned Multiplexed Displays : For interfacing a seven segment LED using 8255, we need one I/O port and a driver per LED to display one hex digit. To overcome this problem we use multiplexing, whereby the data lines and output ports are time shared by various seven segment LEDS.

The basic circuit for multiplexed display is shown in Fig.

The cicuit has two output ports.

(1) One port (P_A) to drive LED segments.

(2) Port (P_B) to turn on the corresponding cathodes since it is common cathode seven segment LED.

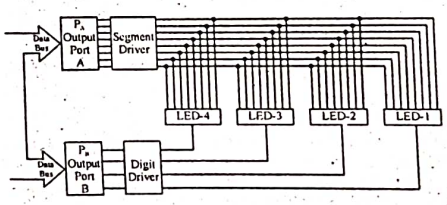


Fig. 1 : Block Diagram of Multiplexed Output Display

(3) The address line of 8085 microprocessor, A_7-A_2 are connected to the decoder 74LS138 and remaining two address lines A_1 and A_0 will be connected directly to the 8255A.

(4) **SN 75492-Digit Driver :** It has sis darlington pairs in package. To display a digit, the seven segment code for the digit is sent to port A the corresponding cathode is turned on and off in sequence.

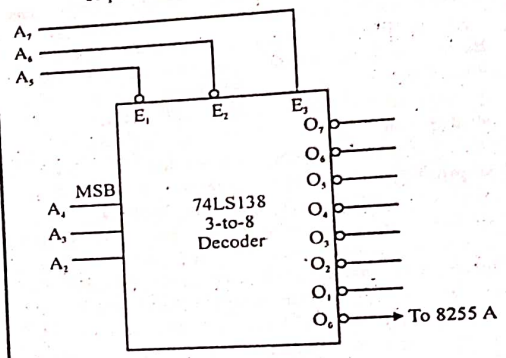


Fig.2 : Address Decoding

(5) Each seven segment LED is turned on an off sequentinally.

(6) The cycle is repeated fast enough that hte display appears stable. This is known as scanned displays.

(7) **SN 75491 Segment Driver**

- Quad device having four darlington pair transistor in a package.
- To dirve eight data lines, we need two devices as shown in Fig.3.
- Pin A, the base of the transistor, is connected to one of the data lines of O/P port and emitter E is connected to one of the LED segments.
- Similarly, the base of the driver is tied to +5V through a resistor to turn on the drive safely.

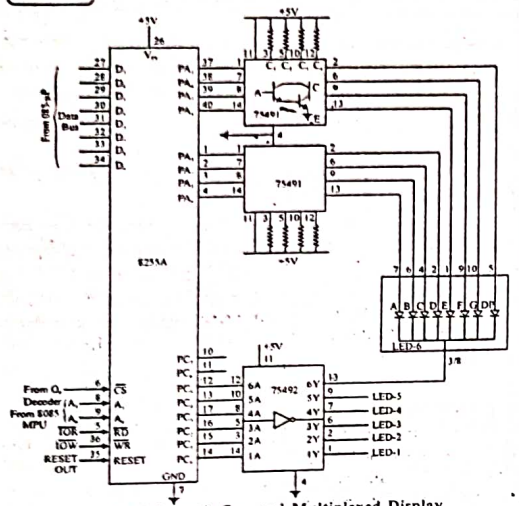


Fig. 3 : Schematic Scanned Multiplexed Display

Liqud crystal display : An LCD is used in systems where low power consumption is necessary. An LCD display consists of cyrstal material placed between two plates in the form of a dot matrix.

- Crystal material is arranged in segments or in the form of a dot matrix.
- The crystal material can pass or block the light that passes through; thus it creates a display.
- To display a character, certain segments or dots are driven by a square wave pattern, which is supplied by a built – in driver.
- This driver is interfaced with the processor using signals such as data lines, enable, read/write and chip select.

For interfacing a 20-character X2-line dot matrix LCD using 8255 following circuit diagram is use :

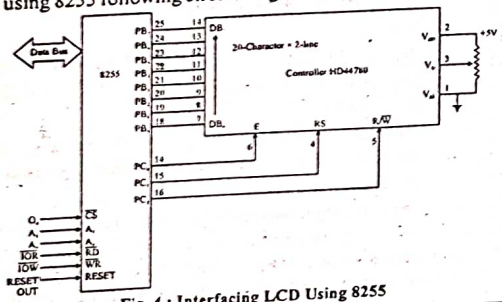


Fig. 4 : Interfacing LCD Using 8255

This LCD can be interfaced with a 4-bit and 8-bit data bus. The pin diagram of LCD includes 8 data lines (DB_7–DB_0), three control signals (RS-Register select R/ $\overline{W}$ and enable). The LCD controller has two 8-bit control registers-IR (Instruction Register) for commands and a data register (DR) for data. Microprocessor write commands when RS is low and write data when RS is high.

Timing diagram for read and write instruction

- Fig. shows the timing diagram of the read and write operation.

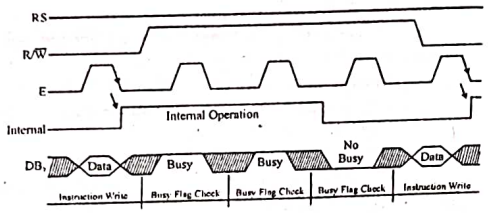


Fig. 5 : Data Transfer Timing Diagram

- Timing diagram shows that after selection of a register (IR or DR) by RS, the R/W signal is asserted and enable signal (E) is pulsed low-high-low.
- Now at the trailing edge of the enable signal, the controller sets the flag (DB_7) and executes the internal operation of either writing a command or displaying a character.
- During the internal operation, MPU should keep checking the busy flag (DB_7) by asserting the enable signal (E) low-high-low until the controller resets the flag.
- On the reset of DB_7, the MPU can write the next command, character or data.

Prob.11 *Write short note on MC 1488 and MC 1489.* [R.T.U. 2010

Sol. MC 1488 and 1489

MC-1488 : The MC-1488 is a quad line driver, used to interface DTE and DCE through RS-232C bus standard. It convents TTL logic 1(> 3. 4 V) into approximately – 9V and TTL logic 0 (< 0.2 V) into + 9V. The pin diagram of MC-1488 is shown in fig. (a) below.

The MC-1488 contains four drivers and each driver converts a TTL level signal to RS-232C level signal.

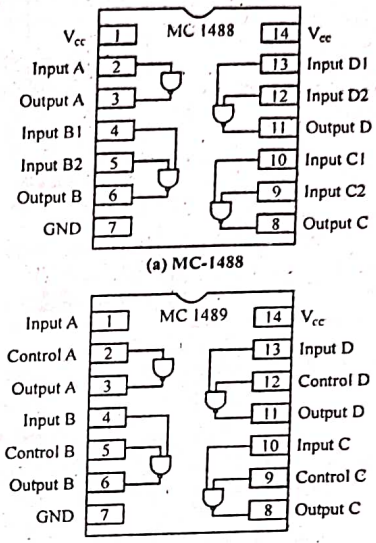


Fig. : Pin Diagram of (a) MC-1488 (b) MC-1489

MC-1489 : The MC-1489 is a quad line receiver to interface DTE and DCE through RS-232C bus standard. It convents RS-232 output into TTL -compatible logic. *i.e.*, – 9V into TTL-logic 1(approximately +3.4V) and +9V into TTL-logic 0 (approximately 0.2V). The pin diagram of MC-1489 is shown in fig. (b).

The MC-1489 contains four receivers and each receiver converts a RS-232C level signal to TTL level signal.

PART-C

Prob.12 *Explain RS422A Standard in detail.* [R.T.U. 2019]

Sol. RS 422A : High speed data transmission between computer system components and peripherals over very long distance, under high noise condition is very difficult with single ended driver and receivers. As an improvement over single ended drivers and receivers EIA standard introduced RS 422A which uses low impedance differential signals. This standard achieves transmission rates from 100 Kbps to 100 Mbps. RS 422A is used for balanced transmission. It support transmission distance which is gerater than 1000 m. RS 422A use a low impedance signal which is a differential signal.

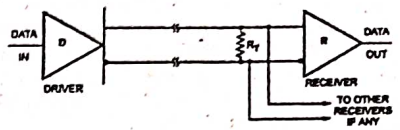


Fig. 1 : RS 422A interface circuit

A basic RS 422A circuit has a data line driver, differential tranmission line and loads, where a load may consists of one or more receivers (R) and the line termination resistor (R_T), as shown in Fig.1.

Following tables-1 and 2 shows the driver and receiver requirements for RS 422A

Table 1 : RS 422A driver requirements

Parameter	Specification
Driver output impedance	100 Ω or less
Output differential voltage	2.0 V to 6.0 V
Difference between opposite polarity differential output voltages	less than 0.4 V
The driver output offset voltage (V_{OS}), measured from the junction of the two 50 Ω terminator and driver ground	less than 3.0 V
Driver output current	less than 150 mA
Output off-state leakage current	less than 100μA
Output voltage transition time (tr)	less than 20 ns
Overshoot and undershoot magnitudes	less than 10% of V_{SS} where V_{SS} is the difference between the two steady state values of the output

Table 2 : RS 422 A receiver requirements

Parameter	Specification
Differential data input threshold sensitivity	± 200 mV
Input impedance	Greater than or equal to 4 kΩ

Drivers and receivers : The SN75172 B and SN75175 are driver and receiver for EIA RS 422A standard, respectively.

Prob.13 *Explain Parallel Interface-Centronics & IEEE 488 in detail.* [R.T.U. 2019]

Sol. Centronics : A standard 36pin parallel interface for connecting printers and other devices to a computer. It defines the plug, socket and signals used and transfers data asynchronously up to 200 Kbytes/sec. The plug has 18 contacts each on the top and bottom. The socket contains one opening with matching contacts.

This de facto standard was developed by Centronics Corporation, maker of the first successful dot matrix printers. Centronics Data Computer Corporation was a pioneering American manufacturer of computer printers, now remembered only for the parallel interface that bears their name.

The Centronics Interface provides a handshake protocol between a computer and a printer and supports a maximum data transfer speed of about 100 kb/s. The printer side of the interface is a 36 pin connector and the PC side is a 25 pin D type connector. The PC uses 36 pin flat cable in which every alternative wire is for the ground. Most of the signals should have twisted pair wiring in the cable. The signals are TTL level signals and the twisted pair return ground wire for each signal is connected to the signal ground level. To prevent noise effects the twisted pair wires are shielded and the shield is connected to the chassis ground in the system box.

Figure shows the signals in the centronics interface. Since a PC uses 25 pin connector, there is a shortage of pins for four twisted pair return signals. Usually, four data signals don't have twisted pair wires. The pin configurations for both printer side and PC side are listed.

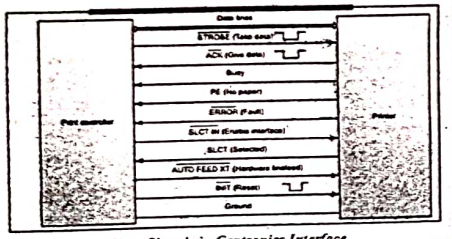


Fig. : Signals in Centronics Interface

IEEE 488 : It is known as General Purpose Interface Bus (GPIB). GPIB is designed by Hewlette Packard so it is also called as *HPIB (Hewlette Packard Interface Bus)*. The GPIB is accepted as standard by IEEE.

So it has been give a number IEEE 488. The GPIB is basically designed to interface instruments to computers such as signal generator, digital voltmeters, printers, etc. as shown in Fig.

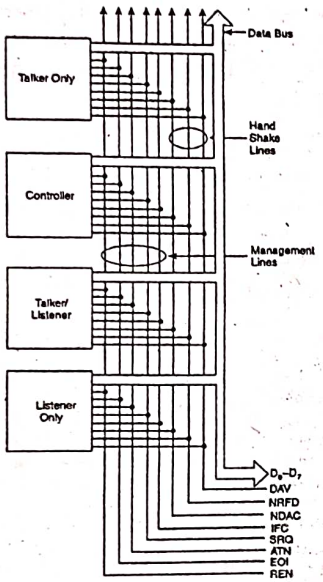


Fig. : IEEE 488 (GPIB Structure)

The interfacing device for IEEE 488 can be divided int 3 categories :

1. Listener
2. Talker
3. Controller

1. Listener : These are instruments which only accep data from other instruments. For example, printers, display programmable power supply, programmable signal generat etc. All the examples of listener will be an output devices.

2. Talker : These are instruments which only send data to other instruments. For example, Digital voltmeter, tap

reader, frequency counter etc. All the examples of talker will be an input devices.

3. Controller : These are instruments which initialises and controls the other instruments in the system. It also gives service to all devices and decides the sequence for the devices.

The IEEE 488 has three group of buses as shown in Fig.

(i) Bidirectional data lines
(ii) Management lines
(iii) Handshake signals

(i) Bidirectional Data Lines : These are eight lines used to transfer 8-bit of information.

(ii) Management Lines : The GPIB has five management lines which are used to manage data transfer on $D_0 - D_7$, the different signals are :

(a) IFC (b) ATN (c) SRQ (d) REN (e) EOI.

(a) IFC (Interface Clear) : This is an output signal generated by controller. It is used to force the instruments in system to a known or predetermined state. It is similar to system reset signal issued by controller to reset the system.

(b) ATN (Attention) : This is an output signal generated by controller. It is used to differentiate between data or status from address commands.

(c) SRQ (Service Request) : This is an input signal to the controller generated by peripherals. It is an interrupt signal requesting service of controller.

(d) REN (Remote Enable) : This is an output signal by controller used to enable the instrument to be controlled by the system. If REN is not generated, the instrument will be controlled by its front panel controls.

(e) EOI (End of Identify) : This is an input signal to controller generally generated by talker to indicate end of data transfer.

(iii) Handshake Signals : The IEEE 488 has three handshake signals which helps in data transfer, these are :

(a) DAV (Data Valid)
(b) NRFD (Not Ready for Data)
(c) NDAC (Not Data Accepted)

The signals DAV, NRFD and NDAC are open collector active low signals. Since it has open collector structure any listener can pull NRFD low to indicate it is not ready for data. The NRFD line will not go high to indicate its readiness unless all the addressed listeners has released it.

Similarly, any listener can pull NDAC low to indicate it has not accepted data. The NDAC line will not go high to indicate data accepted unless all listener accepts data.

Prob.14 *Explain the organization and architecture of 8251 (USART) with a functional block diagram.* [R.T.U. 2019]

OR

How can you interface USART 8251 with microprocessor 8085? Explain.

Sol. The 8251 Universal Synchronous/Asynchronous Receiver/Transmitter (USART) package in a 28 pin dual-in-line package made by Intel. It is typically used for serial communication. The 8251 in I/O mapped mode is shown in fig.

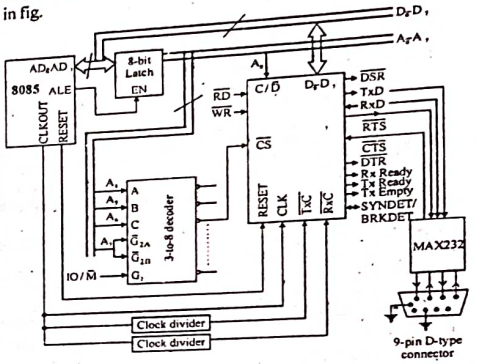


Fig.

1. Address lines A_6–A_4 are used to generate the chip select signal via a 3-to-8 decoder.
2. A_7 and IO/$\overline{M}$ are used as enable for the decoder.
3. A_0 of 8085 is connected to C/$\overline{D}$ of 8251 to provide internal addresses.
4. The data lines D_0–D_7 are connected to D_0–D_7 of the processor to achieve parallel data transfer between 8085 and 8251.
5. The output clock of 8085 is divided by suitable clock dividers to get a suitable clock for serial transmission/ reception.
6. 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.

7. The device which requires serial communication with processor can be connected to this 9-pin D-type connector using 9-core cable.
8. The signals TxEMPTY, TxRDY, and RxRDY can be used as interrupt signals to initiate interrupt driver data transfer between 8085 and 8251.

I/O addresses of 8251 interfaced to 8085 is

Internal device of 8251	Binary address								Hex address
	Decoder input & 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

Prob.15 ***Explain serial communication controller USART 8251. Explain its all modes with example.***

[R.T.U. 2017]

Sol. Serial Communication: Using 8251 : 8251 is a Universal Synchronous and Asynchronous Receiver and Transmitter compatible with Intel's processors. This chip converts the parallel data into a serial stream of bits suitable for serial transmission. It is also able to receive a serial stream of bits and convert it into parallel data bytes to be read by a microprocessor.

1. Basic Modes of Data Transmission

(a) Simplex
(b) Duplex
(c) Half Duplex

(a) Simplex Mode : Data is transmitted only in one direction over a single communication channel. For example, the processor may transmit data for a CRT display unit in this mode.

(b) Duplex Mode : In duplex mode, data may be transferred between two transreceivers in both directions simultaneously.

(c) Half Duplex Mode : In this mode, data transmission may takes place in either direction, but at a time data may be transmitted only in one direction. A computer may communicate with a terminal in this mode. It is not possible to transmit data from the computer to the terminal and terminal to computer simultaneously.

The data buffer interfaces the internal bus of the circuit with the system bus. The read / write control logic controls the operation of the peripheral depending upon the operations initiated by the CPU decides whether the address on internal data bus is control address / data address. The modem control unit handles the modem handshake signals to coordinate the communication between modem and USART.

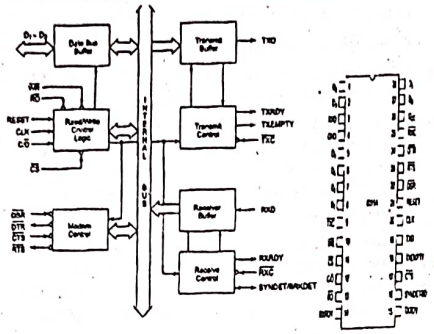


Fig. : Serial Communication Interface 8251

The transmission control unit transmits the data byte received by the data buffer from the CPU for serial communication. The transmission rate is controlled by the input frequency. Transmit control unit also derives two transmitter status signals namely TXRDY and TXEMPTY which may be used by the CPU for handshaking.

The transmit buffer is a parallel to serial converter that receives a parallel byte for conversion into a serial signal for further transmission. The receiver control unit decides the receiver frequency as controlled by the RXC input frequency. The receiver control unit generates a receiver ready (RXRDY) signal that may be used by the CPU for handshaking. This unit also detects a break in the data string while the 8251 is in asynchronous mode. In synchronous mode, the 8251 detects SYNC characters using SYNDET/BD pin.

2. Signal Description of 8251

- **D0 – D7:** This is an 8-bit data bus used to read or write status, command word or data from or to the 8251A.
- **C / D:** (Control Word/Data): This input pin, together with RD and WR inputs, informs the 8251A that the word on the data bus is either a data or control word/ status information. If this pin is 1, control / status is on the bus, otherwise data is on the bus.
- **RD:** This active-low input to 8251A is used to inform it that the CPU is reading either data or status information from its internal registers. This active-low input to 8251A is used to inform it that the CPU is writing data or control word to 8251A.

- **WR:** This is an active-low chip select input of 8251A. If it is high, no read or write operation can be carried out on 8251. The data bus is tristated if this pin is high.
- **CLK:** This input is used to generate internal device timings and is normally connected to clock generator output. This input frequency should be at least 30 times greater than the receiver or transmitter data bit transfer rate.
- **RESET:** A high on this input forces the 8251A into an idle state. The device will remain idle till this input signal again goes low and a new set of control word is written into it. The minimum required reset pulse width is 6 clock states, for the proper reset operation.
- **TXC (Transmitter Clock Input):** This transmitter clock input controls the rate at which the character is to be transmitted. The serial data is shifted out on the successive negative edge of the TXC.
- **TXD (Transmitted Data Output):** This output pin carries serial stream of the transmitted data bits along with other information like start bit, stop bits and parity bit, etc.
- **RXC (Receiver Clock Input):** This receiver clock input pin controls the rate at which the character is to be received.
- **RXD (Receive Data Input):** This input pin of 8251A receives a composite stream of the data to be received by 8251 A.
- **RXRDY (Receiver Ready Output):** This output indicates that the 8251A contains a character to be read by the CPU.
- **TXRDY - Transmitter Ready:** This output signal indicates to the CPU that the internal circuit of the transmitter is ready to accept a new character for transmission from the CPU.
- **DSR - Data Set Ready:** This is normally used to check if data set is ready when communicating with a modem.
- **DTR - Data Terminal Ready:** This is used to indicate that the device is ready to accept data when the 8251 is communicating with a modem.
- **RTS - Request to Send Data:** This signal is used to communicate with a modem.
- **TXE- Transmitter Empty:** The TXE signal can be used to indicate the end of a transmission mode.

3. Operating Modes of 8251

(a) Asynchronous mode
(b) Synchronous mode

(a) Asynchronous Mode (Transmission) : When a data character is sent to 8251A by the CPU, it adds start bits prior to the serial data bits, followed by optional parity bit and stop bits using the asynchronous mode instruction control word format. This sequence is then transmitted using TXD output pin on the falling edge of TXC.

Asynchronous Mode (Receiver) : A falling edge on RXD input line marks a start bit. The receiver requires only one stop bit to mark end of the data bit string, regardless of the stop bit programmed at the transmitting end. The 8-bit character is then loaded into the parallel I/O buffer of 8251.

RXRDY pin is raised high to indicate the CPU that a character is ready for it. If the previous character has not been read by the CPU, the new character replaces it, and the overrun flag is set indicating that the previous character is lost.

(b) Synchronous Mode (Transmission) : The TXD output is high until the CPU sends a character to 8251 which usually a SYNC character. When CTS line goes low, the first character is serially transmitted out. Characters are shifted out on the falling edge of TXC. Data is shifted out at the same rate as TXC, over TXD output line. If the CPU buffer becomes empty, the SYNC character or characters are inserted in the data stream over TXD output.

Synchronous Mode (Receiver) : In this mode, the character synchronization can be achieved internally or externally. The data on RXD pin is sampled on rising edge of the RXC. The content of the receiver buffer is compared with the first SYNC character at every edge until it matches. If 8251 is programmed for two SYNC characters, the subsequent received character is also checked. When the characters match, the hunting stops. The SYNDET pin set high and is reset automatically by a status read operation. In the external SYNC mode, the synchronization is achieved by applying a high level on the SYNDET input pin that forces 8251 out of HUNT mode. The high level can be removed after one RXC cycle. The parity and overrun error both are checked in the same way as in asynchronous mode.

Command Instruction Definition

The command instruction controls the actual operations of the selected format like enable transmit/receive, error reset and modem control. A reset operation returns 8251 back to mode instruction format.

Status Read Definition

This definition is used by the CPU to read the status of the active 8251 to confirm if any error condition or other conditions like the requirement of processor service has been detected during the operation.

Prob.16 *(a)Explain RS232 interface protocol.*
*(b) Explain interfacing of 4*4 key pad with 8085 using 8255.* [R.T.U. 2017]

Sol.(a) *Refer to Prob.5.*

Sol.(b) Port A is used as output port for selecting a row of keys while Port B is used as an input port for sensing a closed key. Thus the keyboard lines are selected one by one through port A and the port B lines are polled continuously till a key closure is sensed. The routine DEBOUNCE is called for key debouncing. The key code is depending upon the selected row and a low sensed column.

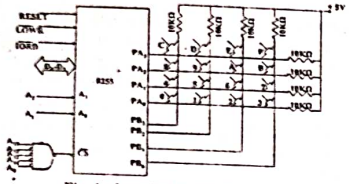


Fig. 1 : Interfacing 4*4 Keyboard

The higher order lines of port A and port B are left unused. The address of port A and port B will respectively 8000H and 8002H while address of CWR will be 8006H. The flow chart of the complete program is as given. The control word for this problem will be 82H. Code segment CS is used for storing the program code.

Key Debounce : Whenever a mechanical push-button is pressed or released once, the mechanical components of the key do not change the position smoothly, rather it generates a transient response .

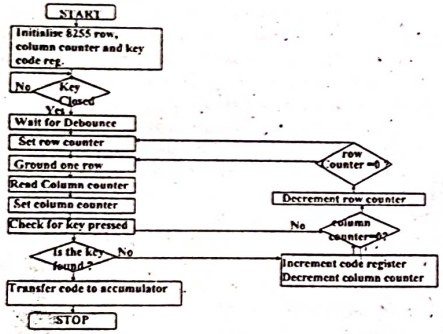


Fig. 2 : Flow Chart

- These transient variations may be interpreted as the multiple key pressure and responded accordingly by the microprocessor system.
- To avoid this problem, two schemes are suggested: the first one utilizes a bistablemultivibrator at the output of the key to debounce.
- The other scheme suggests that the microprocessor should be made to wait for the transient period (usually 10ms), so that the transient response settles down and reaches a steady state.

Prob.17 *Write short note on USART 8251.* [R.T.U. 2016]
OR
Draw the internal Block diagram of 8251 USART and explain its initialization process. [R.T.U. 2015]

Sol. USART 8251 : Fig. 1 shows the block diagram of USART 8251. It includes : data bus buffer read/write control logic, modem control, transmit buffer, transmit control, receiver buffer and receiver control.

1. 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.

2. Read/Write Control Logic

This functional block accepts inputs from the system control bus and generates control signals for overall device operation. It decodes control signals on the 8085 control bus into signals 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.

The input signals to the control logic are as follows :

(i) $\overline{CS}$ – **Chip Select :** When this signal goes low, the 8251A is selected by the MPU for communication. This is usually connected to a decoded address bus.

(ii) C/$\overline{D}$ – **Control/Data :** When this signal is high, the control register or the status register is addressed. When it is low, the data buffer is addressed. The control register and the status register are differentiated by $\overline{WR}$ and $\overline{RD}$ signals, respectively.

(iii) $\overline{WR}$ **–Write :** When this signal goes low. The MPU either writes in the control register or sends output to the data buffer. This is connected to $\overline{IOW}$ or $\overline{MEMW}$.

(iv) $\overline{RD}$**–Read :** When this signal goes low, the MPU either reads a status from the status register or accepts (inputs) data from the data buffer. This is connected to either $\overline{IOR}$ or $\overline{MEMR}$.

(v) **RESET–Reset :** A high on this input resets the 8251 A and forces it into the idle mode.

(vi) **CLK–Clock :** This is the clock input, usually connected to the system clock. This clock does not control either the transmission or the reception rate. The clock is necessary for communication with the microprocessor.

Control Register : This 16-bit register for a control word consists of two independent bytes: the first byte is called the **mode instruction** (word) and the second byte is called the **command instruction** (word). This register can be accessed as an output port when the C/$\overline{D}$ pin is high.

Status Register : This input register checks the ready status of a peripheral. This register is addressed as an input port when the C/$\overline{D}$ pin is high; it has the same port address as the control register.

3. 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 $\overline{TxC}$.

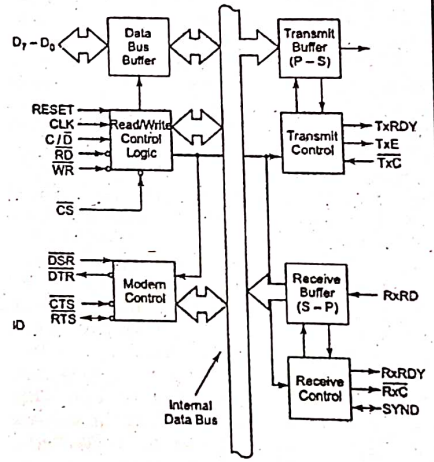


Fig. 1

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 byte 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, $1^1/_2$ or 2 STOP bits. In synchronous mode no extra bits (other than parity, if enable) are generated by the transmitter.

4. 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 function.

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 byte 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.

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

5. Receiver Buffer

The receiver accepts serial data on the RxD line, converts this serial data to parallel format, checks 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 on the 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 line is 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 receiver input register. The data is then transfered 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.

6. Receiver Control

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 byte from receiver buffer is read by the CPU.

$\overline{RxC}$ (Receiver Clock) : This clock controls the rate at which the character is to be received by USART in the synchronous mode. $\overline{RxC}$ is equivalent to the baud rate and is supplied by the modem. In asynchronous mode $\overline{RxC}$ is 1, 15 or 64 times the baud rate. The clock division is programmable. It can be programmed by writing proper mode word in the mode set register.

7. 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 $\overline{RTS}$ and $\overline{DTR}$ and the reception of $\overline{CTS}$ and $\overline{DSR}$.

In addition, a general purpose inverted output and a general purpose input are provided. The output is labeled $\overline{DTR}$ and the input is labeled $\overline{DSR}$. $\overline{DTR}$ 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 $\overline{DTR}$ indicates that the terminal is ready to communicate and $\overline{DSR}$ indicates that it is ready for communication.

Pin Description

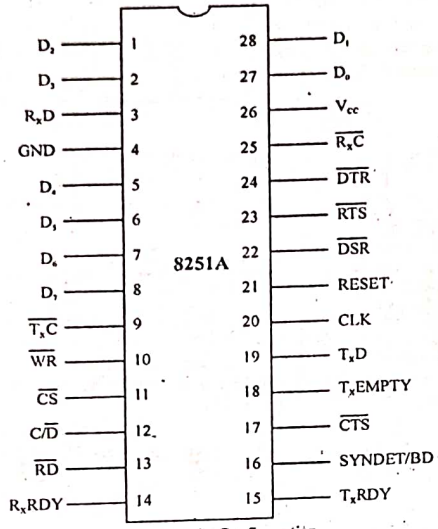


Fig. 2 : Pin Configuration

Table 1 : The 8251: Pin Description

Pin	Name	Pin	Name
RESET	Reset	$\overline{DTR}$	Data terminal ready
CLK	Clock pulse (TTL)	$\overline{DSR}$	Data set read
$\overline{RD}$	Read Data Command	SYNDET/ $\overline{BD}$	Sync detect/ Break detect
$\overline{WR}$	Write data or Control command	$\overline{RTS}$	Request to send data
C/$\overline{D}$	Control or data is to be written or read	$\overline{CTS}$	Clear to send data
$D_7 - D_0$	Data bus(8 bit)	TxE	Transmitter empty (no characters to transmit)
$\overline{CS}$	Chip select	GND	Ground
$\overline{TxC}$	Transmitter clock	Vcc	+5 Volt supply
$\overline{TxD}$	Transmitter data	TxRDY	Transmitter ready
$\overline{RxC}$	Receiver clock	RxRDY	Receiver ready
$\overline{RxD}$	Receiver data		

Mode Word Format

Table 2

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0 = Not valid	X	0 = No parity	0	0 = 5 bits	00	SYN Mod
0	1 = Stop bit	0	1 = Odd parity	0	1 = 6 bits	01	ASYN ×
1	0 = 1$\frac{1}{2}$ Stop bit			1	0 = 7 bits	10	ASYN × 16
1	1 = 2 Stop Bit	1	1 = Even Parity	1	1 = 8 bits	11 = A	SYN × 6
Framing Control		Parity Control		Character Length		Baud Rate Factor	

Command Word Format

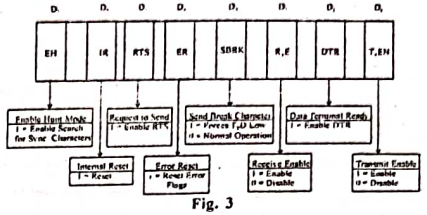


Fig. 3

Status Word Format

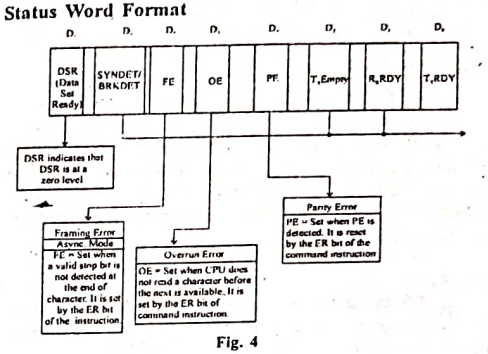


Fig. 4

Prob.18 *Write short note on :*
(a) RS232 C and RS 422 A
(b) Centronics
(c) IEEE 488 [R.T.U. 2016]

Sol.(a) RS232 C : *Refer to Prob.5.*
RS 422 A : *Refer to Prob.12.*
Sol.(b) Centronics : *Refer to Prob.13.*
Sol.(c) IEEE 488 : *Refer to Prob.13.*

Prob.19 *(a) Compare RS-422A and RS-423A serial data standards.*
(b) Briefly describe the communication standards RS-232 C and I EEE 488 by showing configurations. [R.T.U. 2015]

Sol. (a) RS-422A is widely used a differential data transmission standard designed to transmit data at high rate of upto 10 Mb/sec and over distances upto 4000 ft. It is also specified for multi-drop (party-line) applications where only one driver is connected to and transmits on a bus of upto 10 receivers. It helps nullify the effect of ground shifts and noise signals which can appear in a network. It doesn't support variable skew-rate. It is 5 times faster than the single-ended RS-232 data transmission standard which was introduced in 1962. It is useful in applications where noise immunity is an issue. It is often used in the construction of "Quasi" multi-drop networks (4-wire). It is also commonly used as extender of RS-232.

On the other hand RS-423A is a single-ended data transmission standard which is a successor to the RS-232 standard. There were many computers in the old days which used serial ports with RS-232 interface. RS-423A was used back then because of its backward compatibility with RS-232. It also allows higher data transmission rates and longer cable lengths. The transmission rate is usually less than RS-422A (around 100 Kb/s). The above rates are achieved due to its support for variable skew-rate. The skew-rate is set depending on the transmission rate and cable length. The skew-rate for RS-232 was fixed at the speed of 30 V/µs. It needs a system with noise immunity otherwise it won't be able to transmit data at high rate.

Sol.(b) Standard RS 232C and IEEE 488 : The bus is a communication path between the microprocessors or peripherals, it is nothing but a group of wires to carry bits.

Standard of Bus : The serial I/O technique is commonly used to interface terminals, printers, and modems.

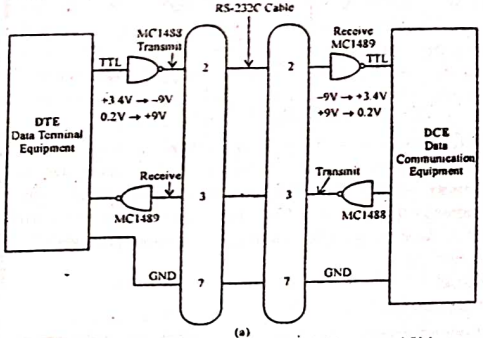


Fig. : Minimum Configuration of RS-232C Signals and Voltage Levels

These peripherals and computers are designed and manufactured by various manufacturers. Therefore, a common understanding must exist, among various manufacturing and user groups, that can ensure compatibility among different equipments. When this understanding is defined and generally accepted in industry (and by users), it is known as a **Standard**. A standard is normally defined by a professional organization (such as IEEE–Institute of Electrical and Electronics

Engineers); however, occasionally, a widespread practice can become a de facto standard. A standard may include such items as assignment of pin positions for signals, voltage levels, speed of data transfer, length of cables and machanical specifications.

In serial I/O, data can be transmitted as either current or voltage. Typically, 20 mA (or 60 mA) current loops are used in teletype equipment. When a teletype is marking or at logic 1, current flows, when it is at logic 0 (or Space), the current flow is interrupted. The advantage of the current loop method is that signals are relatively noise-free and are suitable for transmission over a distance.

When data are transmitted as voltage, the commonly used standard is known as RS 232C. It is defined in reference to Data Terminal Equipment (DTE) and Data Communication Equipment (DCE)–terminal and modem–as shown in fig.; however its voltage levels are not compatible with TTL logic levels. The rate of data transmission in RS-232C is restricted to a maximum of 20 kbaud and a distance of 50 ft.

For high speed data transmission, two new standards like –RS-422A and RS-423A have been developed in recent years; however, they are not yet widely used

To appreciate the difficulties and confusion in this standard, one has to examine its historical background. The RS-232 standard was developed during the initial days of computer time sharing, long before the existence of TTL logic, and its primary focus was to have compatibility between a terminal and a modem. However, the same standard is now being used for communications between computers and peripherals, and the roles of a data terminal and a modem have become ambiguous. Should a computer be considered a terminal on a modem? The answer is that it can be either. Therefore the lines used for transmission and reception will differ, depending on the manufacturer's role-definition of its equipment.

RS-232C : *Refer to Prob.5.*

IEEE 488 : *Refer to Prob.13.*

Prob.20 ***Write short note on :***

(a) Interfacing a matrix keyboard using 8255

(b) 8085 MPU design. **[RTU 2012]**

Sol. (a) Interfacing a Matrix Keyboard using 8255 : Fig. 1 shows a matrix keyboard with 20 keys, the keyboard has five rows and four columns. The 1st 16 Keys in a sequence will represent data 0 to F in Hex and the remaining four will represent various functions such as store and execute.

The circuit shows that the rows are connected to port C and column are connected to port B of 8255.

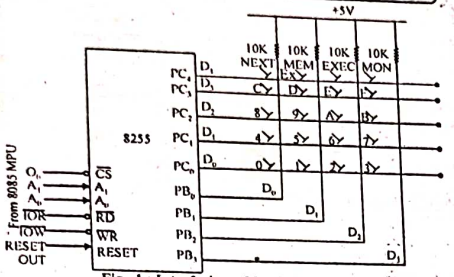


Fig. 1 : Interfacing a Matrix Keyboard

In this fig. 1, the columns and rows make contact only when a key is pressed; otherwise they remain high (+5V). When a key is pressed, the key must be identified by it column and row and the intersection of the column and row must change from high to low. This can be accomplished as explained in following steps :

1. Ground all the rows by sending logic 0 through the output port.
2. Check the columns by reading the input port. If no key is pressed, all columns remain high continue to repeat step 1 and 2 until the reading indicates a change
3. When one of the keys is pressed, the corresponding column goes low, identify and decode the key.

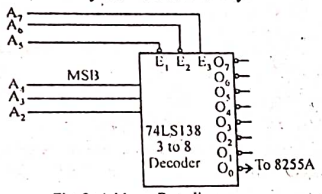


Fig. 2: Address Decoding

Sol. (b) 8085 MPU Design

The MPU design can be divided into the following segments :

(i) Address Bus
(ii) Data Bus
(iii) Control Signals
(iv) Frequency and Power Requirements
(v) Externally Triggered Signals (Reset, Interrupts etc.)

(i) Address Bus : The 8085 has a multiplexed bus; it must be demultiplexed so that general – purpose memory devices can be used. We must use bus drivers to provide sufficient driving capacity.

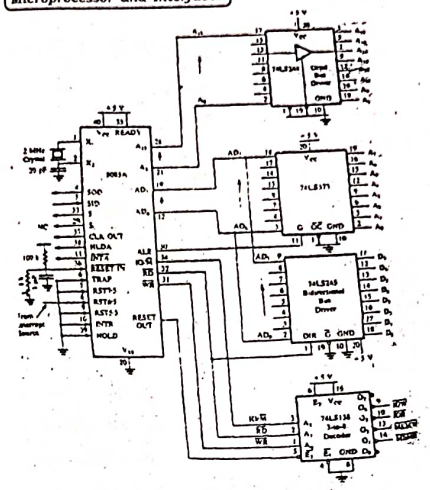


Fig. 1

Fig. (1) shows the 74LS244, an octal bus driver used with high order address bus to increase its driving capacity. The 8085 buses can source 400μA (I_{OH} = 400μA) and link 2mA (I_{OL} = 2mA) of current; they can drive one TTL logic load.

The low order address bus ($A_7 - A_0$) is demultiplexed by using the ALE signal (Address Latch Enable) and latch 74LS373. To demultilexing the address bus. The 74LS373 can serve as a bus driver.

(ii) Data Bus : Fig. (1) shows the 74LS245 as an 8-bit bi-directional bus driver to increase the driving capacity of data bus. It has 8 bi-directional data lines, the direction of the data flow is determined by the direction control line (DIR).

The direction of the data flow is determined by connecting $\overline{RD}$ signal from the MPU to the DIR Signal. When the MPU is writting to peripherals, the $\overline{RD}$ is high and data flow from the MPU to peripherals. When it is reading from peripherals, the RD is low and data flow toward the MPU.

(iii) Control Signals : The 8085 generates three signals :

(a) IO/M (I/O or Memory)
(b) RD (Read)
(c) WR (Write)

Fig. (1) shows that 3 signals – IO/M, $\overline{RD}$ and $\overline{WR}$ - are used as inputs to the 74LS138 3 to 8 decoder to generate four control signals – $\overline{IOR}$, $\overline{IOW}$, $\overline{MEMR}$ and $\overline{MEMW}$. These signals can be used for interfacing with any peripherals.

(iv) Frequency and Power Requirements : The 8085 can operate with a maximum clock frequency of 3 MHz. The 8085 has two clock inputs; X_1 and X_2 at pins 1 and 2. This input can be driven with a crystal or LC turned circuit, or RC Network.

Fig. (2) shows a 2MHz Crystal with a 20pF capacitor to drive the clock inputs. This input frequency is divided in half internally and the system will run on 1MHz clock frequency.

Fig. (3) shows an alternative method of providing a clock input using a RC network.

The 8085 and other components used in this system require one power supply with + 5V.

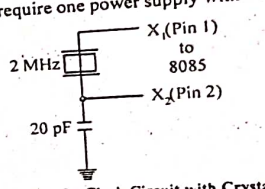


Fig. 2 : Clock Circuit with Crystal

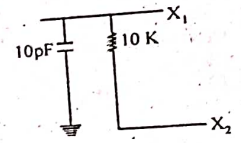


Fig. 3 : RC Clock Circuit

(v) Externally Triggered Signals : The 8085 has provision for four external input signals : (a) Reset, (b) Interrupt, (c) Hold and (d) Ready.

(a) Reset : The $\overline{RESETIN}$ is an active low signal used to reset the system. Fig. (4) shows the reset circuit. This is an RC network with a sufficient long time constant. When the Reset key is pushed the $\overline{RESETIN}$ goes low and slowly rises to +5V.

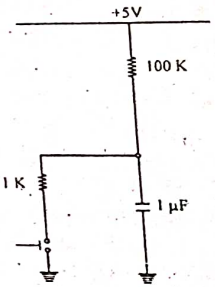


Fig.4 : Reset-Circuit

(b) Interrupts : The 8085 has five interrupt signals, all of them are active high. In this system RST6.5 will be used; the other need to be grounded as shown in circuit, otherwise the floating interrupt pins can cause the system to malfunction.

(c) Hold : This is an active high signal used in the DMA. This signal is also grounded in this system.

(d) Ready : When the signal is high, it indicates that the memory or peripheral is ready to send or receive data. When ready goes low, the MPU enters the wait state until ready goes high.

Prob.21 *(a) Interface 8 switches and a seven segment display to 8085 through 8255 PPI. Write a program to display the switch when a switch is open. Assume that the 8255 is interfaced in memory mapped I/O technique.*

(b) Specify mode word, command word and status word to transmit data. The T_xC is 153.6 kHz, the requirement of system is :

(i) Asynchronous mode with 9600 baud
(ii) Character length 5 bits
(iii) 1 stop bit
(iv) Even parity check. [R.T.U. 2011]

Sol.(a) Fig.1 shows the multiplexed eight 7-segement display connected in the 8085 system using 8255.

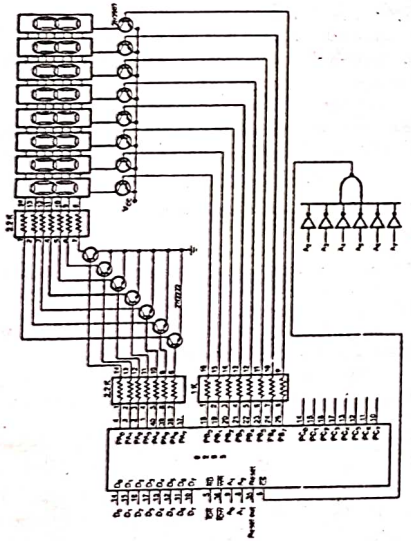


Fig. 1: Interfacing of Multiplexed Eight 7-segment Display using 8255

In this circuit port A and port B are used as simple latched output ports. Port A provides the segment data inputs to the display and port B provides a means of selecting, a display position at a time for multiplexing the display. $A_0 - A_1$ lines are used to decode the addresses for 8255. For this circuit different addresses are :

PA = 00 H PC = 0.2 H
PB = 01 H CR = 03H

The register values are chosen in Fig.1 such that the segment current is 80 mA. This current is required to produce an average of 10 mA per segement as the displays are multiplexed. In this type of display system, only one of the eight display position is 'ON' at any given instant. Only one digit is selected at a time by giving low signal on the corresponding control line. Maximum anode current is 560 mA (7-segments × 80 mA = 560 mA), but the average anode current is 70 mA.

Before going to write the software we must know the control word to program 8255 according to hardware connections. For 8255 Port A and B are used as output ports.

BSR	Mode A		P_A	P_{CU}	Mode B	P_B	P_{CL}	
1	0	0	0	X	0	0	X	= 80H

Fig.2 : Control Word Format for 8255

```
; Software to initialize 8255
    MVI A, 80H   ; Load control word in AL
    OUT CR       ; Load control word in CR
; Subroutine to display message on multiplexed LED display
; Set up registers for display
    MVI B, 08H   ; Load count
    MVI C, 7FH   ; Load select pattern
    LXI H, 6000H ; Starting address of message
; display message
DISP_1 : MOV A, C ; Select digit
    OUT PB
    MOV A, M     ; Get data
    OUT PA       ; Display data
    CALL DELAY ; Wait for some time
    MOV A, C
    RRC
    MOV C, A     ; Adjust selection pattern
    INX H
    DCR B        ; Decrement count
    JNZ DISP_1 ; Repeat 8 times
    RET
```


Sol.(b) Baud rate required = 9600 baud
Frequency T_xC = 153.6 kHz
Baud rate factor = Frequency/ Baud rate
= (153.6/9600)= 16
Baud rate factor required is x 16

Fig. shows the bit pattern for the command word Fig.(a), mode word Fig.(b) and status word format Fig.(c). As only transmitter is used, the status word is used to check T×RDY signal only.

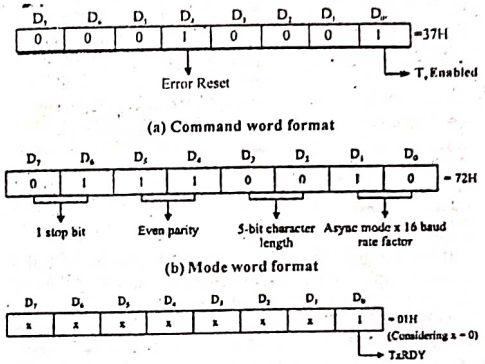


Fig. : Command Word, Mode Word and Status Word Format

Prob.22 *Interface two LEDs using common anode and common cathode technique. Show the complete interface. Write the program to blink them alternately. Assume port addresses in IO mapped IO.* [R.T.U. 2011]

Sol. 7-Segment LED Display : The seven-segment LED display is a multiple display. It can display all decimal digits and some letters. It is very popular among multiple displays as it has the smallest number of separately controlled light emitting diodes (LED). Multiple displays of 9-segment LED, 14-segment LED and dot matrix type are available. These displays give better representation of alphanumeric characters but require complex circuitry.

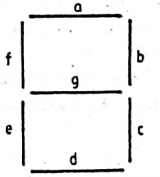


Fig.1: Schematic Diagram of 7-Segment Display

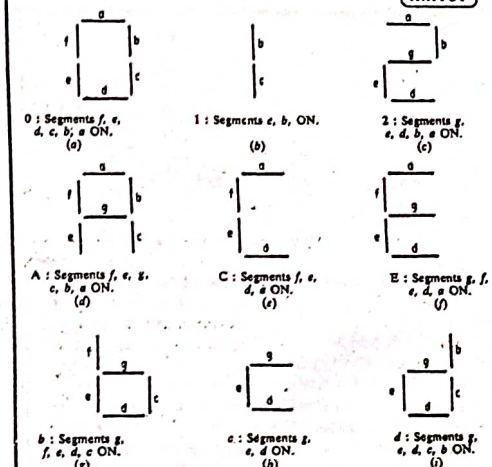


Fig.2: 7-Segment Representation of Digits and Letters

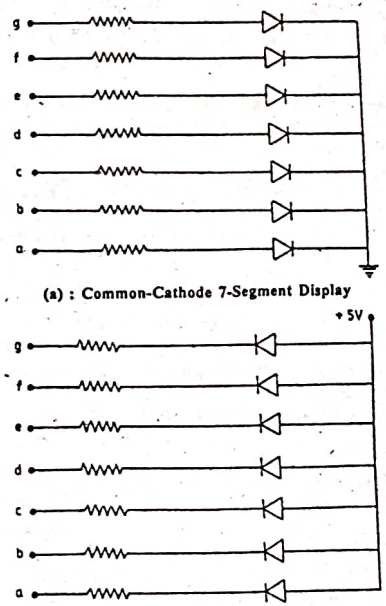


Fig.3

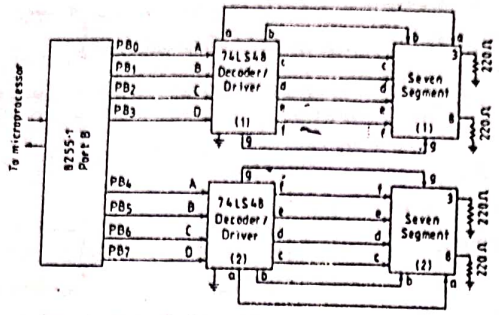


Fig.4 : Interfacing of 7-segment Displays

In seven-segment displays there are seven light emitting diodes (LED) as shown in fig.1. Each LED can be controlled separately. To display a digit or letter the desired segments are made ON as shown in Fig.2(a) to fig.2(i). There are two types of 7-segment displays namely, common-cathode type and common-anode type. In a common-cathode type display all the 7 cathodes of LEDs are tied together to the ground as shown in fig.3(a). When a +5 Vd.c. is applied to any segment, the corresponding diode emits light. Thus applying logic '1' *i.e.*, positive logic to the desired segments, the desired letter or decimal number can be displayed. In a common-anode type display all the 7 anodes are tied together and connected to +5 V supply as shown in fig.3(b). A particular segment will emit light when 0 logic is applied to it.

Display of Decimal Numbers 0 to 9 : A program has been developed to display the decimal numbers 0 to 9, one by one. First of all the decimal number 0 is displayed for some time, then 1 and so on. After displaying all the numbers, the program will repeat the process of displaying the numbers again and again. A delay subroutine has been included in the program. The time for displaying the numbers can be adjusted by adjusting the data for delay subroutine.

The program is as follows :

Memory Address	Machine Codes	Labels	Mnemonics	Operands	Comments
2400	3E, 98		MVI	A, 98	Get control word.
2402	D3, 03		OUT	03	Initialize ports.
2404	21, 00, 25	ABOVE	LXI	H, 2500 H	Get count.
2407	5E		MOV	E, M	Count in register E.
2408	23	LOOP	INX	H	
2409	7E		MOV	A, M	Get next number.
240A	D3, 01		OUT	01	Output at Port B.
240C	06, 0F		MVI	B, 0F	
240E	0E, FF	BEHIND	MVI	C, FF	
2410	16, FF	BACK	MVI	D, FF	
2412	15	GO	DCR	D	
2413	C2, 12, 24		JNZ	GO	Delay subroutine using 3 registers.
2416	0D		DCR	C	
2417	C2, 10, 24		JNZ	BACK	
241A	05		DCR	B	
241B	C2, 0E, 24		JNZ	BEHIND	
241E	1D		DCR	E	Decrement count.
241F	C2, 08, 25		JNZ	LOOP	Go to LOOP to get next number.
2422	C3, 04, 24		JMP	ABOVE	Go to ABOVE to re-start.

DATA

2500—0A	2503—02
2501—00	2504—03
2502—01	2505—04
2506—05	2509—08
2507—06	250A—09
2508—07	

If this program is executed on one displaying unit, the 7-segment display will display the decimal number 0 to 9 one by one. If this program is executed on two displaying units, (Fig.4) the 1st unit displays 0 to 9 and the 2nd unit displays always 0, *i.e.* MSBs of the numbers. MSBs for all the numbers are 0. If we want that the 2nd unit should also display 0 to 9, the data may be fed as given below :

DATA

2500—0A 2506—55
2501—00 2507—68
2502—11 2508—77
2503—22 2509—89
2504—35 250A—93
2505—46

□□□