MICROPROCESSOR AND INTERFACES

IV C.S.

INTRODUCTION TO MICROPROCESSORS 1

PREVIOUS YEARS QUESTIONS

PART-A

Prob.1 ***Specify the size of data, address, memory word, and memory capacity of 8085 microprocessor.*** **[R.T.U. 2024, 2022]**

Sol. Specification of 8085 Microprocessor :

Size of data = 8-bit
Size of address = 16-bit
Memory Word = 8-bit
Memory capacity = Upto 64Kb

Prob.2 ***Draw the flag register configuration for 8085 microprocessor.*** **[R.T.U. 2024]**

Sol.

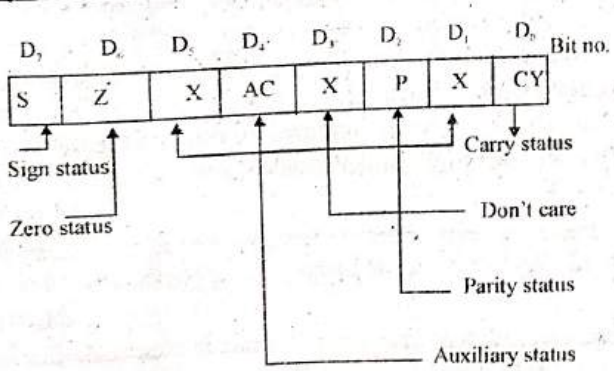


Fig. : Flag Register

Prob.3 ***What is the use of tristate devices in bus oriented system?*** **[R.T.U. 2024]**

Sol. Bus-oriented systems are computer architectures where multiple components share a common communication pathway or "bus." In such systems, it is crucial to control which device is allowed to send data on the bus at any given time to avoid data collisions. Tristate devices play an essential role in this context.

Following are the uses of tristate devices in Bus-Oriented Systems :

1. Enable multiple components to communicate on the same bus by switching between active and high-impedance states.
2. Prevents bus conflicts by ensuring that only one device drives the bus at a time.
3. Used in memory and data transfer operations where different modules need temporary control of the bus.

Prob.4 ***Mention the purpose of HOLD and READY pins of 8085 microprocessor.*** **[R.T.U. 2024]**

OR

What is the purpose of Hold and Ready pins? **[R.T.U. 2022]**

Sol. HOLD Pin : 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.

READY Pin : 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.

Prob.5 ***What is PSW in 8085?*** **[R.T.U. 2023]**

OR

Explain the PSW of 8085 μp. **[R.T.U. 2022]**

Sol. PSW of 8085 Microprocessor : The program status word (PSW) register is an 8-bit register, also known as flag

register. It is 8-bit wide but only 6-bit of it is used. The two unused bits are user-defined flags. Four of the flags are called conditional flags, which means that they indicate a condition which results after an instruction is executed. These four are CY (carry), AC (auxiliary carry), P (parity) and OV (overflow). The bits RS0 and RS1 are used to change the bank registers.

Prob.6 ***What is the clock frequency and duty cycle required for 8085?*** **[R.T.U. 2023]**

Sol. Clock Frequency and Duty Cycle for 8085: The Intel 8085 microprocessor typically operates at a clock frequency ranging from 2MHz to 3MHz. However, specific implementations may vary. The duty cycle for the clock signal in the 8085 microprocessor is usually 50%. This means that the clock signal is high for half of the cycle and low for the other half, ensuring an equal distribution of time for processing and resting.

Prob.7 ***What is the function of ALE signal in 8085 μp?*** **[R.T.U. 2022]**

OR

Write the uses of ALE and HOLD pins of 8085. **[R.T.U. 2023]**

OR

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.

Hold Pin : *Refer to Prob.4.*

Prob.8 ***State the function of following 8085 instructions***
(i) JP ***(ii) JPE***
(iii) JPO ***(iv) JNZ*** **[R.T.U. 2022]**

Sol.

(i) **JP (Conditional Jump) :** The program sequence is transferred to a particular level or a 16-bit address if S=0 (or sign is 0).
Eg. : JP ABC (jump to the level abc if S = 0)

(ii) **JPE (Conditional Jump) :** The program sequence is transferred to a particular level or a 16-bit address if P=1 (or parity flag is 1).
Eg. : JPE ABC (jump to the level abc if P = 1)

(iii) **JPO (Conditional Jump) :** The program sequence is transferred to a particular level or a 16-bit address if P = 0 (or parity flag is 0).
Eg. : JPO ABC (jump to the level abc if P = 0)

(iv) **JNZ (Conditional Jump) :** The program sequence is transferred to a particular level or a 16-bit address if Z=0 (or zero flag is 0).
Eg. : JNZ ABC (jump to the level abc if Z=0)

Prob.9 ***Why data bus is bidirectional?*** **[R.T.U. 2022]**

Sol. The data bus is bidirectional because the data bus has to transfer data among the CPU and memory I/O device.

Prob.10 ***Mention the purpose of SID and SOD lines.*** **[R.T.U. 2022]**

Sol. Purpose of SID and SOD Lines : There are two serial signals, i.e. SID and SOD and these signals are used for serial communication.

SOD Line (Serial Output Data Line) : The output SOD is set/reset as specified by the SIM instruction.

SID (Serial Input Data Line) : The data on this line is loaded into accumulator whenever a RIM instruction is executed.

Prob.11 ***List 16 bit registers of 8085 microprocessor.*** **[R.T.U. 2022]**

Sol. 1. Stack Pointer (SP)
2. Program Counter (PC)

Prob.12 ***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.13 ***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.14 ***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.

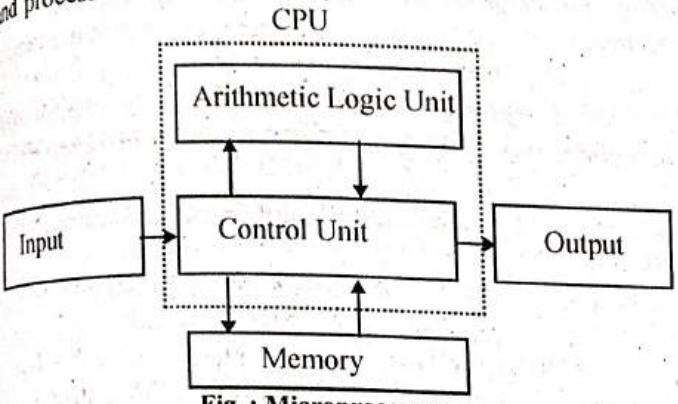


Fig. : Microprocessor

Prob.15 ***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.

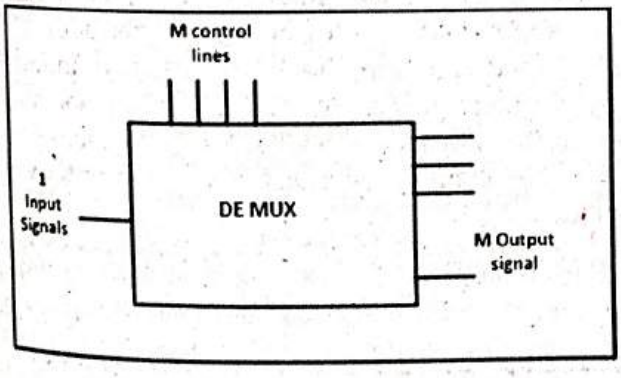


Fig. : De-multiplexer Pin Diagram

Prob.16 ***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.

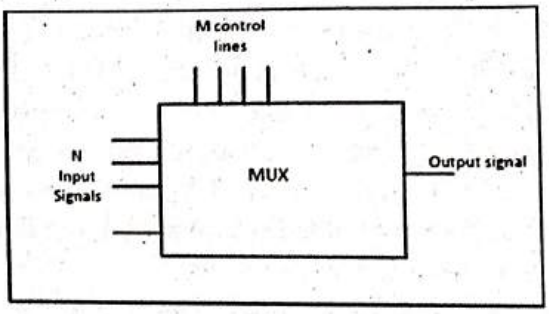


Fig. : Multiplexer Pin Diagram

Prob.17 ***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.

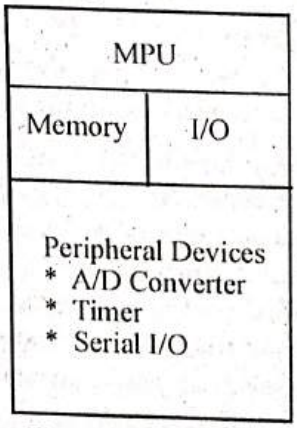


Fig. : Microcontroller

Prob.18 *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.

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.19 *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.20 ***Explain the internal architecture of 8085 microprocessor using neat diagram.*** [R.T.U. 2024]

OR

Draw the architecture of 8085 and explain each block clearly. [R.T.U. 2023]

OR

Draw and explain the functional block diagram of 8085 microprocessor along with the features in detail. [R.T.U. 2019]

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

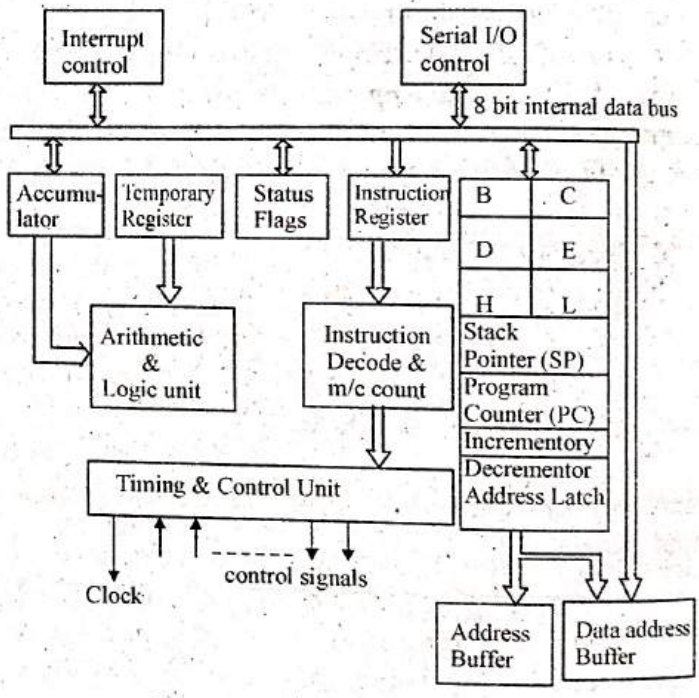


Fig. : Block diagram of 8085 MPU

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

Prob.18 *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.

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.19 *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.20 ***Explain the internal architecture of 8085 microprocessor using neat diagram.*** [R.T.U. 2024]

OR

Draw the architecture of 8085 and explain each block clearly. [R.T.U. 2023]

OR

Draw and explain the functional block diagram of 8085 microprocessor along with the features in detail. [R.T.U. 2019]

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

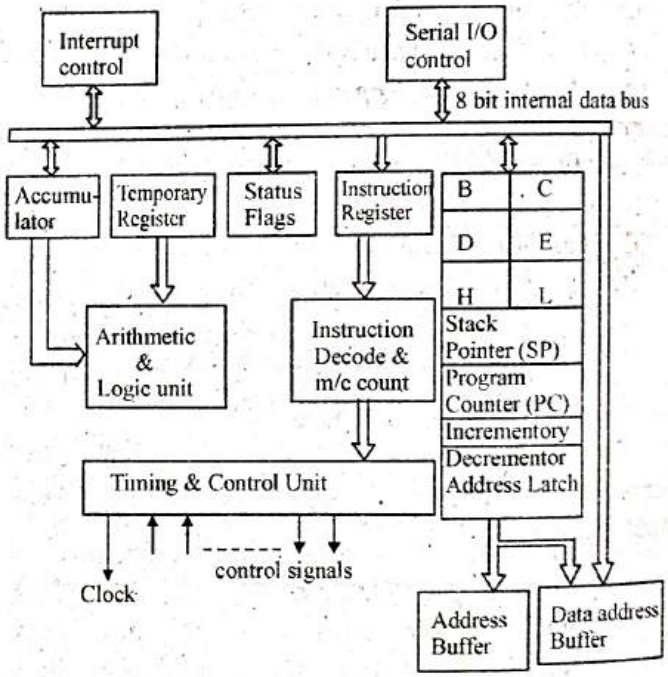


Fig. : Block diagram of 8085 MPU

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. 13.*

(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 : 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.

(ii) 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.

(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.21 ***What are external initiated operations in 8085 microprocessor? Explain briefly.*** **[R.T.U. 2024]**

Sol. External Initiated Operations in 8085 Microprocessor : In the 8085 microprocessor, external initiated operations are those operations that occur due to external signals or hardware components interacting with the microprocessor. These operations help in controlling the processor, managing external devices, and handling interrupts.

Types of External Initiated Operations in 8085:

1. Interrupt Operations : Interrupts are signals from external devices that request the processor to temporarily halt its current execution and perform a specific task.

Types of Interrupts in 8085:

(i) **Maskable Interrupts:** Can be enabled or disabled using software (e.g., RST5.5, RST6.5, RST7.5, INTR).

(ii) **Non-Maskable Interrupts (NMI):** Cannot be ignored (e.g., TRAP).

Example: An external keyboard sending an interrupt signal to the processor.

2. Reset Operations : Resets are used to restart or initialize the processor.

Types of Reset Signals :

(i) **RESET IN :** Resets the program counter to 0000H, clearing the microprocessor.

(ii) **RESET OUT:** Indicates that the processor has been reset.

3. Ready Signal (RDY) Operation : The RDY (Ready) signal is used for synchronization between slow memory devices and the processor. If the RDY pin is LOW, the processor enters a wait state until the memory or I/O device is ready.

4. Hold and HLDA (Hold Acknowledge) Operations :

(i) **HOLD Signal :** An external device (e.g., DMA controller) requests control of the system bus.

(ii) **HLDA (Hold Acknowledge) Signal :** The processor acknowledges the hold request and releases the bus.

5. Direct Memory Access (DMA) Operation : DMA allows external devices to access memory directly without processor intervention. Used for fast data transfer between memory and peripherals.

Prob.22 ***Draw and explain the timing diagram for memory write machine cycle.*** **[R.T.U. 2023]**

Sol. Memory Write Machine Cycle of 8085 :

- A write-to-memory machine cycle is executed by the processor to write a data byte to memory.
- The processor takes 3T states to perform this cycle.
- Instructions that have more than one byte word will use machine cycle after machine cycle to load the opcode.

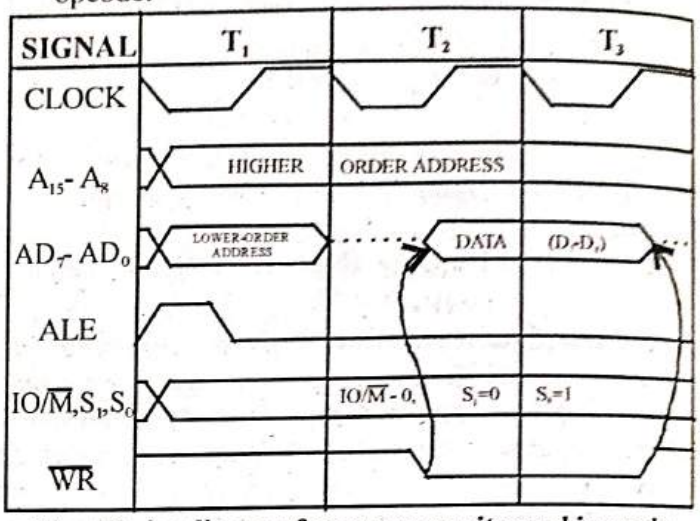


Fig. : Timing diagram for memory write machine cycle

The memory write timing diagram can be explained as below :

- The MP places the 16-bit memory address from the program counter on address bus. At time period T_1, the higher order memory address is placed on the address lines $A_{15} - A_8$. When ALE is high, the lower address is placed on the bus $AD_7 - AD_0$. The status signal $IO/\overline{M}$ goes low indicating the memory operation and two status signals $S_1 = 0$, $S_0 = 1$ to indicate memory write operation.
- At time period T_2, the MP sends $\overline{WR}$ control line to enable the memory write. When memory is enabled with $\overline{WR}$ signal, the data from the processor is placed on the addressed location with ALE low.
- The data is reached at memory location during T_3 state. When data is reached, the $\overline{WR}$ signal goes high. It causes the bus to go into high impedance state.

Prob.23 ***What are the different types of flags available in 8085? Explain in brief.*** **[R.T.U. 2023]**

OR

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.

Flag Register Figure : *Refer to Prob.2.*

In 8085 microprocessor, the flag registor can have a total of eight flags. Thus a flag can be represented by 1 bit of information. But only five flags are implemented in 8085. They are :

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.24 ***Draw the bus architecture of 8085 and explain.*** **[R.T.U. 2023]**

Sol. Bus Architecture/Organization: A bus organization is a group of conducting wires which carries information, all the peripherals are connected to microprocessor through the bus. A system bus is nothing just a group of wires to carry bits.

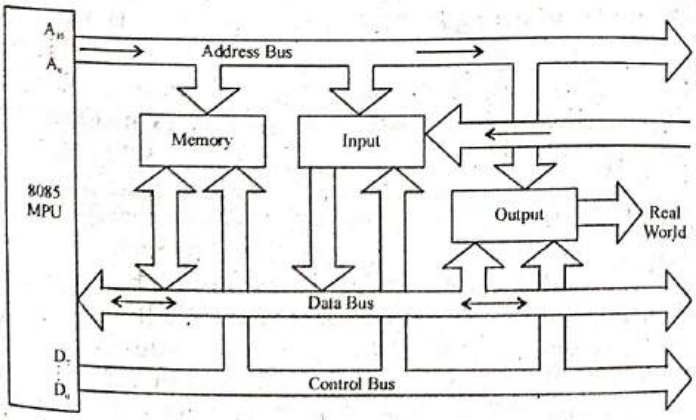


Fig. : 8085 Bus Structure

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*.

Address Bus : *Refer to Prob.18.*

Data Bus : *Refer to Prob.19.*

Control Bus: The connections that carry control information between the CPU and other devices within the computer is called Control Bus. This bus provides timing signals. It is comprised of various single lines that carry synchronization signals. There are three control pins: Input Output/Memory (IO/M′), Read (RD′) and Write (WR′). Using these three pins, four different control signals can be generated: Memory Read, Memory Write, Input Read and Output Write.

Prob.25 ***What are stack related operations? Compare call and push instruction.*** **[R.T.U. 2022]**

Sol. Stack : The stack in an 8085 can be described as a reserved area of the memory in the R/W memory where we can store temporary information. It is a shared resource as it can be shared by the microprocessor and the programmer. Programmers use the stack to store data and the microprocessors use the stack to execute subroutines.

The 8085 has a 16-bit register known as the Stack Pointer. The function of the stack pointer is to hold the starting address of the stack. This address can be decided by the programmer. The stack operates on the Last In, First Out (LIFO) principle. The location of the most recent data on the stack is known as the TOP of the stack. The stack pointer always points to the top of the stack. Contents can be stored in the stack using the PUSH instruction and can restore the contents by using the instruction POP.

Mnemonic	Description
LXI SP, 16-bit	Load the stack pointer register with a 16-bit address.
PUSH Rp	Copies the contents of the specified register pair on the stack
POP Rp	Copies the contents of the top two memory locations of the stack into the specified register pair.

PUSH and POP Operation in 8085

PUSH R_p

This is a 1-byte instruction. This instruction copies the contents of the specified register pair on the stack as described below:

- The stack pointer is decremented and the contents of the higher-order register are copied to the location shown by the stack pointer register.
- The stack pointer is again decremented and the contents of the low-order register are copied to that location.

POP R_p

This is a 1-byte instruction. This instruction copies the contents of the top two memory locations of the stack into the specified register pair.

- First, the contents of the memory location indicated by the stack pointer register are copied into the low-order register and then the stack pointer register is incremented by 1.
- The contents of the next memory location are copied into the high-order register and the stack pointer register is again incremented by 1.

Comparison between CALL and PUSH Instructions : CALL instruction is used to call a subroutine. Subroutines are often used to perform tasks that need to be performed frequently. The programmer uses the PUSH instruction to save the contents of the register pair on the stack.

The differences between CALL and PUSH instructions are :

CALL	PUSH
After the execution of CALL instructions, the microprocessor automatically stores the 16 bit.	PUSH instruction is used to save the contents of specified registers pair on the stack.
After the execution of CALL instructions, the contents of stack pointer register are decremented by 2.	In this case after PUSH operation, contents of stack pointer are also decremented by 2.

Prob.26 *Explain the role of following in 8085-*
(a) Program Counter
(b) Stack Pointer [R.T.U. 2019]

Sol.(a) Program Counter : *Refer to Prob.20.*
(b) Stack Pointer : *Refer to Prob.20.*

Prob.27 ***Explain the function of various control and status signals available on 8085 microprocessor.*** [R.T.U. 2013]

Sol. Control and 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 and 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.28 ***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) Differences 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 micro-processor 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) Differences between 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) Differences between 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) Differences between 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)

PART-C

Prob.29 ***Design 8085 based system with following specifications:***
(i) System frequency 3MHz
(ii) Interface 16 kb EPROM using 8kb chip
(iii) Interface 32 kb RAM using 16 kb chip
[R.T.U. 2024, 2022]

Sol. To design an 8085 microprocessor based system with the given specifications, we need to follow the following steps:

1. Choose an 8085 microprocessor chip that can operate at a frequency of 3 MHz.
2. Select an 8 Kb chip for interfacing with the 16 Kb EPROM. The selected chip should have the same operating frequency as the microprocessor.
3. Connect the address lines A0-A13 of the 16 Kb EPROM to the corresponding address lines of the microprocessor. Connect the data lines D0-D7 of the EPROM to the data lines of the microprocessor.
4. Select a 16 Kb RAM chip for interfacing with the 32 Kb RAM. The selected chip should also operate at a frequency of 3 MHz.
5. Connect the address lines A0-A14 of the 32 Kb RAM to the corresponding address lines of the microprocessor. Connect the data lines D0-D7 of the RAM to the data lines of the microprocessor.
6. Provide control signals such as Chip Enable (CE), Output Enable (OE), and Write Enable (WE) to both the memory chips.
7. Connect other required peripheral devices such as input/output devices and other support chips to the microprocessor.
8. Test and debug the system to ensure it is functioning correctly.

The syntax of the system design can be represented in a block diagram as follows :

```
+----------------------+
8085 Microprocessor
+----------------------+
|A0-A15
v
+----------------------+
| 16 Kb EPROM |
+----------------------+
|D0-D7, CE, OE, WE|
|
v
+----------------------+
| 32 Kb RAM |
+----------------------+
|D0-D7, CE, OE, WE|
|
v
+----------------------+
| Peripheral devices |
```

Fig.

Prob.30 ***Explain the evolution of microprocessor and list the sequence of events that occurs. When 8085 μp reads data from memory.*** **[R.T.U. 2022]**

Sol. Evolution of Microprocessors : We can categorize the microprocessor according to the generations or according to the size of the microprocessor:

First Generation (4-bit Microprocessors) : The first generation microprocessors were introduced in the year 1971-1972 by Intel Corporation. It was named Intel 4004 since it was a 4-bit processor. It was a processor on a single chip. It could perform simple arithmetic and logical operations such as addition, subtraction, Boolean OR and Boolean AND.

It had a control unit capable of performing control functions like fetching an instruction from storage memory, decoding it, and then generating control pulses to execute it.

Second Generation (8-bit Microprocessor) : The second generation microprocessors were introduced in 1973 again by Intel. It was a first 8 bit microprocessor which could perform arithmetic and logic operations on 8-bit words. It was Intel 8008, and another improved version was Intel 8088.

Third Generation (16-bit Microprocessor) : The third generation microprocessors, introduced in 1978 were

represented by Intel's 8086, Zilog Z800 and 80286, which were 16-bit processors with a performance like minicomputers.

Fourth Generation (32-bit Microprocessors) : Several different companies introduced the 32-bit microprocessors, but the most popular one is the Intel 80386.

Fifth Generation (64-bit Microprocessors) : From 1995 to now we are in the fifth generation. After 80856, Intel came out with a new processor namely Pentium processor followed by Pentium Pro CPU, which allows multiple CPUs in a single system to achieve multiprocessing. Other improved 64-bit processors are Celeron, Dual, Quad, Octa Core processors, i3, i5, i7 etc.

Table: Important Intel Microprocessors

Microprocessor	Year of Invention	Word Length	Memory addressing Capacity	Pins	Clock	Remarks
4004	1971	4-bit	1 KB	16	750 KHz	First Microprocessor
8085	1976	8-bit	64 KB	40	3-6 MHz	Popular 8-bit Microprocessor
8086	1978	16-bit	1 MB	40	5-8 MHz	Widely used in PC/XT
80286	1982	16-bit	16MB real. 4 GB virtual	68	6-12.5 MHz	Widely used in PC/AT
80386	1985	32-bit	4GB real, 64TB virtual	132 14 × 14 PGA	20-33 MHz	Contains MMU on chip
80486	1989	32-bit	4GB real, 64TB virtual	168 17 × 17 PGA	25-100 MHz	Contains MMU, cache and FPU, 1.2 million transistors
Pentium	1993	32-bit	4GB real, 32-bit address, 64-bit data bus	237 PGA	60-200 MHz	Contains 2 ALUs, 2 Caches, FPU, 3.3 Million transistors, 3.3 V. 7.5 million transistors
Pentium Pro	1995	32-bit	64GB real, 36- bit address bus	387 PGA	150-200 MHz	It is a data flow processor. It contains second level cache also,3.3 V
Pentium II	1997	32-bit	–	–	233-400 MHz	All features Pentium pro plus MMX technology, 3.3 V, 7.5 million transistors
Pentium III	1999	32-bit	64 GB	370 PGA	600 MHz - 1.3 GHz	Improved version of Pentium II; 70 new SIMD instructions
Pentium IV	2000	32-bit	64 GB	423 PGA	600 MHz - 1.3 GHz	Improved version of Pentium III
Itanium	2001	64-bit	64 address lines	423 PGA	733 MHz-1.3 GHz	64-bit EPIC Processor
i3	2007	64-bit	–	–	2.2 – 3.3 GHz	–
i5	2009	64-bit	–	–	2.4 – 3.6 GHz	–
i7	2010	64-bit	–	–	2.93 – 3.33 GHz	–

Where,

- PGA - Pin Grid Array
- MMX - MultiMedia eXtensions
- EPIC - Explicitly Parallel Instruction Computing
- SIMD - Single Instruction Multiple Data
- ALU - Arithmetic and Logic Unit
- MMU - Memory Management Unit
- FPU - Floating Point Unit

8085 provides two signals - $IO/\overline{M}$ and $\overline{RD}$ to indicate that it is a memory read operation. The $IO/\overline{M}$ and $\overline{RD}$ can be combined to generate the $\overline{MEMR}$ (Memory Read) control signal that can be used to enable the output buffer by connecting to the memory signal $\overline{RD}$.

Prob.31 *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 :

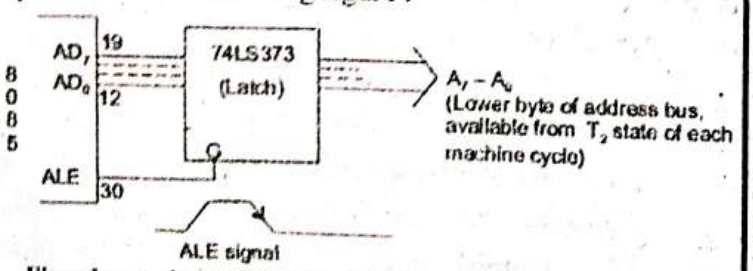
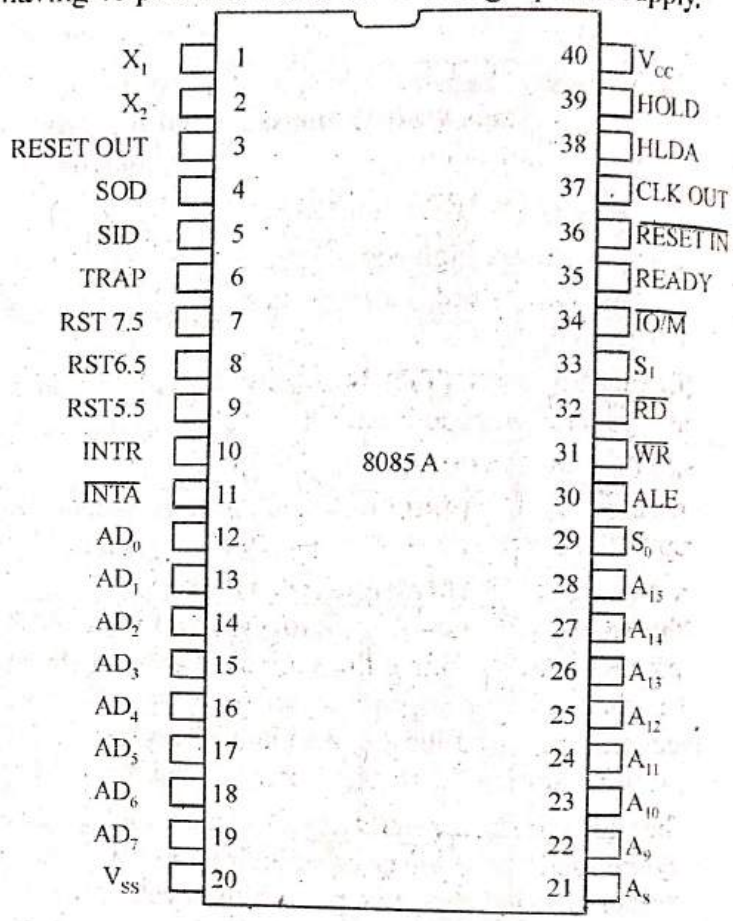


Fig. : Lower byte of address latching achieved by the H to L transition of ALE signal, which occurs at the end of T_1 of each machine cycle

Prob.32 *Draw pin diagram of 8085 and explain each pin.* [R.T.U. 2017]

Sol. PIN Diagram of 8085 Microprocessor : The 8085 microprocesor 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.

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

8. Pin No. 29&33 (S_1 and S_0) : These are status signals. These can identify various operations, but they are rarely used in small systems.

9. Pin No. 30 (ALE) : *Refer to Prob.27.*

10. 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.

11. 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.

12. Pin No. 34 (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.

13. Pin No. 35 (READY) : *Refer to Prob.4.*

14. 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.

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

16. 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.

17. Pin No. 39 (HOLD) : *Refer to Prob. 4*

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

Prob.33 ***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.20.*

(b) Stack Pointer : *Refer to Prob.20.*

(c) General Purpose Programmable Registers : *Refer to Prob.13.*

(d) Instruction Register : *Refer to Prob.20.*

(e) HLDA Pin : *Refer to Prob.32.*

HOLD Pin : *Refer to Prob.4.*

(f) Control and Status Pins (Bus)

1. $IO/\overline{M} = \begin{cases} 1 = \text{data transfer from I/O.} \\ 0 = \text{data transfer from memory.} \end{cases}$
2. $\left.\begin{matrix} S_1 \\ S_0 \end{matrix}\right\} \Rightarrow$ 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,

$$\left.\begin{array}{l}\overline{WR}=0\\ \overline{RD}=0\end{array}\right\}\text{Wrong}$$

$$\left.\begin{array}{l}\overline{WR}=1\\ \overline{RD}=1\end{array}\right\}\text{Right}$$

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.

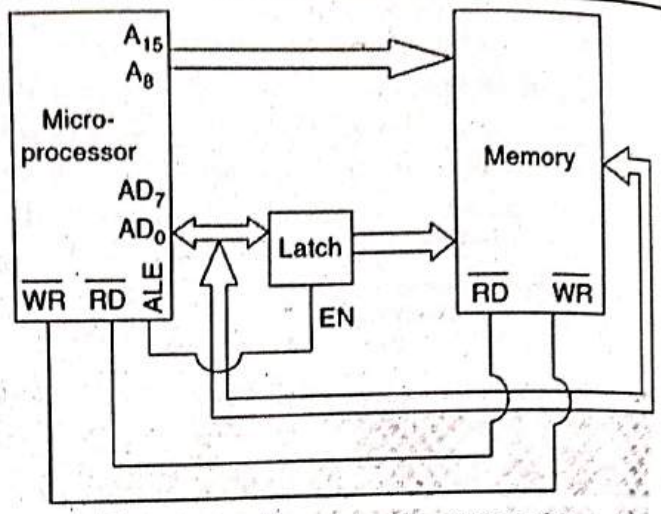


Fig. : Block diagram showing ALE Action

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

(g) Flags : *Refer to Prob.23.*

(h) PSW : *Refer to Prob.5.*

□□□

ASSEMBLY LANGUAGE PROGRAMMING 2

PREVIOUS YEARS QUESTIONS

PART-A

Prob.1 ***State the difference between direct and indirect addressing.*** **[R.T.U. 2024]**

Sol. Difference between Direct and Indirect Addressing :

Direct Addressing	Indirect Addressing
The operand address is specified directly in the instruction.	The instruction contains a memory location that stores the actual address of the operand.
Faster, as it accesses the operand directly.	Slower, as it requires an extra memory access to fetch the operand address.
Requires more instruction bits for addressing.	Requires an additional memory location to store the operand address.
Less flexible, as it has a limited address range.	More flexible, as it can access a larger memory space.
Example : MOV A, 2000H (Fetches data from address 2000H)	**Example :** MOV A, @R0 (R0 holds the address of the operand)

Prob.2 ***Write a program to add 7BH and 6AH using ADI instruction.*** **[R.T.U. 2024, 2022]**

Sol. Program to Add 7BH and 6AH using ADI Instruction: ORG 0000; set program counter to 0000H
MOV A, 7BH; move the value 7B H into the accumulator
ADI 6AH; add the value 6A H to the accumulator
; the ADI instruction will update the carry flag if needed
HLT; halt the program
END; end of program

Prob.3 ***State the difference between JUMP and CALL instruction.*** **[R.T.U. 2024]**

Sol. Difference between JUMP and CALL Instructions :

JUMP Instruction	CALL Instruction
Transfers control to a new address within the program.	Transfers control to a subroutine (function) and returns after execution.
No return; execution continues from the new address.	Uses a return (RET) instruction to go back to the calling function.
Used for looping, decision-making, and branching.	Used for function calls and modular programming.
Does not store the return address.	Stores the return address on the stack before jumping.
Example : JMP 2000H (Moves execution to address 2000H).	**Example :** CALL 3000H (Moves execution to subroutine at 3000H and returns after completion).

Prob.4 ***Name the different machine cycles.*** **[R.T.U. 2023]**

Sol. Types of Machine Cycles : 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

Prob.5 ***Which addressing mode is used in instruction STAX D?*** **[R.T.U. 2023]**

Sol. Addressing Mode used in Instruction STAX D: The STAX D instruction in the 8085 microprocessor is used to store the accumulator (A) contents at the memory location addressed by the contents of the DE register pair. This instruction utilizes the direct addressing mode, where the operand is specified directly as a part of the instruction. In this case, the DE register pair serves as the operand, and the accumulator contents are stored at the memory address specified by the DE register pair.

Prob.6 ***What is the use of RET instruction?*** **[R.T.U. 2023]**

Sol. Use of RET Instruction:

- The RET (Return) instruction in the 8085 microprocessor is used to return control from a subroutine to the main program.
- When executed, the RET instruction retrieves the return address from the top of the stack and transfers program control to that address.
- It is commonly used at the end of a subroutine to return to the calling program or subroutine.
- Additionally, the RET instruction can be conditional, allowing for conditional returns based on the state of certain flags. It is 1-byte instruction.

Prob.7 ***What are the steps involved in instruction cycle.*** **[R.T.U. 2023]**

Sol. Steps Involved in Instruction Cycle: The instruction cycle in a microprocessor consists of several steps:

- **Fetch:** The microprocessor fetches the instruction from memory using the program counter (PC).
- **Decode:** The fetched instruction is decoded to determine the operation to be performed and the operands involved.
- **Execute:** The decoded instruction is executed by performing the specified operation, which may involve fetching additional data from memory or performing arithmetic/logic operations.
- **Store:** If necessary, the result of the executed instruction is stored in the appropriate location, such as a register or memory.

Prob.8 ***What is the use of temporary registers W and Z in 8085?*** **[R.T.U. 2023]**

Sol. Use of Temporary Registers W and Z in 8085: The 8085 microprocessor has a set of temporary registers labeled W and Z. These registers are used internally by the processor during certain arithmetic and logic operations. Register W is often used as a scratchpad register during multiplication and division operations. Whereas register Z serves as a temporary storage register during addition and subtraction operations, particularly for holding the carry flag.

Prob.9 ***Define instruction and 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.10 ***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.11 ***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.12 ***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.13 ***Compare the function of the RST and RET.*** **[R.T.U. 2013]**

Sol. Comparison between RET and RST 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.

PART-B

Prob.14 ***Write a program to do addition of two 8 bit numbers whose 16 bit result is stored at memory location 5080H.*** **[R.T.U. 2024]**

Sol. Addition of Two 8-bit Numbers and Store 16-bit Result at Memory Location 5080H

Assembly Code:

```
LXI H, 5000H   ; Load HL with memory address
                 5000H (first number)
MOV A, M       ; Load first 8-bit number into
                 Accumulator
INX H          ; Increment HL to point to
                 5001H (second number)
MOV B, M       ; Load second 8-bit number
                 into Register B
ADD B          ; Add register B to
                 Accumulator
MOV L, A       ; Store LSB of result in L
MOV A, 00H     ; Clear Accumulator to check
                 for carry
ADC A          ; Add Carry to Accumulator
MOV H, A       ; Store MSB of result in H
LXI D, 5080H   ; Load DE with 5080H (result
                 memory location)
MOV M, L       ; Store LSB at 5080H
INX H          ; Move to next location 5081H
MOV M, H       ; Store MSB at 5081H
HLT            ; Halt the program
```

Explanation of the Program:

1. **Load Numbers from Memory:**
   - The first number is stored at 5000H, loaded into Accumulator (A).
   - The second number is stored at 5001H, loaded into Register B.
2. **Addition Operation:**
   - ADD B adds B to A, and the sum is stored in A.
   - ADC A is used to check if a carry is generated, adding it to the result.
3. **Storing the 16-bit Result:**
   - Lower Byte (LSB) is stored at 5080H.
   - Higher Byte (MSB - Carry, if any) is stored at 5081H.
4. **Program Execution Ends with HLT.**

Prob.15 ***Compare the function of following instruction pairs:***

(i) JMP & CALL

(ii) STAX & LDAX

(iii) LHLD & LXI

(iv) ANA & ANI **[R.T.U. 2024, 2022]**

Sol.(i) JMP vs CALL

JMP	CALL
Unconditionally jumps to the specified address.	Calls a subroutine at the specified address.
Control is transferred without saving the return address.	Saves the return address on the stack before jumping.
No way to return to the previous point.	Must be followed by a RET instruction to return.
Example: JMP 2050H	**Example:** CALL 2050H

(ii) STAX vs LDAX

STAX	LDAX
Stores the content of the accumulator into a memory location pointed by a register pair (B or D).	Loads the content of a memory location pointed by a register pair (B or D) into the accumulator.
Used to write data to memory.	Used to read data from memory.
Syntax: STAX B or STAX D	**Syntax:** LDAX B or LDAX D

(iii) LHLD vs LXI

LHLD	LXI
Loads a 16-bit content from memory into HL pair.	Loads immediate 16-bit data into a register pair.
Uses two consecutive memory locations.	Uses data provided directly in the instruction.
Example: LHLD 2500H (loads H←2501, L←2500)	**Example:** LXI H, 2500H (H←25, L←00)

(iv) ANA vs ANI

ANA	ANI
Performs logical AND between accumulator and the contents of a register/memory.	Performs logical AND between accumulator and immediate 8-bit data.
Operand is a register or memory.	Operand is a constant (immediate) value.
Example: ANA B	**Example:** ANI 0FH

Prob.16 ***Explain various addressing modes in 8085 with suitable example.*** **[R.T.U. 2023]**

OR

What is the use of addressing modes? Explain different types of addressing modes with example. **[R.T.U. 2022]**

OR

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

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 delay subrouting using 8 bit register. What is the maximum possible delay obtainable.*** **[R.T.U. 2023]**

Sol. To create a delay subroutine using an 8-bit register in assembly language for a microcontroller like the Intel 8085, you can use a simple loop that decrements a counter until it reaches zero. Here's a sample delay subroutine written in assembly language for the Intel 8085 microprocessor:

```
DELAY:
        MVI B, OxFF   ; Load B register with 255
                      (maximum delay)
DELAY_LOOP:
        DCX  B        ; Decrement B register
JNZ DELAY_LOOP        ; Continue looping until B becomes
                      zero
        RET           ; Return from subroutine
```

In this subroutine:

- **MVI B, OxFF** initializes the B register with the maximum value of 255, which provides the maximum delay achievable with an 8-bit register.
- **DELAY_LOOP** is a label marking the start of the loop.
- **DCX B** decrements the value in the B register.
- **JNZ DELAY_LOOP** jumps back to the **DELAY_LOOP** label if the zero flag is not set (i.e., if B is not yet zero).
- **RET** returns control from the subroutine back to the calling code.

The maximum possible delay obtainable with this subroutine depends on the clock frequency of the microcontroller and the number of instructions executed within the loop. Since the loop decrements the B register 255 times, and each decrement operation takes a certain number of clock cycles (typically a few cycles per instruction), the total delay is proportional to the clock frequency and the number of instructions executed.

For example, if each instruction takes 1 microsecond to execute, and assuming there are no other delays or overhead, the maximum delay achievable would be approximately 255 microseconds. However, actual delay may vary based on factors such as clock frequency, instruction execution time, and any additional processing overhead.

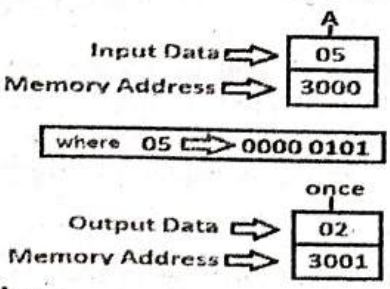
Prob.18 ***Write a program to add the contents of memory locations of 2000 H and 2001 H and place the results in memory location 2002 H.*** **[R.T.U. 2022]**

Sol. Program :

```
LXI H, 2000H  ; HL points to location 2000H
MOV A, M      ; Move the contents of 2000H into
                accumulator
INX H         ; HL points to location 2001H
ADD M         ; Add the contents of 2001H with the
                contents of accumulator
INX H         ; HL points to location 2002H
MOV M, A      ; Store the lower byte of result at 2002H
MVI A, 00H    ; Initialize higher byte result with 00H
ADC A         ; Add carry in the higher byte result
INX H         ; HL points to location 2003H
MOV M, A      ; Store the higher byte of result at 2003H
HLT           ; Terminate program execution
```

Prob.19 ***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.20 ***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.

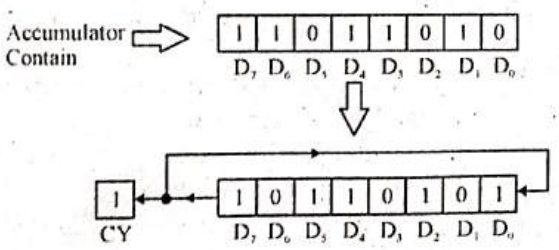
(i) RLC (Rotate Accumulator Left)
(ii) RAL (Rotate Accumulator Left Through Carry)
(iii) RRC (Rotate Accumulator Right)
(iv) RAR (Rotate Accumulator Right Through Carry)

(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).

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

(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.

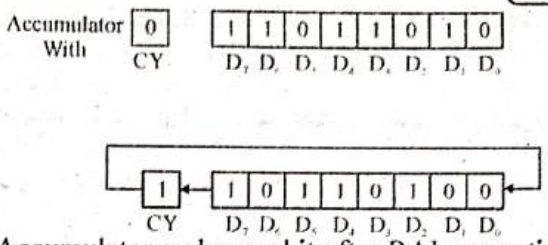
The seventh bit of the accumulator shift to carry bit and carry bit is shift to zero bit of the accumulator.

Syntex :

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

(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.

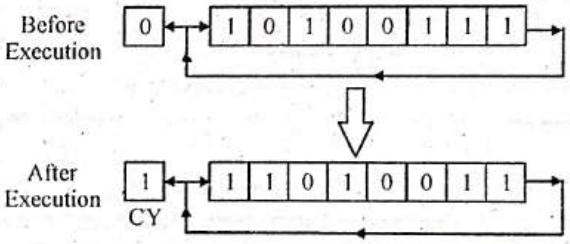
(iii) RRC (Rotate Accumulator Right) : Rotate or shift the contents of accumulator right by one bit without carry. Zero bit of accumulator shift to seventh bit and carry bit.

Syntex :

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

(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.

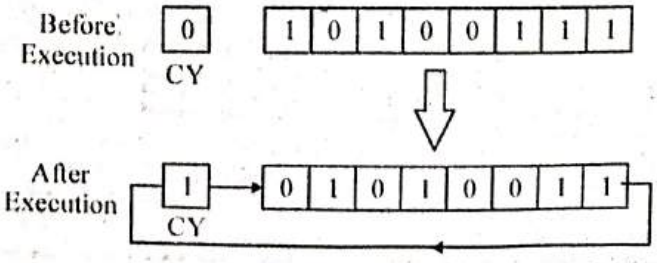
(iv) RAR (Rotate Accumulator Right Through Carry) : Rotae of shift the contents of accumulator right by one bit with carry. Zero bit of Accumulator is shift to carry bit and carry bit is shift to seventh bit.

Syntex :

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

It has same features like RRC instructions.

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



Prob.21 ***Define instruction cycle, machine cycle and T-state.*** [R.T.U. 2016]

Sol. Instruction Cycle: An instruction cycle can be define as total time required by the microprocessor to completely fetch and execute an instruction. An instruction cycle is comprised of one or more number of machine cycles.

Machine Cycle: Machine cycle can be define as time required by the microprocessor to complete an operation of accessing memory or input / output devices. A machine cycle is comprise of some member of T-states.

T-state: The machine cycle and instruction cycle takes multiple clock periods. A portion of an operation carried out in one system clock period is called as T-state.

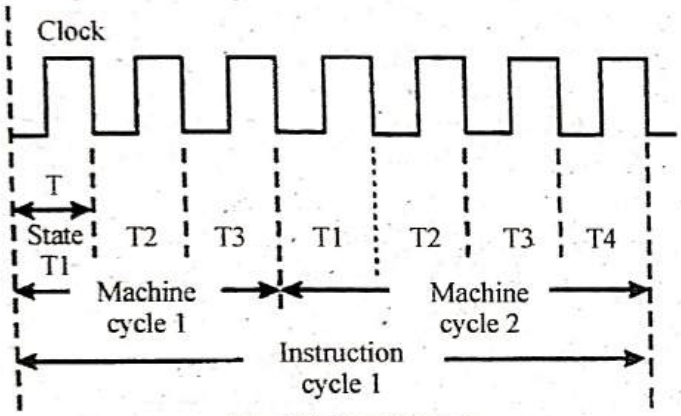


Fig.: Timing diagram

Prob.22 ***Write the assembly Language program for Addition of the two Hexadecimal numbers 3A968A67 and 9B476C8B.***

Sol. 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

PART-C

Prob.23 ***Write a program to transfer a block of memory data starting from one memory location to another memory location in reverse order.*** [R.T.U. 2023]

Sol.

```
ORG 0000H   ; Starting address of the program
MOV C, L    ; Copy the lower byte of the source
            address to register C
MOV B, H    ; Copy the higher byte of the source
            address to register B
INX H       ; Increment the source address to point
            to the last byte of the block
MOV E, L    ; Copy the lower byte of the destination
            address to register E
MOV D, H    ; Copy the higher byte of the destination
            address to register D
INX H       ; Increment the destination address to
            point to the last byte of the block
MOV A, M    ; Load the first byte of data from the
            source address
DCX H       ; Decrement the source address
MOV M, A    ; Store the byte of data to the destination
            address
DCX D       ; Decrement the destination address
LOOP:       ; Loop to transfer the rest of the data
MOV A, M    ; Load the next byte of data from the
            source address
DCX H       ; Decrement the source address
MOV M, A    ; Store the byte of data to the destination
            address
DCX D       ; Decrement the destination address
CMP B       ; Compare the source address with the
            starting address
```

JNZ LOOP ; Repeat the loop until the source address reaches the starting address

HLT ; Halt the program

END ; End of the program

Explanation:

- The program starts at address 0000H.
- It initializes the source and destination addresses (assumed to be stored in HL and DE register pairs respectively).
- It loads the first byte of data from the source address and stores it to the destination address.
- It continues to load bytes from the source address in reverse order, storing them to the destination address, until the entire block of data is transferred.
- The program halts after the transfer is complete.

Prob.24 ***Explain the instruction format of 8085. Draw the timing diagram of instruction MOV A, M and explain it.*** **[R.T.U. 2017]**

Sol. Instruction Formats : The Instruction Format of 8085 set consists of one, two and three byte instructions. The one-byte instruction has an opcode alone and the two-byte instruction has an opcode followed by an eight-bit address or data. The three-byte instruction has an opcode followed by a 16-bit address or data. While storing the three-byte instruction in memory, the sequence of storage is, opcode first followed by a low byte of address or data and then a high byte of address or data.

1. One Byte Instruction : MOV A, B whose opcode is 78H which is one byte. This Instruction and Data Format of 8085 copies the contents of B register in A register.

Opcode

Fig. 1 : One Byte Instruction

2. Two Byte Instruction : The opcode for this instruction is 06H and is always followed by a byte data (02H in this case). This instruction is a two byte instruction which copies immediate data into B register.

Opcode	Operand/address

Fig. 2 : Two Byte Instruction

3. Three Byte Instruction : The opcode for this instruction is C3H and is always followed by 16 bit address (6200H in this case). This instruction is a three byte instruction which loads 16 bit address into program counter.

Opcode	Lower bits of operand	Higher bits of operand

Fig. 3 : Three Byte Instruction

4. 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.

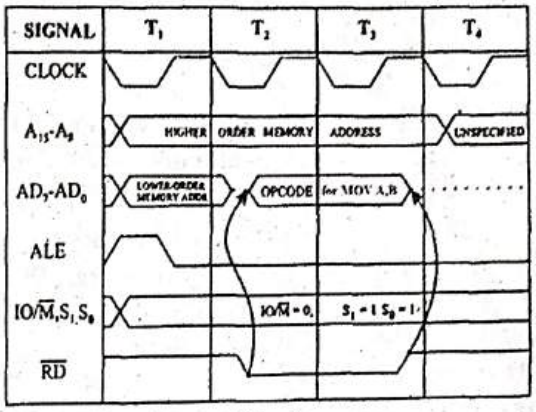


Fig. 4

Prob.25 ***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.

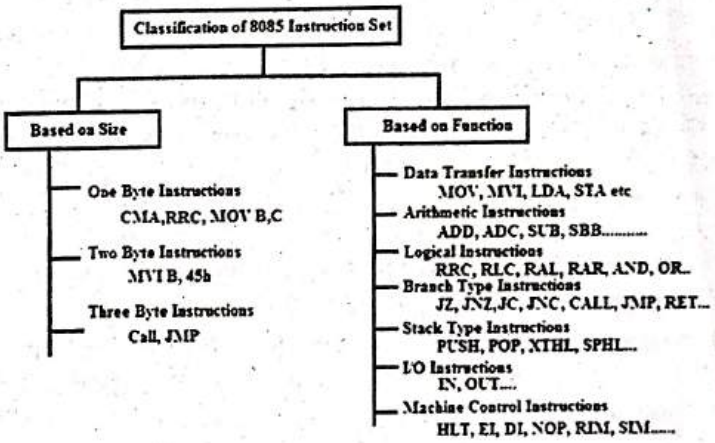


Fig.

Classification of Instruction Set of 8085

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

1. **One byte Instructions:** These instructions are of one byte in size and hence occupy one memory location in RAM. Examples are CMA, RLC, RRC, RAL, RAR, STC, CMC etc. These instructions do not require any operand to be specified with the instructions, instead the operand is implied in the instructions.
2. **Two Byte Instruction:** These instructions of two byte(16-bits) in size and hence will occupy two

memory locations in RAM. Examples of such instructions are MVI C,0A;

3. **Three Byte Instructions:** These are of three byte in size and hence occupy three locations in memory(RAM). Examples of such instructions are CALL, JMP etc.

Instruction Set Based on Function

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

1. Data Transfer Group : The data transfer instructions move data between registers or between memory and registers.

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

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

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

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

ADD	Add to Accumulator
ADI	Add Immediate Data to Accumulator
ADC	Add to Accumulator Using Carry Flag
ACI	Add immediate data to Accumulator Using Carry
SUB	Subtract from Accumulator
SUI	Subtract Immediate Data from Accumulator
SBB	Subtract from Accumulator Using Borrow (Carry) Flag
SBI	Subtract Immediate from Accumulator Using Borrow (Carry) Flag
INR	Increment Specified Byte by One
DCR	Decrement Specified Byte by One
INX	Increment Register Pair by One
DCX	Decrement Register Pair by One
DAD	Double Register Add; Add Content of Register Pair to H & L Register Pair

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.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

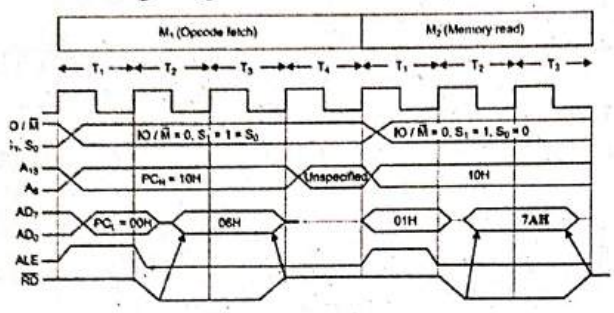


Fig. : Timing Diagram

Prob.27*(a) 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.

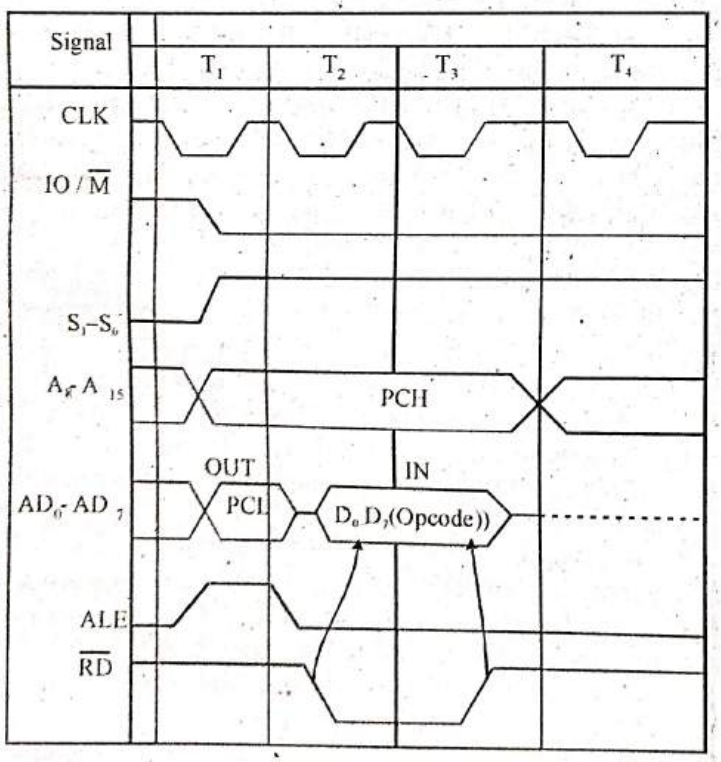


Fig. 1: Timing Diagram for Opcode Fetch Operation

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 during subsequent clock cycles, it is used in time multiplexed mode. Therefore it has to be made available to carry data during T_2 and T_3. To accomplish this the microprocessor sends an address latch enable signal ALE to latch the 8LSBs of memory address either in memory or an external latch.

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

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

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.

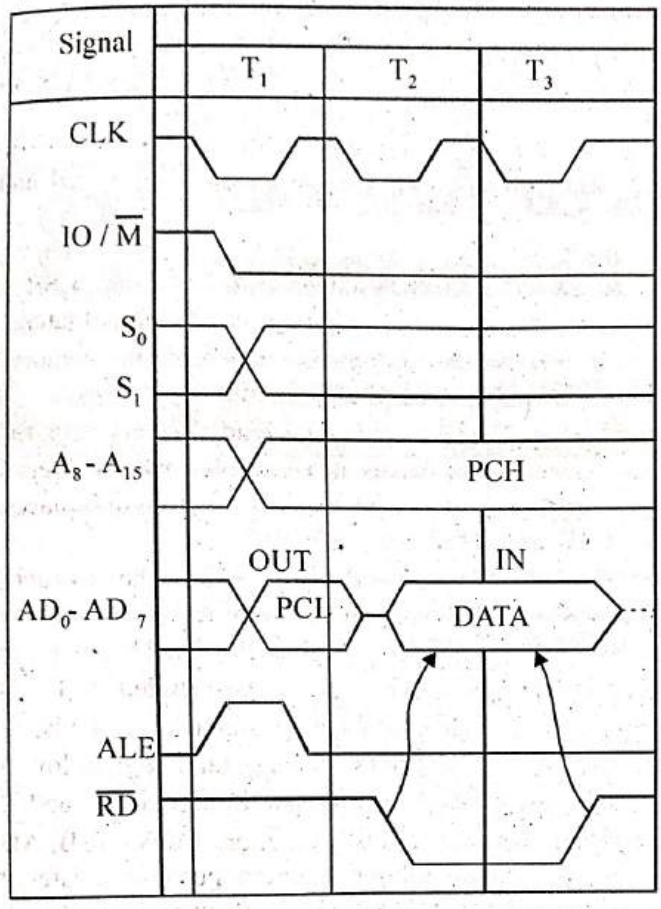


Fig. 2: Timing diagram for Memory Read Operation.

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)(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

❑❑❑

ADVANCED ASSEMBLY LANGUAGE PROGRAMMING 3

PREVIOUS YEARS QUESTIONS

PART-A

Prob.1 ***Describe the use of RIM and SIM instruction.*** **[R.T.U. 2024]**

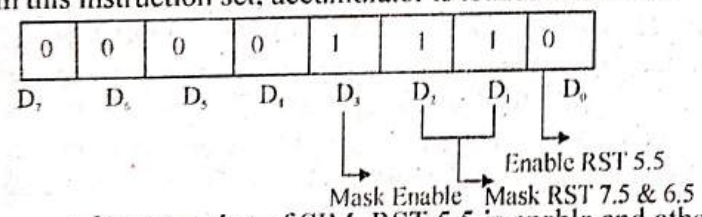
Sol. Use of RIM & SIM Instructions :

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.



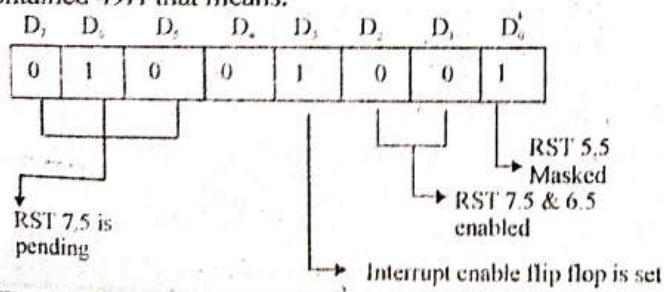
After execution of SIM, RST 5.5 is enable and other interrupts are disabled/ masked.

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.2 ***What is the difference between delay and counter?*** **[R.T.U. 2024]**

Sol. Difference Between Delay and Counter :

Delay	Counter
A mechanism to introduce a time delay in execution.	A register used to count events or pulses.
Used for generating time delays in programs.	Used for counting occurrences of an event.
Uses loops or timers to create a pause.	Increments or decrements a value based on input signals.
Used in timing operations, pulse width modulation, and synchronization.	Used in event counting, frequency measurement, and digital clocks.
Example : A loop in assembly language to delay execution.	**Example :** A program that counts button presses in an embedded system.

Prob.3 ***What happen with stack pointer after executing two PUSH instructions?*** **[R.T.U. 2023]**

Sol. Stack Pointer After Executing Two PUSH Instructions:

- After executing two PUSH instructions in the 8085 microprocessor, the stack pointer (SP) will decrease by two times the size of the pushed data (typically 16 bits).
- The stack pointer points to the memory location and only used for registers pairs BC and DE.

Prob.4 ***What are priority interrupts?*** **[R.T.U. 2022]**

Sol. Priority Interrupts : The 8085 microprocessor has five interrupt inputs. They are TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR. These interrupts have a fixed priority of interrupt service.

If two or more interrupts go high at the similar time, the 8085 will service them on priority basis. The TRAP has the highest priority followed by RST 7.5, RST 6.5, RST 5.5 and INTR.

Prob.5 ***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.6 ***Differentiate between Synchronous and Asynchronous data transfer.*** **[R.T.U. 2019]**

Sol. Difference between Synchronous and Asynchronous Data Transfer :

Synchronous Data Transfer	Asynchronous Data Transfer
Data is transferred in sync with a clock signal.	Data is transferred without using a clock signal.
Sender and receiver use the same clock.	Sender and receiver do not share a common clock.
Faster and more efficient for continuous transmission.	Slower due to start/stop bits and possible gaps.
Suitable for high-speed communication.	Suitable for intermittent or low-speed communication.
No need for start/stop bits, as clock keeps both ends synchronized.	Start and stop bits are required for synchronization.
Examples: SPI, I2C, Ethernet.	**Examples:** RS-232 Serial Port, UART communication.
More complex hardware due to clock synchronization.	Simpler hardware, easier to implement.

Prob.7 ***Differentiate vectored and non-vectored interrupts of 8085.*** **[R.T.U. 2016]**

Sol. Difference between vectored and non-vectored interrupts :

Vectored Interrupts	Non-Vectored Interrupts
The processor automatically branches to a predefined memory address for the Interrupt Service Routine (ISR).	The ISR address is not fixed and must be supplied by the interrupting device.
No need for external hardware to provide the ISR address.	Requires external hardware or device to provide the ISR address.
Faster response due to a known interrupt vector location.	Slower response as the ISR address must be determined at runtime.
Efficient for systems with fixed interrupt priorities.	Flexible and useful for systems with multiple interrupt sources.
Example : RST7.5, RST6.5, RST5.5.	**Example :** INTR interrupt.

PART-B

Prob.8 ***What is the use of stack? Explain the PUSH and POP operations using suitable example.*** **[R.T.U. 2024, 13]**

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 location 3097 in the register C and 3098 in the register B and the stack pointer register again contains the address 3099. (incremented by two).

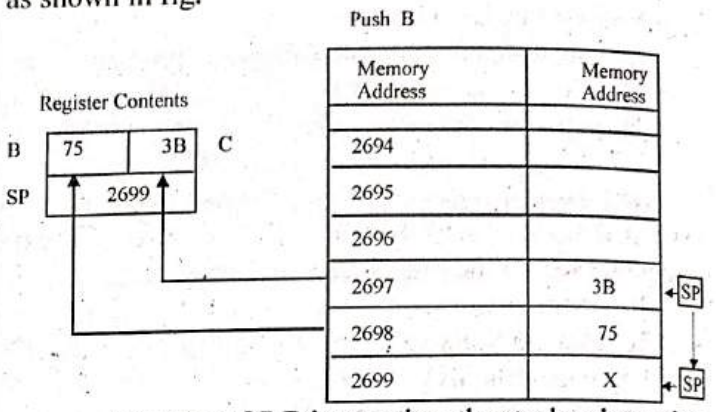
Example :

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

Now the instruction PUSH B will copy the contents of register B in the memory location 2698 and the contents of register C in the memory location 2697. After this PUSH B instruction, the stack pointer register is decremented by 2 and now, it contains the memory address 2697 as shown in fig.

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

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



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

Prob.9 ***Differentiate between Macro and Subroutine.*** **[R.T.U. 2023]**

Sol. Difference between Macro and Subroutine :

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.

Prob.10 ***Show which interrupt will be masked if the following instructions are executed :***

(i) MVI A, 10 H

(ii) SIM **[R.T.U. 2022]**

Sol. EI (Enable Interrupt):

- The interrupt enables flip-flop is set and all the interrupts are enabled.
- No flags are affected.
- After a system reset or the acknowledgment of an interrupt, the interrupt enables flip flop is reset, thus disabling the interrupts.
- This instruction is necessary to reenable the interrupts (except TRAP).

SIM : It is used to set the interrupt mask. It is a 1-byte instruction used for the following three functions:

- To set the mask
- To reset the flip flop
- Implement the I/O

The format of interpreting the contents of the accumulator is shown in the figure below:

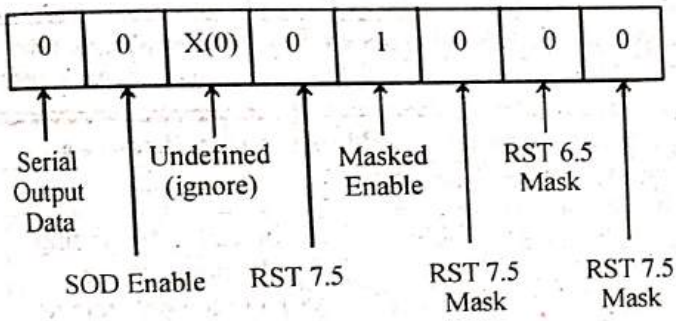


Fig.

The sequence of instructions to enable all the interrupts of 8085 instructions is written as:

EI : Enable Interrupts

MVI A,10 H: LOAD accumulation with a suitable bit pattern

SIM: Enable RST 7.5 and RST 5.5

Prob.11 ***Write a 8085 program that enables RST 7.5 and RST 5.5 interrupts.*** **[R.T.U. 2022]**

Sol. Firstly, we determine the contents of the accumulator. As we have to enable RST5.5 and RST7.5, so bit 0 and bit 2 are 0. RST6.5 is disabled so bit 1 is 1. We want to allow masking so bit 3 is 1. RST7.5 is not reset so bit 4 is 0. Bit 5 is not used, so it can be taken as either 0 or 1. There is no serial communication, meaning that bit 6 and bit 7 is 0.

The contents of accumulator (assuming bit 5 as 0) are as shown below:

SDO	SDE	XXX	R7.5	MSE	M7.5	M6.5	M5.5
0	0	0	0	1	0	1	0

Therefore, accumulator contents is 0AH.

Program :

EI; Enable interrupts including INTR

MVI A, OA; Prepare the mask to enable RST 7.5 and 5.5, disable 6.5

SIM; Apply the settings RST masks

An instruction to check pending interrupts is RIM.

RIM: Read Interrupt Mask

It is also a 1-Byte multi-purpose instruction with the Mnemonic 20H. It is used for the following purposes:

- To check if RST7.5, RST6.5, and RST5.5 are masked.
- To check if the interrupts are enabled.
- To check if RST7.5, RST6.5, or RST5.5 interrupts are pending.
- To perform serial input of data.

The format for contents of Accumulator for RIM instruction are :

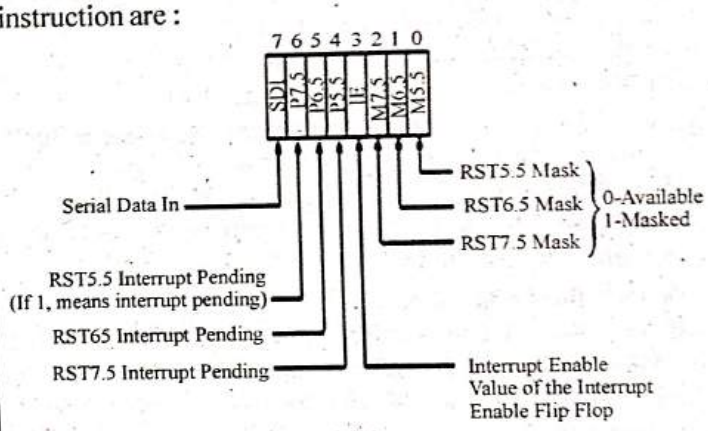


Fig.

Prob.12 ***Explain DAA instruction with an example.*** **[R.T.U. 2022]**

Sol. 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.

Example : Let us consider we want to add two decimal numbers 38 and 45. They will be represented in BCD as 0011 1000 and 0100 0101. The addition results in 0111 1101. But the answer will be incorrect if we want to interpret this result as a BCD number. The result will be not only incorrect but also illegal as 1101, which we obtained as the last nibble in the answer is not a valid BCD number. Here, in such situations, we can use DAA to have the BCD sum as outcome. All that is required to be done is to add the BCD numbers and store the result in A, and then execute the DAA instruction.

Here is the detailing of the calculations :

```
 38 ---> 0011 1000
+45 ---> 0100 0101
--------------------
 83      0111 1101
         -------- --------
            7        D
```

The working of DAA instruction depends on the contents of the AL register, Cy, and AC flags. In effect, it adds 00H, 06H, 60H, or 66H to Accumulator so as to get the correct BCD answer in the Accumulator. So here is the illustration of the remedial actions against the previous example.

```
 38 ---> 0011 1000
+45 ---> 0100 0101
--------------------
 83      0111 1101
         -------- --------
            7        D

                0111 1101
                + 0110 (06H)
                ----------------
                1000 0011 --->83 (Decimal sum)
```

As the summary of all related rules are listed below:

- If the LS hex digit in A is ≤ 9 and AC flag is 0, the LS hex digit value will not be altered.
- If the LS hex digit is > 9, or if AC flag is set to 1, it adds 6 to the LS hex digit of A. If carry results, then it increments the MS hex digit if this addition resulted in a carry to the MS digit position. In this process, the Cy flag will be set to 1 if the MS hex digit was incremented from F to 0.
- If the MS hex digit is ≤ 9 and Cy flag is 0, the MS hex digit will not be altered, and Cy flag is reset to 0.
- If the MS hex digit is > 9, or if Cy flag is set to 1, it adds 6 to the MS hex digit of A and sets Cy flag to 1.

Note that for decimal subtraction DAA instruction cannot be used. Due to unavailability of decimal subtraction in Intel 8085 instruction set, a series of instructions are to be executed to perform decimal subtraction.

Address	Hex Codes	Mnemonic	Comment
2000	3E	MVI A, 38H	A ← 38H
2001	38		38H as operand
2002	06	MVI B, 45H	B ← 45H
2003	45		45H as operand
2004	80	ADD B	A ← A+B; A ← 38H + 45H; A ← 7DH
2005	27	DAA	A ← 83H (Decimal Sum), S = 1, Z = 0, Ac = 1, P = 0, Cy = 0 06H got added with the Accumulator content. As Cy=0, so interpreted result is 83 in decimal

Prob.13 ***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 etc are written and are stored in different memory locations. Whenever these operations are required, these subroutines are called from the specified memory locations and are used in the main program.

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

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

The CALL instruction is used in the main program whereas the RET instruction is used at the subroutine. The subroutine may be used several times in a program. There may be several subroutines (for different operations) used in a program. Subroutines are used to save memory location.

Some important subroutine may be written in the ROM by the manufacture.

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

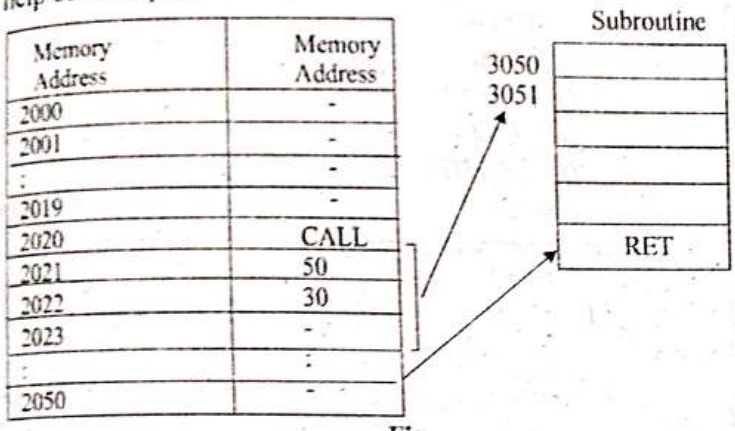


Fig.

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 program sequence to subroutine address • Saves the content of program counter 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 instuction. • 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.14 ***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.15 ***(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) 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) 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.16 ***(a) Write a program to transfer a block of 10 data elements from memory location 5000H to 6000H.***

(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) <u>CMA</u> : 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) <u>CMP</u> : 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 is 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) <u>LDAX</u> :

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 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 L 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.17 ***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 at 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.18 ***Explain the RIM and SIM instruction briefly.*** **[R.T.U. 2013]**

OR

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

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 :

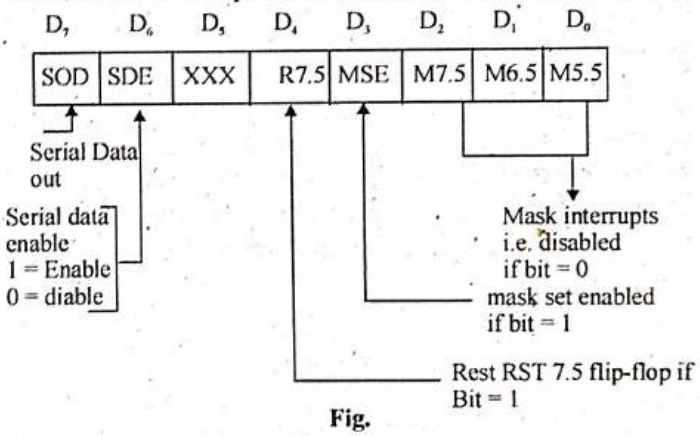


Fig.

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

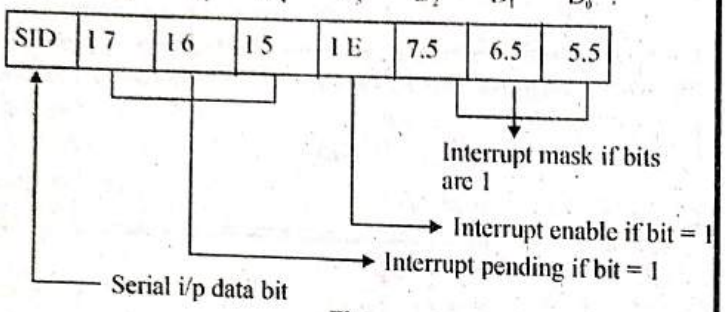


Fig.

Use of RIM & SIM for 8085 Interrupts : *Refer to Prob. 1.*

Prob.19 ***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).

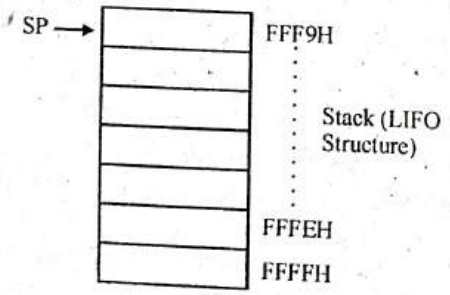


Fig.

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.20 ***Write short note on "Macros and Subroutines."***

Sol. Macros : A sequence of instructions to which a name is called *macro*. The name of macro used, in assembly language programming. The macro facility is available with many assemblers. Each macro is clearly defined and unique name is assigned to it.

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 macro name is assigned to it, the macro name can be frequently used in program. Its name in program is easier to read and understand. Marco and subroutine are similar.

A subroutine requires CALL and RETURN instruction whereas macro do not. Macro execute faster than subroutine.

Macros are used for short sequence of instruction whereas subroutine for larger ones, preferably for 10 instructions and more.

Subroutine : *Refer to Prob.13.*

Difference between Macro and Subroutine : *Refer to Prob.9.*

PART-C

Prob.21 ***Write short note on Memory interfacing*** **[R.T.U. 2024]**

Sol. Memory Interfacing : Memory interfacing refers to the process of connecting external memory (RAM, ROM, EEPROM) with a microprocessor or microcontroller to expand its storage capacity. Since a microprocessor has limited internal memory, external memory is required for executing programs and storing data.

Key Aspects of Memory Interfacing:

1. **Address Bus & Address Decoding:**
- The microprocessor generates an address to access memory locations.
- Address decoding logic (using decoders or address lines) ensures correct memory selection.

2. **Data Bus Connection:**
- The bidirectional data bus allows data transfer between memory and the microprocessor.
- Width of the data bus (e.g., 8-bit, 16-bit, 32-bit) determines data transfer capacity.

3. **Control Signals:**
- Read (RD) and Write (WR) signals control memory operations.
- Chip Enable (CE) / Chip Select (CS) signals activate the required memory chip.

4. **Types of Memory Interfacing:**
- **ROM Interfacing:** Stores firmware and permanent programs.
- **RAM Interfacing:** Provides temporary storage for program execution.
- **EEPROM/Flash Memory:** Stores data even when power is off.

Example of Memory Interfacing in 8085:

- **Memory Address Range:** Suppose a system needs to connect 4 KB of RAM, we require 12 address lines (212=40962^{12} = 4096212=4096).
- **Decoder Circuit:** A 3-to-8 decoder may be used to select the correct memory chip.

Prob.22 ***Draw the block diagram of 8259 programmable interrupt controllers and explain its operation.*** **[R.T.U. 2024, 13]**

OR

Draw the architecture of 8259 PIC and explain. **[R.T.U. 2023]**

OR

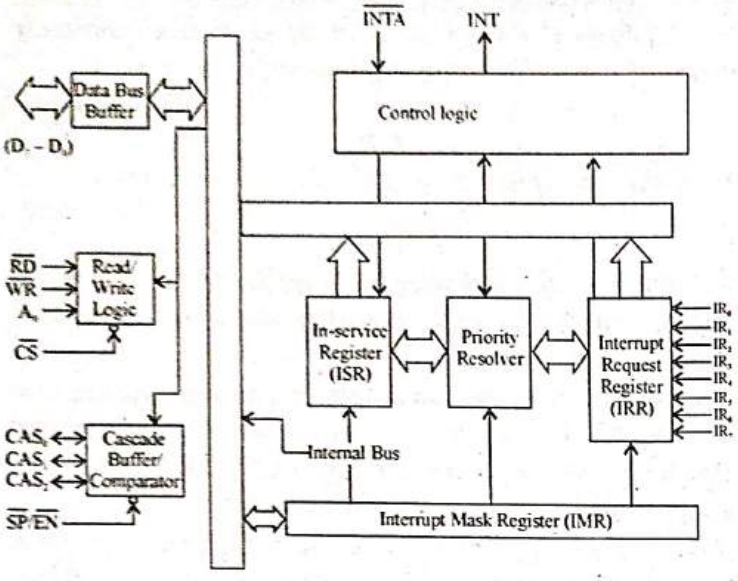
With the help of block diagram explain features of 8259 programmable interrupt controller and write how it can be programmed. **[R.T.U. 2012]**

Sol. The 8259A is a programmable interrupt controller designed to work with intel microprocessor 8085, 8086 and 8088. The 8259A interrupt controller can :

1. Manage eight interrupts according to the instructions written into its control register. This is equivalent to providing eight interrupt pins on the processor in place of one INTR (8085) pin.
2. Vector an interrupt request anywhere in the memory map. However, all eight interrupts are spaced at the interval of either four or eight locations. This eliminates the major drawback of the 8085 interrupts in which all interrupts are vectored to memory locations on page 00H
3. Resolve eight levels of interrupt priorities in a variety of modes, such as fully nested mode, automatic rotation mode, and specific rotation mode.
4. Mask each interrupt request individually.
5. Read the status of pending interrupts, in-service interrupts, and masked interrupts.
6. Be set up to accept either the level-triggered or the edge-triggered interrupt request.
7. Be expanded to 64 priority levels by cascading additional 8259As.
8. Be set up to work with either the 8085 microprocessor mode or the 8086/8088 microprocessor mode.

The 8259A is upward-compatible with its predecessor, the 8259. The main difference between the two is that the 8259A can be used with Intel's 8086/88 16-bit microprocessors. It also includes additional features such as the level triggered mode, buffered mode and automatic end of interrupt mode.

Block Diagram of the 8259A : Fig. 1 shows the internal block diagram of the 8259A. It includes eight blocks : control logic, Read/Write logic, data bus buffer, three registers (IRR, ISR and IMR), priority resolver and cascade buffer. This diagram shows all the elements of a programmable device, plus additional blocks. The functions of some of these blocks need explanation which is given below :



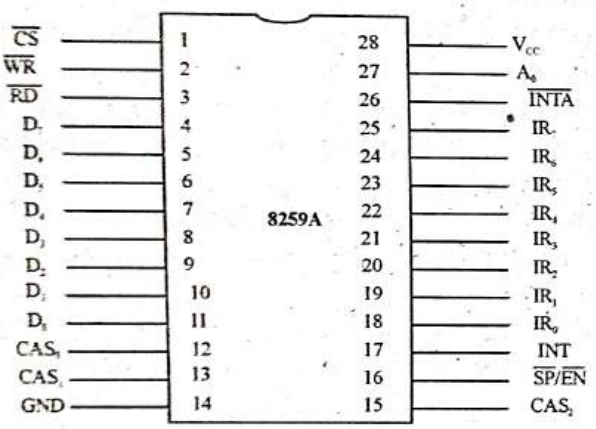


Fig. 1 : The 8259A Block Diagram and Pin Configuration

Read/Write Logic : This is a typical Read/Write control logic. When the address line A_0 is at logic 0, the controller is selected to write a command or read a status. The chip select logic and A_0 determine the port address of the controller.

Control Logic : This block has two pins: INT (Interrupt) as an output, and $\overline{INTA}$ (Interrupt Acknowledge) as an input. The INT is connected to the interrupt pin of the MPU. Whenever a valid interrupt is asserted, this signal goes high. The $\overline{INTA}$ is the interrupt acknowledge signal from the MPU

Interrupt Registers and Priority Resolver : The Interrupt Request Register (IRR) has eight input lines (IR_0–IR_7) for interrupts. When these lines go high, the requests are stored in the register. The In-Service Register (ISR) stores all the levels that are currently being serviced, and the Interrupt Mask Register (IMR) stores the masking bits of the interrupt lines to be masked. The Priority Resolver (PR) examines these three register and determines whether INT should be sent to the MPU.

Table : Pin Description

Pin	Name
$D_0 - D_7$	Bidirectional data bus
$\overline{RD}$	Read input
$\overline{WR}$	Write input
$CAS_2 - CAS_0$	Cascade lines
$\overline{CS}$	Chip select
$\overline{SP}/\overline{EN}$	Slave program / Enable Buffer
INT	Interrupt Output
$\overline{INTA}$	Interrupt Acknowledgement Input
$IR_0 - IR_7$	Interrupt request lines
A_0	Command Select Address

Programming the 8259A : The 8259A requires two types of command word. Initialization Command Words (ICWs) and Operational Command Words (OCWs).

The 8259A can be initialized with four ICWs; the first two are compulsory and the other two are optional based on the modes being used. These words must be issued in a given sequence. After initialization, the 8259 A can be set up to operate in various modes by using three different OCWs; however, they no longer need be issued in a specific sequence.

Flowchart

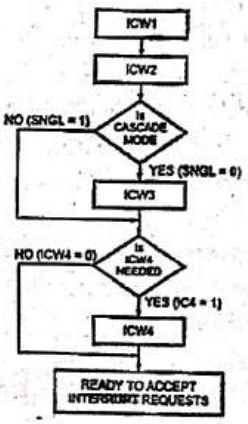


Fig. 2 : 8259A Initialization Flowchart

Initialization Command Word 1 (ICW1)

Fig.3 shows the initialization Command Word 1(1CW1).

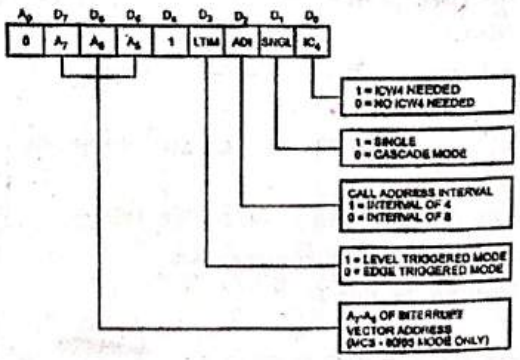


Fig. 3 : Initialization command word 1 (ICW1)

A write command issued to the 8259 with $A_0 = 0$ and $D_4 = 1$ is interpreted as ICW1, which starts the initliazation sequence. It specifies :

1. Single or multiple 8259As in the system.
2. 4 or 8 bit interval between the interrupt vector locations.
3. The address bits $A_7 - A_5$ of the CALL instruction.
4. Edge triggered or level triggered interrupts.
5. ICW_4 is needed or not.

Initialization Command Word 2 (ICW2)

Fig. 4. shows the initialization command word 2 (ICW2)

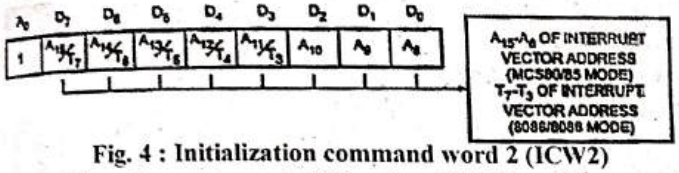


Fig. 4 : Initialization command word 2 (ICW2)

A write command following ICW1, with $A_0 = 1$ is interpreted as IC2. This is used to load the high order byte of the interrupt vector address of all interrupts.

Initialization Command Word 3 (ICW3)

ICW3 is required only if there is more than one 8259 in the system and if they are cascaded. An ICW3 operation loads a slave register in the 8259. The format of the byte to be loaded as an ICW3 for a master 8259 or a slave is shown in the fig.5. For master, each bit in ICW3 is used to specify whether it has a slave 8259 attached to it on its corresponding IR (Interrupt Request) input. For slave, bits D_0–D_2 of ICW3 are used to assign a slave identification code (slave ID) to the 8259.

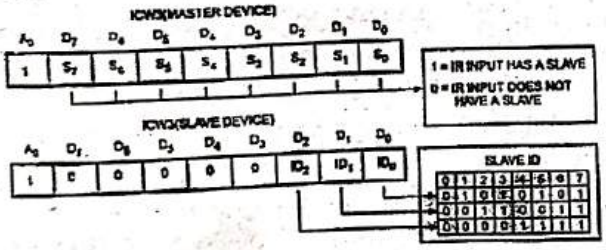


Fig. 5 : Initialization command word 3 (ICW3)

Initialization Command Word 4 (ICW4)

It is loaded only if the D_0 bit of ICW1 (IC4) is set. The format of ICW4 is shown in Fig.6.

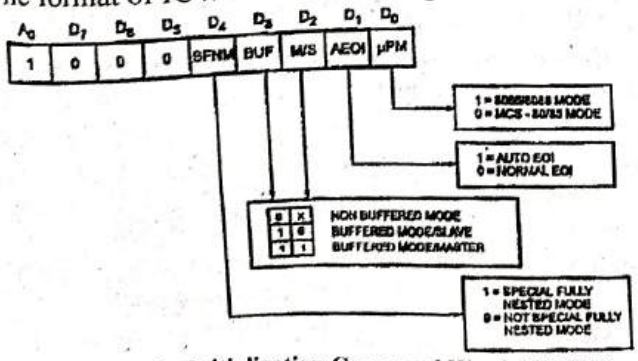


Fig. 6 : Initialization Command Word 4 (ICW4)

It specifies :

1. Whether to use special fully nested mode or non special fully nested mode.
2. Whether to use buffered mode or non buffered mode.
3. Whether to use Automatic EOI or Normal EOI.
4. CPU used, 8086/8088 or 80810.

Operation Command Words (OCWs)

After intialization, the 8259 is ready to process interrupt requests. However, during operation, it might be necessary to change the mode of processing the interrupts. Operation Command Words (OCWs) are used for this prupose. They may be loaded anytime after the 8259's initialization to dynamically alter the priority modes.

Operation Command Word 1 (OCW1)

A write command to the 8259 with $A_0 = 1$ (after ICW2) is interpreted as OCW1. OCW1 is used for enabling or disabling the recognition of specific interrupt requests by programming the IMR.

M = 1 indicates that the interrupt is to be masked and M = 0 indicates that it is to be unmasked as shown in Fig.7.

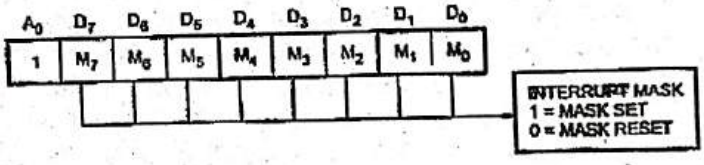


Fig.7 : Operation command word 1 (OCW1)

Operation Command Word 2 (OCW2)

A write command with $A_0 = 1$ and $D_4\ D_3 = 00$ is interpreted as OCW2. The R(Rotate), SL (Select-Level), EOI bits control the Rotate and End of interrupt modes and combinations of the two. Fig. 8 shows the operation command word format. L_2–L_0 are used to specify the interrupt level to be acted upon when the SL bit is active.

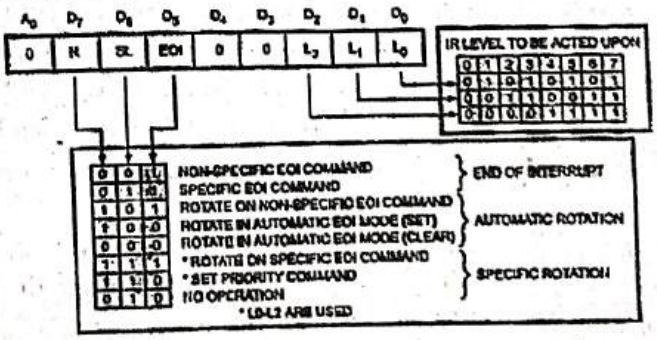


Fig.8 : Operation command word 2 (OCW2)

Operation Command Word 3 (OCW3)

OCW3 is used to read the status of the registers and to set or reset the special mask and polled modes. Fig.9 shows format operational command word 3.

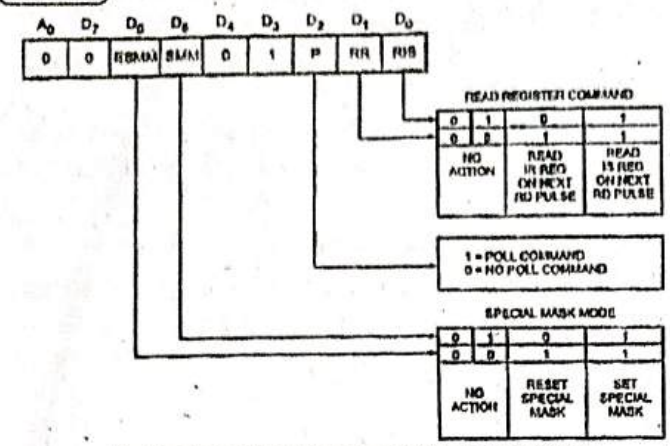


Fig.9 : Operation Command Word 3 (OCW3)

8259 Status Read Operations

The status of the interrupt request register, the in-service register and the interrupt mask register of the 8259 may be read by issuing appropriate read commands as described below.

IRR Status Read : An OCW3 with RR (Read Register) = 1 and RIS (Read ISR) = 0 set up the 8259 for a status read of the interrupt request register.

When the 8259 is not in the polled mode ,after it is set up for an IRR status read operation, all read commands with A_0=1 cause the 8259 to send the IRR stauts word.

ISR Status Read : An OCW3 with RR = 1 and RIS = 1 sets up the 8259 for a status read of the In-service Register. A subsequent read command issued to the 8259 will cause the 8259 to send the contents of the ISR onto the data bus.

IMR Status Read : A Read Command issued to the 8259 with A_0 = 1 (with $\overline{RD}, \overline{CS}$ = 0) causes. The 829 to put out the contents of the interrupt mask register. OCW3 is not required for a status read of the IMR.

As described earlier, the sequence shown in flowchart (fig.2) must be followed to initialize 8259A. Accoring to this flow chart an ICW1 and an ICW2 must be sent to any 8259A in the system. If the system has any slave 8259As (casade mode) them and ICW3 must be sent to the master and a difference ICW3 must be sent to the slave. If the system is an 8086 or if you want to specify certain special conditions, then you have to send an ICW4 to the master and to each slave. To have better understanding the initiation sequences for different specification are given in the next section.

Prob.23 ***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 : *Refer to Prob. 12.*

(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.

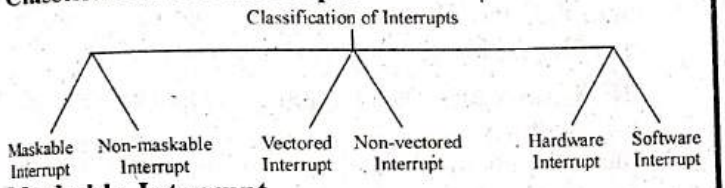
ADI01 H : The 2 bytes data O1H is added to the content of the accumulator and result is placed in the accumulator.

A CD ADI01 Before execution	ADDITION CDH = 11001101 O1H = 01 SUM 1001110	A CE After execution

Prob.24 *Explain :*
(a) Interrupts of 8085.
(b) RST instructions of 8085. **[R.T.U. 2017]**

Sol.(a) Interrupts of 8085 : *Refer to Prob.5.*

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.

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.

(iv) 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 $\overline{\text{INTA}}$ 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	0×8	= 0000H
RST 1	0001×8	= 0008 H
RST 2	0002×8	= 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. Table shows the list of different RST instructions.

Table : 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.

Prob.25 ***Differentiate between followings :***
(i) SIM and RIM instructions
(ii) Maskable and Non-maskable Interrupts
(iii) Vectored and Non-vectored Interrupts.
(iv) Hardware and Software Interrupts
[R.T.U. 2012]

Sol.(i) Difference between SIM and RIM Instruction

	SIM (Set Interrupt Mask)	RIM (Read Interrupt Mask)
1.	It is a one byte instruction. It requires only one machine cycle (4T).	It is single byte instruction. It does not provide status or TRAP flip-flop and INTR pin.
2.	It is used to enable or disable RST 7.5, RST 6.5 and RST 5.5 interrupts.	It is used to check status of all maskable interrupts.
3.	It transfers control word or SIM format from accumulator to the interrupt control logic. Hence the following control format must be loaded into accumulator before execution of SIM instruction.	This instruction transfers contents of interrupt control logic into the accumulator. Hence the accumulator is loaded with the status form at automatically after execution of RIM instruction.
4.	It transfers serial data bit (D_7) to the SOD pin.	It transfers serial data bit from SID line to D_7 bit of accumulator.
5.	It is not used to identify the pending interrupt bits D_4, D_5, D_6.	It is used to identify the pending interrupt bits D_4, D_5 and D_6.

(ii) Difference between Maskable and Non-maskable Interrupts : *Refer to Prob.15(b).*

(iii) Difference between Vectored and Non-vectored Interrupts : *Refer to Prob.7.*

(iv) Difference between Hardware and Software Interrupts

S. No.	Hardware Interrupt	Software Interrupt
1.	It is an asynchronous event.	It is synchronous event.
2.	This interrupt is requested by external device.	This interrupt is requested by executing instruction.
3.	PC is not incremented.	PC is incremented.
4.	The microprocessor executes either interrupt acknowledge cycle bus or idle machine cycle to acknowledge this interrupt.	The microprocessor does not execute any interrupt acknowledge cycle or idle machine cycle to acknowledge this interrupt. The microprocessor execute normal instruction cycle.
5.	It can be masked.	It cannot be ignored or masked.
6.	The priority is lower than that of software interrupt.	It has highest priority among all interrupts.
7.	It affect an interrupt control logic.	It does not affect an interrupt control logic.
8.	It is used to interface peripherals in interrupt driven I/O. It improves throughtput of the system.	It is not used to integrate peripheral that means it does not improve throughput of system. It is used in program debugging.

□□□

8085 MICROPROCESSOR INTERFACING 4

PREVIOUS YEARS QUESTIONS

PART-A

Prob.1 ***What are different modes of 8255 PPI?*** **[R.T.U. 2024]**

OR

List the operating modes of 8255 PPI. **[R.T.U. 2023]**

Sol. Operating Modes of 8255 PPI: There are two different modes of 8255. These modes are:

1. Bit Set Reset (BSR) Mode: This mode is used to set or reset the bits of the Port-C only. For BSR mode always D_7 will be 0.

2. Input/ Output Mode: This mode is selected when the D_7 bit of the control register is 1. This mode has also three different modes. These modes are Mode 0, Mode 1 and Mode 3.

(i) Mode 0 : Simple or basic I/O Mode
(ii) Mode 1 : Handshake or Strobed I/O
(iii) Mode 3 : Bidirectional I/O

Prob.2 ***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.3 ***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.4 ***Find the address of port A, port B, port C and control register of 8255 for following interfacing.***

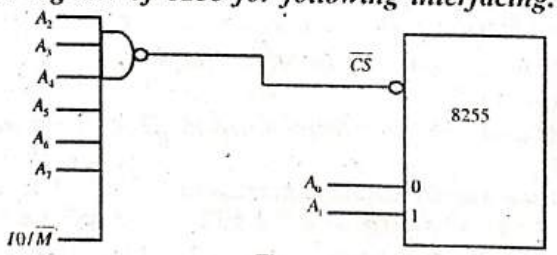


Fig.

[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.5 ***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, 08]

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.6 ***Explain the control word of 8254 program interval timer using suitable diagram.*** **[R.T.U. 2024]**

OR

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 ***Discuss bidirectional handshaking in 8255 PPI.*** **[R.T.U. 2023]**

Sol. The Intel 8255 Programmable Peripheral Interface (PPI) is a widely used integrated circuit that provides parallel I/O capabilities to microprocessor-based systems. It consists of three 8-bit bidirectional ports (Port A, Port B and Port C) which can be programmed for input or output operations. Bidirectional handshaking is a communication mechanism used to ensure synchronized data transfer between two devices, typically between a microprocessor and external peripherals, using the 8255 PPI.

Here's how bidirectional handshaking works with the 8255 PPI:

1. **Control Signals:** The 8255 PPI provides control signals such as "RD" (Read), "WR" (Write), and "ALE" (Address Latch Enable) which are used for communication between the microprocessor and the

external devices.These control signals are used to initiate and acknowledge data transfer operations between the microprocessor and the 8255 PPI.

2. **Handshake Lines:** The 8255 PPI supports handshake lines that facilitate bidirectional communication. These handshake lines include "ACK" (Acknowledge) and "INT" (Interrupt) lines. The ACK line is used by the microprocessor to acknowledge the completion of an input or output operation. The INT line is used by the 8255 PPI to request an interrupt to the microprocessor upon completion of an input or output operation.
3. **Read Operation:** During a read operation, the microprocessor sends a read command (asserts RD) along with the address of the 8255 PPI. The 8255 PPI responds by placing the requested data on the data lines (Port A, Port B, or Port C). After receiving the data, the microprocessor acknowledges the completion of the operation by asserting the ACK signal.
4. **Write Operation:** During a write operation, the microprocessor sends a write command (asserts WR) along with the address of the 8255 PPI and the data to be written. The 8255 PPI latches the data from the microprocessor and performs the specified operation. Upon completing the operation, the 8255 PPI signals the microprocessor by asserting the ACK signal.
5. **Interrupt Handling:** The 8255 PPI can be configured to generate an interrupt (INT) signal to the microprocessor upon completing certain input or output operations. This allows the microprocessor to perform other tasks while waiting for the completion of I/O operations, enhancing system efficiency.

Bidirectional handshaking with the 8255 PPI ensures reliable and synchronized data transfer between the microprocessor and external devices, enabling efficient communication and control in microprocessor-based systems.

Prob.8 ***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.

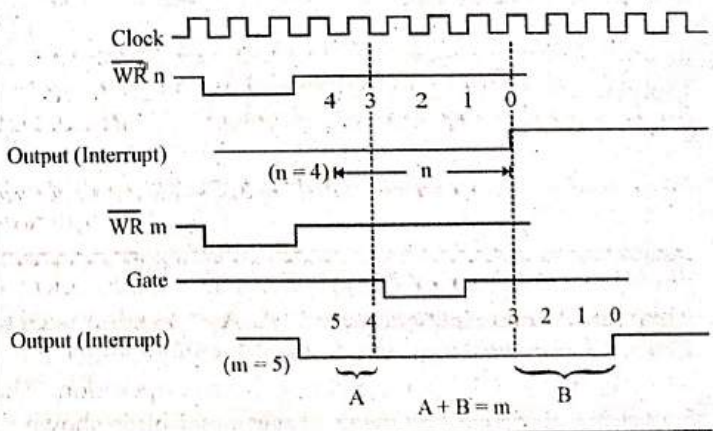
Mode 4 : Software-Triggered Strobe

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



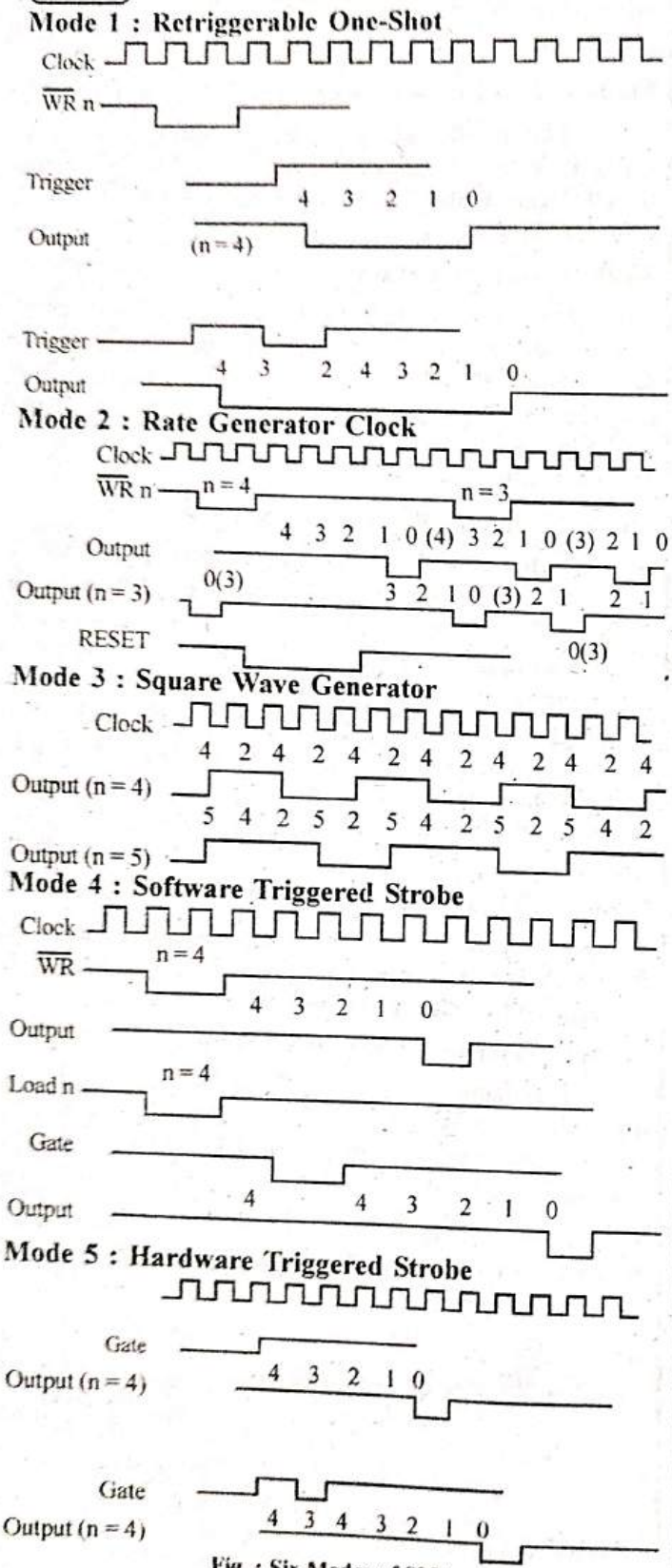


Fig. : Six Modes of 8254

Prob.9 *Draw and explain diagram of 8253.*

[R.T.U. 2015]

Sol.

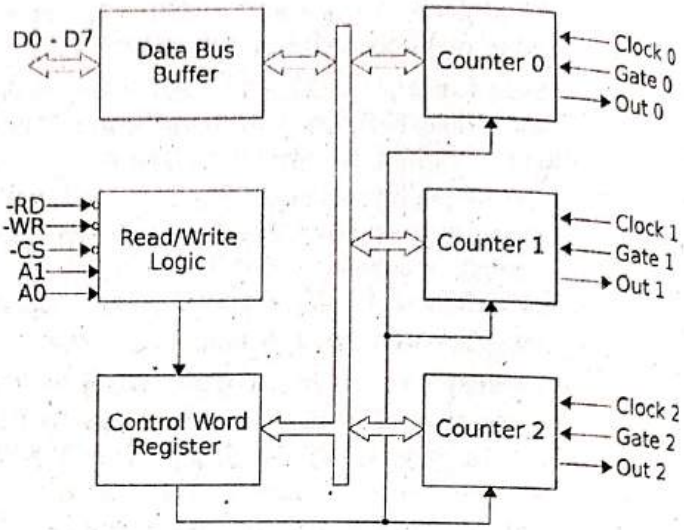


Fig. : Block Diagram of Intel 8253

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.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

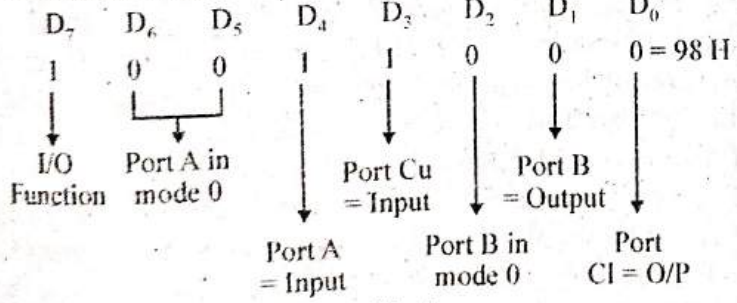


Fig. 1

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 0F 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		

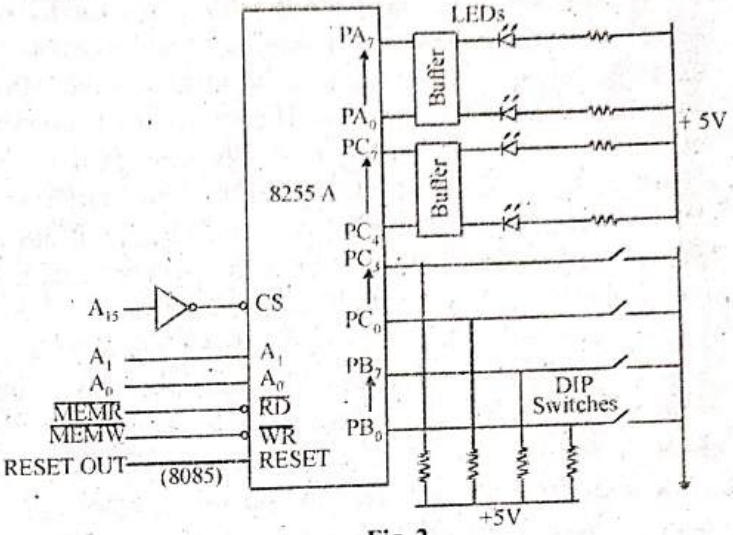


Fig. 2

PART-C

Prob.11 ***Explain the use of control word for 8254 PIT.*** **[R.T.U. 2023]**

Sol. Use of Control Word in 8254 PIT: The 8254 Programmable Interval Timer (PIT) uses a special 8-bit instruction called the Control Word to determine how each of its three counters will operate. This control word sets up the mode, counting method and the read/write format for each timer.

The Control Word is used to:

- Select which counter to configure (Counter 0, Counter 1, or Counter 2).
- Choose the read/write operation format (LSB, MSB, or both).
- Set the operating mode (Mode 0 to Mode 5).
- Define whether counting is in Binary (16-bit) or BCD (Binary Coded Decimal).

For more: *Refer to Prob.6.*

Prob.12 ***Draw the block diagram of 8279 Keyboard/ display interface. Explain its operation also.*** **[R.T.U. 2022, 15]**

OR

Explain 8279 Keyboard Interface in detail. **[R.T.U. 2019]**

OR

Explain working and modes of 8279 keyboard/display interface. [R.T.U. 2014]

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.

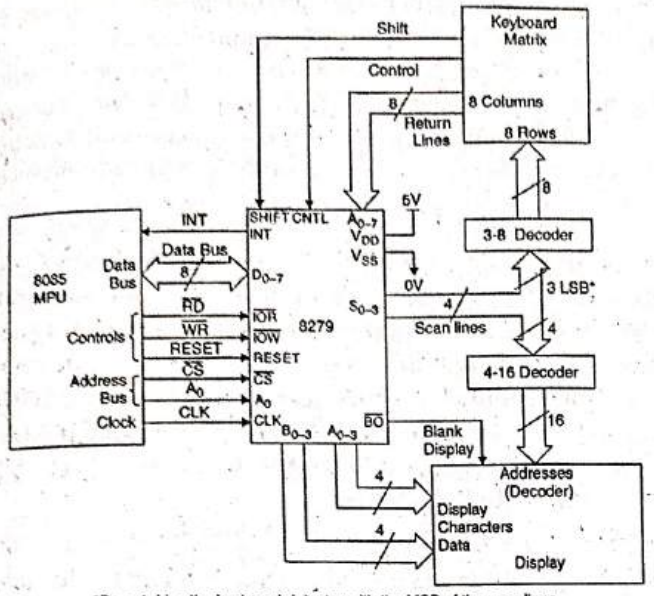
(iv) Sensor Matrix Mode : In the sensor matrix mode, the debounce logic is inhibited. The 8-byte FIFO RAM now acts as 8 × 8-bit memory matrix. The status of the sensor switch matrix is fed directly to sensor RAM matrix. Thus the sensor RAM bits contain the row-wise and column-wise status of the sensors in the sensor matrix. The IRQ line goes high, if any change in sensor value is detected at the end of a sensor matrix scan or the sensor RAM has a previous entry to be read by the CPU. The IRQ line is reset by the first data read operation, if AI = 0, otherwise, by issuing the end interrupt command. AI is a bit in read sensor RAM word.

Display Modes : There are various options of data display. For example, the command number of characters can be 8 or 16, with each character organized as single 8-bit or dual 4-bit codes. Similarly there are two display formats. The first one is known as left entry mode or type writer mode, since in a type writer the first character typed appears at the left-most position, while the subsequent characters appear successively to the right of the first one. The other display format is known as right entry mode, or calculator mode, since in a calculator the first character entered appears at the rightmost position and this character is shifted one position left when the next character is entered. Thus all the previously entered characters are shifted left by one position when a new character is entered.

(i) Left Entry Mode : In the left entry mode, the data is entered from the left side of the display unit. Address 0 of the display RAM contains the leftmost display character and address 15 of the RAM contains the rightmost display character. It is just like writing in our note books, *i.e.* from left to write. If the 8279 is in autoincrement mode, the display RAM address is automatically updated with successive reads or writes. The first entry is displayed on the leftmost display and the sixteenth entry on the rightmost display. The seventeenth entry is again displayed at the leftmost display position.

(ii) Right Entry Mode : In the right entry mode, the first entry to be displayed is entered on the rightmost display. The next entry is also placed in the right most display but after the previous display is shifted left by one display position. The leftmost character is shifted out of that display at the seventeenth entry and is lost, i.e. it is pushed out of the display RAM.

Interfacing of 8279 with microprocessor 8085:



*Do not drive the keyboard detector with the MSB of the scan lines.

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.

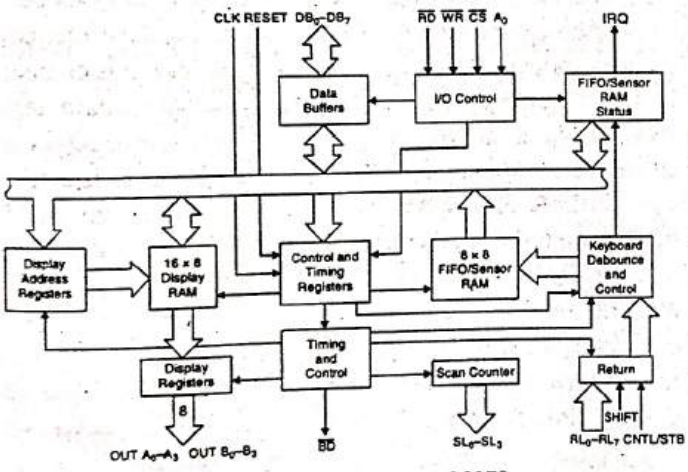


Fig. 2 : Block Diagram of 8279

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.

Prob.13 ***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.

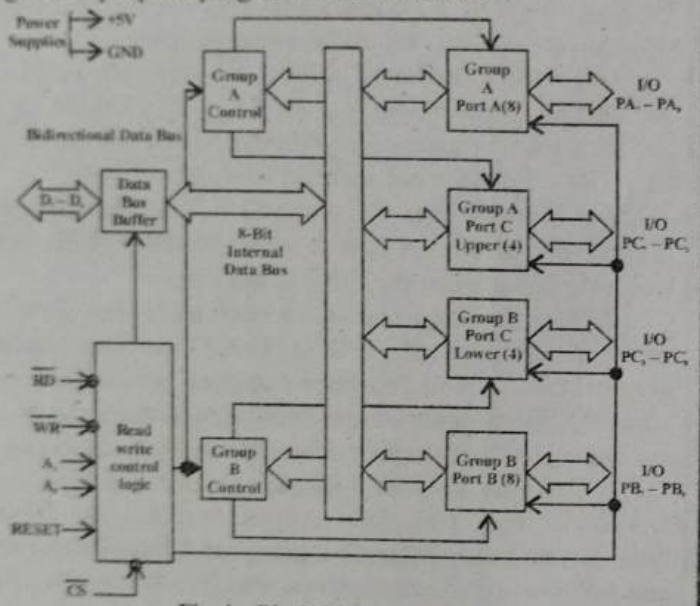


Fig. 1 : Block Diagram of 8255A

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 $D_7 = 1$)

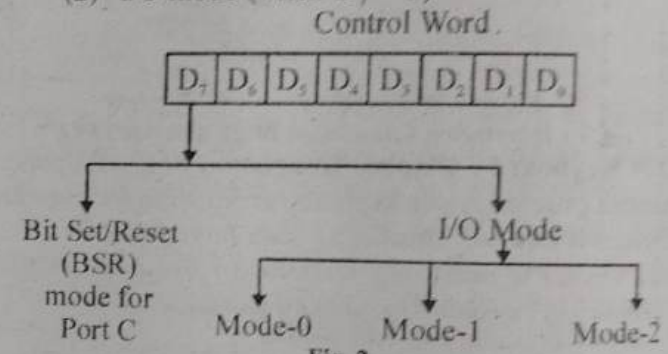


Fig. 2

(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.

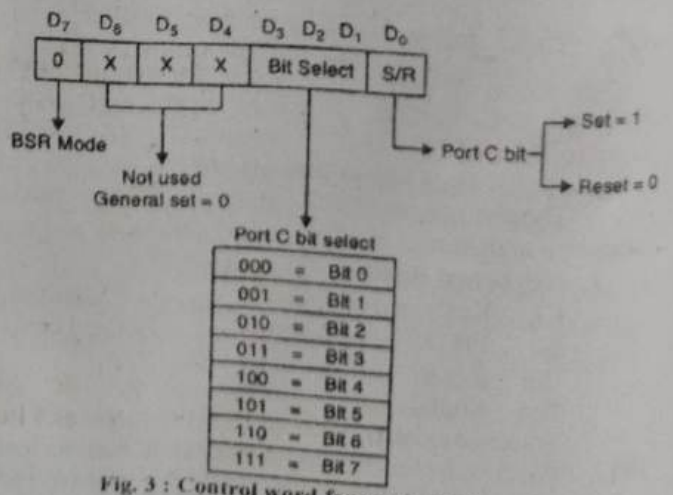


Fig. 3 : Control word format in BSR mode

(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.

Prob.13 ***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.

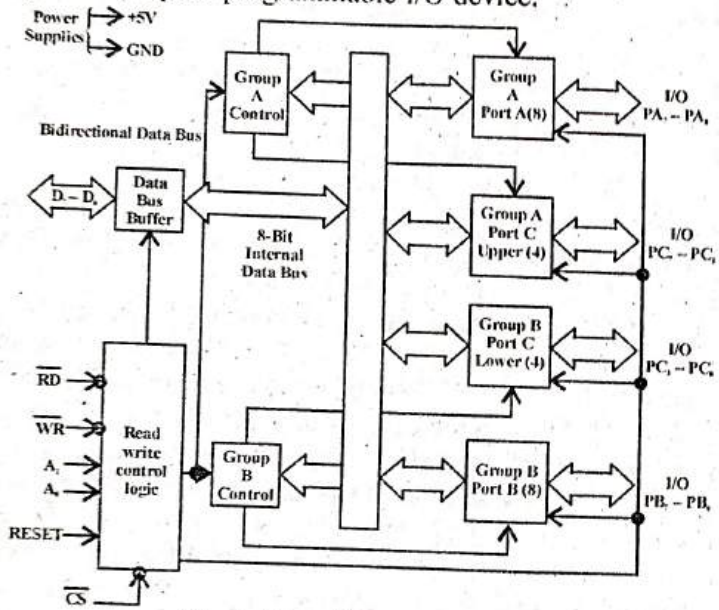


Fig. 1 : Block Diagram of 8255A

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 $D_7 = 1$)

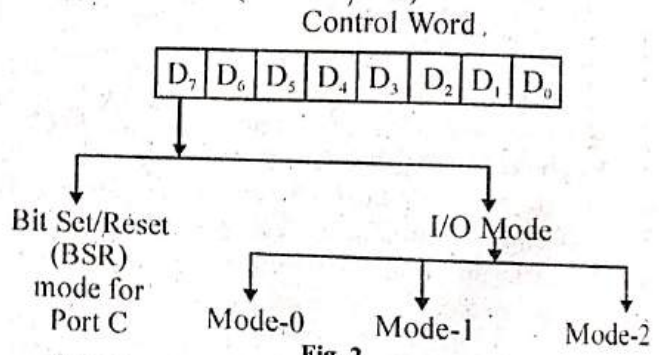


Fig. 2

(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.

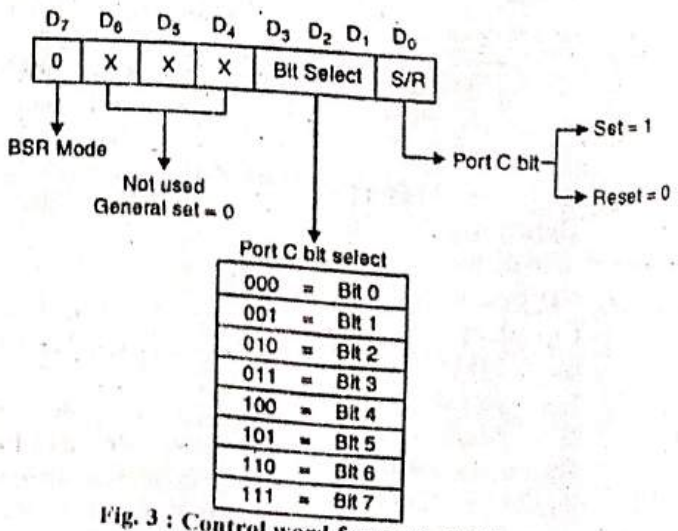


Fig. 3 : Control word format in BSR mode

(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.

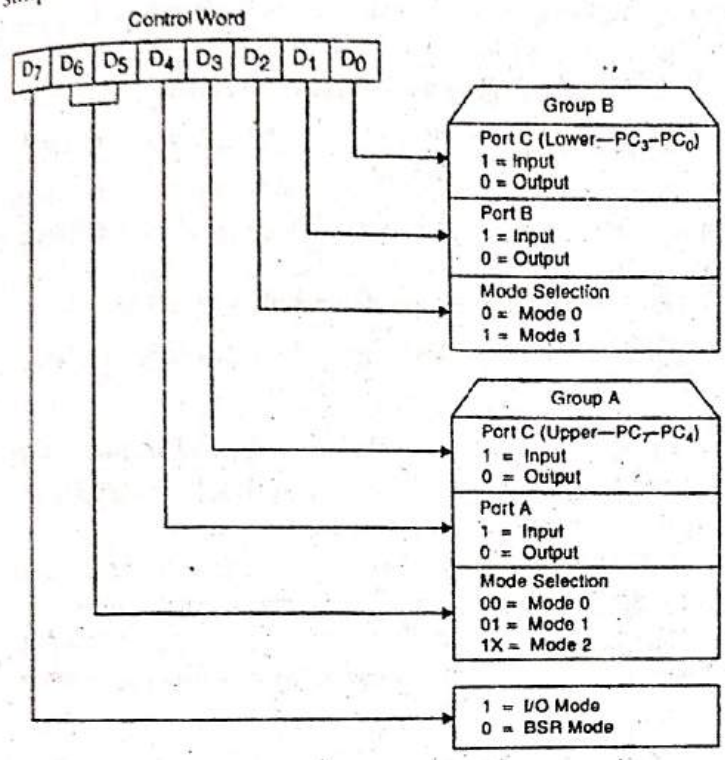


Fig. 4 : 8255 control word format for I/O mode

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:

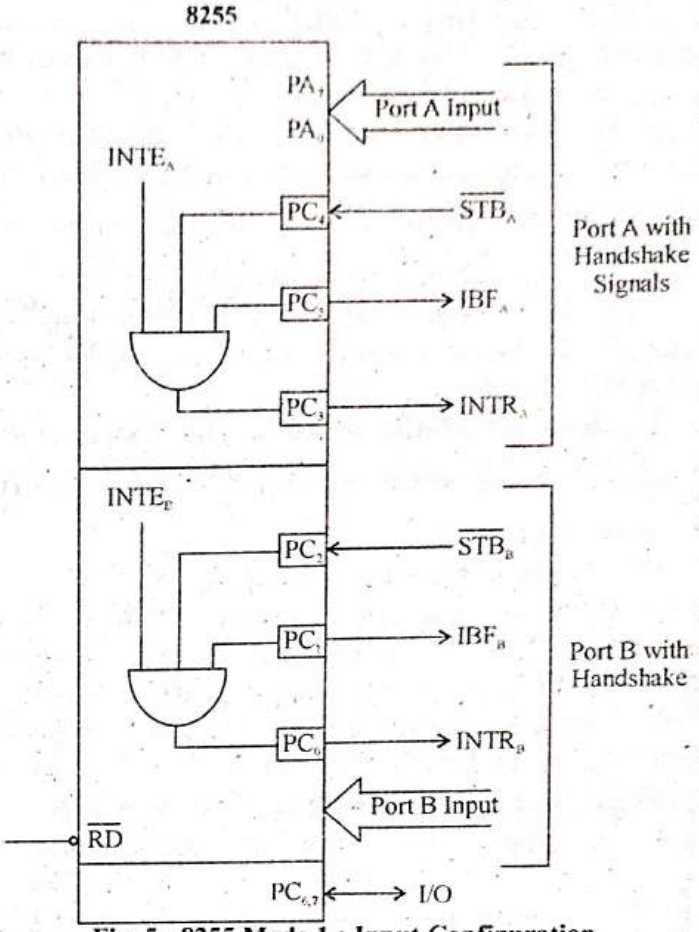


Fig. 5 : 8255 Mode 1 : Input Configuration

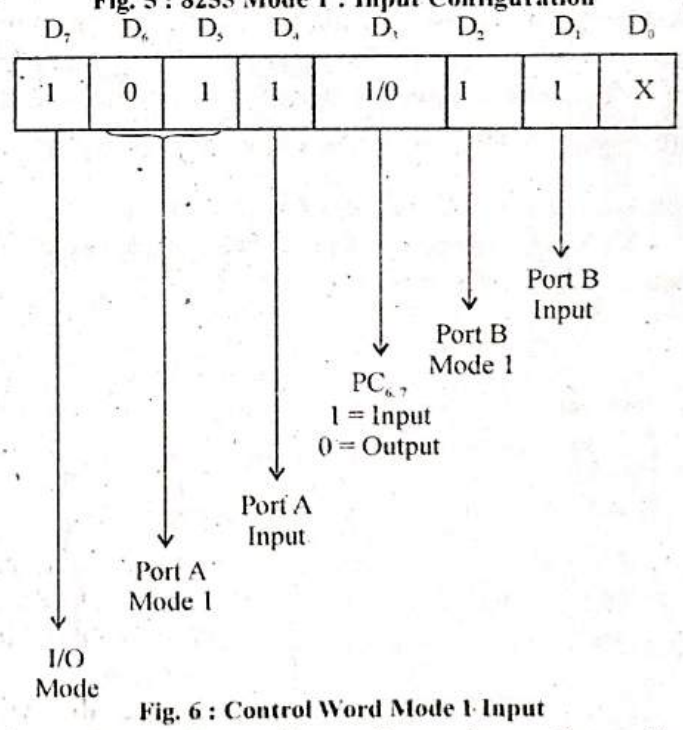


Fig. 6 : Control Word Mode 1 Input

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
I/O	I/O	IBF_A	$INTE_A$	$INTR_A$	$INTE_B$	IBF	$INTR_B$

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.

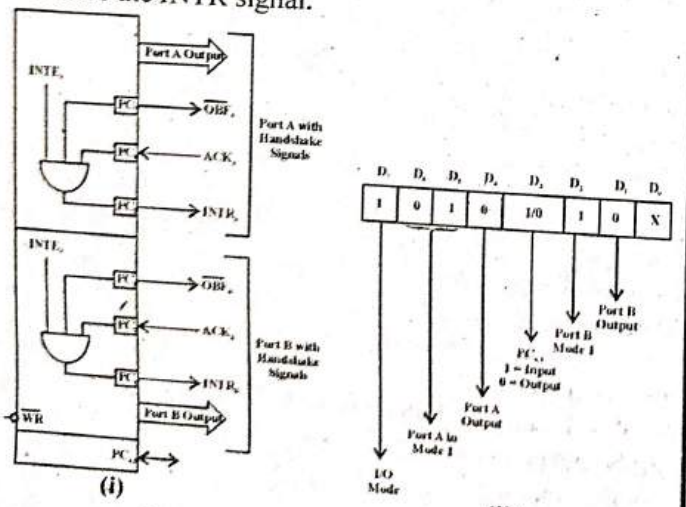


Fig. 8 : 8255 Mode 1 : Output Configuration

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.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
$\overline{OBF_A}$	$INTE_1$	IBF_A	$INTE_2$	$INTR_A$	X	X	X
For Port A					For Port B		

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

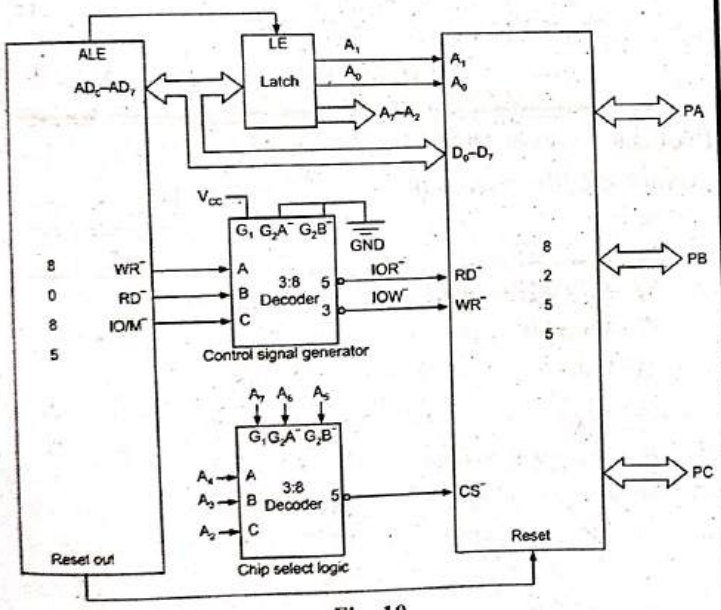


Fig. 10

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.

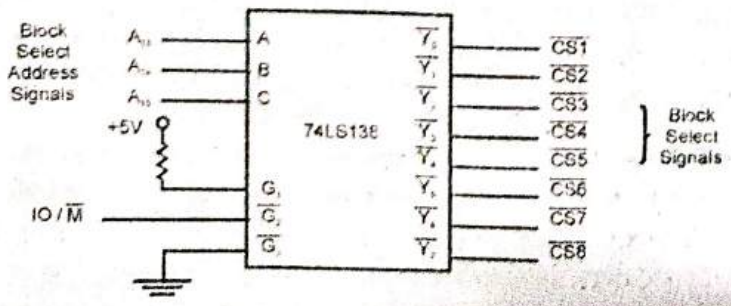


Fig. 11

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.

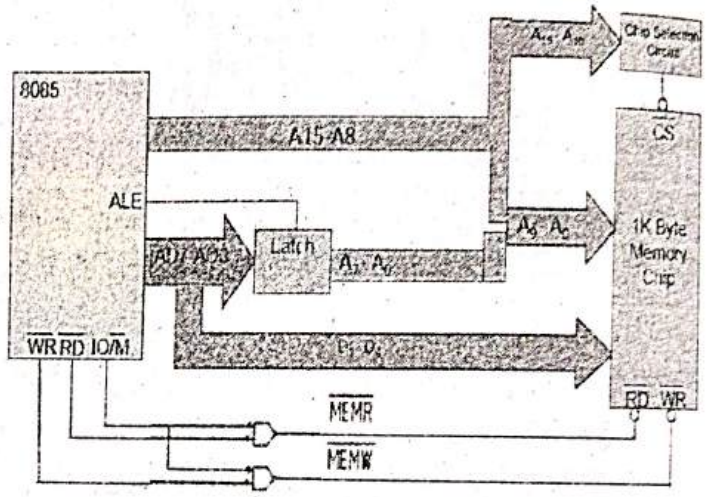


Fig. 12

Prob.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 PIT : *Refer to Prob.8.*

Prob.15 ***Draw and explain the pin diagram of 8255 PPI? Explain all of its modes?*** **[R.T.U. 2015]**

Sol. Pin diagram of 8255 PPI

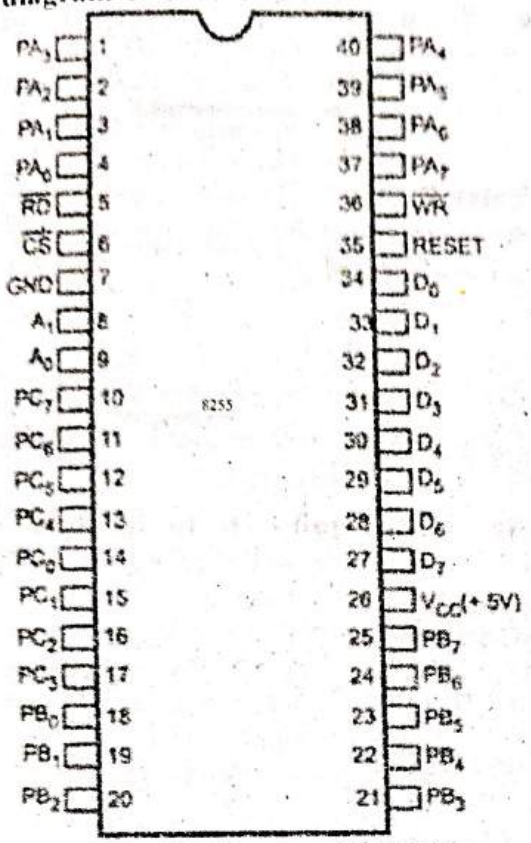


Fig. : Pin diagram of 8255 PPI

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.

Modes of 8255 PPI : *Refer to Prob.13.*

Prob.16 ***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.

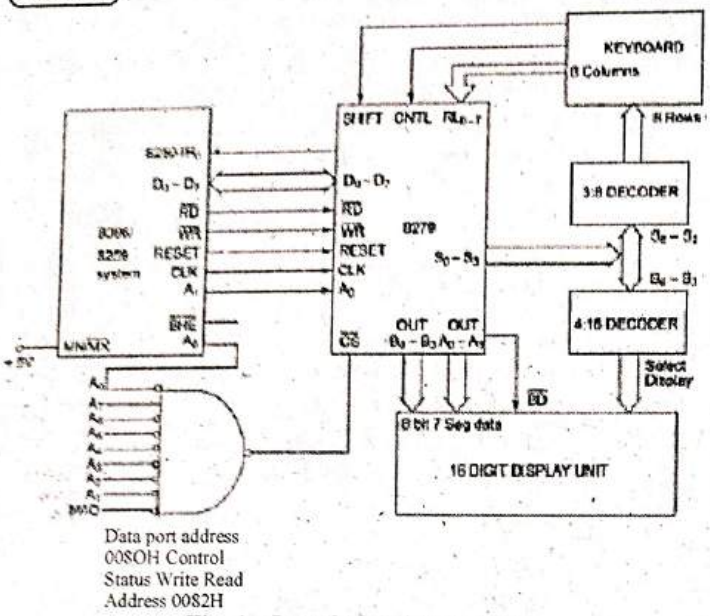


Fig. 1 : Interfacing Schematic

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.

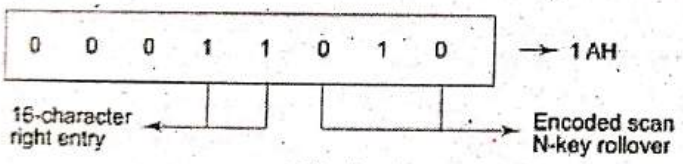


Fig. 2

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.

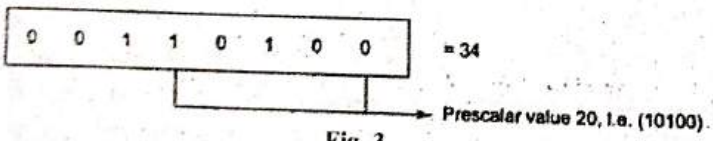


Fig. 3

Clear Display RAM : This command clears the display RAM with the programmable blanking code.

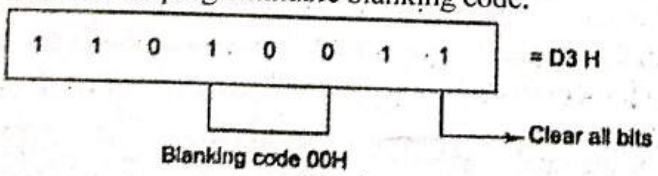


Fig. 4

Read FIFO : This command byte enables the programmer to read a key code from the FIFO RAM.

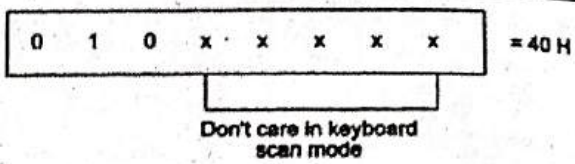


Fig. 5

Write Display RAM : This command enables the programmer to write the addressed display locations of the RAM as presented below.

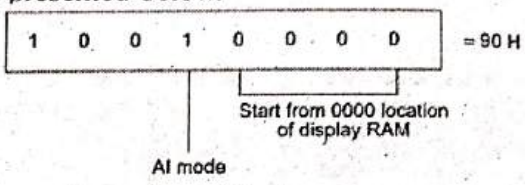


Fig. 6

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, OD3H    :   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
```

□□□

MICROPROCESSOR APPLICATIONS 5

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. Difference between Serial and Parallel Communication:

Serial Communication	Parallel Communication
Data is transmitted bit by bit, one after another on a single line.	Data is transmitted multiple bits at once using multiple lines.
Slower compared to parallel for short distances, but more efficient for long-distance communication.	Faster for short distances due to simultaneous transmission of bits.
Requires fewer wires, reducing cost and complexity.	Requires more wires (one for each bit), increasing cost and complexity.
Less prone to crosstalk and signal degradation over long distances.	More prone to noise and crosstalk due to multiple lines.
Commonly used in USB, RS-232, and UART communication.	Commonly used in data buses inside computers, like RAM to CPU communication.
Cheaper and simpler for long-distance transmission.	Expensive and bulky for long-distance communication.

Prob.3 *Draw a schematic diagram to show how RS 232C cable is connected through 8251 to 8085.*

[RTU 2011]

Sol. Block Diagram

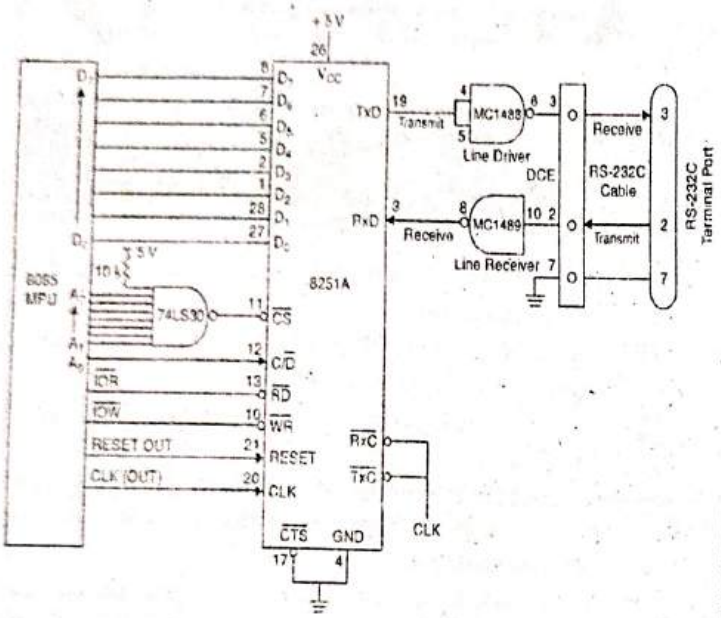


Fig. : Schematic of Interfacing an RS-232 Terminal with an 8085 System using the 8251A

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 *Compare RS232C and RS422A serial data standards.*

[R.T.U. 2024]

Sol. Comparison of RS232C and RS422A Serial Data Standards :

RS-232C	RS-422A
Single-ended signal transmission.	Differential signal transmission.
Supports point-to-point communication.	Supports point-to-multipoint (1 transmitter, up to 10 receivers).
Maximum cable length: 50 feet (15 meters).	Maximum cable length: 4000 feet (1200 meters).
Maximum data rate: up to 20 kbps.	Maximum data rate: up to 10 Mbps.
Uses ±3V to ±25V for signaling.	Uses 0V to ±5V differential signaling.
Less noise immune, more prone to interference.	More noise immune due to differential signaling.
Common in older PCs, modems, printers.	Used in industrial and long-distance applications.
Typically uses DB9 or DB25 connectors.	Typically uses twisted-pair cabling.

Prob.6 *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 standard 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.

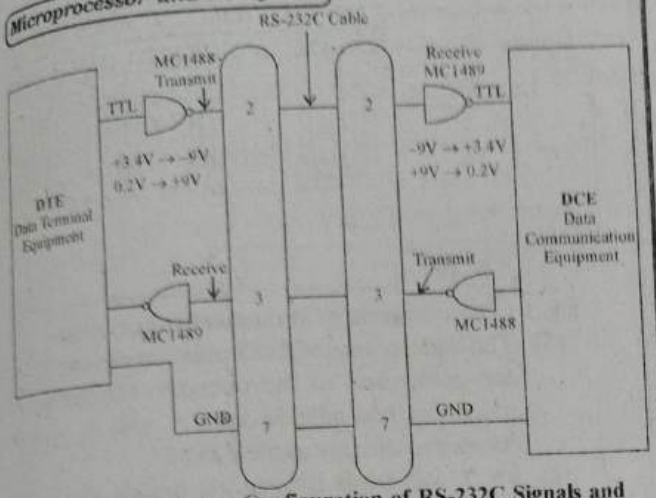


Fig. 1(a) : Minimum Configuration of RS-232C Signals and Voltage Levels

Signals	Pins		Signals
	14	1	Protective Ground
Secondary Transmitted Data	15	2	Transmitted Data (T_xD) → DCE
Transmission Signal Element Timing (DCE Source)	16	3	Received Data (R_xD) → DTE
Secondary Received Data	17	4	Request to Send (RTS) → DCE
Receiver Signal Element Timing (DCE Source)	18	5	Clear to Send (CTS) → DTE
Unassigned	19	6	Data Set Ready (DSR) → DTE
Secondary Request to Send	20	7	Signal Ground
DCE ← Data Terminal Ready (DTR)	21	8	Received Line Signal Detector
Signal Quality Detector	22	9	(Reserved for Data Set Testing)
Ring Indicator	23	10	(Reserved for Data Set Testing)
Data Signal Rate Selector (DTE/DCE Source)	24	11	Unassigned
Transmit Signal Element Timing (DTE Source)	25	12	Sec. Read Line Sig. Detector
Unassigned		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.

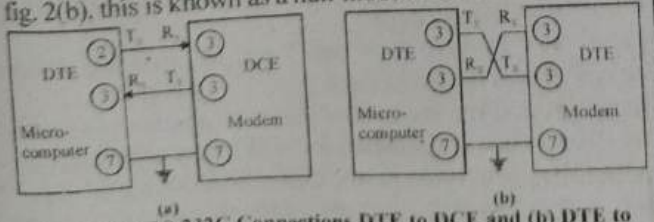


Fig. 2 : (a) RS-232C Connections DTE to DCE and (b) DTE to DTE

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 tq 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.7 ***Write short note on liquid crystal display.***

[R.T.U. 2016]

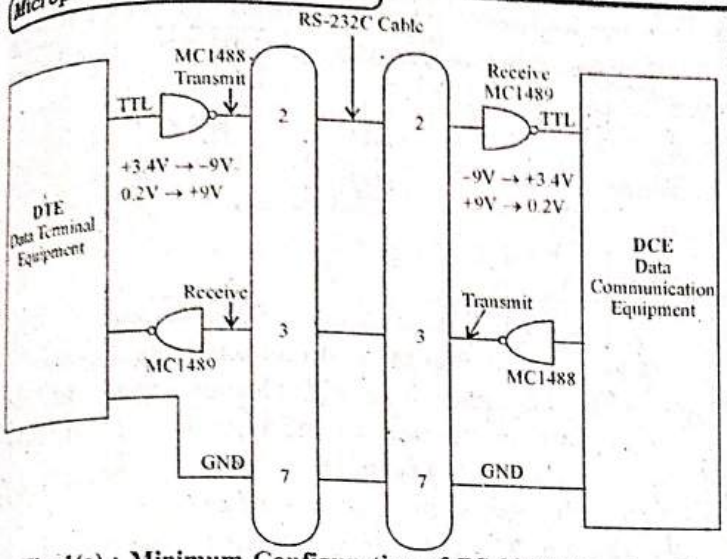


Fig. 1(a) : Minimum Configuration of RS-232C Signals and Voltage Levels

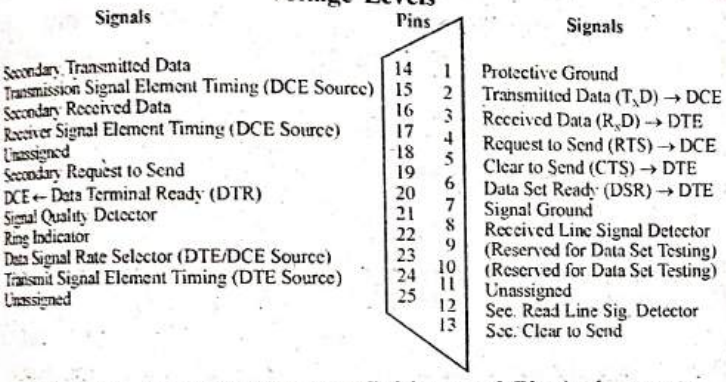


Fig. 1(b) : RS-232C Signal Definitions and Pin Assignments

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.

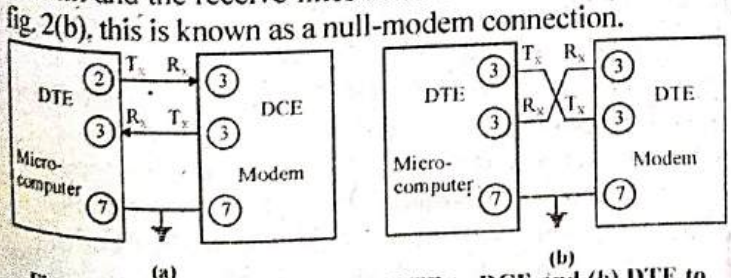


Fig. 2 : (a) RS-232C Connections DTE to DCE and (b) DTE to DTE

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.7 ***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 68HCl2 and the LCD display, you are actually establishing an interface between two processors.

Prob.8 ***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.

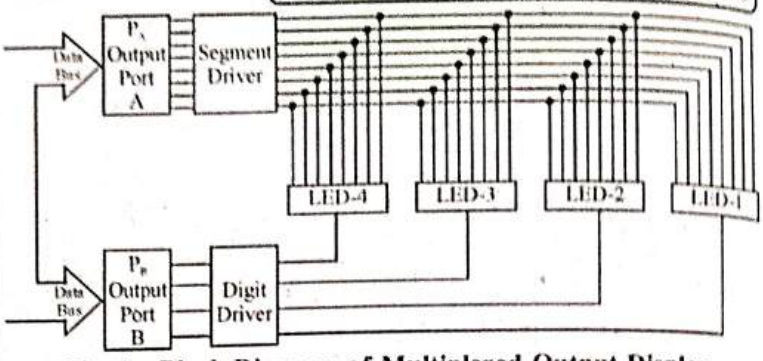


Fig. 1 : Block Diagram of Multiplexed Output Display

(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.

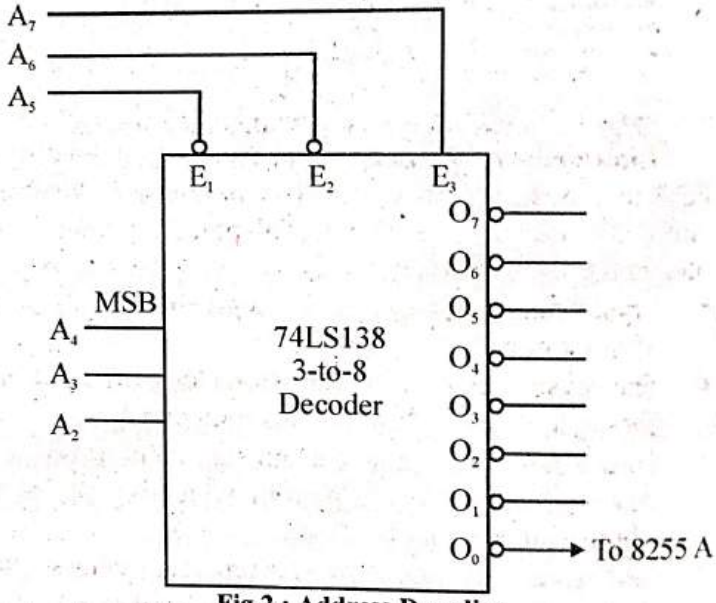


Fig.2 : Address Decoding

(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.

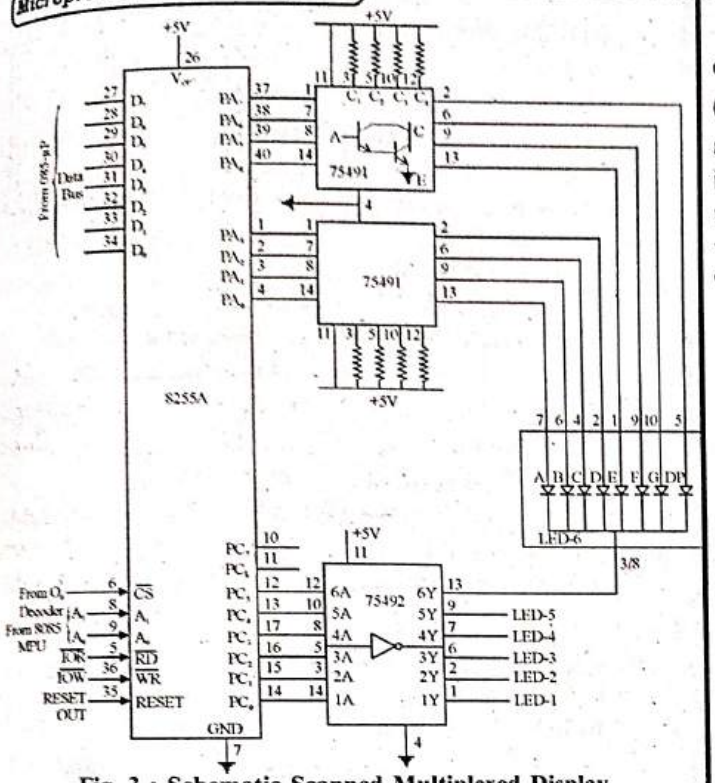


Fig. 3 : Schematic Scanned Multiplexed Display

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 :

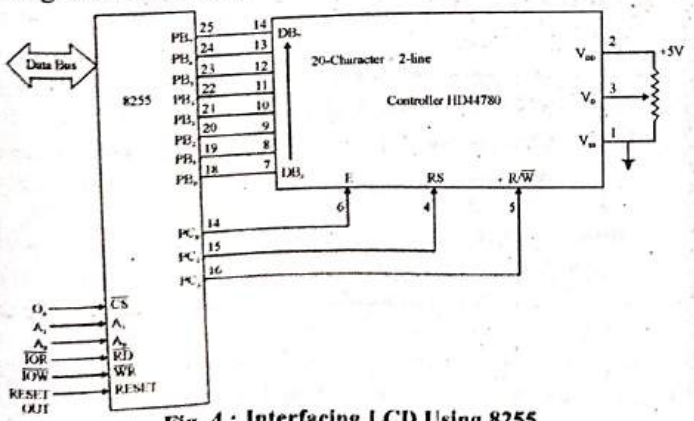


Fig. 4 : Interfacing LCD Using 8255

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.

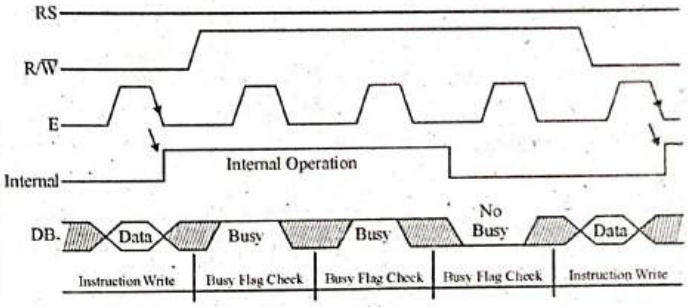


Fig. 5 : Data Transfer Timing Diagram

- 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.

PART-C

Prob.9 ***Write short note on IEEE 488.*** **[R.T.U. 2024]**

Sol. 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.

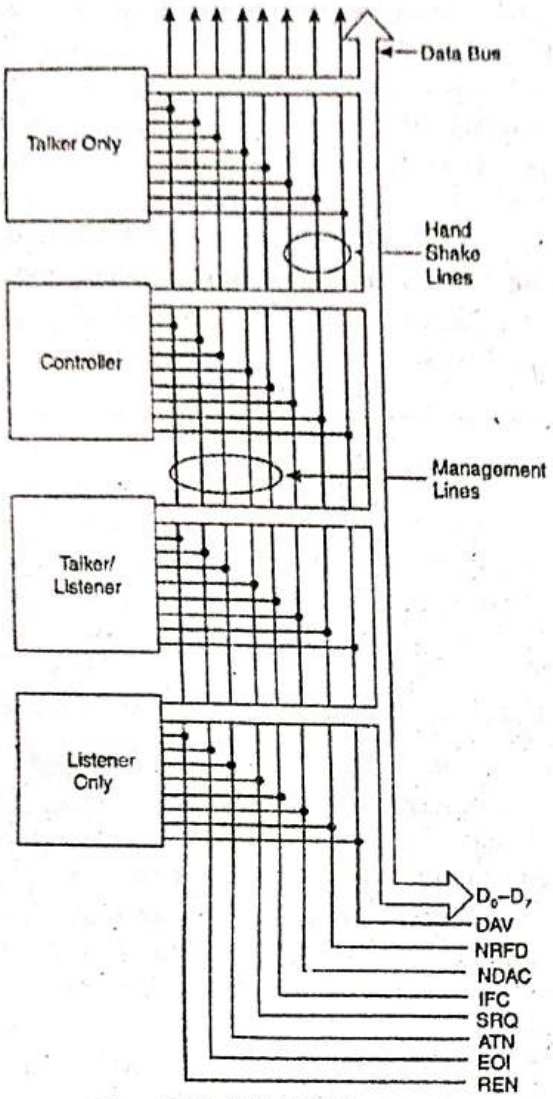


Fig. : IEEE 488 (GPIB Structure)

The interfacing device for IEEE 488 can be divided into 3 categories :

1. Listener
2. Talker
3. Controller

1. Listener : These are instruments which only accepts data from other instruments. For example, printers, displays, programmable power supply, programmable signal generator etc. All the examples of listener will be an output devices.

2. Talker : These are instruments which only sends data to other instruments. For example, Digital voltmeter, tape 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.10 ***How Liquid crystal display is interfaced with 8085 microprocessor? Explain it using neat diagram.*** **[R.T.U. 2024]**

OR

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.

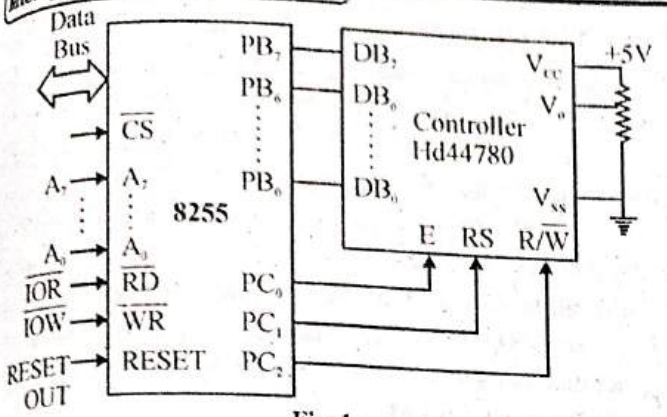


Fig. 1

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_0, 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.

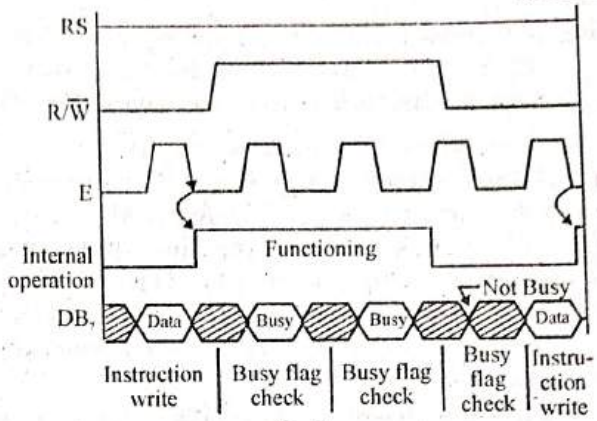


Fig. 2

Prob.11 ***Draw the internal block diagram of 8251 USART and explain its initialization process.*** **[R.T.U. 2024, 22]**

OR

Write short note on USART 8251. **[R.T.U. 2016]**

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 $\overline{TxC}$.

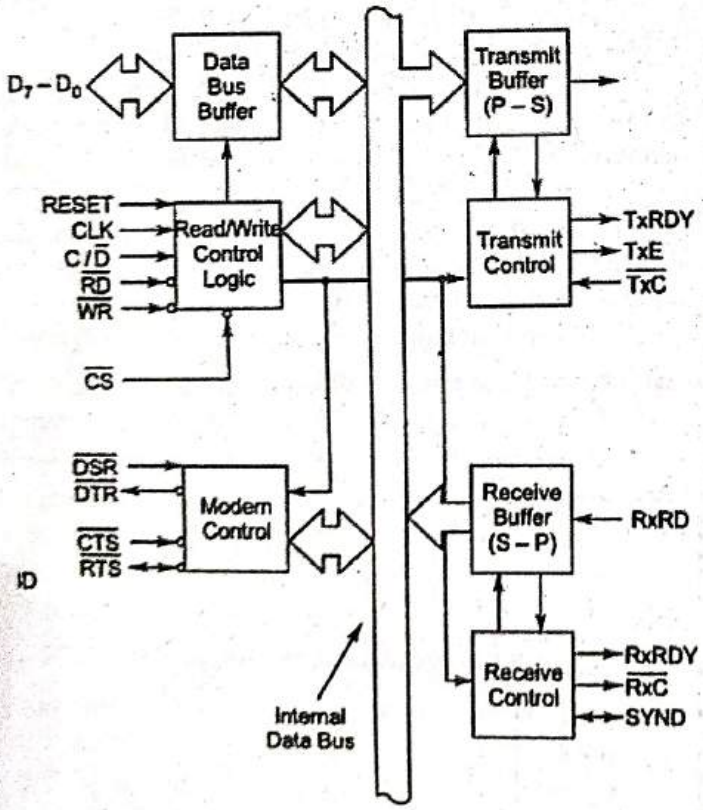


Fig. 1

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

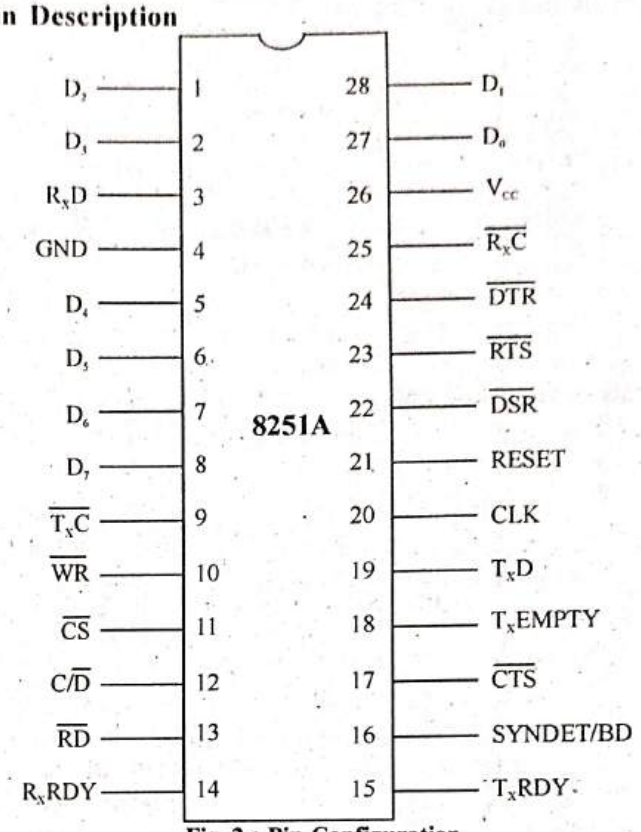


Fig. 2 : Pin Configuration

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	V_{CC}	+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 Mode
0	1 = Stop bit	0	1 = Odd parity	0	1 = 6 bits	01	ASYN × 1
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 × 64
Framing Control		Parity Control		Character Length		Baud Rate Factor	

Command Word Format

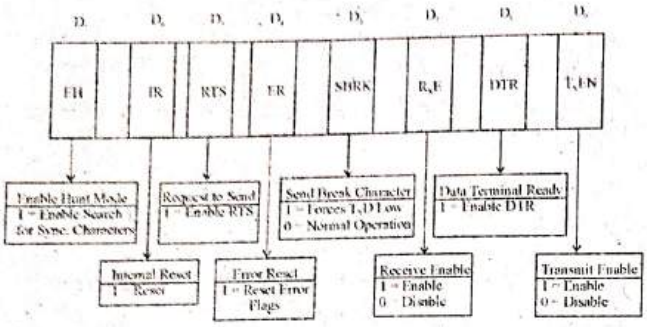


Fig. 3

Status Word Format

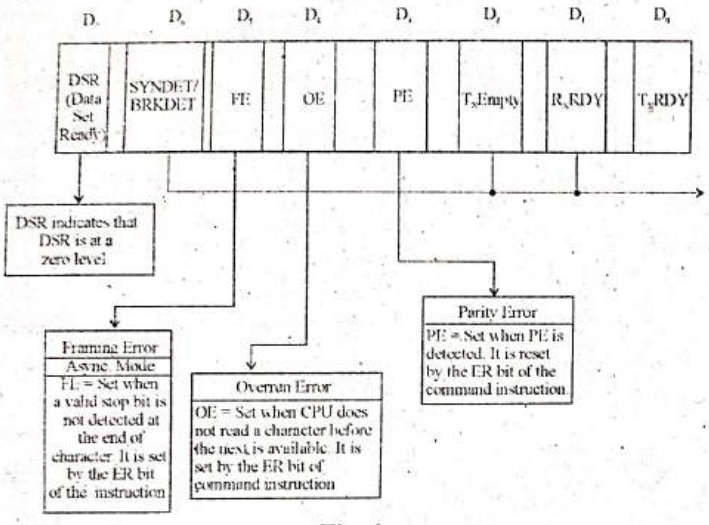


Fig. 4

Prob.12 ***Write technical note on RS422A and IEEE488.*** **[R.T.U. 2023]**

Sol. RS422A : 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.

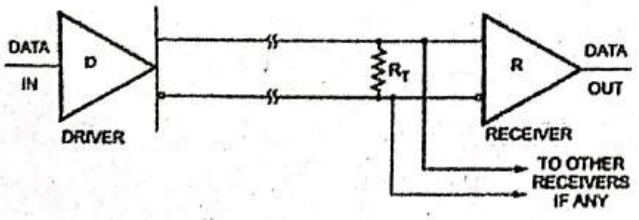


Fig. : RS 422A interface circuit

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.

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.

IEEE488 : *Refer to Prob.9.*

Prob.13 ***Compare RS-422 A and RS-423 A serial data standard.*** **[R.T.U. 2022, 2015]**

Sol. 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.

Prob.14 ***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.

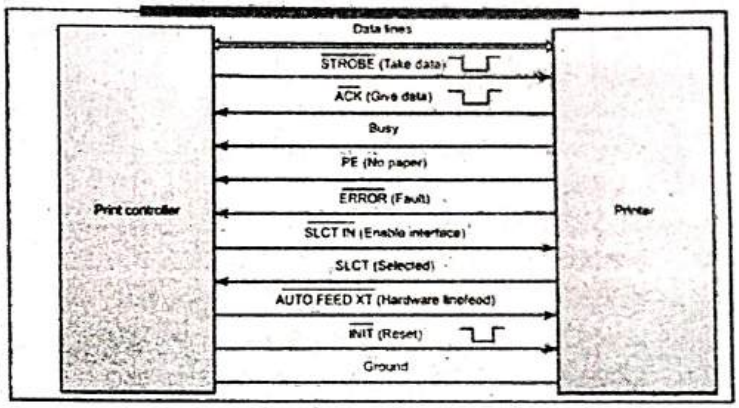


Fig. : Signals in Centronics Interface

IEEE 488 : *Refer to Prob.9.*

Prob.15 ***Explain the organization and architecture of 8251 (USART) with a functional block diagram.*** **[R.T.U. 2019]**

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.

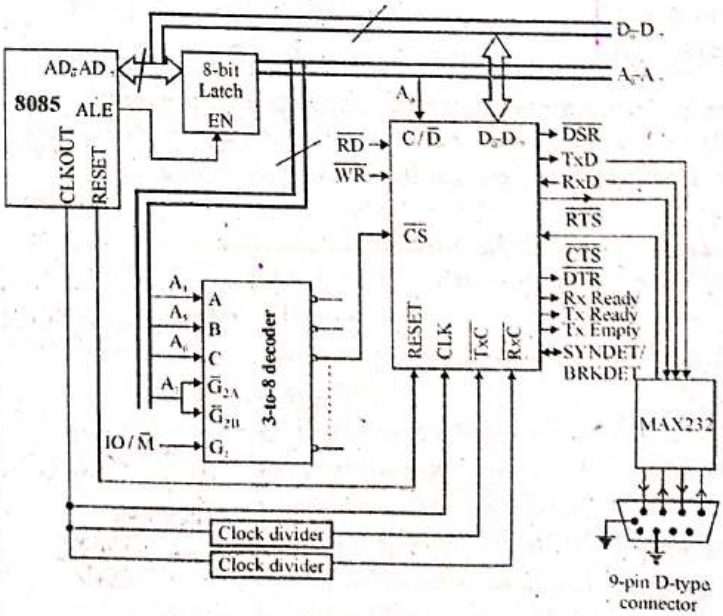


Fig.

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.16 ***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.

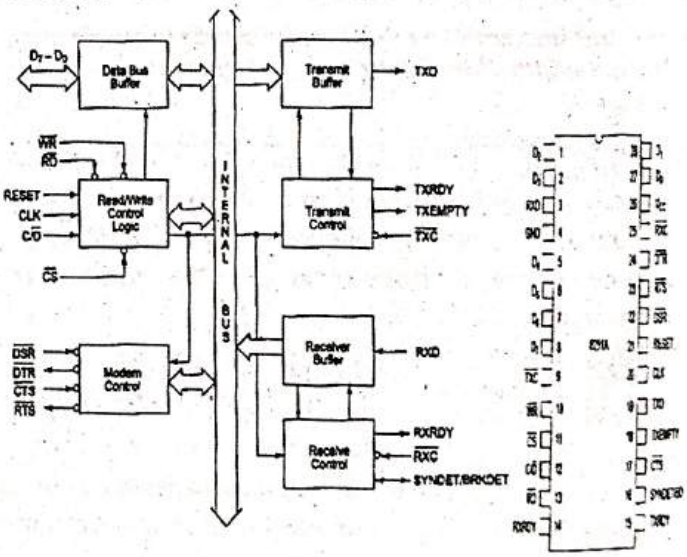


Fig. : Serial Communication Interface 8251

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 is a SYNC character. When CTS line goes low, the first character is serially transmitted out. Characters are

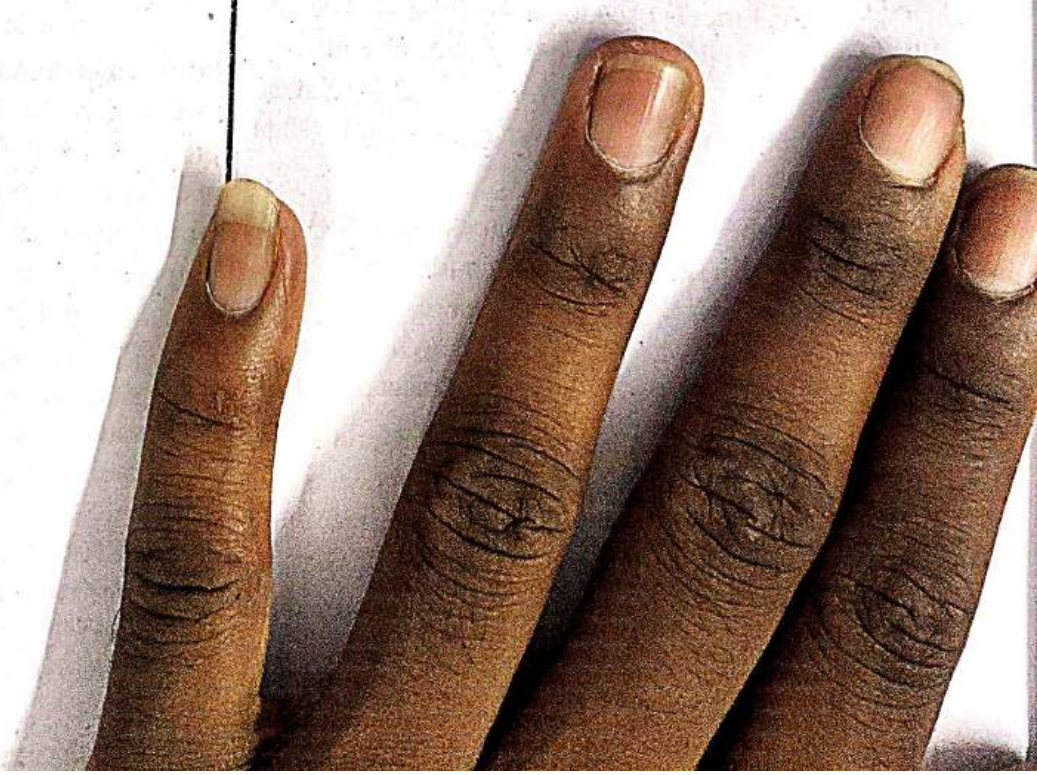
shifted out on the falling edge of TXC. Data is shifted out at the same rate as TXC, over TXD output line. If the CPU buffer becomes empty, the SYNC character or characters are inserted in the data stream over TXD output.

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.

❑❑❑