Arya Institute of Engineering and Technology, Jaipur
Guess Paper IV Semester (2022-2023)
Microprocessor and Interface (4CS3-04)

Unit 1:

Short Answers: (2 Marks Each)

Q1. What is microcontroller?

Microcontroller contains all essential components of a microcomputer such as CPU, RAM, ROM/EPROM, I/O lines etc. Some single chip microcontrollers contain devices to perform specific functions such as DMA channels, A/D converter, serial port, pulse width modulation, etc.

Q.2 What is a Microprocessor? What is difference between Microprocessor and CPU?

Ans.

A microprocessor functions as the CPU of a microcomputer, and includes the ALU, register arrays, and the control unit on one chip; it is manufactured using the LSI technology. On the other hand, the CPU is designed with various discrete boards. Functionally, both are similar; however, technology and processes used for designing is different.

Q3. What is the function of ALE in 8085 mp?

ALE (Address Enable Latch) is the control signal which is nothing but a positive going pulse generated when a new operation is started by microprocessor. So when pulse goes high means ALE=1, it makes address bus enable and when ALE=0, means low pulse makes data bus enable.

Q4. Specify the size of data bus, address bus, memory word and memory capacity of 8085 mp?
Ans:
Size of data bus: 8 bit
Address bus:16 bit
Memory word: 8 bit
Memory capacity: 64Kbyte

Q5. Why data bus is bi-directional?

Data bus is used to transfer data from one unit to another unit of the computer system. Microprocessor can read data from the memory or write data to the memory. So, the data bus is bidirectional.

Q6. Explain the PSW of 8085 mp?

PSW stands for Program Status word. it is a 16-bit word, a combination of contents of the 8-bit flag register and the contents of an 8-bit Accumulator register.

Q7. **List 16 bit register of 8085 mp?**

Ans: Stack Pointer, Program Counter, Register pair(BC,DE,HL)

Q8. What is the function of Accumulator?

Ans: An accumulator is primarily used as a register in a CPU to store intermediate logical or arithmetic data in multistep calculations. For such calculations, it functions as a temporary storage location.

Q9. Explain the program counter and stack pointer?

Stack Pointer: The stack pointer in the 8085 microprocessor is a 16-bit register that stores the address of the top of stack memory. The value of the stack pointer is decremented by 2 in the PUSH operation. It will increment by 2 in POP operation.

Program Counter: PC is a 16-bit register. It contains a memory address. PC contains that very memory address from where the next instruction is to be fetched for execution.

Q10. Explain the flag register?

Ans.

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

Flag Register Bit Format

Descriptive Answers: (5 Marks)

Q1. Draw the Architecture diagram of 8085 microprocessor explain function of various registers.

The architecture of the 8085 microprocessor consists of several key components, including the accumulator, registers, program counter, stack pointer, instruction register, flags register, data bus, address bus, and control bus.

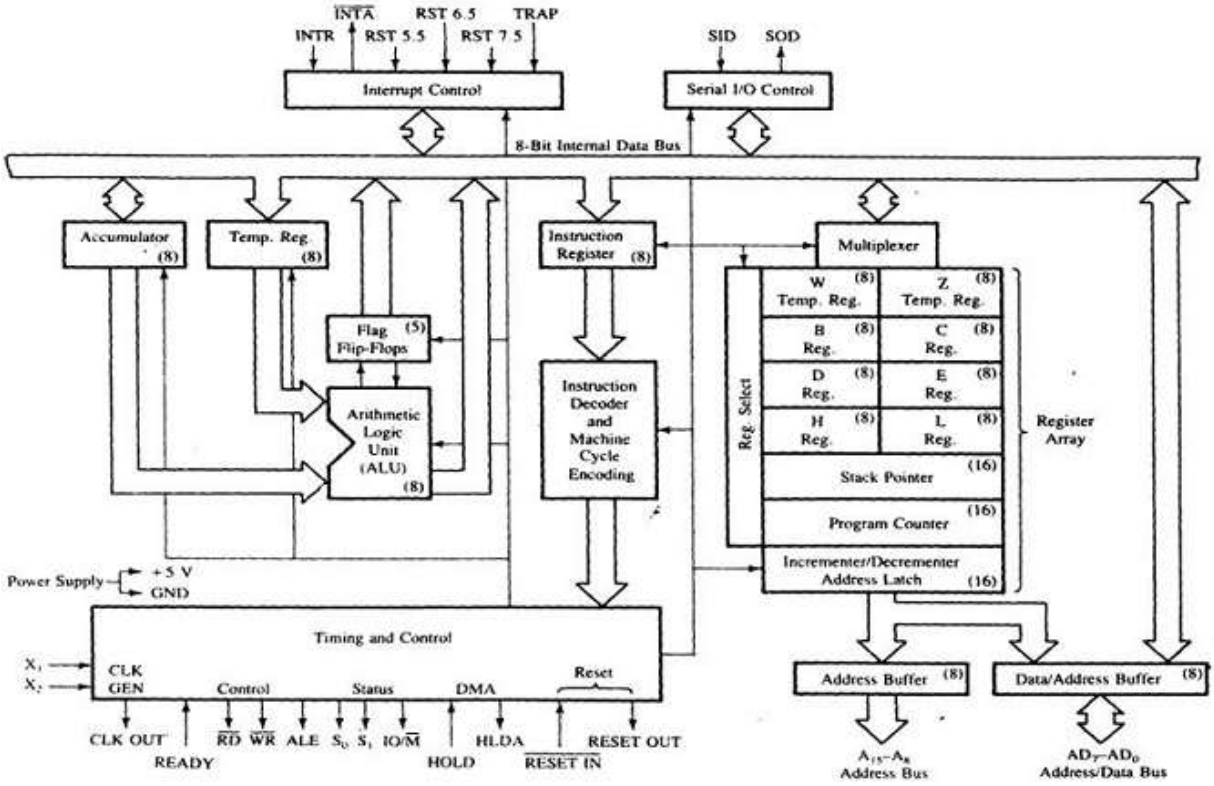
The accumulator is an 8-bit register that is used to store arithmetic and logical results. It is the most commonly used register in the 8085 microprocessor and is used to perform arithmetic and logical operations such as addition, subtraction, and bitwise operations.

The 8085 microprocessor has six general-purpose registers, including B, C, D, E, H, and L, which can be combined to form 16-bit register pairs. The B and C registers can be combined to form the BC register pair, the D and E registers can be combined to form the DE register pair, and the H and L registers can be combined to form the HL register pair. These register pairs are commonly used to store memory addresses and other data.

The program counter is a 16-bit register that contains the memory address of the next instruction to be executed. The program counter is incremented after each instruction is executed, which allows the microprocessor to execute instructions in sequence.

The stack pointer is a 16-bit register that is used to manage the stack. The stack is a section of memory that is used to store data temporarily, such as subroutine addresses and other data. The stack pointer is used to keep track of the top of the stack.

The instruction register is an 8-bit register that contains the current instruction being executed. The instruction register is used by the microprocessor to decode and execute instructions.



Q2. Explain Bus concept and organization with the help of suitable diagram?

Ans.

3.1.1 Microprocessor-Initiated Operations and 8085 Bus Organization

The MPU performs primarily four operations:*

1. Memory Read: Reads data (or instructions) from 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 instructions).
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 (Figure 3.1). In Chapter 1, these buses are shown as one group, called the system bus.

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 (Step 1).

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 sim-

*Other operations are omitted here for clarity and discussed in the next chapter.

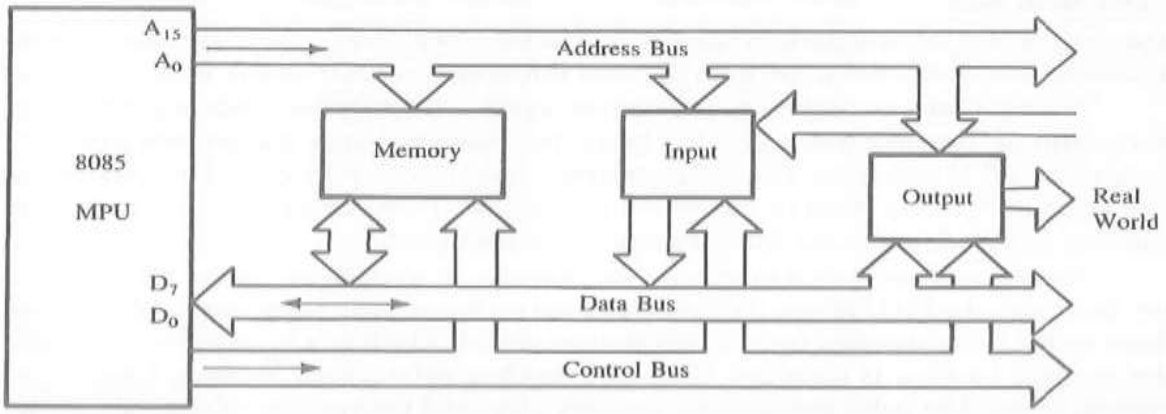


FIGURE 3.1
The 8085 Bus Structure

ilar 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 numbering scheme can identify only a hundred houses, from 00 to 99. On the other hand, the four-digit scheme can identify ten thousand houses, from 0000 to 9999. 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 explained in Chapter 1, 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.

Most 8-bit microprocessors have 16 address lines. This may explain why microcomputer systems based on 8-bit microprocessors have 64K memory. However, not every microcomputer system has 64K memory. In fact, most single-board microcomputers have less than 4K of memory, even if the MPU is capable of addressing 64K memory. The number of address lines is arbitrary; it is determined by the designer of a microprocessor based on such considerations as availability of pins and intended applications of the processor. For example, the Intel 8088 processor has 20 and the Pentium processor has 32 address lines.

DATA BUS

The data bus is a group of eight lines used for data flow (Figure 3.1).* These lines are **bidirectional**—data flow 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 (Step 2).

The eight data lines enable the MPU to manipulate 8-bit data ranging from 00 to FF ($2^8 = 256$ numbers). 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.

CONTROL BUS

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

The term **bus,** in relation to the control signals, is somewhat confusing. These are not groups of lines like address or data buses, but individual lines that provide a pulse to indicate an MPU operation. The MPU generates specific control signals for every operation (such as Memory Read or I/O Write) it performs. These signals are used to identify a device type with which the MPU intends to communicate.

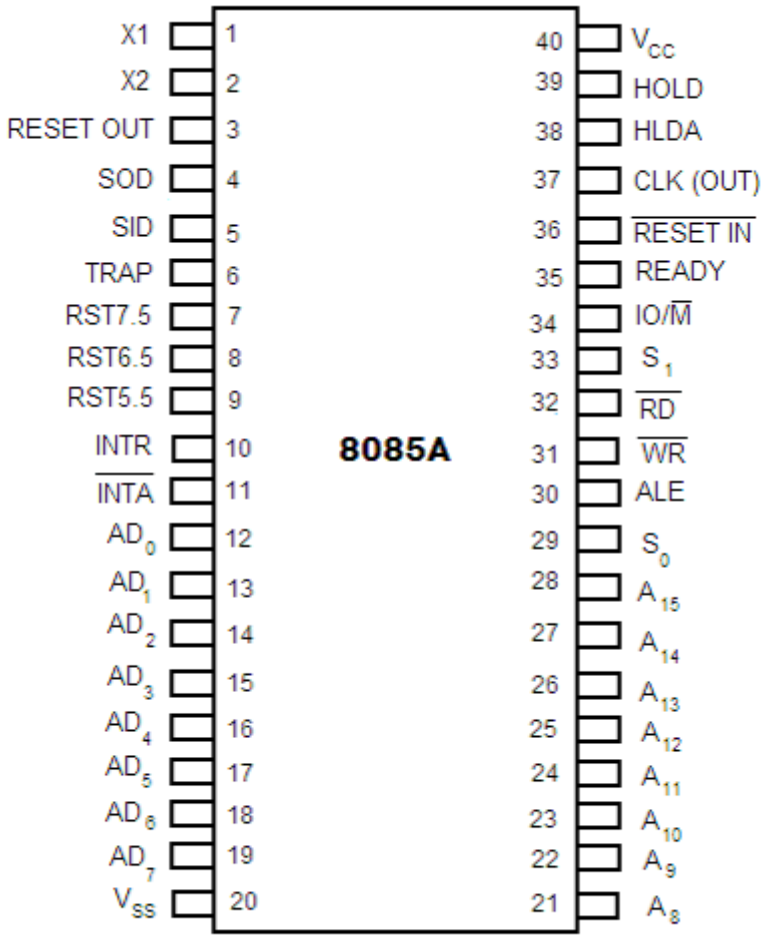
To communicate with a memory—for example, to read an instruction from a memory location—the MPU places the 16-bit address on the address bus (Figure 3.2). The address on the bus is decoded by an external logic circuit, which will be explained later, and the memory location is identified. The MPU sends a pulse called Memory Read as the control signal. The pulse activates the memory chip, and the contents of the memory location (8-bit data) are placed on the data bus and brought inside the microprocessor.

Q3. How will you demultiplex the address and data bus? How will you interface 1024KB RAM with 8085 microprocessor?

ANS : REFER NOTES

Descriptive Answers: (20 Marks)

Q1. Explain the pin diagram of 8085 microprocessor in detail?



1. **Address Bus and Data Bus:**
   The address bus is a group of sixteen lines i.e A0-A15. The address bus is unidirectional, i.e., bits flow in one direction from the microprocessor unit to the peripheral devices and uses the high order address bus.

2. Control and Status Signals:

- **ALE –** It is an Address Latch Enable signal. It goes high during first T state of a machine cycle and enables the lower 8-bits of the address, if its value is 1 otherwise data bus is activated.

- **IO/M' –** It is a status signal which determines whether the address is for input-output or memory. When it is high(1) the address on the address bus is for input-output devices. When it is low(0) the address on the address bus is for the memory.

- **SO, S1 –** These are status signals. They distinguish the various types of operations such as halt, reading, instruction fetching or writing.

IO/M'	S1	S0	Data Bus Status
0	1	1	Opcode fetch
0	1	0	Memory read
0	0	1	Memory write
1	1	0	I/O read
1	0	1	I/O write
1	1	1	Interrupt acknowledge
0	0	0	Halt

- **RD' –** It is a signal to control READ operation. When it is low the selected memory or input-output device is read.

- **WR' –** It is a signal to control WRITE operation. When it goes low the data on the data bus is written into the selected memory or I/O location.

- **READY –** It senses whether a peripheral is ready to transfer data or not. If READY is high(1) the peripheral is ready. If it is low(0) the microprocessor waits till it goes high. It is useful for interfacing low speed devices.

3. Power Supply and Clock Frequency:

- **Vcc –** +5v power supply

- **Vss –** Ground Reference

- **XI, X2 –** A crystal is connected at these two pins. The frequency is internally divided by two, therefore, to operate a system at 3MHZ the crystal should have frequency of 6MHZ.

- **CLK (OUT) –** This signal can be used as the system clock for other devices.

4. Interrupts and Peripheral Initiated Signals:
The 8085 has five interrupt signals that can be used to interrupt a program execution.
(i) INTR
(ii) RST 7.5
(iii) RST 6.5
(iv) RST 5.5
(v) TRAP

The microprocessor acknowledges Interrupt Request by INTA' signal. In addition to Interrupts, there are three externally initiated signals namely RESET, HOLD and READY. To respond to HOLD request, it has one signal called HLDA.

- **INTR –** It is an interrupt request signal.

- **INTA' –** It is an interrupt acknowledgement sent by the microprocessor after INTR is received.

5. Reset Signals:

- **RESET IN' –** When the signal on this pin is low(0), the program-counter is set to zero, the buses are tristated and the microprocessor unit is reset.

- **RESET OUT –** This signal indicates that the MPU is being reset. The signal can be used to reset other devices.

6. DMA Signals:

- **HOLD –** It indicates that another device is requesting the use of the address and data bus. Having received HOLD request the microprocessor relinquishes the use of the buses as soon as the current machine cycle is completed. Internal processing may continue. After the removal of the HOLD signal the processor regains the bus.

- **HLDA –** It is a signal which indicates that the hold request has been received after the removal of a HOLD request, the HLDA goes low.

7. Serial I/O Ports:
Serial transmission in 8085 is implemented by the two signals,

- **SID and SOD –** SID is a data line for serial input where as SOD is a data line for serial output.

Advantages of the 8085 microprocessor pin diagram:

- The pin diagram is easy to understand and remember because of its logical and systematic arrangement.
- It has a simple structure with fewer pins compared to other microprocessors, making it easy to design and implement in electronic circuits.
- It has a dedicated pin for interrupt handling, which makes it easy to interface with external devices that require interrupt-driven communication.

Disadvantages of the 8085 microprocessor pin diagram:

- It has limited addressing capability due to its 8-bit architecture, which can limit its use in applications that require large amounts of memory.
- It has a limited number of pins, which can be a constraint in designing complex systems that require more input/output devices or peripherals.
- It has separate pins for input/output and memory access, which can make it more difficult to design memory-mapped input/output circuits.

Q2. Explain the memory interfacing of 256 bytes? Explain how the range can be changed by modifying the hardware?

Ans.

Figure 3.10(a) shows a memory chip with 256 registers with eight I/O lines; the memory size of the chip is expressed as 256 × 8. It has eight address lines (A_7–A_0), one Chip Select signal ($\overline{CS}$) (active low), and two control signals Read ($\overline{RD}$) and Write ($\overline{WR}$). The eight address lines (A_7–A_0) of the microprocessor are required to identify 256 memory registers. The remaining eight lines (A_{15}–A_8) are connected to the Chip Select ($\overline{CS}$) line through inverters and the NAND gate. The memory chip is enabled or selected when $\overline{CS}$ goes low. Therefore, to select the chip, the address lines A_{15}–A_8 should be at logic 0, which will cause the output of the NAND gate to go low. No other logic levels on the lines A_{15}–A_8 can select the chip. Once the chip is selected (enabled), the remaining address lines A_7–A_0 can assume any combination from 00H to FFH and identify any of the 256 memory registers through the decoder. Therefore, the memory addresses of the chip in Figure 3.10(a) will range from 0000H to 00FFH, as shown below.

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	= 0000H
Chip Enable or Chip Select								Register Select								
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	= 00FFH

The address lines A_{15}–A_8, which are used to select the chip, must have fixed logic levels, and these lines are called high-order address lines. The address lines A_7–A_0, which

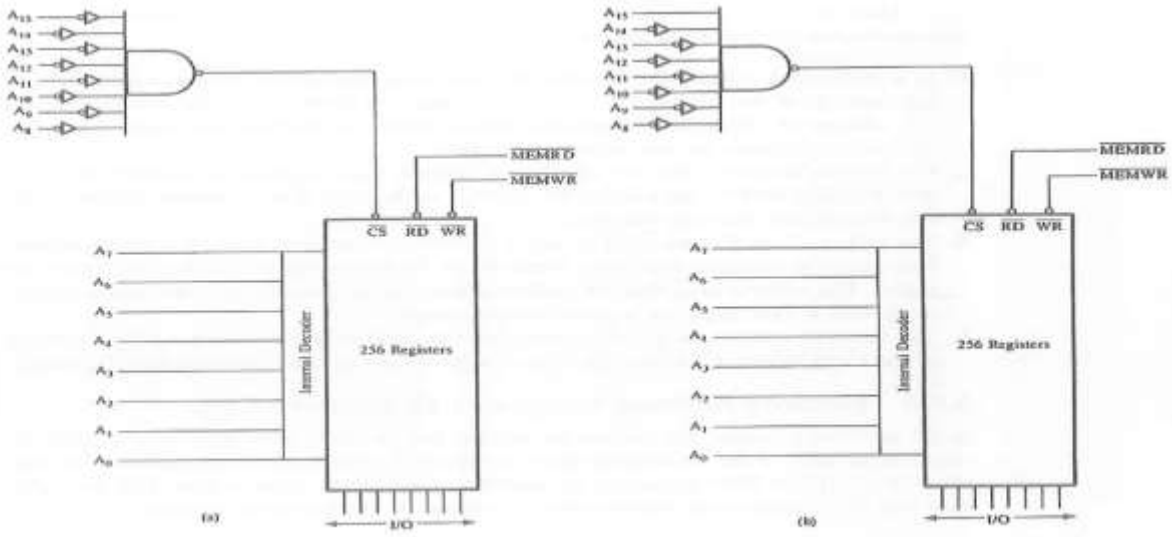


FIGURE 3.10
Memory Maps: 256 Bytes of Memory

are used to select a register, are called low-order address lines, and they can be assigned logic levels from all 0s to all 1s and any in-between combination. For example, when the address lines A_7–A_0 are all 0s, the register number 0 is selected, and when they are all 1s, the register number 255 (FFH) is selected. The Chip Select addresses are determined by the hardware (the inverters and NAND gate); therefore, the memory addresses of the chip can be changed by modifying the hardware. For example, if the inverter on line A_{15} is removed, as shown in Figure 3.10(b), the address required on A_{15}–A_8 to enable the chip will be as follows:

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	
1	0	0	0	0	0	0	0	= 80H

The memory address range in Figure 3.10(b) will be 8000H to 80FFH.

The memory chips in Figures 3.10(a) and (b) are the same chips. However, by changing the hardware of the Chip Select logic, the location of the memory in the map can be changed, and memory can be assigned addresses in various locations over the entire map of 0000 to FFFFH.

Unit 2:
Short Answers :(2 Marks Each)

Q.1 Write a program to subtract A1H and 11H using SBI instruction?

Ans :

```
MVI A,A1H
MVI B,11H
SBI B
STA C000H
HLT
```

Q.2 Find out the Machine cycle and T states for following Instruction:-

ADD M ii) MVI M,45H iii) LXI H, C700H.

Ans. Instruction

a) ADD M 2(Machine Cycle) 7(T-state)

b) **MVI M,45H 3(Machine Cycle) 10(T-state)**
c) **LXI H, C700H 3(Machine Cycle) 10(T-state)**

Q3. Explain difference between conditional and unconditional instruction?

Ans. Conditional Jump

If some specified condition is satisfied in conditional jump, the control flow is transferred to a target instruction. There are numerous conditional jump instructions depending upon the condition and data. Conditional Jump Instructions are an important aspect of the decision making process in programming. These Instruction test for a certain condition (e.g., Zero or Carry Flag) and alter the program sequence when the condition is met. The conditional Jump instruction check the flag conditions and make decisions to change or not to change the sequence of the program.In conditional jump instruction, status conditions at the time of jump instruction execution decide whether or not the jump will occur.

Unconditional jump

In unconditional jump instruction, as the instruction is executed, the jump always ready to takes pace to change the execution sequence. This is performed by the JMP instruction. Conditional execution often involves a transfer of control to the address of an instruction that does not follow the currently executing instruction. Transfer of control may be forward, to execute a new set of instructions or backward, to re-execute the same steps. The assembler determines the smallest encoding possible for the direct unconditional jump. It is the unconditional jump that executes regardless of the state of all flags. So we write an unconditional jump as the very first instruction of our program and jump to the next instruction that follows our data declarations.

Q4. Explain M-Cycle, T-State and Instruction Cycle of 8085?

Ans: Machine cycle:

The basic microprocessor operation such as reading a byte from I/O port or writing a byte to memory is called as machine cycle.

T-state:

One complete cycle of clock is called as T-state. A T-state is measured from the falling edge of one clock pulse to the falling edge of the next clock pulse.

Instruction Cycle:

The time a microprocessor needs to fetch and execute one entire instruction is known as an instruction cycle.

Q5. Explain the instruction format of 8085 with example?

Instruction Format: The Instruction Format of 8085 set consists of one, two and three byte instructions. The first byte is always the opcode; in two-byte instructions the second byte is usually data; in three byte instructions the last two bytes present address or 16-bit data.

1. 1-byte Instruction: The opcode and the operand of an instruction are represented in the one byte. Eg: MOV A,B

2. 2-byte Instruction: The first 8 bits indicates the opcode and the next 8-bits indicates operand. Eg. MVI A,05H

3. 3-byte Instruction: The first 8 bits indicates the opcode and the next 2-bytes indicates the 16 bit address. Eg. LXI H,2050H

Descriptive Answers: (5 Marks)

Q1. Explain the operational difference between the following pair of instruction?

i) SPHL and PCHL
ii) CALL and JMP

Ans. I) SPHL and PCHL

PCHL: Load Program Counter with HL Contents

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
PCHL	None	1	1	6	E9

Description The contents of registers H and L are copied into the program counter. The contents of H are placed as a high-order byte and of L as a low-order byte.

Flags No flags are affected.

Comments: This instruction is equivalent to a 1-byte unconditional Jump instruction. A program sequence can be changed to any location by simply loading the H and L registers with the appropriate address and by using this instruction.

SPHL: Copy H and L Registers to the Stack Pointer

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
SPHL	None	1	1	6 (8085) 5 (8080)	F9

Description 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.

Flags No flags are affected.

iii) CALL and JMP

CALL: Unconditional Subroutine Call

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
CALL	16-bit address	3	5	18	CD

Description The program sequence is transferred to the address specified by the operand. Before the transfer, the address of the next instruction to CALL (the contents of the program counter) is pushed on the stack. The sequence of events is described in the example below.

Flags No flags are affected.

Conditional Call to Subroutine

Op Code	Description	Flag Status	Hex Code
CC	Call on Carry	CY = 1	DC
CNC	Call with No Carry	CY = 0	D4
CP	Call on positive	S = 0	F4
CM	Call on minus	S = 1	FC
CPE	Call on Parity Even	P = 1	EC
CPO	Call on Parity Odd	P = 0	E4
CZ	Call on Zero	Z = 1	CC
CNZ	Call on No Zero	Z = 0	C4

Operand—16-Bit Address

M-Cycles T-States

2/9 (if condition is not true)

5/18 (if condition is true)

Note: If condition is not true it continues the sequence, and thus requires fewer T-states.

If condition is true it calls the subroutine, thus requires more T-states.

JMP: Jump Unconditionally

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
JMP	16-bit	3	3	10	C3

Description The program sequence is transferred to the memory location specified by the 16-bit address. This is a 3-byte instruction; the second byte specifies the low-order byte and the third byte specifies the high-order byte.

Jump Conditionally

Operand: 16-bit address

Op Code	Description	Flag Status	Hex Code	M-Cycles/T-States
JC	Jump on Carry	CY = 1	DA	2M/7T (if condition
JNC	Jump on No Carry	CY = 0	D2	is not true)
JP	Jump on positive	S = 0	F2	3M/10T (if condition
JM	Jump on minus	S = 1	FA	is true)
JPE	Jump on Parity Even	P = 1	EA	
JPO	Jump on Parity Odd	P = 0	E2	
JZ	Jump on Zero	Z = 1	CA	
JNZ	Jump on No Zero	Z = 0	C2	

Q2. Explain Addressing modes of 8085 microprocessor with example?

These are the instructions used to transfer the data from one register to another register, from the memory to the register, and from the register to the memory without any alteration in the content. Addressing modes in 8085 is classified into 5 groups −

Immediate addressing mode:

In this mode, the 8/16-bit data is specified in the instruction itself as one of its operand. **For example:** MVI A, 20F: means 20F is copied into register A.

Register addressing mode

In this mode, the data is copied from one register to another. **For example:** MOV A, B: means data in register B is copied to register A.

Direct addressing mode

In this mode, the data is directly copied from the given address to the register. **For example:** LDA 5000H: means the data at address 5000H is copied to register A.

Indirect addressing mode

In this mode, the data is transferred from one register to another by using the address pointed by the register. **For example:** MOV M, B: means data is transferred from the memory address pointed by the register to the register M.

Implied addressing mode

This mode doesn't require any operand; the data is specified by the opcode itself. **For example:** CMP.

Q3. Write a program to transfer block of 09 data from one memory location to another in same order?

Ans:

```
LXI H,3000H
LXI D,3500 H
MVI C,09H
L1  MOV A,M
    STAX D
    INX H
    INX D
    DCR C
    JNZ L1
    HLT
```

Descriptive Answers: (20 Marks)

Q1. Define Machine Cycle? What are the different types of machine cycle used for calculation of T-States? Explain Opcode Fetch machine Cycle with Timing diagram?

Ans.

The time required by the microprocessor to complete an operation of accessing memory or input/output devices is called *machine cycle*. One time period of frequency of microprocessor is called *t-state*. A t-state is measured from the falling edge of one clock pulse to the falling edge of the next clock pulse Machine Cycle in 8085:

The seven Machine Cycle in 8085 are :

1. **Opcode Fetch**
2. **Memory Read**
3. **Memory Write**
4. **I/O Read**
5. **I/O Write**
6. **Interrupt Acknowledge**
7. **Bus Idle**

1. Opcode Fetch Machine Cycle 8085:

The first Machine Cycle in 8085 of every instruction is opcode fetch cycle in which the 8085 finds the nature of the instruction to be executed. In this Machine Cycle in 8085, processor places the contents of the Program Counter on the address lines, and through the read process, reads the opcode of the instruction.

2.Memory Read Cycle:

The 8085 executes the memory read cycle to read the contents of R/W memory or ROM. The length of this machine cycle is 3-T states ($T_1 – T_3$). In this Machine Cycle in 8085, processor places the address on the address lines from the stack pointer, general purpose register pair or program counter, and through the read process, reads the data from the addressed memory location.

3.Memory Write Cycle:

The 8085 executes the memory write cycle to store the data into data memory or stack memory. The length of this machine cycle is 3T states. (T_1 — T_3). In this Machine Cycle in 8085, processor places the address on the address lines from the stack pointer or general purpose register pair and through the write process, stores the data into the addressed memory location.

4, 5. I/O Read and I/O Write cycles:

The I/O read and I/O write machine cycles are similar to the memory read and memory write machine cycles, respectively, except that the 10/M signal is high for I/O read and I/O write machine cycles. High IO/M signal indicates that it is an I/O operation.

6. Interrupt Acknowledge Cycle of 8085:

In response to INTR signal, 8085 executes interrupt acknowledge machine cycle to read an instruction from the external device. Theoretically, the external device can place any instruction on the data bus in response to INTA. However, only RST and CALL, save the PC contents (return address) before transferring control to the interrupt service routine. The next sections explain Interrupt Acknowledge Cycle of 8085 for RST and CALL instruction.

Q2. (i) Draw the timing diagram of MVIA, 32H?

(ii) Write a program to implement 16 bit counter?

Ans.

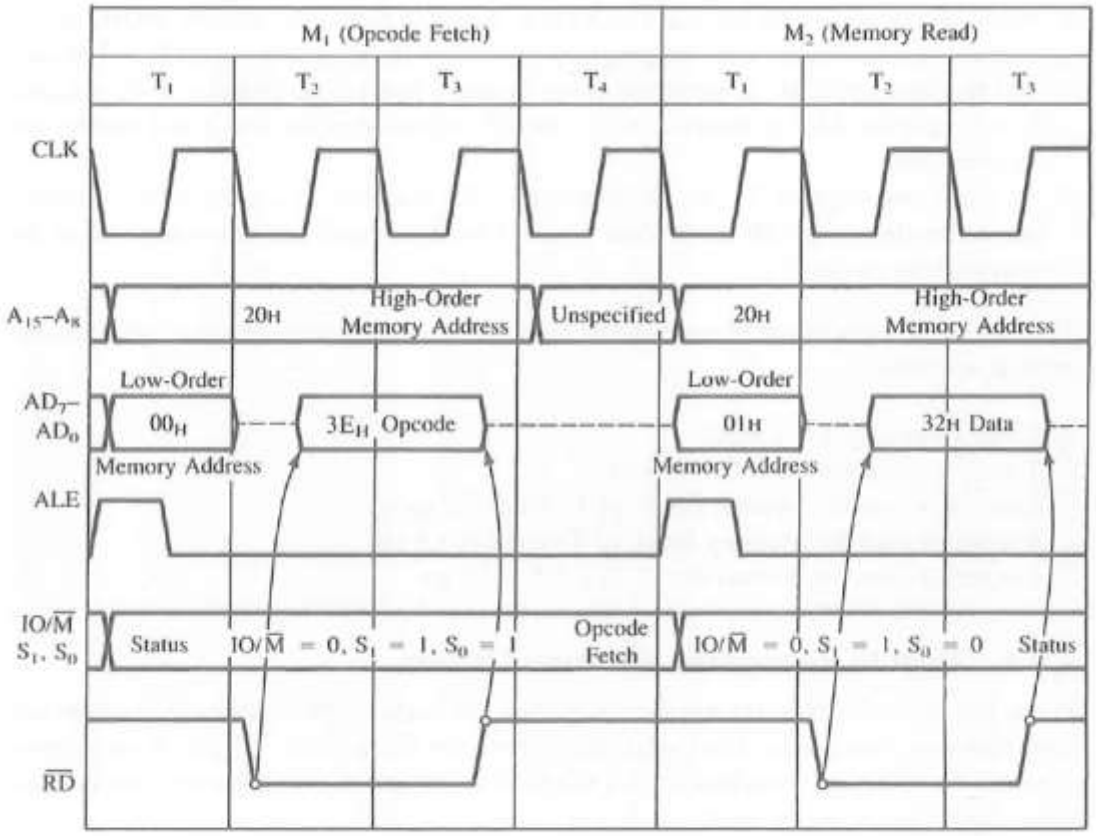


FIGURE 4.10
8085 Timing for Execution of the Instruction MVI A,32H

(ii) 16-bit counter Program:

```
        LXI H,FFFFH
L1:     DCX H
        MOV A,B
        ORA L
        JNZ L1
        HLT
```

Unit 3

Short Answers:(2 Marks Each)

Q.1 What is the difference between Macros and Subroutine? Explain with examples?

Ans.

Macro	Subroutine
A macro is a set of instructions that are stored in a computer program and can be executed with a single command.	A subroutine is a smaller section of a larger program that performs a specific task.
It is used to perform a series of actions automatically, saving time and effort for the programmer.	It can be called upon by the main program whenever it is needed, making it easier to organize and manage the code.
Macros are executed directly by the computer.	Subroutines must be called upon by the main program.
Macros are typically used for simple tasks.	Subroutines are used for more complex tasks.
Macros are defined within the main program.	Subroutines are defined in a separate location.
Macros are expanded at compile time.	Subroutines are executed at runtime.

Q.2 Explain PUSH and PULL instruction?

Ans. PUSH B (1 Byte Instruction)
– Decrement SP
– Copy the contents of register B to the memory location pointed to by SP
– Decrement SP
– Copy the contents of register C to the memory location pointed to by SP

POP D (1 Byte Instruction)
– Copy the contents of the memory location pointed to by the SP to register E
– Increment SP
– Copy the contents of the memory location pointed to by the SP to register D
– Increment SP

Q3. Write 8085 program that enables RST 7.5 and RST 6.5?
Ans:

D7	D6	D5	D4	D3	D2	D1	D0
SOD	SDE	X	R7.5	MSE	M7.5	M6.5	M5.5
0	0	0	0	1	1	1	0

=0EH
PROGRAM:
7000H EI

7001H MVI A,0EH
7002H 0EH
7003H SIM

Q4. Show which interrupt is masked if following instructions are executed:

(i) MVI A,10H (ii) SIM

Ans:

D7	D6	D5	D4	D3	D2	D1	D0
SOD	SDE	X	R7.5	MSE	M7.5	M6.5	M5.5
0	0	0	1	0	0	0	0

(i) MSE(D3) is reset, hence M5.5, M6.5, M7.5 are ignored and remain unchanged.

(ii) RST 7.5 is set which will reset RST 7.5 interrupt flip flop.

(iii) SDE is reset, hence there will be no serial communication and SOD will be ignored.

Q5. Write a program to implement 8-bit counter?

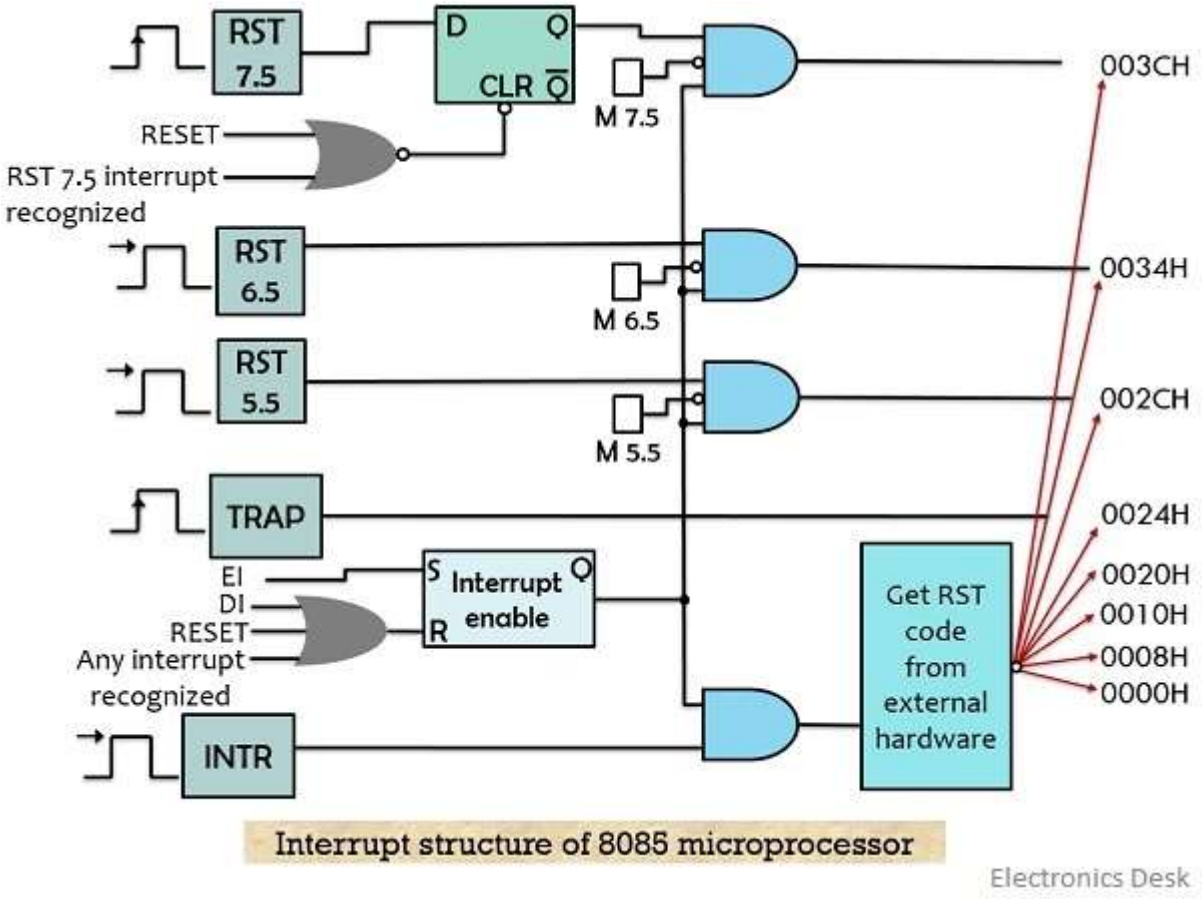
Ans

```
        MVI C,FFH
L1:     DCR C
        JNZ L1
```

Descriptive Answers: (5 / 20 Marks)

Q.1 Draw the diagram of interrupt structure of 8085 MPU?

Ans.



Q.2 Draw block diagram of 8259 Programmable Interrupt Control and explain function of various block?

Ans.

The 8259A is a programmable interrupt controller designed to work with Intel microprocessors 8085, 8086, and 8088. The 8259A interrupt controller can

1. manage eight interrupts according to the instructions written into its control registers. 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 (to be explained later).
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.

15.5.1 Block Diagram of the 8259A

Figure 15.29 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.

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 $\overline{\text{INTA}}$ (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 $\overline{\text{INTA}}$ is the Interrupt Acknowledge signal from the MPU.

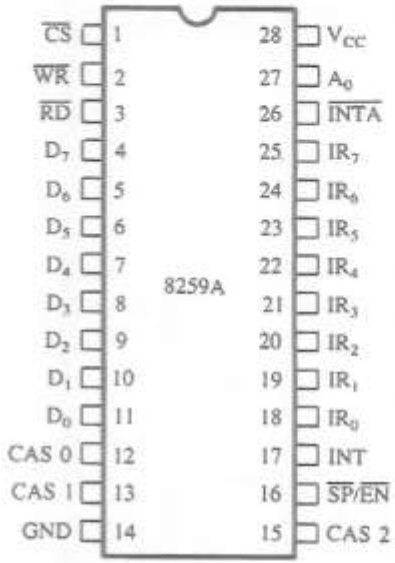
INTERRUPT REGISTERS AND PRIORITY RESOLVER

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 registers and determines whether INT should be sent to the MPU.

CASCADE BUFFER/COMPARATOR

This block is used to expand the number of interrupt levels by cascading two or more 8259As. To simplify the discussion, this block will not be mentioned again.

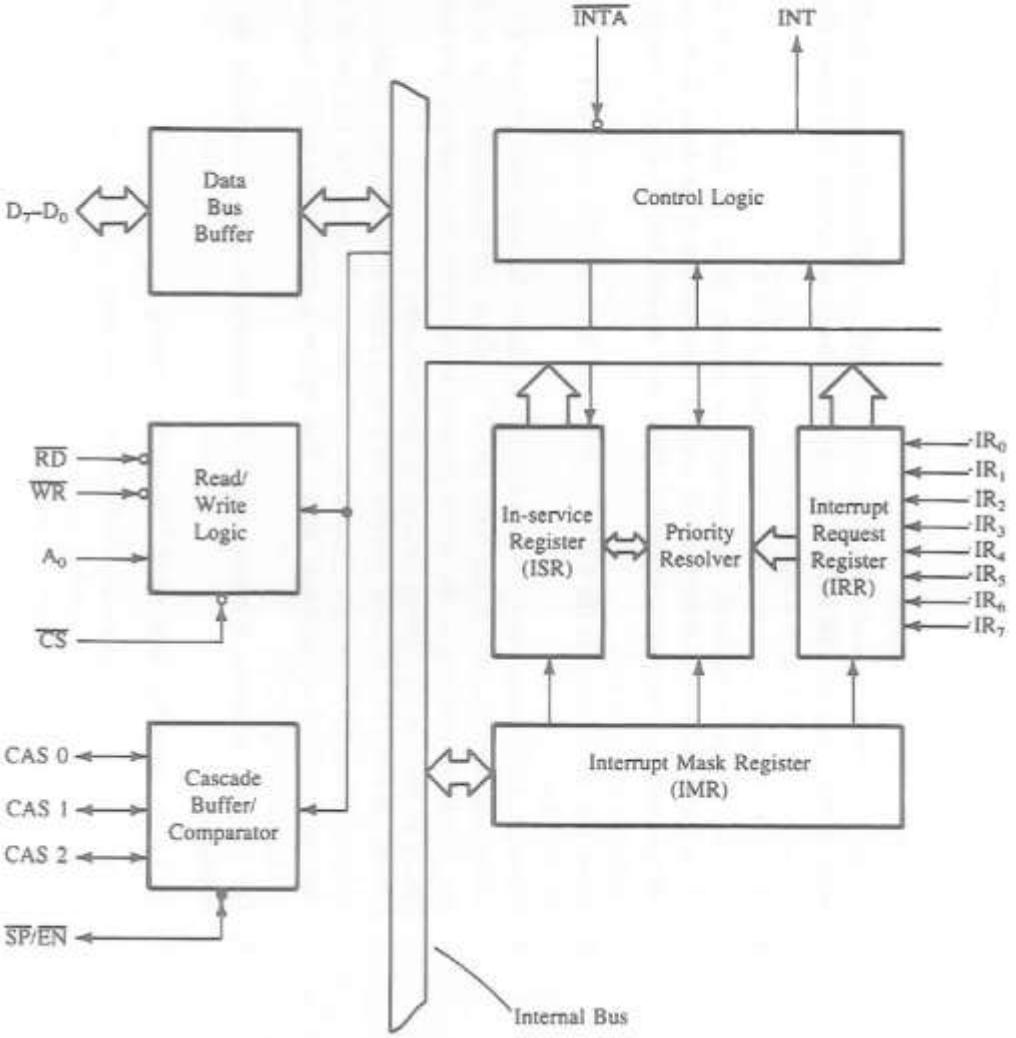
Pin Configuration



Pin Names

D_7–D_0	Data Bus (Bidirectional)
$\overline{RD}$	Read Input
$\overline{WR}$	Write Input
A_0	Command Select Address
$\overline{CS}$	Chip Select
CAS_2–CAS_0	Cascade Lines
$\overline{SP/EN}$	Slave Program/Enable Buffer
INT	Interrupt Output
$\overline{INTA}$	Interrupt Acknowledge Input
IR_0–IR_7	Interrupt Request Inputs

Block Diagram



1. The IRR stores the requests.
2. The priority resolver checks three registers: the IRR for interrupt requests, the IMR for masking bits, and the ISR for the interrupt request being served. It resolves the priority and sets the INT high when appropriate.
3. The MPU acknowledges the interrupt by sending $\overline{\text{INTA}}$.
4. After the $\overline{\text{INTA}}$ is received, the appropriate priority bit in the ISR is set to indicate which interrupt level is being served, and the corresponding bit in the IRR is reset to indicate that the request is accepted. Then, the opcode for the CALL instruction is placed on the data bus.
5. When the MPU decodes the CALL instruction, it places two more $\overline{\text{INTA}}$ signals on the data bus.
6. When the 8259A receives the second $\overline{\text{INTA}}$, it places the low-order byte of the CALL address on the data bus. At the third $\overline{\text{INTA}}$, it places the high-order byte on the data bus. The CALL address is the vector memory location for the interrupt; this address is placed in the control register during the initialization.
7. During the third $\overline{\text{INTA}}$ pulse, the ISR bit is reset either automatically (Automatic-End-of-Interrupt—AEOI) or by a command word that must be issued at the end of the service routine (End-of-Interrupt—EOI). This option is determined by the initialization command word (ICW).
8. The program sequence is transferred to the memory location specified by the CALL instruction.

Q.3 Write a Program that will display FF and 11 repeatedly on the seven segment display. Write a 'delay' subroutine and Call it as necessary.

Ans. C000: LXISP FFFF
C003: MVIA FF
C005: OUT 00
C007: CALL 14 20
C00A: MVIA 11
C00C: OUT 00
C00E: CALL 14 20
C011: JMP 03 C0
DELAY: C014: MVIB FF
C016: MVIC FF
C018: DCR C
C019: JNZ 18 C0
C01C: DCR B
C01D: JNZ 16 C0
C020: RET

Q.4 Examine the delay for the given program?

Label	Opcode	Operand
	MVI	C,FFH
LOOP:	DCR	C
	JNZ	LOOP

Ans.

The last column in Figure 8.2 shows the T-states (clock periods) required by the 8085 microprocessor to execute each instruction. (See Appendix F for the list of the 8085 instructions and their T-states.) The instruction MVI requires seven clock periods. An 8085-based microcomputer with 2 MHz clock frequency will execute the instruction MVI in 3.5 μs as follows:

Clock frequency of the system f = 2 MHz
Clock period $T = 1/f = 1/2 \times 10^{-6} = 0.5$ μs
Time to execute MVI = 7 T-states × 0.5
= 3.5 μs

However, if the clock frequency of the system is 1 MHz, the microprocessor will require 7 μs to execute the same instruction. To calculate the time delay in a loop, we must account for the T-states required for each instruction and for the number of times the instructions are executed in the loop.

In Figure 8.2, register C is loaded with the count FFH (255_{10}) by the instruction MVI, which is executed once and takes seven T-states. The next two instructions, DCR and JNZ, form a loop with a total of 14 (4 + 10) T-states. The loop is repeated 255 times until register C = 0.

The time delay in the loop T_L with 2 MHz clock frequency is calculated as

$$TL = (T \times \text{Loop T-states} \times N10)$$

where T_L = Time delay in the loop
T = System clock period
N_{10} = Equivalent decimal number of the hexadecimal count loaded in the delay register
$T_L = (0.5 \times 10^{-6} \times 14 \times 255)$
= 1785 μs
≈ 1.8 ms

Unit 4:

Short Answers:(2 Marks Each)

Q1. What is difference between 8253 and 8254?

Ans.

- ☐ the 8254 can operate with higher clock frequency range (DC to 8 MHz and 10 MHz for 8254-2), and the 8253 can operate with clock frequency from DC to 2 MHz.
- ☐ the 8254 includes a Status Read-Back Command that can latch the count and the status of the counters.

Q2. Compare Memory mapped I/O and peripheral I/O?

Ans. Difference b/w Memory Mapped I/O and Peripheral Mapped I/O are given below.

Memory Mapped I/O:-

- 16-bit device address
- Data transfer between any general-purpose register and I/O port.
- The memory map (64K) is shared between I/O device and system memory.
- More hardware is required to decode 16-bit address
- Arithmetic or logic operation can be directly performed with I/O data

Peripheral Mapped I/O:-

- 8-bit device address
- Data is transfer only between accumulator and I.O port
- The I/O map is independent of the memory map; 256 input device and 256. output device can be connected
- Less hardware is required to decode 8-bit address
- Arithmetic or logical operation cannot be directly performed with I/O data

Q3. Explain control word format of 8255A for I/O Mode?

Ans

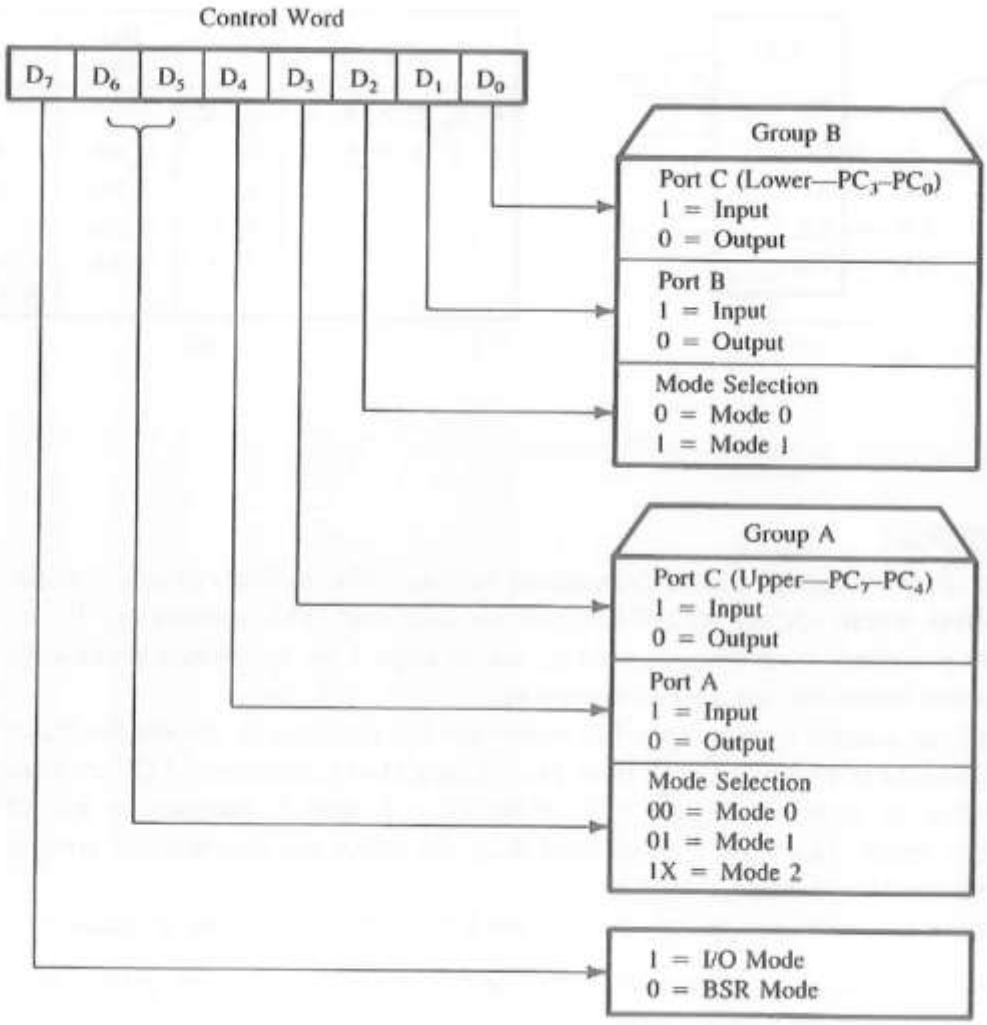


FIGURE 15.4
8255A Control Word Format for I/O Mode

Descriptive Answers: (5 /20 Marks)

Q1. Explain various programming modes of 8279 keyboard and display controller.

Ans.

14.3.2 Programming the 8279

The 8279 is a complex device that can accept eight different commands to perform various functions.*

The initialization commands can specify

1. left or right entry and key rollover.
2. clock frequency prescaler.
3. starting address and incrementing mode of the FIFO RAM.
4. RAM address to read and write data and incrementing mode.
5. blanking format.

To illustrate important command words, the next section will illustrate the keyboard/display circuit shown in Figure 14.18.

14.3.3 Illustration: Keyboard/Display Interfacing Using the 8279

Figure 14.18 shows the keyboard/display circuit of a single-board system.

1. Explain the functions of various components in the circuit.
2. Explain the decoding logic, and identify the port addresses of the 8279 registers.
3. Explain the initialization instructions given later.

CIRCUIT DESCRIPTION

Figure 14.18 shows the following components:

- ☐ The 8279 Programmable Keyboard/Display Interface
- ☐ A matrix keyboard with 20 keys: one matrix of 16 keys (4 × 4) and the other with four keys (1 × 4)
- ☐ Six seven-segment LEDs in groups of two
- ☐ A 3-to-8 decoder to generate additional scan lines
- ☐ A 2981A, a current driver for anodes and transistors as current drivers for cathodes

Lines RL_0–RL_3 (Return Lines) of the 8279 are connected to the columns of the matrix keyboard, and the output lines (A_0–A_3 and B_0–B_3) are connected to drive the LED segments. The three scan lines are connected to the decoder, the 74HC138, to generate eight decoded signals. In this circuit, six output lines of the decoder are connected as digit drivers to turn on six seven-segment LEDs; two output lines are unused. The 8279 has four scan lines that can be decoded to generate 16 output lines to drive 16 displays. The data lines of the 8279 are connected to the data bus of the 8085, and the IRQ (Interrupt Request) is connected to the RST 5.5 of the system.

Four signals—$\overline{RD}$, $\overline{WR}$, CLK, and RESET OUT—are connected directly from the 8085. The system has a 3.072 MHz clock; when the 8279 is reset, the clock prescaler is

set to 31. This divides the clock frequency by 31 to provide the scan frequency of approximately 100 kHz. The RESET signal also sets the 8279 in the mode of 16-character display with two-key lockout keyboard.

After the initialization of the 8279, the respective codes are sent to the display RAM to display any characters. The 8279 takes over the task of displaying the characters by outputting the codes and digit strobes. To read the keyboard, the 8279 scans the columns; if a key closure is detected, it debounces the key. If a key closure is valid, it loads the key code into the FIFO, and the IRQ line goes high to interrupt the system.

DECODING LOGIC AND PORT ADDRESSES

The port addresses of the 8279 registers are determined by two signals: $\overline{CS}$ and A_0. The $\overline{CS}$ of the 8279 is connected to the Y_1 signal generated by the address decoding circuit in Figure 14.15 and A_0 of the 8279 is directly connected to the address line A_0 of the 8085 processor. For commands and status, A_0 should be high and for data transfer A_0 should be low. The addresses of the Data port and the Command/Status port are 40H and 41H, re-

Q. 2 Explain the control word of 8254 PPI with example. Also explain the operation mode of 8254 with diagram?

Ans.

The 8254 includes three identical 16-bit counters that can operate independently in any one of the six modes (to be described later). It is packaged in a 24-pin DIP and requires a single +5 V power supply. To operate a counter, a 16-bit count is loaded in its register and, on command, begins to decrement the count until it reaches 0. At the end of the count, it generates a pulse that can be used to interrupt the MPU. The counter can count either in binary or BCD. In addition, a count can be read by the MPU while the counter is decrementing.

Block Diagram of 8254

DATA BUS BUFFER

This tri-state, 8-bit, bidirectional buffer is connected to the data bus of the MPU.

CONTROL LOGIC

The control section has five signals: $\overline{RD}$ (Read), $\overline{WR}$ (Write), $\overline{CS}$ (Chip Select), and the address lines A_0 and A_1. In the peripheral I/O mode, the $\overline{RD}$ and $\overline{WR}$ signals are connected to $\overline{IOR}$ and $\overline{IOW}$, respectively. In memory-mapped I/O, these are connected to $\overline{MEMR}$ (Memory Read) and MEMW (Memory Write). Address lines A_0 and A_1 of the MPU are usually connected to lines A_0 and A_1 of the 8254, and $\overline{CS}$ is tied to a decoded address.

The control word register and counters are selected according to the signals on lines A_0 and A_1, as shown below:

A_1	A_0	Selection
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Control Register

Q3. Interface the keyboard with seven segment LED using 8255?

Ans.

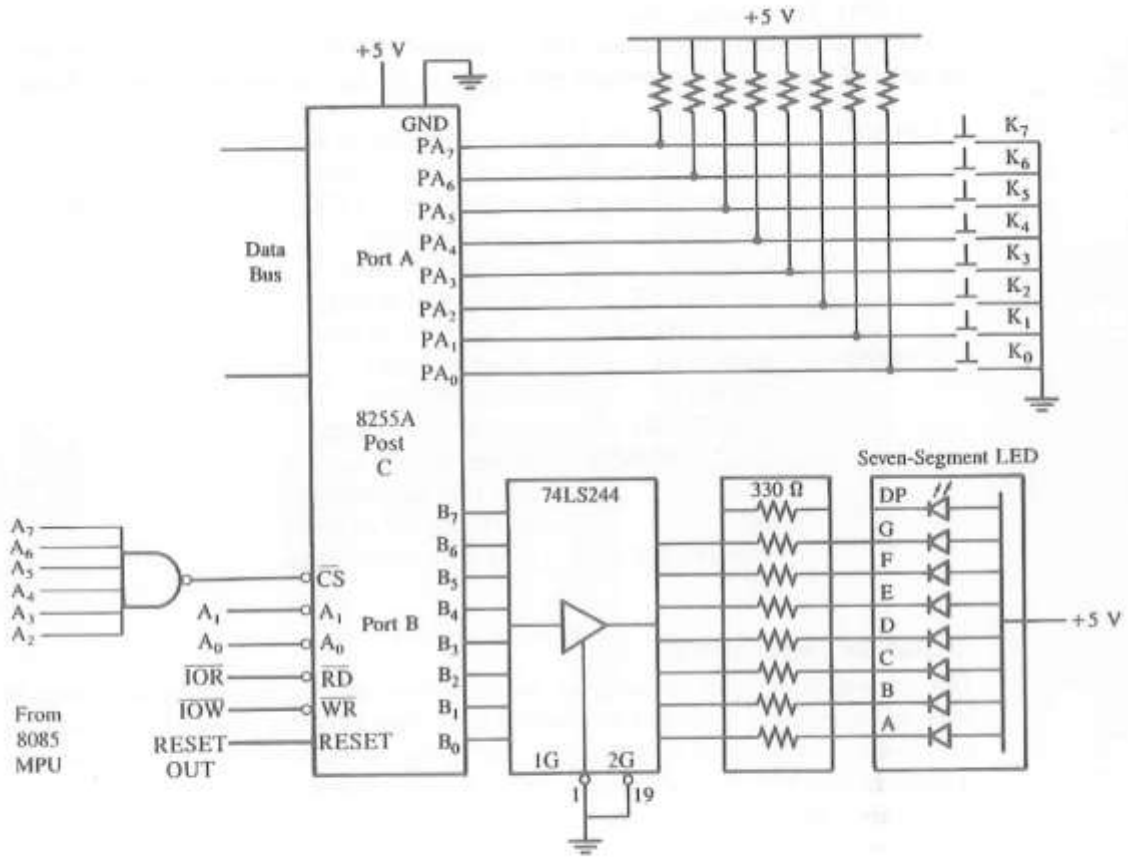
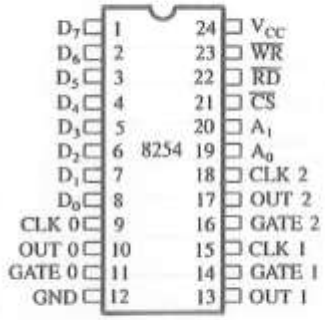


FIGURE 15.15
Interfacing a Keyboard and a Seven-Segment LED

Pin Configuration



Pin Names

D_7–D_0	Data Bus (8 Bit)
CLK N	Counter Clock Inputs
GATE N	Counter Gate Inputs
OUT N	Counter Outputs
$\overline{RD}$	Read Counter
$\overline{WR}$	Write Command or Data
$\overline{CS}$	Chip Select
A_0–A_1	Counter Select
V_{CC}	+5 Volts
GND	Ground

Block Diagram

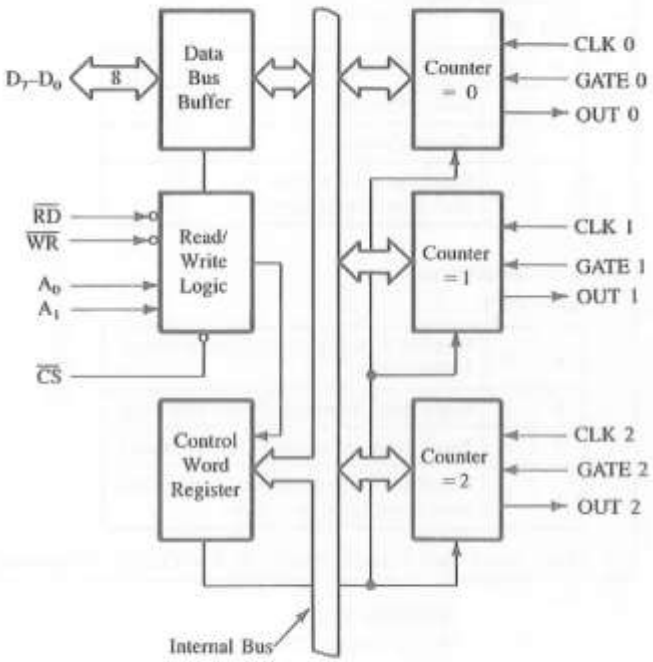


FIGURE 15.23
8254 Block Diagram

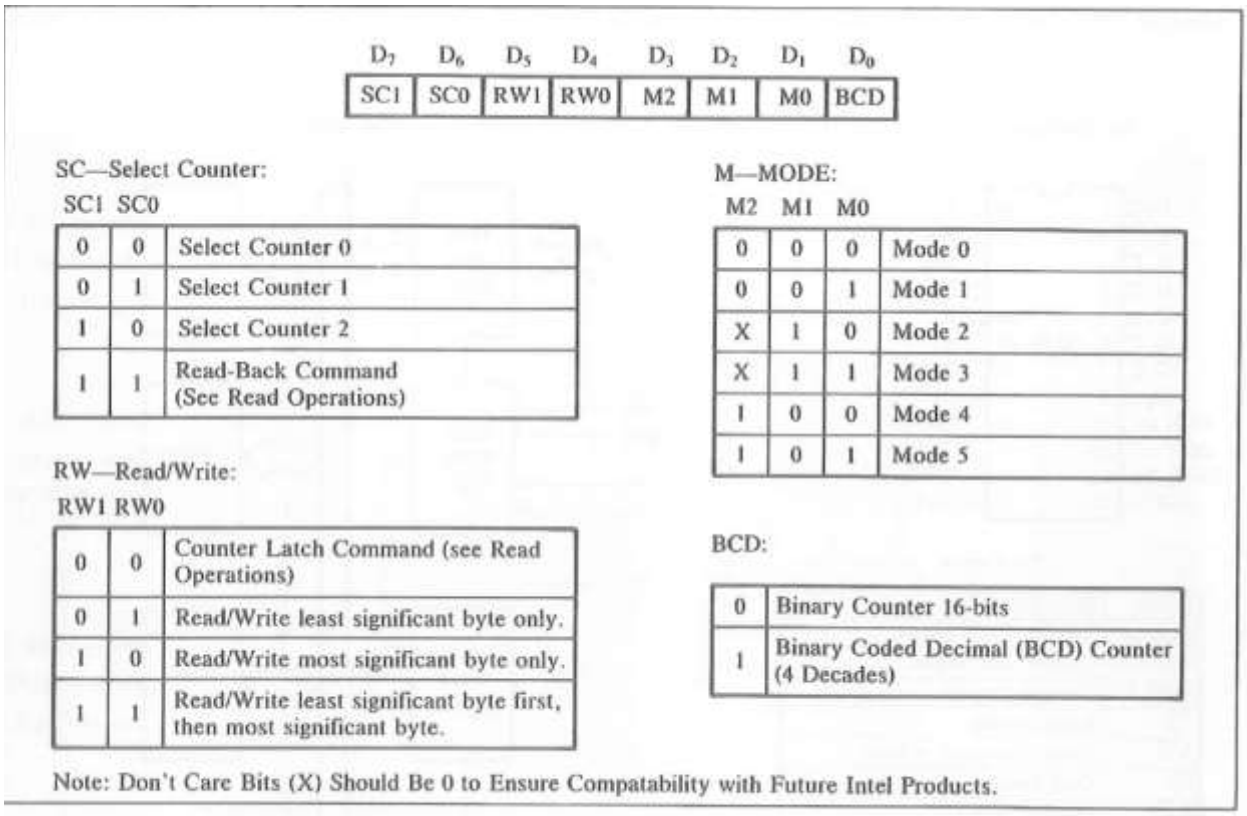
CONTROL WORD REGISTER

This register is accessed when lines A_0 and A_1 are at logic 1. It is used to write a command word which specifies the counter to be used, its mode, and either a Read or a Write operation. The control word format is shown in Figure 15.24.

MODE

The 8254 can operate in six different modes, and the gate of a counter is used either to disable or enable counting, as shown in Figure 15.25. However, to maintain clarity, only one mode (Mode 0) is illustrated first, and details of the remaining modes are discussed in Section 15.4.4.

In Mode 0, after the count is written and if the gate is high, the count is decremented every clock cycle. When the count reaches zero, the output goes high and remains high until a new count or mode word is loaded.



D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

SC—Select Counter:

SC1	SC0	
0	0	Select Counter 0
0	1	Select Counter 1
1	0	Select Counter 2
1	1	Read-Back Command (See Read Operations)

M—MODE:

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

RW—Read/Write:

RW1	RW0	
0	0	Counter Latch Command (see Read Operations)
0	1	Read/Write least significant byte only.
1	0	Read/Write most significant byte only.
1	1	Read/Write least significant byte first, then most significant byte.

BCD:

0	Binary Counter 16-bits
1	Binary Coded Decimal (BCD) Counter (4 Decades)

Note: Don't Care Bits (X) Should Be 0 to Ensure Compatability with Future Intel Products.

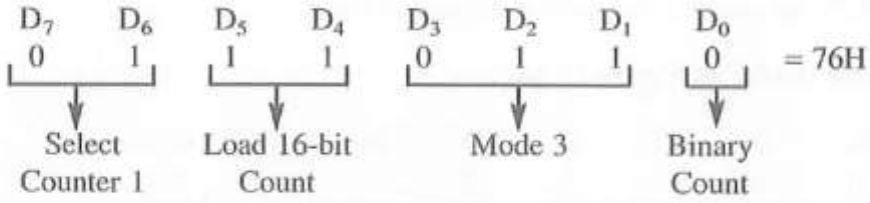
FIGURE 15.24
8254 Control Word Format

Q.4 Write instruction to generate a 1kHz square wave from counter1 . Assume the gate of the counter 1 is tied to +5V through a 10K register. Explain its significance of connecting the gate to +5V.

Ans.

To generate a square wave from Counter 1, it should be initialized in Mode 3 (refer to Figure 15.27 for modes).

Control Word (Refer to Figure 15.24)



Count In Mode 3, the count is decremented by two for every clock period. If the count is N, N/2 clock pulses provide the upper half of the square wave. The count is loaded again and N/2 clock pulses provide the lower half.

In this example, the clock frequency of the 8254 is 2 Mhz (0.5 μs clock period), and the square wave frequency is 1 kHz (1 ms clock period). Therefore, we need a count for 1 ms delay.

$$\text{Count} = \frac{1 \times 10^{-3}}{0.5 \times 10^{-6}} = 2000 = 07D0H$$

This count can also be calculated by dividing the clock frequency by the square wave frequency (2 MHz/1 kHz = 2000).

Instructions

```
SQWAVE:   MVI A, 01110110B   ;Control word Mode 3 and Counter 1
          OUT 83H            ;Write in 8254 control register
          MVI A, D0H         ;Low-order byte of the count
          OUT 81H            ;Load Counter 1 with low-order byte
          MVI A, 07H         ;High-order byte of the count
          OUT 81H            ;Load Counter 1 with high-order byte
          HALT
```

To run Counter 1, the gate of that counter must be tied high; otherwise the counter action is inhibited.

Unit 5:

Short Answers:(2 Marks Each)

Q.1 What is the application of USART?

Ans. 8251 universal synchronous asynchronous receiver transmitter (USART) acts as a mediator between microprocessor and peripheral to transmit serial data into parallel form and vice versa.

1. It takes data serially from peripheral (outside devices) and converts into parallel data.
2. After converting the data into parallel form, it transmits it to the CPU.
3. Similarly, it receives parallel data from microprocessor and converts it into serial form.
4. After converting data into serial form, it transmits it to outside device (peripheral).

Q. 2 Differentiate between

i) Serial port and parallel port

Ans. The main difference between a serial port and a parallel port is that a serial port transmits data one bit after another, while a parallel port transmits all 8 bits of a byte in parallel. Thus a parallel port transmits data much faster than a serial port. Computers have both serial and parallel ports along with newer technology called a USB (Universal Serial Bus) port.

ii) Asynchron ous and synchronous transmission

Ans. One of the major differences is that in Synchronous Transmission, the sender and receiver should have synchronized clocks before data transmission. Whereas Asynchronous Transmission does not require a clock, but it adds a parity bit to the data before transmission.

Furthermore, the synchronous transmission uses synchronization characters while asynchronous method employs start/stop bits, in order to alert the modem when data are being sent and when are these transmissions are completed are known as message characters.

BASIS FOR COMPARISON	SYNCHRONOUS TRANSMISSION	ASYNCHRONOUS TRANSMISSION
Meaning	Transmission starts with the block header which holds a sequence of bits.	It uses start bit and stop bit preceding and following a character respectively.

Transmission manner	Sends data in the form of blocks or frames	Sends 1 byte or character at a time
Synchronization	Present with the same clock pulse.	Absent
Transmission Speed	Fast	Slow
Gap between the data	Does not exist	Exist
Cost	Expensive	Economical
Time Interval	Constant	Random

Q.3 What is application of RS232C and RS422A?

Ans. RS-422 provides for data transmission, using balanced, or differential, signaling, with unidirectional/non-reversible, terminated or non-terminated transmission lines, point to point, or multi-drop.

RS232 was first introduced in 1962 by IBM. RS-232 is the most common serial interface used on most Windows-compatible desktop computers. RS-232 only allows for one transmitter and one receiver on each line. RS-232 also uses a Full-Duplex transmission method. RS232 can transmit up to 1Mbps with maximum distance up to 50 ft, but some RS-232 boards devices are limited to 115.2 kbps. Note that RS-422/RS-485 interface is not available on most IBM Pcs.

Descriptive Answers: (5/20 Marks)

Q. 1 Draw the block diagram and pin description of USART 8251 and briefly explain format of its mode, command word and status word?

Ans. USART

It is possible to use either of the two methods. • There are special IC chips for serial data communication • UART: universal asynchronous receiver transmitter • USART: universal

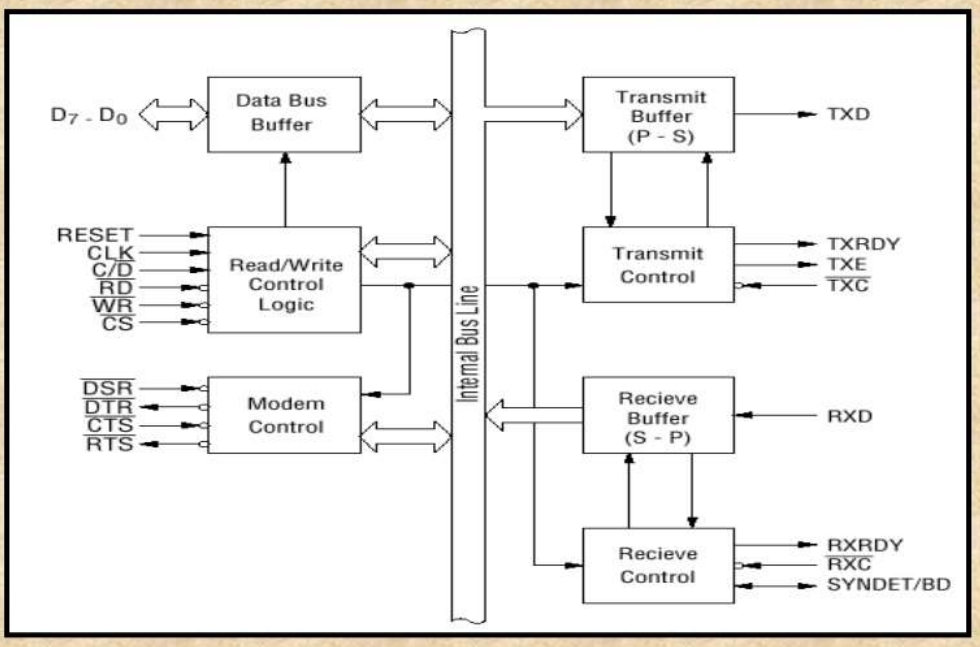
Synchronous/Asynchronous Receiver/Transmitter • COM port in the original IBM PC uses 8250 UART • INTEL has USART 8251 • National Semiconductor's improved version of 8250A is 16450. • 16450, 16550, 16552 (dual 16550) • Data Transmission – simplex – half duplex – full duplex

8251 Receiver

The receiver section: whenever RxD line goes low, control assumes it is a start bit, waits for half bit time and samples again. – responsible for reading the serial bit stream of data atRxD(receive data) input and converting it into parallel form. – RxRDY(receive ready) output switched to logic 1 level to tell the microprocessor that a char. is available and is sitting inside the USART and should be read from the receive –buffer register. – RxC Receiver Clock. Controls the rate which bits are received by the USART. In Asynchronous Mode, the clock can be set to 1,16 or 64 times the baud.

8251 Transmitter

Transmitter section receives parallel data from the microprocessor over the data bus. The character is then automatically framed with the start bit, parity bit, correct number of stop bits, and put into the transmit data buffer register. – Finally, it is shifted out of this register to produce a bit serial output on the TxD line. – TxRDY is switched to logic 1 when the transmit buffer register is empty. – TxE transmitter Empty –This is an output signal. Logic 1 on this line indicates the output register is empty. Reset when a byte is transferred from the buffer to output registers. – TxC Transmitter Clock. Controls the rate which bits are transmitted by the USART. The clock can be set to 1,16 or 64 times the baud (using Mode Word –next slide)



Q. 2 Write Short note on

(a) Centronics

(b) IEEE 488

Ans.

IEEE 488

IEEE 488 is a short-range digital communications 8-bit parallel multi-master interface bus specification developed by Hewlett-Packard (now Agilent and Keysight) as HP-IB (Hewlett-Packard Interface Bus). It subsequently became the subject of several standards, and is generically known as GPIB (General Purpose Interface Bus). Although the bus was created in the late 1960s to connect together automated test equipment, it also had some success during the 1970s and 1980s

as a peripheral bus for early microcomputers, notably the Commodore PET. Newer standards have largely replaced IEEE 488 for computer use, but it is still used by some test equipment.

In the late 1960s, Hewlett-Packard (HP)[1] manufactured various automated test and measurement instruments, such as digital multimeters and logic analyzers. They developed the *HP Interface Bus (HP-IB)* to enable easier interconnection between instruments and controllers (computers and other instruments).

The bus was relatively easy to implement using the technology at the time, using a simple parallel bus and several individual control lines. For example, the HP 59501 Power Supply Programmer and HP 59306A Relay Actuator were both relatively simple HP-IB peripherals implemented in TTL, without the need for a microprocessor.

HP licensed the HP-IB patents for a nominal fee to other manufacturers. It became known as the General Purpose Interface Bus (GPIB), and became a de facto standard for automated and industrial instrument control. As GPIB became popular, it was formalized by various standards organizations.

In 1975, the IEEE standardized the bus as *Standard Digital Interface for Programmable Instrumentation*, **IEEE 488**; it was revised in 1978 (producing IEEE 488-1978). The standard was revised in 1987, and redesignated as **IEEE 488.1** (IEEE 488.1-1987). These standards formalized the mechanical, electrical, and basic protocol parameters of GPIB, but said nothing about the format of commands or data.

In 1987, IEEE introduced *Standard Codes, Formats, Protocols, and Common Commands*, **IEEE 488.2**. It was revised in 1992.[3] IEEE 488.2 provided for basic syntax and format conventions, as well as device-independent commands, data structures, error protocols, and the like. IEEE 488.2 built on IEEE 488.1 without superseding it; equipment can conform to IEEE 488.1 without following IEEE 488.2.

While IEEE 488.1 defined the hardware and IEEE 488.2 defined the protocol, there was still no standard for instrument-specific commands. Commands to control the same class of instrument, *e.g.*, multimeters, varied between manufacturers and even models.

The United States Air Force, and later Hewlett-Packard, recognized this as a problem. In 1989, HP developed their TML language which was the forerunner to Standard Commands for

Programmable Instrumentation (SCPI), introduced as an industry standard in 1990. SCPI added standard generic commands, and a series of instrument classes with corresponding class-specific commands. SCPI mandated the IEEE 488.2 syntax, but allowed other (non-IEEE 488.1) physical transports.

The IEC developed their own standards in parallel with the IEEE, with **IEC 60625-1** and **IEC 60625-2** (IEC 625), later replaced by IEC 60488.

National Instruments introduced a backward-compatible extension to IEEE 488.1, originally known as **HS-488**. It increased the maximum data rate to 8 Mbyte/s, although the rate decreases as more devices are connected to the bus. This was incorporated into the standard in 2003 (IEEE 488.1-2003),[7] over HP's objections.

Characteristics

IEEE 488 is an 8-bit, electrically parallel bus which employs sixteen signal lines — eight used for bi-directional data transfer, three for handshake, and five for bus management — plus eight ground return lines.

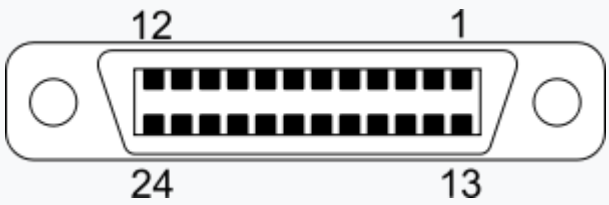
The bus supports 31 five-bit primary addresses numbered form 0 to 30, allocating a unique address to each device on the bus.[13][14]

The standard allows up to 15 devices to share a single physical bus of up to 20 meters total cable length. The physical topology can be linear or star (forked). Active extenders allow longer buses, with up to 31 devices theoretically possible on a logical bus.

Control and data transfer functions are logically separated; a controller can address one device as a "talker" and one or more devices as "listeners" without having to participate in the data transfer. It is possible for multiple controllers to share the same bus, but only one can be the "Controller In Charge" at a time.

In the original protocol, transfers use an interlocked, three-wire *ready–valid–accepted* handshake. The maximum data rate is about one megabyte per second. The later HS-488 extension relaxes the handshake requirements, allowing up to 8 Mbyte/s. The slowest participating device determines the speed of the bus.

Connectors



Female IEEE 488 connector

Pin 1	DIO1	Data input/output bit.
Pin 2	DIO2	Data input/output bit.
Pin 3	DIO3	Data input/output bit.
Pin 4	DIO4	Data input/output bit.
Pin 5	EOI	End-or-identify.
Pin 6	DAV	Data valid.
Pin 7	NRFD	Not ready for data.
Pin 8	NDAC	Not data accepted.
Pin 9	IFC	Interface clear.
Pin 10	SRQ	Service request.
Pin 11	ATN	Attention.
Pin 12	SHIELD	
Pin 13	DIO5	Data input/output bit.

Pin 14	DIO6	Data input/output bit.
Pin 15	DIO7	Data input/output bit.
Pin 16	DIO8	Data input/output bit.
Pin 17	REN	Remote enable.
Pin 18	GND	(wire twisted with DAV)
Pin 19	GND	(wire twisted with NRFD)
Pin 20	GND	(wire twisted with NDAC)
Pin 21	GND	(wire twisted with IFC)
Pin 22	GND	(wire twisted with SRQ)
Pin 23	GND	(wire twisted with ATN)
Pin 24	Logic ground	

IEEE 488 specifies a 24-pin Amphenol-designed micro ribbon connector. Micro ribbon connectors have a D-shaped metal shell, but are larger than D-subminiature connectors. They are sometimes called "Centronics connectors" after the 36-pin micro ribbon connector Centronics used for their printers.

One unusual feature of IEEE 488 connectors is they commonly use a "double-headed" design, with male on one side, and female on the other. This allows stacking connectors for easy daisy-chaining. Mechanical considerations limit the number of stacked connectors to four or fewer, although a workaround involving physically supporting the connectors may be able to get around this.

They are held in place by screws, either UTS (now largely obsolete) or metric M3.5×0.6 threads. Early versions of the standard suggested that metric screws should be blackened to avoid confusion with the incompatible UTS threads. However, by the 1987 revision this was no longer considered necessary because of the prevalence of metric threads.[19]

The IEC 60625 standard prescribes the use of 25-pin D-subminiature connectors (the same as used for the parallel port on IBM PC compatibles). This connector did not gain significant market acceptance against the established 24-pin connector.

CENTRONICS

The Centronics parallel interface is an older and still widely-used standard I/O interface for connecting printer s and certain other devices to computers. The interface typically includes a somewhat cumbersome cable and a 36- pin male and female connector at the printer or other device. The cable plugs into a 25-pin parallel port on the computer. Data flows in one direction only, from the computer to the printer or other device. In addition to eight parallel data lines, other lines are used to read status information and send control signals. Centronics Corporation designed the original Centronics parallel interface for dot matrix printers. In 1981, IBM used this interface as an alternative to the slower one-bit-at-a-time serial interface.

When the Centronics parallel interface was first developed, the main peripheral was the printer. Since then, portable disk drives, tape backup drives, and CD-ROM players are among devices that have adopted the parallel interface. These new uses caused manufacturers to look at new ways to make the Centronics parallel interface better. In 1991, Lexmark, IBM, Texas instruments, and others met to discuss a standard that would offer more speed and bi-directional communication. Their effort and the sponsorship of the IEEE resulted in the IEEE 1284 committee. The IEEE 1284 standard was approved for release in March, 1994.

The IEEE 1284 standard specifies five modes of operation, each mode providing data transfer in either the forward direction (computer to peripheral), backward direction (peripheral to computer), or bi-directional (one direction at a time).

- **Compatibility mode** is the original Centronics parallel interface and intended for use with dot matrix printers and older laser printers. The compatibility mode can be combined with the nibble mode for bi-directional data transfer.
- **Nibble mode** allows data transfer back to the computer. The nibble mode uses the status lines to send 2 nibble s (4-bit units) of data to the computer in two data transfer cycles. This mode is best used with printers.
- **Byte mode** uses software driver s to disable the drivers that control the data lines in order for data to be sent from the printer to the computer. The data is sent at the same speed as when data is sent from the computer to the printer. One byte of data is transferred instead of the two data cycles required by the nibble mode.
- **ECP mode** (Enhanced Capability Port mode) is an advanced bi-directional mode for use with printers and scanner s. It allows data compression for image s, FIFO (first in, first out) for items in queue s, and high-speed, bi-directional communication. Data transfer occurs at two to four megabytes per second. An advanced feature of ECP is channel addressing . This is used for multifunction devices such as printer/fax/modem devices. For example, if a printer/fax/modem device needs to print and send data over the modem at the same time, the channel address software driver of the ECP mode assigns a new channel to the modem so that both devices can work simultaneously.
- **EPP mode** (Enhanced Parallel Port mode) was designed by Intel, Xircom, and Zenith Data Systems to provide a high-performance parallel interface that could also be used with the standard interface. EPP mode was adopted as part of the IEEE 1284 standard. The EPP mode uses data cycles that transfer data between the computer and the peripheral and address cycles that assign address, channel, or command information. This allows data transfer speeds of 500 kilobytes to 2 megabytes per second, depending on the speed of the slowest interface. The EPP mode is bi-directional. It is suited for network adapters, data acquisition, portable hard drives, and other devices that need speed.

The computer must determine what the capabilities of the attached peripheral are and which mode to utilize. The concept developed to determine these factors is called negotiation. Negotiation is a sequence of events on the parallel port interface that determines which IEEE 1284 modes the device can handle. An older device will not respond to the negotiation sequence and compatibility

mode is selected to operate that device. A newer device will respond to the negotiation sequence and a more advanced mode can be set.

Robert Howard and Prentice Robinson began development of a low-cost printer at Centronics, a subsidiary of Wang Laboratories that produced specialty computer terminals. The printer used the dot matrix printing principle, with a print head consisting of a vertical row of seven metal pins connected to solenoids. When power was applied to the solenoids, the pin was pushed forward to strike the paper and leave a dot. To make a complete character glyph, the print head would receive power to specified pins to create a single vertical pattern, then the print head would move to the right by a small amount, and the process repeated. On their original design, a typical glyph was printed as a matrix seven high and five wide, while the "A" models used a print head with 9 pins and formed glyphs that were 9 by 7.

This left the problem of sending the ASCII data to the printer. While a serial port does so with the minimum of pins and wires, it requires the device to buffer up the data as it arrives bit by bit and turn it back into multi-bit values. A parallel port makes this simpler; the entire ASCII value is presented on the pins in complete form. In addition to the seven data pins, the system also needed various control pins as well as electrical grounds. Wang happened to have a surplus stock of 20,000 Amphenol 36-pin micro ribbon connectors that were originally used for one of their early calculators. The interface only required 21 of these pins, the rest were grounded or not connected. The connector has become so closely associated with Centronics that it is now popularly known as the "Centronics connector".

The Centronics Model 101 printer, featuring this connector, was released in 1970. The host sent ASCII characters to the printer using seven of eight data pins, pulling them high to +5V to represent a 1. When the data was ready, the host pulled the *STROBE* pin low, to 0 V. The printer responded by pulling the *BUSY* line high, printing the character, and then returning BUSY to low again. The host could then send another character. Control characters in the data caused other actions, like the CR or EOF. The host could also have the printer automatically start a new line by pulling the *AUTOFEED* line high, and keeping it there. The host had to carefully watch the BUSY line to ensure it did not feed data to the printer too rapidly, especially given variable-time operations like a paper feed.

The printer side of the interface quickly became an industry *de facto* standard, but manufacturers used various connectors on the system side, so a variety of cables were required. For example, NCR used the 36-pin micro ribbon connector on both ends of the connection, early VAX systems used a DC-37 connector, Texas Instruments used a 25-pin card edge connector and Data General used a 50-pin micro ribbon connector. When IBM implemented the parallel interface on the IBM PC, they used the DB-25F connector at the PC-end of the interface, creating the now familiar parallel cable with a DB25M at one end and a 36-pin micro ribbon connector at the other.

In theory, the Centronics port could transfer data as rapidly as 75,000 characters per second. This was far faster than the printer, which averaged about 160 characters per second, meaning the port spent much of its time idle. The performance was defined by how rapidly the host could respond to the printer's BUSY signal asking for more data. To improve performance, printers began incorporating buffers so the host could send them data more rapidly, in bursts. This not only reduced (or eliminated) delays due to latency waiting for the next character to arrive from the host, but also freed the host to perform other operations without causing a loss of performance. Performance was further improved by using the buffer to store several lines and then printing in both directions, eliminating the delay while the print head returned to the left side of the page. Such changes more than doubled the performance of an otherwise unchanged printer, as was the case on Centronics models like the 102 and 308.

Q. 3 Interface the 8085 with 8279 and draw the suitable diagram?

Ans.

In a microprocessor b system, when keyboard and 7-segment LED display is interfaced using ports or latches then the processor has to carry the following task.

- Keyboard scanning
- Key debouncing
- Key code generation
- Sending display code to LED
- Display refreshing

Interfacing 8279 with 8085 processor:

- A typical Hexa keyboard and 7-segment LED display interfacing circuit using 8279 is shown.

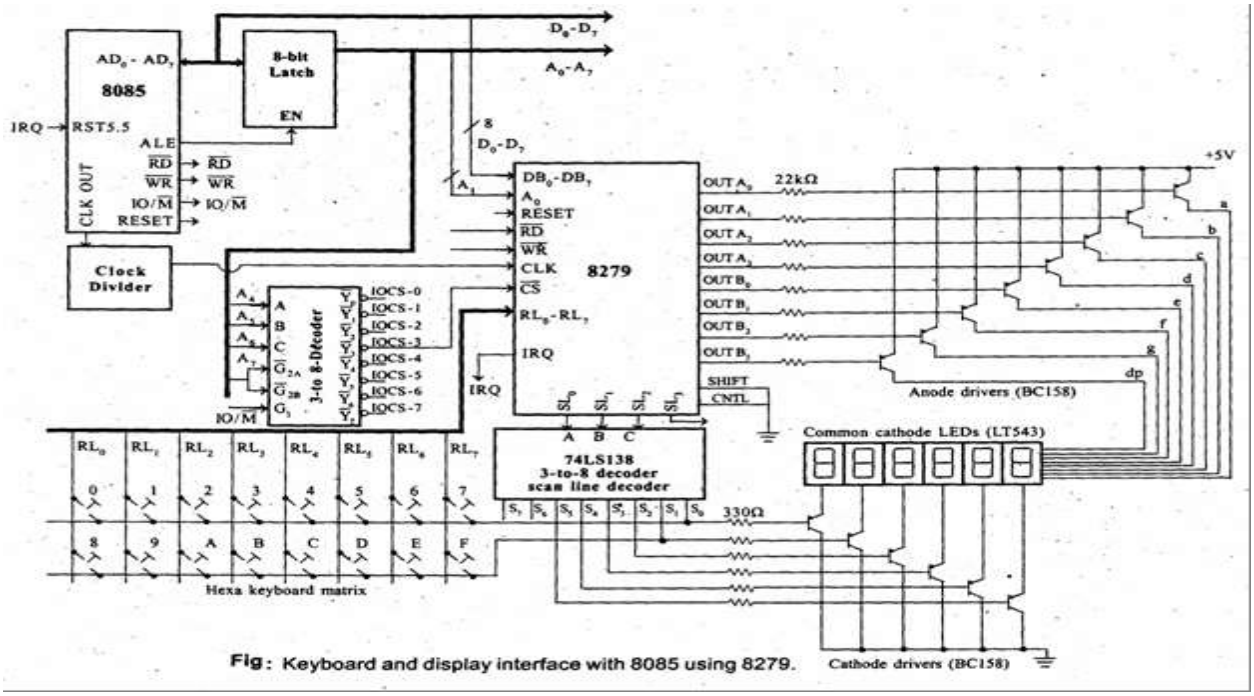


Fig: Keyboard and display interface with 8085 using 8279.

- The circuit can be used in 8085 microprocessor system and consist of 16 numbers of hexa-keys and 6 numbers of 7-segment LEDs.
- The 7-segment LEDs can be used to display six digit alphanumeric character.
- The 8279 can be either memory mapped or I/O mapped in the system. In the circuit shown is the 8279 is I/O mapped.
- The address line A0 of the system is used as A0 of 8279.
- The clock signal for 8279 is obtained by dividing the output clock signal of 8085 by a clock divider circuit.

- The chip select signal is obtained from the I/O address decoder of the 8085 system. The chip select signals for I/O mapped devices are generated by using a 3-to-8 decoder.
- The address lines A4, A5 and A6 are used as input to decoder.
- The address line A7 and the control signal IO/M (low) are used as enable for decoder.
- The chip select signal IOCS-3 is used to select 8279.
- The I/O address of the internal devices of 8279 are shown in table.

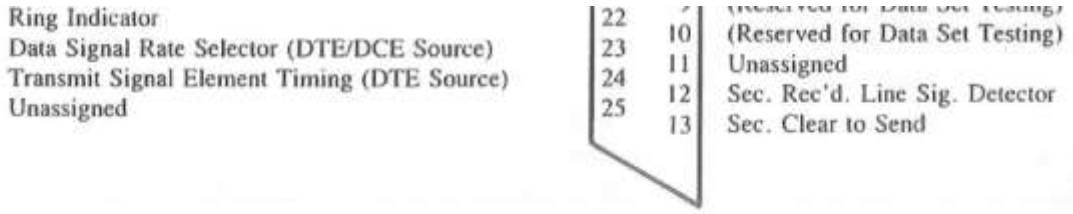
Internal Device	Binary Address								Hexa Address
	Decoder input and enable				Input to address line of 8279				
	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
Data register	0	0	1	1	x	x	x	0	30
Control register	0	0	1	1	x	x	x	1	31

- The circuit has 6 numbers of 7-segment LEDs and so the 8279 has to be programmed in encoded scan. (Because in decoded scan, only 4 numbers of 7-segment LEDs can be interfaced)
- In encoded scan the output of scan lines will be binary count. Therefore an external, 3-to-8 decoder is used to decode the scan lines SL0, SL1 and SL2 of 8279 to produce eight scan lines S0 to S7.
- The decoded scan lines S0 and S1 are common for keyboard and display.
- The decoded scan lines S2 to S5 are used only for display and the decoded scan lines S6 and S7 are not used in the system.
- Anode and Cathode drivers are provided to take care of the current requirement of LEDs.
- The pnp transistors, BC 158 are used as driver transistors.

- The anode drivers are called segment drivers and cathode drivers are called digit drivers.
- The 8279 output the display code for one digit through its output lines (OUT A0 to OUT A3 and OUT B0 to OUT B3) and send a scan code through, SL0- SL3.
- The display code is inverted by segment drivers and sent to segment bus.
- The scan code is decoded by the decoder and turns ON the corresponding digit driver. Now one digit of the display character is displayed. After a small interval (10 milli-second, typical), the display is turned OFF (i.e., display is blanked) and the above process is repeated for next digit. Thus multiplexed display is performed by 8279.
- The keyboard matrix is- formed using the return lines, RL0 to RL3 of 8279 as columns and decoded scan lines S0 and S1 as rows.
- A hexa key is placed at the crossing point of each row and column. A key press will short the row and column. Normally the column and row line will be high.
- During scanning the 8279 will output binary count on SL0 to SL3, which is decoded by decoder to make a row as zero. When a row is zero the 8279 reads the columns. If there is a key press then the corresponding column will be zero.
- If 8279 detects a key press then it wait for debounce time and again read the columns to generate key code.
- In encoded scan keyboard mode, the 8279 stores an 8-bit code for each valid key press. The keycode consist of the binary value of the column and row in which the key is found and the status of shift and control key.
- After a scan time, the next row is made zero and the above process is repeated and so on. Thus 8279 continuously scan the keyboard.

Q. 4 Explain RS232C with the help of suitable diagram?

Ans.



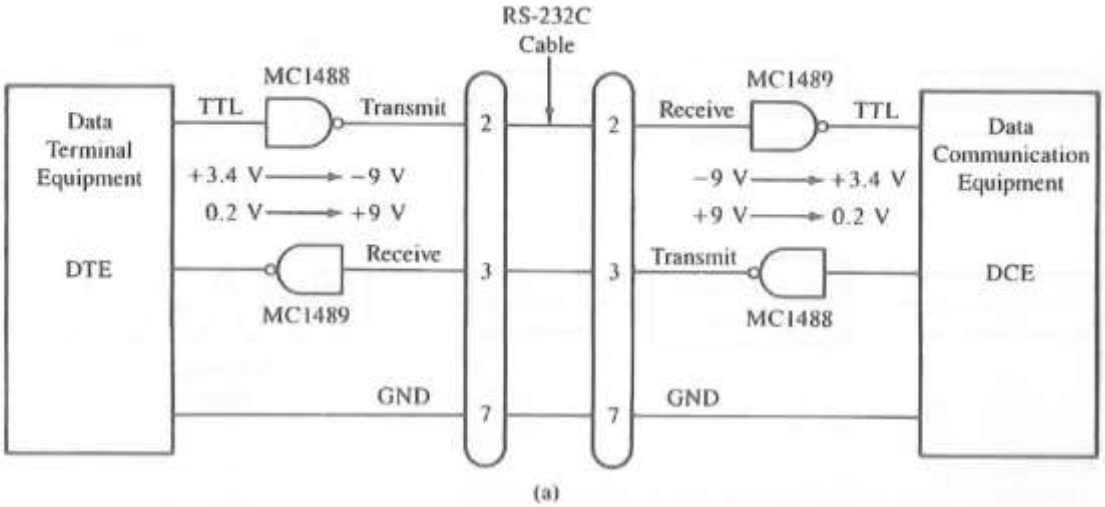
(b)

FIGURE 16.5
Minimum Configuration of RS-232C Signals and Voltage Levels (a) and RS-232C Signal Definitions and Pin Assignments (b)

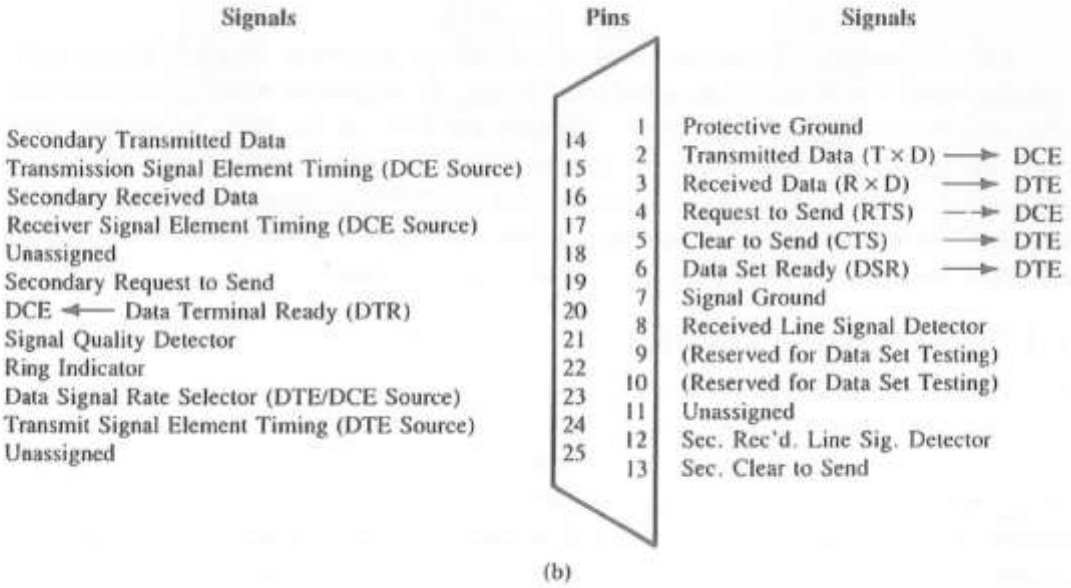
SOURCE: Courtesy of Electronic Industries Association.

speed data transmission, two new standards—RS-422A and RS-423A—have been d oped in recent years; however, they are not yet widely used.

To appreciate the difficulties and confusion in this standard, one has to examir historical background. The RS-232 standard was developed during the initial day computer timesharing, long before the existence of TTL logic, and its primary focus to have compatibility between a terminal and a modem. However, the same standa now being used for communications between computers and peripherals, and the rol a data terminal and a modem have become ambiguous. Should a computer be consid



(a)



(b)

FIGURE 16.5
Minimum Configuration of RS-232C Signals and Voltage Levels (a) and RS-232C Signal Definitions and Pin Assignments (b)

RS-232C

Figure 16.5(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 ±12 V). 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 Figure 16.5(a). The line driver, MC1488, converts logic 1 into approximately −9 V and logic 0 into +9 V, as shown in Figure 16.5(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 the transmitted signal to the higher level? The primary reason is that the standard was defined before the TTL levels came into existence; before 1960, most equipment was designed to handle higher voltages. The other reason is that this standard provides a higher level of noise margin—from −3 V to +3 V.

The minimum interface between a computer and a peripheral requires three lines: pins 2, 3, and 7, as shown in Figure 16.5(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 Figure 16.6(b). In Figure 16.6, the microcomputer is defined as a DTE, and it can be connected to the modem, defined as a DCE, without any modification in the RS-232 cable, as shown in Figure 16.6(a). However, when it is connected to the printer, the transmit and the re-

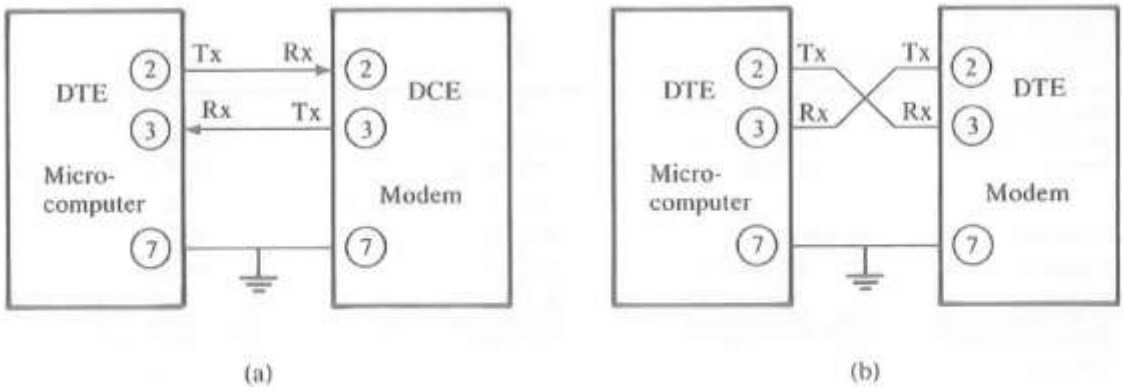


FIGURE 16.6
RS-232C Connections: DTE to DCE (a) and DTE to DTE (b)