Unit 6

Unit 1

1 Digital Logic Families

1.1 : Classification of Logic Families

Q.1 What is logic family ? Give the classification of logic families.

☞ [SPPU : May-10,12, Dec.-08, Marks 4]

Ans. : A digital logic family is a group of compatible devices with the same logic levels and supply voltages. According to components used in the logic family, digital logic families are classified as shown in the Fig. Q.1.1.

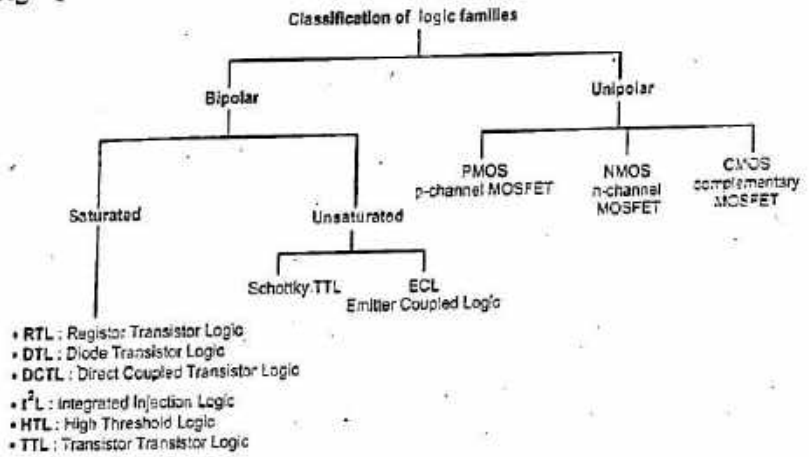


Fig. Q.1.1 Classification of logic families

1.2 : Digital IC Characteristics

Q.2 State and explain any four characteristics of digital ICs.

☞ [SPPU : Dec.-08,15, May-17, Marks 6]

Ans. : Propagation Delay : The propagation delay of a gate is basically the time interval between the application of an input pulse and the occurrence of the resulting output pulse. The propagation delay is a very

important characteristic of logic circuits because it limits the speed at which they can operate. The shorter the propagation delay, the higher the speed of the circuit and vice-versa.

Power Dissipation : The amount of power that an IC dissipates is determined by the average supply current, I_{CC}, that it draws from the V_{CC} supply. It is the product of I_{CC} and V_{CC}.

Current and Voltage Parameter

$V_{IH\,(min)}$- High-Level Input Voltage : It is the minimum voltage level required for a logical 1 at an input. Any voltage below this level will not be accepted as a HIGH by the logic circuit.

$V_{IL\,(max)}$- Low-Level Input Voltage : It is the maximum voltage level required for a logic 0 at an input. Any voltage above this level will not be accepted as a LOW by the logic circuit.

$V_{OH\,(min)}$- High-Level Output Voltage : It is the minimum voltage level at a logic circuit output in the logical 1 state under defined load conditions.

$V_{OL\,(max)}$- Low-Level Output Voltage : It is the maximum voltage level at a logic circuit output in the logical 0 state under defined load conditions.

I_{IH}- High-Level Input Current : It is the current that flows into an input when a specified high-level voltage is applied to that input.

I_{IL}- Low-Level Input Current : It is the current that flows into an input when a specified low-level voltage is applied to that input.

I_{OH}- High-Level Output Current : It is the current that flows from an output in the logical 1 state under specified load conditions.

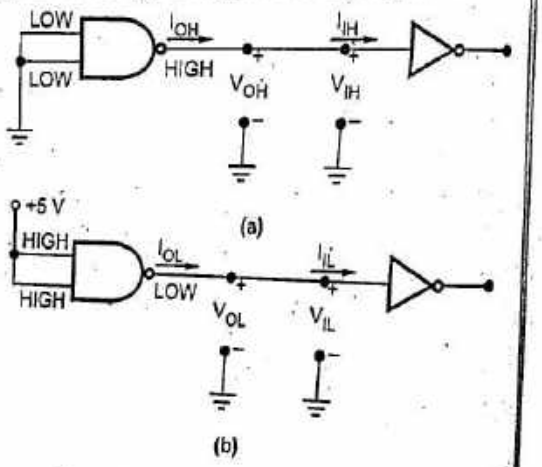


Fig. Q.2.1 Currents and voltages in the two logic states

I_{OL}- Low-Level Output Current : It is the current that flows from an output in the logical 0 state under specified load conditions.

Noise Margin : The **noise immunity** of at a logic circuit refers to the circuit's ability to tolerate the noise without causing spurious changes in the output voltage. To avoid this problem due to noise, voltage level $V_{IH(min)}$ is kept at a few fraction of volts below $V_{OH(min)}$ and voltage level $V_{IL(max)}$ is kept above $V_{OL(max)}$, at the design time.

V_{NH} is the difference between the lowest possible HIGH output, $V_{OH(min)}$ and the minimum voltage, $V_{IH(min)}$ required for a HIGH input. This voltage difference, V_{NH} is called high-state noise margin. Similarly, we have low-state noise margin. It is the voltage difference between the largest possible low output, $V_{OL(max)}$ and the maximum voltage, $V_{IL(max)}$ required for a LOW input.

In short we can write as,

$$V_{NH} = V_{OH(min)} - V_{IH(min)} \quad \text{and} \quad V_{NL} = V_{IL(max)} - V_{OL(max)}$$

Fan-in and Fan-out : The maximum number of inputs of several gates that can be driven by the output of a logic gate is decided by the parameter called **fan-out.** In general, the fan-out is defined as the maximum number of inputs of the same IC family that the gate can drive maintaining its output levels within the specified limits.

The **fan-in** of a digital logic gate refers to the number of inputs.

Speed Power Product (Figure of Merit) : • In general, for any digital IC, it is desirable to have shorter propagation delays (higher speed) and lower values of power dissipation. There is usually a trade-off between switching speed and power dissipation in the design of a logic circuit i.e. speed is gained at the expense of increased power dissipation. Therefore, a common means for measuring and comparing the overall performance of an IC family is the **Speed-Power Product (SPP). It is also called Figure of Merit.**

Operating Temperature Range : • It is the temperature range specified by the logic family within which devices are guaranteed to work reliably.

Power Supply Requirements : • Power supply requirements differ from logic family to family. For eample, it is 5V for TTL family and 3-15 volts for CMOS family. Further more, power supply tolerance also depends on logic family. For example, for 74 series TTL family it is ± 0.25 V and for 54 series TTL family it is ± 0.5 V.

1.3 : TTL : Standard TTL Characteristics, Operation of TTL NAND Gate

Q.3 With neat circuit diagram explain the operation of two-input TTL NAND gates. [SPPU : May-05,10,12,13,16, Dec.-07, Marks 8]

Ans. : The Fig. Q.3.1 (a) shows the circuit diagram of 2-input NAND gate. Its input structure consists of multiple-emitter transistor and output structure consists of totem-pole output. Here, Q_1 is an NPN transistor having two emitters, one for each input to the gate. Although this circuit looks complex, we can simplify its analysis by using the diode equivalent of the multiple-emitter transistor Q_1, as shown in Fig. Q.3.1 (b). Diodes D_2 and D_3 represent the two E-B junctions of Q_1 and D_4 is the collector-base (C-B) junction.

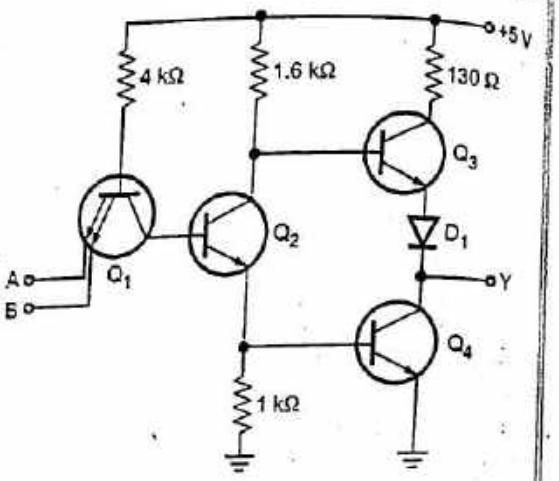


Fig. Q.3.1 (a) Two input TTL NAND gate

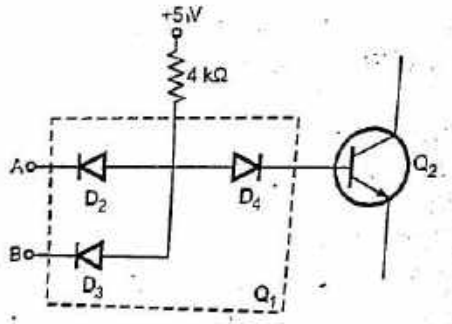


Fig. Q.3.1 (b) Diode equivalent for Q_1

The input voltages A and B are either LOW (ideally grounded) or HIGH (ideally +5 volts). If either A or B or both are low, the corresponding diode conducts and the base of Q_1 is pulled down to approximately 0.7 V. This reduces the base voltage of Q_2 to almost zero. Therefore, Q_2 cuts off. With Q_2 open, Q_4 goes into cut-off and the Q_3 base is pulled HIGH. Since Q_3 acts as an emitter follower, the Y output is pulled up to a HIGH voltage. On the other hand, when A and B both are HIGH, the emitter diode of

Q_1 are reversed biased making them off. This causes the collector diode D_4 to go into forward conduction. This forces Q_2 base to go HIGH. In turn, Q_4 goes into saturation, producing a low output. Table Q.3.1 summarizes all input and output conditions.

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Table Q.3.1 Truth table for 2-input NAND gate

Without diode D_1 in the circuit, Q_3 will conduct slightly when the output is low. To prevent this, the diode is inserted; its voltage drop keeps the base-emitter diode of Q_3 reverse-biased. In this way, only Q_4 conducts when the output is low.

Q.4 Draw three input standard TTL NAND gate circuit and explain its operation. [SPPU : Dec.-08,10, Marks 8]

Ans. : The Fig. Q.4.1 shows the three input TTL NAND gate. It is same as that of two input TTL NAND gate except that its Q_1(NPN) transistor has three emitters instead of two. Rest of the circuit is same. For three input NAND gate if all the inputs are logic 1 then and then only output is logic 0; otherwise output is logic 1. The operation is similar to the 2-input NAND gate. The Table Q.4.1 shows the truth table for 3-input NAND gate.

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Table Q.4.1 Truth table of 3-input NAND gate

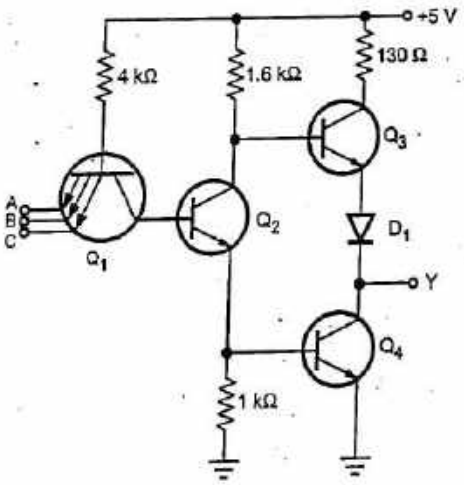


Fig. Q.4.1 Three input TTL NAND gate

Q.5 What do you mean by open collector output ? Explain with suitable circuit diagram ?

☞ [SPPU : May-15, Marks 4]

Ans. : Some TTL devices provide another type of output called **open collector output.** The outputs of two different gates with open collector output can be tied together. This is known as wired logic.

Fig. Q.5.1 shows a 2-input NAND gate with an open-collector output eliminates the pull-up transistor Q_3, D_1 and R_4. The output is taken from the open collector terminal of transistor Q_4.

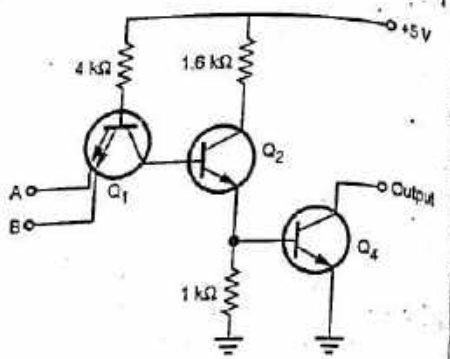


Fig. Q.5.1 Open collector 2-input TTL NAND gate

Because the collector of Q_4 is open, a gate like this will not work properly until you connect an external pull-up resistor, as shown in Fig. Q.5.2 When Q_4 is ON, output is low and when Q_4 is OFF output is tied to V_{CC} through an external pull up resistor.

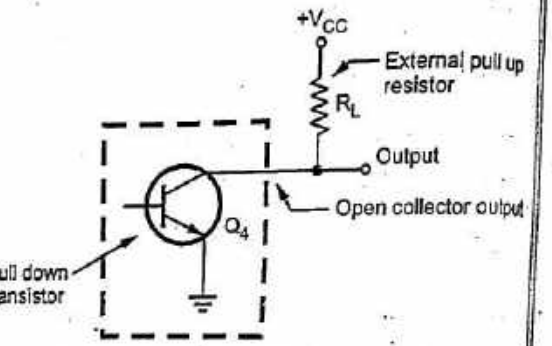


Fig. Q.5.2 Open collector output with pull-up resistor

Q.6 Give the advantages and disadvantages of totem-pole output stage arrangement.

☞ [SPPU : May-05, Dec.-07, Marks 4]

OR Explain the advantages of open collector output.

☞ [SPPU : Dec.-06,15, May-08,15 Marks 4]

OR Give comparison between totem-pole and open colletor outputs.

Ans. :

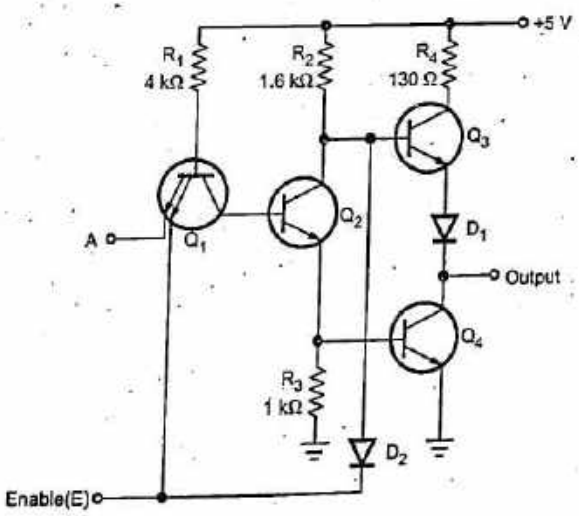
Sr.No.	Totem-pole	Open collector
1.	Output stage consists of pull-up transistor (Q_3), diode resistor and pull-down transistor (Q_4).	Output stage consists of only pull-down transistor.
2.	External pull-up resistor is not required.	External pull-up resistor is required for proper operation of gate.
3.	Output of two gates cannot be tied together.	Output of two gates can be tied together using wired AND technique.
4.	Operating speed is high.	Operating speed is low.

Table Q.6.1 Comparison of totem-pole and open collector output

Q.7 Draw and explain the circuit diagram of tri-state TTL NAND gate.

☞ [SPPU : May-05, Dec.-05, 07, Marks 6]

Ans. : The tristate configuration is a third type of TTL output configuration. It utilizes the high-speed operation of the totem-pole arrangement while permitting outputs to be wired-ANDed (connected together). It is called tristate TTL because it allows three possible output stages : HIGH, LOW and high-impedance.



A	E	Output
0	1	1
1	1	0
X	0	Floating

Truth table

Fig. Q.7.1 Tristate TTL Inverter

Fig. Q.7.1 shows the simplified circuit for tristate inverter. It has two inputs A and E.

A is the normal logic input whereas E is an ENABLE input. When ENABLE input is HIGH, the circuit works as a normal inverter. Because when E is HIGH, the state of the transistor Q_1 (either ON or OFF) depends on the logic input A, and the additional component diode is open circuited as its cathode is at logic HIGH. When ENABLE input is LOW, regardless of the state of logic input A, the base-emitter junction of Q_1 is forward biased and as a result it turns ON. This shunts the current through R_1 away from Q_2 making it OFF. As Q_2 is OFF, there is no sufficient drive for Q_4 to conduct and hence Q_4 turns OFF. The LOW at ENABLE input also forward-biases diode D_2, which shunt the current away from the base of Q_3, making it OFF. In this way, when ENABLE input is LOW, both transistors are OFF and output is at high impedance state.

Q.8 Explain the following characteristics of TTL logic families :
i) Power dissipation ii) Noise margin iii) Propagation delay
iv) Fan out.

OR Explain standard TTL characteristics in detail.

☞ [SPPU : Dec.-08,10,12, May-10,13, Marks 8]

OR Define the following terms related to logic families. Mention typical values for standard TTL family :
i) Power dissipation ii) Fan-in iii) V_{IL}, V_{OH} iv) Noise margin.

☞ [SPPU : May-12, Marks 8]

OR Define the following terms and mention its standard values for TTL family : i) Voltage parameters ii) Power dissipation iii) Fan out

☞ [SPPU : May-16, Maks 6]

Ans. : Supply voltage and temperature range : Both the 74 series and 54 series operate on supply voltage of 5 V. The 74 series works reliably over the range 4.75 V to 5.25 V, while the 54 series can tolerate a supply variation of 4.5 to 5.5 V.

Voltage levels and noise margin : Table Q.8.1 shows the input and output logic voltage levels for the standard 74 series. The minimum and maximum values shown in the Table Q.8.1 are for worst case conditions of power supply, temperature and loading conditions.

Voltages	Minimum	Typical	Maximum
V_{OL}	–	0.2	0.4
V_{OH}	2.4	3.4	–
V_{IL}	–	–	0.8
V_{IH}	2.0	–	–

Table Q.8.1 Voltage levels

For TTL, Low state noise margin, V_{NL} and high state noise margin, V_{NH} both are equal and 0.4 V.

Power dissipation and propagation delay : A standard TTL gate has an average power dissipation of about 10 mW.

Fan-out : A standard TTL output can typically drive 10 standard TTL inputs. Therefore, standard TTL has fan-out 10.

1.4 : CMOS : Standard CMOS Characteristics, Operation of CMOS NAND Gate

Q.9 Draw the structure of CMOS inverter gate. Explain its working.

☞ [SPPU : Dec.-07, Marks 3; May-12, Marks 4]

Ans. : Fig. Q.9.1 shows the basic CMOS inverter circuit. It consists of two MOSFETs in series in such a way that the P-channel device has its source connected to + V_{DD} (a positive voltage) and the N-channel device has its source connected to ground. The gates of the two devices are connected together as the common input and the drains are connected together as the common output.

1. When input is HIGH, the gate of Q_1 (P-channel) is at 0 V relative to the source of Q_1 i.e. $V_{gs1} = 0$ V. Thus, Q_1 is OFF. On the other hand, the gate of Q_2 (N-channel) is at $+ V_{DD}$ relative to its source i.e. $V_{gs2} = + V_{DD}$. Thus, Q_2 is ON. This will produce $V_{OUT} = 0$ V.

2. When input is LOW, the gate of Q_1 (P-channel) is at a negative potential relative to its source while Q_2 has $V_{gs} = 0$ V. Thus, Q_1 is ON and Q_2 is OFF. This produces output voltage approximately $+V_{DD}$.

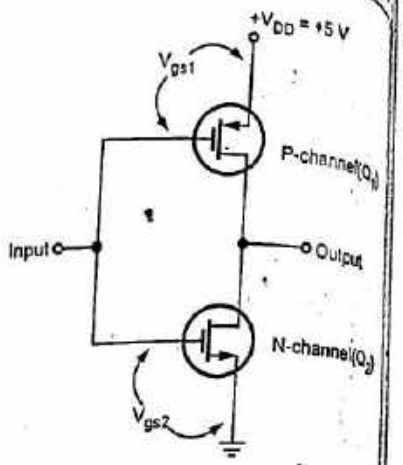


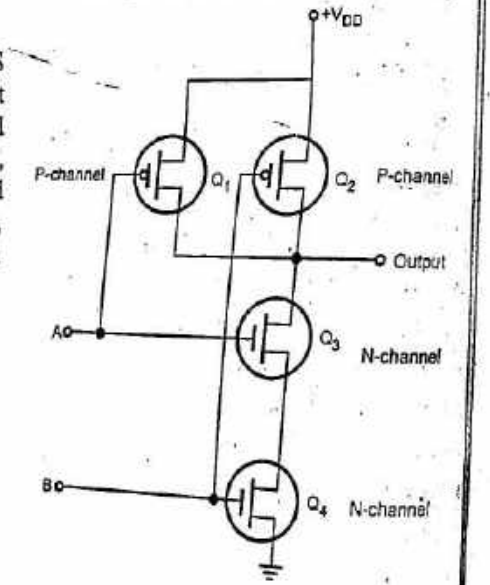
Fig. Q.9.1 CMOS inverter circuit

Q.10 Explain with neat diagram two input CMOS NAND gate.

☞ [SPPU : Dec.-12,14, May-14, Marks 8]

Ans. :

Fig. Q.10.1 shows CMOS 2-input NAND gate. It consists of two P-channel MOSFETs, Q_1 and Q_2, connected in parallel and two N-channel MOSFETs, Q_3 and Q_4 connected in series.



Schematic

Fig. Q.10.1 CMOS NAND gate

A	B	Q_1	Q_2	Q_3	Q_4	Output
0	0	ON	ON	OFF	OFF	1
0	1	ON	OFF	OFF	ON	1
1	0	OFF	ON	ON	OFF	1
1	1	OFF	OFF	ON	ON	0

Table Q.10.1 Truth table of NAND gate

Here, the gates of both P-channel MOSFETs are negative with respect to their sources, since the sources are connected to + V_{DD}. Thus, Q_1 and Q_2 are both ON. Since the gate - to - source voltages of Q_3 and Q_4 (N-channel MOSFETs) are both 0 V, those MOSFETs are OFF. The output is therefore connected to $+V_{DD}$ (HIGH) through Q_1 and Q_2 and is disconnected from ground. When A = 0 and B = $+V_{DD}$. Q_1 is on because $V_{GS_1} = -V_{DD}$ and Q_4 is ON because $V_{GS_4} = +V_{DD}$. MOSFETs Q_2 and Q_3 are off because their gate-to-source voltages are 0 V. Since Q_1 is ON and Q_3 is OFF, the output is connected to + V_{DD} and it is disconnected from ground. When A = $+V_{DD}$ and B = 0 V, the situation is similar (not shown); the output is connected to + V_{DD} through Q_2 and it is disconnected from ground because Q_4 is OFF. Finally, when both inputs are high (A = B = $+V_{DD}$), MOSFETs Q_1 and Q_2 are both OFF and Q_3 and Q_4 are both ON. Thus, the output is connected to the ground through Q_3 and Q_4 and it is disconnected from $+V_{DD}$. The Table Q.10.1 summarizes the operation of 2-input CMOS NAND gate.

Q.11 Explain with a neat diagram interfacing of TTL gate driving CMOS gates and vice-versa.

☞ [SPPU : May-05,07,13, Dec.-05, Marks 6; Dec.-08,11, Marks 8]

OR How will you connect the output of CMOS logic circuit as an input to TTL logic circuit ? Explain your reason with suitable diagram.

☞ [SPPU : Dec.-16, Marks 6]

Ans. : TTL Driving CMOS : The input current values for CMOS are extremely low compared with the output current capabilities of any TTL series. Thus, TTL has no problem meeting the CMOS input current requirements.

But when we compare the TTL output voltages with the CMOS input voltage requirements we find that :

$V_{OH\ (min)}$ for TTL << $V_{IH\ (min)}$ for CMOS for these situations TTL output must be raised to an acceptable level for CMOS. This can be done by connecting pull-up resistor at the output of TTL, as shown in the Fig. Q.11.1. The pull-up resistor causes the TTL output to rise to approximately 5 V in the HIGH state, thereby providing an adequate CMOS input voltage level.

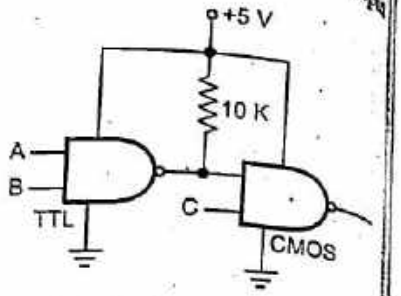


Fig. Q.11.1 TTL driving CMOS using external pull-up resistor

TTL Driving HIGH Voltage CMOS : When output CMOS circuit is operating with V_{DD} greater than 5 V, the situation becomes more difficult. The outputs of many TTL devices cannot be operated at more than 5 V. In such cases some alternative arrangements are made. Two of them are discussed below :

1. When the TTL output cannot be pulled up to V_{DD}, one can use open collector buffer as an interface between totem-pole TTL output and CMOS operating at V_{DD} > 5 V, as shown in the Fig. Q.11.2.

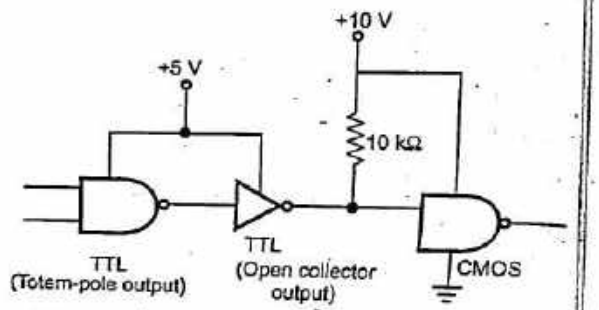


Fig. Q.11.2 Open collector buffer used as interface circuit

2. The second alternative is to use **level translator** circuit, such as the 40104. This is a CMOS chip that is designed to take a low-voltage input (e.g. from TTL) and translate it to high voltage output for CMOS. Fig. Q.11.3 shows the circuit arrangement.

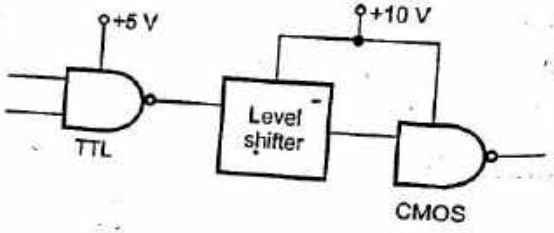


Fig. Q.11.3 Level shifter used as interface circuit

CMOS Driving TTL : CMOS outputs can easily supply enough voltage (V_{OH}) to satisfy the TTL input requirement in the HIGH state (V_{IH}). CMOS outputs can supply more than enough current (I_{OH}) to meet the TTL input current requirements (I_{IH}). Thus no special consideration is required for CMOS driving TTL in the HIGH state.

CMOS output voltage (V_{OL}) satisfies TTL input requirement in the LOW state (V_{IL}). However, the current requirements in the LOW state are not satisfied. The TTL input has a relatively high input current in the LOW state (1.6 mA) and CMOS output current at LOW state (I_{OL}) is not sufficient to drive even one input of the TTL. In such situations some type of interface circuit is needed between the CMOS and TTL devices.

In Fig. Q.11.4 the CMOS 4050 B, non-inverting buffer is used as an interfacing circuit. It has an output current rating of $I_{OL\ (max)}$ = 3 mA which satisfies the TTL input current requirement.

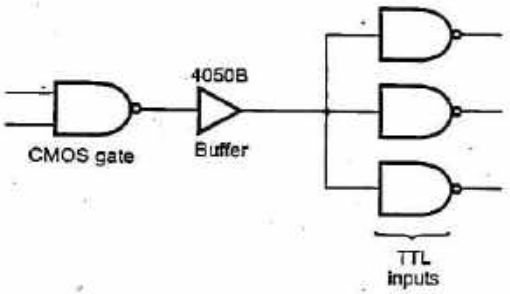


Fig. Q.11.4 CMOS driving TTL in LOW state using buffer

HIGH Voltage CMOS Driving TTL : Some IC manufacturers have provided several 74LS TTL devices that can withstand input voltages as high as 15 V. These devices can be driven directly from CMOS outputs operating at V_{DD} = 15 V. However, most TTL inputs cannot handle more than 7 V and so interface is necessary if they are to be driven from high-voltage CMOS. In such situations **voltage level translators** are used.

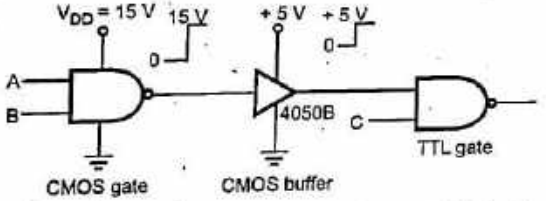


Fig. Q.11.5 Level translation using CMOS buffer

They convert the high-voltage input to a 5 V output that can be connected to TTL.

Fig. Q.11.5 shows how the 4050B can be used to perform this level translation between 15 V and 5 V.

Q.12 Why is a buffer required between some CMOS outputs and TTL inputs ? ☞ [SPPU : May-13, Marks 2]

Ans. : The current requirement of CMOS output at LOW state are not satisfied by TTL input. The TTL input has a relatively high input current in the LOW state (1.6 mA) and CMOS output current at LOW state (I_{OL}) is not sufficient to drive even one input of the TTL. Hence buffer is used for interfacing.

1.5 : Comparison of TTL and CMOS

Q.13 Differentiate between standard TTL and CMOS logic circuit w.r.t. i) Propagation delay ii) FANOUT iii) Figure of merit. List the differences between CMOS and TTL. ☞ [SPPU : May-15,16, Marks 6]

Ans. :

Sr.No.	Parameter	CMOS	TTL
1.	Device used	n-channel and p-channel MOSFET	Bipolar junction transistor
2.	$V_{IH(min)}$	3.5 V	2 V
3.	$V_{IL(max)}$	1.5 V	0.8 V
4.	$V_{OH(min)}$	4.95 V	2.7 V
5.	$V_{OL(max)}$	0.005 V	0.4 V
6.	High level noise margin	V_{NH} = 1.45 V	0.4 V
7.	Low level noise margin	V_{NL} = 1.45 V	0.4 V

8.	Noise immunity	Better than TTL	Less than CMOS
9.	Propagation delay	70 ns	10 ns
10.	Switching speed	Less than TTL	Faster than CMOS
11.	Power dissipation per gate	0.1 mW	10 mW
12.	Speed power product	0.7 pJ	100 pJ
13.	Fan-out	50	10
14.	Power supply voltage	3 - 15 V	Fixed 5 V
15.	Power dissipation	Increase with frequency	Increase with frequency
16.	Application	Portable instrument where battery supply is used.	Laboratory instruments.

Table Q.13.1 Comparison between TTL and CMOS families

END...

Unit 1

2 Signed Binary Representation and Arithmetic and Codes

2.1 : Introudction to Number Systems

Concepts : Number Systems

- Number system is a basis for counting various items.
- The decimal number system has 10 digits : 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9.
- Modern computers communicate and operate with binary numbers which use only the digits 0 and 1.
- When decimal quantities are represented in the binary form, they take more digits.
- For large decimal numbers people have to deal with very large binary strings and therefore, they do not like working with binary numbers. This fact gave rise to three new number systems : **Octal, Hexadecimal** and **Binary Coded Decimal (BCD).**
- Each number system has r set of characters. For example, in decimal number system r equals to 10 has 10 characters from 0 to 9, in binary number system r equals to 2 has 2 characters 0 and 1 and so on.
- In general we can say that, a number represented in radix r, has r characters in its set and r can be any value. This is illustrated in Table 2.1.
- Table 2.2 shows the relationship between decimal, binary, octal and hexadecimal.

Radix (Base) r	Characters in set
2 (Binary)	0, 1
3	0, 1, 2
4	0, 1, 2, 3
5	0, 1, 2, 3, 4
6	0, 1, 2, 3, 4, 5
7	0, 1, 2, 3, 4, 5, 6
8 (Octal)	0, 1, 2, 3, 4, 5, 6, 7
9	0, 1, 2, 3, 4, 5, 6, 7, 8
10 (Decimal)	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
:	
:	
16 (Hexadecimal)	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Table 2.1 Radix and character set

Decimal	Binary	Octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Table 2.2 Relation between decimal, binary, octal and hexadecimal numbers

Q.1 Convert $(1101.101)_2$ to decimal number and explain the process of conversion.

Ans. : • By adding each digit of a binary number in a power of 2 we can find the decimal equivalent of the given binary number. This is illustrated in Fig. Q.1.1.

	2^3	2^2	2^1	2^0	·	2^{-1}	2^{-2}	2^{-3}
N =	1	1	0	1	·	1	0	1
	1×2^3	1×2^2	0×2^1	1×2^0	·	1×2^{-1}	0×2^{-2}	1×2^{-3}

$N = 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} = (13.625)_{10}$

Fig. Q.1.1

Q.2 Convert $(5632.471)_8$ to decimal number and explain the process of conversion.

Ans. : • By adding each digit of an octal number in a power of 8 we can find the decimal equivalent of the given octal number. This is illustrated in Fig. Q.2.1.

	8^3	8^2	8^1	8^0	·	8^{-1}	8^{-2}	8^{-3}
N =	5	6	3	2	·	4	7	1
	5×8^3	6×8^2	3×8^1	2×8^0	·	4×8^{-1}	7×8^{-2}	1×8^{-3}

$N = 5\times8^3 + 6\times8^2 + 3\times8^1 + 2\times8^0 + 4\times8^{-1} + 7\times8^{-2} + 1\times8^{-3} = (2970.611328)_{10}$

Fig. Q.2.1

Q.3 Convert $(3FD.84)_{16}$ into decimal number and explain the process of conversion.

Ans. : By adding each digit of a hexadecimal number in a power of 16 we can find decimal equivalent of the given hexadecimal number.

	16^2	16^1	16^0	·	16^{-1}	16^{-2}
N =	3	F	D	·	8	4
	3×16^2	$F\times16^1$	$D\times16^0$	·	8×16^{-1}	4×16^{-2}

$N = 3\times16^2 + F\times16^1 + D\times16^0 + 8\times16^{-1} + 4\times16^{-2}$

$= 3\times16^2 + 15\times16^1 + 13\times16^0 + 8\times16^{-1} + 4\times16^{-2} = (1021.515625)_{10}$

Q.4 What is the radix called in case of decimal, binary, octal and hexadecimal number systems ?

Ans. : The radix or base for above number system are shown in Table 1.1.

Q.5 Convert 10101101.0111 to octal equivalent and explain the conversion process.

Ans. :

Step 1 : • Make group of 3-bits starting from LSB for integer part and MSB for fractional part, by adding 0s at the end, if required.

Step 1.	0	1	0	1	0	1	1	0	1	·	0	1	1	1	0	0
Step 2.		2			5			5		·		3			4	

Binary (Base 2)

Octal (Base 8)

Adding 0 to make a group of 3-bits

Adding 0s to make a group of 3-bits

Step 2 : • Write equivalent octal number for each group of 3-bits.

$\therefore\ (10101101.0111)_2 = (255.34)_8$

Q.6 Convert $(125.62)_8$ to binary and explain the conversion process.

Ans. :

Step 1 : Write equivalent 3-bit binary number for each octal digit.

Step 2 : Remove any leading or trailing zeros.

	1			2			5		·	6			2	
Step 1.	0	0	1	0	1	0	1	0	1	·	1	1	0	0	1	0
Step 2.			1	0	1	0	1	0	1	·	1	1	0	0	1	

Octal (Base 8)

Leading zeros

Trailing zero

$\therefore\ (125.62)_8 = (1010101.11001)_2$

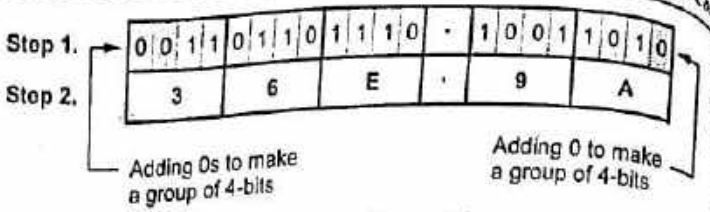
Q.7 Convert 1101101110 · 1001101 to hexadecimal equivalent and explain the conversion process.

Ans. :

Step 1 : Make group of 4-bits starting from LSB for integer part and MSB for fractional part, by adding 0s at the end, if required.

Step 2 : Write equivalent hexadecimal number for each group of 4-bits.

$\therefore\ (1101101110.1001101)_2 = (36E.9A)_{16}$



Q.8 Convert the following : $(62.7)_8 = (\ )_{16} = (\ )_2$.

Ans. :

	6		2		.		7		Octal number
0	0	1	1	0	0	1	0	. 1 1 1 0	Binary number
3			2		.		E		Hex number

$$\therefore (62.7)_8 = (110010.1110)_2 = (32.E)_{16}$$

Q.9 Convert the base - 7 number $(35614)_7$ to base - 12.

Ans. : Step 1 : Convert to decimal

$$(35614)_7 =$$
$$3\times7^4 + 5\times7^3 + 6\times7^2 + 1\times7^1 + 4\times7^0$$
$$= 7203 + 1715 + 294 + 7 + 4$$
$$= 9223$$

12	9223	7
12	768	0
12	64	4
12	5	5
	0	

Step 2 : Convert to base 12

$\therefore \quad (35614)_7 = (5407)_{12}$

Q.10 Convert $(735.25)_{10} = (?)_{16}$ ☞ [SPPU : May-17, Marks 2]

Ans. :

16	735	15 - F
16	45	13 - D
16	2	2
	0	

$0.25 \times 16 = 4.0 \rightarrow 4$

$\therefore (735)_{10} = (2DF)_{16}$

$\therefore (735.25)_{10} = (2DF.4)_{16}$

Q.11 Convert $(101011.111011)_2 = (?)_8 = (?)_{16}$

☞ [SPPU : May-17, Marks 2]

Ans. :

101	011	.	111	011
5	3		7	3

0010	1011	.	1110	1100
2	B		E	C

$$(101011.111011)_2 = (53.73)_8 = (2B.EC)_{16}$$

Q.12 Do the following conversions :

i) $(27.125)_{10} = (?)_2$ ii) $(3A.2F)_{16} = (?)_{10}$ iii) $(1101.0011)_2 = (?)_{10}$

☞ [SPPU : Dec.-16, Marks 6]

Ans. :

i)

2	27	1
2	13	1
2	6	0
2	3	1
2	1	1
	0	

$0.125 \times 2 = 0.25 \rightarrow 0$

$0.25 \times 2 = 0.5 \rightarrow 0$

$0.5 \times 2 = 1.0 \rightarrow 1$

$\therefore (27.125)_{10} = (11011.001)_2$

ii) $(3A.2F)_{16} = 3\times16^1 + 10\times16^0 + 2\times16^{-1} + 15\times16^{-2}$
$= 48 + 10 + 0.125 + 0.05859375$
$= (58.18359375)_{10}$

iii) $(1101.0011)_2 = 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 + 0\times2^{-1}$
$+ 0\times2^{-2} + 1\times2^{-3} + 1\times2^{-4}$
$= 8 + 4 + 1 + 0.125 + 0.0625 = (13.1875)_{10}$

Q.13 Convert the following binary numbers to octal then to decimal. Show the steps of conversions.

i) 11011100.101010 ii) 01010011.010101 iii) 10110011

☞ [SPPU : Dec.-17, Marks 6]

Ans. :

i) $\underset{3}{\underline{011}}\ \underset{3}{\underline{011}}\ \underset{4}{\underline{100}} \cdot \underset{5}{\underline{101}}\ \underset{2}{\underline{010}}$ Binary

Octal

$(334.52)_8 = 3\times8^2 + 3\times8^1 + 4\times8^0 + 5\times8^{-1} + 2\times8^{-2}$

$= 192 + 24 + 4 + 0.625 + 0.03125 = (220.65625)_{10}$

ii) $\underset{1}{\underline{001}}\ \underset{2}{\underline{010}}\ \underset{3}{\underline{011}} \cdot \underset{2}{\underline{010}}\ \underset{5}{\underline{101}}$ Binary

Octal

$(123.25)_8 = 1\times8^2 + 2\times8^1 + 3\times8^0 + 2\times8^{-1} + 5\times8^{-2}$

$= 64 + 16 + 3 + 0.25 + 0.078125 = (83.328125)_{10}$

iii) $\underset{2}{\underline{010}}\ \underset{6}{\underline{110}}\ \underset{3}{\underline{011}}$ Binary

Octal

$(263)_8 = 2\times8^2 + 6\times8^1 + 3\times8^0 = 128 + 48 + 3 = (179)_{10}$

Q.14 Convert the following numbers in binary form :

(i) $(125.12)_{10} = (?)_2$ (ii) $(337.025)_8 = (?)_2$ iii) $(5DB.FA)_{16} = (?)_2$

☞ **[SPPU : Dec.-18, Marks 6]**

Ans. :

i) $(125.12)_{10}$ =

Integer part			Fractional part
2	125	1	$0.12 \times 2 = 0.24$
2	62	0	$0.24 \times 2 = 0.48$
2	31	1	$0.48 \times 2 = 0.96$
2	15	1	$0.96 \times 2 = 1.92$
2	7	1	$0.92 \times 2 = 1.84$
2	3	1	$0.84 \times 2 = 1.68$
2	1	1	$0.68 \times 2 = 1.36$
	0		

$\therefore (125.12)_{10} = (1111101.0001111)_2$

ii) $(337.025)_{10} = (011\ 011\ 111.000\ 010\ 101)_2$

iii) $(5DB.FA)_{18} = (0101\ 1101\ 1011.1111\ 1010)_2$

Unsolved Examples

Q.15 Convert the following numbers as indicated :

i) $(62.7)_8 = (?)_{16} = (?)_2$, ii) $(BC64)_{16} = (?)_{10} = (?)_2$ iii) $(111011)_2 = (?)_5$

Ans. : i) $(32.E)_{16}$, $(110010.111)_2$ ii) $(48228)_{10} = (1011110001100100)_2$ iii) $(214)_5$

Q.16 Convert the following decimal numbers into their equivalent hexadecimal numbers and octal numbers.

1) 936 2) 1507 3) 23.56 4) 1.025 5) 100.5

☞ **[SPPU : May-13, Marks 10]**

Ans. : 1) $(3A8)_{16}$, $(1650)_8$, 2) $(5E3)_{16}$, $(2743)_8$, 3) $(17.8F2)_{16}$, $(27.436)_8$, 4) $(1.066)_{16}$, $(1.0146)_8$, 5) $(64.8)_{16}$, $(144.4)_8$

Q.17 Convert the following hexadecimal numbers into their equivalent octal numbers and binary numbers.

1) A72E 2) BD6.7 3)0.AF54 4) DF 5) FF

☞ **[SPPU: May-13, Marks 10]**

Ans. : 1)$(123456)_8$, $(1010011100101110)_2$,

2) $(5726.34)_8$, $(101111010110.0111)_2$,

3) $(0.53652)_8$, $(0.1010111101010100)_2$,

4) $(337)_8$, $(11011111)_2$, 5) $(3FF)_8$, $(11111111)_2$]

Q.18 Convert the following numbers, show all steps :

i) $(2598.675)_{10} = (\ )_{16}$ ii) $(110101.101010)_2 = (\ )_8$

iii) $(A72E)_{16} = (\ )_8$

☞ **[SPPU : Dec.-15, Marks 6]**

Ans. : i) $(2598.675)_{10} = (A26.ACC)_{16}$

ii) $(110101.101010)_2 = (65.52)_8$

iii) $(A72E)_{16} = (1010011100101110)_2 = (123456)_8$]

2.2 : Signed Binary Number Representation - Signed and True Magnitude, 1's Complement and 2's Complement Representation

Q.19 State different ways to represent negative binary number.

Ans. : There are three ways to represent negative numbers :

- Signed - magnitude representation.
- 1's complement representation.
- 2's complement representation.

Q.20 Find 1's complement of $(11010100)_2$.

Ans. :

1	1	0	1	0	1	0	0	Number
↓	↓	↓	↓	↓	↓	↓	↓	NOT operation
0	0	1	0	1	0	1	1	1's complement of number

The 1's complement of a binary number is the number that results when we change all 1's to zeros and the zeros to ones.

Q.21 Find 2's complement of $(11000100)_2$.

Ans. : The 2's complement is the binary number that results when we add 1 to the 1's complement.

1	1	0	0	0	1	0	0	Number
					1	1		Carry
0	0	1	1	1	0	1	1	1's complement of number
+							1	Add 1
0	0	1	1	1	1	0	0	2's complement of number

Q.22 What is signed magnitude representation ? Represent + 9 and −9 using 8-bit signed - magnitude form.

OR What do you mean by sign magnitude representation ? Discuss.

Ans. : In signed magnitude representation most significant bit is used to code sign of a number (1 = negative, 0 = +) and remaining bits are used to represent magnitude of the number in the range of $+ 2^{n-1}$ to -2^{n-1}.

$+9 = 0000\ 1001$

$-9 = 1000\ 1001$

Q.23 Represent decimal number "– 13" in all three methods of negative binary number representation using eight bits.

Ans. :

1	0	0	0	1	1	0	1	– 13 in sign magnitude representation
1	1	1	1	0	0	1	0	– 13 in 1's complement representation
1	1	1	1	0	0	1	1	– 13 in 2's complement representation

2.3 : Binary Arithmetic

Q.24 Explain the binary addition operation.

Ans. :

1. Add bits column-wise starting from LSB with carry if any.
2. Put the sum at the bottom of the same column.
3. Put the carry, if any, on the top of next column.

Q.25 Add $(28)_{10}$ and $(15)_{10}$ by converting them into binary.

Ans. : Using decimal to binary conversion technique we have,

$(28)_{10} = (011100)_2$ and $(15)_{10} = (01111)_2$

		1	1	1				Carry	$(28)_{10}$
	0	0	1	1	1	0	0	Binary equivalent of $(28)_{10}$	+ $(15)_{10}$
Sign Extension →	0	0	0	1	1	1	1	Binary equivalent of $(15)_{10}$	$(43)_{10}$
Sign →	0	1	0	1	0	1	1	Result : Binary equivalent of $(43)_{10}$	

Q.26 State the procedure to perform binary subtraction using 1's complement method.

Ans. : The operation A – B is performed using 1's complement method as follows :

1. Take 1's complement of B.
2. Result ← A + 1's complement of B.
3. If carry is generated then the result is positive and in the true form. Add carry to the result to get the final result..

4. If carry is not generated then the result is negative and in the 2's complement form.

Q.27 Perform $(28)_{10} - (15)_{10}$ using 6-bit 1's complement representation.

Ans. : $(28)_{10} = (011100)_2$ $(15)_{10} = (001111)_2$

0	0	1	1	1	1	$(15)_{10}$
1	1	0	0	0	0	1's complement of $(15)_{10}$

	1	1					Carry
	0	1	1	1	0	0	Binary equivalent of $(28)_{10}$
+	1	1	0	0	0	0	1's complement of 15, i.e. $(-15)_{10}$
	1	0	0	1	1	0	Result
Sign Extension →						1	Add end around carry
+	0	0	1	1	0	1	Final result : Binary equivalent of $(13)_{10}$

$(28)_{10}$ + $(-15)_{10}$ = $(13)_{10}$

Q.28 Perform $(15)_{10} - (28)_{10}$ using 6-bit 1's complement representation.

Ans. : $(15)_{10} = (001111)_2$ $(28)_{10} = (011100)_2$

0	1	1	1	0	0	Binary equivalent of $(28)_{10}$
		1	1			Carry
1	0	0	0	1	1	1's complement of $(28)_{10}$

		1	1	1	1		Carry
	0	0	1	1	1	1	Binary equivalent of $(15)_{10}$
+	1	0	0	0	1	1	1's complement of $(28)_{10}$
	1	1	0	0	1	0	Result = Binary equivalent of $(-13)_{10}$

$(15)_{10}$ + $(-28)_{10}$ = $(-13)_{10}$

Verification		0	0	1	1	0	1	1's complement of result (Binary equivalent of $(13)_{10}$)

Q.29 State the procedure to perform binary subtraction using 2's complement method.

Ans. : The operation A – B is performed using 2's complement method as follows :

1. Take 2's complement of B.
2. Result ← A + 2's complement of B.
3. If carry is generated then the result is positive and in the true form. In this case, carry is ignored.
4. If carry is not generated then the result is negative and in the 2's complement form.

Q.30 Perform the following operations using 2's complement method.
i) $-(48)_{10} - (23)_{10}$ ii) $-(48)_{10}$ $(-23)_{10}$ ☞ [SPPU : Dec.-13, May-19, Marks 6]

Ans. : i) $-(48)_{10} = 1010000$
$-(23)_{10} = 101001$

	1							Carry
	1	0	1	0	0	0	0	$-(48)_{10}$
Sign extension → +	1	1	0	1	0	0	1	$-(23)_{10}$
	1	0	1	1	1	0	0	1	Result = $-(71)_{10}$

$\therefore$ $(48)_{10} - (23)_{10} = -(71)_{10}$

ii) $-(48)_{10} - (-23)_{10} = -(48)_{10} + (23)_{10}$
$-(48)_{10} = 1010000$
$(23)_{10} = 010111$

$\therefore$

	1						Carry
1	0	1	0	0	0	0	$-(48)_{10}$
+ 0	0	1	0	1	1	1	$(23)_{10}$
1	1	0	0	1	1	1	Result = $-(25)_{10}$

$\therefore$ $-(48)_{10} - (-23)_{10} = -(25)_{10}$

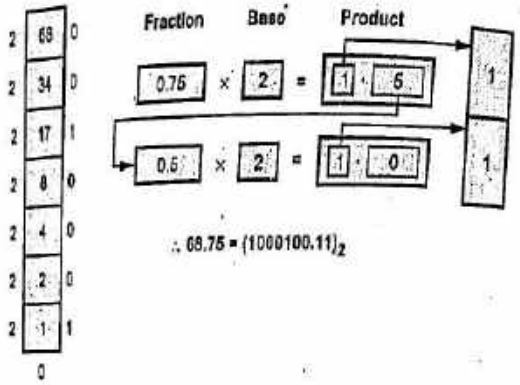
Q.31 Subtract $(27.50)_2$ from $(68.75)_2$ using 2's complement method.

☞ [SPPU : Dec.-14, May-17, Marks 4

Ans. : Step 1 : Convert numbers to binary

$0.5 \times 2 = 1.0$

$\therefore (27.50) = (11011.1)_2$



$\therefore 68.75 = (1000100.11)_2$

Step 2 : Take 2's complement of $(110111)_2$ V

0	1	1	0	1	1	·	1
1	0	0	1	0	0	·	0
+							1
1	0	0	1	0	0	·	1

2's complement

$\therefore$ $68.75 - 27.50 = (10100101)_2$

	1	1			1		1				Carry
	0	1	0	0	0	1	0	0	·	1	1	$(68.75)_{10}$
+	1	1	1	0	1	0	1	0	·	1	0	2's complement of $(27.5)_{10}$
	1	0	1	0	0	1	0	1	·	0	1	Result

Sign extension → Ignore carry

Q.32 Do $(7F)_{16} = (5C)_{16}$ using 2's complement method.

Ans. : $(7F)_{16} = (1111111)_2$ and $(5C)_{16} = (1011100)_2$

1	0	1	1	1	0	0	$(5C)_{16}$
0	1	0	0	0	1	1	1's complement
0	1	0	0	1	0	0	2's complement

	1	1	1	1	1			Carry
		1	1	1	1	1	1	$(7F)_{16}$
+	0	1	0	0	1	0	0	2's complement of $(5C)_{16}$
Discard carry	X	0	1	0	0	0	1	1	Result = (23)H

Q.33 Perform 2's complement arithmetics of :

i) $(7)_{10} - (11)_{10}$ **ii)** $(-7)_{10} - (11)_{10}$ **iii)** $(-7)_{10} + (11)_{10}$

☞ [SPPU : May-15, Marks 6]

Ans. : $(7)_{10} = (0111)_2$ $(11)_{10} = (01011)_2$

0	1	1	1	$(7)_{10}$
1	0	0	0	1's complement of 7
1	0	0	1	2's complement of 7

0	1	0	1	1	$(11)_{10}$
1	0	1	0	0	1's complement of 11
1	0	1	0	1	2's complement of 11

i) $(7)_{10} - (11)_{10}$

		1	1	1	Carry
	0	0	1	1	1	$(7)_{10}$
+	1	0	1	0	1	2's complement of 11
	1	1	1	0	0	Result is in 2's complement form

ii) $(-7)_{10} - (11)_{10}$

	1				1	Carry
Sign extension →	1	1	0	0	1	2's complement of 7
	1	0	1	0	1	2's complement of 11
	1	0	1	1	0	Result is in 2's complement form

ii) $(-7)_{10} + (11)_{10}$

	1	1		1	1	Carry
Sign extension →	1	1	0	0	1	2's complement of 7
+		0	1	0	1	1	$(11)_{10}$
Discard carry	X	0	0	1	0	0	Result = 4

Reflective codes : • A code is said to be reflective when the code for 9 is the complement for 0, the code for 8 is complement for 1, 7 for 2, 6 for 3 and 5 for 4.

• Like 2421, codes 5211 and excess-3 are also reflective. The 8421 code is not reflective.

Sequential codes : • In sequential codes each succeeding code is one binary number greater than its preceding code.

• The 8421 and excess-3 are sequential, whereas the 2421 and 5211 codes are not.

Q.36 What are error detecting and correcting codes ?

Ans. : • To maintain the data integrity between transmitter and receiver, extra bit or more than one bit are added in the data. These extra bits allow the detection and sometimes correction of error in the data.

• The data along with the extra bit/bits forms the code. Codes which allow only error detection are called **error detecting codes** and codes which allow error detection and correction are called **error detecting and correcting codes.**

Q.37 What is BCD code ? What are rules for BCD addition ?

Ans. : • BCD is an abbreviation for binary coded decimal.

• BCD is a numeric code in which each digit of a decimal number is represented by a separate group of 4-bits. The most common BCD code is 8-4-2-1 BCD.

• It is called 8-4-2-1 BCD because the weights associated with 4 bits are 8-4-2-1 from left to right. This means that, bit-3 has weight 8, bit-2 has weight 4 bit-1 has weight 2 and bit-0 has weight 1.

• The Table Q.37.1 shows the 4-bit 8-4-2-1 BCD code used to represent a single decimal digit.

• In multidigit coding, each decimal digit is individually coded with 8-4-2-1 BCD code, as shown in the Fig. Q.37.1. Total 8-bits are required to encode 58_{10} in 8-4-2-1 BCD.

Decimal	5				8			
8-4-2-1 BCD	0	1	0	1	1	0	0	0

Fig. Q.37.1

2.4 Codes : BCD, Excess-3, Gray Binary Code and Their Conversion

Q.34 Give the classification of binary codes.

Ans. : The Fig. Q.34.1 shows the classification of codes.

- The codes are broadly classified as

1. Weighted codes
2. Non-weighted codes
3. Reflective codes
4. Sequential codes
5. Alphanumeric codes
6. Error detecting and correcting codes.

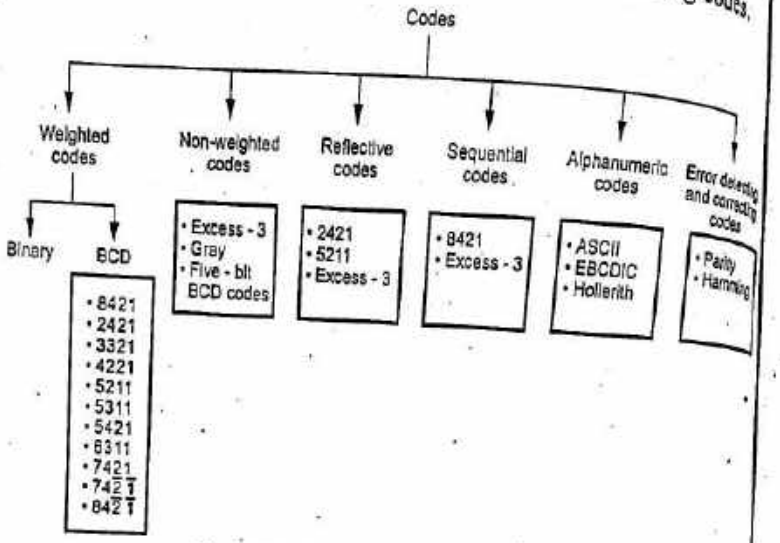


Fig. Q.34.1 Classification of codes

Q.35 Define weighted, non-weighted, reflective and sequential codes with examples.

Ans. : Weighted codes : • In weighted codes, each digit position of the number represents a specific weight.

• In weighted binary code each bit has a weight 8, 4, 2 or 1 and each decimal digit is represented by a group of four bits.

Non-weighted codes : • Non-weighted codes are not assigned with any weight to each digit position, i.e., each digit position within the number is not assigned fixed value.

• Excess-3 and gray codes are the non-weighted codes.

Reflective codes : • A code is said to be reflective when the code for 9 is the complement for 0, the code for 8 is complement for 1, 7 for 2, 6 for 3 and 5 for 4.

• Like 2421, codes 5211 and excess-3 are also reflective. The 8421 code is not reflective.

Sequential codes : • In sequential codes each succeeding code is one binary number greater than its preceding code.

• The 8421 and excess-3 are sequential, whereas the 2421 and 5211 codes are not.

Q.36 What are error detecting and correcting codes ?

Ans. : • To maintain the data integrity between transmitter and receiver, extra bit or more than one bit are added in the data. These extra bits allow the detection and sometimes correction of error in the data.

• The data along with the extra bit/bits forms the code. Codes which allow only error detection are called **error detecting codes** and codes which allow error detection and correction are called **error detecting and correcting codes.**

Q.37 What is BCD code ? What are rules for BCD addition ?

Ans. : • BCD is an abbreviation for binary coded decimal.

• BCD is a numeric code in which each digit of a decimal number is represented by a separate group of 4-bits. The most common BCD code is 8-4-2-1 BCD.

• It is called 8-4-2-1 BCD because the weights associated with 4 bits are 8-4-2-1 from left to right. This means that, bit-3 has weight 8, bit-2 has weight 4 bit-1 has weight 2 and bit-0 has weight 1.

• The Table Q.37.1 shows the 4-bit 8-4-2-1 BCD code used to represent a single decimal digit.

• In multidigit coding, each decimal digit is individually coded with 8-4-2-1 BCD code, as shown in the Fig. Q.37.1. Total 8-bits are required to encode 58_{10} in 8-4-2-1 BCD.

Decimal	5				8			
8-4-2-1 BCD	0	1	0	1	1	0	0	0

Fig. Q.37.1

Rules for BCD addition

- BCD addition rules are as follows :

1. Add two BCD numbers using ordinary binary addition.
2. If four-bit sum is equal to or less than 9, no correction is needed. The sum is in proper BCD form.

Decimal Digit	BCD code 8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Table Q.37.1 8-4-2-1 BCD code

3. If the four-bit sum is greater than 9 or if a carry is generated from the four-bit sum, the sum is invalid.
   - To correct the invalid sum, add 6 (0110_2) to the four-bit sum and ignore carry of this sum.
   - Add $1(0001)_2$ to the next higher-order BCD digit.

Q.38 Represent the unsigned decimal number 965 and 672 in BCD and then show the steps necessary to find their sum.

Ans. :

		1 1			
	1001	0110		0101	$(965)_{10}$
+	0110	0111		0010	$(672)_{10}$
	1111	1101		0111	
	0110	0110			Add 6 for correction
1	0101	0011		0111	
	1				Add carry
0001	0110	0011		0111	BCD Sum
1	6	3		7	Equivalent decimal

Q.39 What are the rules for BCD subtraction using 9's complement ?

Ans. :

Step 1 : Find the 9's complement of a negative number.

Step 2 : Add two numbers using BCD addition (Minuend + 9's complement of subtrahend).

Step 3 : If carry is not generated result is negative and find the 9's complement of the result, otherwise result is positive and add carry to the result.

Q.40 Perform the following subtractions of BCD numbers using 9's complement : i) 68 – 24 ii) 24 – 29

Ans. : i)

Step 1 : 9's complement of 24 = 99 – 24 = 75

Step 2 : Add 68 and 9's complement of 24.

	1	1						
	0	1	1	0	1	0	0	(68) BCD
+	0	1	1	1	0	1	0	(75) 9's complement of (24)BCD
	1	1	0	1	1	1	0	

1	1	1	1	1	1			
	1	1	0	1	1	1	0	Invalid BCD numbers
+	0	1	1	0	0	1	1	Add 6 in each digit
1	0	1	0	0	0	0	1	
+							1	Add end around carry
	0	1	0	0	0	1	0	
	4				4			Result

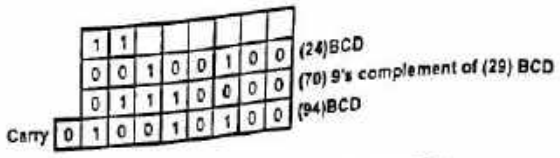
Since there is a carry, result is positive and true

$\therefore (68)_{BCD} - (24)_{BCD} = (44)_{BCD}$

II)

Step 1 : 9's complement of 29 = 99 – 29 = 70

Step 2 : Add 24 and 9's complement of 70



Since carry is zero the answer is negative.

Step 3 : Take 9's complement of answer

9's complement of 94 = 99 – 94 = (5) BCD

∴ (24) BCD – (29) BCD = – (5) BCD

Unsolved Examples

Q.41 Perform $(46)_{10} - (22)_{10}$ in BCD using 9's complement.

Q.42 Perform $(24)_{10} - (56)_{10}$ in BCD using 9's complement.

Q.43 What are the rulers for BCD subtraction using 10's complement ?

Ans. : Steps for subtraction of BCD numbers using 10's complemen method

Step 1 : Find the 10's complement of a negative number.

Step 2 : Add two numbers using BCD addition (Minuend + 10' complement of subtrahend).

Step 3 : If carry is not generated result is negative and find the 10' complement of the result, otherwise result is positive an discard carry.

Q.44 Perform $(46)_{10} - (22)_{10}$ in BCD using 10's complement.

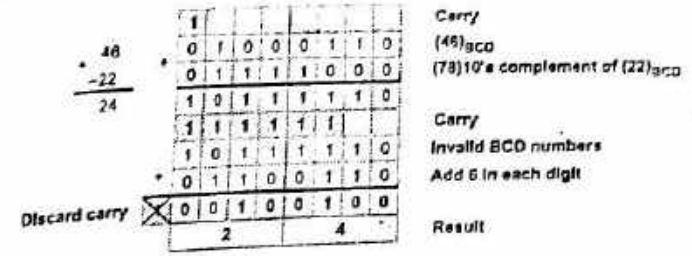
Ans. :

Step 1 : Find 10's complement of 22.

10's complement of 22 = 9's complement of 22 + 1

= (99 – 22) + 1 = 78

Step 2 : Add 46 and 10's complement of 22.



Since there is a carry the result is positive and true.

∴ $(46)_{BCD} - (22)_{BCD} = (24)_{BCD}$

Q.45 Perform $(24)_{10} - (56)_{10}$ in BCD using 10's complement.

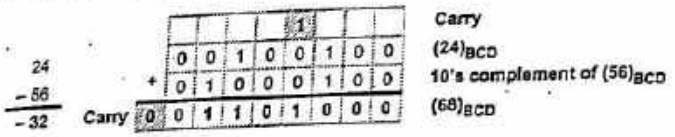
Ans. :

Step 1 : Find 10's complement of 56.

10's complement of 56 = 9's complement of 56 + 1

= (99 – 56) + 1 = 44

Step 2 : Add $(24)_{10}$ and 10's complement of 56.



Since carry is 0 the answer is negative.

Step 3 : Take 10's complement of answer.

10's complement of 68 = 9's complement of 68 + 1

= (99 – 68) + 1 = 32

∴ $(24)_{10} - (56)_{10} = -(32)_{10}$

Q.46 Find 9's and 10's complement of the following decimal numbers : (i) 24,681,234 (ii) 63,325,600

Ans. : i) 99999999 – 24681234 = 75318765 9's complement

75318765 + 1 = 75318766 10's complement

ii) 99999999 – 63325600 = 36674399 9's complement

36674399 + 1 = 36674400 10's complement

Q.47 Write a note on Excess-3 code.

Ans. : • The excess-3 code can be derived from the natural BCD code by adding 3 to each coded number.

• For example, decimal 12 can be represented in BCD as 0001 0010. Now adding 3 to each digit we get Excess-3 code as 0100 0101 (12 in decimal). It is a non-weighted code.

Decimal digit	Excess-3 code
0	0 0 1 1
1	0 1 0 0
2	0 1 0 1
3	0 1 1 0
4	0 1 1 1
5	1 0 0 0
6	1 0 0 1
7	1 0 1 0
8	1 0 1 1
9	1 1 0 0

Table Q.47.1 Excess-3 code

• Table Q.47.1 shows excess-3 codes to represent single decimal digit. It is **sequential code** because we get any code word by adding binary 1 to its previous code word as shown in the Table Q.47.1.

• In excess-3 code we get 9's complement of a number by just complementing each bit. Due to this excess-3 code is called **self-complementing code or reflective code.**

Q.48 Represent the decimal number 6 in Excess-3 code.

Ans. :

0 1 1 0	6 in BCD	
− 0 0 1 1	3	
1 0 0 1	6 in Excess-3	

Q.49 Convert the decimal number 25 into binary format, excess - 3 format and BCD format. ☞ [SPPU : May-18, Marks 3]

Ans. :

2	25	1
2	12	0
2	6	0
2	3	1
2	1	1
	0	

$\therefore \quad (25)_{10} = (11001)_2 = (0010\ 0101)_{BCD}$

$= (0101\ 1000)_{excess-3}$

Q.50 State the rules for Excess-3 addition.

Ans. : To perform excess-3 addition we have to

• Add two excess-3 numbers using binary addition.

• If carry = 1 → add 3 $(0\ 0\ 1\ 1)_2$ to the sum of two digits.

= 0 → subtract 3 $(0\ 0\ 1\ 1)_2$ from the sum.

Q.51 Perform the excess-3 addition of a) 8, 6 b) 1, 2.

Ans. :

a) 8 + 6

	1 1 1	Carry
	1 0 1 1	Excess-3 for 8
+	1 0 0 1	Excess-3 for 6
1	0 1 0 0	Carry = 1, hence
0 0 1 1	0 0 1 1	Add 3
0 1 0 0	0 1 1 1	Excess-3 for 14
1	4	Result in decimal

b) 1 + 2

	1	Carry
	0 1 0 0	Excess-3 for 1
+	0 1 0 1	Excess-3 for 2
Carry	0 1 0 0 1	Carry = 0, hence
−	0 0 1 1	Sub 3
	0 1 1 0	Excess-3 for 3
	3	Result in decimal

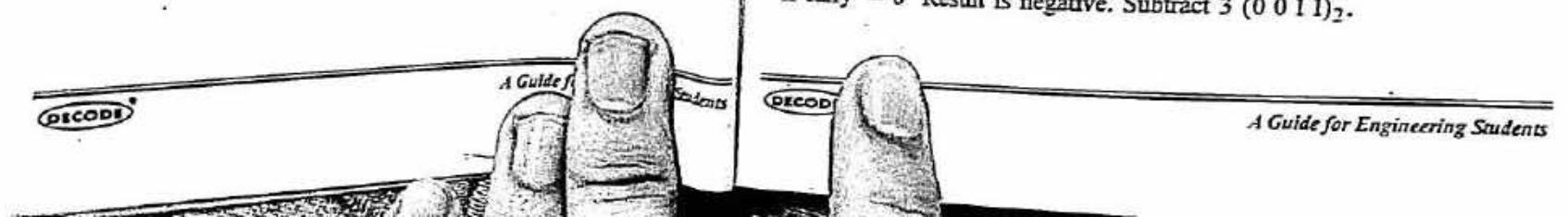
Q.52 State the rules for Excess-3 subtraction.

Ans. : To perform excess-3 subtraction we have to

• Complement the subtrahend.

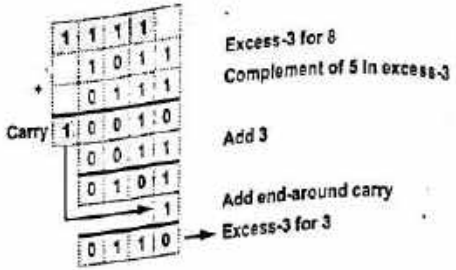
• Add complemented subtrahend to minuend.

• If carry = 1 Result is positive. Add 3 $(0\ 0\ 1\ 1)_2$ and end-around carry.
If carry = 0 Result is negative. Subtract 3 $(0\ 0\ 1\ 1)_2$.

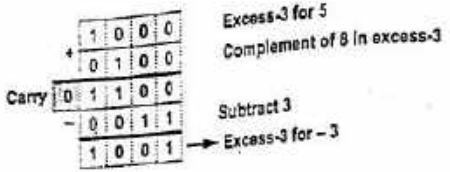


Q.53 Perform the excess-3 subtraction of a) 8 – 5, b) 5 – 8.

Ans. : a) 8 – 5



b) 5 – 8



Q.54 Write a short note on gray code.

Ans. :

- Gray code is a **non-weighted code** and is a special case of **unit-distance code.**
- In unit-distance code, bit patterns for two consecutive numbers differ in only one bit position. These codes are also called **cyclic codes.**
- The Table Q.54.1 shows the bit patterns assigned for gray code from decimal 0 to decimal 15.

Decimal code	Gray code
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

Table Q.54.1 Gray code

- As shown in the Table Q.54.1 for gray code any two adjacent code groups differ only in one bit position.
- **Reflective Property :** The gray code is also called **reflected code.** Notice that the two least significant bits for 4_{10} through 7_{10} are the mirror images of those for 0_0 through 3_{10}. Similarly, the three least significant bits for 8_{10} through 15_{10} are the mirror images of those for 0_{10} through 7_{10}.

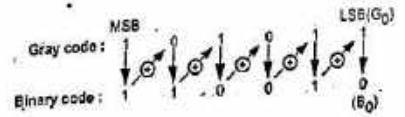
Q.55 State the rules for gray to binary code conversion.

Ans. : Steps for gray to binary code conversion

1. The Most Significant Bit (MSB) of the binary number is the same as the Most Significant Bit (MSB) of the gray code number. **So write down MSB as it is.**
2. To obtain the next binary digit, perform an exclusive-OR-operation between the bit just written down and the next gray code bit. Write down the result.
3. Repeat step 2 until all gray code bits have been exclusive-ORed with binary digits.

Q.56 Convert gray code 101011 into its binary equivalent.

Ans. :



$$(101011)_{gray} = (110010)_2$$

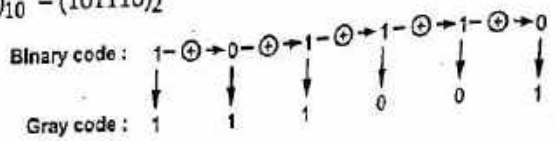
Q.57 State the rules for binary to gray code conversion.

Ans. : 1. The MSB of the gray code is the same as the MSB of the binary number. **So write down MSB as it is.**

2. To obtain the next gray digit, perform an exclusive-OR-operation between previous and current binary bit. Write down the result.
3. Repeat step 2 until all binary bits have been exclusive-ORed with their previous ones.

Q.58 Encode the decimal number 46 to Gray code.

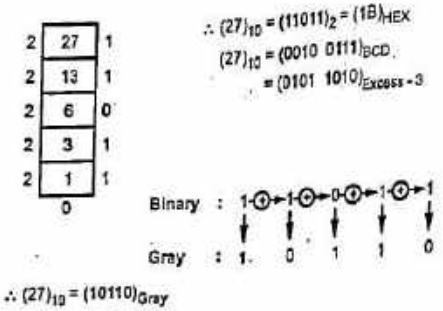
Ans. : $(46)_{10} = (101110)_2$



$\therefore \quad (46)_{10} = (111001)_{Gray}$

Q.59 Convert the decimal number 27 into : i) Binary ii) Excess-3 iii) Gray iv) HEX. ☞ [SPPU : May-16, Marks 6]

Ans. :



Unsolved Examples

Q.60 Represent the decimal number 6 in gray code.

Ans. : 0101

2.5 : IEEE Standard 754 Floating Point Number Representation

Q.61 Explain the IEEE754 standards for representing floating point numbers. ☞ [SPPU : May-13,14,16, Dec.-15]

Ans. : To accommodate very large integers and very small fractions, a computer must be able to represent numbers and operate on them in such a way that the position of the binary point is variable and is automatically adjusted as computation proceeds. In this case, the binary point is said to **float, and the numbers are called floating-point numbers.**

- The standards for representing floating point numbers in 32-bits and 64-bits have been developed by the Institute of Electrical and Electronics Engineers (IEEE), referred to as **IEEE 754 standards.** Fig. Q.61.1 shows these IEEE standard formats.

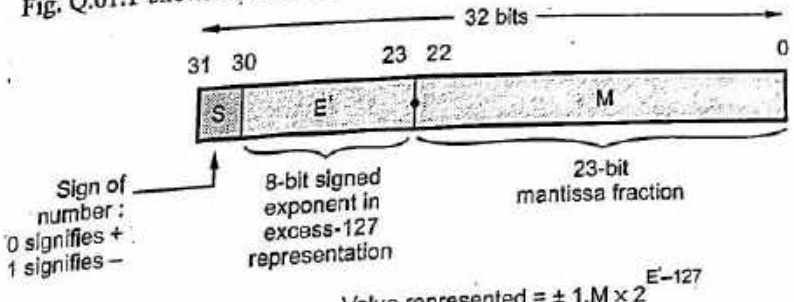


Fig. Q.61.1 (a) Single-precision

- The 32-bit standard representation shown in Fig. Q.61.1 (a) is called a **single-precision representation** because it occupies a single 32-bit word. The 32-bits are divided into three fields as shown below :
  (field 1) Sign ← 1 - bit
  (field 2) Exponent ← 8 - bits
  (field 3) Mantissa ← 23 - bits
- Instead of the signed exponent, E, the value actually stored in the exponent field is E' = E (scaling factor) + bias.
- In the 32-bit floating point system (single precision), bias is 127. Hence E' = E (scaling factor) + 127. This representation of exponent is also called as the *excess-127 format.*
- The end values of E', namely, 0 and 255, are used to indicate the floating point values of exact zero and infinity, respectively in single precision.
- Thus range of E' for normal values in single precision is $0 < E' < 255$. This means that for 32-bit representation the actual exponent E is in the range $-126 \le E \le 127$.
- The 64-bit standard representation shown in Fig. Q.61.1 (b) is called a **double-precision representation** because it occupies two 32-bit words.

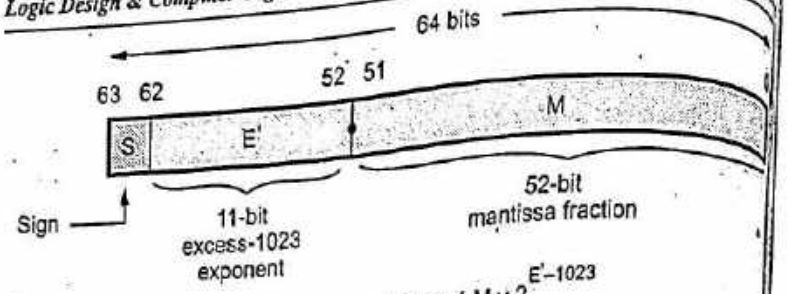


Fig. Q.61.1 (b) Double precision

The 64-bits are divided into three fields as shown below :

(field 1) Sign ← 1 - bit

(field 2) Exponent ← 11 - bit

(field 3) Mantissa ← 52 - bits

- In the double precision format value actually stored in the exponent field is given as

$$E' = E + 1023$$

- Here, bias value is 1023 and hence it is also called **excess-1023 format**.
- The end values of E' , namely, 0 and 2047, are used to indicate the floating point exact values of exact zero and infinity, respectively.
- Thus the range of E' for normal values in double precision is $0 < E' < 2047$. This means that for 64-bit representation the actual exponent E is in the range.

Normalization Process

To represent the number in floating point format, first binary point is shifted to right of the first bit and the number is multiplied by the correct scaling factor to get the same value. The number is said to be in the **normalized form** and is given as

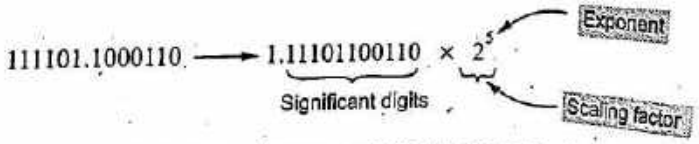


Fig. Q.61.2 Number represented in normalized form

Q.62 Represent the following in single precision floating point format as well as double precision format.

i) 17.125 ii) 12.5 [SPPU : Dec.-08, May-12, Marks 8]

Ans. : i) 17.125

Step 1 : Decimal to Binary conversion

$(17)_{10} = (10001)_2$ $0.125 \times 2 = 0.25$ 0

$\therefore$ $(17.125)_{10} = (10001.001)_2$ $0.25 \times 2 = 0.5$ 0

$0.5 \times 2 = 1.0$ 1

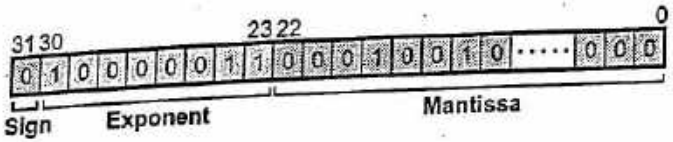
Step 2 : Normalization

$10001.001 = 1.0001001 \times 2^4$

Step 3 : Single precision representation

$S = 0$, $E = 4$ and $M = 0001001$

$E' = 127 + 4 = (131)_{10} = (10000011)_2$



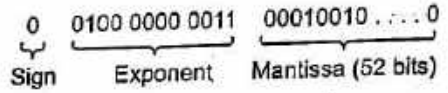
Step 4 : Double precision format

Bias for double precision format = 1023

$\therefore$ $E' = E + 1023 = 4 + 1023 = (1027)_{10} = (0100\,0000\,0011)_2$

$\therefore$ Number in double precision format is given as

ii) 12.5



Step 1 : Decimal to Binary conversion

$\therefore$ $(12)_{10} = (1100)_2$ $0.5 \times 2 = 1.0 = 1$

$\therefore$ 12.5= 1100.1

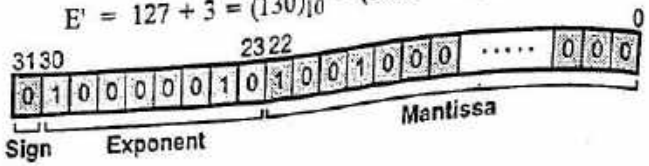
Step 2 : Normalization

$1100.1 = 1.1001 \times 2^3$

Step 3 : Single precision representation

$S = 0,\ E = 3$ and $M = 1001$

$E' = 127 + 3 = (130)_{10} = (10000010)_2$

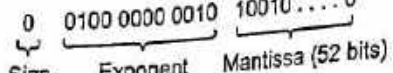


Step 4 : Double precision format

Bias for double precision format = 1023

$\therefore \quad E' = E + 1023 = 3 + 1023 = (1026)_{10} = (0100\ 0000\ 0010)_2$

$\therefore$ Number in double precision format is given as



Q.63 Represent following number in single precision format : $(0.625)_{10}$. ☞ [SPPU : May-13, Marks 10]

Ans. : Step 1 : Convert the decimal number into binary format

Fractional part :

		MSD
$0.625 \times 2 = 1.25$	1	↓
$0.25 \times 2 = 0.5$	0	
$0.5 \times 2 = 1.0$	1	↓
		LSD

$\therefore \quad (0.625)_{10} = (0.101)_2$

Step 2 : Normalize the number

$0.1010 = 1.010 \times 2^{-1}$

Step 3 : Single precision format

For the given number

$S = 0,\ E = -1,\ M = 01$

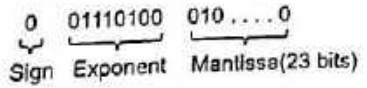
Bias for single precision format = 127

$E' = E + 127$

$= -1 + 127 = (126)_{10}$

$= (0111\ 0100)_2$

$\therefore$ Number in single precision format is given as



END...

Unit 1

3 Logic Minimization

3.1 : Review of Boolean Algebra

Postulates and Theorems

- The postulates and theorems of Boolean algebra as shown in Table 3.1.

Postulates	(a)	(b)
Postulate 1	Result of each operation is either 0 or 1 $\in$ B	
Postulate 2 (Identity)	$A + 0 = A$	$A \cdot 1 = A$
Postulate 3 (Commutative)	$A + B = B + A$	$AB = BA$
Postulate 4 (Distributive)	$A(B + C) = AB + AC$	$A + BC = (A + B)(A + C)$
Postulate 5 (Complement)	$A + \overline{A} = 1$	$A \cdot \overline{A} = 0$
Theorems	**(a)**	**(b)**
Theorem 1 (Idempotency)	$A + A = A$	$A \cdot A = A$
Theorem 2	$A + 1 = 1$	$A \cdot 0 = 0$
Theorem 3 (Involution)	$\overline{\overline{A}} = A$	
Theorem 4 (Absorption)	$A + AB = A$	$A(A + B) = A$
Theorem 5	$A + \overline{A}B = A + B$	$A \cdot (\overline{A} + B) = AB$
Theorem 6 (Associative)	$A + (B + C) = (A + B) + C$	$A(BC) = (AB)C$

Table 3.1 Postulates and basic theorems of Boolean algebra

DeMorgan's Theorems

1. $\overline{AB} = \overline{A} + \overline{B}$
2. $\overline{A + B} = \overline{A} \cdot \overline{B}$

Principle of Duality

- The principle of duality theorem says that, starting with a Boolean relation, you can derive another Boolean relation by
  1. Changing each OR sign to an AND sign
  2. Changing each AND sign to an OR sign and
  3. Complementing any 0 or 1 appearing in the expression.

For example : Dual of relation $A + \overline{A} = 1$ is $A \cdot \overline{A} = 0$

Q.1 Prove that $A + \overline{A}B = A + B$

Ans. :

$A + \overline{A}B = A + AB + \overline{A}B$ by Theorem : 4(a)

$= A + B \cdot (A + \overline{A})$ by Postulate : 4(a)

$= A + B \cdot 1$: 5(a)

$= A + B$: 2(b)

Q.2 Prove the following equations using Boolean algebra.

i) $XY + XYZ + XY\overline{Z} + \overline{X}YZ = Y(X + Z)$

ii) $\overline{A}BC\overline{D} + BC\overline{D} + B\overline{C}\,\overline{D} + B\overline{C}D = B(\overline{D} + \overline{C})$

[SPPU : Dec.-11, June-11, Marks 6]

Ans. : i) $xy + xyz + xy\overline{z} + \overline{x}yz = xy(1 + z + \overline{z}) + \overline{x}yz = xy + \overline{x}yz$

$= y(x + \overline{x}z) = y(x + z)$

$\because A + \overline{A}B = A + B$ ... Proved

ii) $\overline{A}BC\overline{D} + BC\overline{D} + B\overline{C}\,\overline{D} + B\overline{C}D = BC\overline{D}(\overline{A} + 1) + B\overline{C}(\overline{D} + D)$

$= BC\overline{D} + B\overline{C}$ $\because \overline{A} + 1 = 1$ and $\overline{A} + A = 1$

$= B(\overline{C} + C\overline{D}) = B(\overline{C} + \overline{D})$ $\because A + \overline{A}B = A + B$

... Proved

Q.3 Using Boolean algebra show that :

i) $\overline{A}BC\overline{D} + BC\overline{D} + B\overline{C}\,\overline{D} + B\overline{C}D = B(\overline{D} + \overline{C})$

ii) $AB + \overline{AC} + A\overline{B}C(AB + C) = 1$

iii) $\overline{A}B\overline{C}D + \overline{A}BCD + ABD = BD$. [SPPU : Dec.11, Marks 10]

Ans. : i) Refer Q. 2 (ii).

ii) $AB + \overline{AC} + A\overline{B}C(AB + C) = AB + \overline{A} + \overline{C} + AAB\overline{B}C + A\overline{B}CC$

$\because \overline{AC} = \overline{A} + \overline{C}$

$= AB + \overline{A} + \overline{C} + A\overline{B}C$ $\because A\overline{A} = 0$

$= \overline{A} + AB + \overline{C} + \overline{A} + A\overline{B}C$ $\because A = A + A$

$= \overline{A} + B + \overline{C} + \overline{A} + \overline{B}C$ $\because A + \overline{A}B = A + B$

$= \overline{A} + B + \overline{C} + \overline{B}C$

$= \overline{A} + B + \overline{C} + \overline{B}$ $\because A + \overline{A}B = A + B$

$= 1$ $\because B + \overline{B} = 1$ ... Proved

iii) $\overline{A}B\overline{C}D + \overline{A}BCD + ABD = \overline{A}BD(\overline{C} + C) + ABD$

$= \overline{A}BD + ABD$ $\because A + \overline{A} = 1$

$= BC(\overline{A} + A)$

$= BC$ $\because A + \overline{A} = 1$ ... Proved

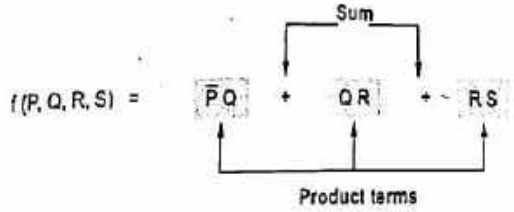
3.2 : Representation of Logic Functions

Concepts

- Each occurrence of a variable in either a complemented or an uncomplemented form in the Boolean function is called a **literal**.
- A **product term** is defined as either a literal or a product (also called conjunction) of literals.
- A **sum term** is defined as either a literal or a sum (also called disjunction) of literals.
- Literals and terms are arranged in one of the two forms :
  - Sum Of Product form (SOP) and
  - Product Of Sum form (POS)

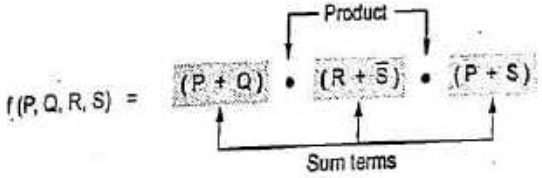
- A Sum Of Products (SOP) is a group of product terms ORed together.

Example



- A product of sums is any groups of sum terms ANDed together.

Example



- If each term in SOP form contains all the literals then the SOP form is known as **canonical SOP form**.
- Each individual term in the canonical SOP form is called **minterm**.
- If each term in POS form contains all the literals then the POS form is known as **canonical POS form**.
- Each individual term in the canonical POS form is called **maxterm**.
- The POS and SOP functions derived from the same truth table are logically equivalent.
- In terms of minterms and maxterms we can then write

$f(A, B, C) = m_0 + m_1 + m_3 + m_4 + m_6 + m_7 = M_2 \cdot M_5$

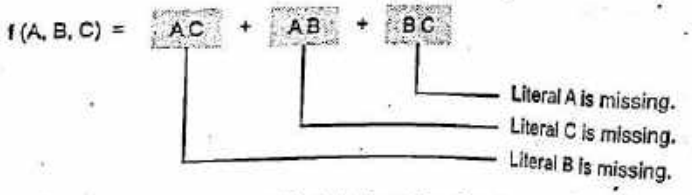
$\therefore \quad f(A, B, C) = \sum m(0, 1, 3, 4, 6, 7) = \pi M(2, 5)$

Variables			Minterms	Maxterms
A	B	C	m_i	M_i
0	0	0	$\bar{A}\bar{B}\bar{C} = m_0$	$A + B + C = M_0$
0	0	1	$\bar{A}\bar{B}C = m_1$	$A + B + \bar{C} = M_1$
0	1	0	$\bar{A}B\bar{C} = m_2$	$A + \bar{B} + C = M_2$
0	1	1	$\bar{A}BC = m_3$	$A + \bar{B} + \bar{C} = M_3$
1	0	0	$A\bar{B}\bar{C} = m_4$	$\bar{A} + B + C = M_4$
1	0	1	$A\bar{B}C = m_5$	$\bar{A} + B + \bar{C} = M_5$
1	1	0	$AB\bar{C} = m_6$	$\bar{A} + \bar{B} + C = M_6$
1	1	1	$ABC = m_7$	$\bar{A} + \bar{B} + \bar{C} = M_7$

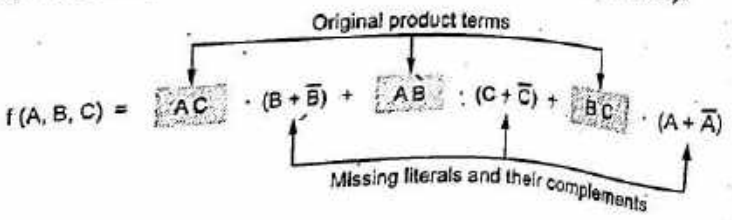
Table 3.2 Minterms and maxterms for three variables

Q.4 Convert the given expression in canonical SOP form.
f (A, B, C) = AC + AB + BC

Ans. :
Step 1 : Find the missing literal/s in each product term.



Step 2 : AND product term with (missing literal + its complement).



Step 3 : Expand the terms and reorder literals.
Expand : $f(A, B, C) = ACB + AC\bar{B} + ABC + AB\bar{C} + BCA + BC\bar{A}$
Reorder : $f(A, B, C) = ABC + A\bar{B}C + ABC + AB\bar{C} + ABC + \bar{A}BC$

Note *After having sufficient practice student should expand product term and reorder literals in it in a single step.*

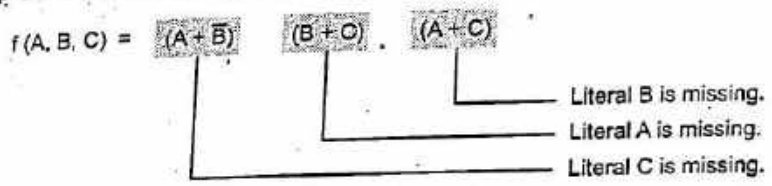
Step 4 : Omit repeated product terms.

$$f(A, B, C) = ABC + A\bar{B}C + \boxed{ABC} + AB\bar{C} + \boxed{ABC} + \bar{A}BC$$

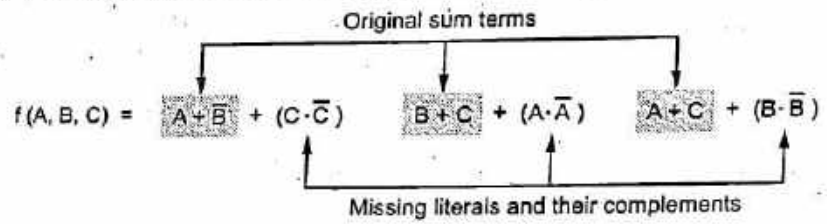
$$\therefore \quad f(A, B, C) = ABC + A\bar{B}C + AB\bar{C} + \bar{A}BC$$

Q.5 Convert the given expression in canonical POS form.
f (A, B, C) = (A + B̄) (B + C) (A + C)

Ans. :
Step 1 : Find the missing literal/s in each sum term.



Step 2 : OR sum term with (missing literal · its complement).



Step 3 : Expand the terms and reorder literals.

Expand :

Since $A + BC = (A + B)(A + C)$ we have,

$$f(A, B, C) = (A + \bar{B} + C)(A + \bar{B} + \bar{C})(B + C + A)(B + C + \bar{A})$$
$$(A + C + B)(A + C + \bar{B})$$

Reorder :

$$f(A, B, C) = (A + \overline{B} + C)\,(A + \overline{B} + \overline{C})\,(A + B + C)\,(\overline{A} + B + C)\,(A + B + C)\,(A + \overline{B} + C)$$

Step 4 : Omit repeated sum terms.

Repeated sum terms

$$(A, B, C) = (A + \overline{B} + C)\,(A + \overline{B} + \overline{C})\,(A + B + C)\,(\overline{A} + B + C)\,(A + B + C)\,(A + \overline{B} + C)$$

$$\therefore f(A, B, C) = (A + \overline{B} + C)\,(A + \overline{B} + \overline{C})\,(A + B + C)\,(\overline{A} + B + C)$$

3.3 : Simplification of Logical Functions, using K-Maps up to 4 Variables

Q.6 What is K-map ? How to represent truth table on K-map ?

Ans. : • K-map (Karnaugh map) is a graphical way to represent a truth table and it is a method by which we can simplify boolean expressions.

- Each Karnaugh map consists of 2^n cells - where n represents number of input variables. These cells contain the state of the output for corresponding states of input variables.
- Fig. Q.6.1 (a), (b) and (c) shows representation of 2-variable, 3-variable and 4-variable truth tables on K-maps, respectively.
- A Boolean expression in the sum of products form can be plotted on the Karnaugh map by placing a 1 in each cell corresponding to a term **(minterm)** in the sum of products expression. Remaining cells are filled with zeros.
- A Boolean expression in the product of sums can be plotted on the Karnaugh map by placing a 0 in each cell corresponding to a term **(maxterm)** in the expression. Remaining cells are filled with ones.

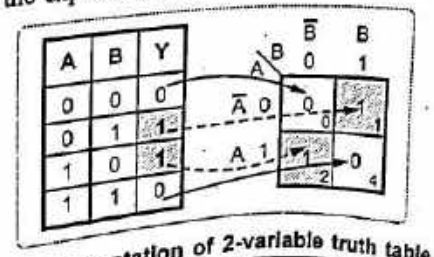


Fig. Q.6.1 (a) Representation of 2-variable truth table on K-map

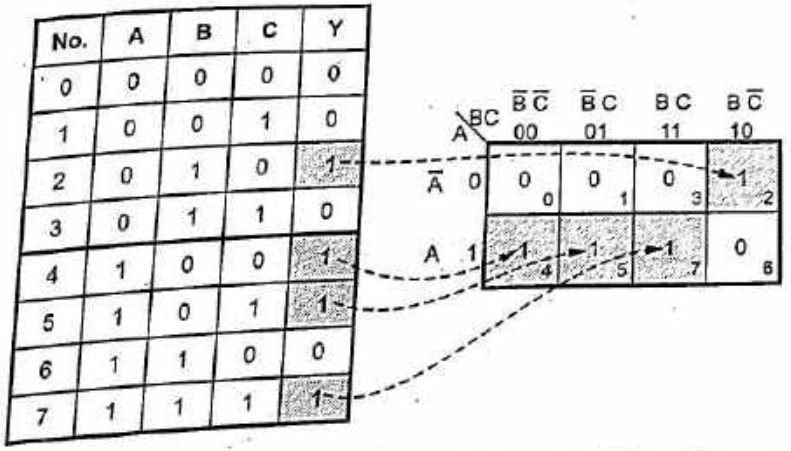
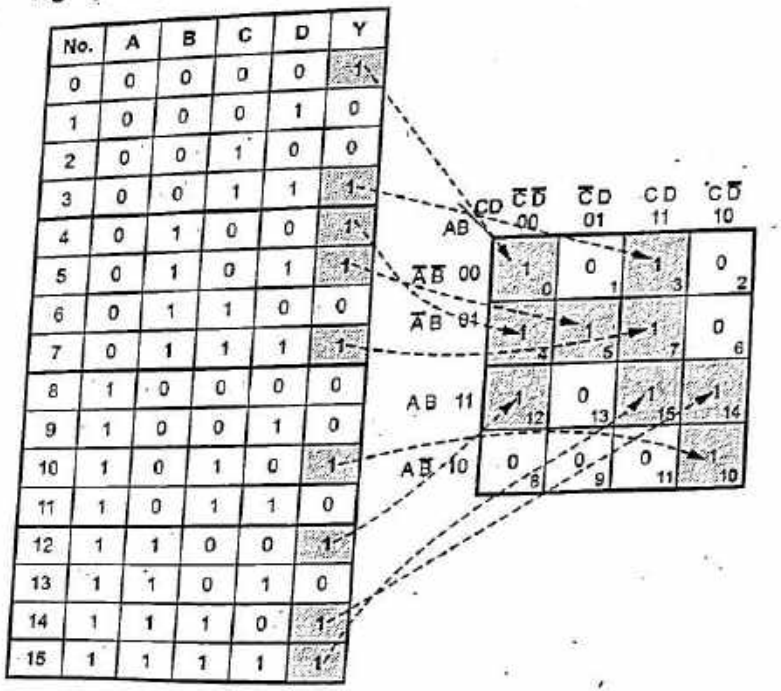


Fig. Q.6.1 (b) Representation of 3-variable truth table on K-map



(c) Representation of 4-variable truth table on K-map

Fig. Q.6.1 Plotting truth table on K-map

Q.7 Plot Boolean expression $Y = A\bar{B}\bar{C} + ABC + \bar{A}\bar{B}C$ on the Karnaugh map.

Ans. : The expression has 3-variables and hence it can be plotted using 3-variable as in Fig. Q.7.1.

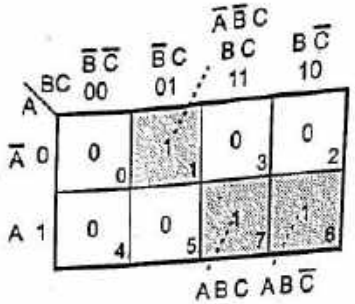


Fig. Q.7.1

Q.8 Plot Boolean expression.

$Y = (A+B+C+\bar{D})\ (A+\bar{B}+\bar{C}+D)\ (A+B+\bar{C}+\bar{D})$

$(\bar{A}+\bar{B}+C+\bar{D})\ (\bar{A}+\bar{B}+\bar{C}+D)$

Ans. : The expression has 4-variables and hence it can be plotted using 4-variable map as shown in Fig. Q.8.1.

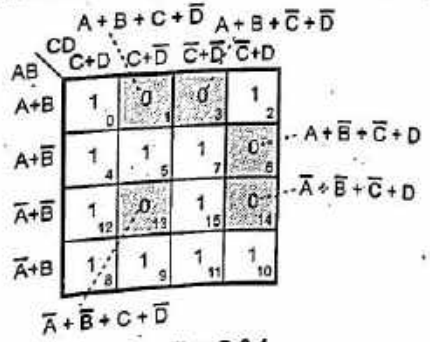


Fig. Q.8.1

$(A + B + C + \bar{D}) = M_1$, $(A + \bar{B} + \bar{C} + D) = M_6$, $(A + B + \bar{C} + \bar{D}) = M_3$,

$(\bar{A} + \bar{B} + C + \bar{D}) = M_{13}$, $(\bar{A} + \bar{B} + \bar{C} + D) = M_{14}$

Q.9 What is a pair in K-map ? How is it useful in simplification of Boolean expression ?

Ans. : • A pair is a group of two adjacent cells in a Karnaugh map. It cancels one variable in a K-map simplification.

• Fig. Q.9.1 (a) shows A pair of 1s that are horizontally adjacent to each other; the first represents $\bar{A}\bar{B}C$ and the second represents $\bar{A}BC$. Note that in these two terms only the B variable appears in both normal and complemented form ($\bar{A}$ and C remain unchanged). Thus the expression reduces to $\bar{A}C$.

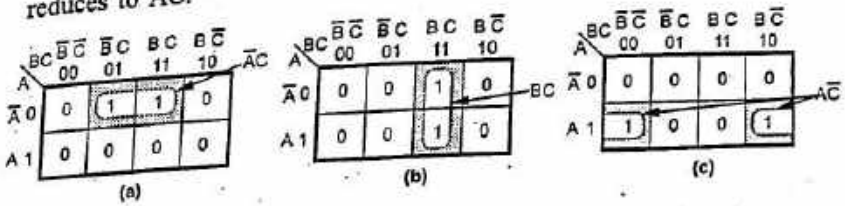


Fig. Q.9.1

• Fig. Q.9.1 (b) shows an example of two vertically adjacent 1s. These two can be combined to eliminate A variable since it appears in both its uncomplemented and complemented forms. Thus the expression reduces to BC.

• In a Karnaugh map the corresponding cells in the leftmost column and rightmost column are considered to be adjacent. Thus, the two 1s in these columns with a common row can be combined to eliminate one variable. This is illustrated in Fig. Q.9.1 (c).

• Here variable B has appeared in both its complemented and uncomplemented forms and hence eliminated to give expression as $A\bar{C}$

• Same pairing rules can be applied for 2 or 4 variable K-maps. In 4-variable K-maps, the corresponding cells in the top row and bottom row are considered to be adjacent.

Q.10 What is quad in K-map ? How is it useful in simplification of Boolean expression ?

Ans. : • Quad is a group of four adjacent cells in a Karnaugh map. It cancels two variables in a K-map simplification.

• Fig. Q.10.1 shows several examples of quad

Q.7 Plot Boolean expression $Y = A\bar{B}\bar{C} + ABC + \bar{A}\bar{B}C$ on the Karnaugh map.

Ans. : The expression has 3-variables and hence it can be plotted using 3-variable as in Fig. Q.7.1.

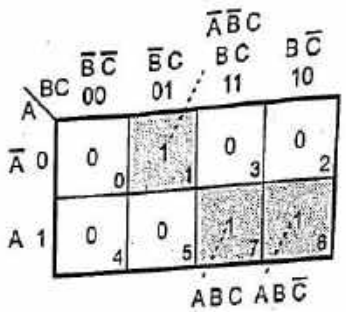


Fig. Q.7.1

Q.8 Plot Boolean expression.

$Y = (A+B+C+\bar{D})\ (A+\bar{B}+\bar{C}+D)\ (A+B+\bar{C}+\bar{D})$
$(\bar{A}+\bar{B}+C+\bar{D})\ (\bar{A}+\bar{B}+\bar{C}+D)$

Ans. : The expression has 4-variables and hence it can be plotted using 4-variable map as shown in Fig. Q.8.1.

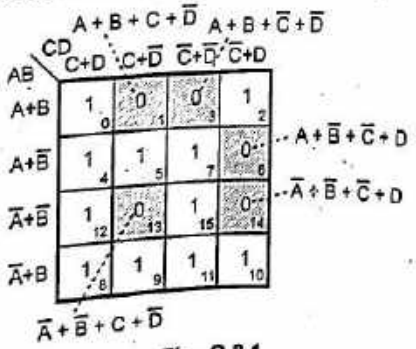


Fig. Q.8.1

$(A + B + C + \bar{D}) = M_1$, $(A + \bar{B} + \bar{C} + D) = M_6$, $(A+B+\bar{C}+\bar{D}) = M_3$,
$(\bar{A} + \bar{B} + C + \bar{D}) = M_{13}$, $(\bar{A} + \bar{B} + \bar{C} + D) = M_{14}$

Q.9 What is a pair in K-map ? How is it useful in simplification of Boolean expression ?

Ans. : • A pair is a group of two adjacent cells in a Karnaugh map. It cancels one variable in a K-map simplification.

• Fig. Q.9.1 (a) shows A pair of 1s that are horizontally adjacent to each other; the first represents $A\bar{B}C$ and the second represents $\bar{A}BC$. Note that in these two terms only the B variable appears in both normal and complemented form ($\bar{A}$ and C remain unchanged). Thus the expression reduces to $\bar{A}C$.

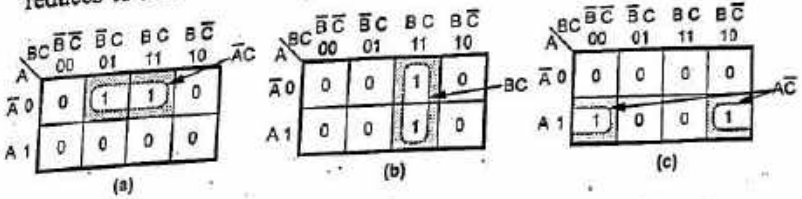


Fig. Q.9.1

• Fig. Q.9.1 (b) shows an example of two vertically adjacent 1s. These two can be combined to eliminate A variable since it appears in both its uncomplemented and complemented forms. Thus the expression reduces to BC.

• In a Karnaugh map the corresponding cells in the leftmost column and rightmost column are considered to be adjacent. Thus, the two 1s in these columns with a common row can be combined to eliminate one variable. This is illustrated in Fig. Q.9.1 (c).

• Here variable B has appeared in both its complemented and uncomplemented forms and hence eliminated to give expression as $A\bar{C}$

• Same pairing rules can be applied for 2 or 4 variable K-maps. In 4-variable K-maps, the corresponding cells in the top row and bottom row are considered to be adjacent.

Q.10 What is quad in K-map ? How is it useful in simplification of Boolean expression ?

Ans. : • Quad is a group of four adjacent cells in a Karnaugh map. It cancels two variables in a K-map simplification.

• Fig. Q.10.1 shows several examples of quad

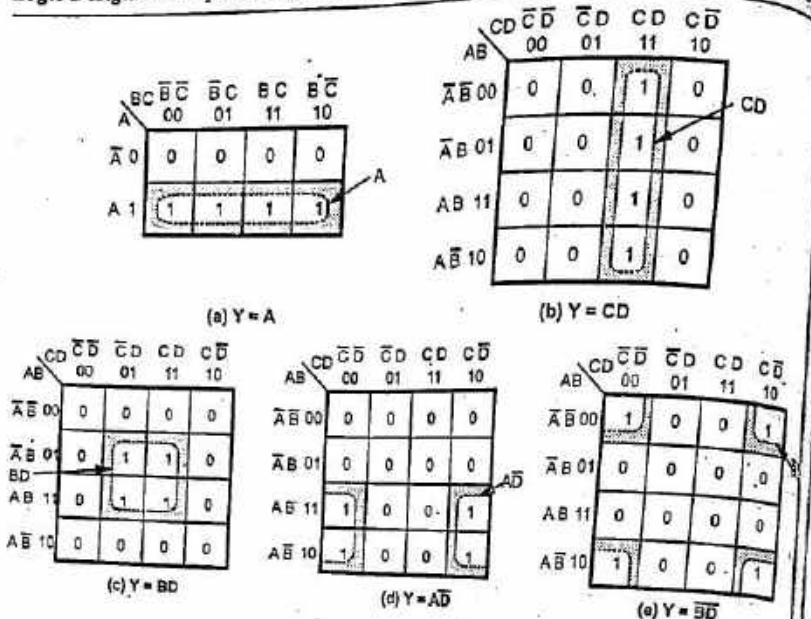


Fig. Q.10.1

Q.11 What is octet ? How is it useful in simplification of Boolean expression ?

Ans. : Octet is a group of eight adjacent cells in a Karnaugh map. It cancels three variables in a K-map simplifications.

Fig. Q.11.1 shows several examples of octet. (Refer Fig. Q.11.1 on next page)

Q.12 State the rules for simplifying logic function using K-map.

Ans. : Rules for simplifying logic function using K-map are :

1. Group should not include any cell containing a zero.
2. The number of cells in a group must be a power of 2, such as 1, 2, 4, 8 or 16.
3. Group may be horizontal, vertical but not diagonal.
4. Cell containing 1 must be included in at least one group.
5. Groups may overlap.
6. Each group should be as large as possible to get maximum simplification.
7. Groups may be wrapped around the map. The leftmost cell in a row may be grouped with the rightmost cell and the top cell in a column may be grouped with the bottom cell.

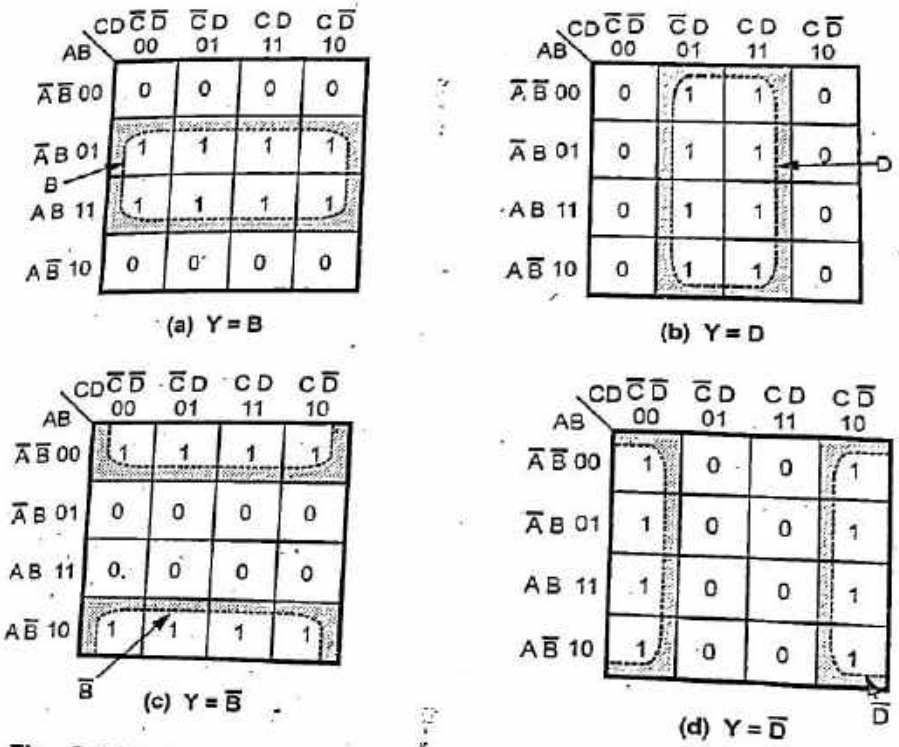


Fig. Q.11.1 Examples of combining octets of adjacent ones

8. A cell may be grouped more than once. The only condition is that every group must have at least one cell that does not belong to any other group. Otherwise, redundant terms will result.
9. We need not group all don't care cells, only those that actually contribute to a maximum simplification.
10. All above rules are stated considering the SOP simplification. In case of POS simplification all rules are same except 0 (zero) takes place of 1 (one).

Q.13 What do you mean by don't care conditions ?

Ans. :

- In some logic circuits, certain input conditions never occur, therefore the corresponding output never appears. In such cases the output level is not defined, it can be either HIGH or LOW. These output levels are indicated by 'X' or 'd' in the truth tables and are called **don't care outputs** or **don't care conditions** or **incompletely specified functi**

- A circuit designer is free to make the output for any "don't care" condition either a '0' or a '1' in order to produce the simplest output expression.

Q.14 Simplify the following function

$f_1 (A, B, C, D) = \sum m (0, 3, 5, 6, 9, 10, 12, 15)$ [SPPU : May-13, Marks 4]

Ans. :

$$f_1 = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}CD + \overline{A}B\overline{C}\,D + \overline{A}BCD + AB\overline{C}\,\overline{D} + ABCD + A\overline{B}\overline{C}D + A\overline{B}C\overline{D}$$

$$= \overline{A}\,\overline{B}\,(\overline{C}\,\overline{D} + CD) + \overline{A}B\,(\overline{C}D + C\overline{D}) + AB(\overline{C}\,\overline{D} + CD) + A\overline{B}\,(\overline{C}D + C\overline{D})$$

$$= \overline{A}\,\overline{B}\,(\overline{C \oplus D}) + \overline{A}B\,(C \oplus D) + AB\,(\overline{C \oplus D}) + A\overline{B}(C \oplus D)$$

$$= (\overline{A}\,\overline{B} + AB)\,(\overline{C \oplus D}) + (\overline{A}B + A\overline{B})\,(C \oplus D)$$

$$= (\overline{A \oplus B})\,(\overline{C \oplus D}) + (A \oplus B)(C \oplus D)$$

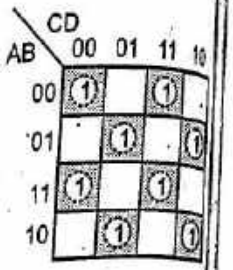


Fig. Q.14.1

Substituting $A \oplus B = A$ and $C \oplus D = B$ we have

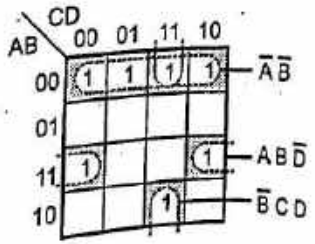
$$= \overline{A}\,\overline{B} + AB = \overline{A \oplus B} = A \cdot B$$

$$\therefore \quad f_1 = (A \oplus B) \odot (C \oplus D)$$

Q.15 Simplify the following function

$f_3 (A, B, C, D) = \sum m (0, 1, 2, 3, 11, 12, 14)$ [SPPU : May-13, Marks 4]

Ans. :



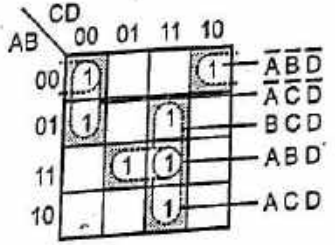
$\therefore f_3 = \overline{A}\,\overline{B} + AB\overline{D} + \overline{B}CD$

Fig. Q.15.1

Q.16 Solve the following using minimization technique

$Z = f(A, B, C, D) = \sum (0, 2, 4, 7, 11, 13, 15)$ [SPPU : Dec.-09, Marks 5]

Ans. :



$\therefore Z = \overline{A}\,\overline{B}\,\overline{D} + \overline{A}\,\overline{C}\,\overline{D} + BCD + AB\overline{D} + AC\overline{D}$

Fig. Q.16.1

Q.17 Using K-map convert the following standard POS expression into a minimum POS expression, a standard SOP expression and minimum SOP expression

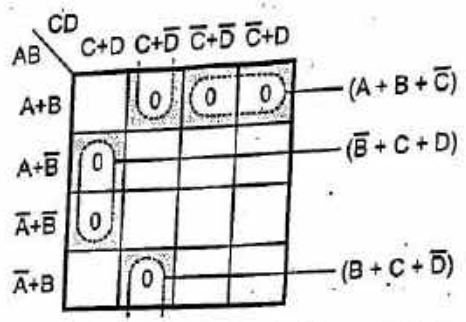
(A'+ B'+C+D) (A+B'+C+D) (A+B+C+D')
(A+B+C'+D') (A'+B+C+D') (A+B+C'+D). [SPPU : Dec.-15, Marks 6]

Ans. : $(\overline{A} + \overline{B} + C + \overline{D})\,(A + \overline{B} + C + D)(A + B + C + \overline{D})(A + B + \overline{C} + \overline{D})$

$(\overline{A} + B + C + \overline{D})(A + B + \overline{C} + D) = \pi M(12, 4, 1, 3, 9, 2)$

$= \pi M(1, 2, 3, 4, 9, 12)$

Minimum POS Expression using K-map



$\therefore F(A, B, C, D) = (A + B + \overline{C})\,(\overline{B} + C + D)\,(B + C + \overline{D})$

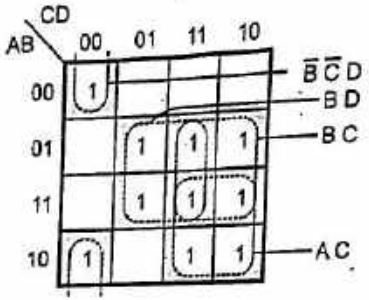
Standard SOP Expression

$\pi M(1, 2, 3, 4, 9, 12) = \Sigma m(0, 5, 6, 7, 8, 10, 11, 13, 14, 15)$

$= \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}\overline{C}D + \overline{A}B\overline{C}\overline{D} + \overline{A}BCD + A\overline{B}\overline{C}\overline{D}$

$+ A\overline{B}C\overline{D} + A\overline{B}CD + AB\overline{C}D + ABC\overline{D} + ABCD$

Minimum SOP Expression



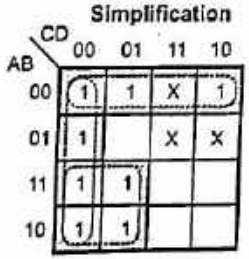
$\therefore F(A, B, C, D) = \overline{B}\,\overline{C}\,D + BD + BC + AC$

Q.18 Minimize the following function using K-map and implement using basic logic gates $f(A,B,C,D) = \Sigma\, m(0,1,2,4,8,9,12,13) + d(3,6,7)$.

☞ [SPPU : May-14, Marks 6]

Ans. :

Simplification



$F = \overline{A}\,\overline{B} + \overline{C}\,\overline{D} + A\overline{C}$

Implementation

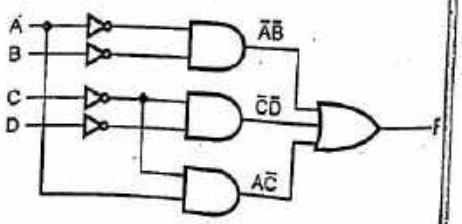


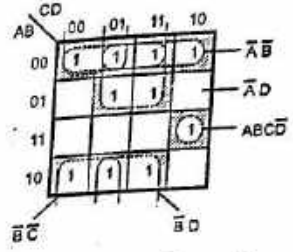
Fig. Q.18.1

Q.19 Minimize the following equation using K-map and realize it using NAND gates only.

$y = \sum m(0, 1, 2, 3, 5, 7, 8, 9, 11, 14)$ ☞ [SPPU : Dec.-11, Marks 10]

Ans. :

K-map simplification



$\therefore F = \overline{B}\,\overline{C} + \overline{B}D + \overline{A}\,\overline{B} + \overline{A}D + ABC\overline{D}$

Implementation

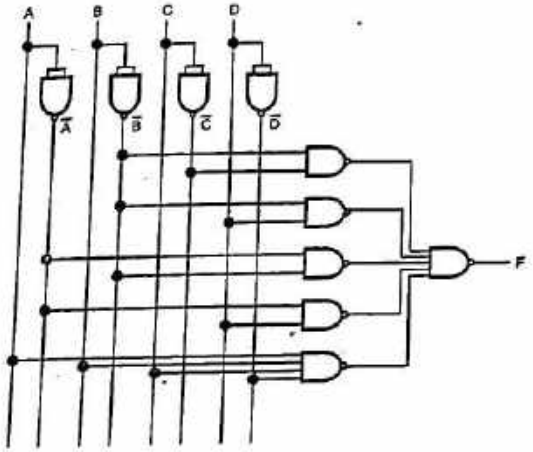
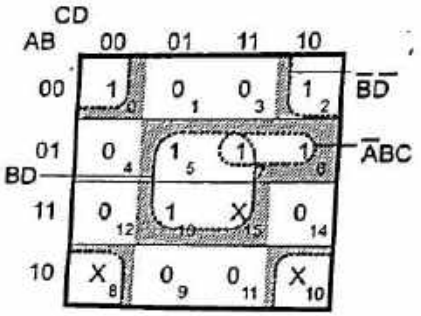


Fig. Q.19.1

Q.20 Minimize $f(A, B, C, D) = \sum m\,(0, 2, 5, 6, 7, 13) + d\,(8, 10, 15)$ Implement using NAND gates. ☞ [SPPU : May-07, Marks 8]

Ans. :

K-map simplification :



$f(A, B, C, D) = \overline{B}\,\overline{D} + BD + \overline{A}\,BC$

Implementation : **Implementation using NAND gates :**

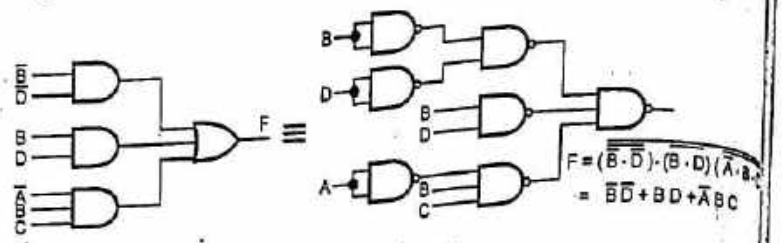


Fig. Q.20.1

Q.21 Minimize function using K-map and implement using NOR gate : f (A, B, C, D) = π M (1, 4, 5, 6, 7, 8, 12) + d (3, 9, 11, 14)

☞ [SPPU : Dec.-07, Marks 6]

Ans. : K-map simplification :

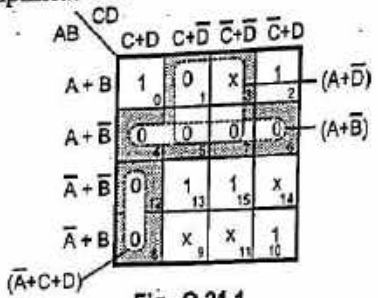


Fig. Q.21.1

$\therefore$ f (ABCD) = (A + $\overline{D}$) (A + $\overline{B}$) ($\overline{A}$ + C + D)

Implementation using basic gates **Implementation using only NOR gate**

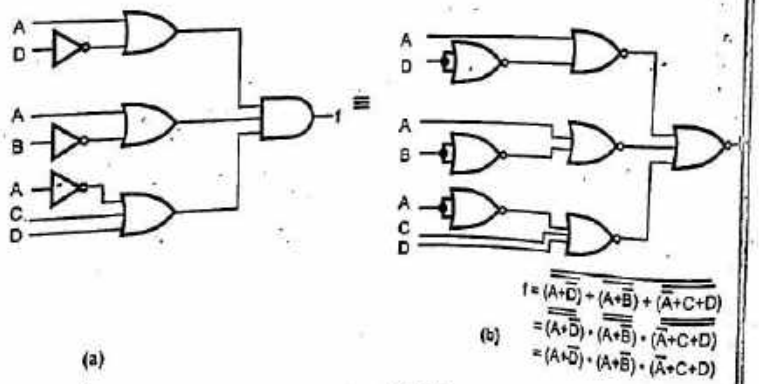


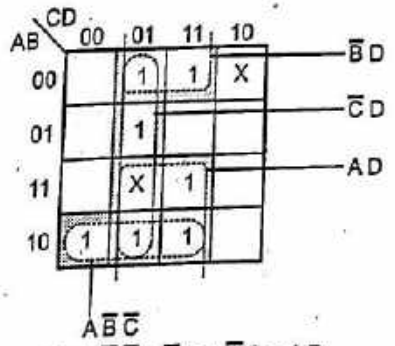
Fig. Q.21.2

Q.22 Minimize the following function using K-map and implement using basic logic gates.

$F(A, B, C, D) = \sum m(1, 3, 5, 8, 9, 11, 15) + d(2, 13)$.

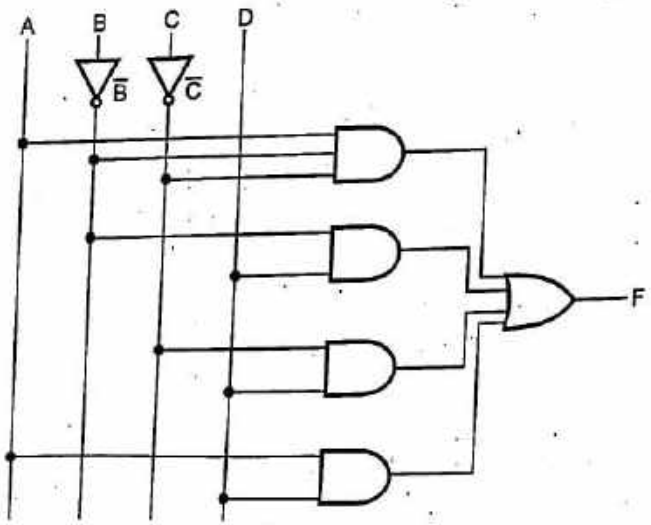
☞ [SPPU : Dec.-17, Marks 6]

Ans. : K-map Simplification :



$\therefore F = A\overline{B}\,\overline{C} + \overline{B}D + \overline{C}D + AD$

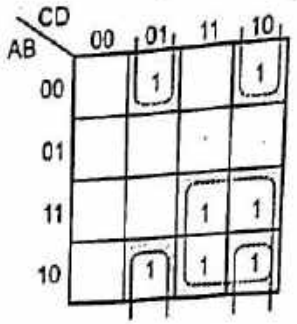
Implementation :



Q.23 Use K-map minimization technique to realize following expression using minimum number of gates.

$$Y = \sum m(1,2,9,10,11,14,15).$$ [SPPU : May-19, Marks 8]

Ans. :



$Y = \overline{B}\,\overline{C}\,D + \overline{B}\,C\,\overline{D} + AC$

Fig. Q.23.1

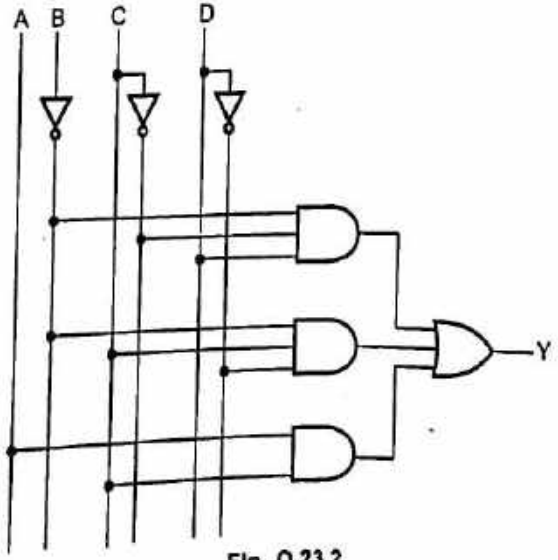


Fig. Q.23.2

END...

Unit 2

4 Design using SSI Chips

4.1 : Code Converters

Q.1 State the procedure to design code converters.

Ans. : Step 1 : Write the truth table showing the relationship between input code and output code.

Step 2 : For each output code bit determine the simplified Boolean expression using K-map.

Step 3 : Realize the code converter using logic gates.

Q.2 Design a logic circuit to convert the 8421 BCD to Excess-3 code. [SPPU : Dec.-12,13, Marks 8]

Ans. : Step 1 : Form the truth table relating BCD and Excess-3 code

Excess-3 code is a modified form of a BCD number. The Excess-3 code can be derived from the natural BCD code by adding 3 to each coded number. For example, decimal 12 can be represented in BCD as 0001 0010. Now adding 3 to each digit we get Excess-3 code as 0100 0101 (12 in decimal). With this information the truth table for BCD to Excess-3 code converter can be determined as shown in Table Q.2.1.

Decimal	D_3	D_2	D_1	D_0	E_3	E_2	E_1	E_0
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1

5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

Table Q.2.1

Input code : BCD code : $D_3\ D_2\ D_1\ D_0$ (D_0 LSB)

Output code : Excess-3 code : $E_3\ E_2\ E_1\ E_0$ (E_0 LSB)

Step 2 : K-map simplification for each Excess-3 code output.

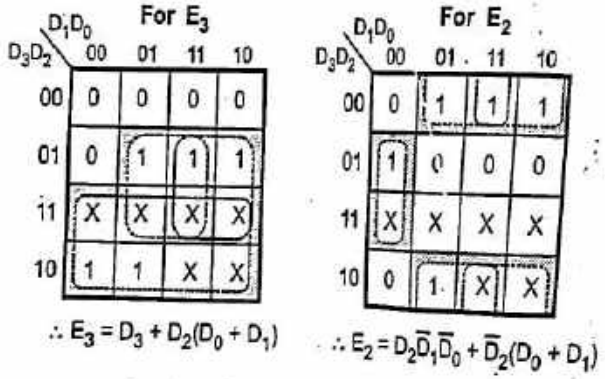


Fig. Q.2.1

Step 3 : Realization of code converter.

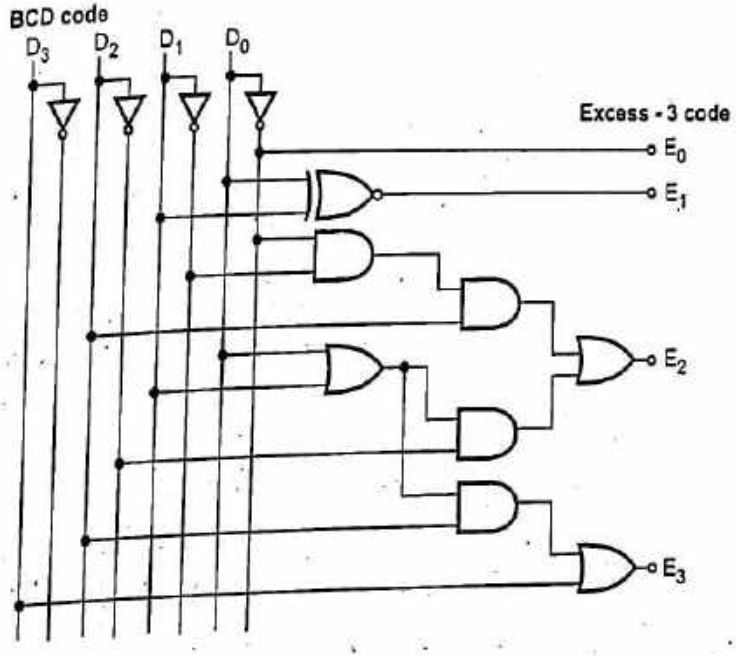


Fig. Q.2.2 BCD to excess-3 code converter

Q.3 Design and implement a 8421 to Gray code converter. Realize the converter using only NAND gates. [SPPU : May-12, Marks 4]

Ans. : Step 1 : Form the Truth table relating 8421 binary code and Gray code

Input code : Binary code : $B_3\ B_2\ B_1\ B_0$

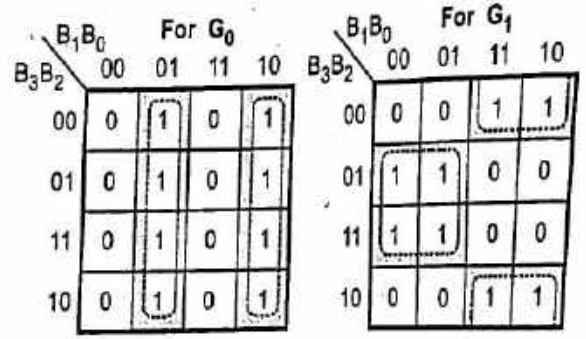
Output code : Gray code : $G_3\ G_2\ G_1\ G_0$

Decimal	Binary code				Gray code			
	B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0

5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

Table Q.3.1

Step 2 : K-map simplification for each gray code output



$G_0 = B_1\bar{B}_0 + \bar{B}_1B_0 = B_1 \oplus B_0$

$G_1 = B_2\bar{B}_1 + \bar{B}_2B_1 = B_2 \oplus B_1$

$G_2 = \bar{B}_3B_2 + B_3\bar{B}_2 = B_3 \oplus B_2$

$G_3 = B_3$

Fig. Q.3.1

Step 3 : Realization of code converter using XOR-gates

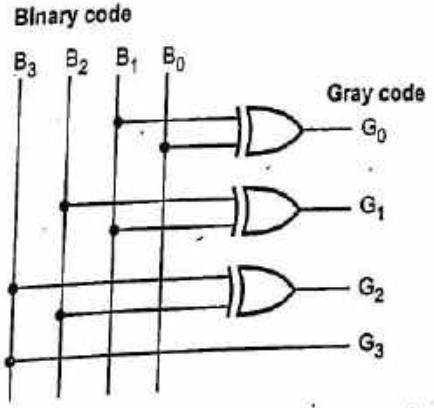


Fig. Q.3.2 Binary to gray code converter

Step 4 : Realization of code converter using NAND gates

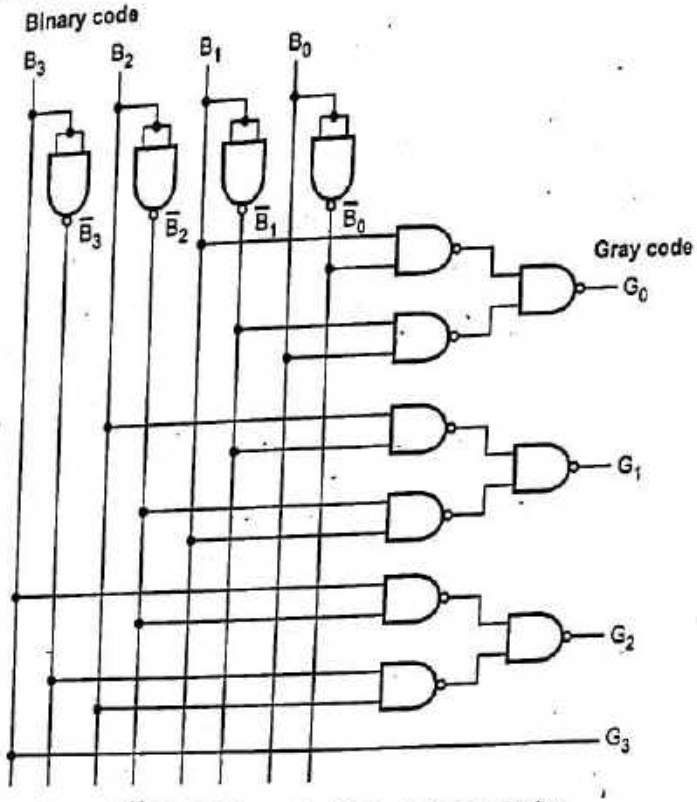


Fig. Q.3.3 Binary to Gray code converter

For this converter we have derived the Boolean expressions for each output code output in the sum of product (SOP) form. We can implement SOP expression using AND-OR logic or NAND-NAND logic. Let us see the implementation of code converter using NAND-NAND logic.

Q.4 Design a 3-bit excess 3 to 3-bit BCD code converter using logic gates. ☞[SPPU : May-15, Marks 8]

Ans. :

E_2	E_1	E_0	B_2	B_1	B_0
0	1	1	0	0	0
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	0	1	1
1	1	1	1	0	0

K-map simplification

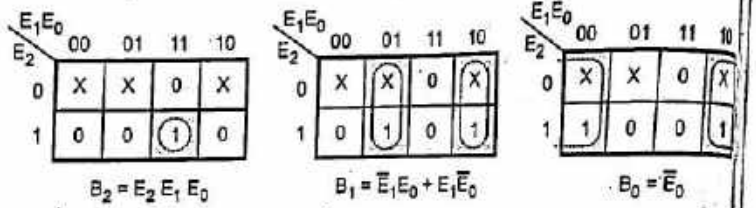


Fig. Q.4.1

Logic diagram

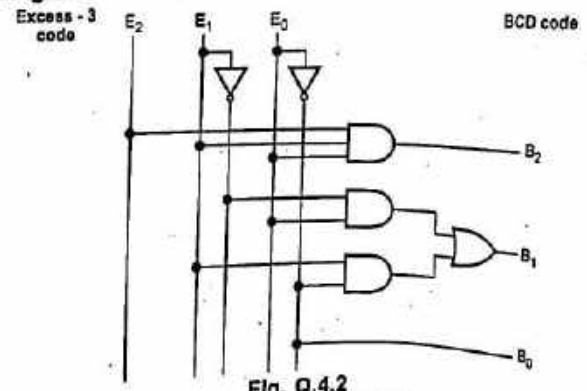


Fig. Q.4.2

4.2 : Half - Adder, Full Adder, Half Subtractor, Full Subtractor, Binary Adder (IC 7483)

Q.5 What is half adder and full adder ? ☞[SPPU : May-07, 14, Marks 4]

Ans. : • The logic circuit which performs addition of two bits is called a **half-adder.**

• The circuit which performs addition of three bits (two significant bits and a previous carry) is a **full-adder.**

Q.6 Draw and explain the function of half-adder and full adder with suitable diagram.

OR Draw the suitable diagram of full adder.

OR What are the full adder's inputs that will produce each of the following outputs ? i) $\Sigma = 0, C_{out} = 0$ ii) $\Sigma = 1, C_{out} = 0$ iii) $\Sigma = 1, C_{out} = 1$ iv) $\Sigma = 0, C_{out} = 1$ ☞[SPPU : Dec.-15, Marks 2]

Ans. : Half-adder

• The half-adder operation needs two binary inputs : augend and addend bits; and two binary outputs : sum and carry.

• The truth table shown in Table Q.6.1 gives the relation between input and output variables for half-adder operation.

Inputs		Outputs	
A	B	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Table Q.6.1
Truth table for half-adder

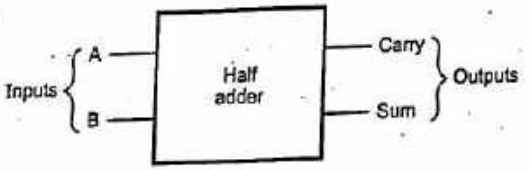


Fig. Q.6.1
Block schematic of half-adder

K-map simplification for carry and sum

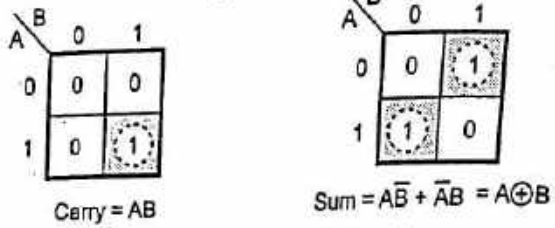


Fig. Q.6.2 Maps for half-adder

Logic diagram

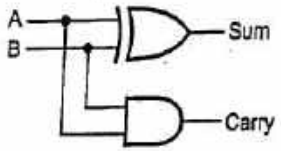


Fig. Q.6.3 Logic diagram for half-adder

Full adder

- A full-adder is a combinational circuit that forms the arithmetic sum of three input bits.
- It consists of three inputs and two outputs.
- Two of the input variables, denoted by A and B, represent the two significant bits to be added. The third input C_{in}, represents the carry from the previous lower significant position.

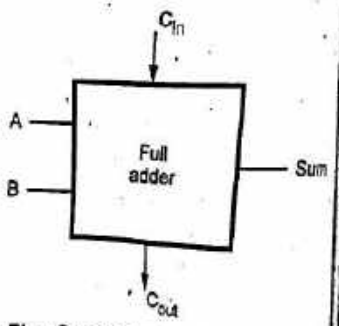


Fig. Q.6.4 Block schematic of full-adder

- The truth table for full-adder is shown in Table Q.6.2.

Inputs			Outputs	
A	B	C_{in}	Carry	Sum
0	0	0	0	0
0	0	1	0	1

0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Table Q.6.2 Truth table for full-adder

K-map simplification for carry and sum

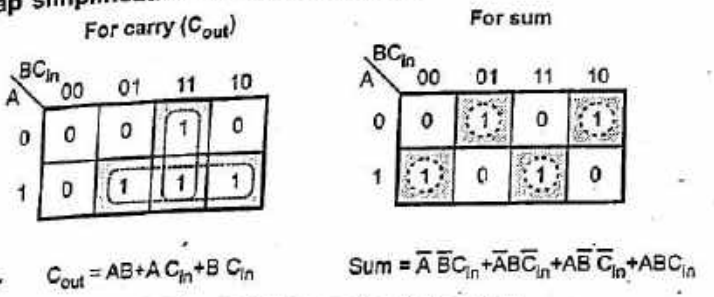


Fig. Q.6.5 Maps for full-adder

Logic diagram

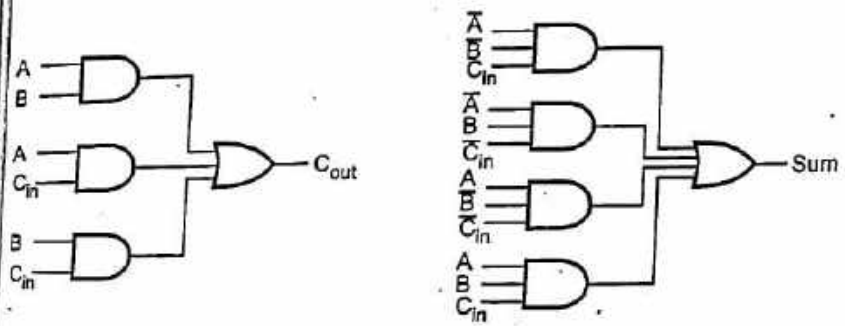


Fig. Q.6.6 Sum of product implementation of full-adder

- The Boolean function for sum can be further simplified as follows :

$$\text{Sum} = \overline{A}\,\overline{B}\,C_{in} + \overline{A}\,B\,\overline{C}_{in} + A\,\overline{B}\,\overline{C}_{in} + A\,B\,C_{in}$$
$$= C_{in}\,(\overline{A}\,\overline{B} + AB) + \overline{C}_{in}\,(\overline{A}\,B + A\,\overline{B})$$
$$= C_{in}\,(A \odot B) + \overline{C}_{in}\,(A \oplus B)$$
$$= C_{in}\,(\overline{A \oplus B}) + \overline{C}_{in}\,(A \oplus B) = C_{in} \oplus (A \oplus B)$$

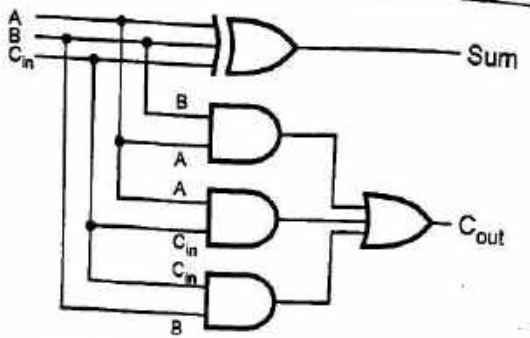


Fig. Q.6.7 Implementation of full-adder

Q.7 Draw half adder using NAND gates.

Ans. : For half adder :

$$\text{Sum} = A\overline{B} + \overline{A}B \qquad C_{out} = AB$$
$$= \overline{\overline{A\overline{B}} + \overline{\overline{A}B}} \qquad = \overline{\overline{AB}}$$
$$= \overline{\overline{A\overline{B}} \cdot \overline{\overline{A}B}}$$

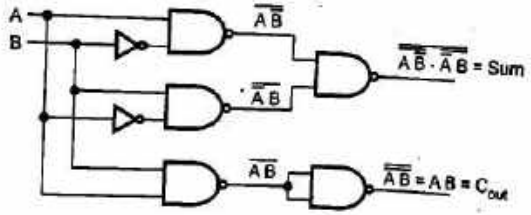


Fig. Q.7.1

Q.8 Design full adder using NAND gates only.

Ans. : Fig. Q.6.6 shows sum of product implementation of full adder using basic gates. We know that sum of product expressions can be implemented by NAND-NAND logic. Thus by replacing AND and OR gates by NAND gates in Fig. Q.6.6. We can implement full adder using NAND gates.

Q.9 Implement full adder using two half adders.

☞ [SPPU : May-07,14, Marks 3]

Ans. : • A full-adder can also be implemented with two half-adders and one OR gate, as shown in the Fig. Q.9.1.

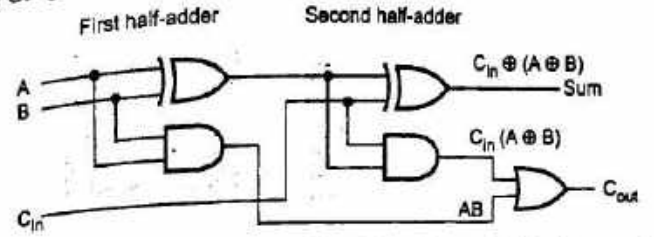


Fig. Q.9.1 Implementation of a full-adder with two half-adders and an OR gate

• The sum output from the second half-adder is the exclusive-OR of C_{in} and the output of the first half-adder, giving

$$C_{out} = AB + A\,C_{in} + B\,C_{in}$$
$$= AB + A\,C_{in}\,(B + \overline{B}) + B\,C_{in}\,(A + \overline{A})$$
$$= AB + ABC_{in} + A\,\overline{B}\,C_{in} + ABC_{in} + \overline{A}\,BC_{in}$$
$$= AB\,(1 + C_{in} + C_{in}) + A\,\overline{B}C_{in} + \overline{A}\,BC_{in}$$
$$= AB + A\overline{B}C_{in} + \overline{A}BC_{in} = AB + C_{in}\,(A\overline{B} + \overline{A}B)$$
$$= AB + C_{in}\,(A \oplus B)$$

Q.10 Implement the following Boolean function F using three half-adder circuits : $F(A, B, C) = A \oplus B \oplus C$.

Ans. : We can implement the given function using only two half-adders as shown in Fig. Q.9.1.

Q.11 Draw and explain the function of half-subtractor and full subtractor with suitable diagram. ☞ [SPPU : Dec.-08, Marks 8]

Ans. : Half subtractor :

• A half-subtractor is a combinational circuit that subtracts two-bits and produces their difference.

• It also has an output to specify if a 1 has been borrowed.

• Let us designate minuend bit as A and the subtrahend bit as B. The result of operation A – B for all possible values of A and B is tabulated in Table Q.11.1.

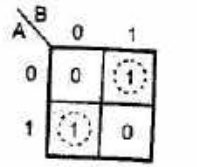
- The Boolean expression for the outputs of half-subtractor can be determined as follows.

Inputs		Outputs	
A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Table Q.11.1 Truth table for half-subtractor

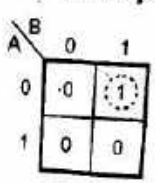
K-map simplification for half-subtractor

For difference



Difference = $A\overline{B} + \overline{A}B$
= $A \oplus B$

For borrow



Borrow = $\overline{A}B$

Fig. Q.11.1

Logic diagram

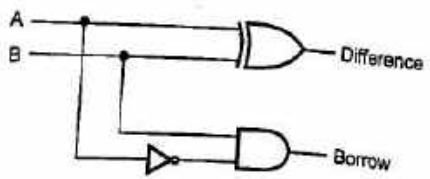


Fig. Q.11.2 Implementation of half-subtractor

Full-subtractor

- A full-subtractor is a combinational circuit that performs a subtraction between two bits, taking into account borrow of the lower significant stage.
- This circuit has three inputs and two outputs.

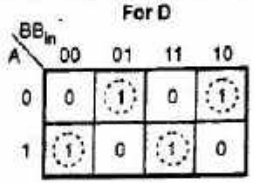
- The three inputs are A, B and B_{in}, denote the minuend, subtrahend, and previous borrow, respectively.
- The two outputs, D and B_{out}, represent the difference and output borrow, respectively.
- The Table Q.11.2 shows the truth table for full-subtractor.

Inputs			Outputs	
A	B	B_{in}	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Table Q.11.2 Truth table for full-subtractor

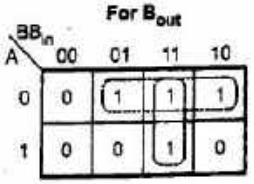
K-map simplification of D and B_{out}

For D



$D = \overline{A}\,\overline{B}B_{in} + \overline{A}B\overline{B}_{in} + A\overline{B}\,\overline{B}_{in} + ABB_{in}$

For B_{out}



$B_{out} = \overline{A}B_{in} + \overline{A}B + BB_{in}$

Fig. Q.11.3 Maps for full-subtractor

Logic diagram

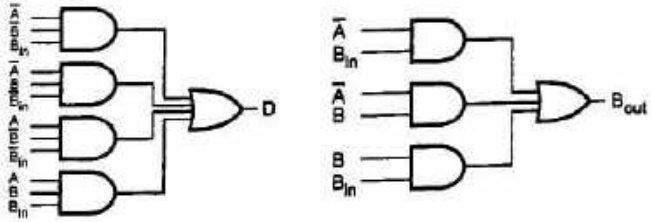


Fig. Q.11.4 Sum of product implementation of full-subtractor

- The Boolean function for D (difference) can be further simplified as follows :

$$D = \overline{A}\,\overline{B}\,B_{in} + \overline{A}\,B\,\overline{B}_{in} + A\,\overline{B}\,\overline{B}_{in} + A\,B\,B_{in}$$

$$= B_{in}(\overline{A}\,\overline{B} + AB) + \overline{B}_{in}(\overline{A}\,B + A\,\overline{B})$$

$$= B_{in}(A \odot B) + \overline{B}_{in}(A \oplus B)$$

$$= B_{in}\overline{(A \oplus B)} + \overline{B}_{in}(A \oplus B) = B_{in} \oplus (A \oplus B)$$

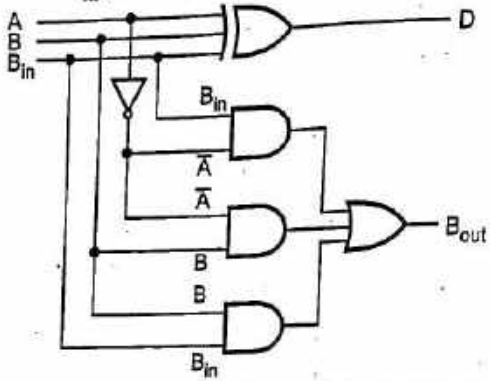


Fig. Q.11.5 Implementation of full-subtractor

Q.12 Draw half subtractor using NAND gates.

Ans. : For half subtractor :

Difference $= A\overline{B} + \overline{A}B$

$$= \overline{A\overline{B} + \overline{A}B} = \overline{A\overline{B}} \cdot \overline{\overline{A}B}$$

Borrow $= \overline{A}B$

$$= \overline{\overline{A}B}$$

Implementation :

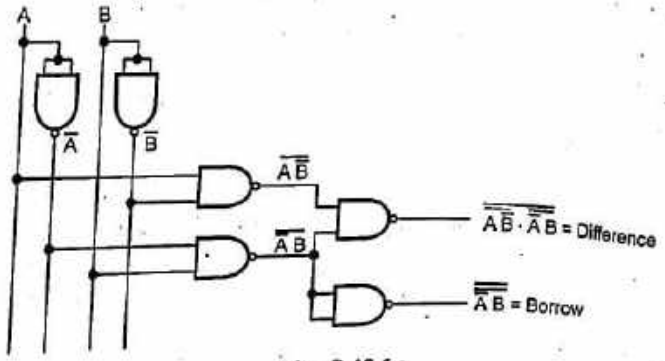


Fig. Q.12.1

Q.13 Implement a full subtractor with two half subtractors and an OR gate.

Ans. : • A full subtractor can also be implemented with two half-subtractors and one OR gate, as shown in the Fig. Q.13.1.

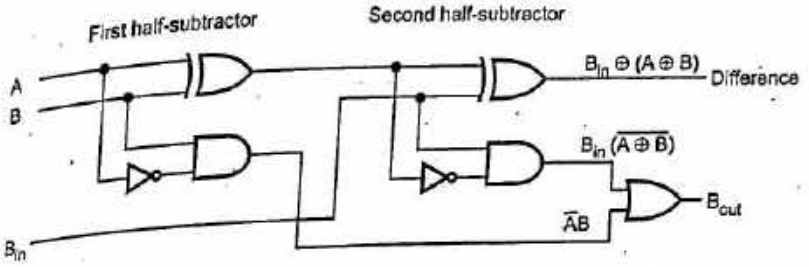


Fig. Q.13.1 Implementation of a full-subtractor with two half-subtractors and an OR gate

- The difference output from the second half-subtractor is the exclusive-OR of B_{in} and the output of the first half-subtractor, which is same as difference output of full-subtractor.
- The borrow output for full subtractor is given as,

$$B_{out} = \overline{A}B_{in} + \overline{A}B + BB_{in}$$

$$= \overline{A}B_{in}(B + \overline{B}) + \overline{A}B + BB_{in}(A + \overline{A})$$

$$= \overline{A}BB_{in} + \overline{A}\,\overline{B}B_{in} + \overline{A}B + ABB_{in} + \overline{A}BB_{in}$$

$$= \overline{A}B(B_{in} + 1 + B_{in}) + \overline{A}\,\overline{B}B_{in} + ABB_{in}$$

$$= \overline{A}B + \overline{A}\,\overline{B}B_{in} + ABB_{in}$$

$$= \overline{A}B + B_{in}(\overline{A}\,\overline{B} + AB) = \overline{A}B + B_{in}\overline{(A \oplus B)}$$

- This Boolean function is same as borrow out of the full-subtractor. Therefore, we can implement full-subtractor using two half-subtractors and OR gate.

4.3 : n-bit Binary Adder

Q.14 Explain the working of n-bit binary adder.

Ans. : Parallel Adder

- A single full-adder is capable of adding two one-bit numbers and a input carry. In order to add binary numbers with more than one bi additional full-adders must be employed.
- A n-bit, parallel adder can be constructed using number of full adde circuits connected in parallel.
- Fig. Q.14.1 shows the block diagram of n-bit parallel adder using number of full-adder circuits connected in cascade, i.e. the carry outpu of each adder is connected to the carry input of the next higher-orde adder.

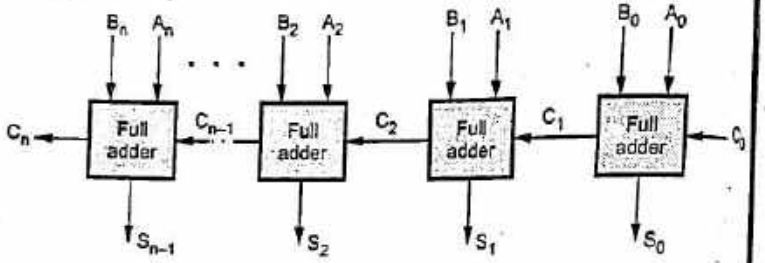


Fig. Q.14.1 Block diagram of n-bit parallel adder

- It should be noted that either a half-adder can be used for the least significant position or the carry input of a full-adder is made 0 because there is no carry into the least significant bit position.

END...

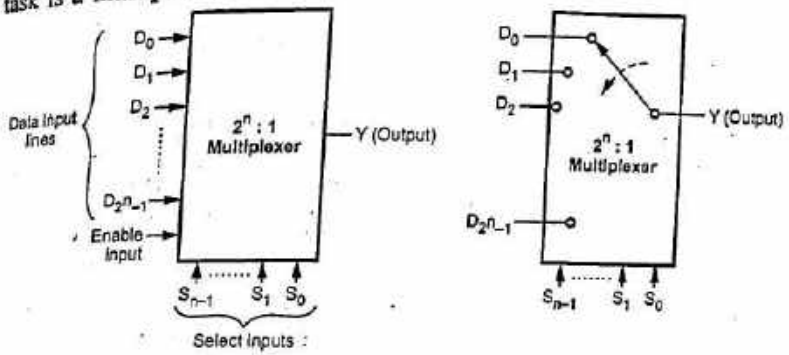
Unit 2

5 Introduction to MSI Chips and Design using MSI Chips

5.1 : Multiplexers (IC 74153) and Implementation of Logic Function using IC 74153

Q.1 What is multiplexer ?

Ans. : • In digital systems, many times it is necessary to select single data line from several data-input lines, and the data from the selected data line should be available on the output. The digital circuit which does this task is a **multiplexer.**



(a) Block diagram of $2^n : 1$ multiplexer (b) Equivalent circuit

Fig. Q.1.1

- The selection of a particular input line is controlled by a set of selection lines.
- Normally, there are 2^n input lines and n selection lines whose bit combinations determine which input is selected. Therefore, multiplexer is **'many into one'** and it provides the digital equivalent of an analog selector switch.

Q.2 Draw and explain the logic diagram of 4 : 1 line multiplexer.

Ans. : Fig. Q.2.1 (a) shows 4-to-1 line multiplexer.

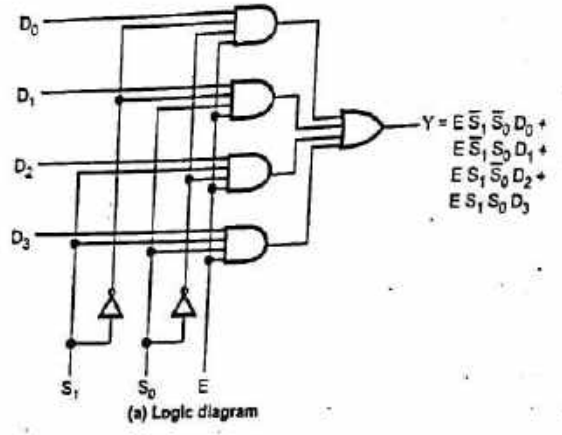


Fig. Q.2.1

- Each of the four lines, D_0 to D_3, is applied to one input of an AND gate.
- Selection lines are decoded to select a particular AND gate.
- For example, when $S_1S_0 = 0\ 1$, the AND gate associated with data input D_1 has two of its inputs equal to 1 and the third input connected to D_1. The other three AND gates have at least one input equal to 0, which makes their outputs equal to 0. The OR gate output is now equal to the value of D_1, thus we can say data bit D_1 is routed to the output when $S_1\ S_0 = 0\ 1$.

E	S_1	S_0	Y
1	0	0	D_0
1	0	1	D_1
1	1	0	D_2
1	1	1	D_3
0	X	X	0

(b) Function table

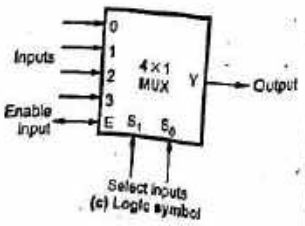


Fig. Q.2.1 4 to 1 line multiplexer

Q.3 Show how 4-input multiplexer can be realize using 2-input multiplexers.

Ans. : • It is possible to expand range of inputs for multiplexer beyond the available range by interconnecting several multiplexers in cascade.

• The circuit with two or more multiplexers connected to obtain the multiplexer with more number of inputs is known as multiplexer tree.

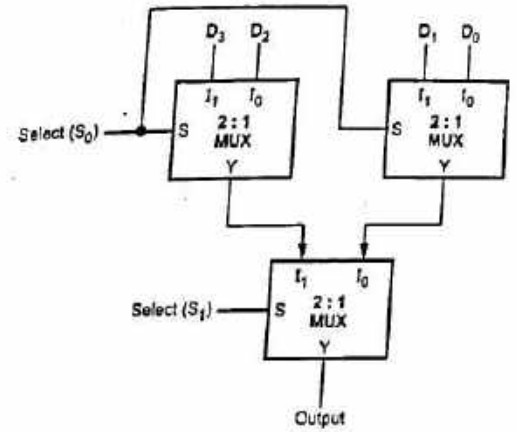


Fig. Q.3.1

Q.4 Explain the process of implementation of combinational circuit using multiplexer.

Ans. : Let us implement $F(A, B, C) = \Sigma m\ (1, 3, 5, 6)$ Boolean function using 4 : 1 multiplexer.

Step 1 : Connect least significant variables as select inputs of multiplexer. Here, connect C to S_0 and B to S_1.

Step 2 : Derive inputs for multiplexer using implementation table.

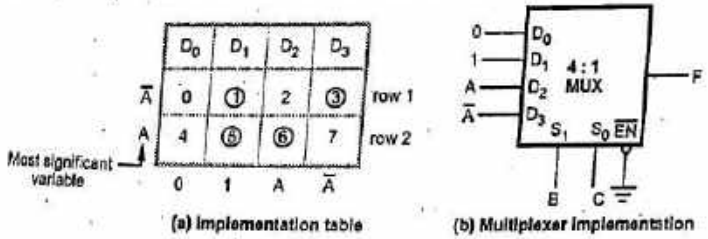


Fig. Q.4.1

As shown in the Fig. Q.4.1 (a) the implementation table is nothing but a list of the inputs of the multiplexer and under them list of all the minterms in two rows. The first row lists all those minterms where A is complemented, and the second row lists all the minterms with A uncomplemented. The minterms given in the function are circled and then each column is inspected separately as follows :

- If the two minterms in a column are not circled, 0 is applied to the corresponding multiplexer input (see column 0).
- If the two minterms in a column are circled, 1 is applied to the corresponding multiplexer input (see column 1).
- If the minterm in the second row is circled and minterm in the first row is not circled, A is applied to the corresponding multiplexer input (see column 2).
- If the minterm in the first row is circled and minterm in the second row is not circled, $\overline{A}$ is applied to the corresponding multiplexer input (see column 3).

Q.5 Design 16 : 1 multiplexer using 4 : 1 multiplexers.

☞ [SPPU : May-11, Marks 8]

Ans. : Since there are 16-inputs for the multiplexers we require four 4 : 1 multiplexers to satisfy input needs. The four outputs of 4 : 1 multiplexers are again multiplexed by 4 : 1 multiplexer to generate final output.

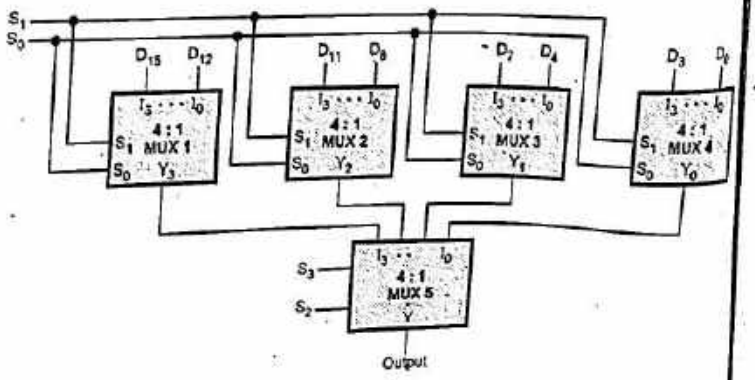


Fig. Q.5.1

Step 1 : Connect the select lines (S_1 and S_0) of four multiplexers in parallel.

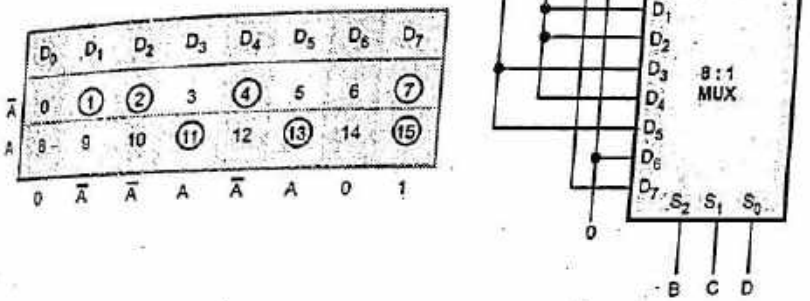
Step 2 : Connect the most significant select lines (S_3 and S_2) to the MUX 5.

Step 3 : Connect the outputs Y_0, Y_1, Y_2 and Y_4 of four multiplexers as data inputs for the MUX 5, as shown in the Fig. Q.5.1.

Q.6 Implement the following Boolean function with 8 : 1 multiplexer

$$F(A, B, C, D) = \pi M (0, 3, 5, 8, 9, 10, 12, 14)$$

Ans. : Here, instead of minterms, maxterms are specified. Thus, we have to circle maxterms which are not included in the Boolean function. Fig. Q.6.1 shows the implementation of Boolean function with 8 : 1 multiplexer.



(a) Implementation table (b) Multiplexer implementation

Fig. Q.6.1

Q.7 Design and implement BCD to Excess-3 code converter using dual 4:1 multiplexers and some logic gates. ☞ [SPPU : May-13, Marks 8]

Ans. : Refer Q.2 of Chapter - 4 for truth table of BCD to excess - 3 code conversion.

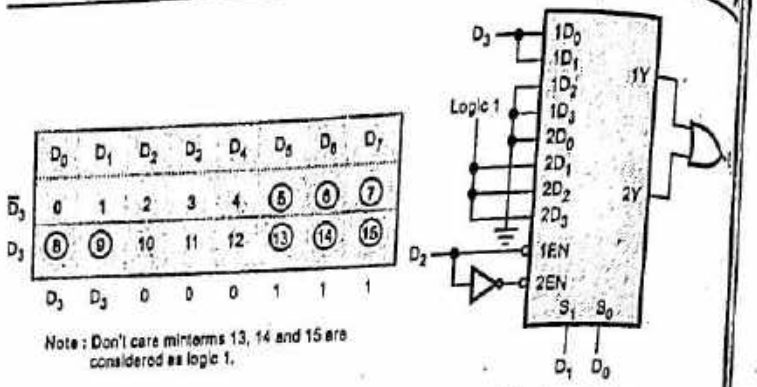


Fig. Q.7.1

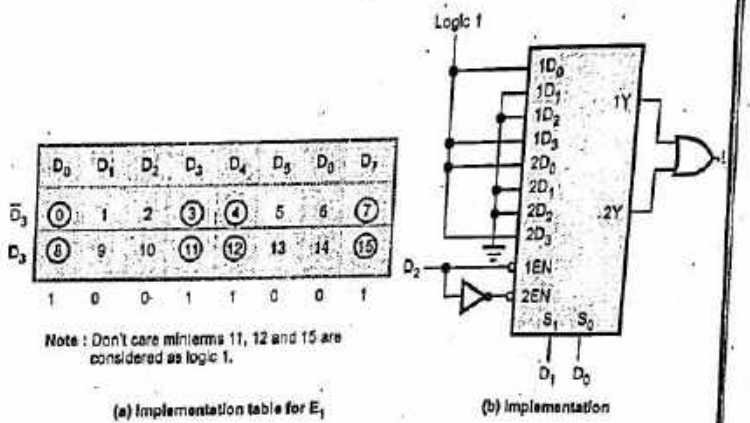


Fig. Q.7.2

Q.8 Implement the following Boolean function using single 8 : 1 multiplexer.

$F(A, B, C, D) = \sum m(1, 4, 6, 9, 13)$. ☞ [SPPU : May-17, Marks 6]

Ans. :

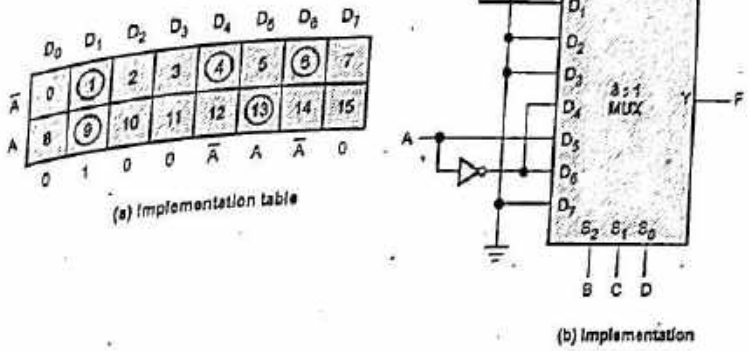


Fig. Q.8.1

Q.9 Implement given function using 8 : 1 MUX and logic gates

$F(A, B, C, D) = \sum m(0, 1, 3, 4, 8, 9, 15)$ ☞ [SPPU : Dec.-18, Marks 6]

Ans. :

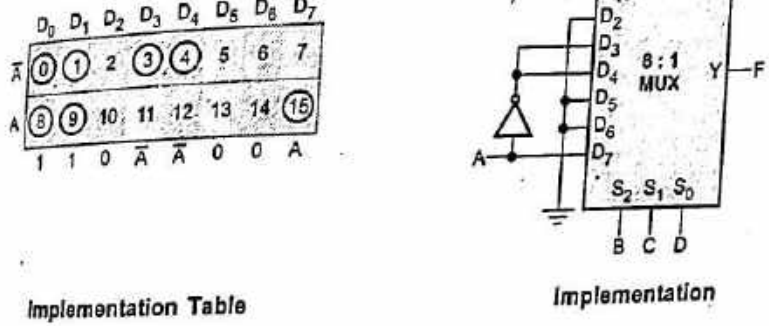


Fig. Q.9.1

Q.10 Design and implement 8 : 1 MUX using two 4 : 1 MUX and implement given function $F(X, Y, Z) = \sum m(1,3,4,7)$.

☞ [SPPU : May-19, Marks 6]

Ans. :

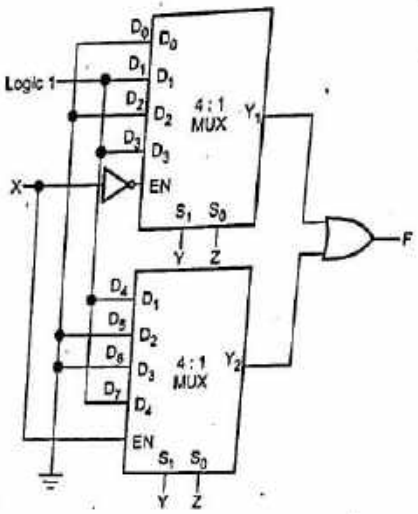


Fig. Q.10.1

Q.11 Write a note on IC MUX 74153.

Ans. :

Inputs				Outputs	
1EN	2EN	B	A	1Y	2Y
0	0	0	0	$1D_0$	$2D_0$
0	0	0	1	$1D_1$	$2D_1$
0	0	1	0	$1D_2$	$2D_2$
0	0	1	1	$1D_3$	$2D_3$
0	1	0	0	$1D_0$	0
0	1	0	1	$1D_1$	0
0	1	1	0	$1D_2$	0
0	1	1	1	$1D_3$	0
1	0	0	0	0	$2D_0$
1	0	0	1	0	$2D_1$
1	0	1	0	0	$2D_2$
1	0	1	1	0	$2D_3$
1	1	x	x	0	0

Table Q.11.1 Truth table for 74XX153, dual 4-to-1 multiplexer

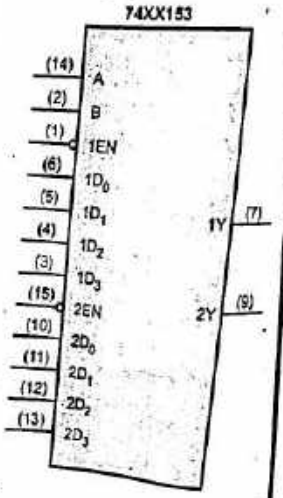


Fig. Q.11.1 Logic symbol for 74XX153

The 74XX153 is a dual 4 to 1 multiplexer. Fig. Q.11.1 shows the logic symbol for 74XX153. It contains two identical and independent 4-to-1 multiplexers. Each multiplexer has separate enable inputs. The Table Q.11.1 shows the truth table for 74XX153.

5.2 : Decoder / Demultiplexer (IC 74138, IC 74238) and Implementation of Logic Function using IC 74138

Q.12 What is decoder ? Draw and explain a 2 to 4 line decoder.

Ans. : A decoder is a multiple-input, multiple-output logic circuit which converts coded inputs into coded outputs, where the input and output codes are different.

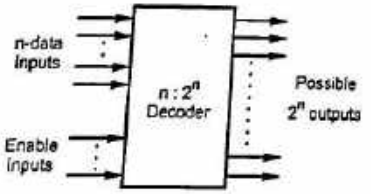


Fig. Q.12.1 General structure of decoder

- Fig. Q.12.2 shows 2-to-4 decoder. 2 inputs are decoded into four outputs, each output representing one of the minterms of the 2 input variables.

Inputs			Outputs			
EN	A	B	Y_3	Y_2	Y_1	Y_0
0	X	X	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

Table Q.12.1 Truth table for a 2 to 4 decoder

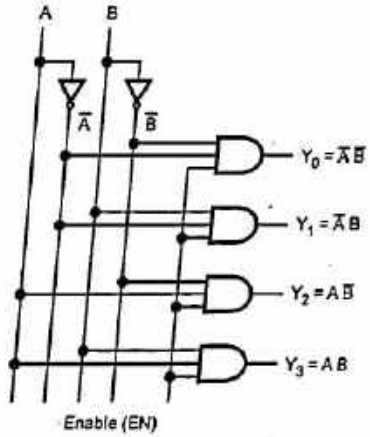


Fig. Q.12.2 2 to 4 line decoder

- The two inverters provide the complement of the inputs, and each one of four AND gates generates one of the minterms.
- The Table Q.12.1 shows the truth table for a 2-to-4 decoder.
- If enable input is 1 (EN = 1), one, and only one, of the outputs Y_0 to Y_3, is active for a given input.

- The output Y_0 is active, i.e. $Y_0 = 1$ when inputs A = B = 0, the output Y_1 is active when inputs A = 0 and B = 1.
- If enable input is 0, i.e. EN = 0, then all the outputs are 0.

Q.13 Explain the procedure to implement combination circuit using decoder with the help of example.

Ans. : • The combination of decoder and external logic gates can be used to implement single or multiple output functions.

- When decoder output is active high, it generates minterms (product terms) for input variables; i.e. it makes selected output logic 1. In such case to implement SOP function we have to take sum of selected product terms generated by decoder.
- Let us see the Implementation of Boolean function $F = \sum m(1, 2, 3, 7)$ using 3 : 8 decoder.

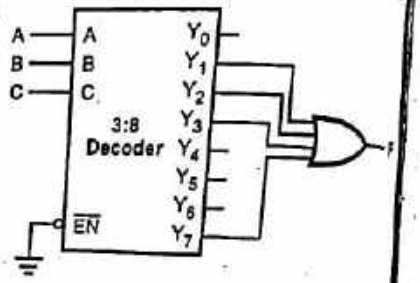


Fig. Q.13.1

Step 1 : Connect function variables as inputs to the decoder.

Step 2 : Logically OR the outputs correspond to present minterms to obtain the output.

Q.14 Design and implement a full adder circuit using a 3 : 8 decoder.

[SPPU : May-10, Dec.-10,16 Marks 4]

Ans. : Truth table for full adder is as shown in the Table Q.14.1.

Inputs			Outputs	
A	B	C_{in}	Carry	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Table Q.14.1 Truth table for full-adder

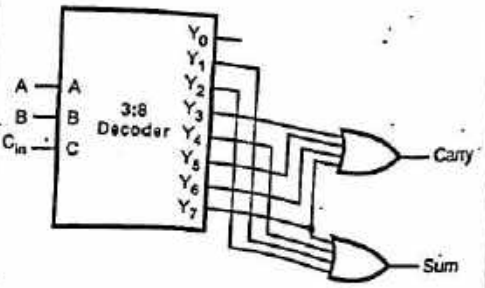


Fig. Q.14.1

Q.15 Design a 4 × 16 decoder using 3 × 8 decoders.

Ans. : • Fig. Q.15.1 shows the 4 × 16 decoder using two 3 × 8 decoders.

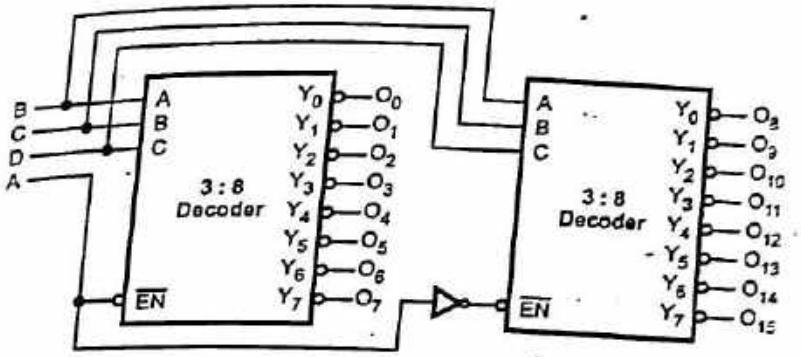
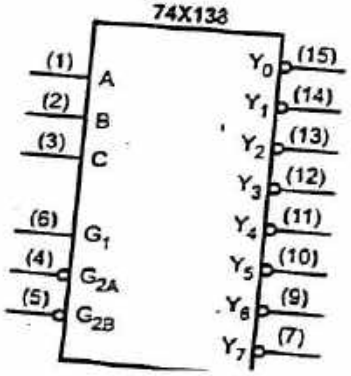


Fig. Q.15.1

- Here, one input line (D) is used to enable/disable the decoders.
- When D = 0, the top decoder is enabled and the other is disabled. Thus the bottom decoder outputs are all 1s, and the top eight outputs generate minterms 0 0 0 0 to 0 1 1 1.
- When D=1, the enable conditions are reversed and thus bottom decoder outputs generate minterms 1000 to 1Y111, while the outputs of the top decoder are all 1s.

Q.16 Write a note on IC decoder 74138.

Ans. : The 74X138 is a commercially available 3-to-8 decoder. It accepts three binary inputs (A, B, C) and when enabled, provides eight individual active low outputs (Y_0 - Y_7). The device has three enable inputs : two active low ($\overline{G}_{2A}$, $\overline{G}_{2B}$) and one active high (G_1). Fig. Q.16.1 and Table Q.16.1 show logic symbol and function table respectively.



Inputs						Outputs							
G_{2B}	G_{2A}	G_1	C	B	A	$\overline{Y}_7$	$\overline{Y}_6$	$\overline{Y}_5$	$\overline{Y}_4$	$\overline{Y}_3$	$\overline{Y}_2$	$\overline{Y}_1$	$\overline{Y}_0$
1	X	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X	X	1	1	1	1	1	1	1	1
X	X	0	X	X	X	1	1	1	1	1	1	1	1
0	0	1	0	0	0	1	1	1	1	1	1	1	0
0	0	1	0	0	1	1	1	1	1	1	1	0	1
0	0	1	0	1	0	1	1	1	1	1	0	1	1
0	0	1	0	1	1	1	1	1	1	0	1	1	1
0	0	1	1	0	0	1	1	1	0	1	1	1	1
0	0	1	1	0	1	1	1	0	1	1	1	1	1
0	0	1	1	1	0	1	0	1	1	1	1	1	1
0	0	1	1	1	1	0	1	1	1	1	1	1	1

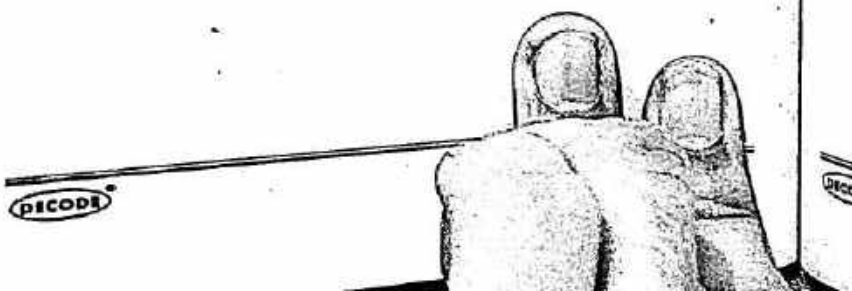
Table Q.16.1 Function table

Q.17 Design a 3-bit binary to 3-bit gray code converter using IC-74138.

[SPPU : Dec.-13, Marks 4]

Ans. : Truth table

Inputs			Outputs		
B_2	B_1	B_0	G_2	G_1	G_0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0



Implementation

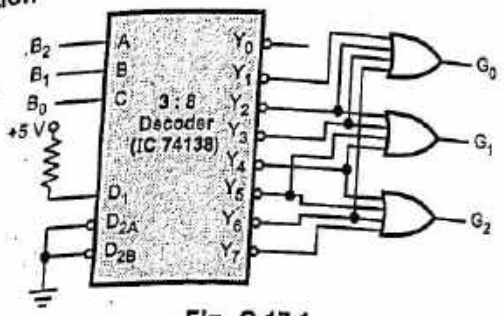


Fig. Q.17.1

Q.18 Design multiple output functions using IC 74138 and logic gates : $F_1(A, B, C) = \pi M(0, 1, 3, 7)$ and $F_2(A, B, C) = \pi M(2, 3, 7)$.

[SPPU : May-16, Marks 6]

Ans. :

$\pi M(0, 1, 3, 7) = \Sigma m(2, 4, 5, 6)$

and

$\pi M(2, 3, 7) = \Sigma m(0, 1, 4, 5, 6)$

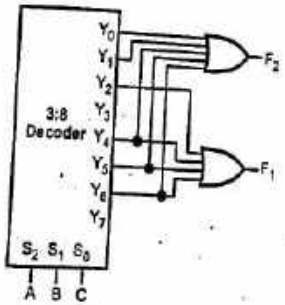


Fig. Q.18.1

Q.19 Design full subtractor using decoder IC 74138.

[SPPU : Dec.-15, Marks 6]

Ans. : Refer Q.11 of Chapter 4.

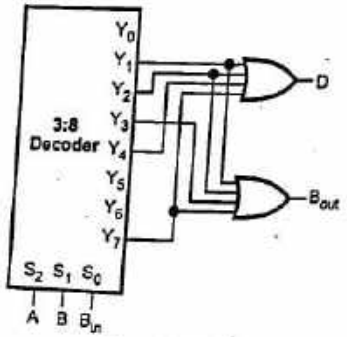


Fig. Q.19.1

Q.20 Write a note on IC 74238.

Ans. :

- The 74238 is 3 : 8 decoder. It decodes three binary weighted address inputs (A_0, A_1 and A_2) to eight mutually exclusive outputs ($Y_0 + Y_7$). The device features three enable inputs ($\overline{E}_1$ and $\overline{E}_2$ and E_3). Every output will be LOW unless $\overline{E}_1$ and $\overline{E}_2$ are LOW and E_3 is HIGH. The 74238 can be used as an eight output de-multiplexer by using one of the active LOW enable inputs as the data input and the remaining enable inputs as strobes. Fig. Q.20.1 shows the logic symbol of IC 74238. Table Q.20.1 shows function table of IC 74238.

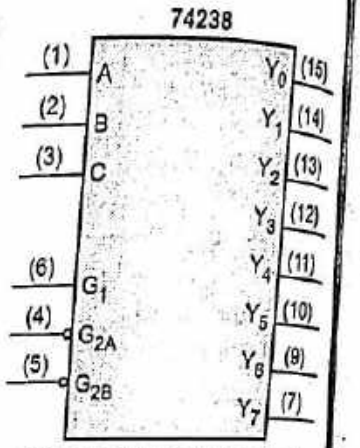


Fig. Q.20.1 Logic symbol of 74238

Inputs						Outputs							
$\overline{E}_1$	$\overline{E}_2$	E_3	A_0	A_1	A_2	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
1	X	X	X	X	X	0	0	0	0	0	0	0	0
X	1	X	X	X	X	0	0	0	0	0	0	0	0
X	X	0	X	X	X	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
0	0	1	0	1	0	0	0	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0	1	0	0	0	0
0	0	1	0	0	1	0	0	0	0	1	0	0	0
0	0	1	1	0	1	0	0	0	0	0	1	0	0
0	0	1	0	1	1	0	0	0	0	0	0	1	0
0	0	1	1	1	1	0	0	0	0	0	0	0	1

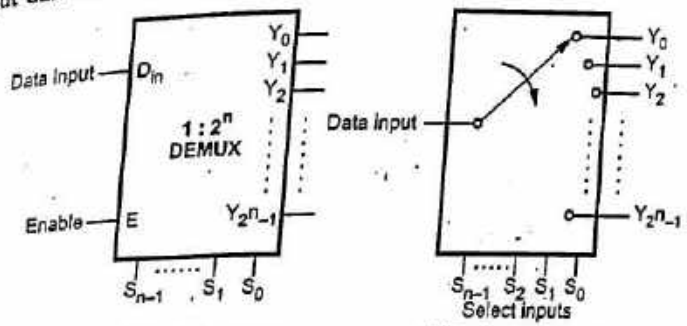
Table Q.20.1 Function table of IC 74238

Note : Important difference between 74138 and 74238 is that for 74138 outputs are active LOW and for 74238 outputs are active high.

Q.21 What is de-multiplexer ?

Ans. : A demultiplexer is a circuit that receives information on a single line and transmits this information on one of 2^n possible output lines.

- The selection of specific output line is controlled by the values of n selection lines.
- The Fig. Q.21.1 shows the block diagram of a demultiplexer. It has one input data line, 2^n output lines, n select lines and one enable input.



(a) Block diagram (b) Equivalent circuit

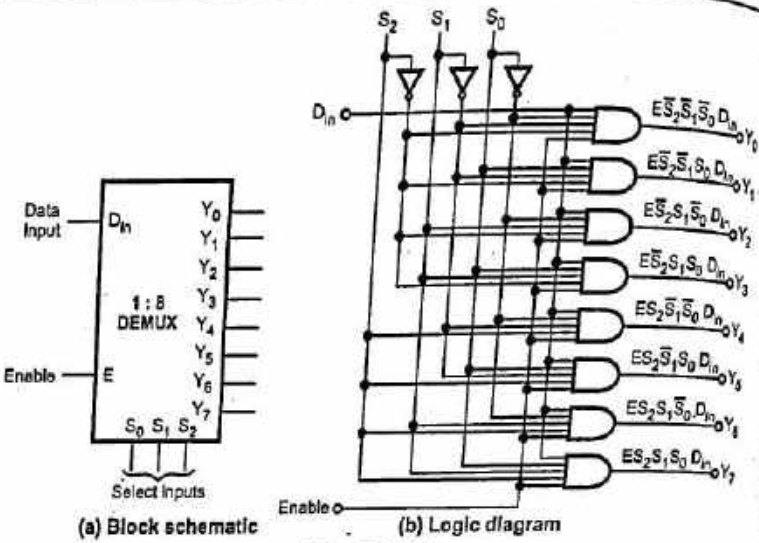
Fig. Q.21.1

Q.22 Draw and explain the logic diagram of one line to 8 - line demultiplexer.

Ans. : The Fig. Q.22.1 shows 1 : 8 demultiplexer. (Refer Fig. Q.22.1 on next page)

- The single input data D_{in} has a path to all eight outputs, but the input information is directed to only one of the output lines depending on the select inputs.

Q.23 Differentiate between multiplexer and demultiplexer.



(a) Block schematic (b) Logic diagram

Fig. Q.22.1

Ans. : Differentiate between Multiplexer and Demultiplexer

Parameter	Multiplexer	Demultiplexer
Definition	Multiplexer is a digital switch which allows digital information from several sources to be routed onto a single output line.	Demultiplexer is a circuit that receives information on a single line and transmits this information on one of 2^n possible output lines
Number of data inputs	2^n	1
Number of data outputs	1	2^n
Relationship of input and output	Many to one	One to many
Applications	• Used as a data selector • In time division multiplexing at the transmitting end	• Used as a data distributor. • In time division multiplexing at the receiving end

Q.24 Implement the full subtractor using a 1 : 8 demultiplexer.

Ans. : Table shows the truth table for full subtractor.

Inputs			Outputs	
A	B	B_{in}	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Table Q.24.1 Truth table for full-subtractor

$D = \sum m\,(1, 2, 4, 7)$ and $B_{out} = \sum m\,(1, 2, 3, 7)$

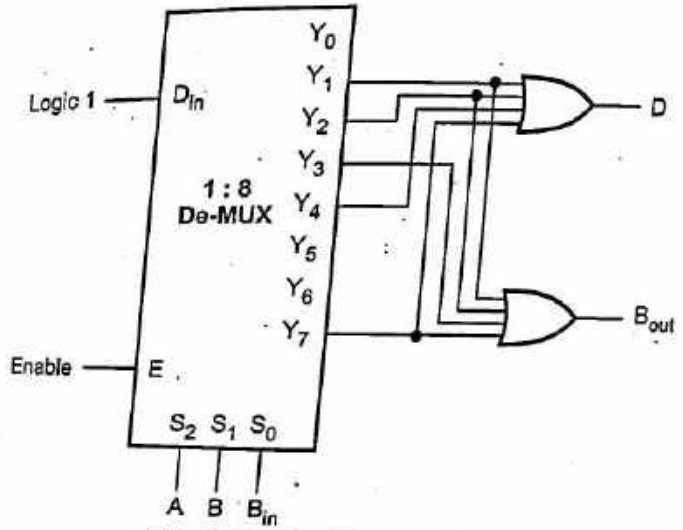


Fig. Q.24.1 Implementation

Q.25 Design 1 : 8 demultiplexer using two 1 : 4 demultiplexers.

Ans. : **Step 1 :** Connect D_{in} signal to D_{in} input of both the demultiplexers.

Step 2 : Connect select lines B and C to select lines S_1 and S_0 of both demultiplexers, respectively.

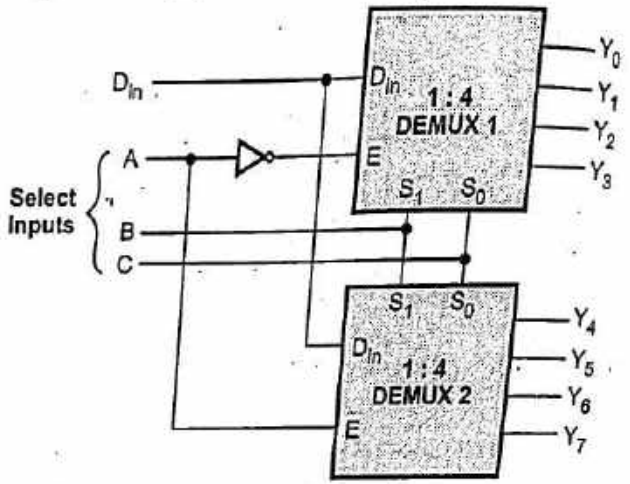


Fig. Q.25.1 Cascading of demultiplexers

Step 3 : Connect most significant select line (A) such that when A = 0 DEMUX 1 is enabled and when A = 1 DEMUX 2 is enabled.

5.3 : Encoder (IC 74147)

Q.26 What is encoder ?

Ans. : • An encoder is a digital circuit that performs the inverse operation of a decoder.

• An encoder has 2^n (or fewer) input lines and n output lines.

• In encoder the output lines generate the binary code corresponding to the input value.

• The Fig. Q.26.1 shows the general structure of the encoder circuit. As shown in the Fig. Q.26.1 the decoded information is presented as 2^n inputs producing n possible outputs.

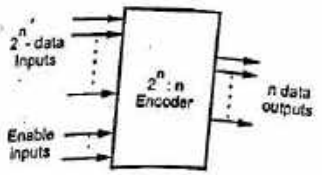


Fig. Q.26.1 General structure of encoder

Q.27 What is priority encoder ? Explain with the help of suitable example.

Ans. : • A priority encoder is an encoder circuit that includes the priority function. In priority encoder, if two or more inputs are equal to 1 at the same time, the input having the highest priority will take precedence.

• Table Q.27.1 shows truth table of 4-bit priority encoder.

Inputs				Outputs		
D_0	D_1	D_2	D_3	Y_1	Y_0	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

Table Q.27.1 Truth table of 4-bit priority encoder

• Table Q.27.1 shows D_3 input with highest priority and D_0 input with lowest priority. When D_3 input is high, regardless of other inputs output is (Y_1 Y_0 = 11) 11.

• The D_2 has the next priority. Thus, when $D_3 = 0$ and $D_2 = 1$, regardless of other two lower priority input, output is 10.

• The output for D_1 is generated only if higher priority inputs are 0, and so on.

• The output V (a valid output indicator) indicates, one or more of the inputs are equal to 1. If all inputs are 0, V is equal to 0, and the other two outputs (Y_1 and Y_0) of the circuit are not used.

K-map simplification

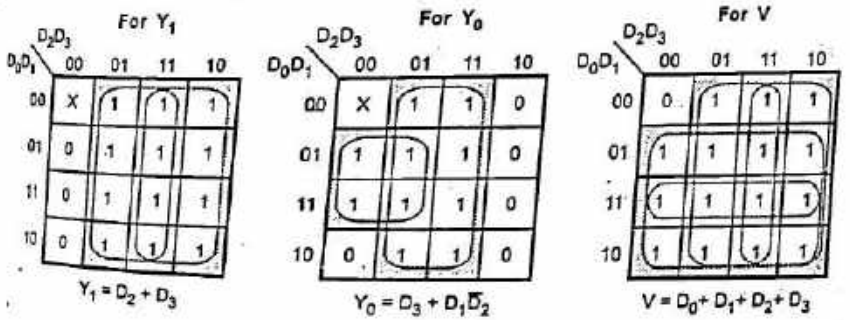


Fig. Q.27.1

Logic diagram

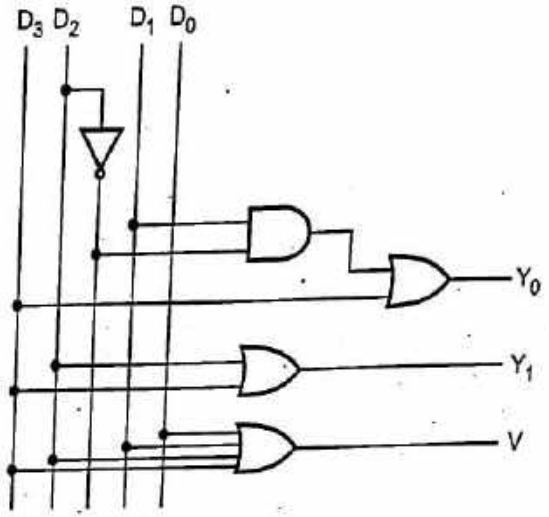


Fig. Q.27.2

Q.28 Write a note on IC 74147.

Ans. : The Fig. Q.28.1 shows the logic symbol for decimal to BCD encoder IC, IC 74XX147. It has nine input lines and four output lines. Both input and output lines are asserted active low. It is important to note that there is no input line for decimal zero. When this condition occurs, all output lines are 1. The function table for the 74XX147 is shown in Table Q.28.1.

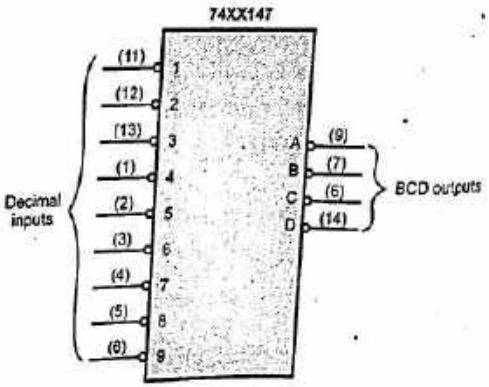


Fig. Q.28.1 Logic symbol for 74XX147 (Decimal to BCD encoder)

Decimal value	Inputs									Outputs			
	1	2	3	4	5	6	7	8	9	D	C	B	A
0	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	0
2	x	0	1	1	1	1	1	1	1	1	1	0	1
3	x	x	0	1	1	1	1	1	1	1	1	0	0
4	x	x	x	0	1	1	1	1	1	1	0	1	1
5	x	x	x	x	0	1	1	1	1	1	0	1	0
6	x	x	x	x	x	0	1	1	1	1	0	0	1
7	x	x	x	x	x	x	0	1	1	1	0	0	0
8	x	x	x	x	x	x	x	0	1	0	1	1	1
9	x	x	x	x	x	x	x	x	0	0	1	1	0

x indicates don't care condition

Table Q.28.1 Truth table for decimal to BCD encoder

5.4 : Binary Adder

Q.29 What is IC 7483 ?

Ans. : The most common is a 4-bit parallel adder IC (74LS83/74S283) that contains four inter connected full-adders and the look-ahead carry circuitry needed for high-speed operation. The 7483 and 74283 are a TTL Medium Scale Integrated (MSI) circuit with same pin

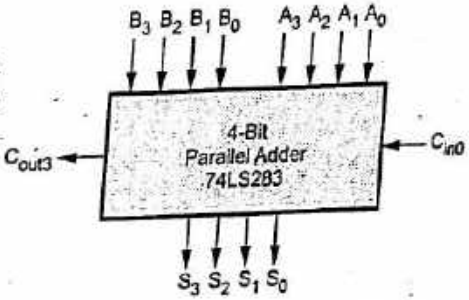


Fig. Q.14.1 Functional symbol for the 74LS283

configuration. Fig. Q.29.1 shows the functional symbol for the 74LS283 4-bit parallel adder. The inputs to this IC are two 4-bit numbers, A_3 A_2 A_1 A_0 and B_3 B_2 B_1 B_0, and the carry, C_{in0}, into the LSB position. The outputs are the sum bits S_3 S_2 S_1 S_0, and the carry, C_{out3}, output of the MSB position.

Two or more parallel adder blocks can be connected (cascaded) to accommodate the addition of larger binary numbers.

5.5 : BCD Adder

Q.30 What is BCD / Decimal adder ? Design one digit BCD / Decimal adder. [SPPU : May-05, 12, Dec.-05, 12, 14, Marks 8]

Ans. : • A BCD adder is a circuit that adds two BCD digits and produces a sum digit also in BCD.

- To implement BCD adder we require :
  - 4-bit binary adder for initial addition
  - Logic circuit to detect sum greater than 9 and
  - One more 4-bit adder to add 0110_2 in the sum if sum is greater than 9 or carry is 1.
  - The logic circuit to detect sum greater than 9 can be determined by simplifying the boolean expression of given truth table.
- Table Q.30.1 shows the truth table for BCD adder.
- Fig. Q.30.1 (a) shows the K-map simplification for the logic circuit and Fig. Q.30.1 (b) shows the block diagram of BCD adder.

Inputs				Output
S_3	S_2	S_1	S_0	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Table Q.30.1

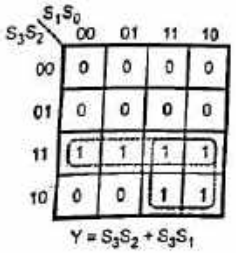


Fig. Q.30.1 (a) K-map simplification

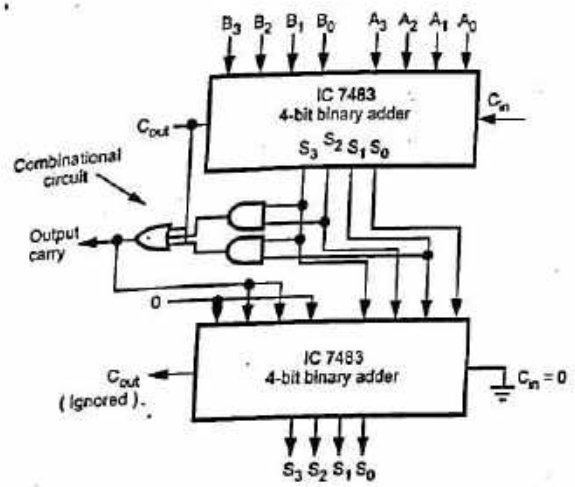


Fig. Q.30.1 (b) Block diagram of BCD adder

Q.31 Design an 8-bit BCD adder using 4-bit binary adder. [SPPU : May-05, 10, 12, Marks 8]

Ans. : To implement 8-bit BCD adder we have to cascade two 4-bit BCD adders. In cascade connection carry output of the lower position (digit) is connected as a carry input of the higher position (digit). Fig. Q.31.1 shows the block diagram of 8-bit BCD adder.

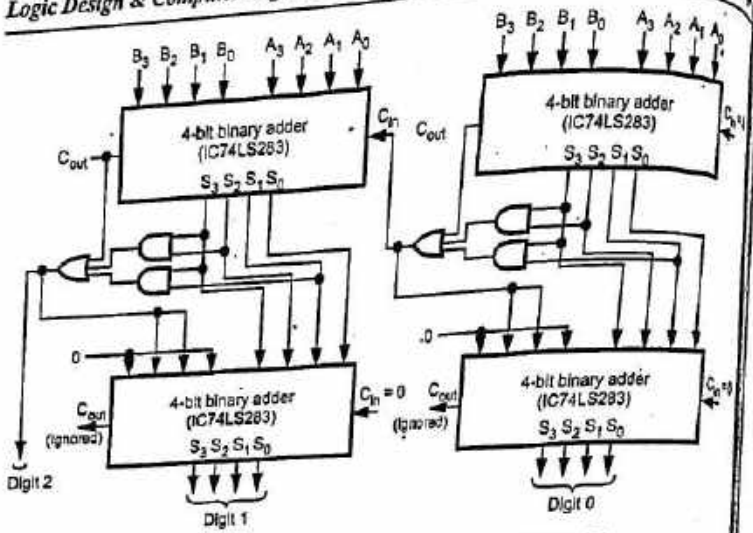


Fig. Q.31.1 8-bit BCD adder using IC 74283

Q.32 Design a BCD to Ex-3 code converter using binary parallel adder.

Ans. :

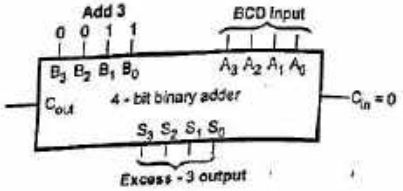


Fig. Q.32.1

5.6 : BCD Subtractor using IC 7483

Q.33 Design BCD subtractor using 9's complement method.

Ans. : The steps for 9's complement BCD subtraction as follows :

- Find the 9's complement of a negative number
- Add two numbers using BCD addition
- If carry is generated add carry to the result otherwise find the 9's complement of the result.

Fig. Q.33.1 shows the logic diagram of the circuit to implement above mentioned steps to perform BCD subtraction using 9's complement method. As shown in the Fig. Q.33.1, first binary adder finds the 9's complement of the negative number. It does this by inverting each bit of BCD number and adding 10 ($1\ 0.1\ 0_2$) to it. Let us find the 9's complement of 2. (Refer Fig. Q.33.1 on next page)

	0	0	1	0	← BCD for 2
	1	1	0	1	← Inverting each bit
+	1	0	1	0	← Add 10 (1010_2)
Ignore carry →	1	0	1	1	← 9's complement for 2

Next two 4-bit binary adders perform the BCD addition. The last adder finds the 9's complement of the result if carry is not generated after BCD addition otherwise it adds carry in the result.

Q.34 Design BCD subtractor using 10's complement method.

Ans. : The steps for 10's complement BCD subtraction as follows.

- Find the 10's complement of a negative number
- Add two numbers using BCD addition
- If carry is not generated find the 10's complement of the result.

Fig. Q.34.1 shows the logic diagram of the circuit to implement above mentioned steps to perform BCD subtraction using 10's complement method. As shown in the Fig. Q.33.1, first binary adder finds the 10's complement of the negative number (9's complement + 1). Next two 4-bit binary adders perform the BCD addition. Finally, last 4-bit binary adder finds the 10's complement of the number if carry is not generated after BCD addition.

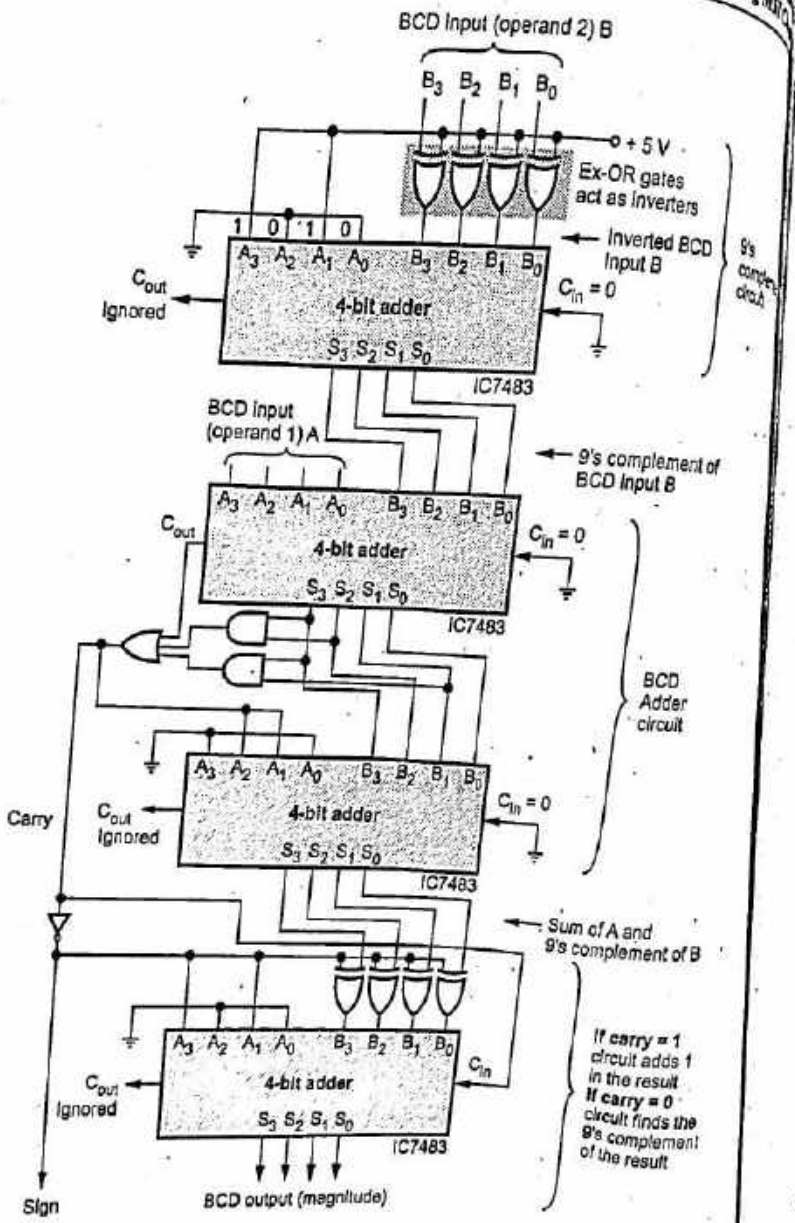


Fig. Q.33.1 4-bit BCD subtractor using 9's complement method

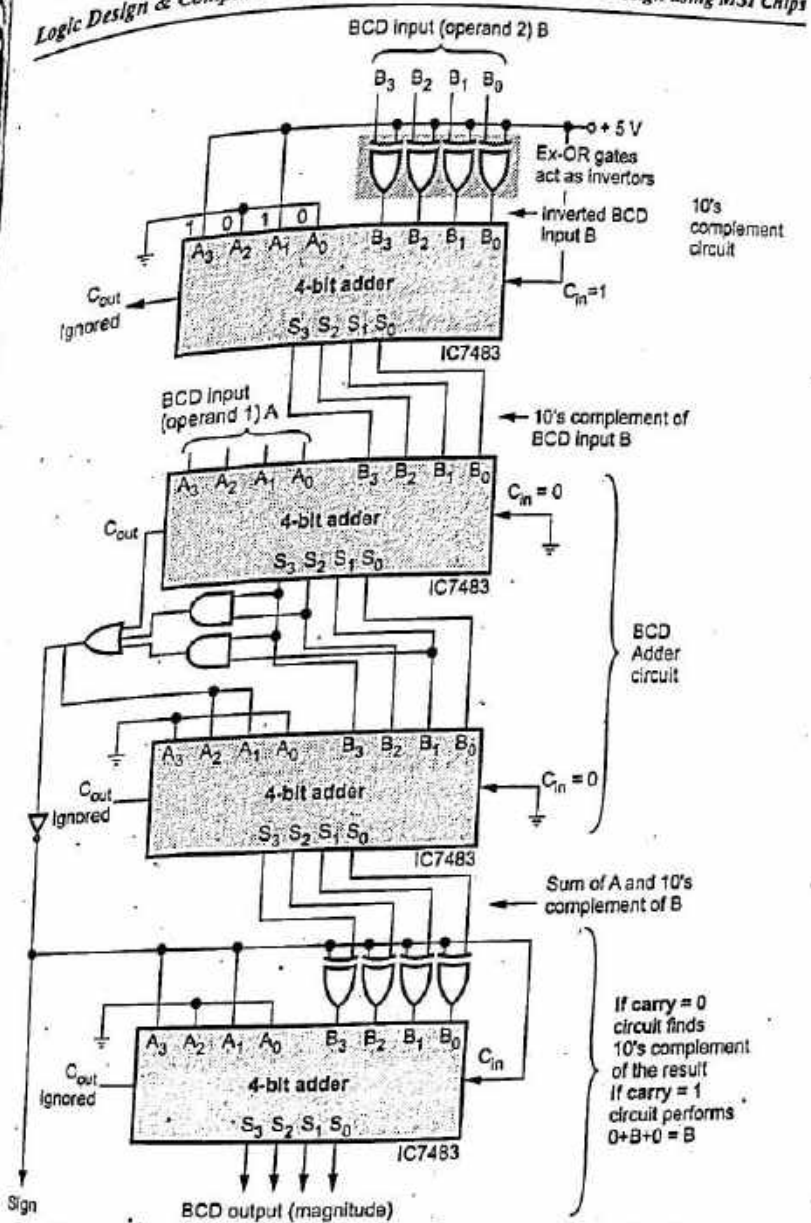


Fig. Q.34.1 4-bit BCD subtractor using 10's complement method

END...

Unit 3

6 Introduction to Sequential Circuits and Flip-Flops

6.1 : Introduction

Q.1 Draw and explain the block diagram of sequential circuit.

Ans. : • Fig. Q.1.1 shows the block diagram of sequential circuit/Finite State Machine (FSM).

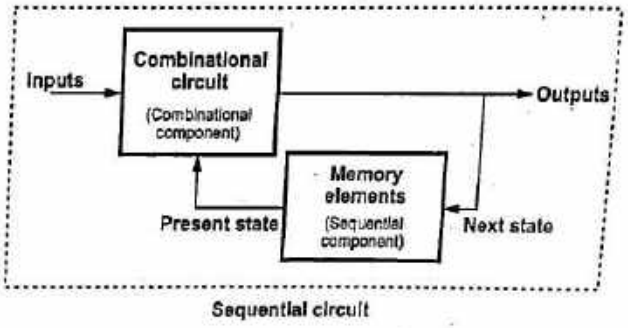


Fig. Q.1.1 Block diagram of sequential circuit / FSM

- Memory elements are connected to the combinational circuit as a feedback path.
- The information stored in the memory elements at any given time defines the **present state** of the sequential circuit.
- The present state and the external inputs determine the outputs and the **next state** of the sequential circuit.
- Thus we can specify the sequential circuit by a time sequence of external inputs, internal states (present states and next states), and outputs.
- The counters and registers are the common examples of sequential circuits.

Q.2 Differentiate between combinational logic circuits and sequential logic circuits. [SPPU : June-22, Marks 9]

Ans. :

Sr. No.	Combinational circuits	Sequential circuits
1.	In combinational circuits, the output variables are at all times dependent on the combination of input variables.	In sequential circuits, the output variables depend not only on the present input variables but they also depend upon the past history of these input variables.
2.	Memory unit is not required in combinational circuits.	Memory unit is required to store the past history of input variables in the sequential circuit.
3.	Combinational circuits are faster in speed because the delay between input and output is due to propagation delay of gates.	Sequential circuits are slower than the combinational circuits.
4.	Combinational circuits are easy to design.	Sequential circuits are comparatively harder to design.
5.	Parallel adder is a combinational circuit.	Serial adder is a sequential circuit.

Q.3 Name the two storage elements.

Ans. : 1. Latches 2. Flip-Flop.

Q.4 What is flip-flop ?

Ans. : Flip-flop is a sequential circuit driven by clock input either by positive edge or negative edge. It is a binary storage device capable of storing one bit of information.

Q.5 What is level triggering and edge triggering ?

Ans. : Level Triggering : In the level triggering, the output state is allowed to change according to input(s) when active level (either positive or negative) is maintained at the enable input. There are two types of level triggered latches :

- **Positive level triggered :** The output of flip-flop responds to the input changes only when its enable input is 1 (HIGH).

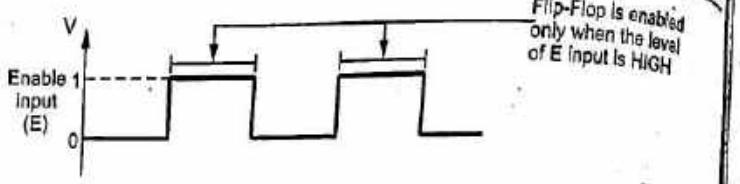


Fig. Q.5.1 Positive level triggering

- **Negative level triggered :** The output of flip-flop responds to the input changes only when its enable input is 0 (LOW).

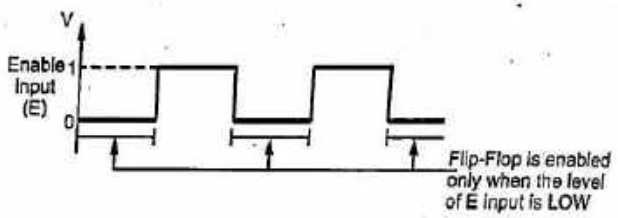


Fig. Q.5.2 Negative level triggering

Edge triggering : In the edge triggering, the output responds to the changes in the input only at the positive or negative edge of the clock pulse at the clock input. There are two types of edge triggering.

- **Positive edge triggering :** Here, the output responds to the changes in the input only at the positive edge of the clock pulse at the clock input.

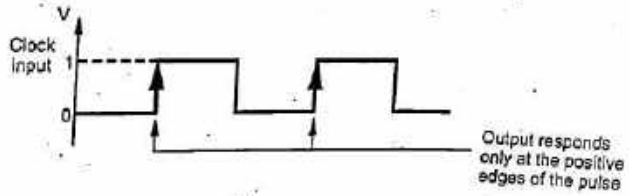


Fig. Q.5.3 Positive edge triggering

- **Negative edge triggering :** Here, the output responds to the changes in the input only at the negative edge of the clock pulse at the clock input.

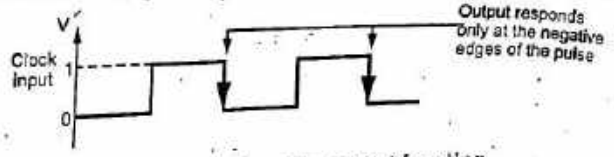
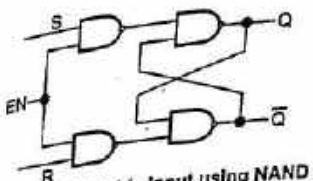
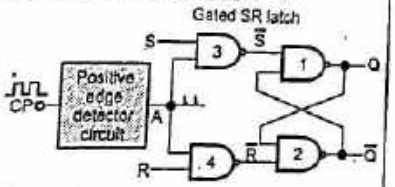


Fig. Q.5.4 Negative edge triggering

Q.6 Give the comparison between latch and flip-flop.

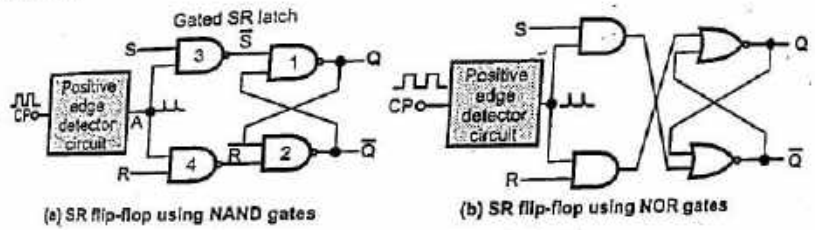
Latch	Flip-Flop
Latches are controlled by signal levels. Latches are level triggered.	Flip-Flop are controlled by clock transitions. Flip-Flops are edge triggered.
 SR latch with enable input using NAND gates	 SR flip-flop using NAND gates

6.2 : Flip-Flop : SR, JK, D, T

Q.7 Draw and explain the working of SR flip-flop. Give the truth table and characteristic equation of SR flip-flop.

☞ [SPPU : May-11, Marks 10, May-12, Marks 4]

Ans. : The Fig. Q.7.1 shows the positive edge triggered clocked SR flip-flop.



(a) SR flip-flop using NAND gates (b) SR flip-flop using NOR gates

Fig. Q.7.1 Clocked SR flip-flop

Operation

Case 1 : If S = R = 0 and the clock pulse is applied, the output do not change, i.e. $Q_{n+1} = Q_n$. This is indicated in the first row of the truth table.

Case 2 : If S = 0, R = 1 and the clock pulse is applied, $Q_{n+1} = 0$. This is indicated in the second row of the truth table.

Case 3 : If S = 1, R = 0 and the clock pulse is applied, $Q_{n+1} = 1$. This is indicated in the third row of the truth table.

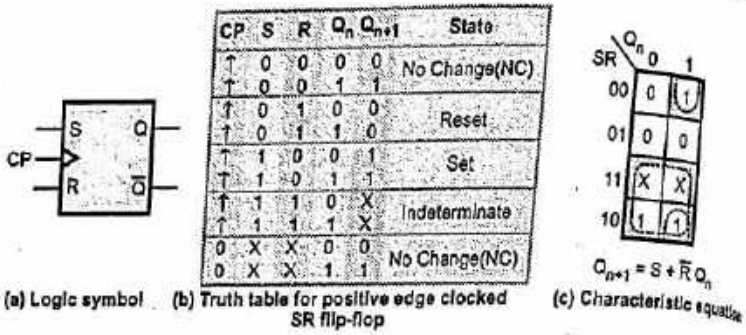


Fig. Q.7.2

Case 4 : If S = R = 1 and the clock pulse is applied, the state of the flip-flip is undefined and therefore is indicated as indeterminate in the fourth row of the truth table.

Q.8 Draw and explain the working of D flip - flop. Give truth table and characteristic equation.

OR Draw a positive edge triggered D flip - flop using NAND gates and explain its function.

Ans. : • In SR Flip-Flop, when both inputs are same the output either does not change or it is invalid (Inputs → 00, no change and inputs → 11, invalid).

- These input conditions can be avoided by making them complement of each other. This modified SR flip-flop is known as D flip-flop.
- The D input goes directly to the S input, and its complement is applied to the R input. Due to these connections, only two input conditions exists, either S = 0 and R = 1 or S = 1 and R = 0.

Truth table : • The truth table for D flip-flop consider only these two conditions and it is as shown in the Fig. Q.8.1 (b).

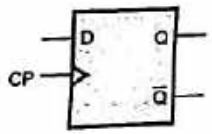


Fig. Q.8.1 (a) Logic symbol

CP	D	Q_{n+1}
↑	0	0
↑	1	1
0	X	Q_n

Fig. Q.8.1 (b) Truth table of D flip-flop

- Q_{n+1} function follows D input at the positive going edges of the clock pulses. Hence the characteristic equation for D flip-flop is $Q_{n+1} = D$.

Q.9 Draw the logic diagram and give the characteristic table of JK flip - flop.

OR Explain the operation of JK flip - flop.

Ans. : Fig. Q.9.1 shows the logic diagram, symbol and truth table of positive edge triggered JK flip-flop.

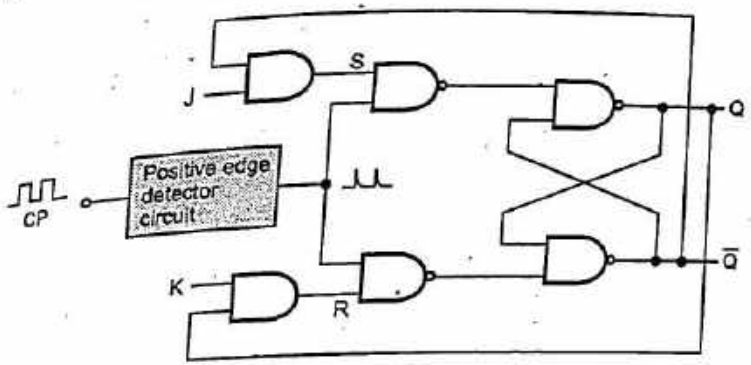
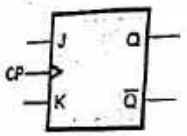


Fig. Q.9.1 (a) Clocked JK flip-flop



Q_n	J	K	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

≡

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q_n}$

Fig. Q.9.1 (b) Logic symbol Fig. Q.9.1 (c) Truth table

Q_n \ JK	00	01	11	10
0	0	0	1	1
1	1	0	0	1

$Q_{n+1} = \overline{Q}_n J + Q_n \overline{K} = J \overline{Q}_n + \overline{K} Q_n$

Fig. Q.9.1 (d) Characteristics equation

Operation of JK flip-flop

Case 1 : J = K = 0

When J = K = 0, S = R = 0 and according to truth table of SR flip-flop there is no change in the output.

When inputs J = K = 0, output does not change.

Case 2 : J = 1 and K = 0

$Q = 0, \overline{Q} = 1$: When J = 1, K = 0 and Q = 0, S = 1 and R = 0. According to truth table of SR flip-flop it is set state and the output Q will be 1.

$Q = 1, \overline{Q} = 0$: When J = 1, K = 0 and Q = 1, S = 0 and R = 0. Since SR = 00, there is no change in the output and therefore, Q = 1 and $\overline{Q} = 0$.

The inputs J = 1 and K = 0, makes Q = 1, i.e. set state.

Case 3 : J = 0 and K = 1

$Q = 0, \overline{Q} = 1$: When J = 0, K = 1 and Q = 0, S = 0 and R = 0. Since SR = 00, there is no change in the output and therefore, Q = 0 and $\overline{Q} = 1$.

$Q = 1, \overline{Q} = 0$: When J = 0, K = 1 and Q = 1, S = 0 and R = 1. According to truth table of SR flip-flop it is a reset state and the output Q will be 0.

The inputs J = 0 and K = 1, makes Q = 0, i.e., reset state.

Case 4 : J = K = 1

$Q = 0, \overline{Q} = 1$: When J = K = 1 and Q = 0, S = 1 and R = 0. According to truth table of SR flip-flop it is a set state and the output Q will be 1.

$Q = 1, \overline{Q} = 0$: When J = K = 1 and Q = 1, S = 0 and R = 1. According to truth table of SR flip-flop it is a reset state and the output Q will be 0.

The input J = K = 1, toggles the flip-flop output.

Q.10 Discuss the race around condition and its solution.

[SPPU : Dec.-15, May-15, Marks 6]

OR What is race around condition ? Explain in brief.

Ans. : • In JK flip-flop, when J = K = 1, the output toggles (output changes either from 0 to 1 or from 1 to 0).

• Consider that initially Q = 0 and J = K = 1. After a time interval Δt equal to the propagation delay through two NAND gates in series, the output will change to Q = 1 and after another time interval of Δt the output will change back to Q = 0. This toggling will continue until the flip-flop is enabled and J = K = 1. At the end of clock pulse the flip-flop is disabled and the value of Q is uncertain. This situation is referred to as the **race-around condition.** This is illustrated in Fig. Q.10.1.

• This condition exists when $t_p \geq \Delta t$. Thus by keeping $t_p < \Delta t$ we can avoid race around condition.

• We can keep $t_p < \Delta t$ by keeping the duration of edge less than Δt.

• A more practical method for overcoming this difficulty is the use of the Master-Slave (MS) configuration.

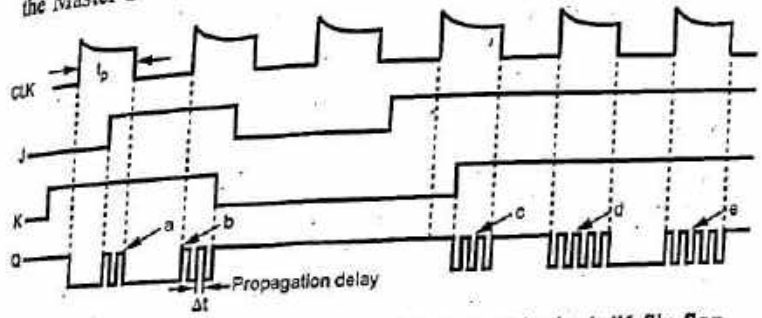


Fig. Q.10.1 Input and output waveforms for clocked JK flip-flop

Q.11 Explain the race condition in context of SR/RS flip-flop.

Ans. : In SR/RS flip-flop, when both inputs are logic (S = R = 1) and the clock pulse is applied, the state of the flip-flop is undefined. In this case, both the outputs Q and $\overline{Q}$ try to become 1. This violates the rule that the outputs (Q and $\overline{Q}$) of the flip-flop are complement of each other. Such condition is known as the race condition in context of SR/RS flip - flop.

Q.12 Explain the operation of T flip - flop.

Ans. : • T flip-flop is also known as **'Toggle flip-flop'.**

- As shown in the Fig. Q.12.1, the T flip-flop is obtained from a JK flip-flop by connecting both inputs, J and K together.

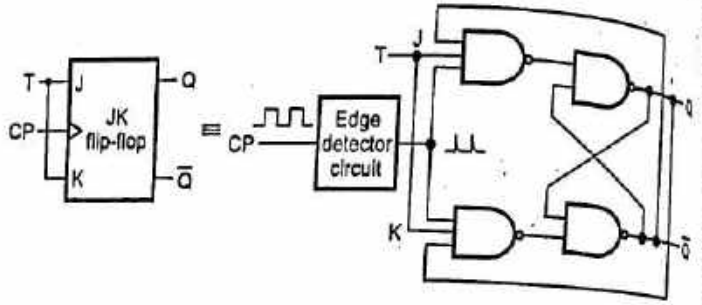
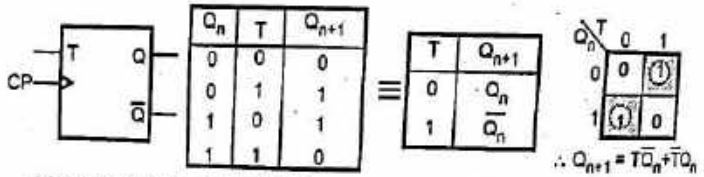


Fig. Q.12.1 T flip-flop using NAND gates

- When T = 0, J = K = 0 and hence there is no change in the output. When T = 1, J = K = 1 and hence output toggles.
- The Fig. Q.12.2 shows logic symbol, truth table and the characteristic equation for T flip-flop.



(a) Logic symbol (b) Truth table (c) Characteristic equation

Fig. Q.12.2

6.3 : Preset and Clear

Q.13 What are preset and clear ?

Ans. : When power is turn ON, the state of the flip-flop is uncertain. It may come to set (Q = 1) or reset (Q = 0) state. In many applications, it is necessary to initially set or reset the flip-flop. Such initial state of flip-flop can be accomplished by using the direct or asynchronous inputs of the flip-flop. These inputs are : Preset ($\overline{P}$) and Clear ($\overline{C}$). They can be applied at any time between clock pulses and are not in synchronism with the clock.

6.4 : Master and Slave Flip Flop

Q.14 Explain working of master - slave JK flip-flop with necessary logic diagram, state equation and state diagram.

[GTU : Winter-17, Marks 7]

OR Explain MS J-K flip-flop.

[GTU : May-14, Dec.-13, May-12, Winter-17, Marks 4]

Ans. : • Fig. Q.14.1 shows the master-slave JK flip-flop. Positive clock pulses are applied to first flip-flop and inverted (negative) clock pulses are applied to second flip-flop.

• When CK = 1, the first flip-flop is enabled and the outputs Q_M and $\overline{Q}_M$ responds to the inputs of J and K according to the Table Q.14.1. At this time, the second flip-flop is inhibited because its clock is low, $\overline{CK} = 0$.

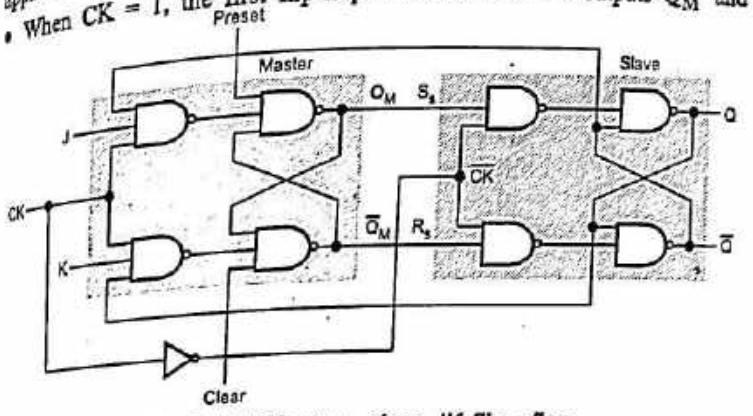


Fig. Q.14.1 Master - slave JK flip - flop

- When CK goes Low ($\overline{CK}$ = 1), the first flip-flop is inhibited and second flip-flop is enabled. At this time, the output of second flip-flop (Q and $\overline{Q}$) follow the outputs Q_M and $\overline{Q}_M$, respectively.
- Since the second flip-flop follows the first one, it is referred to as the slave and the first one as the master.
- In master-slave JK flip-flop state change occurs when flip-flop goes through both positive transition (first half) of clock and negative transition of the clock (second half). Thus, race-around condition does not exist in the master-slave JK flip-flop.

Q_n	J	K	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

≡

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q_n}$

Table Q.14.1 Truth table

Characteristics equation : $Q_{n+1} = \overline{Q}_n J + Q_n \overline{K} = J\overline{Q}_n + \overline{K}Q_n$

State Diagram :

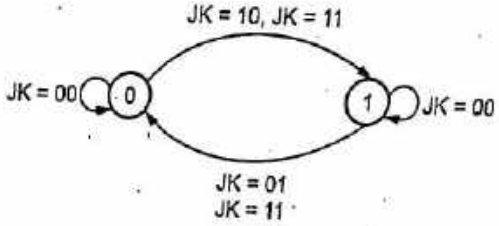


Fig. Q.14.2

6.5 : Excitation Tables

Q.15 What do you mean by excitation table ? Give excitation table of SR, JK, D and T flip-flops ?

Ans. : • The table that gives the required inputs of the flip-flop for a given change of state is known as an excitation table of the flip-flop.

Q_n	Q_{n+1}	S	Ra
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

SR excitation table
Table Q.15.1

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

JK excitation table
Table Q.15.2

Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

D excitation table
Table Q.15.3

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Table Q.15.4

6.6 : Conversion from One FF to Another

Q.16 Explain how SR-FF is converted into D FF.

[SPPU : May-12, Marks 2]

Ans. : The excitation table for above conversion is as shown in Table Q.16.1.

Input	Present state	Next state	Flip-flop inputs	
D	Q_n	Q_{n+1}	S	R
0	0	0	0	X
0	1	0	0	1
1	0	1	1	0
1	1	1	X	0

Table Q.16.1

K- map simplification

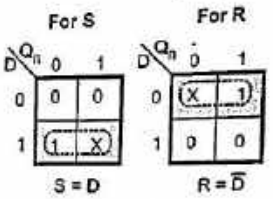


Fig. Q.16.1

Logic diagram

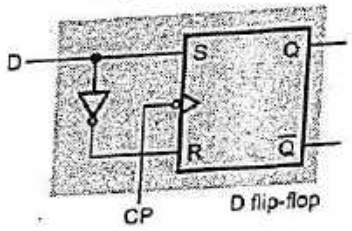


Fig. Q.16.2 SR to D flip-flop conversion

Q.17 Explain how SR-FF is converted to JK FF.

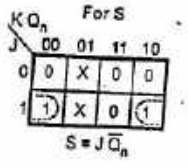
☞ [SPPU : May-12, 15, Marks]

Ans. : The excitation table for above conversion is as shown Table Q.17.1.

Inputs		Present state	Next state	Flip-flop inputs	
J	K	Q_n	Q_{n+1}	S	R
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	X	0
1	1	0	1	1	0
1	1	1	0	0	1

Table Q.17.1

K- map simplification



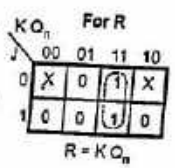


Fig. Q.17.1

Logic diagram

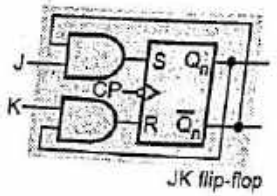


Fig. Q.17.2 SR to JK flip-flip conversion

Q.18 Explain how JK FF is converted to D FF.

☞ [SPPU : May-16, Marks 3]

Ans. : The excitation table for above conversion is as shown in the Table Q.18.1.

Input	Present state	Next state	Flip-flop inputs	
D	Q_n	Q_{n+1}	J	K
0	0	0	0	X
0	1	0	X	1
1	0	1	1	X
1	1	1	X	0

Table Q.18.1

K- map simplification

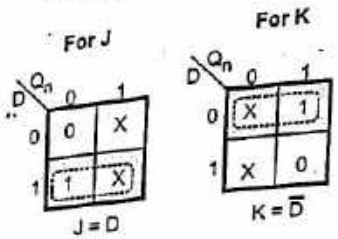


Fig. Q.18.1

Logic diagram

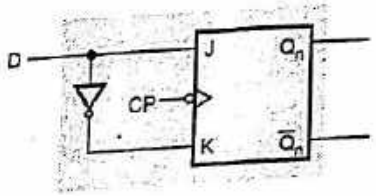


Fig. Q.18.2 JK to D flip-flip conversion

Q.19 Explain how D FF is converted to T FF.

☞ [SPPU : June-22, Marks 9]

Ans. : The excitation table for above conversion is as shown in the Table Q.19.1.

Input	Present state	Next state	Flip-flop input
T	Q_n	Q_{n+1}	D
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

Table Q.19.1

K- map simplification

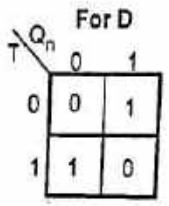


Fig. Q.19.1

Logic diagram

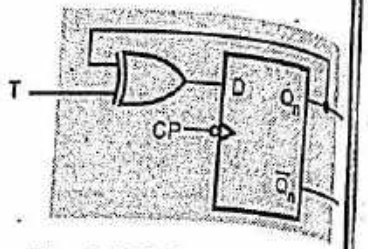


Fig. Q.19.2 D to T flip-flop conversion

$$D = \overline{T}Q_n + T\overline{Q}_n = T \oplus Q_n$$

Q.20 Explain how to convert SR flip-flop to T flip-flop ?

☞ [SPPU : May-06, Dec.-16, Marks 3]

Ans. : The excitation table for above conversion is as shown in the Table Q.20.1.

Input	Present state	Next state	Flip-flop inputs	
T	Q_n	Q_{n+1}	S	R
0	0	0	0	X
0	1	1	X	0
1	0	1	1	0
1	1	0	0	1

Table Q.20.1

K- map simplification

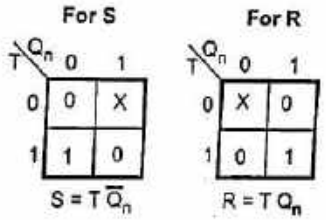


Fig. Q.20.1

Logic diagram

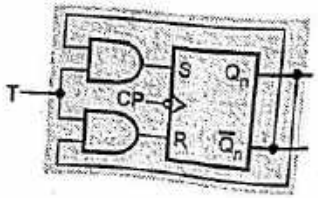


Fig. Q.20.2 SR to T flip-flop conversion

Q.21 Explain how to convert JK flip-flop to T flip-flop ?

☞ [SPPU : May-06,16,17 Marks 3]

Ans. : The excitation table for above conversion is as shown in Table Q.21.1.

Input	Present state	Next state	Flip-flop inputs	
T	Q_n	Q_{n+1}	J_A	K_A
0	0	0	0	X
0	1	1	X	0
1	0	1	1	X
1	1	0	X	1

Table Q.21.1

K- map simplification

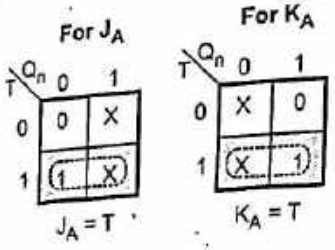


Fig. Q.21.1

Logic diagram

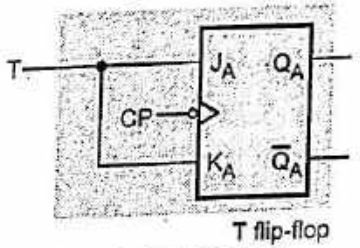


Fig. Q.21.1 JK to T flip-flop conversion

Q.22 Write the Excitation Table of S-R flip-flop. Prepare the Truth Table for the following circuit and Determine the type of flip-flop.

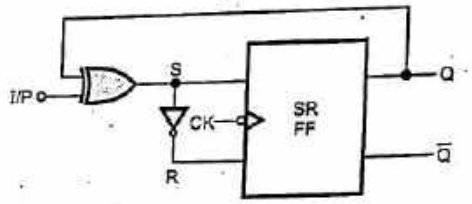


Fig.Q.22.1

[6]

Ans. : Refer Table Q.15.1

I/P	Q_n	S	R	Q_{n+1}	
0	0	0	1	0	When input is 0 output does not change
0	1	1	0	1	
1	0	1	0	1	When input is 1 output toggles
1	1	0	1	0	

∴ Type of flip-flop is T

Truth Table

∴ Type of flip-flop is T.

6.7 : Study of 7474 and 7476 Flip-Flop ICs

Q.23 Write a short note on IC 7474.

Ans. : IC 7474 contains two independent positive-edge-trig... D flip-flops with complementary outputs. The information on the D ... is accepted by the flip-flops on the positive going edge of the clock pu... A low logic level on the preset or clear inputs will set or reset the o... regardless of the logic levels of the other inputs.

Inputs				Outputs	
PR	CLR	CLK	D	Q	$\overline{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H*	H*
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q_0	$\overline{Q}_0$

*This configuration is nonstable; that is, it will not persist when either the preset and/or clear inputs return to their inactive (high) level.

Table Q.23.1 Function table

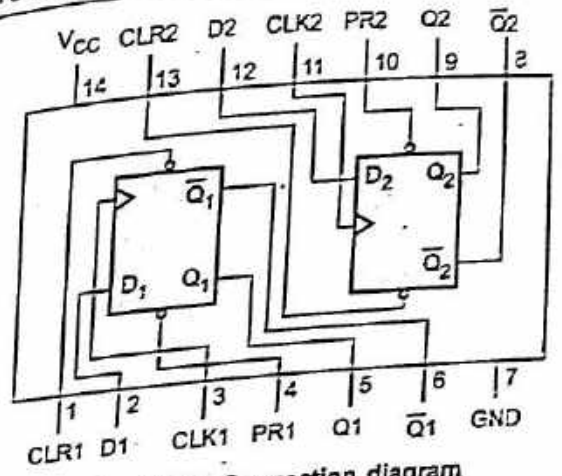


Fig. Q.23.1 Connection diagram

Q.24 Write a short note on IC 7476.

Ans. : IC 7476 contains two independent positive pulse triggered J-K flip-flops with complementary outputs. The J and K data is processed by the flip-flops after a complete clock pulse. While the clock is low the slave is isolated from the master. On the positive transition of the clock, the data form the J and K inputs is transferred to the master. While the clock is high the J and K inputs are disabled. On the negative transition of the clock, the data from the master is transferred to the slave. The logic state of J and K inputs must be allowed to change while the clock is high. The data is transferred to the outputs on the falling edge of the clock pulse. A low logic level on the preset or clear inputs will set or reset the outputs regardless of the logic levels of the other inputs.

Inputs					Outputs	
PR	CLR	CLK	J	K	Q	$\overline{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H*	H*
H	H		L	L	Q_0	$\overline{Q}_0$

H	H		H	L	H	L
H	H		L	H	L	H
H	H		H	H	Toggle	

*This configuration is nonstable; that is, it will not persist when the preset and clear inputs return to their inactive (high) level.

Table Q.24.1 Function table

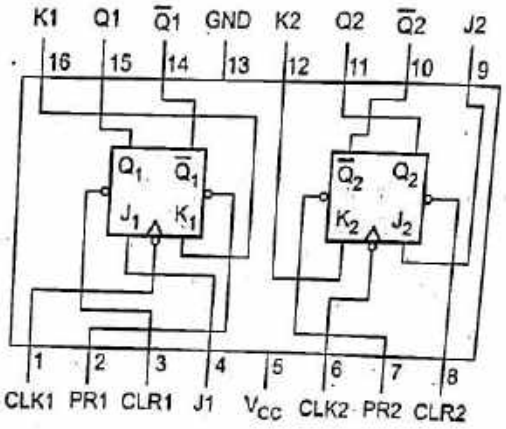


Fig. Q.24.1 Connection diagram

END...E

Unit 3

7 Applications of Flip-Flops-Registers

7.1 : Buffer Register and Shift Register

Q.1 What is buffer register ?

Ans. : Fig. Q.1.1 shows the simplest register constructed with four D flip-flops. This register is also called **buffer register**. Each D flip-flop is triggered with a common negative edge clock pulse. The input bits set up the flip-flops for loading. Therefore, when the first negative clock edge arrives, the stored binary information becomes,

$Q_A Q_B Q_C Q_D$ = ABCD

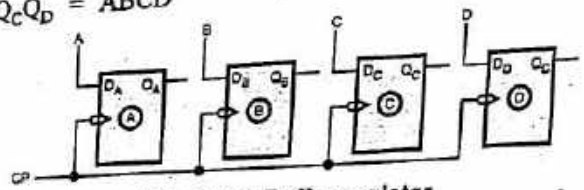


Fig. Q.1.1 Buffer register

In this register, four D flip-flops are used. So it can store 4-bit binary information. Thus the number of flip-flop stages in a register determines its total storage capacity.

Q.2 What are shift registers ? [SPPU : June-22, Marks 4]

Ans. : • A group of flip-flops can be used to store a word, which is called register.

- The binary information (data) in a register can be moved from stage to stage within the register or into or out of the register upon application of clock pulses. Such registers are called **'shift registers'**.

Q.3 Explain the operational types of shift register. [SPPU : June-22, Marks 5]

Ans. : Fig. Q.3.1 gives the symbolical representation of the different types of data movement in shift register operations.

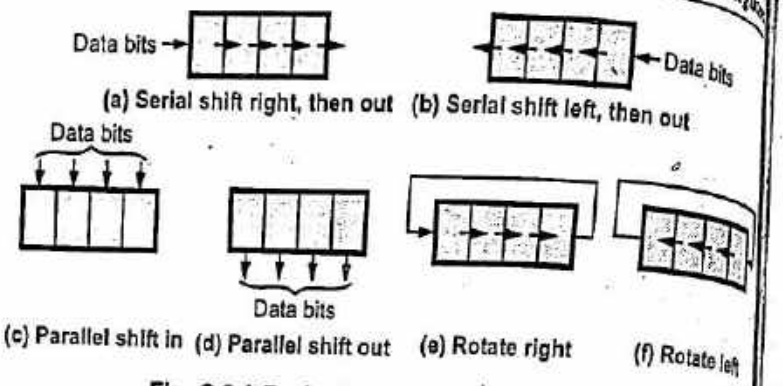


Fig. Q.3.1 Basic data movement in registers

Q.4 Explain the operation of SISO shift register.

[SPPU : Dec.-14, Marks 6]

Ans. : Fig. Q.4.1 shows serial-in serial-out shift-left register.

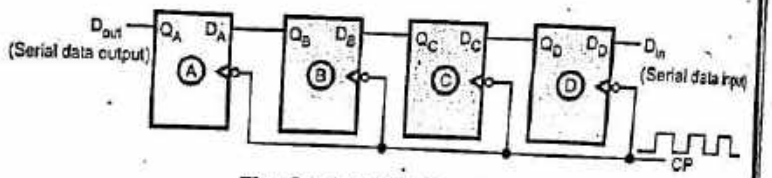


Fig. Q.4.1 Shift-left register

In this shift register, data within the shift register is shifted left one bit position at each clock pulse. The data input bit is loaded in the right most flip-flop.

Fig. Q.4.2 shows serial-in serial-out shift-right register.

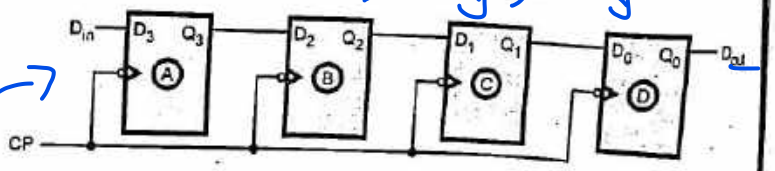


Fig. Q.4.2 Shift-right register

In this shift register, data within the shift register is shifted right one bit position at each clock pulse. The data input bit is loaded in the left most flip-flop.

Q.5 Explain the operation of SIPO shift register.

[SPPU : Dec.-12, Marks 5]

Ans. : • In SIPO, the data bits are entered serially into the register but the output is taken in parallel.

• Once the data are stored, each bit appears on its respective output line and all bits are available simultaneously as shown in Fig. Q.5.1.

CP	Q_3	Q_2	Q_1	Q_0
–	NC	NC	NC	NC

Table Q.5.1 Truth table

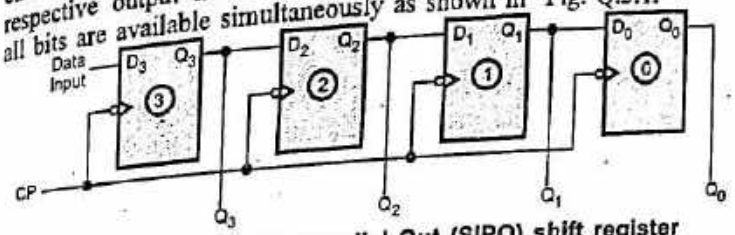


Fig. Q.5.1 A Serial In Parallel Out (SIPO) shift register

Q.6 Explain the operation of PISO shift register.

[SPPU : May-10, 14, Marks 6]

Ans. : • Fig. Q.6.1 illustrates a four-bit parallel in serial out register.

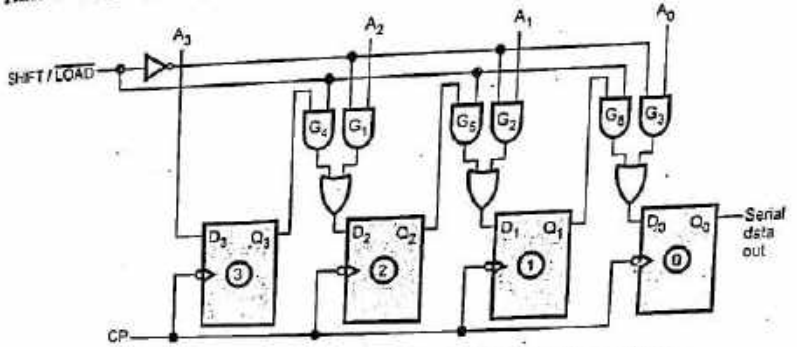


Fig. Q.6.1 Parallel In Serial Out (PISO) shift register

- There are four input lines A_3, A_2, A_1, A_0 for entering data in parallel into the register.
- SHIFT/$\overline{\text{LOAD}}$ is the control input which allows shift or loading data operation of the register.
- When SHIFT/$\overline{\text{LOAD}}$ is low, gates G_1, G_2, G_3 are enabled, allowing each input data bit to be applied to D input of its respective flip-flop.
- When a clock pulse is applied, the flip-flops with D = 1 will SET and those with D = 0 will RESET.

- All four bits are stored simultaneously.
- When SHIFT/$\overline{LOAD}$ is high, gates G_1, G_2, G_3 are disabled and G_4, G_5, G_6 are enabled. This allows the data bits to shift right from one stage to the next.
- The OR gates at the D-inputs of the flip-flops allow either the parallel data entry operation or shift operation, depending on which AND gates are enabled by the level on the SHIFT/$\overline{LOAD}$ input.

Q.7 Explain parallel in parallel out shift register.

Ans. : • In 'parallel in parallel out register', there is simultaneous entry of all data bits and the bits appear on parallel outputs simultaneously.

- Fig. Q.7.1 shows this type of register.

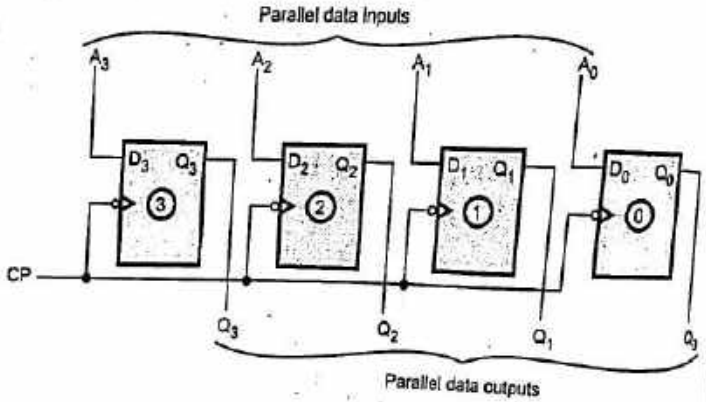


Fig. Q.7.1 Parallel In Parallel Out (PIPO) shift register

Q.8 Design a 4-bit serial-in-serial-out shift register using JK flip-flops.

Ans.: In Q.4 we have seen 4-bit serial-in serial-out shift registers using D Flip-Flops. By replacing D flip-flop using equivalent JK flip-flops we can implement SISO shift register using JK flip-flops. By giving complement inputs to JK flip-flops we can use it as a D flip-flop.

Q.9 Draw and explain the working of universal shift register.

☞ [SPPU : Dec.-06, Marks 7, Dec.-07, Marks 8]

Ans. : • If the register has both shifts (right shift and left shift) and parallel load capabilities, it is referred to as **universal shift register.**

- The Fig. Q.9.1 shows the 4-bit universal shift register.

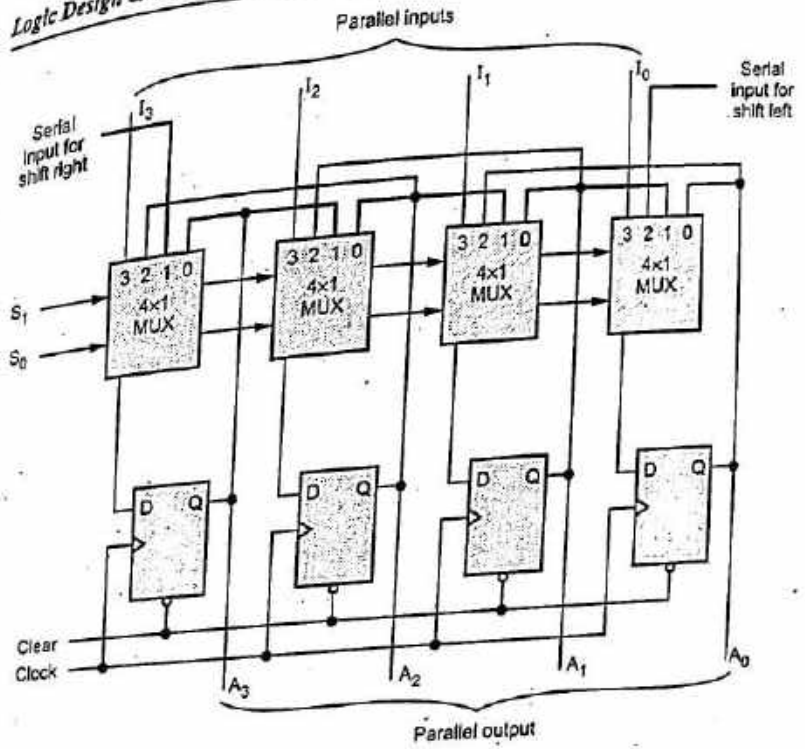


Fig. Q.9.1 4-bit universal shift register

- It consists of four flip-flops and four multiplexers.
- The four multiplexers have two common selection inputs S_1 and S_0, and they select appropriate input for D flip-flop.
- The Table Q.9.1 shows the register operation depending on the selection inputs of multiplexers.
- When S_1S_0 = 00, input 0 is selected and the present value of the register is applied to the D inputs of the flip-flops. This results no change in the register value.

Mode control		Register operation
S_1	S_0	
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load

Table Q.9.1 Mode control and register operation

- When S_1S_0 = 01, input 1 is selected and circuit connections are such that it operates as a right shift register.
- When S_1S_0 = 10, input 2 is selected and circuit connections are such that it operates as a left shift register.
- Finally, when S_1S_0 = 11, the binary information on the parallel input lines is transferred into the register simultaneously and it is a parallel load operation.

7.2 : Applications of Shift Registers

Q.10 What are the applications of shift register ?

Ans. : • A Serial-In-Serial-Out (SISO) shift register can be used to introduce time delay Δt in digital signals.

- A Serial-In-Parallel-Out (SIPO) shift register can be used to convert data in the serial form to the parallel form.
- A Parallel-In-Serial-Out (PISO) shift register can be used to convert data in the parallel form to the serial form.
- A shift register can also be used as a counter.
- Shift register is a pseudo-random binary sequence generator.
- The shift register can be used to generate a particular bit pattern repetitively.
- The shift register can be used to detect the desired sequence.

7.3 : Sequence Detector using Shift Register

Q.11 Write a note on IC 74194.

Ans. : We know that a register may operate in any of the modes, like, SISO, SIPO, PISO, PIPO or bidirectional. If a register can be operated in all the five possible ways, it is known as **Universal Shift Register**. The IC 74194 is a 4-bit universal shift register. Fig. Q.11.1 shows the pin configuration of IC 74194.

As shown in the Fig. Q.11.1, 74194 has 4 parallel data inputs (D_0 - D_3), and S_0 and S_1 are the control inputs.

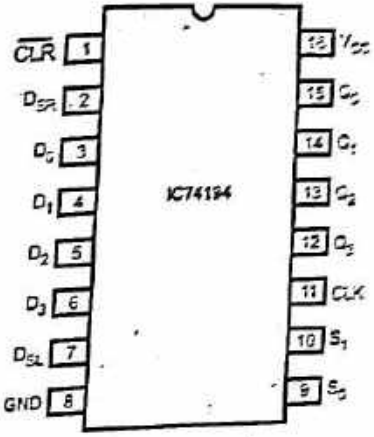


Fig. Q.11.1 Pin configuration

Operating Modes

S_1	S_0	Operation	Description
Inputs			
0	0	Hold	Do nothing.
0	1	Shift right	Serial data is entered at the shift-right serial input, DSR.
1	0	Shift left	Serial data is entered at the shift-left serial input, DSL.
1	1	Parallel load	Data appearing on D_0-D_3 inputs is transferred to the Q_0-Q_3 outputs, respectively.

Table Q.11.1 Operating modes of 74194

Q.12 Explain the design process of sequence generator using shift register.

Ans. : The simplest way of designing sequence generator using shift register is to take shift register of n-bits where n is equal to the length of sequence. Then load the bit sequence in the shift register by parallel load operation and apply the clock signal. The Fig. Q.12.1 illustrates the operation of such a sequence generator. Here, the sequence to be generated is 11001.

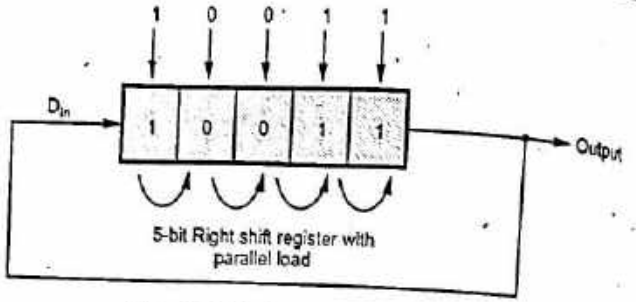


Fig. Q.12.1 Sequence generator

Another design approach is used for sequence generator to reduce the required number of flip-flop stages in the shift register. In this approach a shift register with a next state decoder and preset logic is used. The Fig. Q.12.1 shows the block diagram of this approach. Here, the output of the next state decoder is a function of Q_A, Q_B, ... Q_n and it is used to determine the D_{in} input for the shift right register. Initially, start button is pressed to activate parallel load operation. This loads initial value in the shift register. Then at each clock pulse shift register contents are shifted right by 1 bit position. The next state decoder circuit decodes the output and generates D_{in} input for the shift register such that output Q_A generates the desired sequence. After completion of one complete sequence, the register is again loaded with initial value to start the next train of sequence.

After completion of one complete sequence, the register is again loaded with initial value to start the next train of sequence.

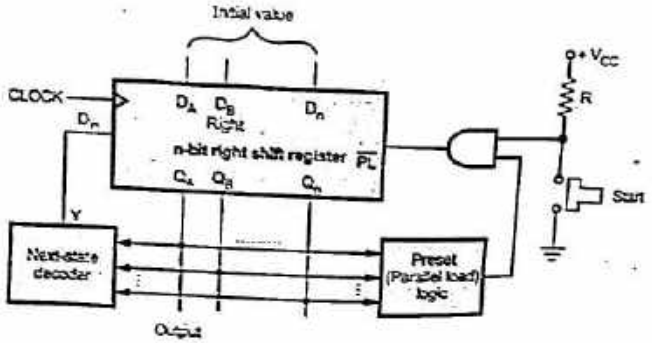


Fig. Q.12.2 Sequence generator using shift register

Q.13 Design a sequence generator to generate the sequence 1101011 by shift register method. [May-10, 15, Marks 8]

Ans. : In this approach, the minimum number of flip-flops n, required to generate a sequence of length N is given by

$$N \leq 2^n - 1$$

In this example, N = 7, therefore, the minimum value of n, which may generate this sequence is 3. However, it is not guaranteed to lead to a solution. Let us try with 3 flip-flops. The Table Q.13.1 shows sequence generation with three flip-flops.

As shown in the Table Q.13.2, the state 6 is repeated. This means that n = 3 is not sufficient. Let us try with 4 flip-flops. The table shows sequence generation with four flip-flops.

CP	Flip-flop outputs			D_{in}	States
	Q_A	Q_B	Q_C		
1	1	0	0	1	4
2	1	1	0	0	6
3	0	1	1	1	3
4	1	0	1	0	5
5	0	1	0	1	2
State is repeated → 6	1	0	1	–	5
–	–	–	–	–	–

Table Q.13.1

	CP	Flip-flop outputs				D_{in}	$\overline{Preset}$	States
		Q_A	Q_B	Q_C	Q_D			
Initial value	1	1	0	0	0	1	1	8
	2	1	1	0	0	0	1	12
	3	0	1	1	0	1	1	6
Sequence	4	1	0	1	1	0	1	11
	5	0	1	0	1	1	1	5
	6	1	0	1	0	1	1	10
	7	1	1	0	1	1	1	13
Preset flip-flop	1	1	1	1	0	X	0	14
	1	1	0	0	0	1	1	8

Table Q.13.2

As shown in the Table Q.13.2, states are not repeated. After completion of one complete sequence, register is preset to value 1000 to start the next train of sequence. Fig. Q.13.1 (b) shows the logic diagram.

K-map simplification for D_{in}

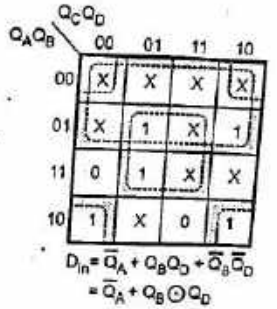


Fig. Q.13.1 (a)

Logic diagram

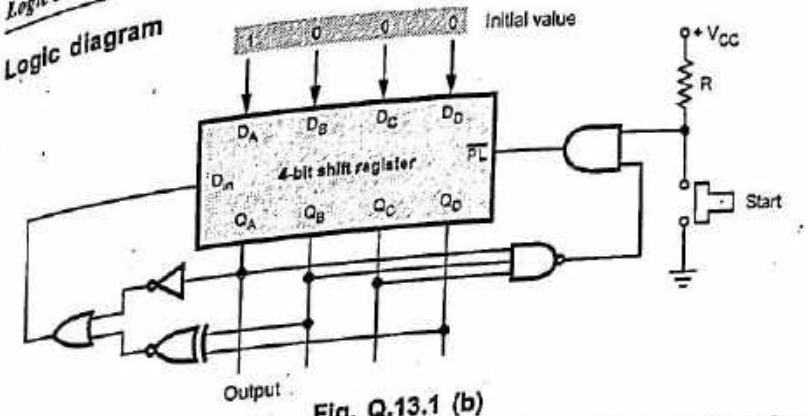


Fig. Q.13.1 (b)

Q14 Design a sequence generator using shift register and decoder circuit to generate the sequence .....1101011.....

[May-15, Dec.-16, Marks 6]

Ans. : Referring Table Q.13.2 we have,

Here, the combinational logic to derive D_{in} is implemented using decoder. Same decoder is used to generate preset signal.

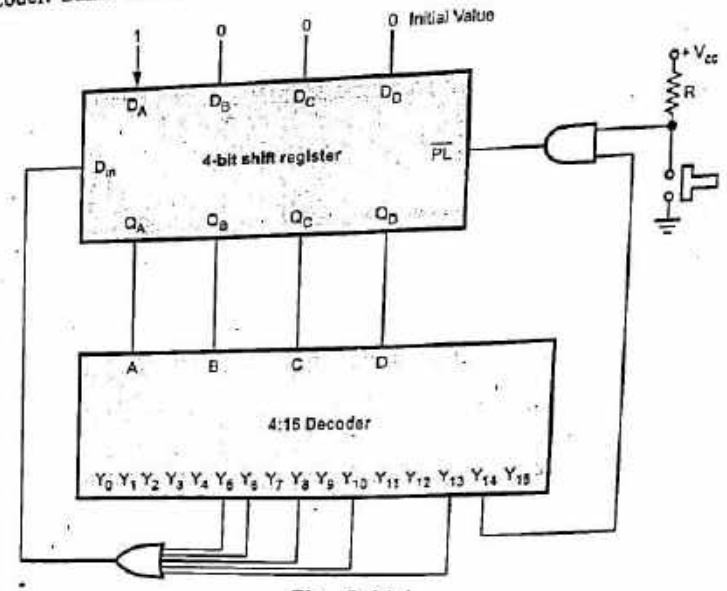


Fig. Q.14.1

Q.15 Design a sequence generator to generate the sequence .....10110 .....using IC 74194. [Dec.-15, Marks]

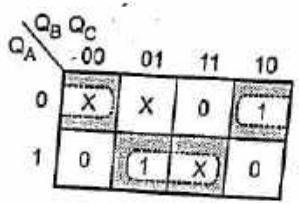
Ans. : The minimum number of flip-flops n can be given as,

$$N \leq 2^{n-1}$$

Here N = 3, therefore n = 3. Table shows sequence generation with 3 flip flops with Q_0 as output.

	CP	Flip-flop outputs			DSR	S_1	S_0	States	Operation
		Q_0	Q_1	Q_2					
	1	1	0	0	0	0	1	4	Shift right
Sequence	2	0	1	0	1	0	1	2	Shift right
	3	1	0	1	1	0	1	5	Shift right
	4	1	1	0	0	0	1	6	Shift right
	5	0	1	1	0	0	1	3	Shift right
Preset flip-flop	6	0	0	1	X	1	1	1	Parallel load
	7	1	0	0	0	0	1	4	Shift right

K-map simplification for D_{in}



$$D_{in} = \overline{Q}_A \overline{Q}_C + Q_A Q_C$$
$$= Q_A \odot Q_A$$

Fig. Q.15.1 (a)

When start button is pressed $S_0 = S_1 = 1$ and 74194 operates in parallel load mode. Therefore, $Q_0\ Q_1\ Q_2 = D_0\ D_1\ D_2$. When $S_0 = 1$ and $S_1 = 0$, IC 74194 operates in shift right mode; it goes through states 4, 2, 5, 6, 3 and 1. When 1 state is reached S_1 input is made logic 1 and parallel load operation is activated. This sequence is repeated to get desired pulse sequence (10 11 0) at Q_0 output.

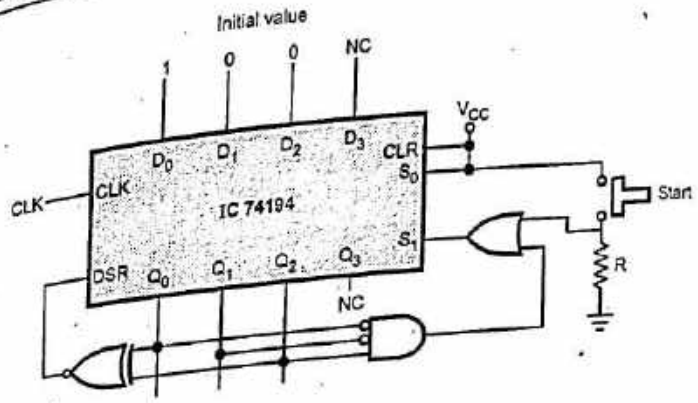


Fig. Q.15.1 (b)

Example for Practice

Q.16 Design sequence generator to generate the sequence 1011 using shift register IC 74194. [May-17, Marks 6]

END...

Unit 3

8 Applications of Flip-Flops-Counter

8.1 : Introduction

Q.1 What is counter ? Give the difference between synchronous and asynchronous counters. ☞ [SPPU : May-07, Dec.-07, Marks 2]

Ans. : • A counter is a register capable of counting the number of clock pulses arriving at its clock input.

• There are two types of counters : synchronous counter and asynchronous counter.

Sr. No.	Asynchronous counters	Synchronous counters
1.	In this type of counter flip-flops are connected in such a way that output of first flip-flop drives the clock for the next flip-flop.	In this type there is no connection between output of first flip-flop and clock input of the next flip-flop.
2.	All the flip-flops are not clocked simultaneously.	All the flip-flops are clocked simultaneously.
3.	Logic circuit is very simple even for more number of states.	Design involves complex logic circuit as number of states increases.
4.	Main drawback of these counters is their low speed as the clock is propagated through number of flip-flops before it reaches last flip-flop.	As clock is simultaneously given to all flip-flops there is no problem of propagation delay. Hence they are high speed counters and are preferred when number of flip-flops increases in the given design.

Table Q.1.1 Synchronous Vs asynchronous counters

8.2 : Asynchronous (Ripple Counters)

Q.2 Draw and explain the working of 2-bit asynchronous binary counter.

Ans. : • Fig. Q.2.1 (a) shows 2-bit asynchronous counter using JK flip-flops.

• The clock signal is connected to the clock input of only first stage flip-flop.

• The clock input of the second stage flip-flop is triggered by the Q_A output of the first stage.

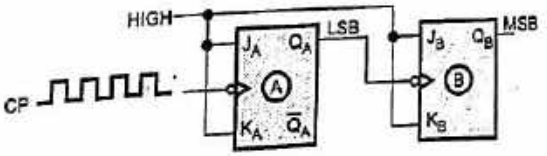


Fig. Q.2.1 (a) A two-bit asynchronous binary counter

• Because of the inherent propagation delay time through a flip-flop, a transition of the input clock pulse and a transition of the Q_A output of first stage can never occur at exactly the same time. Therefore, the two flip-flops are never simultaneously triggered, which results in asynchronous counter operation.

• Fig. Q.2.1 (b) shows the timing diagram for two-bit asynchronous counter. It illustrates the changes in the state of the flip-flop outputs in response to the clock.

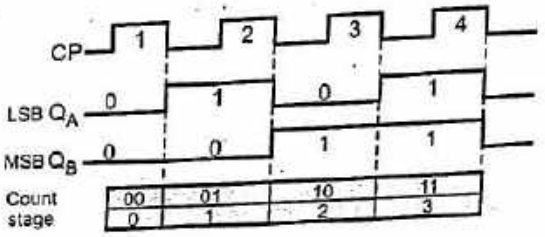


Fig. Q.2.1 (b) Timing diagram for the counter of Fig. Q.2.1 (a)

• J and K input of JK flip-flops are tied to logic HIGH hence output will toggle for each negative edge of the clock input.

Q.3 Explain the working of 4-bit asynchronous down counter.

OR Draw diagram of a 4-bit binary ripple counter using flip - flops that trigger on negative edge transition. Also draw a timing diagram of the counter.

Ans. : • The Fig. Q.3.1 shows the 4-bit asynchronous down counter using JK flip-flops.

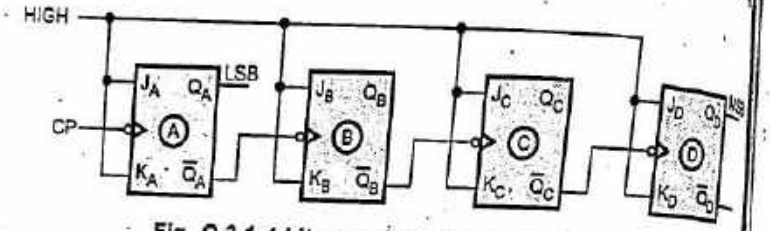


Fig. Q.3.1 4-bit asynchronous down counter

- The clock signal is connected to the clock input of only first flip-flop.
- The clock input of the remaining flip-flops is triggered by the $\overline{Q}$ output of the previous stage instead of Q_A output of the previous stage.

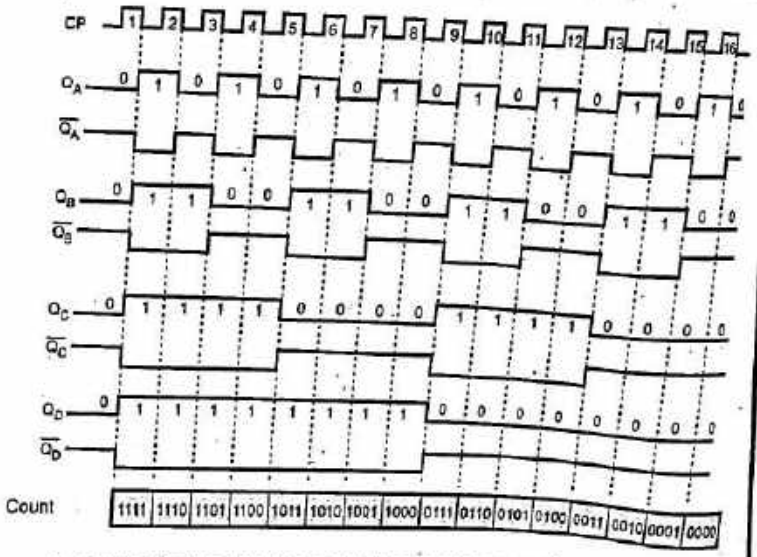


Fig. Q.3.2 Timing diagram of 4-bit asynchronous down counter

- The Fig. Q.3.2 shows the timing diagram for 4-bit asynchronous down counter. It illustrates the changes in the state of the flip-flop outputs in response to the clock.
- The J and K inputs of JK flip-flops are tied to logic HIGH hence output will toggle for each negative edge of the clock input.

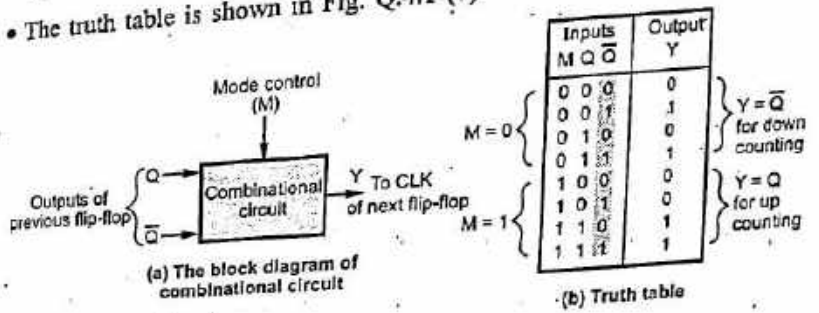
Q.4 Design a 3-bit asynchronous up-down counter.

OR Design a 3-bit binary up/down counter. Draw its timing diagram.

[SPPU : Dec.-04, 12, Marks 8]

Ans. : • To form an asynchronous up/down counter one control input say M is necessary to control the operation of the up/down counter.

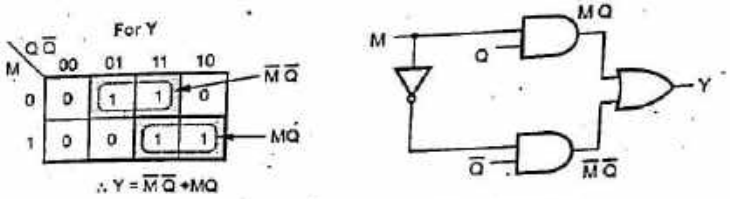
- When M = 0, the counter will count up and when M = 1, the counter will count down. To achieve this the M input should be used to control whether the normal flip-flop output (Q) or the inverted flip-flop output ($\overline{Q}$) is fed to drive the clock signal of the successive stage flip-flop, as shown in Fig. Q.4.1 (a).
- The truth table is shown in Fig. Q.4.1 (b).



(a) The block diagram of combinational circuit

(b) Truth table

Fig. Q.4.1



a) K-map simplification

b) Logic diagram

Fig. Q.4.2

- Fig. Q.4.3 shows the 3-bit up/down counter that will count from 000 to 111 when the mode control input M is 1 and from 111 down to 000 when mode control input M is 0.

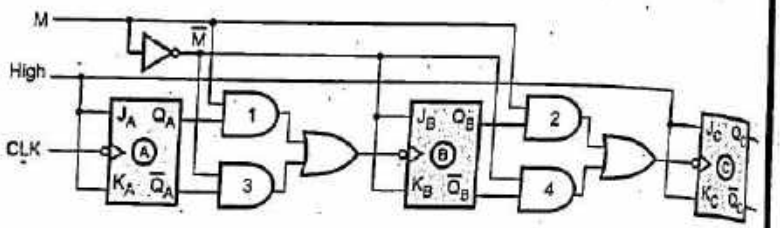


Fig. Q.4.3 3-bit asynchronous up/down counter

- A logic 1 on M enables AND gates 1 and 2 and disables AND gates 3 and 4. This allows the Q_A and Q_B outputs to drive the clock inputs of their respective next stages. So that counter will count up.
- When M is logic 0, AND gates 1 and 2 are disabled and AND gate 3 and 4 are enabled. This allows the $\overline{Q}_A$ and $\overline{Q}_B$ outputs to drive the clock inputs of their respective next stages so that counter will count down.
- Fig. Q.4.4 shows the timing diagram for 3-bit up/down ripple counter. (See Fig. Q.4.4 on next page page.)

Q.5 Design BCD (mod-10) ripple counter using JK flip-flop.

Ans. : Step 1 : Determine the number of flip-flops needed. The BCD counter goes through states 0-9, i.e. total 10 states. Thus, N = 10 and for $2^n \geq N$, we need n = 4, i.e. 4 flip-flops required.

Step 2 : Type of flip-flops to be used : JK

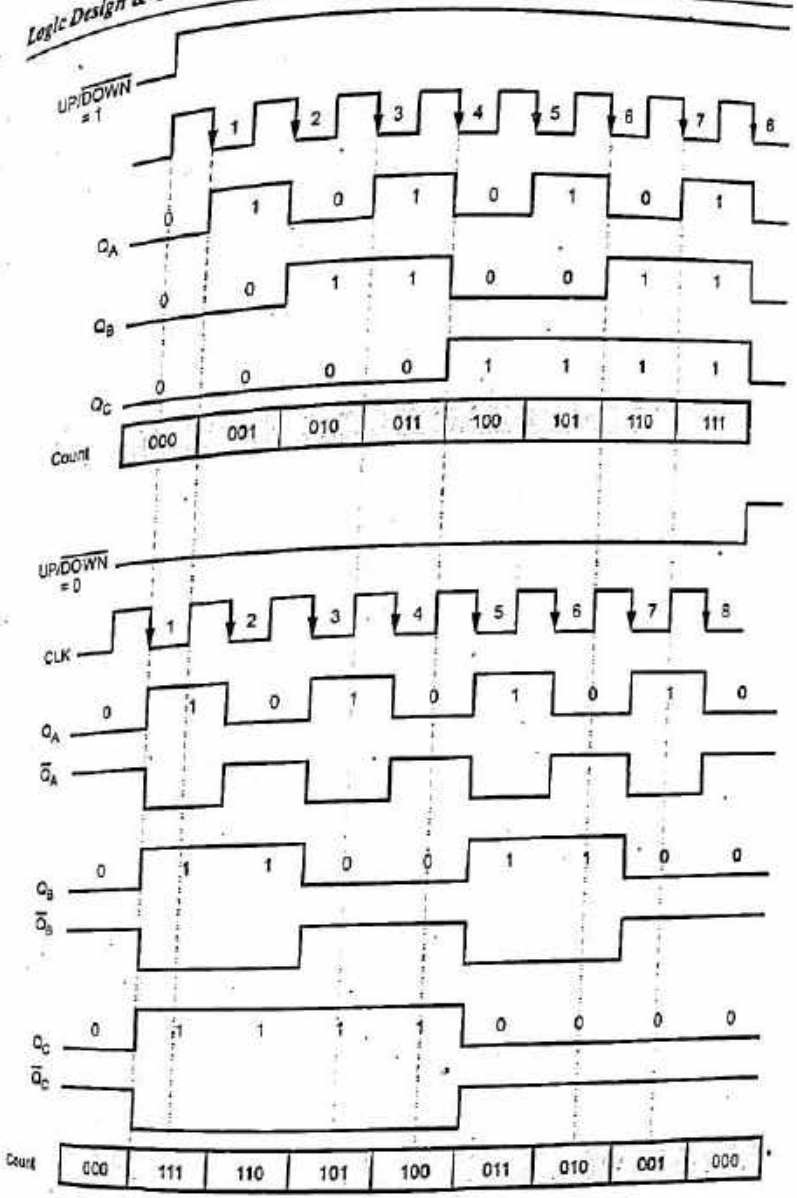


Fig. Q.4.4 Timing diagram for 3-bit UP/ $\overline{\text{DOWN}}$ ripple counter

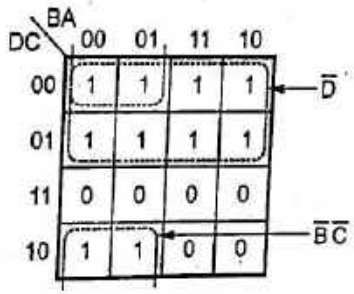
Step 3 : Write the truth table for the counter

CLK	D	C	B	A	Output of reset logic Y	
0	0	0	0	0	1	Valid states
1	0	0	0	1	1	
2	0	0	1	0	1	
3	0	0	1	1	1	
4	0	1	0	0	1	
5	0	1	0	1	1	
6	0	1	1	0	1	
7	0	1	1	1	1	
8	1	0	0	0	1	
9	1	0	0	1	1	
–	1	0	1	0	0	Invalid states
–	1	0	1	1	0	
–	1	1	0	0	0	
–	1	1	0	1	0	
–	1	1	1	0	0	
–	1	1	1	1	0	

Note : The reset input (CLR) of each Flip-Flop is active-low input. By making CLR input of all Flip-Flops logic 0, we can reset the counter. Thus reset logic is designed such a way that for invalid states, Y = 0 and counter resets.

Table Q.5.1 Truth table for BCD counter

Step 4 : Derive reset logic



$\therefore Y = \overline{D} + \overline{B}\overline{C}$.

Fig. Q.5.1

Step 5 : Draw logic diagram.

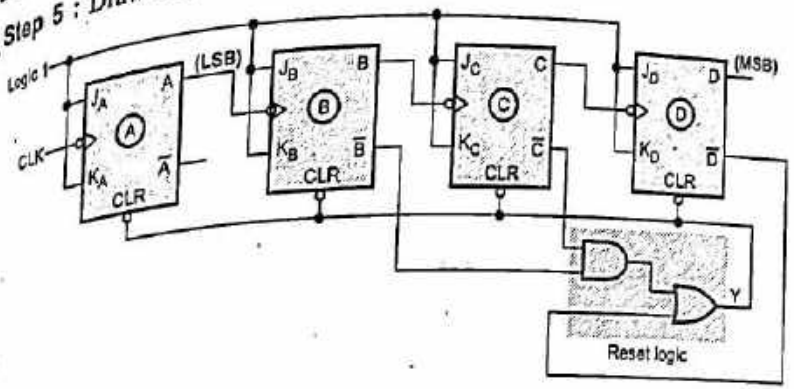


Fig. Q.5.2 Logic diagram of BCD ripple counter

Q.6 Design mod 6 ripple counter using T flip-flops.

Ans. : Step 1 : Determine the number of flip-flop required. Here, counter goes through 0 - 5 states, i.e., total 6 states. Thus N = 6 and for $2^n \geq N$ we need n = 3, i.e. 3 flip-flops.

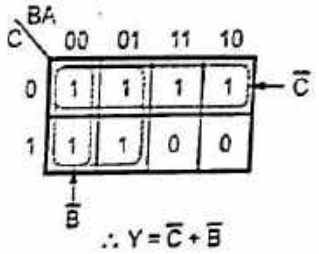
Step 2 : Type of flip-flops to be used : T

Step 3 : Write the truth table for counter

CLK	C	B	A	Output of reset logic Y	
0	0	0	0	1	Valid states
1	0	0	1	1	
2	0	1	0	1	
3	0	1	1	1	
4	1	0	0	1	
5	1	0	1	1	
–	1	1	0	0	Invalid states
–	1	1	1	0	

Fig. Q.6.1

Step 4 : Derive reset logic



$\therefore Y = \overline{C} + \overline{B}$

Fig. Q.6.2

Step 5 : Draw logic diagram

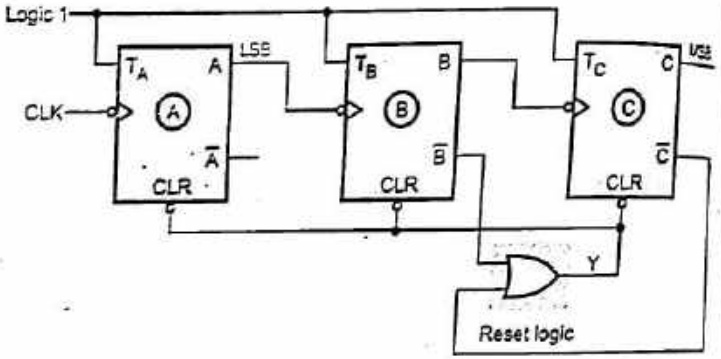


Fig. Q.6.3

Q.7 Design ripple counter for state diagram shown.

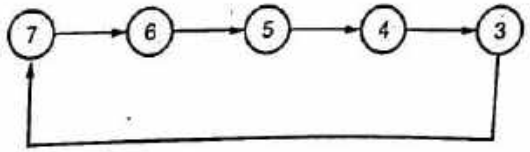


Fig. Q.7.1

Ans. : Step 1 : Determine the number of flip-flops needed.

We know that $2^n \geq N$. Here, $N = 8$ $\therefore$ $n = 3$ i.e. 3 flip-flops needed.

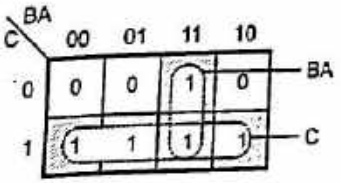
Step 2 : Type of flip-flops to be used : JK

Step 3 : Write truth table for counter.

CLK	C	B	A	Output of reset logic Y
0	1	1	1	1
1	1	1	0	1
2	1	0	1	1
3	1	0	0	1
4	0	1	1	1
5	0	1	0	0
6	0	0	1	0
7	0	0	0	0

Table Q.7.1 Truth table

Step 4 : Derive preset logic. Since it is a down counter we need to derive preset logic instead of reset logic.



$Y = BA + C$

Fig. Q.7.2

Step 5 : Draw logic diagram.

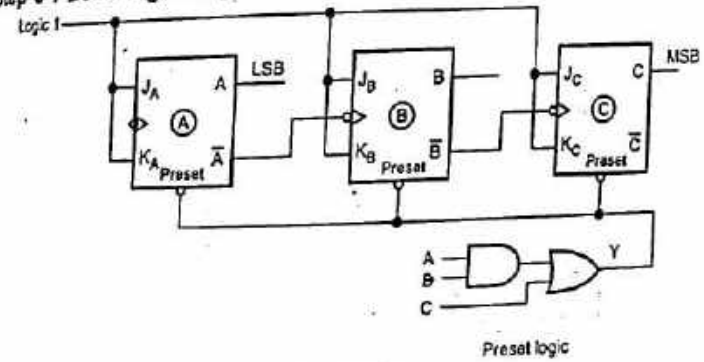


Fig. Q.7.3

- Since it is a down counter, the clock of the next flip-flop is given by the $\overline{Q}$ output of the previous flip-flop.

8.3 : Synchronous Counters

Q.8 Draw and explain the working of 3-bit synchronous counter.

Ans. : • Fig. Q.8.1 (a) shows 3-bit synchronous binary counter and its timing diagram.

• The state sequence for this counter is shown in Table Q.8.1. (Refer Table Q.8.1 on next page)

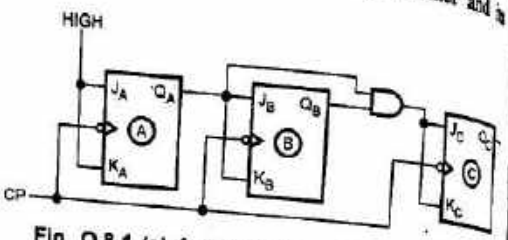


Fig. Q.8.1 (a) A three-bit synchronous binary counter

• Looking at Fig. Q.8.1(b), we can see that Q_A changes on each clock pulse as we progress from its original state to its final state and then back to its original state.

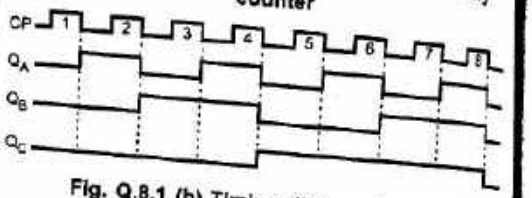


Fig. Q.8.1 (b) Timing diagram for 3-bit synchronous binary counter

• Flip-flop A is held in the toggle mode by connecting J and K inputs to HIGH.

• Flip-flop B toggles, when Q_A is 1.

• When Q_A is 0, flip-flop B is in the no-change mode and remains in its present state.

• Looking at the Table Q.8.1 we can notice that flip-flop C has to change its state only when Q_B and Q_A both are at logic 1. This condition is detected by AND gate and applied to the J and K inputs of flip-flop C. Whenever both Q_A and Q_B are HIGH, the output of the AND gate makes the J and K inputs of flip-flop C HIGH and flip-flop C toggles on the following clock pulse. At all other times, the J and K inputs of flip-flop C are held LOW by the AND gate output and flip-flop does not change state.

CP	Q_C	Q_B	Q_A
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Table Q.8.1 State sequence for 3-bit binary counter

Q.9 Design and implement 3 bit synchronous counter using JK FF.

[SPPU : May-12, Marks 8]

Ans. : Design of 3-bit synchronous binary up-counter :

Step 1 : Number of flip-flops : 3-bit counter so we require 3 flip-flops.

Step 2 : Types of flip-flops to be used JK

	Present state			Next state			Flip-flops inputs				
Decoder	C	B	A	C_{+1}	B_{+1}	A_{+1}	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	0	0	1	0	×	0	×	1	×
1	0	0	1	0	1	0	0	×	1	×	×	1
2	0	1	0	0	1	1	0	×	×	0	1	×
3	0	1	1	1	0	0	1	×	×	1	×	1
4	1	0	0	1	0	1	×	0	0	×	1	×
5	1	0	1	1	1	0	×	0	1	×	×	1
6	1	1	0	1	1	1	×	0	×	0	1	×
7	1	1	1	0	0	0	×	1	×	1	×	1

Table Q.9.1 Excitation table

Step 3 : Determine the excitation table for the counter.

Q_n	Q_{n+1}	J	K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

Table Q.9.2 Jk flip-flop excitation table

K-map simplification :

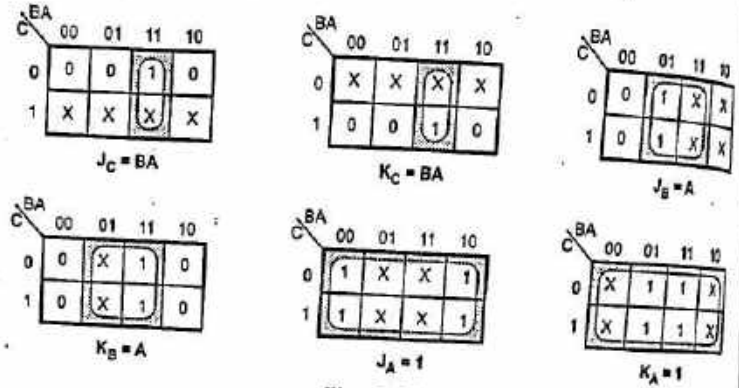


Fig. Q.9.1

Step 5 : Draw logic diagram.

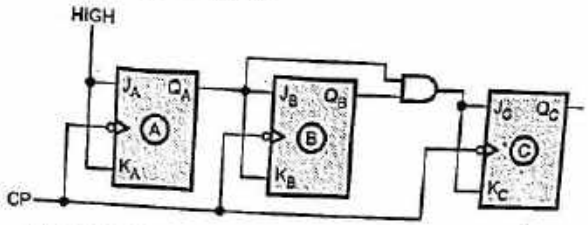


Fig. Q.9.2 A three-bit synchronous binary counter

Q.10 Design divide by 6 counter using T-flip-flops. Write state table and reduce the expression using K-map.

[SPPU : Dec.-09, May-14, Marks 8]

Ans. :

Step 1 : Determine the number of flip-flops needed.

For designing mod 6 counter using the formula

$$2^n \geq N$$

$N = 6 \therefore n = 3$ i.e. 3 flip-flops are required.

Here Type of flip-flops to be used : T

Step 2 : Determine the excitation table for counter.

Step 3 :

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Table Q.10.1 Excitation table for T-flip-flop

CP	Q_C	Q_B	Q_A	Q_{C+1}	Q_{B+1}	Q_{A+1}	T_C	T_B	T_A
0	0	0	0	0	0	1	0	0	1
1	0	0	1	0	1	0	0	1	1
2	0	1	0	0	1	1	0	0	1
3	0	1	1	1	0	0	1	1	1
4	1	0	0	1	0	1	0	0	1
5	1	0	1	0	0	0	1	0	1

Table Q.10.2 Excitation table for counter

Step 4 : K-map simplification.

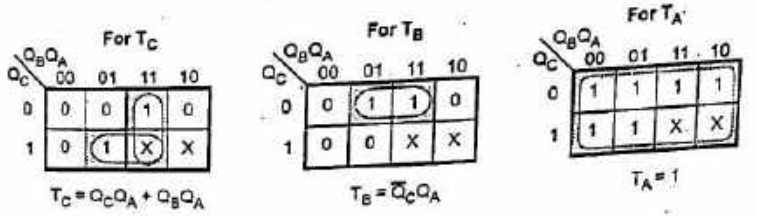


Fig. Q.10.1

Step 5 : Draw the logic diagram.

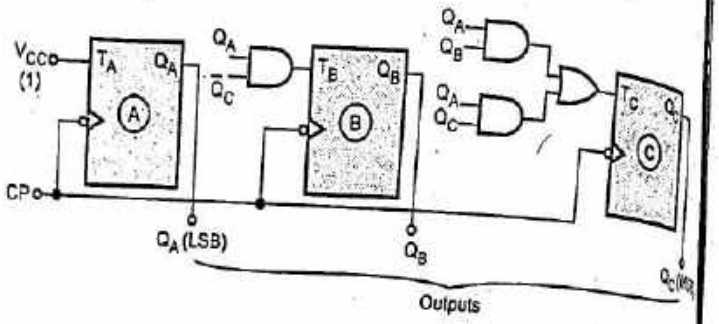
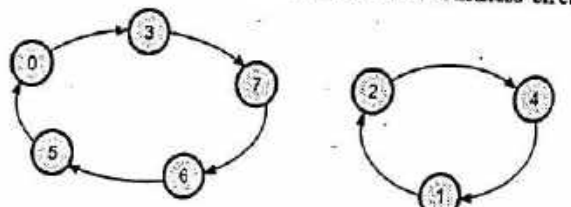


Fig. Q.10.2 Logic diagram

Q.11 What is lock-out condition and bushless circuit ?

Ans. : In a counter if the next state of some unused state is again a unused state and if by chance the counter happens to find itself in the unused states and never arrived at a used state then the counter is said to be in the lockout conditions. This is illustrated in the Fig. Q.11.1. The counter which never goes in lockout condition is called **self starting counter.**

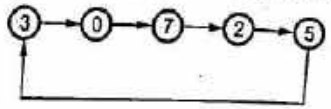
The circuit that goes in lockout condition is called **bushless circuit.**



(a) Desired sequence (b) Unused state forming lockout

Fig. Q.11.1

Q.12 Design synchronous counter which will go through the following step, using JK flip-flop. (*Avoid lock out condition.*).



☞ **[SPPU : May-11, Marks 8]**

Ans. : The Fig. Q.12.1 shows the state diagram for the given counter. To avoid lock-out condition states 1, 4 and 6 are force to enter into state 3.

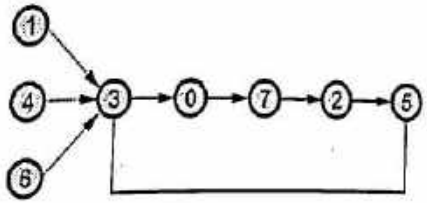


Fig. Q.12.1

Flip-flop excitation table is as shown below.

Present state			Next state		
A	B	C	A_{+1}	B_{+1}	C_{+1}
0	0	0	1	1	1
0	0	1	0	1	1
0	1	0	1	0	1
0	1	1	0	0	0
1	0	0	0	1	1
1	0	1	0	1	1
1	1	0	0	1	1
1	1	1	0	1	0

Table Q.12.1

K-map simplification

For D_A

A \ BC	00	01	11	10
0	1	0	0	1
1	0	0	0	0

$D_A = \overline{A}\,\overline{C}$

For D_B

A \ BC	00	01	11	10
0	1	1	0	0
1	1	1	1	1

$D_B = \overline{B} + A$

For D_C

A \ BC	00	01	11	10
0	1	1	0	1
1	1	1	0	1

$D_C = \overline{B} + \overline{C}$

Fig. Q.12.2

Logic diagram

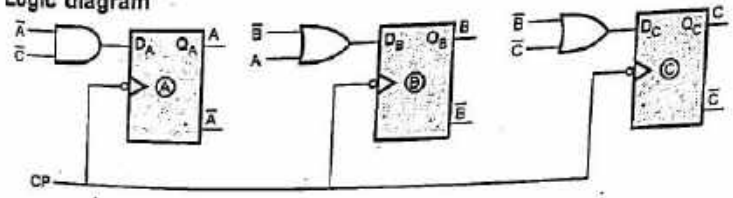


Fig. Q.12.3

Q.13 Design a synchronous counter for 4 → 6 → 7 → 3 → 1 → 4. Avoid lockout condition. Use JK type design.

☞ [SPPU : Dec.-15, Marks 6]

Ans. : Step 1 : State diagram

Here, states 5, 2 and 0 are forced to go into 6, 3 and 1 state, respectively to avoid lockout condition.

Step 2 : Excitation table

Present states			Next states			Flip-flop inputs					
A	B	C	A_{+1}	B_{+1}	C_{+1}	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	1	0	0	1	X	0	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	0	0	1	0	X	X	1	X	0
1	0	0	1	1	0	X	0	1	X	0	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	1	1	X	1	X	0	X	0

Table Q.13.1

Step 3 : K-map simplification

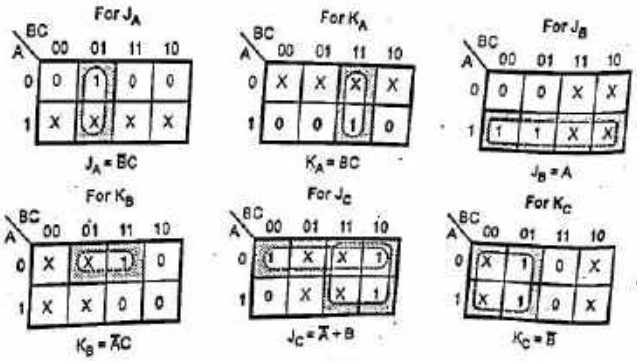


Fig. Q.13.1

Step 4 : Logic diagram

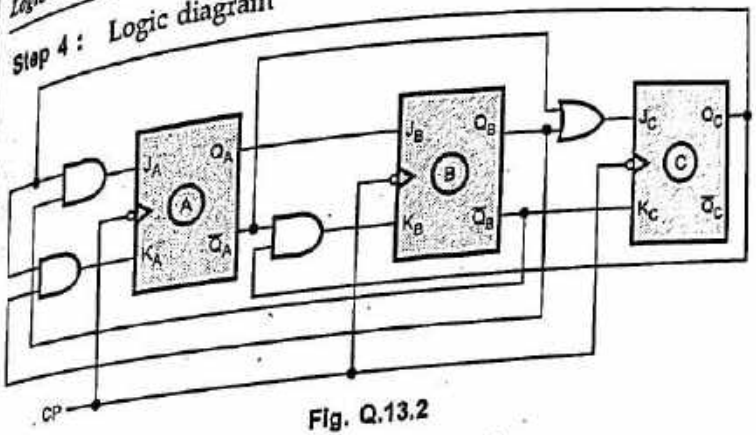


Fig. Q.13.2

Q.14 Design a synchronous decade counter using D flip-flop.

Ans. : The decade counter is a mod-10 counter. It has ten states : 0 - 9.

Step 1 : Determine the number of flip-flops needed.

We know that $2^n \geq N$. Here, $N = 10$ $\therefore$ $n = 4$ i.e. 4 flip-flops needed.

Step 2 : Types of flip-flops to be used : D

Step 3 : Determine the excitation table for counter.

Present state				Next state			
Q_D	Q_C	Q_B	Q_A	Q_{D+1}	Q_{C+1}	Q_{B+1}	Q_{A+1}
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0

Table Q.14.1 Excitation table for counter

Step 4 : K-map simplification

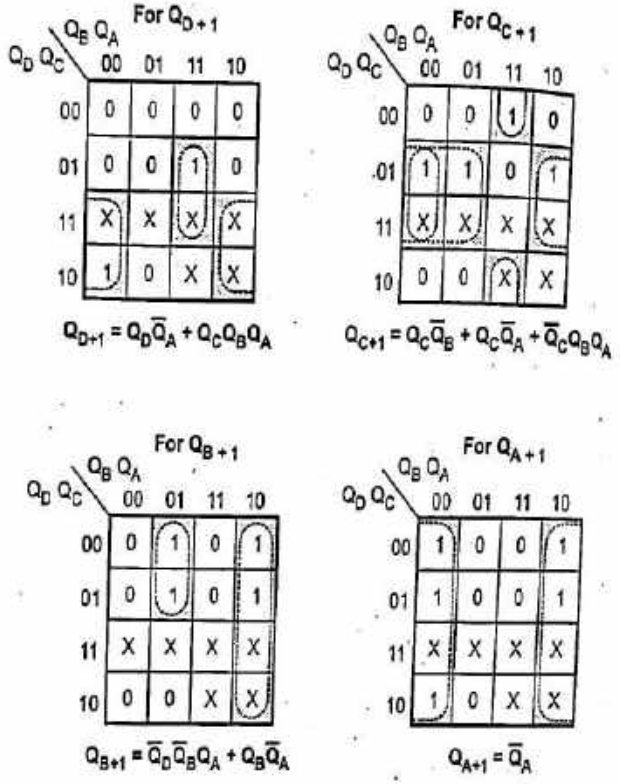


Fig. Q.14.1

$$Q_{D+1} = Q_D\overline{Q}_A + Q_CQ_BQ_A$$

$$Q_{D+1} = Q_D\overline{Q}_A + Q_CQ_BQ_A$$

$$Q_{C+1} = Q_C\overline{Q}_B + Q_C\overline{Q}_A + \overline{Q}_CQ_BQ_A$$

$$Q_{B+1} = \overline{Q}_D\overline{Q}_BQ_A + Q_B\overline{Q}_A$$

$$Q_{A+1} = \overline{Q}_A$$

Step 5 : Draw the logic diagram.

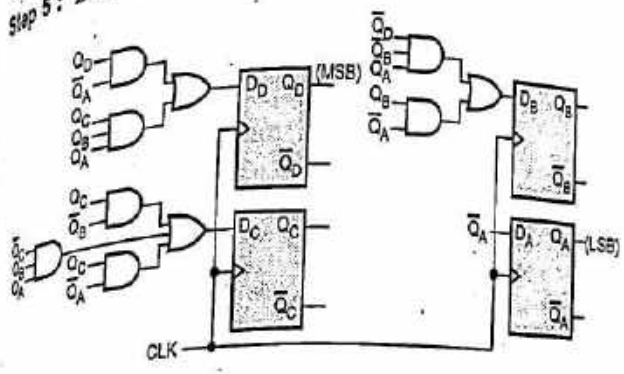


Fig. Q.14.2 Logic diagram

8.4 : Study of 7490 Modulus - n Counter IC

Q.15 Design a MOD-11 counter using IC 74191. Use RC signal.

[SPPU : May-06, Dec.-15, Marks 6]

Ans. : IC 74191 is a 4-bit counter. Thus it is MOD-16 counter. However, we require MOD-11 counter. The difference between 16 and 11 is 5. Hence 5 steps must be skipped from the full modulus sequence. This can be achieved by presetting counter to value 5. Each time when counter recycles it starts counting from 5 upto 16 on each full cycle. Therefore, each full cycle of the counter consists of 11 states.

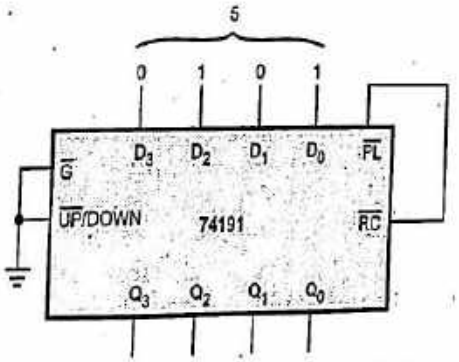


Fig. Q.15.1 MOD-11 counter using 74191

Q.16 How output frequency, f_{out} and clock frequency, f_{CLK} are related in case of binary counter, IC 74191 in up and down counting mode ? If f_{CLK} = 500 Hz and f_{out} = 50 Hz, design the programmable frequency divider using IC 74191 in up counting mode.

[SPPU : Dec.-05, Marks 6]

Ans. : The IC 74191 is a 4-bit binary counter, therefore $f_{out} = f_{CLK}/16$ in up and down counting mode. If f_{CLK} = 500 Hz and f_{out} = 50 Hz we need mod 10 (500/50) counter. The Fig. Q.16.1 shows the mod-10 counter using IC 74191.

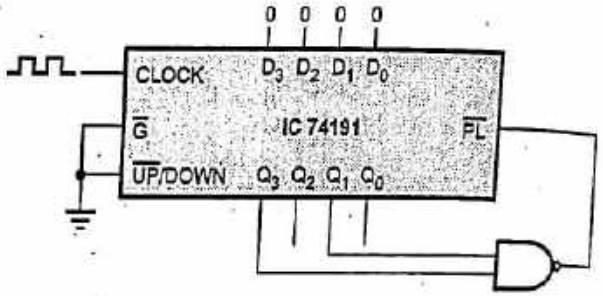


Fig. Q.16.1

Q.17 Explain how IC 74191 can also be used as programmable frequency divider. [SPPU : Dec.-07, Marks 2]

Ans. : IC 74191 is a 4-bit binary counter. Thus it divides the input frequency by 16. However, we can design MOD-N counter using IC 74191. For MOD-N counter the output frequency will be $f_{out} = \frac{f_{in}}{N}$. Thus by changing N we can change the output frequency. The Fig. Q.17.1 shows the programmable frequency divider using IC 74191.

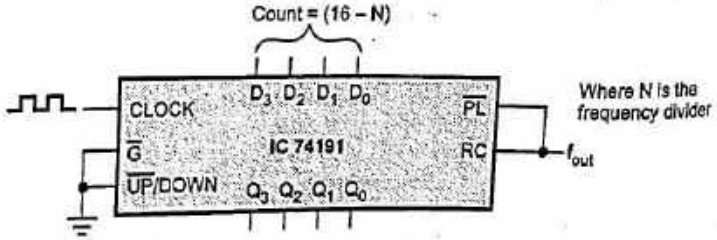


Fig. Q.17.1 Programmable frequency divider

Q.18 Design divide-by-2 for upcounting and divide by 5 for down-counting using frequency divider IC 74191. [SPPU : May-08, Marks 6]

Ans. : Divide-by-2 for up counting :

Divide-by-2 is a mod-2 counter. Since, after preset above counter goes through 2 states 1110 and 1111, it is a mod-2 counter. Thus, above circuit is a divide by 2 counter for up counting mode.

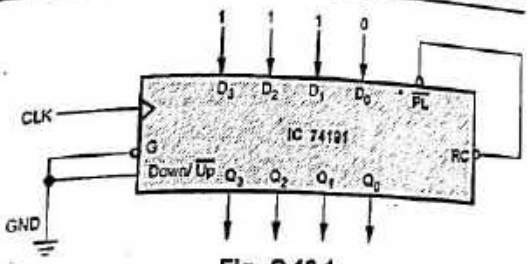


Fig. Q.18.1

Divide-by-5 for down counting mode :

Q_3	Q_2	Q_1	Q_0	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Truth table

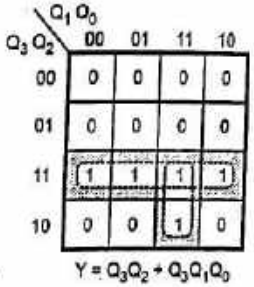


Fig. Q.18.2 K-map simplification

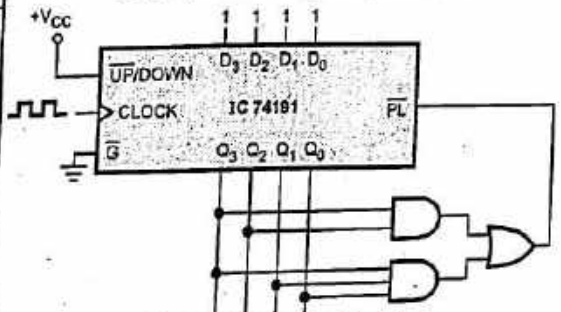


Fig. Q.18.3 Logic diagram

8.5 : Ring and Johnson Counters

Q.19 Draw a six stage ring counter and explain its operation. Mention about the use of presetting the counter. [SPPU : Dec.-08, Dec.-15, May-17, Marks 6]

Ans. : The Fig. Q.19.1 shows the six stage ring counter. The counter is present to value $(000001)_2$ by setting bit 0 = 1 and remaining bits = 0.

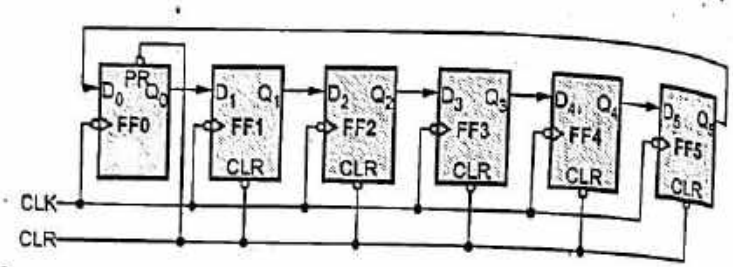


Fig. Q.19.1 Six stage ring counter

Operation : The Fig. Q.19.2 shows the operation of six-stage ring counter. On preset, FF0 (flip-flop 0) is set and FF1 to FF5 are reset. After each falling edge of the clock contents of ring counter are shifted 1 bit from LSB to MSB.

CLR	CLK	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Operation
⊓	X	1	0	0	0	0	0	Preset
1	↓	0	1	0	0	0	0	'1' Bit follows a circular path to form ring counter
1	↓	0	0	1	0	0	0	
1	↓	0	0	0	1	0	0	
1	↓	0	0	0	0	1	0	
1	↓	0	0	0	0	0	1	
1	↓	1	0	0	0	0	0	

Fig. Q.19.2 Illustrating operation of six-stage ring counter

Q.20 Draw and explain the operation of 4-bit Johnson counter.

☞ [SPPU : Dec.-08, Dec.-16, Marks 3]

OR Draw 4-bit twisted ring counter using D flip-flop. Consider initially all flip-flop. Consider initially all flip-flops are cleared. What

will be the output after 5th clock pulse and prove that the modulus of this twisted ring counter is 8.

☞ [SPPU : May-16, Marks 6]

Ans. : • In a Johnson counter, the Q output of each stage of flip-flop is connected to the D input of the next stage.

• The single exception is that the complement output of the last flip-flop is connected back to the D-input of the first flip-flop as shown in Fig. Q.20.1.

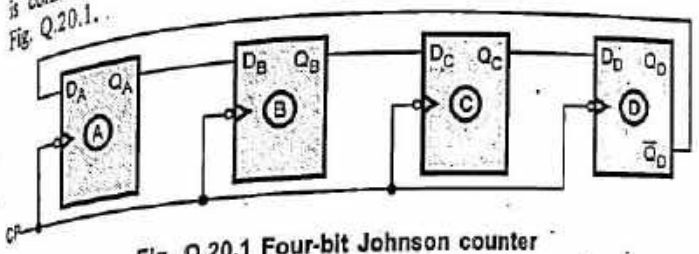


Fig. Q.20.1 Four-bit Johnson counter

Note : Johnson counter can be implemented with SR or JK flip-flops as well.

• As shown in Fig. Q.20.1 there is a feedback from the rightmost flip-flop complement output to the leftmost flip-flop input. This arrangement produces a unique sequence of states.

• Initially, the register (all flip-flops) is cleared. So all the outputs, Q_A, Q_B, Q_C, Q_D are zero.

Clock pulse	Q_A	Q_B	Q_C	Q_D
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1

Table Q.20.1 Four-bit Johnson sequence

• The output of last stage, Q_D is zero. Therefore complement output of last stage, $\overline{Q}_D$ is one. This is connected back to the D input of first stage. So D_A is one.

• The first falling clock edge produces $Q_A = 1$ and $Q_B = 0$, $Q_C = 0$, $Q_D = 0$ since D_B, D_C, D_D are zero.

- The next clock pulse produces $Q_A = 1$, $Q_B = 1$, $Q_C = 0$, $Q_D = 0$.
- The sequence of states is summarized in Table Q.20.1.
- After 8 states the same sequence is repeated.
- In this case, four-bit register is used. So the four-bit sequence has a total of eight states.
- Fig. Q.20.2 gives the timing sequence for a four-bit Johnson counter.

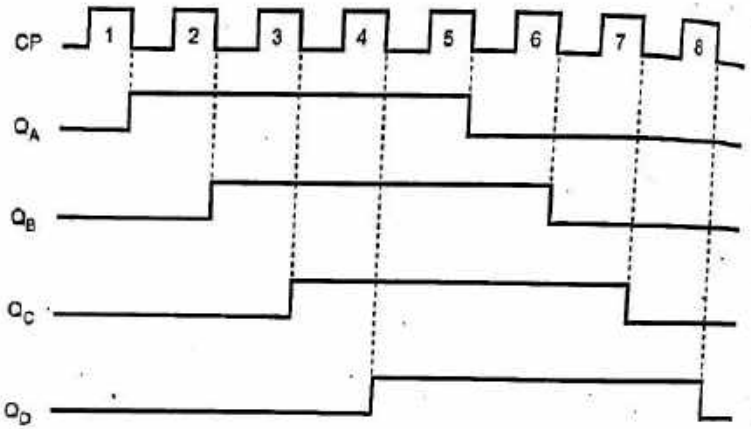


Fig. Q.20.2 Timing sequence for a four-bit Johnson counter

- If we design a counter of five-bit sequence, it has a total of ten states.
- An n-stage Johnson counter will produce a modulus of $2 \times n$, where n is the number of stages (i.e. flip-flops) in the counter.
- Johnson counter requires only half the number of flip-flops compared to the standard ring counter. However, it requires more flip-flop than binary counter.

Q.21 State all possible states if initial state of Johnson counter is 1110.

Ans. :

CP	Q_A	Q_B	Q_C	Q_D
0	1	1	1	0
1	1	1	1	1
2	0	1	1	1
3	0	0	1	1
4	0	0	0	1
5	0	0	0	0
6	1	0	0	0
7	1	1	0	0

Fig. Q.21.1

END...

Unit 4

9 Computer Organization and Computer Architecture

9.1 : Introduction

Q.1 Distinguish between computer architecture and computer organization with the help of appropriate examples.

Ans. : • **Computer architecture** refers to those attributes of a system visible to a programmer. In other words, we can also say that the computer architecture refers to the attributes that have a direct impact on the logical execution of the program.

- **Computer organisation** refers to the operational units and their interconnections that realize the architectural specifications.
- The **architectural attributes** include the instruction set, data types, number of bits used to represent data types, I/O mechanism, and techniques for addressing memory.
- **The organisational attributes** include those hardware details transparent to programmer, such as control signals, interfaces between the computer, memory and I/O peripherals.
- **For example,** it is an **architectural issue** whether a computer will have a multiply and division instructions. It is an **organisational issue** whether to implement multiplication and division by special unit or by a mechanism that makes repeated use of the add and subtract unit to perform multiplication and division, respectively.

9.2 : Functions and Types of Computer Units

Q.2 Explain functional units of computer.

[SPPU : May-12, Dec.-12, Marks 6]

Ans. : • The computer consists of five functionally independent units :

- Input
- Memory
- Arithmetic and logic
- Output and
- Control units.

• Fig Q.2.1 shows the block diagram of a digital computer.

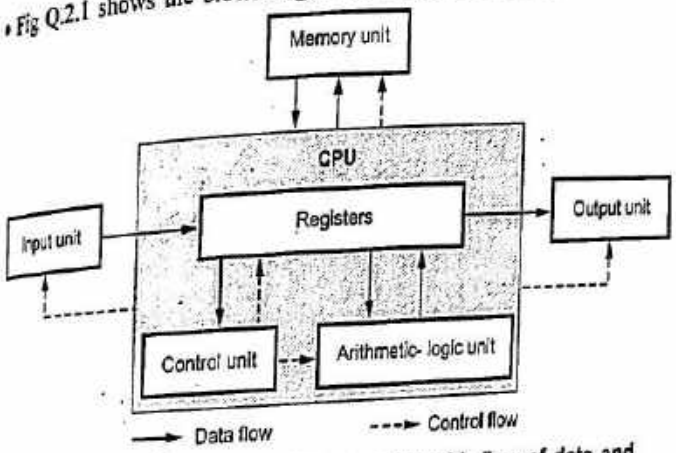


Fig. Q.2.1 Block diagram of a computer with flow of data and instructions

• It consists of input unit, control unit, Arithmetic-Logic Unit (ALU), registers, memory unit and output unit.

1. **Input unit :** It accepts instructions and data from the user and applies them as input to the computer.
2. **Central Processing Unit (CPU) :** As shown in Fig. Q.2.1, it consists of arithmetic logic unit. It processes and controls instructions and data inside the computer.

3. **Output unit :** After completing processing, this unit communicates results to the user.
4. **Memory unit :** The memory unit stores the data read from input device and provides it for processing when required. It also stores the intermediate and final results of processing.

9.3 : Central Processing Unit (CPU)

Q.3 What is CPU ?

Ans. : • The CPU is the brain of the computer system. It works as an administrator of a system.

• All the operations within the system are supervised and controlled by CPU. It interprets and co-ordinates the instructions.

• CPU controls all internal and external devices, performs arithmetic and logical operations, controls the memory usage and control the sequence of operations.

Q.4 Explain the components of the CPU with block diagram.

Ans. : • For performing all these operations, the CPU has three subunits :

- Arithmetic and Logic Unit (ALU)
- Control Unit
- Memory (CPU registers) Unit

• Fig. Q.4.1 shows the subsystem in the CPU and CPU interaction with other units.

ALU :

• It performs arithmetic operations like addition, subtraction and logic operations like OR, AND, invert, exclusive-OR on binary words. The data stored in memory unit is transferred to ALU. The ALU performs the operation, that is, the data is processed and the result is stored in internal memory unit of CPU.

• Arithmetic and logic operations performed by ALU sets flags to represent certain conditions such as equal to condition, zero condition,

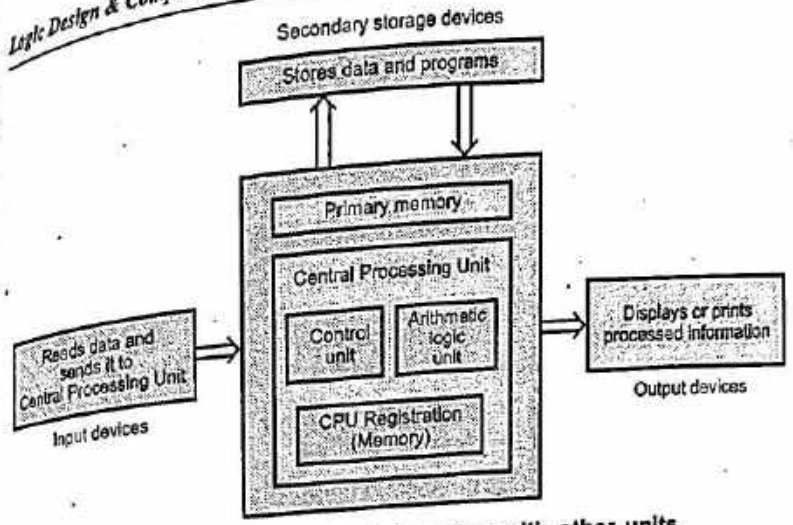


Fig. Q.4.1 CPU and its interaction with other units

greater than condition and so on. These conditions are checked by program instructions to change the sequence of program execution.

Control Unit

• It controls all the operations which internally take place within the CPU and also the operations of CPU related to input/output devices. The control unit directs the overall functioning of a computer system.

• It interprets program instructions and generates control signals to ensure correct execution of the program. The control signals generated by the control unit direct the overall functioning of the other units of the computer.

Memory unit

• It consists of two sections :

- internal memory section
- external memory section

• An internal memory section for storage of active data and instructions and an external memory section for long term storage.

9.4 : Memory

Q.5 Give the classification of computer memory.

Ans. : Fig. Q.5.1 gives the classification of computer memory.

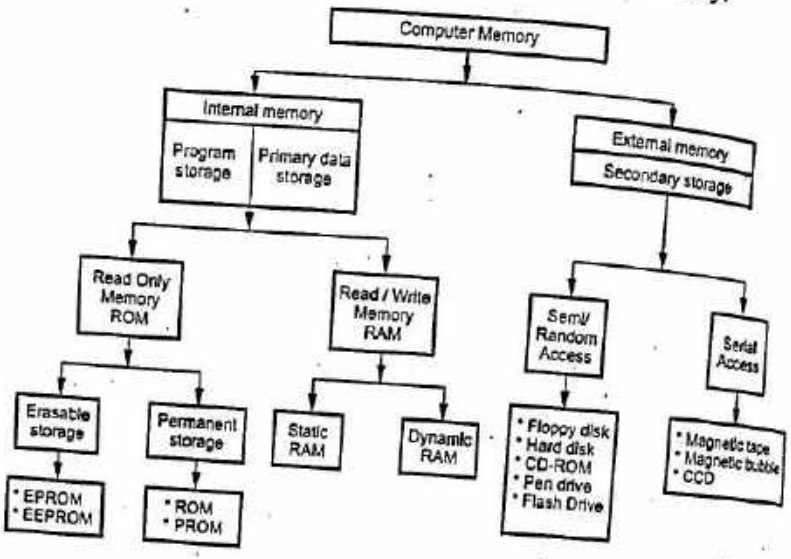


Fig. Q.5.1 Types of memory

- The internal memory is also called **primary memory** as it can be directly accessed by processor with a random access. Due to this such memories are able to respond fast enough to synchronize with the execution speed of the processor.
- On the other hand external memory devices cannot be directly accessed by the processor. These devices are also called **secondary devices.**

External Memory

- External memory consists of floppy drive, hard disk, compact disks, DVDs, magnetic tapes, flash drives, memory cards, charged coupled devices (CCD) etc.
- The primary feature of these devices are :
  - High capacity

  - Slow access
  - Low cost.

- Flash drive is a small device that can store and transfer data to and from various other devices. It has plenty of other names such as USB drive, pen drive, thumb drive, and more.
- Memory cards are smaller and thinner external memory that users usually store in other devices. The most common types of memory cards are CompactFlash, a Memory Stick, and SD card.
- Nowadays, you don't have to have physical devices to have external memory storage. Online, or "cloud storage" is another option that utilizes the Internet.

Internal Memory

- Internal memory is further subdivided into program storage memory and data storage memory. Typically, internal memory is implemented with both, ROM and RAM ICs.
- Data, whether they are to be interpreted as numbers, characters, or instructions, can be stored in either ROM or RAM. RAM and ROM memories consist of an array of registers, in which each register has unique address.

Q.6 Give the difference between primary and secondary memory.

Ans. :

Parameter	Primary Storage Memory	Secondary Storage Memory
Types	It includes static RAM and dynamic RAM.	It includes Floppy disk, Hard disk, CDROM, Magnetic tape, Magnetic bubble memory etc.
Storage capacity	It has less storage capacity ranging from 1 kbytes to 512 kbytes.	It has a high storage capacity and it is practically unlimited because when one disk or tape is full, the next one can be used.

Access	Microprocessor can access and process data/program directly from these memories.	Microprocessor cannot directly access or process data/program stored in these memories. Data/program need to be copied into primary storage memory for microprocessor access.
Speed	It can be accessed with a greater speed.	Its access speed is slow.
Cost	Its cost in high for per unit storage capacity.	Its cost is low for per unit storage capacity.
Physical size	Physical size is large for per unit storage capacity.	Physical size is small for per unit storage capacity.

9.5 : Input / Output

Q.7 What are the functions of input/output devices ?

Ans. : We can interact with the personal computer using input-output (I/O) devices. Using input devices, computer can accept data and instructions from the user or another computer system (For example, computer on the Internet). Using output devices, computer can send the processed data to the user or to another computer system.

Q.8 List and explain the functions of various input devices.

Ans. : The function of an input device is to apply data to the computer for processing.

- **Keyboard :** It is the most commonly used input device which accepts text and numbers.
- **Mouse :** It is again the commonly used input device to position the screen cursor.
- **Trackball :** It allows to produce screen cursor movement. It is two-dimensional positioning device.

- **Spaceball :** It is usually used in three-dimensional positioning and selecting operations in virtual-reality systems.
- **Joystick :** It has a small, vertical lever (called the stick) mounted on the base and used to steer the screen cursor around. Both x and y co-ordinate positions can be simultaneously altered by the motion of a single lever in a joystick.
- **Scanner :** The scanner is a device, which is used to store drawings, graphs, photos or text available in printed form for computer processing.
- **Digital Camera :** The still images can be recorded with it. These images can be viewed and edited any time on the computer.
- **Light Pen :** It is a pencil shaped device used to select positions by detecting the light coming from points on the CRT screen. It consists of photoelectric cell housed in a pencil like case.
- **Microphone :** It enables to input data which is in the form of voice or music.
- **Digitizers :** It is used for applications such as tracing. It consists of flat surface which can detect the position of a movable stylus.

Q.9 List and explain the functions of various output devices.

Ans. : The function of an output device is to present processed data to the user.

- **Monitor :** The computer sends processed data (i.e. output) to the monitor when the user needs to observe the output.
- **Printer/Plotter :** When the user needs hard-copy (i.e. paper-copy) of an output, the computer sends output to the printer/Plotter.
- **Head-phones or Speakers :** When the user needs a sound-output, the computer sends output to the head-phones or speakers.

Some devices act as both, input and output devices.

- **Touch Screen :** This is a monitor having touch sensing mechanism. It displays text or icons you can touch. When you touch the screen, special sensors detect the touch and the computer calculates the screen

co-ordinates of point of contact. According to the location of the touch, the computer displays the information or determines the next action which to be performed.

- **Communication Devices :** These devices are used to connect two or more computers to each other, that is, for networking. For example, modems. They enable the computers to communicate through telephone lines.
- Another example is network interface cards (NICs) which allow to connect a group of computers to share data and devices.

9.6 : System Bus

Q.10 What is system bus ?

Ans. : The central processing unit, memory unit and I/O unit are the hardware components/modules of the computer. They work together with communicating each other and have paths for connecting the modules together.

- A group of wires, called **bus** is used to provide necessary signals for communication between modules.
- A bus that connects major computer components/modules (CPU, memory, I/O) is called a **system bus.** The system bus is a set of conductors that connects the CPU, memory and I/O modules.
- Usually, the system bus is separated into three functional groups :
  - Data Bus
  - Address Bus
  - Control Bus

Data Bus : It consists of 8, 16, 32 or more parallel signal lines. These lines are used to send data to memory and output ports, and to receive data from memory and input port. Therefore, data bus lines are **bi-directional.**

Address Bus : It is an **unidirectional** bus.

- The address bus consists of 16, 20, 24 or more parallel signal lines.

- On these lines the CPU sends out the address of the memory location or I/O port that is to be written to or read from.
- Here, the communication is one way, the address is send from CPU to memory and I/O port and hence these lines are **unidirectional.**

Control Bus : • These lines regulate the activity on the bus.

- The CPU sends signals on the control bus to enable the outputs of addressed memory devices or port devices.

Q.11 Explain the single bus structure.

Ans. :

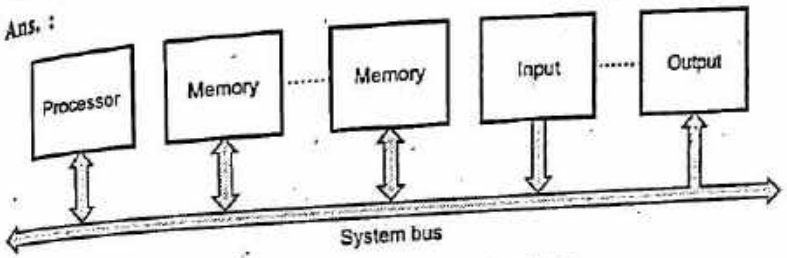
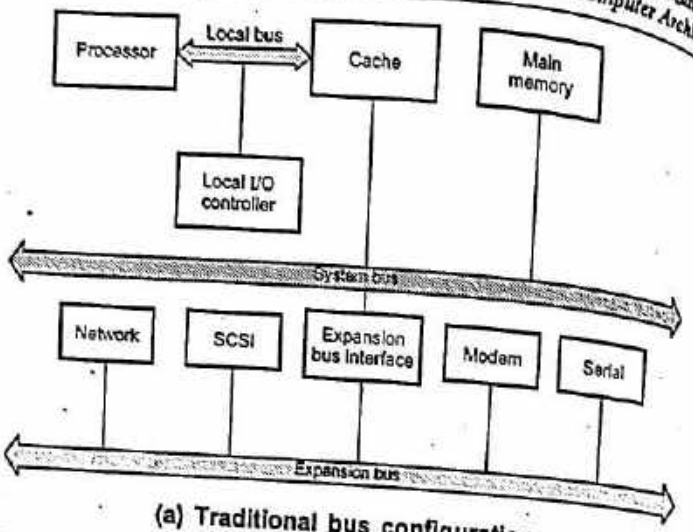


Fig. Q.11.1 Single bus structure

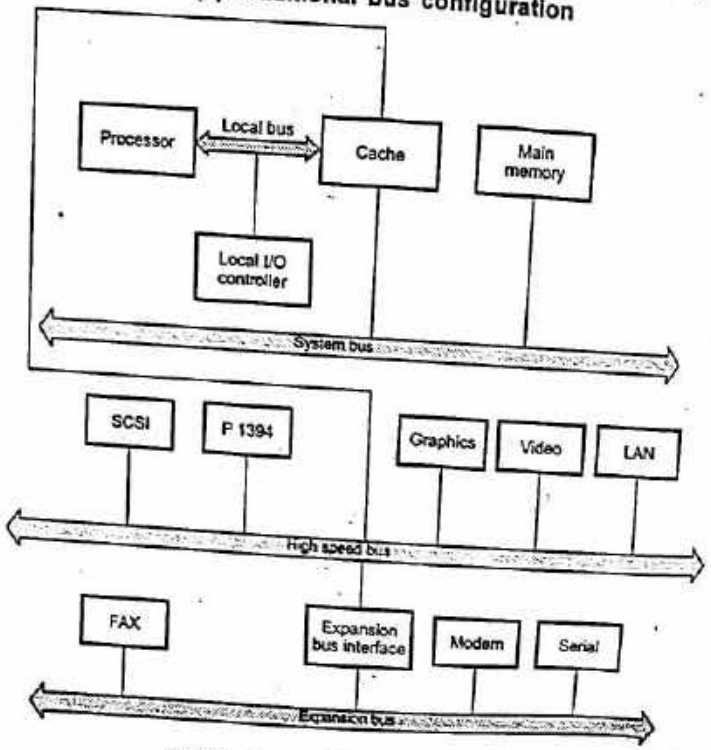
- Here, address bus, data bus and control bus are shown by single bus called **system bus.** Hence such interconnection bus structure is called **single bus structure.** Refer Fig. Q.11.1.
- In a single bus structure all units are connected to common bus called **system bus.**
- However, with single bus only two units can communicate with each other at a time.
- The bus control lines are used to arbitrate multiple requests for use of the bus.
- The main advantage of single bus structure is its low cost and its flexibility for attaching peripheral devices.

Q.12 Explain the multibus structure.

Ans. : • The need of high speed shared bus is impractical to satisfy with a single bus. Thus, most computer systems use the multiple buses.



(a) Traditional bus configuration



(b) High-speed bus configuration

Fig. Q.12.1

- These buses have the hierarchical structure.
- Fig. Q.12.1 shows two bus configurations. The traditional bus connection uses three buses : local bus, system bus and expanded bus.
- The high speed bus configuration uses high-speed bus along with the three buses used in the traditional bus connection.
- Here, cache controller is connected to high-speed bus.
- This bus supports connection to high-speed LANs, such as Fiber Distributed Data Interface (FDDI), video and graphics workstation controllers, as well as interface controllers to local peripheral buses including **Small Computer System Interface SCSI** and P1394.

9.7 : Von Neumann Architecture

Q.13 Draw and explain the Von Neumann architecture.

☞ **[SPPU : Dec.-05,08, May-07, Marks 6, June-22, Marks 8]**

Ans. : • Fig. Q.13.1 shows the general structure of a Von Neumann machine (IAS - Institute for Advanced Study, computer).

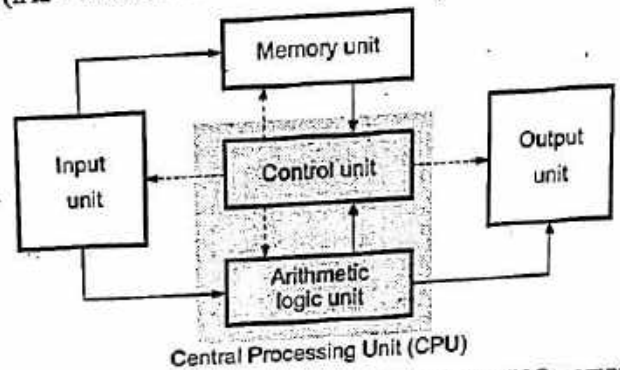


Fig. Q.13.1 Structure of Von Neumann machine (IAS computer)

- It consists of five basic units whose functions can be summarized as follows :
  - **The input unit** transmits data and instructions from the outside world to machine. It is operated by control unit.

- The memory unit stores both, data and instructions.
- The Arithmetic-Logic Unit (ALU) performs arithmetic and logical operations.
- The control unit fetches and interprets the instructions in memory and causes them to be executed.
- The output unit transmits final results and messages to the outside world.

Q.14 List the features of Von Neumann architecture.

Ans. : Features of Von Neumann architecture are :

- It uses stored program concept. The program (instructions) and data are stored in a single read-write memory.
- The contents of read-write memory are addressable by location without regard to the type of data contained there.
- Execution of instructions occurs in a sequential manner (unless explicitly modified) from one instruction to the next.

Q.15 What is Von Neumann bottleneck ?

Ans. : Because of the stored program architecture of Von-Neumann machine, the processor performance is tightly bound to the memory performance. That is, since we need to access memory at least once per cycle to read an instruction, the processor can only operate as fast as the memory. This is sometimes known as the **Von Neumann bottleneck** or **memory wall.**

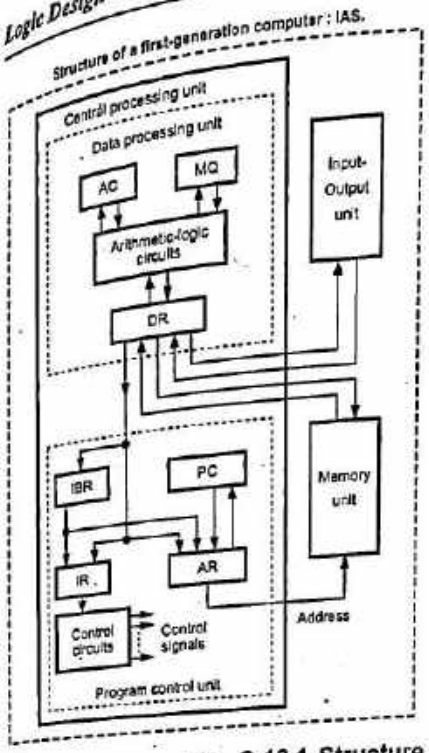
Q.16 Draw and explain detail structure of IAS computer.

Or Draw IAS (Von Neumann) architecture and explain function of registers in it.

☞ **[SPPU : May-13 Marks 8]**

Ans. : • Fig. Q.16.1 shows detail structure of IAS computer.

- It consists of various processing and control units, along with a set of high speed registers (AC, MQ, DR, IBR, PC, IR and AR). These registers are used to store instructions, memory addresses and data.
- The complete instruction cycle involves three operations : Instruction fetching, opcode decoding and instruction execution.



Register	Abbreviation
Accumulator	AC
Multiplier Quotient	MQ
Data Register	DR
Program Counter	PC
Instruction Buffer Register	IBR
Address Register	AR
Instruction Register	IR

Fig. Q.16.1 Structure of IAS computer

- The control circuits in the program control unit are responsible for fetching instructions, decoding opcodes, routing information correctly through the system and providing proper control signals for Central Processing Unit (CPU) actions.
- After decoding, the arithmetic logic circuits of the data processing unit perform actions specified by the instruction.
- An electronic clock circuit (not shown in the Fig. Q.16.1) is used to generate the basic timing signals to synchronize the operation of the different parts of the system.
- The functioning of different registers is as given below :
  - **PC (Program Counter)** : It is an address register. It is used to store the address of the next instruction to be executed and hence also referred to as instruction address register.

- **AR (Address Register)** : It is a 12-bit address register. It is used to specify the address in memory of the word to be written into or read from the DR.
- **DR (Data Register)** : It is a 40-bit register. It is used to store any 40-bit word. A word transfer can take place between the 40-bit data register DR of the CPU and any memory location. The DR may be used to store an operand during the execution of an instruction.
- **AC (Accumulator) and MQ (Multiplier-Quotient)** : These are two 40-bit registers used for the temporary storage of operands and results.
- **IR (Instruction Register) and IBR (Instruction Buffer Register)** : Program control unit fetches two instructions simultaneously from memory. The opcode of the first instruction is placed in the Instruction Register (IR) and the instruction that is not to be executed immediately (second instruction) is placed in the Instruction Buffer Register (IBR).

Q.17 Explain the instructions supported by IAS computer.

Ans. : The instructions of IAS computer are divided in five groups :

- Data transfer
- Unconditional branch
- Conditional branch
- Arithmetic
- Address modify

Table Q.17.1 shows the instruction set of IAS computer.

Instruction type	Shorthand notation	Description
Data transfer	AC ← MQ	Transfer contents of register MQ to the accumulator AC
	MQ ← M(X)	Transfer contents of memory location X to MQ

	M(X) ← AC	Transfer contents of accumulator to memory location X
	AC ← M(X)	Transfer M(X) to the accumulator
	AC ← – M(X)	Transfer – M(X) to the accumulator
	AC ← | M(X)|	Transfer absolute value of M(X) to the accumulator
	AC ← – |M(X)|	Transfer – |M(X)| to the accumulator
Unconditional branch	go to M(X, 0:19)	Take next instruction from left half of M(X)
	go to M(X, 20:39)	Take next instruction from right half of M(X)
Conditional branch	if AC ≥ 0 then go to M(X, 0:19)	If number in the accumulator is non-negative, take next instruction from left half of M(X)
	if AC ≥ 0 then go to M(X, 20:39)	If number in the accumulator is non-negative, take next instruction from right half of M(X)
Arithmetic	AC ← AC + M(X)	Add M(X) to AC; put the result in AC
	AC ← AC + |M(X)|	Add |M(X)| to AC; put the result in AC
	AC ← AC – M(X)	Subtract M(X) from AC; put the result in AC

	$AC \leftarrow AC - \lvert M(X) \rvert$	Subtract $\lvert M(X) \rvert$ from AC; put the result in AC
	$AC.MQ \leftarrow MQ \times M(X)$	Multiply M(X) by MQ; put most significant bits of result in AC, put least significant bits in MQ
	$MQ.AC \leftarrow AC \times M(X)$	Divide AC by M(X), put the quotient in MQ and the remainder AC
	$AC \leftarrow AC \times 2$	Multiply accumulator by 2, i.e., shift left one bit position
	$AC \leftarrow AC \div 2$	Divide accumulator by 2, i.e., shift right one bit position
Address modify	$M(X, 8{:}19) \leftarrow AC(28{:}39)$	Replace left address field at M(X) by 12 rightmost bits of AC
	$M(X, 28{:}39) \leftarrow AC(28{:}39)$	Replace right address field at M(X) by 12 rightmost bits of AC

Table Q.17.1 Instruction set of IAS computer

9.8 : Harvard Architecture

Q.18 Draw block diagram for Harvard architecture and explain each block. What are its advantages and disadvantages ?

☞ [SPPU : May-13,14, Marks 6]

Ans. : • Harvard architecture provides separate memory banks for program storage (code memory), the processor stack and variable RAM (data memory), as shown in the Fig. Q.18.1.

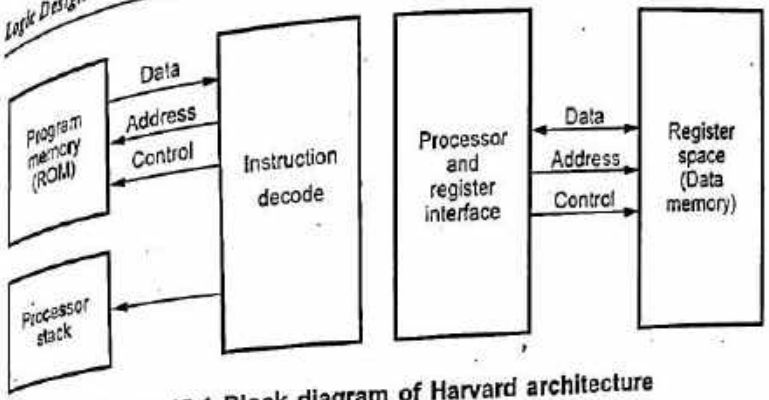


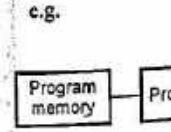
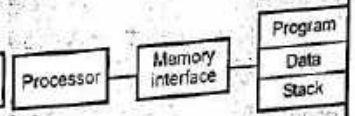
Fig. Q.18.1 Block diagram of Harvard architecture

- It has advantage of executing instructions in fewer instructions cycles than Princeton (Von Neumann) architecture.
- In Harvard architecture, greater amount of instruction parallelism can be achieved due to separate memory banks.

Q.19 Give the comparison between Harvard and Von-Neumann Architectures.

☞ [SPPU : June-22, Marks 8]

Ans. :

Sr. No.	Harvard	Von-Neumann
1.	It consist of separate memory bank, processor, instruction decode. e.g. 	It consist of processor, instruction decoder, memory interface unit. 
	This architecture consist of separate code (program) and data memory.	This architecture consist of memory i.e. program (code) and data used same memory area.

2.	Execution of instruction is faster because fetching of code and data separately or simultaneously.	Execution of instruction is relatively slower than Harvard architecture because not possible to fetch code and data simultaneously.
3.	RISC architecture used only.	RISC and CISC architecture used.
4.	More parallelism.	Less parallelism compared to Harvard architecture.
5.	Microcontroller is the example of Harvard architecture. e.g. Real-time system.	General purpose microprocessor is the example of Von-Neumann architecture.

9.9 : Instruction Cycle

Q.20 Explain the instruction cycle. ☞ [SPPU : May-05, 07, Marks 5]

Or Draw the instruction cycle state diagram.

Ans. :

- An instruction cycle involves three subcycles,
  - **Fetch** : The fetch phase reads the next instruction from memory into the CPU.
  - **Decode** : The decode phase interprets the opcode by decoding it.
  - **Execute** : The execute phase performs the indicated operation.
- Fig. Q.20.1 shows the basic instruction cycle. (See Fig. Q.20.1 on next page).
- Actually, processor checks for valid interrupt request after each instruction cycle. If any valid interrupt request is present, Processor saves the current process state and services the interrupt. Servicing the interrupt means executing interrupt service routine. After completing it, processor starts the new instruction cycle from where it has been interrupted. Fig. Q.20.2 shows this instruction cycle with interrupt cycle.

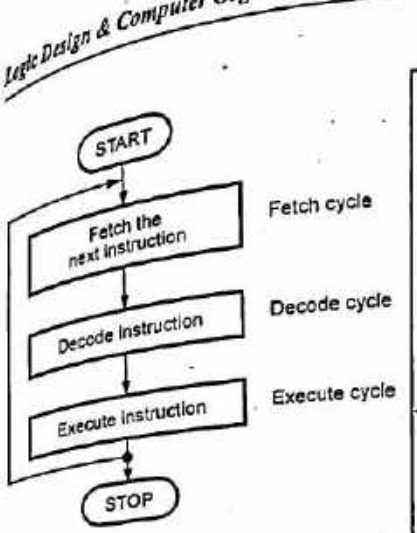


Fig. Q.20.1 Basic instruction cycle

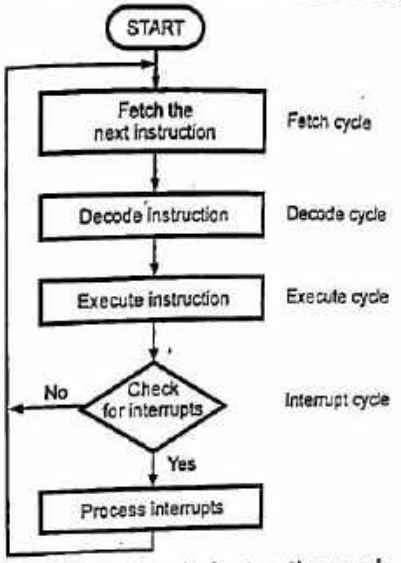


Fig. Q.20.2 Basic instruction cycle with interrupt cycle

The Indirect cycle

- If the operands on which the instruction works are present within the processor-registers, a memory access is not required. But if the execution of an instruction involves one or more operands in memory, each requires a memory access.
- For fetching the indirect addresses, one or more instruction sub cycles are required.
- After fetching the instruction, it is decoded and if any indirect addressing is involved, the required operands are fetched using indirect addressing.
- Also, after performing the operation on the operands according to the opcode, a similar process may be needed to store the result in memory.

Q.21 Draw the instruction cycle state diagram.

Ans. :

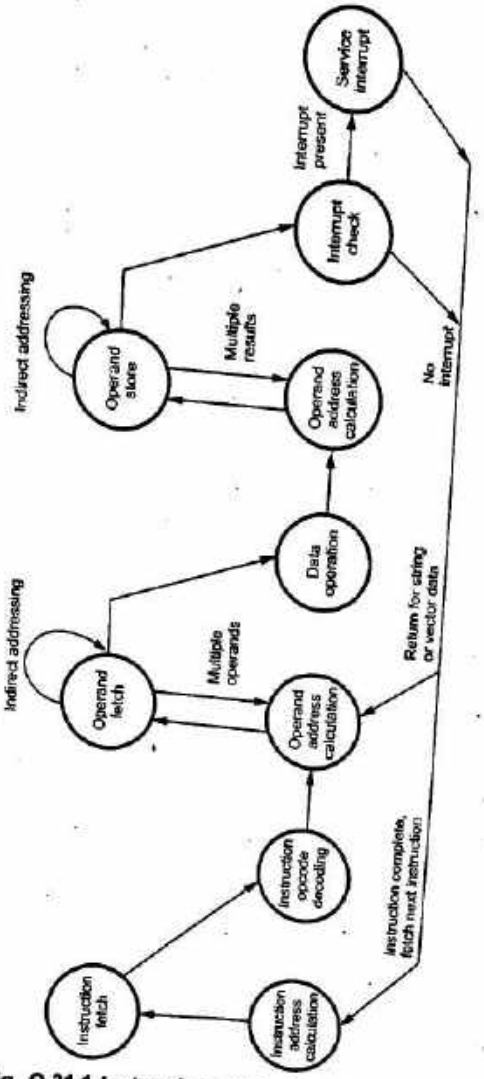


Fig. Q.21.1 Instruction cycle state diagram

END...

Unit 4

10 Processor

10.1 : Single Bus Organization of CPU

Q.1 Draw the neat diagram of single bus organization of the CPU showing ALU, all types of registers and the data paths among them.

☞ [SPPU : May-05,07,09,10,12,13,17,19, Dec.-06,08,09,17,19, Marks 8]

Ans. : • Fig. Q.1.1 shows the single bus organization of processor unit. It shows how the arithmetic and logic unit and all processor registers are organized and how they are interconnected.

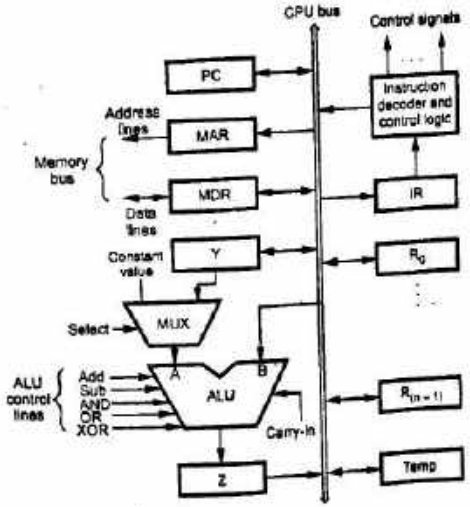


Fig. Q.1.1 Single bus organisation of processor

- The special function registers include Program Counter (PC), instruction register (IR), memory address register (MAR) and memory data register (MDR). It also shows the external memory bus connected to memory address (MAR) and data register (MDR).
- **Program Counter (PC) :** It keeps track of which instruction is being executed and what the next instruction will be.
- **Instruction Register (IR) :** It is used to hold the instruction that is currently being executed. The contents of IR are available to the control logic, which generate the timing signals that control the various processing elements involved in executing the instruction.
- **Memory Address Register (MAR) and Memory Data Register (MDR) :** These registers are used to handle the data transfer between the main memory and the processor. The MAR holds the address of the main memory to or from which data is to be transferred. The MDR sometimes also called MBR (Memory Buffer Register) contains the data to be written into or read from the addressed word of the main memory.
- The registers Y, Z and Temp in Fig. Q.1.1, are used only by the CPU unit for temporary storage during the execution of some instructions. These registers are never used for storing data generated by one instruction for later use by another instruction. The programmer can't access these registers.
- The IR and the instruction decoder are integral parts of the control circuitry in the CPU unit.
- All other registers and the ALU are used for storing and manipulating data. The data registers, ALU and the interconnecting bus are referred to as DATA PATH.
- Register R0 through R(n - 1) are the CPU registers. These registers include general purpose registers and special purpose registers such as stack pointer, index registers and pointers.
- There are two options provided for A input of the ALU. The multiplexer (MUX) is used to select one of the two inputs. It selects either output of Y register or a constant number as an A input for the ALU according to the status of the select input.

- It selects output of Y when select input is 1 (select Y) and it selects a constant number when select input is 0 (select C) as an input A for the multiplexer. The constant number is used to increment the contents of program counter.
- In a single bus organization, only one data word can be transferred over the bus in a clock cycle. This increases the steps required to complete the execution of the instruction. To reduce the number of steps needed to execute instructions, most commercial processors provide multiple bus organization, i.e., multiple internal paths that enable several transfers to take place in parallel.

Q.2 Compare single bus organization with multiple bus organization of CPU. [SPPU : Dec.-09, Marks 2]

Ans. : • Table Q.2.1 gives comparison between single bus organization and multiple bus organization.

Sr. No.	Single bus organization	Multiple bus organization
1.	Provides single internal path for data transfer.	Provides multiple paths of internal data transfer.
2.	Only one data word can be transferred over the bus in a clock cycle.	Multiple words can be transferred over the bus in a clock cycle.
3.	More number of steps required to complete the instruction execution as compared to multiple bus organization.	Less number of steps required to complete the instruction execution as compared to single bus organization.

Table Q.2.1

Q.3 What is flag register / status register ? Explain its use and its structure.

Ans. : The status register is used to store the results of certain condition when certain operations are performed during execution of the program. The **status register** is also referred to as **flag register.** ALU operations and certain register operations may set or reset one or more bits in the status register. Status bits lead to a new set of CPU instructions. These

instructions permit the execution of a program to change flow on the basis of the condition of bits in the status register. So the condition bits in the status register can be used to take logical decisions within the program. Some of the common status register bits are :

1) Carry/Borrow : The carry bit is set when the summation of two 8-bit numbers is greater than 1111 1111 (FFH). A borrow is generated when a large number is subtracted from a smaller number.

2) Zero : The zero bit is set when the contents of register are zero after any operation. This happens not only when you decrement the register, but also when any arithmetic or logical operation causes the contents of register to be zero.

3) Negative or sign : In 2's complement arithmetic, the most significant bit is a sign bit. If this bit is logic 1, the number is negative number, otherwise a positive number. The negative bit or sign bit is set when any arithmetic or logical operation gives a negative result.

4) Auxiliary Carry : The auxiliary carry bit of status register is set when an addition in the first 4 bits causes a carry into the fifth bit. This is often referred as half carry or intermediate carry. This is used in the BCD **arithmetic.**

5) Overflow Flag : In 2's complement arithmetic, most significant bit is used to represent sign and remaining bits are used to represent magnitude of a number (see Fig. Q.3.1). This flag is set if the result of a signed operation is too large to fit in the number of bits available (7-bits for 8-bit number) to represent it.

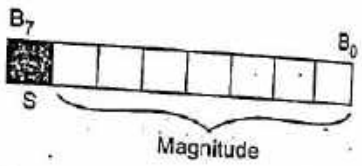


Fig. Q.3.1 2's Complement 8-bit number

For example, if you add the 8-bit signed number 01110110 (+118 decimal) and the 8-bit signed number 00110110 (+ 54 decimal). The result will be 10101100 (+ 172 decimal), which is the correct binary result, but in this case it is too large to fit in the 7-bits allowed for the magnitude in an 8-bit signed number. The overflow flag will be set after this operation to indicate that the result of the addition has overflowed into the sign bit.

6) Parity : When the result of an operation leave the indicated register with an even number of 1s, parity bit is set.

Q.4 What are general purpose registers ?

Ans. : In addition to the special purpose register, most CPUs have other registers called **general purpose registers.** The general purpose registers are used as simple storage area, mainly these are used to store intermediate results of the operation. Getting the operand from the general purpose registers is more faster than from memory so it is better to have sufficient number of general purpose registers.

10.2 : Operations of Control Unit

Q.5 What is instruction sequencing ?

Ans. : Every processor has some basic types of instructions such as data transfer instructions, arithmetic instructions, logical instructions, branch instructions and so on. To perform a particular task on the computer, it is programmer's job to select and write appropriate instructions one after the other, i.e. programmer has to write instructions in a proper sequence. This job of programmer is known as **instruction sequencing.**

Q.6 What is straight - line sequencing ?

Ans. : Processor executes a program with the help of Program counter (PC). PC holds the address of the instruction to be executed next. To begin execution of a program, the address of its first instruction is placed into the PC. Then, the processor control circuits use the information (address of memory) in the PC to fetch and execute instructions, one at a time, in the order of increasing addresses. This is called **straight-line sequencing.**

Q.7 Write a short note on register transfers.

Ans. : • In Fig. Q.7.1, the data transfer between registers and common bus is shown by a line with arrow heads. But in actual practice each register has input and output gating and these gates are controlled by corresponding control signals. This is illustrated in Fig. Q.7.1.

• As shown in Fig. Q.7.1, control signals. $R_{i\,in}$ and $R_{i\,out}$ controls the input and output gating of register R_i. When $R_{i\,in}$ is set to 1, the data

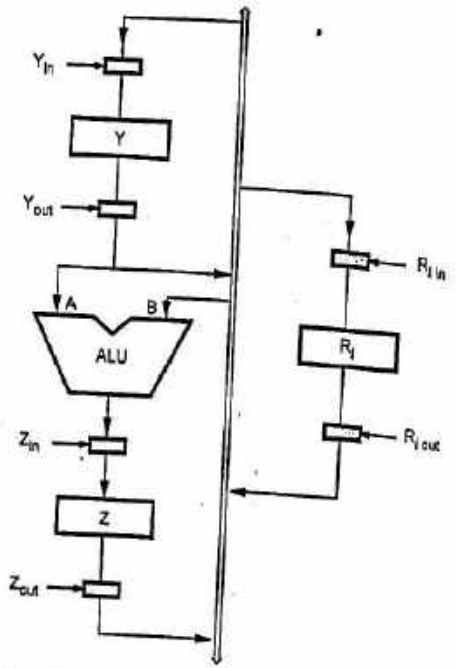


Fig. Q.7.1 Input and output gating for the register

available on the common data bus is loaded into register R_i. Similarly, when $R_{i\,out}$ is set to 1, the contents of register R_i are placed on the common data bus.

- The signals $R_{i\,in}$ and $R_{i\,out}$ are commonly known as input enable and output enable signals of registers, respectively.
- The transfer of data from register R_1 to R_2 can be accomplished as follows :
  - Activate the output enable signal of R_1, $R_{1out} = 1$. This places the contents of R_1 on the common bus.
  - Activate the input enable signal of $R_2 R_{2out} = 1$. This loads data from the common bus into the register R_2.
- All operations and data transfers within the processor take place in synchronisation with the clock signal.

Q.8 What are micro-operations ?

Ans. : • To perform fetch, decode and execute cycles, the processor unit has to perform set of operations called **micro-operations.** These operations include :

- Transfer a word of data from one CPU register to the another or to the ALU.
- Perform the arithmetic or logic operations on the data from the CPU registers and store the result in a CPU register.
- Fetch a word of data from specified memory location and load them into a CPU register.
- Store a word of data from a CPU register into a specified memory location.

Q.9 Explain the sequence of operations needed to perform fetching a word from memory and storing a word in memory.

☞ [SPPU : May-12, June-22, Marks 8]

Ans. : Fetching a word from memory

- To fetch a word of data from memory, the processor gives the address of the memory location where the data is stored on the address bus and activates the read operation.
- The processor loads the required address in MAR, whose output is connected to the address lines of the memory bus.
- At the same time processor sends the read signal of memory control bus to indicate the read operation.
- When the requested data is received from the memory it is stored into the MDR, from where it can be transferred to other processor registers.

Storing a word in memory

- To write a word of data into a memory location processor has to load the address of the desired memory location in the MAR, load the data to be written in memory, in MDR and activate write operation.

Q.10 Why is wait-for-memory-function-completed steps needed when reading from or writing from main memory ?

Ans. : • When access time of memory is larger than the maximum access time allowed in memory read cycle we need to check wait-for-memory-function - complete signal to varify the completion of memory read operation before going to start the next operation.

Q.11 State the actions needed to execute MOVE R_3, (R_2) instruction and also draw the timing diagram.

Ans. : The actions needed to execute MOVE R_3, (R_2) instruction are as follows :

1. MAR ← [R_2]

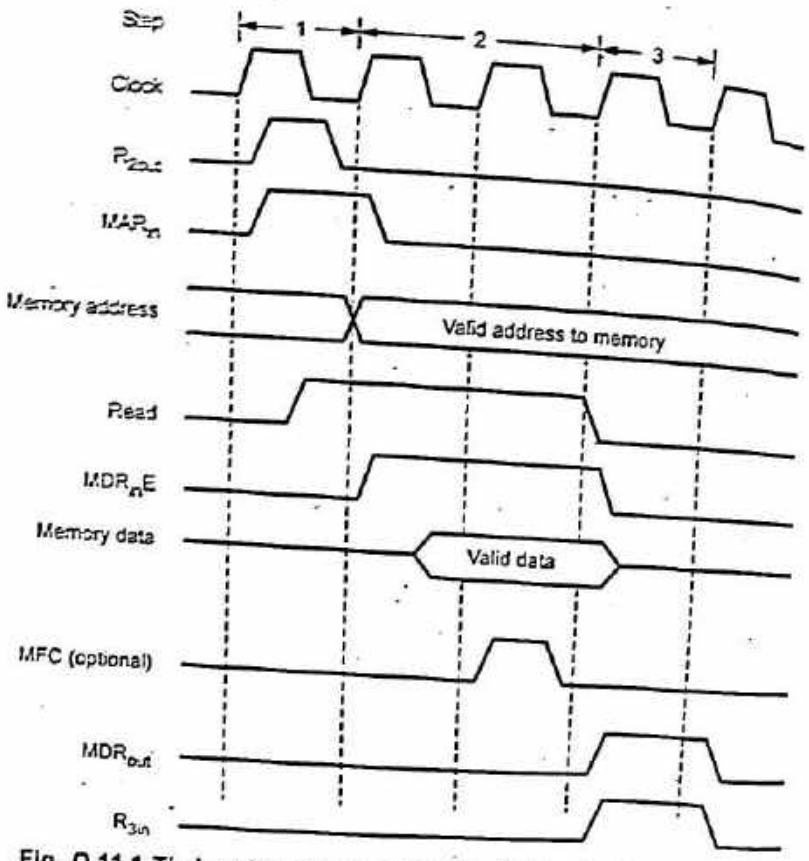


Fig. Q.11.1 Timing diagram for MOVE R_3 (R_2) instruction (memory read operation)

2. Activate the control signal to perform the Read operation. If memory is slow, activate wait for Memory Function Complete (MFC).

3. Load MDR from the memory bus

4. R_3 ← [MDR]

The Fig. Q.11.1 shows the timing diagram of a memory read operation.

Q.12 State the actions needed to execute MOVE (R_2), R_1 instruction and also draw the timing diagram.

Ans. :

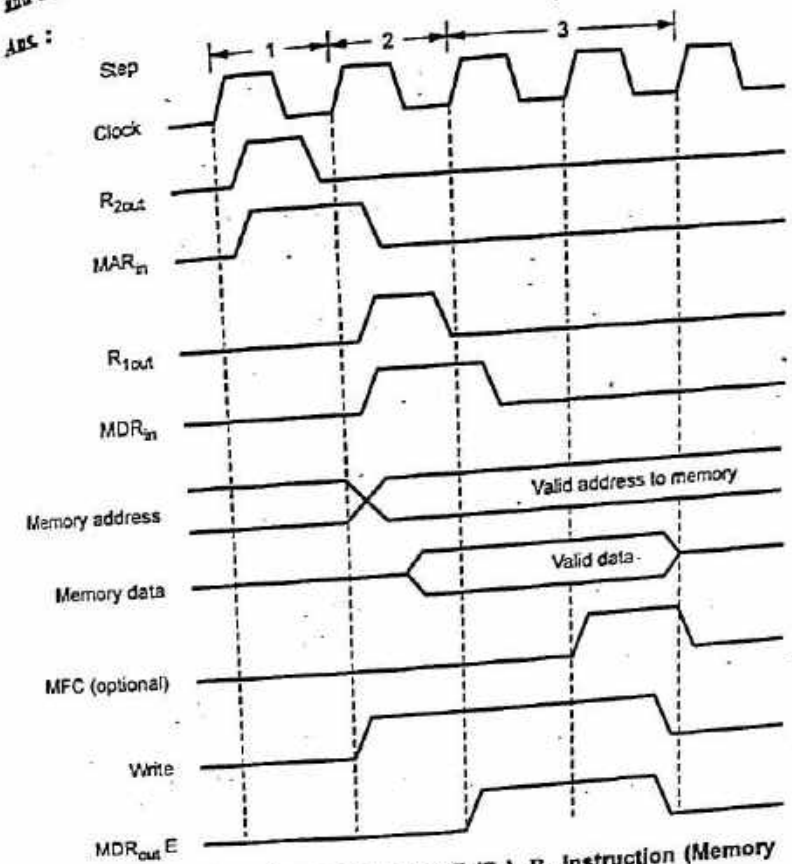


Fig. Q.12.1 Timing diagram for MOVE (R_2), R_1 instruction (Memory write operation)

- The actions needed to execute MOVE $(R_2), R_1$ instruction are as follow.

1. MAR $\leftarrow [R_2]$
2. MDR $\leftarrow [R_1]$
3. Activate the control signal to perform the write operation. If memory is slow, wait for Memory Function Complete (MFC).

The Fig. Q.12.1 (see on previous page) shows the timing diagram of a memory write operation.

Q.13 Explain the sequence of operation required for performing an arithmetic or logic operation.

Ans. : • ALU performs arithmetic and logic operations.

- It is a combinational circuit that has no internal memory.
- The ALU shown in Fig. Q.13.1 has two inputs A and B, and one output. Its A input gets the operand from the output of the multiplexer and its B input gets the operand directly from the bus. The result produced by the ALU is stored temporarily in register Z.
- The sequence of operations required to subtract the contents of register R_2 from register R_1, and store the result in register R_3 is given as follows :

Sr. No.	Action	Description
1.	R_{1out}, Y_{in}	The contents from register R_1 are loaded into register Y.
2.	R_{2out}, Select Y, Sub, Z_{in}	The contents from Y and from register R_2 are applied to the A and B inputs of ALU, respectively, subtraction is performed, and result is stored in the Z register.
3.	Z_{out}, R_{3in}	The contents of Z-register (result) is stored in the R_3 register.

Q.14 Explain the control sequence for unconditional branch instruction.

Ans. : The branch instruction loads the branch target address in PC so that PC will fetch the next instruction from the branch target address. The branch target address is usually obtained by adding the offset in the contents of PC. The offset is specified within the instruction. The control sequence for unconditional branch instruction is as follows :

Sr. No.	Action	Description
1.	PC_{out}, MAR_{in}, Read, SelectC, Add, Z_{in}	The instruction fetch operation is initiated by loading the contents of the PC into the MAR and activating Read signal. By activating select C input of multiplexer, a constant value is added to the operand at input B, which is the contents of the PC. By activating Z_{in} signal result is stored in the register Z.
2.	Z_{out}, PC_{in}, Yin WMFC	The contents of register Z are transferred to PC register by activating Z_{out} and PC_{in} signal. This completes the PC increment operation and PC will now point to next instruction. After receiving WMFC signal, the contents of specified location are available in MDR register.
3.	MDR_{out}, IR_{in}	The contents of MDR register are transferred to the Instruction Register (IR) of the processor. The step 1 through 3 constitute the instruction fetch phase. At the beginning of step 4, the instruction decoder interprets the contents of the IR. This enables the control circuitry to activate the control signals for steps 4 through 7, which constitute the execution phase.

4.	Offset_field_Of_IR_{out}, SelectY, Add, Z_{in}	The contents of PC and the offset field of IR register are added and result is saved in register Z by activating corresponding signals.
5.	Z_{out}, PC_{in}, End	The contents of register Z are transferred to PC by activating Z_{out} and PC in signals.

Q.15 Write a control sequence for branch on negative.

Ans. : • The Fig. Q.15.1 shows the implementation of instruction Branch on negative. (Branch < 0). When this instruction is loaded into IR, a branch microinstruction transfers control to the corresponding microroutine, which is assumed to start at location 45 in the control memory.

- This address is the output of the starting address generator block in Fig. Q.15.1.
- The microinstruction at location 45 tests the N bit of the condition codes. If this bit is equal to 0, a branch takes place to location 0 to fetch a new machine instruction. Otherwise, the microinstruction at location 46 is executed to put the branch target address into register Z. The microinstruction in location 47 loads this address into the PC.

Address	Microinstruction
0	PC_{out}, MAR_{in}, Read, Y_{in}, SelectC, Add, Z_{in}
1	Z_{out}, PC_{in}, WMFC
2	MDR_{out}, IR_{in}
3	Branch to starting address of appropriate microroutine
45	If N=0, then branch to microinstruction 0
46	offset_field_of_IR_{out}, SelectY, Add, Z_{in}
47	Z_{out}, PC_{in}, End

Fig. Q.15.1 Microroutine for the instruction branch < 0

Q.16 Explain the need of multibus organization.

Ans. : • With the single bus organization only one data word can be transferred over the bus in a clock cycle. This increases the steps required to complete the execution of the instruction. To reduce the number of steps needed to execute instructions, most commercial processors provide multiple internal paths that enable several transfer to take place in parallel.

Q.17 Draw and explain the multi-bus organization.

Ans. : • Fig. Q.17.1 shows a three-bus structure of the processor. Here, three buses are used to connect registers and the ALU of the processor. In the figure all general purpose registers are shown by a single block called register file. The register file shown in Fig. Q.17.1 has three ports : one input port and two output ports. So it is possible to access data of three register in one clock cycle; the value can be loaded in one register from bus C and data from two register can be accessed to bus A and bus B, respectively. Buses A and B are used to transfer the source operands to the A and B inputs of the ALU. After performing arithmetic or logic operation result is transferred to the destination operand over bus C. (Refer Fig. Q.17.1 on next page)

- To increment the contents of PC after execution of each instruction to fetch the next instruction, separate unit is provided. This unit is known as **incrementer**. Incrementer increments the contents of PC accordingly to the length of the instruction so that it can point to next instruction in the sequence. The incrementer eliminates the need of multiplexer connected at the A input of ALU.

Q.18 Explain the execution of Add R_1, R_2, R_3 instruction with multi-bus organization.

Ans. : • Add R_1, R_2, R_3. This instruction adds the contents of registers R_2 and the contents of register R_3 and stores the result in R_1. With three-bus organisation control steps required for execution of instruction Add R_1, R_2, R_3 are as follows :

1. PC_{out}, R = B, MAR_{in}, Read, IncPC 2. WMFC
3. MDR_{out}, IR_{in}, R = B 4. R_{2out}, R_{3out}, Add, R_{1in}, End

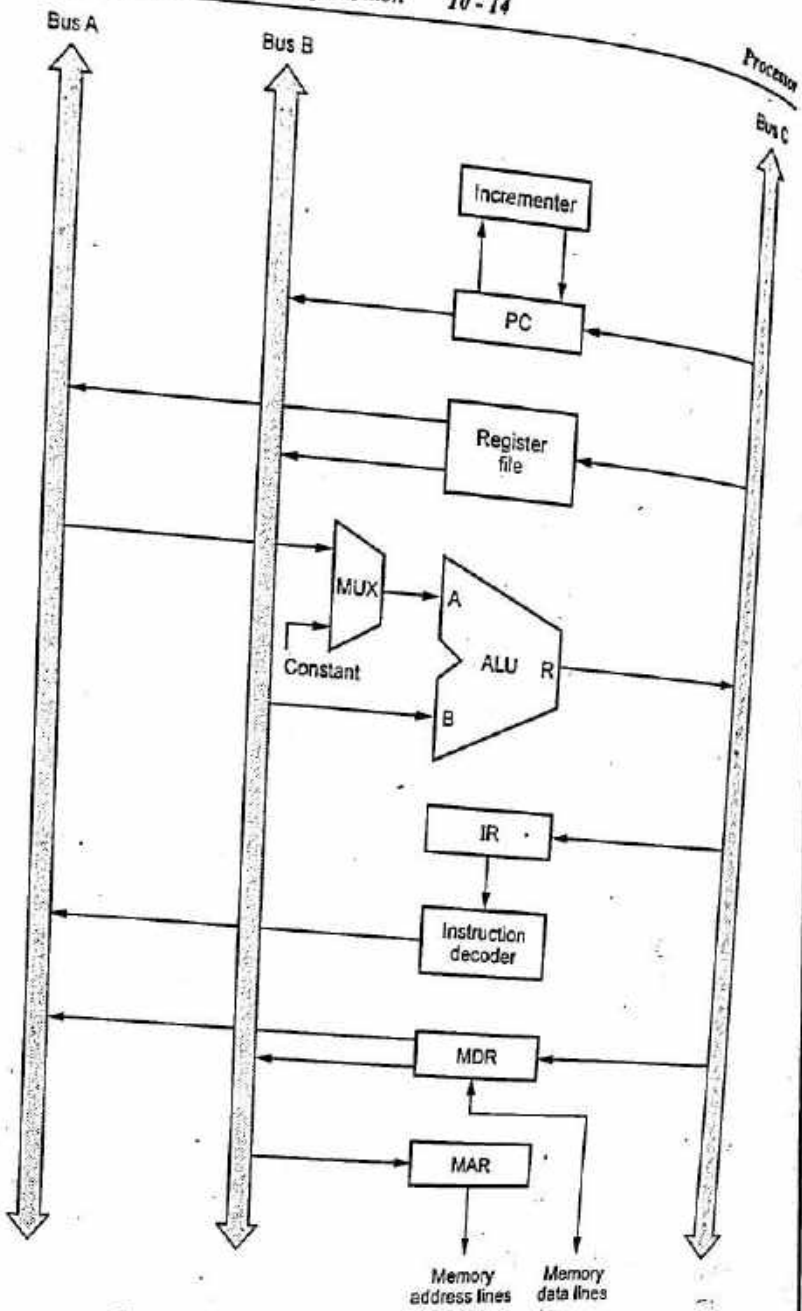


Fig. Q.17.1 Three-bus organisation of processor

- In step 1, the contents of PC are transferred to MAR through Bus B to start the Read operation and simultaneously PC is incremented to point the next instruction . In step 2, the processor waits for MFC. In step 3, the instruction code is transferred from MDR to IR register. At the beginning of step 4, the instruction decoder interprets the contents of the IR. This enables the control circuitry to activate the control signals for step 4, which constitute the execution phase.
- In step 4, two operands from register R_2 and register R_3 are made available at A and B inputs of ALU through BUS A and BUS B. These two inputs are added by activation of Add signal and result is stored in R_1 through Bus C.

Q.19 Explain how the instruction A = B – C gets executed in a system in detail.

Ans. : • Let us consider that registers A, B and C are available in the three-bus organization. With three-bus organization control steps required for execution of instruction A = B – C are as follows :

1. PC_{out}, R = B, MAR_{in}, Read, IncPC
2. WMFC
3. MDR_{out}, IR_{in}, R = B
4. B_{out}, C_{out}, Sub, A_{in}, End

- In step 1, the contents of PC are transferred to MAR through Bus B to start the Read operation and simultaneously PC is incremented to point the next instruction. In step 2, the processor waits for MFC. In step 3, the instruction code is transferred from MDR to IR register. At the beginning of step 4, the instruction decoder interprets the contents of the IR. This enables the control circuitry to activate the control signals for step 4, which constitute the execution phase.
- In step 4, two operands from register B and register C are made available at A and B inputs of ALU through BUS A and BUS B. The operand C is subtracted from operand B by activation of sub signal and result is stored in register A through bus C.

10.3 : Basic Concepts of Functional Organization of Hardwired Control Unit

Q.20 Draw and explain the block diagram of hardwired control unit.

☞ [SPPU : Dec.-06,16,17,18,19, May-14,18, Marks 6]

Ans. : • In the hardwired control, the control units use fixed logic circuits to interpret instructions and generate control signals from them.

- The fixed logic circuits use contents of the control step counter, contents of the instruction register, contents of the condition code flag and the external input signals such as MFC and interrupt requests to generate control signals.
- Fig. Q.20.1 shows the typical hardwired control unit. Here, the fixed logic circuit block includes combinational circuit (decoder and encoder) that generates the required control outputs, depending on the state of all its inputs.

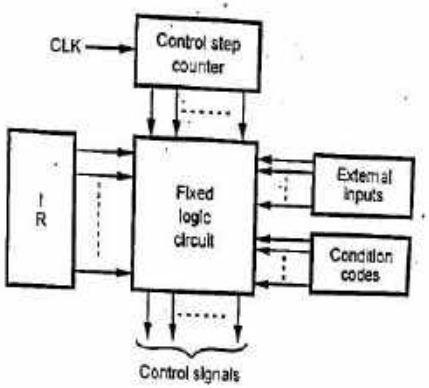


Fig. Q.20.1 Typical hardwired control unit

- By separating the decoding and encoding functions, we can draw more detail block diagram for hardwired control unit as shown in the Fig. Q.20.2.
- The instruction decoder decodes the instruction loaded in the IR. If IR is an 8-bit register then instruction decoder generates 2^8, i.e. 256 lines; one for each instruction.

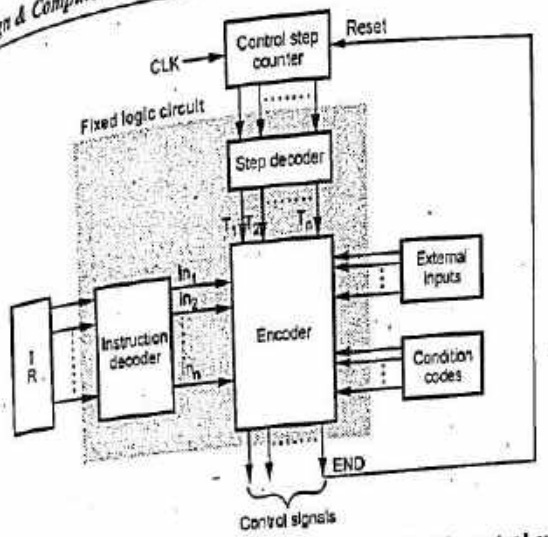


Fig. Q.20.2 Detail block diagram for hardwired control unit

- According to code in the IR, only one line amongst all output lines of decoder goes high i.e., set to 1 and all other lines are set to 0.
- The step decoder provides a separate signal line for each step or time slot, in a control sequence.
- The encoder gets in the input from instruction decoder, step decoder, external inputs and condition codes. It uses all these inputs to generate the individual control signals.
- After execution of each instruction end signal is generated which resets control step counter and make it ready for generation of control step for next instruction.

Q.21 With the help of circuit diagram, explain how Z_{in} signal is generated.

☞ [SPPU : Dec.-07, May-09, Marks 4]

Ans. : For the single bus organization shown in Fig. Q.1.1, the encoder circuit implements the following logic function to generate Z_{in}.

$$Z_{in} = T_1 + T_6 \cdot ADD + T_4 \cdot BR + \ldots\ldots$$

- Fig. Q.21.1 shows the generation of the Z_{in} control signal for single bus organization shown in Fig. Q.1.1. This signal is asserted during time slot T_1 for all instructions, during T_6 of an Add instruction, during T_4 for an unconditional branch instruction and so on.

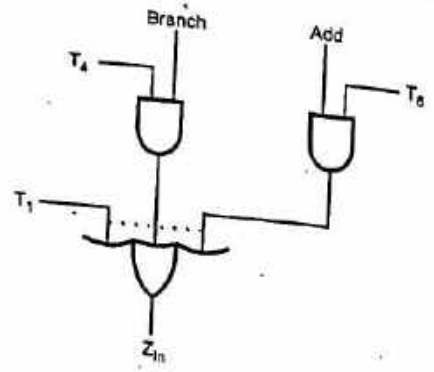


Fig. Q.21.1 Generation of the Z_{in} control signal

Q.22 With the help of circuit diagram, explain how end signal is generated.

[SPPU : Dec.-07, May-09, Marks 4]

Ans. : The logic function, to generate end signal is

$$End = T_7 \cdot ADD + T_5 \cdot BR + (T_5 \cdot N + T_4 \cdot \overline{N}) \cdot BRN + \ldots\ldots$$

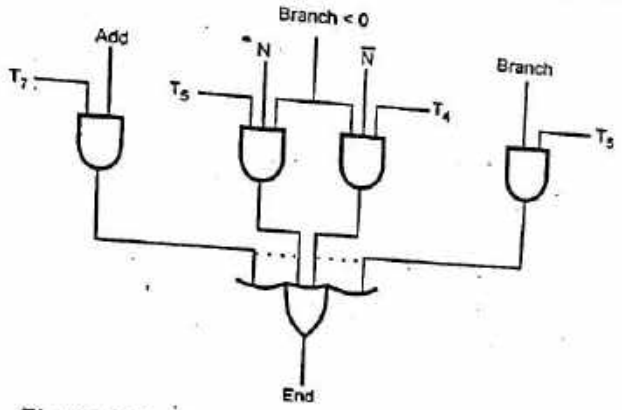


Fig. Q.22.1 Generation of the end control signal

Q.23 State the advantages and disadvantages of hardwired control unit.

Ans. : Advantages of Hardwired Control Unit

- Hardwired control unit is fast because control signals are generated by combinational circuits.
- The delay in generation of control signals depends upon the number of gates.
- It has greater chip area efficiency since its uses less area on-chip.

Disadvantages of Hardwired Control Unit

- More the control signals required by CPU; more complex will be the design of control unit.
- Modifications in control signal are very difficult. That means it requires rearranging of wires in the hardware circuit.
- It is difficult to correct mistake in original design or adding new feature in existing design of control unit.

10.4 : Basic Concepts of Functional Organization of Micro-programmed Control Unit

Q.24 Define the following terms :
i) Microoperation ii) Microinstruction
iii) Microcode iv) Microprogram.

Ans. :

i) **Micro-operation :** To perform fetch, decode and execute cycles, the processor unit has to perform set of operations called **micro-operations.**

ii) **Microinstruction :** Each word in the control memory is a microinstruction which specifies the control signals to be activated to perform one or more micro-operations.

iii) **Microcode :** The translation of symbolic microprogram to binary produces a binary microprogram called **microcode.**

iv) **Microprogram :** A sequence of one or more micro-operations designed to perform specific operation, such as addition, multiplication is called a microprogram.

Q.25 Define control memory.

Ans. : Microprogramming is a method of control unit design in which the control signal selection and sequencing information is stored in a ROM or RAM called a **control memory CM.**

Q.26 Discuss the basic structure of microprogrammed control unit.

☞ [SPPU : May-08,11,17,19, Dec.-08,10, Marks 8]

Ans. : Fig. Q.26.1 shows the microprogrammed control unit. It consists of control memory, control address register, micro instruction register and microprogram sequencer.

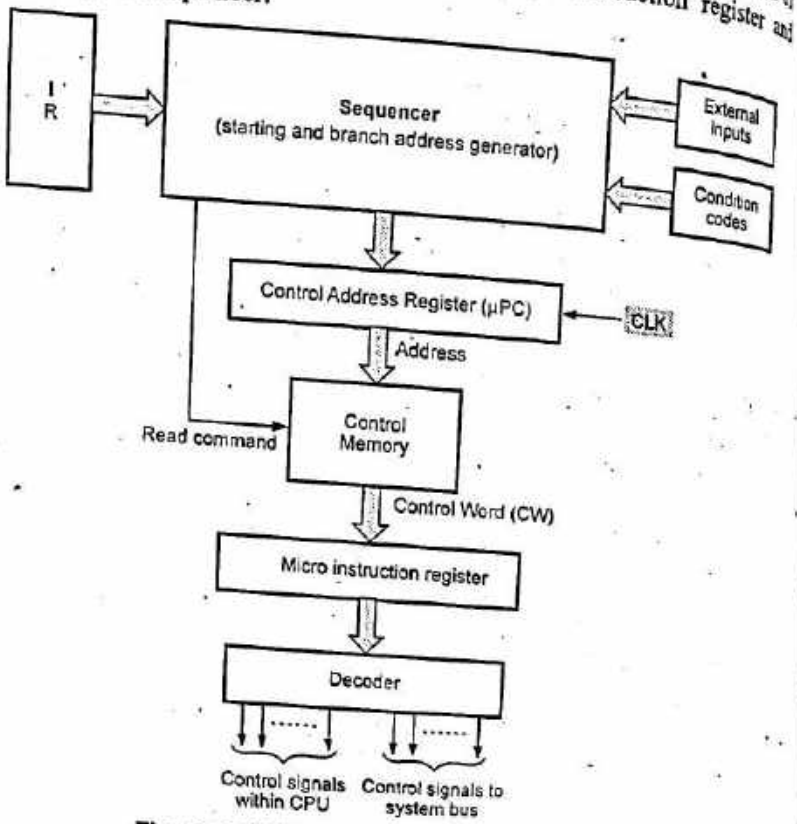


Fig. Q.26.1 Microprogrammed control unit

The components of control unit work together as follows :

- The control address register (μpc) holds the address of the next microinstruction to be read. Every time a new instruction is loaded into the IR, the output of the block labeled "starting address generator" is loaded into the μpc.
- When address is available in control address register, the sequencer issues READ command to the control memory.
- After issue of READ command, the word from the addressed location is read into the microinstruction register.
- The μpc is then automatically incremented by the clock, causing successive microinstructions to be read from the control memory.
- The content of the micro instruction register generates control signals which are delivered to various parts of the processor in the correct sequence.

- Number of times the control unit is required to check the status of the condition codes or external inputs to choose between alternative courses of action. In such situation, microprogrammed control use conditional branch microinstructions. In additions to the branch address, these instructions specify which of the external inputs, condition codes, or, possibly bits of the instruction register should be checked as a condition for branching to take place.

Q.27 State the advantages and disadvantages of microprogrammed control unit.

☞ [SPPU : May-06, 13, Dec.-12, Marks 8]

Ans. : Advantages :

- It simplifies the design of control unit. Thus it is both, cheaper and less error prone to implement.
- Control functions are implemented in software rather than hardware.
- The design process is orderly and systematic.
- More flexible, can be changed to accommodate new system specifications or to correct the design errors quickly and cheaply.
- Complex function such as floating point arithmetic can be realised efficiently.
- The new or modified instruction set of CPU can be easily implemented by simply rewriting or modifying the contents of control memory.

- The fault can be easily diagnosed in the micro-program control unit using diagnostics tools by maintaining the contents of flags, registers and counters.

Disadvantages :

- A microprogrammed control unit is somewhat slower than the hardwired control unit, because time is required to access the microinstructions from CM.
- The flexibility is achieved at some extra hardware cost due to the control memory and its access circuitry.
- The design duration of micro-program control unit is more than hardwired control unit for smaller CPU.

Q.28 Define verfical and horizontal organization.

Ans. : • Highly encoded scheme that use compact codes to specify only a small number of control functions in each microinstruction are referred to as a **vertical organisation.**

- The minimally encoded scheme, in which resources can be controlled with a single instruction, is called a **horizontal organisation.**

Q.29 Compare horizontal and vertical organisation.

☞ [SPPU : May-10,11,13, Dec.-07,08,09,10,12]

Ans. : Table Q.29.1 shows the comparison between horizontal and vertical organisation.

Sr. No.	Horizontal	Vertical
1.	Long formats.	Short formats.
2.	Ability to express a high degree of parallelism.	Limited ability to express parallel microoperations.
3.	Little encoding of the control information.	Considerable encoding of the control information.
4.	Useful when higher operating speed is desired.	Slower operating speeds.

Table Q.29.1

Q.30 Draw and explain the microinstruction sequencing organization.

☞ [SPPU : May-14, Marks 6]

Ans. : • Fig. Q.30.1 shows microprogram sequencer.

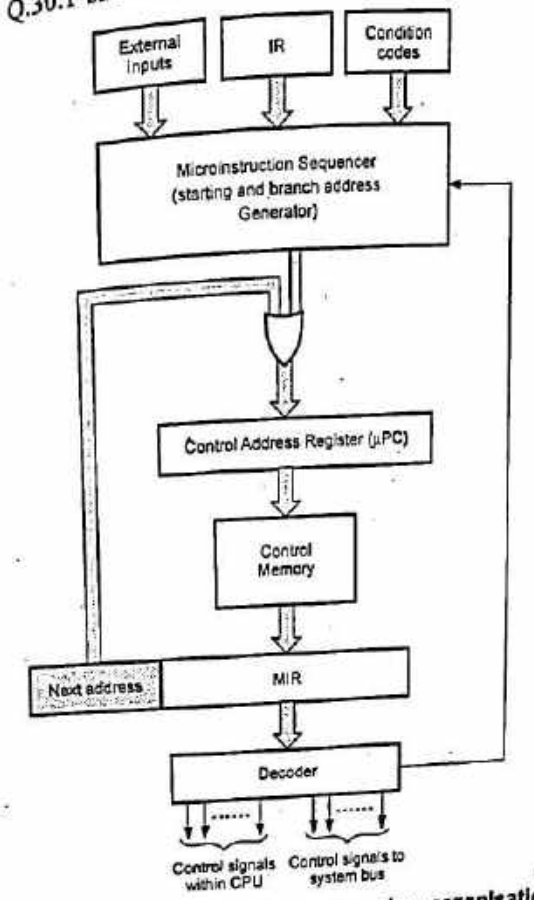


Fig. Q.30.1 Microinstruction sequencing organisation

- The two basic tasks performed by a microprogrammed control unit are :
  - **Microprogram sequencing :** Get the next microinstruction from the control memory.
  - **Microinstruction execution :** Generate the necessary control signals to execute the microinstruction.

- The task of microprogram sequencing is done by microprogram sequencer.
- In this microprogram sequencer, the address of the next instruction is stored in the special address field within the instruction.
- The branch address is loaded in CM address register when a branch condition is satisfied. The special address field within the microinstruction increases the size of the microinstruction. The size of the microinstruction can be reduced by storing part of the address (low order bits) in the microinstruction. This restricts the range of branch instructions to a small region of the CM and may therefore increase the difficulty of writing some microprograms.

Q.31 Explain how microinstructions can be shared using microinstruction branching ?

Ans. : Consider instruction ADD R_{src}, R_{dst}. The instruction adds the source operand to the contents of register R_{dst} and places the sum in R_{dst}, the destination register. Let us assume that the source operand can be specified in the following addressing modes : Indexed, autoincrement, autodecrement, register indirect and register direct. We now use this instruction in conjunction with the CPU structure shown in Fig. Q.31.1 to demonstrate a possible microprogrammed implementation. Fig. Q.31.2 shows a flowchart of a microprogram for the ADD R_{src}, R_{dst} instruction. Each box in the flowchart corresponds to a microinstruction that controls the transfers and operations indicated within the box. The microinstruction is located at the address indicated by the number above the upper right-hand corner of the box. During the execution of the microinstruction, the branching takes place at point A. The branching address is determined by the addressing mode used in the instruction. (Refer, Fig. Q.31.2 on page no. 10 - 26).

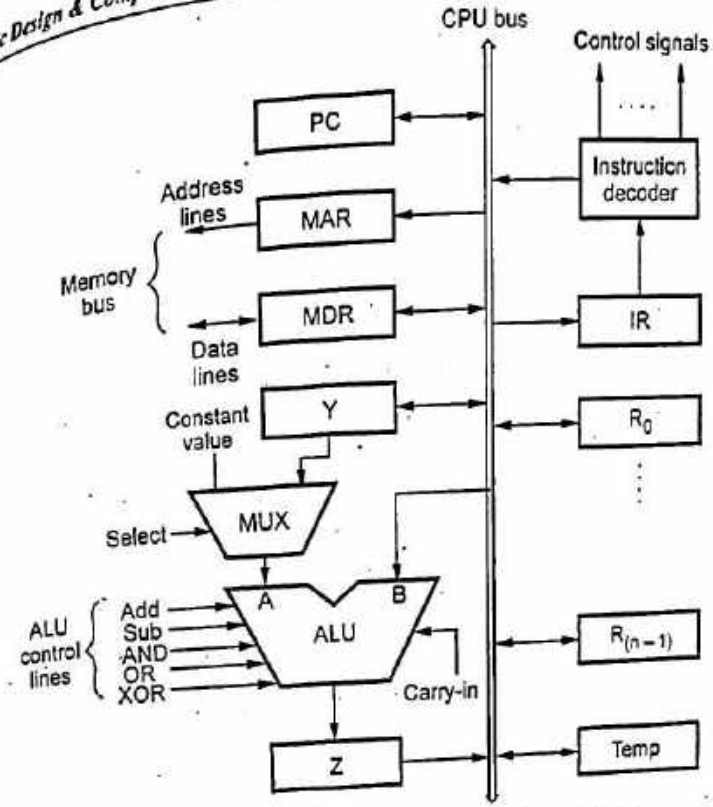


Fig. Q.31.1 Single bus organisation of processor

At point B, it is necessary to choose between actions required by direct and indirect addressing modes. If the indirect mode is specified in the instruction, then the microinstruction in location 170 is performed to fetch the operand from the memory. If the direct mode is specified, this fetch must be bypassed by branching immediately to location 171. The remaining microoperations required to complete execution of instruction are same and hence shared by the instructions having operand with different addressing modes.

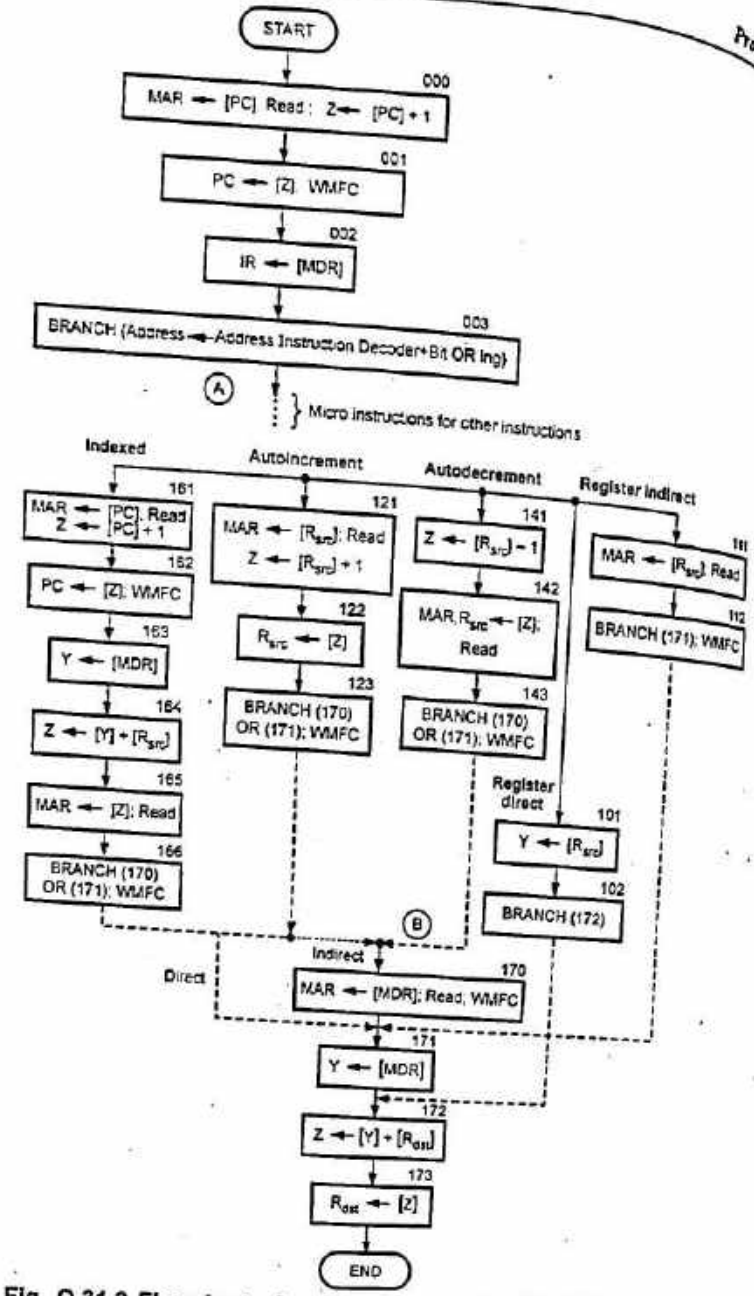


Fig. Q.31.2 Flowchart of a microprogram for the ADD (R_{src}), R_{dst} instruction

Q.32 Explain the techniques for modification or generation of branch addresses.

Ans. : Bit-ORing

- In this technique, the branch address is determined by ORing particular bit or bits with the current address of the microinstruction.
- For example, if the current address is 170 and the branch address is 172 then the branch address can be generated by ORing 02 (bit 1), with the current address.

Using Condition Variables

- In this technique the condition variables are used to modify the contents of the CM address register directly, thus eliminating whole or in part the need for branch addresses in microinstructions.
- For example, let the condition variable CY indicate occurrence of CY = 1 and no carry when CY = 0.
- Suppose that we want to execute a SKIP_ON_CARRY microinstruction. This can be done by logically connecting CY to the count enable input of μpc at on appropriate point in the microinstruction cycle.
- This allows the overflow condition to increment μpc an extra time, thus performing the desired skip operation.

Wide-Branch Addressing

- Generating branch addresses becomes more difficult as the number of branches increases. In such situations Programmable Logic Array can be used to generate the required branch addresses.
- This simple and inexpensive way of generating branch addresses is known as **wide-branch addressing.**
- Here, the opcode of a machine instruction is translated into the starting address of the corresponding micro-routine.

- This is achieved by connecting the opcode bits of the instruction register as inputs to the PLA, which acts as a decoder. The output of the PLA is the address of the desired microroutine.

Q.33 For a single bus organisation of CPU, write a microprogram for instruction. Add $(R_{src})+$, R_{dst}. ☞ [SPPU : Dec.-06, May-07, Marks 8]

Ans. : Let us examine the path needed for the flowchart in Fig. Q.31.2 to execute the instruction Add$(R_{src})+$, R_{dst}.

In this instruction the source operand is accessed in the autoincrement mode and the R_{src} and R_{dst} are general purpose registers in the processor. We assume that the processor has 16 registers that can be used for addressing purpose, each specified using a 4-bit code. We also assume that the instruction has a 3-bit field, (bits 8-10) used to specify the addressing mode for the source operand, as shown in the Fig. Q.33.1. Bit patterns 11,10,01 and 00 located in bits 10 and 9 denote the indexed, autodecrement, autoincrement and register modes respectively. For each of these modes bit-8 is used to specify the indirect version. For example, 100 in the mode field specifies the direct version of the autodecrement mode, whereas 101 specifies the indirect version.

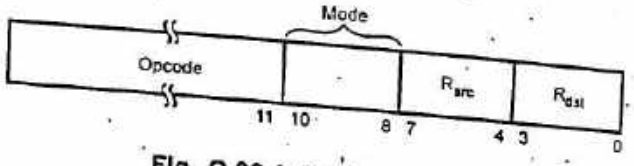


Fig. Q.33.1 Contents of IR

As a part of execution, first the opcode and mode fields are decoded to determine that an R_{src} or R_{dst} register is involved. The decoded output is then used to gate the contents of the R_{src} or R_{dst} fields in the IR into a second decoder, which produces the gating signals for the actual registers R_0 to R_{15}.

The flowchart of a microprogram for the ADD R_{src}, R_{dst} instruction in Fig. Q.31.2 is drawn by combining the microroutines for all possible values of the mode field, resulting in a structure that requires many branch points. The instruction Add $(R_{src})+$, R_{dst} requires two branch microinstructions. In each branch microinstruction, the expression in brackets indicates the branch address that is to be loaded into the µpc and how this address is modified using the bit-ORing scheme. For example,

the branch instruction at location 123 modifies the branch address 170 to 171 by ORing the bit-8 in the IR with bit µpc0 to change the addressing mode from indirect to direct, as shown in Fig. Q.33.2.

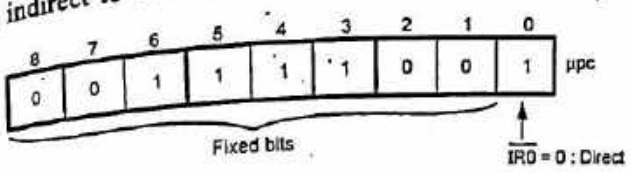


Fig. Q.33.2

The address for branch microinstruction at location 003 is generated as shown in Fig. Q.33.3.

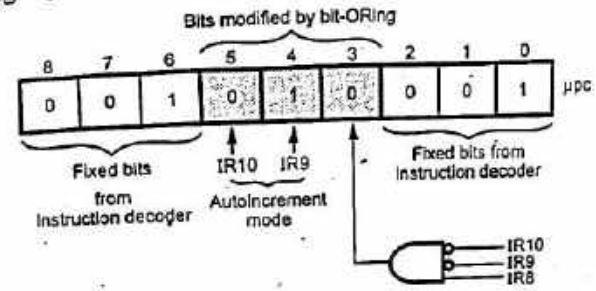


Fig. Q.33.3 Bit ORing for branch microinstruction at location 003

Table Q.33.1 gives the microinstruction sequence for the execution of Add $(R_{src})+$, R_{dst} instruction.

Address (Octal)	Microinstruction
000	PC_{out}, MAR_{in}, Read, SelectC, Add, Z_{in}
001	Z_{out}, PC_{in}, Y_{in}, MWFC
002	MDR_{out}, IR_{in}

0 0 3	μBranch $\begin{cases}\mu pc \leftarrow (101)_8 \text{ from instruction decoder} \\ \mu pc_{5,4} \leftarrow [IR_{10,9}],\ \mu pc_3 \leftarrow [\overline{IR}_{10}] \cdot [\overline{IR}_9] \cdot [IR_8]\end{cases}$ i.e. $\mu pc_{5,4,3} \leftarrow (010)_2$
⋮	
1 2 1	R_{srcout}, MAR_{in}, Read, SelectC, Add, Z_{in}
1 2 2	Z_{out}, R_{srcin}
1 2 3	μBranch {$\mu pc \leftarrow (170)_8$ from instruction decoder $\mu pc_0 \leftarrow \overline{IR}_8$}, WMFC
⋮	
1 7 1	MDR_{out}, Y_{in}
1 7 2	R_{dstout}, SelectY, Add, Z_{in}
1 7 3	Z_{out}, R_{dstout}, End

Table Q.33.1 Microinstruction for Add $(R_{src})+$, R_{dst}

Q.34 For a single bus organisation of CPU draw flowchart of a microprogram for instruction Add $(R_{src})+$, R_{dst}.

☞ [SPPU : Dec.-06, May-07, Marks 8]

Ans. : Flowchart

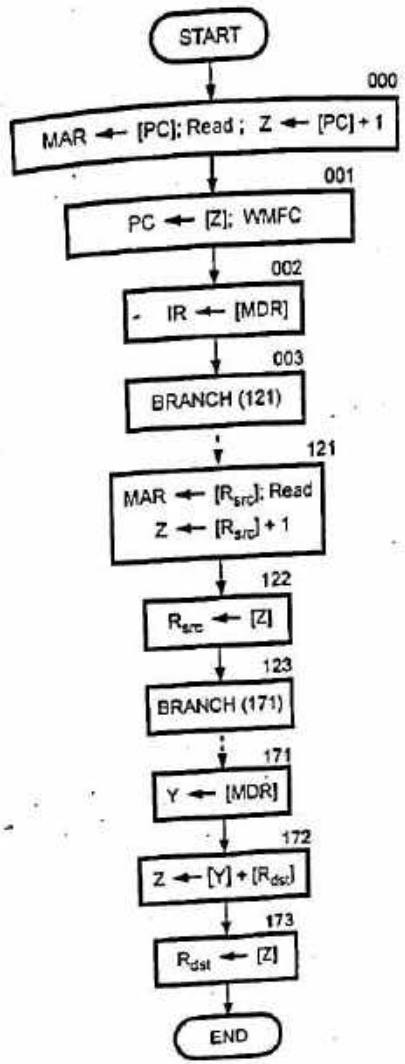


Fig. Q.34.1

Q.35 For a single bus organisation of data paths inside the CPU, write a microprogram of micro-instructions and draw chart of a microprogram for the following instruction. *MOV* (R_{src}), R_{dst}. R_{dst} uses direct and R_{src} autoincrement addressing.

☞ [SPPU : Dec.-10, Marks 8]

Ans. :

Address (Octal)	Microinstruction
0 0 0	PC_{out}, MAR_{in}, Read, SelectC, Add, Z_{in}
0 0 1	Z_{out}, PC_{in}, Y_{in}, MWFC
0 0 2	MDR_{out}, IR_{in}
0 0 3	μBranch $\begin{cases} \mu pc \leftarrow (101)_8$ from instruction decoder $\mu pc_{5,4} \leftarrow [IR_{10,9}]$, $\mu pc_3 \leftarrow [\overline{IR}_{10}] \cdot [\overline{IR}_9] \cdot [IR_8]$ $\end{cases}$
:	
1 1 1	R_{srcout}, MAR_{in}, Read
1 1 2	μBranch {$\mu pc \leftarrow (270)_8$ from instruction decoder $\mu pc_0 \leftarrow \overline{IR}_8$}, WMFC
2 7 1	MDR_{out}, R_{dstin}, End

Flowchart

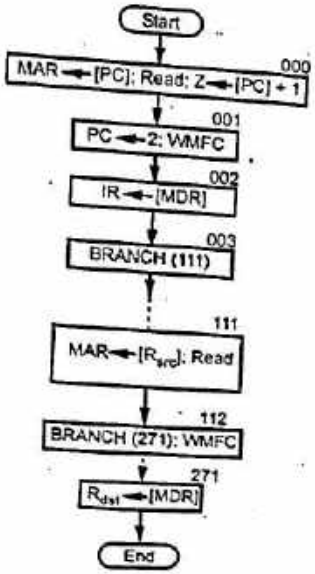


Fig. Q.35.1

Q.36 For a single bus organisation of CPU draw a flowchart and write microinstructions for instruction ADD (R_{src}), R_{dst}.

☞ [SPPU : Dec.-07,10,18, May-09,10,18, Marks 8]

Ans. : Microinstructions

Address (Octal)	Microinstruction
0 0 0	PC_{out}, MAR_{in}, Read, SelectC, Add, Z_{in}
0 0 1	Z_{out}, PC_{in}, Y_{in}, MWFC
0 0 2	MDR_{out}, IR_{in}
0 0 3	μBranch $\begin{cases} \mu pc \leftarrow (101)_8$ from instruction decoder $\mu pc_{5,4} \leftarrow [IR_{10,9}]$, $\mu pc_3 \leftarrow [\overline{IR}_{10}] \cdot [\overline{IR}_9] \cdot [IR_8]$ $\end{cases}$ i.e. $\mu pc_{5,4,3} \leftarrow (001)_2$
1 1 1	R_{srcout}, MAR_{in}, Read
	μBranch {$\mu pc \leftarrow (170)_8$ from instruction decoder $\mu pc_0 \leftarrow \overline{IR}_8$}, WMFC
1 1 2	μBranch {$\mu pc \leftarrow (170)_8$ from instruction decoder $\mu pc_0 \leftarrow \overline{IR}_8$}, WMFC
:	
1 7 1	MDR_{out}, Y_{in}
1 7 2	R_{dstout}, SelectY, Add, Z_{in}
1 7 3	Z_{out}, R_{dstin}, End

Table Q.36.1 Microinstruction for Add (R_{src}), R_{dst}

Flowchart

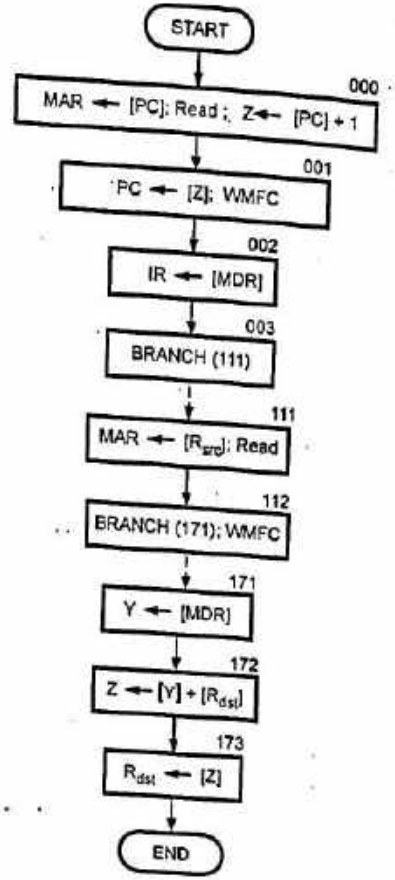


Fig. Q.36.1

Q.37 For a single bus organisation of CPU draw a flowchart and write microinstructions for instruction SUB (R_{src}), R_{dst}.

☞ **[SPPU : May-06,08, Dec.-05,12, Marks 8]**

Ans. : Microinstructions

Address (Octal)	Microinstruction
000	PC_{out}, MAR_{in}, Read, SelectC, Add, Z_{in}
001	Z_{out}, PC_{in}, Y_{in}, MWFC
002	MDR_{out}, IR_{in}
003	μBranch $\{\mu pc \leftarrow (101)_8$ from instruction decoder; $\mu pc_{5,4} \leftarrow [IR_{10,9}]$, $\mu pc_3 \leftarrow [\overline{IR}_{10}] \cdot [\overline{IR}_9 \cdot [IR_8]\}$; i.e. $\mu pc_{5,4,3} \leftarrow (001)_2$
111	R_{srcout}, MAR_{in}, Read
	μBranch $\{\mu pc \leftarrow (170)_8$ from instruction decoder; $\mu pc_0 \leftarrow \overline{IR}_8\}$, WMFC
112	μBranch $\{\mu pc \leftarrow (170)_8$ from instruction decoder; $\mu pc_0 \leftarrow \overline{IR}_8\}$, WMFC
	⋮
171	MDR_{out}, Y_{in}
172	R_{dstout}, SelectY, Sub, Z_{in}
173	Z_{out}, R_{dstin}, End

Table Q.37.1 Microinstruction for Sub (R_{src}), R_{dst}

Flowchart

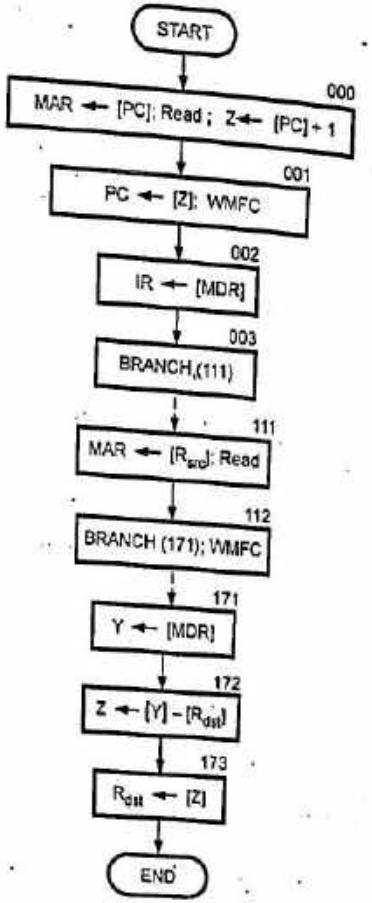


Fig. Q.37.1

Q.38 Give comparison between hardwired control and micro - programmed control.

Ans. :

Attribute	Hardwired control	Microprogrammed control
Speed	Fast	Slow
Control functions	Implemented in hardware	Implemented in software
Flexibility	Not flexible, to accommodate new system specifications or new instructions.	More flexible, to accommodate new system specification or new instructions redesign is required.
Ability to handle large/complex instruction sets	Somewhat difficult	Easier
Ability to support operating systems and diagnostic features	Very difficult (unless anticipated during design)	Easy
Design process	Somewhat complicated	Orderly and systematic
Applications	Mostly RISC microprocessors	Mainframes, some microprocessors
Instruction set size	Usually under 100 instructions	Usually over 100 instructions
ROM size	—	2K to 10K by 20-400 bit microinstructions
Chip area efficiency	Uses least area	Uses more area

END...

11.3 : Instruction Formats (0-1-2-3 Address Formats)

Q.5 What are the common fields in the instruction format ?

Ans. : The most common fields in instruction formats are :

1. **Opcode field :** It specifies operation to be performed.
2. **Address field :** It designates a memory address or a processor register.
3. **Mode field :** It specifies the way the operand or the effective address is determined.
4. **Addressing mode field :** It specifies the addressing mode to be used to access operand.

Q.6 If an instruction code has 4 bit opcode and 12 bit address field then

i) How many operations this code can perform.

ii) How many memory locations can be addressed.

Ans. :

i) Number of operations can be performed = $2^4 = 16$

ii) Number of memory locations can be addressed = $2^{12} = 4096$

Q.7 A computer uses a memory unit with 256 K words of 32 bits each. A binary instruction code is stored in one word of memory. The instruction has four parts : An indirect address, an operation code, the register code part to specify one of 64 registers and an address part : 10

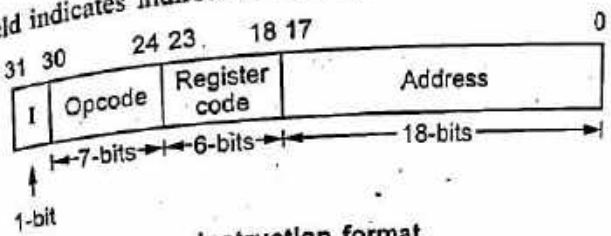
i) How many bits are there in the operation code, the register code part and the address part ? ii) Draw the instruction word format and indicate the number of bits in each part.

iii) How many bits are there in the data and address inputs of the memory ?

Ans. : i) The total bits specified in the instruction are 32. For address part we need 18 bits ($2^{18} = 256$ K), for register code part we need 6 bits ($2^6 = 64$) and for I field we need 1 bit. The remaining bits ($32 - 18 - 1 - 6 - 1) = 7$) are used for opcode.

ii) The instruction format for the given specification is as shown in the below

The I field indicates indirect addressing mode.



Instruction format

iii) The memory has 256 K memory locations and hence it has 18 address inputs. Since it stores 32-bits words, its data inputs are 32.

Q.8 The memory unit of a computer has 256 k works of 32 bit each. The computer has an instruction format with 4 fields. An operation field a mode field to specify one of seven addressing modes, a register address a field to specify one of 60 processor registers and a memory address. Specify the instruction format and the number of bits in each field if the instruction is in one memory word.

Ans. : To address 256 K words we need 18 ($2^{18} = 256$ K) address lines.

To specify seven addressing modes we need 3 bits.

To specify 60 processor registers we need 6 bits

Number of opcode bits = $32 - 18 - 3 - 6 = 5$

Instruction format

Opcode	Mode	Regiser	Address
5	3	6	18

= 32

Q.9 Write a short note on instruction types.

☞ [SPPU : Dec.-08, Marks 4, June-22, Marks 3]

OR Explain different types of instruction format according to address references.

Ans. : According to address references there are four types of instruction formats.

- Three address,
- Two address,
- One address and
- Zero address reference instructions.
- **Three address instructions :** The three address instruction can be represented symbolically as ADD C, A, B where A, B, C are the variables. (C ← A + B) These variable names are assigned to distinct locations in the memory. In this instruction operands A and B are called source operands and operand C is called destination operand and ADD is the operation to be performed on the operands.
- **Two address instructions :** The two address instruction can be represented symbolically as ADD A, B. This instruction adds the contents of variables A and B and stores the sum in variable A destroying its previous contents (A ← A + B). Here, operand B is source operand; however operand A serves as both source and destination operand.
- **One address instruction :** The one address instruction can be represented symbolically as ADD B. This instruction adds the contents of variable B into the processor register called **accumulator** (AC) and stores the sum back into the accumulator destroying the previous contents of the accumulator (AC ← AC+B).
- **Zero address instructions :** In these instructions, the locations of all operands are defined implicitly. For example, CMA instruction complements the contents of accumulator (AC) and stores complemented contents back into the accumulator (AC ← $\overline{AC}$). Such instructions are also found in machines that store operands in a structure called a pushdown **stack.**

Q.10 Write a program to evaluate the arithmetic statement $Y = (A+B)*(C+D)$ using three-address, two-address, one-address and zero-address instructions.

Ans. : Using three address instructions

```
ADD  R1, A, B  ;  R1 ← M[A] + M[B]
ADD  R2, C, D  ;  R2 ← M[C] + M[D]
```



```
MUL  Y, R1, R2 ;  M[Y] ← R1 * R2
```

Using two address instructions

```
MOV  R1,  A   ;  R1 ← M[A]
ADD  R1,  B;     R1 ← R1 + M[B]
MOV  R2,  C   ;  R2 ← M[C]
ADD  R2,  D   ;  R2 ← R2 + M[D]
MUL  R1,  R2  ;  R1 ← R1 * R2
MOV  Y,   R1  ;  M[Y] ← R1
```

Using one address instruction

```
LOAD  A ; AC ← M[A]
ADD   B ; AC ← AC + M[B]
STORE T ; M[T] ← AC
LOAD  C ; AC ← M[C]
ADD   D ; AC ← AC + M[D]
MUL   T ; AC ← AC * M[T]
STORE Y ; M[Y] ← AC
```

Using zero address instructions

```
PUSH  A ; TOS ← A
PUSH  B ; TOS ← B
ADD     ; TOS ← (A + B) - zero address instruction
PUSH  C ; TOS ← C
PUSH  D ; TOS ← D
ADD     ; TOS ← (C + D) - zero address instruction
MUL     ; TOS ← (C + D) * (A + B) ← zero address instruction.
POP   Y ; M[Y] ← TOS
```

Q.11 Write a program to evaluate the following arithmetic statement
$X = [A*(B+C)-D]/(E+F-G)$
i) Using a general register computer with three–address instructions,
ii) Using an accumulator type computer with one address instructions,
iii) Using a stack organized computer with zero–address operation instructions.

Ans. : i) Using general register computer with three-address instructions

```
ADD   R1,B, C ;  R1 ← M[B] + M[C]
MUL   R2,R1,A ;  R2 ← R1 * M [A]
SUB   R3,R2,D ;  R3 ← R2 – M[D]
ADD   R1,E, F ;  R1 ← M[E] + M[F]
SUB   R2,R1, G ;  R2 ← R1 – M[G]
DIV   X, R3,R2 ;  X ← R3/R2
```

ii) Using an accumulator type computer with one address instructions

```
LOAD   E ;  AC ← M [E]
ADD    F ;  AC ← AC + M [F]
SUB    G ;  AC ← AC – M [G]
STORE  T ;  M[T] ← AC
LOAD   B ;  AC ← M [B]
ADD    C ;  AC ← AC + M [C]
MUL    A ;  AC ← AC * M [A]
SUB    D ;  AC ← AC – M [D]
DIV    T ;  AC ← AC/ M [T]
STORE  X ;  M [X] ← AC
```

iii) Using stack organized computer with zero-address instructions

```
PUSH   B ;  TOS ← B
PUSH   C ;  TOS ← C
ADD      ;  TOS ← (B + C)
PUSH   A ;  TOS ← A
MUL      ;  TOS ← A (B + C)
PUSH   D ;  TOS ← D
SUB      ;  TOS ← A(B + C) – D
PUSH   E ;  TOS ← E
PUSH   F ;  TOS ← F
ADD      ;  TOS ← (E + F)
PUSH   G ;  TOS ← G
ADD      ;  TOS ← (E + F – G)
DIV      ;  TOS ← [A (B + C) – D] / (E + F – G)
POP    X ;  M[X] ← TOS
```


Q.12 Write the program to evaluate the expression $X = \frac{A[B+C(D+E)]}{F(G+H)}$ using the zero address instructions and one address instructions.

Ans. : Program using zero address instructions

```
PUSH   D ;  TOS ← D
PUSH   E ;  TOS ← E
ADD      ;  TOS ← (D + E)
PUSH C   ;  TOS ← C
MUL      ;  TOS ← C × (D + E)
PUSH   B ;  TOS ← B
ADD      ;  TOS ← B + C × (D + E)
PUSH   A ;  TOS ← A
MUL      ;  TOS ← A [B + C × (D + E)]
PUSH   G ;  TOS ← G
PUSH   H ;  TOS ← H
ADD      ;  TOS ← G + H
PUSH   F ;  TOS ← F
MUL      ;  TOS ← F × (G + H)
DIV      ;  TOS ← A[B + C × (D + E)]/F × (G + H)
POP    X ;  M [X] ← TOS
```

Program using one address instructions

```
LOAD   H ;  AC ← M[H]
ADD    G ;  AC ← AC + M[G]
MUL    F ;  AC ← AC * M[F]
STORE  T ;  M[T] ← AC
LOAD   D ;  AC ← M[D]
ADD    E ;  AC ← AC + M[E]
MUL    C ;  AC ← AC * M[C]
ADD    B ;  AC ← AC + M[B]
MUL    A ;  AC ← AC * M[A]
DIV    T ;  AC ← AC/M[T]
STORE  X ;  M[X] ← AC
```

Q.13 $X = A \times B + C \times C$

Explain how the above expression will be executed in one address, two address and three address processors in an accumulator organization.

Ans. : Using one address instruction

```
LOAD   A ;  AC ← M[A]
MUL    B ;  AC ← AC * M[B]
STORE  T ;  M[T] ← AC
LOAD   C ;  AC ← M[C]
MUL    C ;  AC ← AC * M[C]
ADD    T ;  AC ← AC + M[T]
STORE  X ;  M[X] ← AC
```

Using two address instructions

```
MOV A, R1    ;  R1 ← M[A]
MUL B, R1    ;  R1 ← R1 * M[B]
MOV C, R2    ;  R2 ← M[C]
MUL C, R2    ;  R2 ← R2 * M[C]
ADD R2, R1   ;  R1 ← R1 + R2
MOV R1, X    ;  M[T] ← R1
```

Using three address instructions

```
MUL A, B, R1 ;  R1 ← M[A] * M[B]
MUL C, C, R2 ;  R2 ← M[C] + M[C]
ADD R1, R1, X;  M[X] ←R1 * R2
```

11.4 : Types of Operands

Q.14 Write a note on type of operands. [SPPU : June-22, Marks 9]

Ans. : The operand may appear in different forms; these are -

- **Addresses :** The addresses are in fact a form of data. In many situations, some calculation must be performed on the operand reference in an instruction to determine physical address. In this context, addresses can be considered as unsigned integer operands.

- **Numbers :** All computer supports numeric data types. The common numeric data types are :
  - Integer or fixed point
  - Floating point
  - Decimal.
- **Characters :**
  - For documentation a common form of data is text or character strings.
  - Today, most of the computers use ASCII (American Standard Code for Information Interchange) code for character represented by a unique 7-bit pattern ; thus, 123 different characters can be represented.
  - However, the ASCII encoded characters are always stored and transmitted using 8-bits per character. The eighth bit may be set to 0 or used as a parity bit for error detection.
- **Logical data :**
  - Most of the processors interpret data as a bit, byte, word or double word. These are referred to as units of data.
  - When data is viewed as n 1-bit items of data, each item having the value 0 or 1, it is considered as a **logical data.**
  - The logical data is used to store an array of Boolean or binary data items and with logical data we can manipulate the bits of data items.

11.5 : Types of Operations/Instructions

Q.15 List the various types of operations supported by most of the processors.

Ans. : Generally type of operations supported by most of the machines can be categorized as follows :

- Data transfer operations
- Arithmetic operations

- Logical operations
- Conversion operations
- I/O operations
- System control operations
- Transfer of control operations.

Q.16 State any four data transfer operations.

Ans. : Data transfer operations :

Operation	Description
Move (transfer)	Transfers word or block from source to destination.
Store	Transfers word from processor to memory.
Load (fetch)	Transfers word from memory to processor.
Exchange	Swaps contents of source and destination.
Clear (reset)	Transfers word of 0s to destination.
Set	Transfers word of 1s to destination.
Push	Transfers word from source to top of stack.
Pop	Transfers word from top of stack to destination.

Q.17 State any four arithmetic operations.

Ans. : Arithmetic operations :

Operation	Description
Add	Performs addition of two operands.
Subtract	Performs subtraction of two operands.
Multiply	Performs multiplication of two operands.
Divide	Performs division of two operands.
Absolute	Replaces operand by its absolute value.

Negate	Changes sign of operand.
Increment	Adds 1 to operand.
Decrement	Subtracts 1 from operand.

Q.18 List any four logical operations.

OR State the difference between shift and rotate operations.

OR State the difference between logical shift and arithmetic shift operation.

Ans. : Logical operations :

Operation	Description
AND	Performs logical AND.
OR	Performs logical OR.
NOT (complement)	Performs logical NOT.
Exclusive - OR	Performs logical XOR.
Test	Tests specified condition and sets flag(s) accordingly.
Compare	Makes logical or arithmetic comparison or two of more operands and sets flag(s) accordingly.
Set control variables	Sets controls for protection purposes, interrupt handling, timer control etc.
Logical shift	There are two logical shift instructions logical shift left (LShiftL) and logical shift right (LShiftR). These two instructions shift an operand by a number of the positions specified in a count operand. The vacant positions created within the register due to shift operation are filled with zeroes.

Arithmetic shift	Arithmetic shift right operation repeats the sign bit as the fill-in bit for the vacant position. Arithmetic shift left operation is same as logical shift left operation.
Rotate	In shift instructions, the bits shifted out of the operand are lost, except for last bit shifted out which is retained in the carry flag C. The rotate left (right) instructions, preserve all bits. They move the bits that are shifted out of one end of the operand back into the other end.

Q.19 State the use of translate and convert operations.

Ans. : Conversion operations :

Operation	Description
Translate	Translates values in a section of memory based on a table of correspondences.
Convert	Converts the contents of a word from one form to another (e.g. packed decimal to binary).

Q.20 Explain any two I/O operations.

Ans. : I/O operations :

Operation	Description
Input (read)	Transfer data from specified I/O port or device to destination (e.g., main memory or processor register).
Output (write)	Transfer data from specified source to I/O port or device.
Start I/O	Transfer instructions to I/O processor to initiate I/O operation.
Test I/O	Transfer status information from I/O system to specified destination.

Q.21 Explain any two transfer of control operations.

Ans. : Transfer control operations :

Operation	Description
Jump (branch)	Unconditional transfer; loads PC with specified address.
Jump conditional	Tests specified condition. If condition is true, loads PC with specified address; otherwise, do nothing.
Call to subroutine	Places current programs return address on stack and jump to specified address.
Return	Load return address from stack into PC.
Execute	Fetches operand from specified location and executes as instruction : do not modify PC.
Skip	Increments PC to skip next instruction.
Skip conditional	Tests specified condition. If condition is true, skip the next instruction; otherwise, do nothing.
Halt	Stops program execution.
Wait (hold)	Stops program execution; test specified condition repeatedly; resume execution when condition is satisfied.
No operation	No operation is performed, but program execution is continued.

11.6 : Addressing Modes

Q.22 Define addressing mode ? [SPPU : May-06, Marks 2]

Ans. : Part of the programming flexibility for each processor is the number and different kind of ways the programmer can refer to data stored in the memory or I/O device. The different ways that a processor can access data are referred to as **addressing schemes or addressing modes.**

Q.23 Define effective address.

Ans. : An address computed by the processor when executing a memory access or branch instruction or when fetching the next sequential instruction is known as **Effective Address (EA).** An effective address can be made up from as many as three elements : The **base, index** and **displacement.**

Q.24 List different addressing modes and explain any two with suitable diagrams and examples.

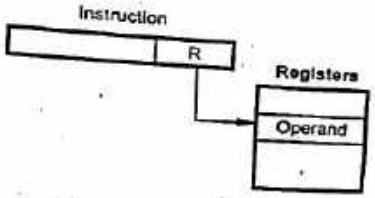
☞ [SPPU : May-06,08,17,19, Dec.-05,16,17,19, Marks 6]

Ans. : Types of addressing modes are :

1. Register addressing mode.
2. Absolute or direct addressing mode.
3. Immediate addressing mode.
4. Indirect addressing mode.
5. Register indirect addressing mode.
6. Displacement addressing mode.
7. Relative addressing mode.
8. Base register addressing.
9. Index addressing mode.
10. Auto-increment addressing mode.
11. Auto-decrement addressing mode.
12. Stack addressing mode.

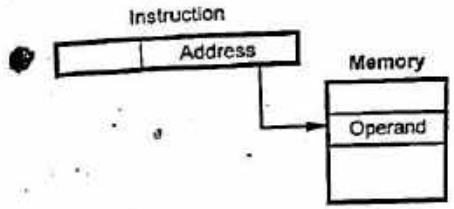
1. Register addressing mode : The operand is the contents of processor register. The name of register is specified in the instruction.

- **Example :** MOV R1, R2 : This instruction copies the contents of register R2 to register R1.



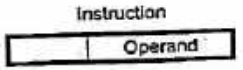
2. Absolute or direct addressing mode : The address of the location of the operand is given explicitly as a part of the instruction.

- **Example :** MOV A, 2000 : This instruction copies the contents of memory location 2000 into the A register. As shown in the instruction, here, address of operand is given explicitly in the instruction.

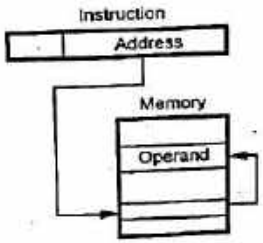


3. Immediate addressing mode : The operand is given explicitly in the instruction.

- **Example :** MOV A, #20 : This instruction copies operand 20 in the register A. The sign # in front of the value of an operand is used to indicate that this value is an immediate operand.

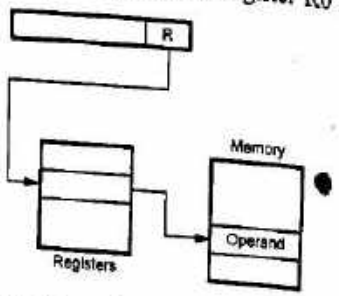


4. Indirect addressing mode : In this addressing mode, the instruction contains the address of memory which refers the address of the operand.



5. Register indirect addressing mode : The effective address of the operand is the contents of a register or the main memory location whose address is given explicitly in the instruction.

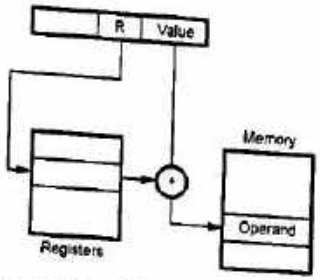
- **Example** : MOV A, (R0) : This instruction copies the contents of memory addressed by the contents of register R0 into the register A.



6. **Displacement addressing mode** : This addressing mode combines the capabilities of direct addressing and register indirect addressing. In this addressing mode, instruction has two address fields : Value and Referenced **register**. The effective address is computed by adding contents of referenced register to value.

EA = Value + (R). Three common variation of displacement addressing are :

- Relative addressing
- Base register addressing
- Index addressing.



7. **Relative addressing mode** : Here, the referenced register is Program Counter (PC) and hence this addressing mode is also known as **PC-relative addressing.**

The effective address is determined by adding the contents of PC to the address field. EA = PC + Address part of instruction.

The address part is a signed number so that it is possible to have branch target location either before or after the branch instruction. This addressing mode is commonly used to specify the target address in branch instructions.

- **Example** : JNZ BACK : This instruction causes program execution to go to the branch target location identified by the name BACK, if the branch condition is satisfied.

8. **Base register addressing** : In this addressing mode, the referenced register contains the main memory address and address field contains the displacement. Displacement is usually unsigned integer number. EA = (R) + Displacement.

- **Example** : MOV A, [R+8] : This instruction copies the contents of memory whose address is determined by adding the contents of register R and displacement 8 to the register A.

9. **Index addressing mode** : In this addressing mode, the address field references the main memory and the referenced register contains a positive displacement from that address.
EA = Memory address + (R).

The indexing is a technique that allows programmer to point or refer the data (operand) stored in sequential memory locations one by one. It is an efficient mechanism for performing iterative operations.

- **Example** : MOV R, [R1 + RI] : In this instruction main memory address is given by register R1 and the referenced register RI gives the positive displacement. The contents of the memory address generated by the addition of main memory address and displacement is copied to register R.

10. **Autoincrement addressing mode** : The effective address of the operand is the contents of a register specified in the instruction. After accessing the operand, the contents of this register are incremented to address the next location.

- **Example** : MOV (R2) +, R0 : The above instruction copies the contents of register R0 into the memory location whose address is specified by the contents of register R2. After copy operation, the contents of register R2 are automatically incremented by 1.

11. **Autodecrement addressing mode** : The contents of a register specified in the instruction are decremented and then they are used as an effective address to access a memory location.

- **Example** : MOV R1, – (R0) : This instruction, initially decrements the contents of register R0 and then the decremented contents of register R0 are used to address the memory location. Finally, the contents from the addressed memory location are copied into the register R1.

12. **Stack addressing mode** : A stack is linear array of reserved memory locations. It is associated with a pointer called **Stack Pointer** (SP).

In stack addressing mode, stack pointer always contains the address of Top Of Stack (TOS) where the operand is to be stored or located. Thus, the address of the operand (source or destination) is the contents of stack pointer.

This addressing mode is the special case of register indirect addressing where referenced register is a stack pointer.

Usually, stack grows in the direction of descending addresses, (descending stack), starting from a high address and progressing to lower one. In this stack, SP is decremented before any items are appended (pushed) on stack and SP is incremented after any items popped from the stack.

- **Example** : PUSH R : This instruction decrements SP and copies the contents of register R on to the top of stack pointed by stack pointer.

END...

Unit 5

12 Processor Enchancements

12.1 : Key RISC and CISC Characteristics

Q.1 Write a note on CISC architecture.

OR Explain the key characteristics of CISC architecture.

Ans. : • CISC is an acronym for Complex Instruction Set computers or computing. It is based on the concept of using very large instruction set having simple as well as complex instructions and making instruction set more flexible to keep program length as small as possible.

- In CISC complex instructions may take multiple cycles for execution.
- CISC instructions vary in size, often specify a sequence of operations, and can require serial (slow) decoding algorithms.
- They tend to have few registers, and the registers may be special purpose, which restrict the way in which they can be used.
- In CISC, there are many instructions that support memory reference. For example, we can add memory contents to the registers.
- CISC instruction sets are designed to take advantage of microcode.
- To add the flexibility in the instruction set, they support more and complex addressing modes.
- Example of CISCs are : Intel X86, Motorola 68000 series, DEC VAX, etc.

Q.2 Write a short note on RISC architecture.

OR Explain the key architecture of RISC architecture.

Ans. : RISC refers to Reduced Instruction Set Computers or Computing. RISC microprocessors are very different from CISC microprocessors. RISC use concept of keeping the instruction set as simple as possible to

allow the microprocessor's program to be written using only simple instructions.

- In RISC processors, there is an **one instruction per machine cycle**.
- RISC machine **instructions are not complicated** and can execute about as fast as, microinstruction on CISC machines.
- The **machine instructions are hardwired**. These instructions are executed faster than the instructions implemented with microinstructions.
- This architecture encourages the optimization of register use, so that frequently accessed operands remain in high-speed storage to implement register to register operations. For this RISC processors provide **multiple sets of registers**.
- RISC processor provides **limited number of instructions**, which simplifies the design of control unit.
- RISC processor uses **simple addressing modes**. Almost all instructions use simple register addressing.
- RISC processors use **simple instruction formats with fixed instruction length**. The instruction length is aligned on word boundaries. Field locations, especially opcodes are fixed.

Because of this, they provide following benefits :

- With fixed fields, opcode decoding and register operands addressing can occur simultaneously.
- It simplifies the design of control unit.
- Instruction fetching is optimized since word-length units are fetched.
- To speed-up instruction execution, RISC uses instruction pipelining.
- Examples of RISCs are : ARM, ATMEL, 8051 family, AVR, MIPS, PIC etc.

Q.3 Differentiate between RISC and CISC processors.

[SPPU : May-19, Marks 6]

Ans. :

No.	Characteristics	CISC	RISC
1.	Instruction size	Varies	Fixed
2.	Instruction length	1, 2, 3 or 4 bytes	4 bytes
3.	Number of Instruction	More	Less
4.	Instruction decoding	Serial (slow) to decode	Easy (quick) to decode
5.	Instruction sementics	Varies from simple to complex ; possibly many dependent operations per instruction	Almost always one simple operation
6.	Addressing Modes	Supports complex addressing modes.	Complex addressing modes are synthesized in software.
7.	Instruction execution speed	Slow (depend on complexity of instruction)	Medium
8.	Instruction execution	By microprogram. Complex instructions taking multiple cycles.	By hardware. Simple instructions taking one cycle.
9.	Registers	Few, may be special purpose	Many, general purpose
10.	Memory references	Combined with operations in many different types of instructions	Not combined with operations, i.e., load/store architecture
11.	Hardware	Complicated	Simple

12.	Hardware design focus	Take the advantage of microcoded implementations	Take the advantage of implementations with one pipeline and no microcode
13.	Memory access	Frequently	Rarely
14.	Instruction format	Field placement varies	Regular, consistent placement of fields
15.	Pipelined	Not pipelined or less pipelined.	Highly pipelined.
16.	Conditional jump	Conditional jump is usually based on status register bit.	Can be based on a bit anywhere in memory.
17.	Compiler	Simple	Complicated
18.	Examples	Intel X86, Motorola 68000 series.	ARM, 8051, ATMEL, AVR, etc.

Table Q.3.1 Comparison between RISC and CISC processors characteristics

12.2 : Interrupt and its Purpose

Q.4 What is the purpose of interrupt ?

Ans. : Sometimes it is necessary to have the computer automatically execute one of a collection of special routines whenever certain conditions exists within a program or the computer system e.g. It is necessary that computer system should give response to devices such as keyboard, sensor and other components when they request for service. The purpose of interrupt is to provide above functionality .

Q.5 Define interrupt and interrupt service routine.

Ans. : The interrupt provides an external asynchronous input that would inform the processor that it should complete whatever instruction that is currently being executed and fetch a new routine (Interrupt Service Routine) that will service the requesting device. Once this servicing is completed, the processor would resume exactly where it left off. The

event that causes the interruption is called **interrupt** and the special routine executed to service the interrupt is called **Interrupt Service Routine (ISR).**

Q.6 State the difference between ISR and IRET.

Ans. : The interrupt service routine is different from subroutine because the address of ISR is predefined or it is available in **Interrupt Vector Table (IVT)**, whereas subroutine address is necessarily to be given in subroutine CALL instruction. IRET instruction is used to return from the ISR whereas RET instruction is used to return from subroutine. IRET instruction restores flag contents along with CS and IP in the IA-32 architecture; however RET instruction only restores CS and IP contents.

12.3 : Types of Interrupts

Q.7 What are hardware and software interrupts ?

Ans. : • There are two main classes of interrupts :

- Hardware interrupts
- Software interrupts

• An interrupt caused by an external signal is referred as **hardware interrupt.**

• Conditional interrupts or interrupts caused by special instructions are called **software interrupts.**

Q.8 What are vector interrupts ?

OR What is vectoring ?

Ans. : • If the internal control circuit of the processor produces a CALL to a predetermined memory location which is the starting address of interrupt service routine, then that address is called vector address and such interrupts are called **vector interrupts.**

• For vector interrupts fastest and most flexible response is obtained since such an interrupt causes a direct hardware-implemented transition to the correct interrupt-handling program. This technique is called **vectoring.** When processor is interrupted, it reads the vector address and loads it into the PC.

Q.9 How does processor respond to non-vector interrupt ?

Ans. : In case of **non-vectored interrupt**, when the processor receives the interrupt signal from the external device, the processor completes the execution of the current instruction and sends a signal called INTA (interrupt acknowledge). After receiving the INTA signal, external hardware sends the interrupt vector to the processor. After receiving interrupt vector address, the processor loads it into the PC.

Q.10 Define maskable and non-maskable interrupts.

Ans. : • Most of the processors provide the masking facility. In the processor those interrupts which can be masked under software control are called **maskable interrupts**. The interrupts which can not be masked under software control are called **non-maskable interrupts.**

- Maskable interrupts are enabled and disabled under program control. By setting or resetting particular flip-flops in the processor, interrupts can be masked or unmasked, respectively.
- When masked, processor does not respond to the interrupt even though the interrupt is activated.

Q.11 Define nested interrupts.

Ans. : For some devices, a long delay in responding to an interrupt request may cause error in the operation of computer. Such interrupts are acknowledged and serviced even though processor is executing an interrupt service routine for another device. A system of interrupts that allows an interrupt service routine to be interrupted is known as **nested interrupts.**

12.4 : Interrupt Handling

Q.12 What do you mean by single level interrupts and multi level interrupts ?

Ans. : • Interrupts can be classified according to how the multiple interrupts are handled by the system.

- Single level interrupts
- Multi level interrupts

Single Level Interrupts

- In a single level interrupts there can be many interrupting devices. But all interrupt requests are made via a single input pin of the CPU.
- When interrupted, CPU has to poll the I/O ports to identify the request device. Polling software routine that checks the logic state of each device. Once the interrupting I/O port is identified, the CPU will service it and then return to task it was performing before the interrupt.
- Fig. Q.12.1 shows single level interrupt system, in which the interrupt requests from all the devices are logically ORed and connected to the interrupt input of the processor. Thus, the interrupt request from any device is routed to the processor interrupt input.

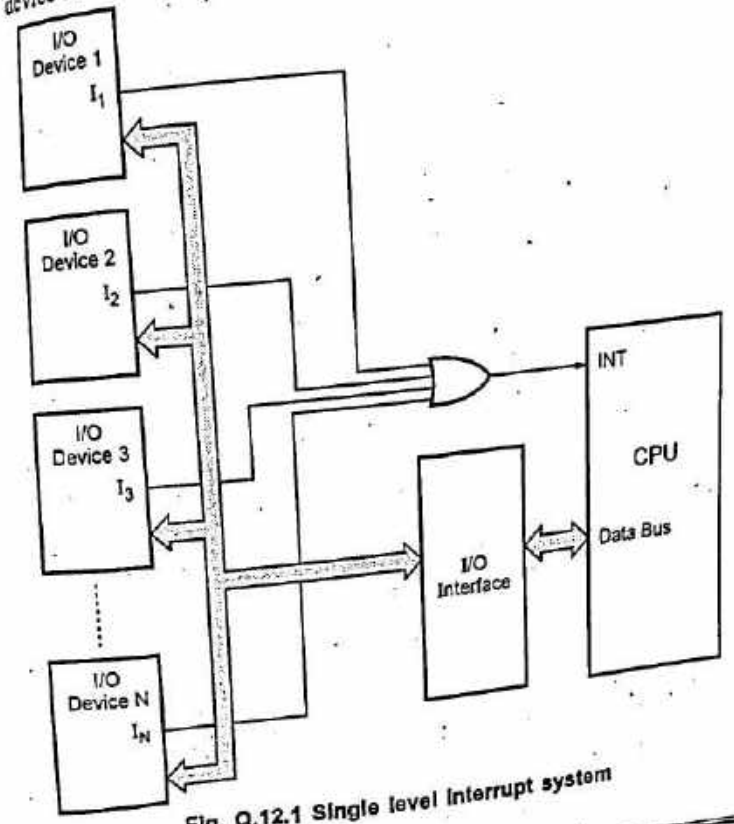


Fig. Q.12.1 Single level interrupt system

- After getting interrupted, processor identifies requesting device by reading interrupt status of each device.

Multi Level Interrupts

- In multi level interrupts, processor has more than one interrupt pins. The I/O devices are tied to the individual interrupt pins. Thus, the interrupts can be immediately identified by the CPU upon receiving an interrupt request from it. This allows processor to go directly to that I/O device and service it without having to poll first. This obviously saves the time in processing interrupt.
- Fig. Q.12.2 shows the multi level interrupt system.

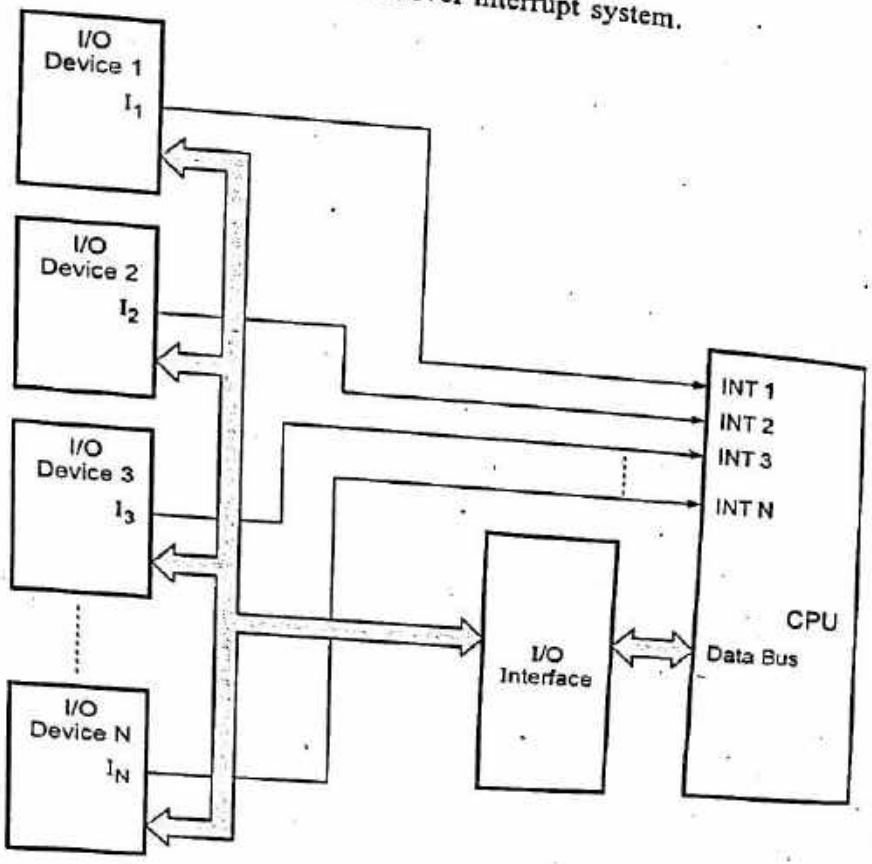


Fig. Q.12.2 Multilevel interrupt system

- In multi level interrupt system, when a processor is interrupted, it stops executing its current program and calls a special routine which "services" the interrupt.

Q.13 How does CPU recognize an interrupt ? What is the response of the CPU after recognition of the interrupt ?

[SPPU : May-05, Marks 6]

Ans. : • When the external asynchronous input (interrupt input) is asserted (a signal is sent to the interrupt input), a special sequence in the control logic begins.

1. The processor completes its current instruction. No instruction is cut-off in the middle of its execution.
2. The program counter's current contents are stored on the stack. Remember, during the execution of an instruction the program counter is pointing to the memory location for the next instruction.
3. The program counter is loaded with the address of an interrupt service routine.
4. Program execution continues with the instruction taken from the memory location pointed by the new program counter contents.
5. The interrupt program continues to execute until a return instruction is executed.
6. After execution of the RET instruction processor gets the old address (the address of the next instruction from where the interrupt service routine was called.) of the program counter form the stack and puts it back into the program counter. This allows the interrupted program to continue executing at the instruction following the one where it was interrupted.

- Fig. Q.13.1 shows the response to an interrupt with the flowchart and diagram. (Refer Fig. Q.13.1 on next page)

Q.14 What is the use of interrupt priority ?

Ans. : • When interrupt requests arrive from two or more devices simultaneously, the processor has to decide which request should be serviced first and which one should be delayed. The processor takes the decision with the help of interrupt priorities.

- It accepts the request having the highest priority.

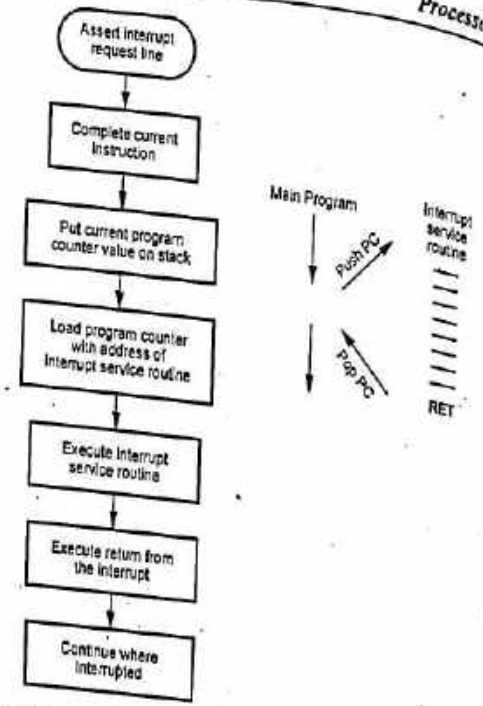


Fig. Q.13.1 Response to an Interrupt with the flowchart and diagram

Q.15 What is pending interrupt ?

Ans. : The interrupt which took place but cannot be executed immediately (for instance, if another higher-priority interrupt is running), is called **pending interrupt.**

12.5 : Exceptions

Q.16 Write a note on exceptions.

Ans. : • An interrupt is an event that suspends the processing of currently executing program and begins the execution of another program. So far we have seen the interruption is activated by an I/O device in the form of a hardware signal; however, many events can cause an interrupts. All such events are called **exceptions.**

- Therefore, an I/O interrupt is a subtype of exception. Apart from I/O interrupts the exceptions can be classified as **faults, traps, or aborts**

depending on the way they are reported, and whether or not restart of the instruction causing the exception is supported.

- **Faults :** Faults are the exceptions that are detected and serviced before the execution of the faulting instruction. For example, in virtual memory system if page or segment referenced by processor is not present, the operating system fetches the page or segment from disk using fault exception routine, and then processor restarts processing using referenced page or segment.
- **Traps :** Traps are exceptions that are reported immediately after the execution of the instructions which causes the problem. User defined interrupts are the examples of traps.
- **Aborts :** Aborts are exceptions which do not permit the precise location of the instruction causing the exception to be determined. Aborts are used to report severe errors, such as hardware error, or illegal values in the system tables.

12.6 : Instruction Pipelining

Q.17 What is instruction pipeline ? Explain how it affects speed of execution ? ☞ [SPPU : May-13, Dec-15, Marks 7]

OR What are the advantages of pipelining ? ☞ [SPPU : May-16, Marks 3]

Ans. : Various cycles involved in the instruction cycle. These fetch, decode and execute cycles for several instructions are performed simultaneously to reduce overall processing time. This process is referred to as **instruction pipelining.**

Let us assume that instruction execution is divided into following five stages :

- S_1 - **Fetch Instruction (FI) :** Read the next instruction into an instruction buffer
- S_2 - **Decode Instruction (DI) :** Decode the opcode
- S_3 - **Fetch Operands (FO) :** Fetch each memory referenced operand.
- S_4 - **Execute Instruction (EI) :** Perform the indicated operation.

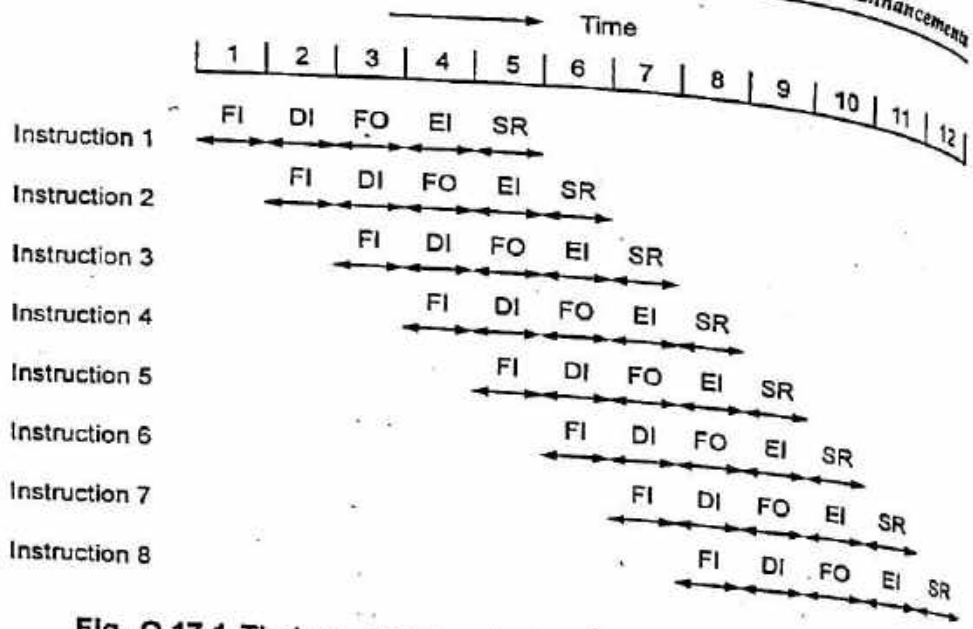


Fig. Q.17.1 Timing diagram for instruction pipeline operation

- **S_5 - Store Results (SR) :** Store the result in memory.
- With this subdivision and assuming equal duration for each stage we can reduce the execution time for 8 instructions from 40 time units to 12 time units. This is illustrated in Fig. Q.17.1.

Q.18 Write a note on pipeline hazards.

Ans. : 1. Structural hazards : These hazards are because of conflicts due to insufficient resources when even with all possible combination, it may not be possible to overlap the operation.

2. **Data or data dependent hazards :** These result when instruction in the pipeline depends on the result of previous instructions which are still in pipeline and not completed.

3. **Instruction or control hazards :** They arise while pipelining branch and other instructions that change the contents of program counter. The simplest way to handle these hazards is to stall the pipeline. Stalling of the pipeline allows few instructions to proceed to completion while stopping the execution of those which results in hazards.

Q.19 Explain with example working of multistage pipelining ?

☞ [SPPU : May-16, Marks 3]

Ans. : The Fig. Q.19.1 shows the implementation of four-stage instruction pipelining. As shown in the Fig. Q.19.1 the CPU is directly connected to a cache memory, which is split into instruction and data parts, called the **I-cache** and **D-cache**, respectively. This splitting of the cache permits both an instruction word and a memory data word to be accessed in the same clock cycle.

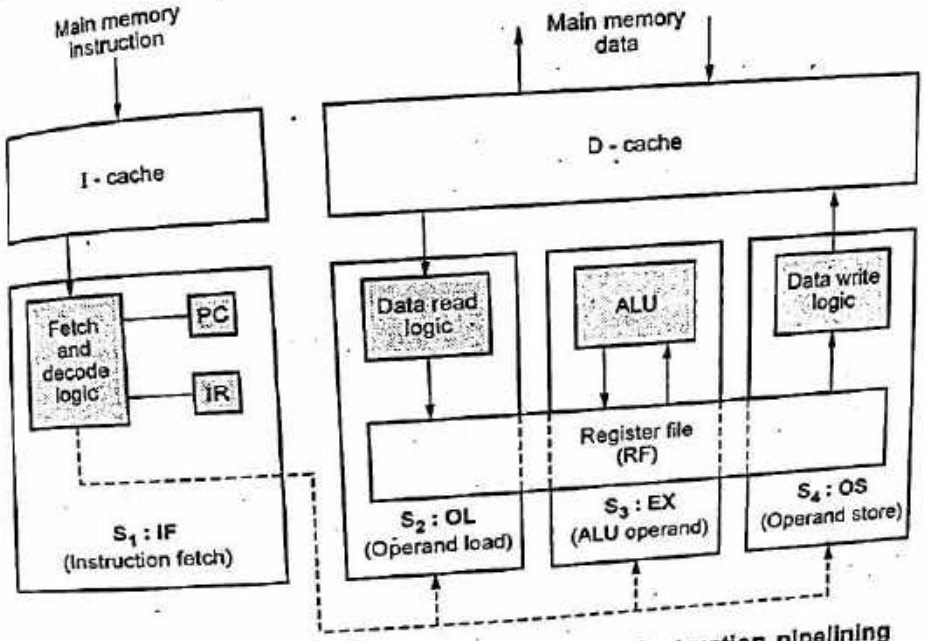


Fig. Q.19.1 Organization of CPU for four-stage instruction pipelining

- The four stages S_1 : S_4 shown in Fig. Q.19.1 perform the following functions :
  - **S_1 : IF :** Instruction fetching and decoding using the I cache.
  - **S_2 : OL :** Operand loading from the D-cache to register file.
  - **S_3 : EX :** Data processing using the ALU and register file.
  - **S_4 : OS :** Operand storing to the D-cache from register file.

- Stages S_2 and S_4 implements memory load and store operations, respectively.
- Stages S_2, S_3 and S_4 share the CPU's local Register File (RF). The registers in the register file act as interstage buffer registers.
- The stage 3 implements data transfer and data processing operations of the register to register type using ALU of the CPU.

Q.20 Write a note on branch prediction techniques.

Ans. : • Prediction techniques can be used to check whether a branch will be valid or not valid. These techniques reduce the **branch penalty.**

- The common prediction techniques are :
  - Predict Never Taken
  - Predict Always Taken
  - Predict By Opcode
  - Taken/Not Taken Switch
  - Branch History Table
- In the first two approaches if prediction is wrong a page fault or protection violation error occurs. The processor then halts prefetching and fetches the instruction from the desired address.
- In the third prediction technique, the prediction decision is based on the opcode of the branch instruction. The processor assumes that the branch will be taken from certain branch opcodes and not for others.
- The fourth and fifth prediction techniques are dynamic; they depend on the execution history of the previously executed conditional branch instruction.

Q.21 Explain various performance measures and instruction pipeline.

Ans. : • A pipeline's performance can be measured by its throughput in terms of millions of instructions executed per second or **MIPS.**

- Another popular measure of performance is the number of clock cycles per instruction or **CPI.** These quantities are related by the equation.

$$CPI = f/MIPS$$

where f is the pipeline's clock frequency in MHz, and the values of CPI and MIPS are average figures that can be determined experimentally by processing number of representative programs.

- Another general measure of pipeline performance is the speedup S(m) defined by

$$S(m) = \frac{T(1)}{T(m)}$$

where T(m) is the execution time for some target program on an m-stage pipeline and T(1) is the execution time for the same program on a similar, nonpipelined processor.

Q.22 Define stage delay and interstage delay.

Ans. : Each pipeline stage is a combinational logic circuit. It requires a specific amount of time in terms of propagation delay to process the input data. The precessing time in each stage is known as **stage delay.** It is denoted as τ. The delays are also introduced due to interstage transfer data. These time delays are known as **interstage delay** and it is denoted as d.

Q.23 How does the time periods for the clock cycle determined ?

Ans. : To determine the time period for the clock cycle it is necessary to consider stage delay and interstage delay. The interstage delays can be same between the stages. However, stage delays may not be same for different stages. In such cases, maximum stage delay, denoted as τ_m is considered to determine the time period for the clock cycle. The time period for clock cycle, τ can be given as

$$\tau = \max_i \{\tau_i\}_1^m + d = \tau_m + d$$

Usually, $\tau_m >> d$ and maximum stage delay dominates the clock period. Therefore, **pipeline frequency** which is inverse of clock period can be given as

$$f = \frac{1}{\tau}$$

Q.24 What is clock skewing ?

Ans. : Ideally, we expect the clock pulse to arrive at all stage latches at the same time. In practice, the same clock pulse may arrive at different

stages with a time offset of s. This problem is known as clock skewing. The offset s is large for longer logic paths within the stages.

Q.25 Draw and explain the space-time diagram of four stage pipeline.

Ans. : Fig. Q.25.1 shows the space-time diagram of a four stage pipeline processor. As shown in the Fig. Q.25.1, once the pipe is filled up, it outputs one result per clock period independent of the number of stages in the pipe. Ideally, a linear pipeline with m stages can process n tasks in $T_m = m + (n - 1)$ clock cycles, where m cycles are needed to complete the execution of the first task and the remaining $n - 1$ tasks require $n - 1$ cycles. Thus the total time required is

$$T_m = [m + (n - 1)]\tau$$

where τ is the clock period.

- The space-time diagram shown in Fig. Q.25.1 has four stages and five tasks. Therefore, the ideal total time required is

$$T_m = [4 + (5 - 1)]\,\tau$$

$$= 8 \text{ clock cycles}$$

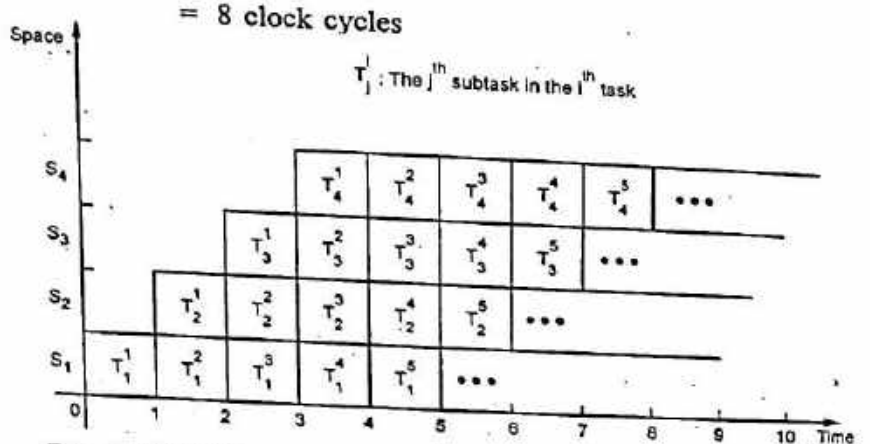


Fig. Q.25.1 The space-time diagram showing the overlapped operations

Q.26 What is speedup factor ? Explain with the help of example.

Ans. : The speedup factor of a m-stage pipeline over an equivalent non-pipelined processor is defined as

$$S_m = \frac{T_1}{T_m} = \frac{n\,m\tau}{[m+(n-1)]\tau} = \frac{n\,m}{m+(n-1)}$$

- For example, if pipeline has four stages and five tasks, its speedup factor is

$$S_4 = \frac{4\times 5}{4+(5-1)} = \frac{20}{8}$$

$$= 2.5$$

Q.27 Define efficiency of linear pipeline.

Ans. : The efficiency of a linear pipeline is defined as a ratio of speedup factor and the number of stages in the pipeline. The efficiency of m-stage pipeline is given as

$$E_m = \frac{S_m}{m} = \frac{nm}{m+(n-1)} + m$$

$$= \frac{n}{m+(n-1)}$$

- For example, if speedup factor is 2.5 and number of stages are four, the efficiency can be given as

$$E_m = \frac{2.5}{4} = 0.625$$

- It can be noticed that the efficiency approaches to unity when $n \to \infty$. When $n = 1$, efficiency is minimum. It is $1/m$.

Q.28 Define throughput of the pipeline.

Ans. : The pipeline throughput H_m is defined as the number of results (tasks) that can be completed by a pipeline per unit time. It is given as

$$H_m = \frac{n}{[m+(n-1)]\tau} \qquad \because E_m = \frac{n}{[m+(n-1)]}$$

$$= \frac{E_m}{\tau} \qquad \because f = 1/\tau$$

$$= E_m f$$

- In the ideal case, $H_m = 1/\tau = f$ when $E_m \to 1$. This means that the maximum throughput of a linear pipeline is equal to its frequency, which corresponds to one output result per clock period. The overall throughput of pipeline is always less than f. This is because usually $E_m < 1$.

Q.29 Define performance / cost ratio.

Ans. : The maximum throughput for m-stage pipeline can be given as

$$f = \frac{1}{\tau} = \frac{1}{(t/m+d)}$$

- The total pipeline cost is roughly estimated by $c + mh$, where c represents cost of all logic stages and h represents the cost of each latch. With this information, in 1973, Larson has defined a pipeline **performance/cost ratio** as

$$PCR = \frac{f}{c+mh} = \frac{1}{(t/m+d)(c+mh)}$$

12.7 : Multiprocessor Systems and Multicore Processor

Q.30 What is parallel processing ?

Ans. : • To fulfil increasing demands for higher performance it is necessary to process data concurrently to achieve better throughput instead of processing each instruction sequentially as in a conventional computer. Processing data concurrently is known as **parallel processing.**

Q.31 State the basic ways to achieve parallelism.

Ans. : There are two basic ways by which we can achieve parallelism. These are :

- **Multiple Functional Units :** System may have two or more ALUs so that they can execute two or more instructions at the same time.
- **Multiple Processors :** System may have two or more processors operating concurrently.

Q.32 Define multiprocessor system, task-level parallelism, parallel processing program and cluster.

Ans. : • **Multiprocessor system :** A computer system with at least two processors is called multiprocessor system.

- **Task-level parallelism or process-level parallelism :** Utilizing multiple processors for executing independent programs simultaneously is known as Task-level parallelism or process-level parallelism.

- **Parallel processing program :** It is referred to a single program that runs on multiple processors simultaneously.

Cluster : A set of computers connected over a local area network that function as a single large multiprocessor is called cluster.

Q.33 Why Multicore ?

Ans. : • A multicore is an architecture design that places multiple processors on a single die (computer chip) to enhance performance and allow simultaneous processing of multiple tasks more efficiently. Each processor is called a core. The multicore architecture designs are known as **Chip Multiprocessors (CMPs)** because they allow single chip multiprocessing.

Q.34 Explain the limitations to increase clock frequency or processor speed to increase performance.

Ans. : • One way to increase system performance is to increase clock frequency or processor speed. However, this approach has following limitations in terms of cost effectiveness :

- Higher frequency requires more power.
- More power consumption results it harder and more expensive to cool the system.
- More power consumption also affects sizing and packaging considerations.

• Thus, instead of trying to make the processor faster to gain performance, adding more processors on a single chip (multicore architecture) is a better approach.

12.8 : Flynn's Taxonomy of Parallel Processor Architectures

Q.35 Write about Flynn's taxonomy for multiple processor organizations. [SPPU : Dec.-16, Marks 6]

OR What is Flynn's taxonomy for multiple processor organizations ? Explain with diagram. [SPPU : June-17, Marks 6]

Ans. : • The classification made by Micheal J. Flynn divides computers into four major groups.

- Single Instruction Stream-Single Data stream (SISD).
- Single Instruction Stream-Multiple Data streams (SIMD).
- Multiple Instruction Streams-Single Data stream (MISD).
- Multiple Instruction Streams-Multiple Data streams (MIMD).

1. **Single Instruction stream Single Data Stream (SISD) :** Most conventional machines with one CPU containing a single arithmetic-logic unit capable only of scalar arithmetic fall into this category. SISD computers and sequential computers are thus synonymous. In SISD computers instructions are executed sequentially but may overlap in their execution stages (Pipelining). They may have more than one functional unit, but all functional units are control by a single control unit.

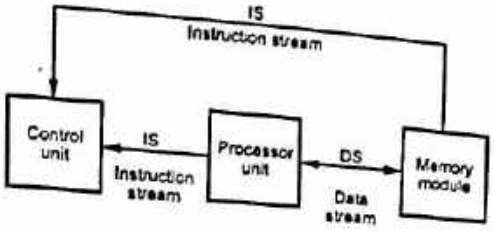


Fig. Q.35.1 SISD computer

2. **Single Instruction stream Multiple Data streams (SIMD) :** This category corresponds to array processors. They have multiple processing/execution units and one control unit. Therefore, all processing/execution units are supervised by the single control unit. Here, all processing elements received same instruction from control unit but operate on different data sets from distinct data streams. This category is also known as single program, multiple data stream (SPMD).

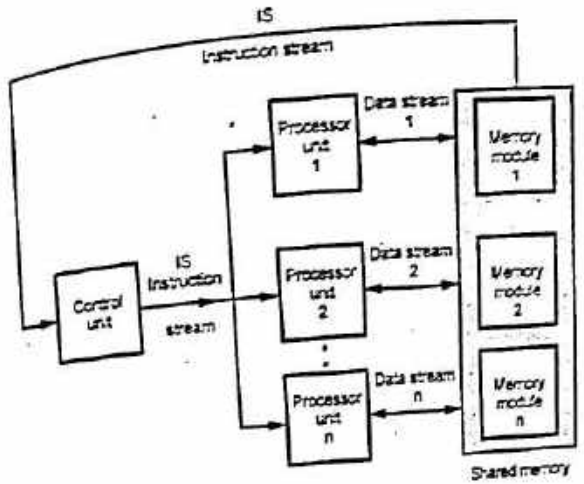


Fig. Q.35.2 SIMD computer

3. **Multiple Instruction streams Single Data stream (MISD) :** Not many parallel processors fit well into this category. In MISD, there are n processor units. Each receiving distinct instructions operating over the same data stream and its derivatives. The results of one processor

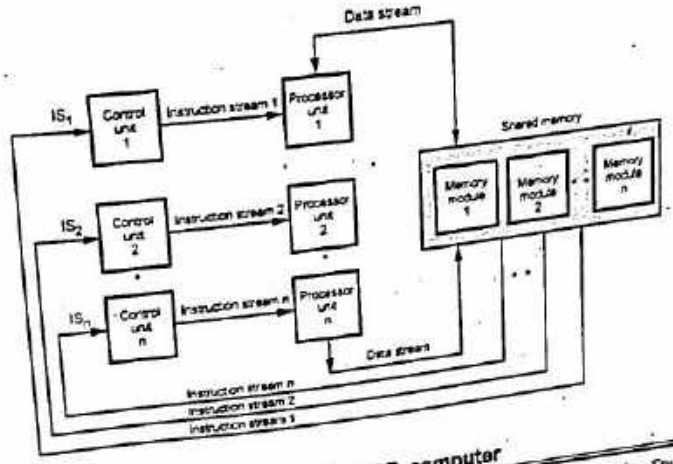


Fig. Q.35.3 MISD computer

become the input of the next processor in the micropipe. The fault-tolerant computers where several processing units process the same data using different programs belongs to the MISD class. The results of such apparently redundant computations can be compared and used to detect and eliminate faulty results.

4. **Multiple Instruction streams Multiple Data streams (MIMD) :** Most multiprocessors system and multiple computers system can be classified in this category. In MIMD, there are more than one processor unit having the ability to execute several program simultaneously. MIMD computer implies interactions among the multiple processors because all memory streams are derived from the same data space shared by all processors. If the n data streams are derived from disjointed sub spaces of the shared memories then we would have the so-called Multiple SISD (MSISD) operation.
5. **Vector systems :** A more efficient interpretation of SIMD is called a vector architecture. Rather than having 64 ALUs perform 64 additions simultaneously, like the old array processors, the vector architectures pipelined the ALU to get good performance at lower cost.

- Vector architecture consists of a set of vector registers.
- Vector architectures collect data elements from memory, put them in order into a large set of registers, operate on them sequentially in registers and then write the results back to memory.

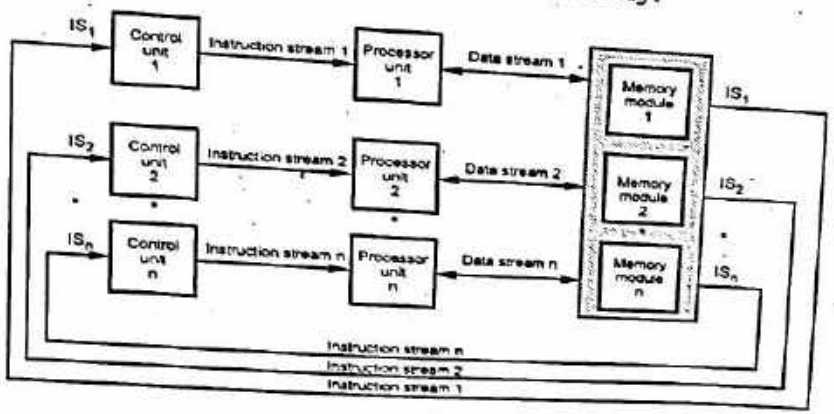


Fig. Q.35.4 MIMD computer

12.9 : Closely Coupled and Loosely Coupled Multiprocessor Systems

Q.36 What are the advantages of the multiprocessor systems ?

OR Define multiprocessing system.

OR State the types of multiprocessor system.

Ans. : Definition : If a microprocessor system contains two or more components that can execute instructions independently, then the system is called **multiprocessor system**. Multiprocessor system uses a distributed approach. Here, more than one processors are used to do the subtasks instead of doing entire task by a single processor.

Advantages : This system has following advantages :

1. Improves cost/performance ratio of the system.
2. Several processors may be combined to fit the needs of an application while avoiding the expense of the unneeded capabilities of a centralized system. Yet this system provides room for expansion.
3. Tasks are divided among the modules. If failure occurs, it is easier and cheaper to find and replace the malfunctioning processor than replacing the failing part of complex processor.

Types : The multiprocessor systems are implemented using one of the two basic architectures : **Loosely coupled architecture** and **closely coupled architecture**. The systems using these architectures are known as loosely coupled systems and closely coupled systems respectively.

Q.37 Explain closely coupled and loosely coupled microprocessor system. [SPPU : June-17, Dec.-17,18, May-19, Marks 7]

Ans. : Closely Coupled Multiprocessor System : • In the Closely Coupled System (CCS) the processors or supporting processors (coprocessor, maths processor) share clock generator, bus control logic, entire memory and I/O subsystem. Such systems communicate through a shared main memory. Hence the rate at which data can communicate from one processor to the other is on the order of the bandwidth of the memory. One of the limiting factors to the expansion of a CCS is the performance degradation due to memory contentions which occur when two or more processors attempt to access the same memory unit

concurrently. However, when high-speed or real-time processing is desired, Closely Coupled Systems (CCS) may be used.

Loosely Coupled Multiprocessor System : • In loosely coupled systems, each processor has a set of input-output devices and a large local memory where it accesses most of the instructions and data. The processor, its local memory and input-output interfaces are together called **computer module**. Processes which execute on different computer modules communicate by exchanging messages through a **Message Transfer System (MTS)**. The coupling in such a system is very loose. Hence, such systems are also referred to as **distributed systems**. These systems are usually efficient when the interactions between tasks are minimal.

Q.38 State the advantages of loosely coupled system.

Ans. :

1. Better system throughput by having more than one processor.
2. Each processor may have a local bus to access local memory or I/O devices so that a greater degree of parallel processing can be achieved.
3. System structure is more flexible. As the system consists of different modules, one can easily add or remove modules to change the system configuration ; without affecting the other modules in the system.
4. A failure in one module normally does not cause a breakdown of the entire system. The faulty module can be detected and replaced.

Q.39 Give the comparison between closely coupled and loosely coupled systems.

Ans. :

Sr. No.	Closely Coupled System	Loosely Coupled System
1.	A multiprocessor system with common shared memory is known as closely coupled system.	A multiprocessor system in which each processor has its own private local memory is known as loosely coupled system.

2.	Here, the information can be shared among the CPUs by placing it in the common global memory.	Here, the information is transferred from one processor to other by message-passing system.
3.	Parallelism can be implemented less efficiently.	Parallelism can be implemented more efficiently.
4.	System structure is less flexible.	System structure is more flexible.

12.10 : Symmetric Multiprocessors (SMP)

Q.40 Explain Symmetric Multiprocessor (SMP) Organization with features. [SPPU : Dec.-16, Marks 7]

Ans. : • In computing, **symmetric multiprocessing** or **SMP** involves a multiprocessor computer-architecture where two or more identical processors can connect to a single shared main memory. Most common multiprocessor systems today use an SMP architecture. In the case of multi-core processors, the SMP architecture applies to the cores, treating them as separate processors.

Features / Characteristics of SMP :

1. Two or more similar processors are employed in a stand-alone system.
2. The processors share the same main memory and I/O facilities.
3. The processors access memory and I/O in approximately equal amount of time.
4. All processors share access to I/O devices, either through the same channels or through different channels that provide paths to the same device.
5. All processors are capable of performing the same functions.
6. The operating system controls the interaction between processors and their programs at different levels.
7. The distribution of workload between processors is performed by the operating system in such a way that multiple independent or dependent

tasks are shared between processors without any special consideration in the application program.

- Fig. Q.40.1 shows a block diagram of a typical SMP machine. There are two or more processors. Each processor unit consists of ALU, registers, control unit and typically one or more levels of cache. Each processor can access a shared main memory and the I/O devices through some form of interconnection mechanism. Many times, memory is organized to provide simultaneous accesses to separate blocks of memory. Apart from shared memory, each processor may have its own private main memory and I/O channels.

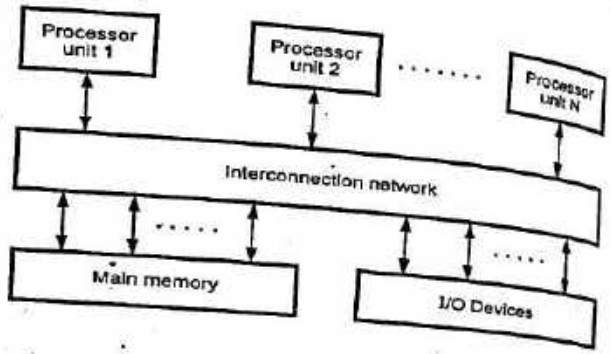


Fig. Q.40.1 Block diagram of typical SMP machine

Q.41 State the advantages and disadvantages of SMP.

Ans. : Advantages of SMP :

1. **Performance** : In situations where more than one program executes at the same time, an SMP system will have considerably better performance than a uni-processor because different programs can run on different processors simultaneously.
2. **Availability** : In a symmetric multiprocessor, all processors are capable of performing same functions. Thus, the failure of a single processor does not halt the machine; the task of failure processor is done by other processors in the system. But this action reduces the performance of the system.

3. **Incremental growth (scaling)** : The machine is scalable. This means that we can add additional processor to increase the performance or to maintain the performance level in case of increased data processing.

Disadvantages of SMP :

1. System programmers must build support for SMP into the operating system; otherwise, the additional processor remain idle and the system functions as a uniprocessor system.
2. In cases where an SMP environment processes many jobs, administrators often experience a loss of hardware efficiency. Software programs should be developed to schedule jobs so that the processor utilization reaches its maximum potential.

Q.42 List various interconnections mechanisms used in the SMP.

Ans. : According to interconnection mechanism used in the SMP, they are classified as

- Time-shared or common bus
- Multi-part memory
- Central control unit.

Q.43 Write a note on time shared bus.

Ans. : • Fig. Q.43.1 shows a block diagram of typical SMP machine with a common time shared bus. The common time-shared bus provides the simplest and most cost-effective way to interconnect the processors together. It also, however, limits the number of processors that can be used. Since all memory and I/O references pass through the shared bus, the performance of the system is limited by the maximum band-width of this bus. One way to reduce the shared bus bottle-neck is the extensive use of cache memory. Most SMP systems have at least two levels of cache memory as shown in Fig. Q.43.1. Typically level 1 cache is on the same chip as the processor and level 2 cache may or may not be internal to the processor.

- Using cache memory in a multiprocessor system introduces the problem of cache coherency : how to guarantee that when a data in a specific cache are altered, this change is reflected in other caches and in main memory.

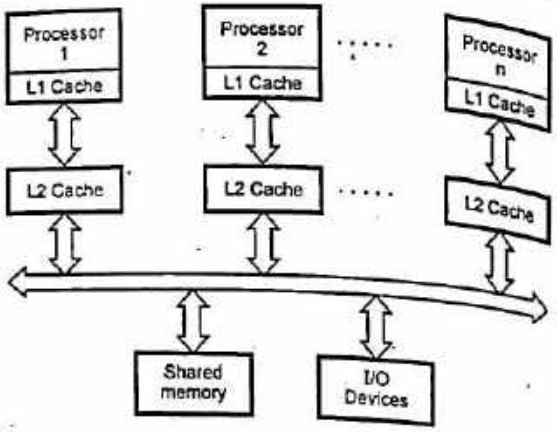


Fig. Q.43.1 SMP machine with a common time shared bus

Q.44 Explain the crossbar switch system organisation for multiprocessors.

Ans. : Fig. Q.44.1 shows the crossbar switch system organisation for multiprocessors which provides separate path for each memory module. The interconnection network shown in Fig. Q.44.1 is called nonblocking **crossbar.**

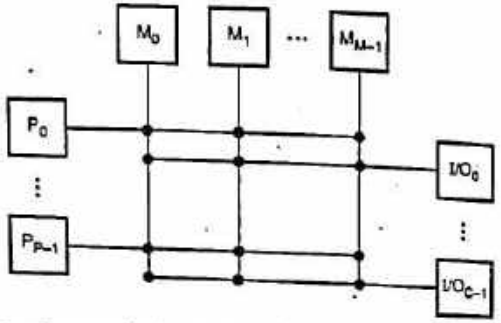


Fig. Q.44.1 Crossbar switch system organisation for multiprocessor

- The crossbar switch has a separate bus associated with each memory module. Therefore, the maximum number of transfers that can take place simultaneously is limited by the number of memory modules and the bandwidth-speed product of the buses rather than the number of paths available.

- Important characteristics of a system utilizing a crossbar matrix are :
  1. Extreme simplicity of the switch-to-functional unit interfaces.
  2. Ability to support simultaneous transfers for all memory units.
- To provide these features, it requires hardware capabilities which are capable of switching parallel transmissions and capable of resolving multiple requests for access to the same memory module occurring during a single memory cycle.

Q.45 Write a short note on multiport memory.

Ans. : • The multiport memory approach allows the direct, independent access of main memory modules by each processor and I/O module as shown in Fig. Q.45.1.

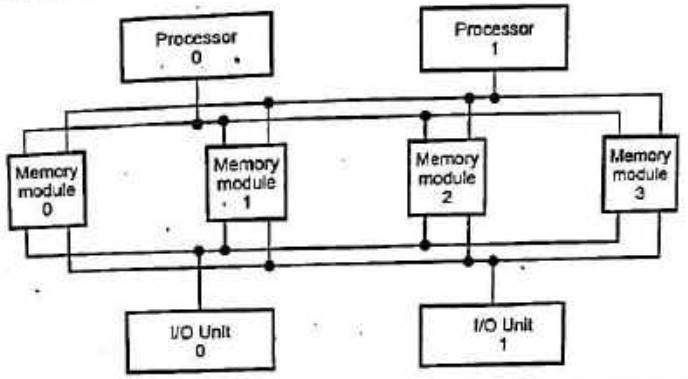


Fig. Q.45.1 Multiport memory without fixed priority assignment

- Here, memory-access conflicts are resolved by assigning permanently designated priorities at each memory port.
- In these organisations, it is possible to designate portions of memory as private to certain processors, input/output units or combinations thereof.
- In Fig. Q.45.2 memory modules 0 and 3 are private to processors 0 and 1 respectively. Such organisation has an advantage that due this there is increase in protection against unauthorized access and it also permit the storage of recovery routines in memory areas that are not susceptible to modifications by other processors.

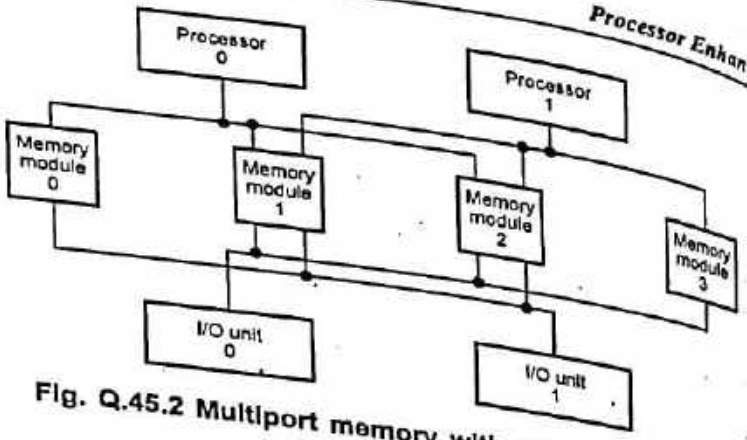


Fig. Q.45.2 Multiport memory with private memories

- However, the main disadvantage of such system is the system recovery if the other processors are not able to access control and status information in a memory block associated with a faulty processor. This organization can also support non blocking access to memory if a full-connected topology is used.

Q.46 Give comparison between time shared bus, crossbar switch and multiport memory organizations.

Ans. :

Sr. No.	Time shared bus	Crossbar switch	Multiport memory
1.	Lowest overall system cost for hardware and Least complex	Most complex. Highest total transfer rate.	Most expensive since most of control and switching circuitry is in the memory unit.
2.	Easy to modify hardware system configuration.	Functional units are simplest and cheapest.	Functional units permit low cost uniprocessor.

3.	Overall system capacity limited by bus transfer rate. Bus fail then whole system fail.	Usually cost-effective for multiprocessors only as basic switching matrix is required to assemble any functional units into a working configuration.	Potential for a very high total transfer rate in overall system.
4.	Expanding may degrade system performance.	System expansion improves overall performance.	Difficult to modify as the design decision is made quite early.
5.	Lowest efficiency.	Expansion of system is limited only by size of the switch matrix.	Large number of cables and connector are required.
6.	Organization is suitable for smaller systems only.	Reliability of system can be improved by segmentation or redundancy within switch.	

Table Q.46.1

12.11 : Multithreading

Q.47 What is multithreading ? [SPPU : May-18, Marks 2]

Ans. : • In multithreading, the instruction stream is divided into several smaller streams, called threads, such that the threads can be executed in parallel. Here, a high degree of instruction-level parallism can be achieved without increasing circuit complexity or power consumption.

Q.48 Define process, resource ownership, scheduling / execution process switch, thread and thread switch.

Ans. :

- **Process :** A process is an instance of a program running on a computer. The process image is the collection of program data, stack and attributes that define the process. The process image is stored at a virtual address space. Two important characteristics of a process are discussed below.

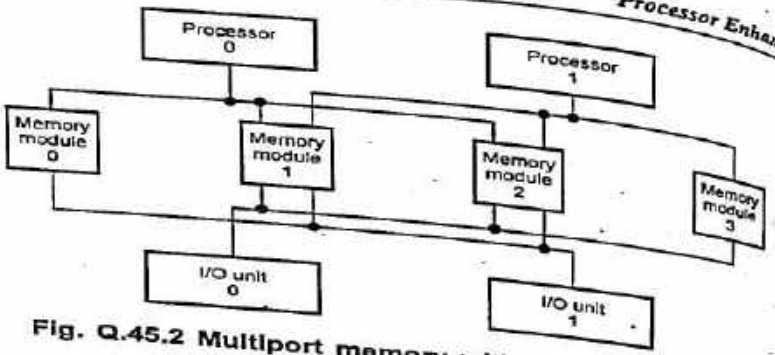


Fig. Q.45.2 Multiport memory with private memories

- However, the main disadvantage of such system is the system recovery if the other processors are not able to access control and status information in a memory block associated with a faulty processor. This organization can also support non blocking access to memory if a full-connected topology is used.

Q.46 Give comparison between time shared bus, crossbar switch and multiport memory organizations.

Ans. :

Sr. No.	Time shared bus	Crossbar switch	Multiport memory
1.	Lowest overall system cost for hardware and Least complex	Most complex. Highest total transfer rate.	Most expensive since most of control and switching circuitry is in the memory unit.
2.	Easy to modify hardware system configuration.	Functional units are simplest and cheapest.	Functional units permit low cost uniprocessor.
3.	Overall system capacity limited by bus transfer rate. Bus fail then whole system fail.	Usually cost-effective for multiprocessors only as basic switching matrix is required to assemble any functional units into a working configuration.	Potential for a very high total transfer rate in overall system.
4.	Expanding may degrade system performance.	System expansion improves overall performance.	Difficult to modify as the design decision is made quite early.
5.	Lowest efficiency.	Expansion of system is limited only by size of the switch matrix.	Large number of cables and connector are required.
6.	Organization is suitable for smaller systems only.	Reliability of system can be improved by segmentation or redundancy within switch.	

Table Q.46.1

12.11 : Multithreading

Q.47 What is multithreading ? [SPPU : May-18, Marks 2]

Ans. : • In multithreading, the instruction stream is divided into several smaller streams, called **threads**, such that the threads can be executed in parallel. Here, a high degree of **instruction-level parallism** can be achieved without increasing circuit complexity or power consumption.

Q.48 Define process, resource ownership, scheduling / execution process switch, thread and thread switch.

Ans. :

- **Process** : A process is an instance of a program running on a computer. The process image is the collection of program data, stack and attributes that define the process. The process image is stored at a virtual address space. Two important characteristics of a process are discussed below.

- **Resource ownership :** A process may get control of resources such as main memory, I/O channels, I/O devices and files from time to time.
- **Scheduling / execution :** A process execution takes places through one or more programs. This execution may interleaved with that of other processes. An operating system decides an execution state of each process such as running, ready, dispatching priority.
- **Process switch :** A process switch is an operation that switches the process or control from one process to another. It first saves all the process control data, registers and other information and then replaces them with the process information for the second.
- **Thread :** A thread includes the program counter, stack pointer and its own area for a stack. It executes sequentially and can be interrupted to transfer control to an another thread.
- **Thread switch :** A thread switch is an operation that switches the processor control from one thread to another within the same process. This is cheaper than a process switch.

Q.49 Give comparison of process switch and thread switch.

Sr. No.	Process switch	Thread switch
1.	It is an operation that switches the process or control from one process to another.	It is an operation that switches the processor control from one thread to another thread.
2.	When the processor control is transferred from one process to another, the control or ownership of resources is also transferred. So process switch is time consuming than thread switch.	The multiple threads within a process share the same resources. So a thread switch is much less time consuming than a process switch.
3.	It is much costly than a thread switch.	It is much less costly than process switch.

Q.50 What do you mean by implicit and explicit multithreading.

Ans. :

- User level threads which are visible to the application program and kernel-level threads which are visible only to operating system, both are referred to as explicit threads.
- Implicit multithreading refers to the concurrent execution of multiple threads extracted from a single sequential program.
- Explicit multithreading refers to the concurrent execution of instructions from different explicit threads, either by interleaving instructions from different threads on shared pipelines or by parallel execution on parallel pipelines.

Q.51 Discuss various approaches to explicit multithreading.

OR **What is multicore ?**

Ans. : **Approaches to Explicit Multithreading :**

- **Interleaved or fine-grained multithreading :** The processor executes two or more threads at a time. It switches from one thread to another at each clock cycle. During execution, if a thread is blocked because of data dependencies or memory latencies, that thread is skipped and a ready thread is executed.
- **Blocked or coarse-grained multithreading :** The processor executes instructions of a thread sequentially and if an event (e.g. cache miss) that causes any delay occurs, it switches to another thread.
- **Simultaneous multithreading (SMT) :** The wide superscalar instruction is executed by executing multiple threads simultaneously using multiple execution units of a superscalar processor.
- **Chip multiprocessing :** The processor is replicated on a single chip and each processor executes separate threads. This approach effectively utilizes the available logic data on a chip without increasing pipeline design complexity. This is referred to as multicore. Chip multiprocessing enables simultaneous execution of instructions from different threads.

12.12 : Clusters and Cluster Configurations

Q.52 What are clusters ?

OR What is cluster computing ? ☞[SPPU : June-17, May-18,19, Marks 2]

Ans. : • An important and relatively recent development in computer system design is clustering. Clustering is an alternative to symmetric multiprocessing (SMP) as an approach to provide high performance and high availability and is particularly attractive for server applications. A **computer cluster** is a group of linked computers, working together closely so that in many respects they form a single computer. The components of a cluster are commonly, but not always, connected to each other through fast local area networks. Here, computer means a system that can run its own, a part from the cluster. Such a computer in a cluster is typically referred to as a **node**. Clusters are usually deployed to improve performance and/or availability provided by a single computer, while typically being much more cost-effective than single computers of comparable speed or availability.

Q.53 State the advantages of clustering ?

OR Explain benefits of clustering. ☞[SPPU : Dec.-16, May-18, Marks 3]

Ans. :

1. **Absolute scalability :** It is possible to have a cluster with dozens or even hundreds of machines and each machine can be a multiprocessor.
2. **Incremental scalability :** It is possible to upgrade existing machines in the cluster with modern machine or machines with higher performance.
3. **High availability :** Because each node in a cluster is a standalone computer, the failure of one node does not halt the service. Another node is assigned to perform the task of failure node.
4. **Cost effective :** Clusters are cost-effective than single computers of comparable speed or computing power.

Q.54 Explain cluster configurations. ☞[SPPU : Dec.-16,17,19, Marks 4]

OR What are the types of clustering ? ☞[SPPU : June-17, Marks 4]

Ans. : • Clusters are classified in a number of different ways. The classification can be based on whether the computers in a cluster share access to the same disk, whether every single node in the cluster is

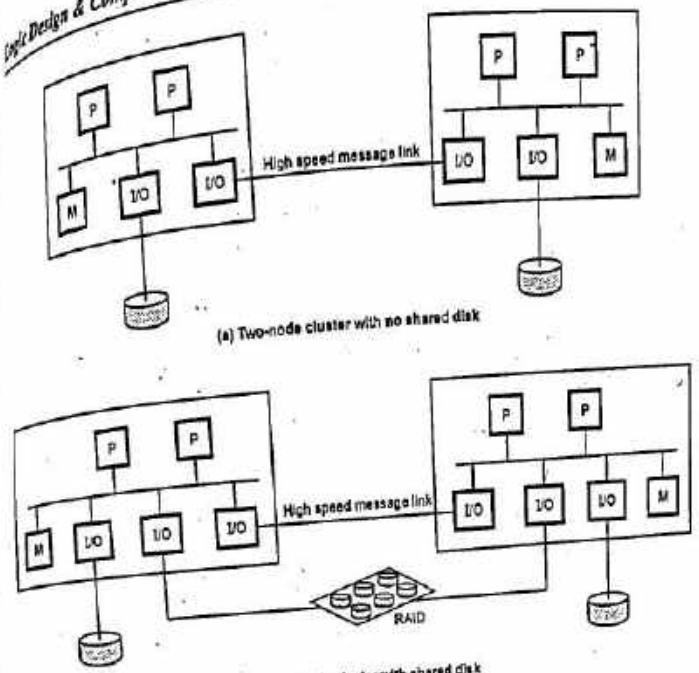


Fig. Q.54.1

exactly same and so on. The Fig. Q.54.1 shows the classification of cluster according to shared access. The Fig. Q.54.1 (a) shows a two-node cluster with no shared memory between two nodes. The interconnection between two nodes is by means of high-speed link. On the other hand, Fig. Q.54.1 (b) shows a two-node cluster with shared disk.

Homogeneous clusters : • It is a cluster type in which every single node is exactly same, from mother and memory, to the disk drives and the network controller cards.

Heterogeneous clustors : • Heterogeneous clusters come in two general forms :

1. Made from different kinds of computers. For e.g. a few Sun SPARC station IPXs, a few Intel 486 machines, and a DEC Alpha.
2. Made from different machines in the same architecture family: For e.g. a collection of Intel boxes where the machines are of different generations (e.g. a mixture of 486, Pentium I and Pentium II).

Q.55 State various clustering methods, their benefits and limitations.

Ans. : • Table Q.55.1 shows various clustering methods, their benefits and limitations.

Sr. No.	Clustering method	Description	Benefit	Limitations
1.	Passive Standby	• Older method • Primary server handles all of the processing load while secondary server remains inactive. • Secondary server takes over in case of failure of primary server. • Since only one machine does processing work it is not referred to as a cluster.	It is easy to implement.	Effective cost is high since only one server is active at a time.
2.	Active standby or Active secondary	• Secondary server is also used for processing tasks. • They are classified as - Separate servers - Shared nothing - Shared memory	Effective cost is reduced because secondary servers are used for processing tasks.	Complexity is more.
3.	Active server with separate servers	• Separate servers with own disk. • There is no shared disk • Data is continuously copied among systems so that each system has access to the current data of the other systems. This makes it possible to take over the processing by other system in case of failure of any one system.	High performance. High availability.	High data exchange overhead.

4.	Active server with servers connected to disks	• The common disks are partitioned into volumes, and each volume is owned by single computer. • If one computer fails, the other computer takes the ownership of the volumes of the failed computer.	Reduces data exchange overhead.	Requires disk mirroring or RAID technology to compensate for risk of disk failure.
5.	Active server with shared disks	• Multiple computers share same disk at same time, so that each computer has access to all of the volumes on all of the disks.	Reduces data exchange overhead. Reduces risk of downtime caused by disk failure.	Required lock manager software.

Table Q.55.1 Clustering methods, their benefits and limitations

Q.56 State the operating system design issues in the clustered organization.

Ans. :

1. **Failure management :** Fail over recovery means that if one of the participants in the cluster is temporary or permanently unavailable, for instance because of a hardware crash, its functions are undertaken by another participant. As a result, requests are automatically redirected to a working server in the cluster and users and applications are not aware of what is happening, because the system continues to process requests properly.

   The function of switching an application and data resources over from a failed the participant to the working server in the cluster is referred to as **failover.** A related function is the restoration of application and data resources to the original system once the system is out of failure is referred to as **failback.**

   The operating system should support failover and failback functions.

2. **Load balancing :** The operating system should distribute the tasks properly among available computers in the cluster to achieve the optimum performance when a new computer is added to the cluster, the operating system should restructure the distribution of task.

8.	Since the scalability is more, can dominate high performance server market.	Since the scalability is low, less preferred in high performance server market.
9.	Clusters have higher availability.	SMPs have lower availability than clusters.

12.13 : UMA (Uniform Memory Access)

Q.59 Write a short note on UMA. [SPPU : Dec.-18, Marks 2]

Ans. : • Uniform Memory Access (UMA) is a shared memory architecture used in parallel computers. All the processors in the UMA model share the physical memory uniformly. In a UMA architecture, access time to a memory location is independent of which processor makes the request or which memory chip contains the transferred data.

- Uniform Memory Access computer architectures are often contrasted with Non-Uniform Memory Access (NUMA) architectures. In the UMA architecture, each processor may use a private cache. Peripherals are also shared. The UMA model is suitable for general purpose and time sharing applications by multiple users. It can be used to speed up the execution of single large program in time critical applications.

Types of UMA architectures

1. UMA using bus-based SMP architectures.
2. UMA using crossbar switches.
3. UMA using multistage switching networks.

12.14 : NUMA and CC-NUMA

Q.60 What is NUMA ? [SPPU : Dec.-18,19, Marks 2]

Ans. : • Non Uniform Memory Access or Non-Uniform Memory Architecture (NUMA) is a computer memory design used in multiprocessors, where the memory access time depends on the memory location relative to a processor. Under NUMA, a processor can access its

own local memory faster than non-local memory, that is, memory local to another processor or memory shared between processors.

Q.61 Draw and explain the CC-NUMA organization. [SPPU : Dec.-18, Marks 2]

Ans. : • The term CC-NUMA stands for **Cache-Coherent Non Uniform Memory Access**. In the CC-NUMA model, the system runs one operating system and shows only a single memory image to the user eventhough the memory is physically distributed over the processors. Since processors can access their own memory much faster than that of other processors, memory access is non uniform (NUMA). In this architecture the contents of the various processor caches should be coherent requiring extra hardware and a cache coherency protocol. A NUMA computer fulfilling these requirements is called a **CC-NUMA** machine.

• Fig. Q.61.1 shows the CC-NUMA organization. There are multiple independent nodes, each of which, in effect, an SMP organization. Thus, each node consists of multiple processors, each with its own L_1 and L_2 caches along with the main memory.

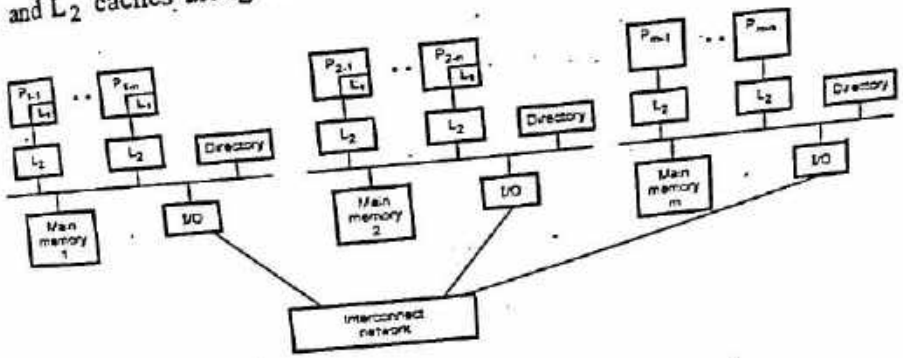


Fig. Q.61.1 CC-NUMA organization

- Comparing a CC-NUMA computer to cluster-based systems and SMP (Symmetric Multiprocessing) systems shows that CC-NUMA machines can be regarded as an attempt to get the best out of both worlds.
- SMP computers have multiple processors using the same memory which makes it easy to parallelize a code on loop level using compiler options and/or directives. However, due to the fact that processors have

to perform data exchange with their memory over the same bus having only a limited bandwidth, these systems will only scale to 10s of processors.

- Cluster-based systems do not have this drawback. The nodes in cluster-based machines have their own private memory and therefore each node possesses only a part of the data. Such machines will scale upto a very large number of processors if the computation to communication ratio of the program is high. The programming model for these computers is based on message passing and processors explicitly have to perform the communication with other processors by sending and receiving data. Programs have to take care of data distribution over the processors and have to be adapted for explicit communication. This implies that programs developed for single processor and SMP machines cannot be applied to these machines straight away. A substantial amount of redesigning and reprogramming of these codes is necessary.
- CC-NUMA machines combine the benefits of cluster-based systems and SMP machines. The fact that CC-NUMA machines behave like shared memory computers from a user point of view simplifies the porting of programs developed for single processor or SMP machines. Moreover, CC-NUMA computers allow for loop-level parallelism by means of compiler options or compiler directives similar to SMP systems. The good scalability properties are inherited from MPP/cluster-based systems since memory is distributed over the nodes.

12.15 : Multicore Architectures

Q.62 Write a note on multicore architecture.

☞ [SPPU : June-17,18, Marks 7]

OR What is multicore computers ? ☞ [SPPU : Dec.-16,17,18, Marks 3]

Ans. : • Multicore architectures are classified according to :

- Number of processor : Two processors (dual core), four processors (quad core), and eight processors (octa-core) configurations.
- Approaches to processor - to - processor communication.

- Cache and memory implementations.
- Implementation of I/O bus and the Front Side Bus (FSB).

Fig. Q.62.1 shows three common configurations that support multiprocessing.

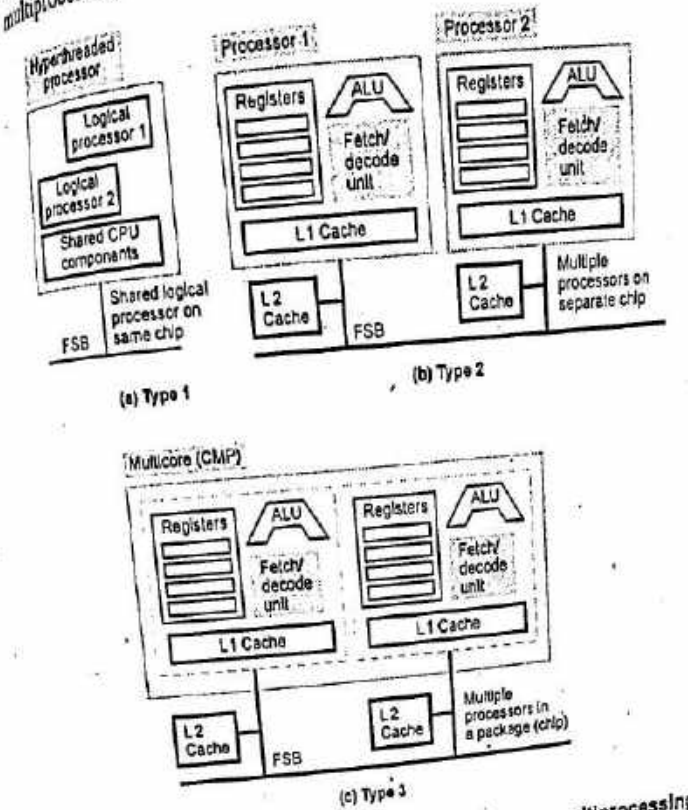


Fig. Q.62.1 Common configurations that support multiprocessing

- **Type 1 :** It uses hyperthreading technology. It allows a single processor to act like two separate processors to the operating system and the application programs that use it; however, there is a single processor running multiple threads. Thus, in hyperthreaded technology the multiple processors are logical instead of physical. In hyperthreaded technology has some duplication of hardware but not enough to qualify a separate physical processor.

- **Type 2 :** It is the classic multiprocessor. A multiprocessor is a tightly coupled computer system having two or more processing units (Multiple Processors) on a separate chip with its own hardware.
- **Type 3 :** It represents multicore system. It provides two or more complete processors on a single chip.

12.16 : Hardware Performance Issues

Q.63 Explain hardware performance issues of same.

☞ [SPPU : Dec.-16, Marks 5]

OR Describe hardware and software performance issues.

☞ [SPPU : May-19, Marks 5]

Ans. : Hardware Performance Issues : • Increase in Parallelism

- Pipelining
- Superscalar (multi-issue)
- Simultaneous multithreading (SMT)

All these improvement increase the complexity.

- **Power Consumption :** To maintain the trend of higher performance
  - Number of transistors per chip increase,
  - Designers have to choose more complicated processor designs (pipelining, superscalar, SMT) and to use high clock frequencies.
  - However, power requirements have grown exponentially as chip density and clock frequency have risen.

Software Performance Issues

- Performance benefits dependent on effective exploitation of parallel resources.
- Even small amounts of serial code impact performance due to communication, distribution of work and cache coherence overheads. 10% inherently serial on 8 processor system gives only 4.7 times performance.
- Some applications effectively exploit multicore processors.

12.17 : Multicore Organizations

Q.64 Explain various multicore organization. ☞ [SPPU : May-19, Marks 2]

Ans. :

- Multicore organizations are classified according to :
  - The number of core processors on the chip.
  - The number of levels of cache memory.
  - The amount of cache memory that is shared.
- Fig. Q.64.1 shows four general organizations for multicore systems. The organization shown in Fig. Q.64.1 (a) has the only on-chip cache is L1 cache, with each core having its own dedicated L1 cache. Most of the times, the L1 cache is divided into instruction and data caches. An example of this organization is the ARM11 MPCore.

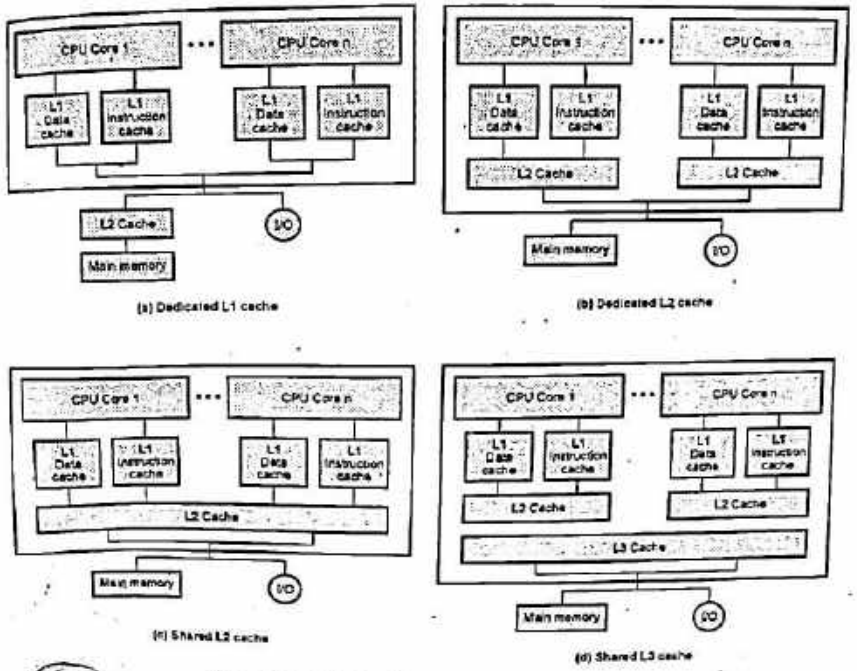


Fig. Q.64.1 Multicore organizations

- The organization shown in Fig. Q.64.1 (b) has no on-chip cache sharing. In this, there is enough area available on the chip to allow for L2 cache. An example of this organization is the AMD Opteron.
- The organization shown in Fig. Q.64.1 (c) has a similar allocation of chip space to memory, but with the use of a shared L2 cache. The Intel Core Duo has this organization.
- The organization shown in Fig. Q.64.1 (d) has shared L3 cache, with dedicated L1 and L2 caches for each core processor. The Intel Core i7 is an example of this organization.

12.18 : Intel X86 Multicore Organizations

Q.65 Draw and explain the organization of Intel Core Duo.

☞ [SPPU : May-18, Marks 3]

Ans. : • **Caches and Thermal Unit :** The Intel Core Duo, implements two x86 superscalar processors with a shared L2 cache. Fig. Q.65.1 shows the general structure of the Intel Core Duo. As shown in the Fig. Q.65.1, each core has a 32-kB instruction cache and a 32-kB data cache and an independent thermal control unit. The Core Duo thermal control unit is designed to manage chip heat dissipation to maximize processor performance within thermal constraints.

- **Advanced Programmable Interrupt Controller (APIC) :** It performs a number of functions, including the following :
  - Can provide interprocessor interrupts, which allow any process to interrupt any other processor or set of processors. In Core Duo, a thread in one core can generate an interrupt, which is accepted by the local APIC, routed to the APIC of the other core, and communicated as an interrupt to the other core.
  - Accepts I/O interrupts and routes these to the appropriate core.
  - Timer provided by each APIC can be set by the OS to generate an interrupt to the local core.
- **Power Management Logic :** It is responsible for reducing power consumption when possible, thus increasing battery life. It also monitors thermal conditions and adjusts voltage levels. It includes an advanced power-gating capability that allows for an ultra fine-grained

logic control that turns on individual processor logic subsystems only if and when they are needed. With power management logic, data required in some modes of operation can be put in a low power state when not needed.

- **Shared 2-MB L2 cache :** The cache logic allows for a dynamic allocation of cache space based on current core needs, so that one core can be assigned up to 100% of the L2 cache. The L2 cache includes logic to support the MESI protocol for the attached L1 caches.
- **Bus Interface :** It connects to the external bus, known as the Front Side Bus, which connects to main memory, I/O controllers, and other processor chips.

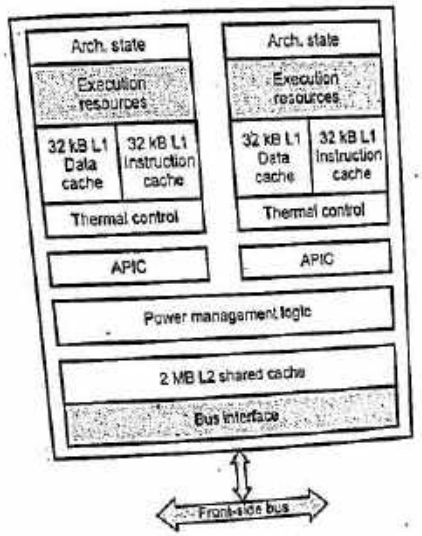


Fig. Q.65.1 Intel Core Duo block diagram

Q.66 Draw and explain the organization of Intel Core i7.

☞ [SPPU : May-18, Dec.-19, Marks 6]

Ans. : • The Intel Core i7, implements four x86 SMT processors, each with a dedicated L2 cache, and with a shared L3 cache. Fig. Q.66.1 shows the general structure of the Intel Core i7. Each core has its own dedicated L2 cache and the four cores share an 8-MB L3 cache. Prefetching is used to make its caches more effective.

- With the use of the dedicated L2 caches and a relatively high-speed access to the L3 cache, the Core i7 gives improved performance.
- The Core i7 chip supports two forms of external communications to other chips : The **DDR3 memory controller** and **QuickPath Interconnect (QPI).**
- **DDR3 memory controller :** It supports three channels that are 8 bytes wide for a total bus width of 192 bits, for an aggregate data rate of up to 32 GB/s. With the memory controller on the chip, the Front Side Bus is eliminated.
- **QuickPath Interconnect (QPI) :** It is a cache-coherent, point-to-point link based electrical interconnect specification for Intel processors and chipsets. It enables high-speed communications among connected processor chips. The QPI link operates at 6.4 GT/s (transfers per second). At 16 bits per transfer, that adds up to 12.8 GB/s, and since QPI links involve dedicated bidirectional pairs, the total bandwidth is 25.6 GB/s.

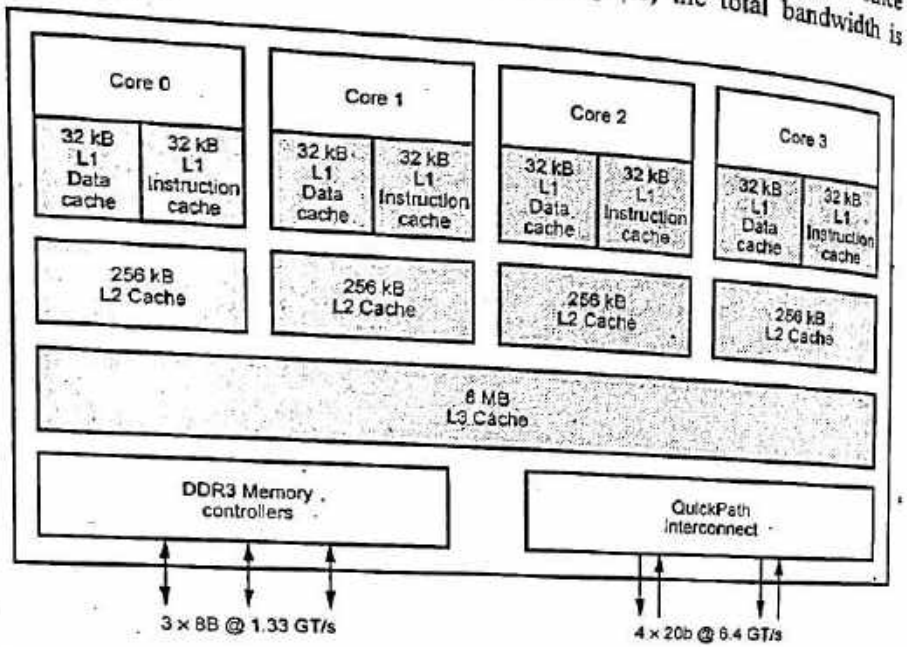


Fig. Q.66.1 Intel Core i7 block diagram

END...

Unit 6

13 Memory Systems

13.1 : Characteristics of Memory Systems

Q.1 State the key characteristics of memory systems.

☞ [SPPU : Dec.-15, Marks 2]

Ans. : Table Q.1.1 lists the key characteristics of memory systems.

Location	:	CPU Internal (main) External (Secondary)
Capacity	:	Word size Number of words
Unit of transfer	:	Word Block
Access method	:	Sequential access Direct access Random access Associative access
Performance	:	Access time, Cycle time, Transfer rate
Physical type	:	Semiconductor Magnetic surface
Physical characteristics	:	Volatile / non-volatile
Organization	:	erasable / non-erasable

Table Q.1.1

13.2 : Memory Hierarchy

Q.2 Explain memory hierarchy.

Ans. :

- Ideally, computer memory should be fast, large and inexpensive. Unfortunately, it is impossible to meet all the three of these requirements using one type of memory.
- Increased speed and size are achieved at increased cost.
- In make efficient computer system it is not possible to rely on a single memory component, but to employ a memory hierarchy. Using memory hierarchy all of different types of memory units are employed to give efficient computer system. A typical memory hierarchy is illustrated in Fig. Q.2.1.
- In summary, we can say that a huge amount of cost-effective storage can be provided by magnetic disks. A large, yet affordable, main memory can be built with DRAM technology along with the cache memory to achieve better speed performance.

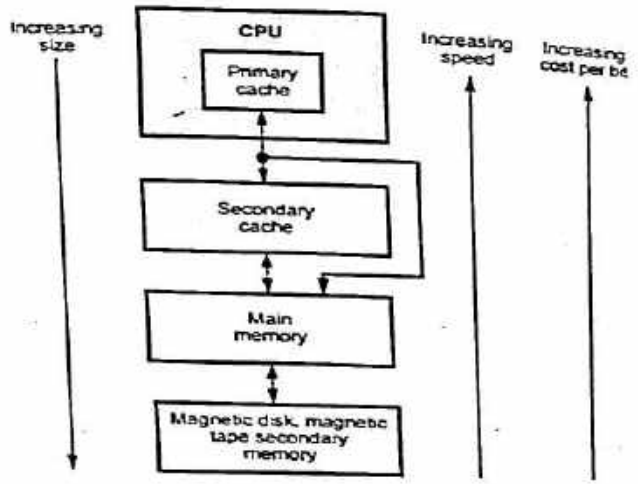


Fig. Q.2.1

13.3 : Signals to Connect Memory to Processor

Q.3 Draw the connection between memory and processor and explain how data transfer takes place between them.

Ans. :

- The maximum size of the memory that can be used in any computer is determined from the number of address lines provided by the processor used in the computer. For example if processor has 20 address lines, it is capable of addressing upto 2^{20} = 1 M (Mega) memory locations. Similarly, processors whose instructions generate 32-bit address can access a memory that contains up to 2^{32} = 4 G (Giga) memory locations. The number of locations represents the size of the address space of the computer.
- The maximum number of bits that can be transferred from memory or to the memory depend on the data lines supported by the processor. The number of data lines supported by processor gives the word length of the processor and hence the computer.
- The control lines from the processor decides the memory operation. In case of read operation $\overline{RD}$ signal is activated. It is used to enable the active low output enable signal of the memory. In case of write operation $\overline{WR}$ signal is activated to indicate the write operation. The data transfer between the memory and processor takes place through the use of two processor registers, usually called MAR (Memory Address Register) or simply AR (Address Register) and MDR (Memory Data Register) or simply DR (Data Register). This is

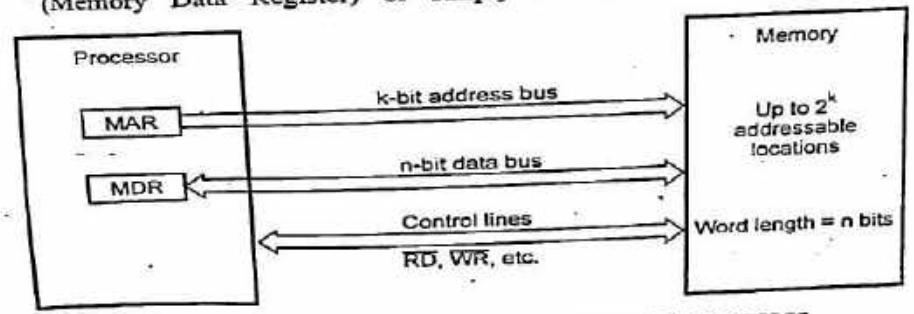


Fig. Q.3.1 Connection between memory and processor

illustrated in Fig. Q.3.1. If MAR is k-bit long and MDR is n bit long it is possible to across upto 2^k memory locations, and during one memory cycle it is possible to transfer n-bit data.

13.4 : Memory Read and Write Cycle

Q.4 Draw and explain the memory read cycle.

Ans. : • Fig. Q.4.1 shows the timing diagram for memory read cycle.

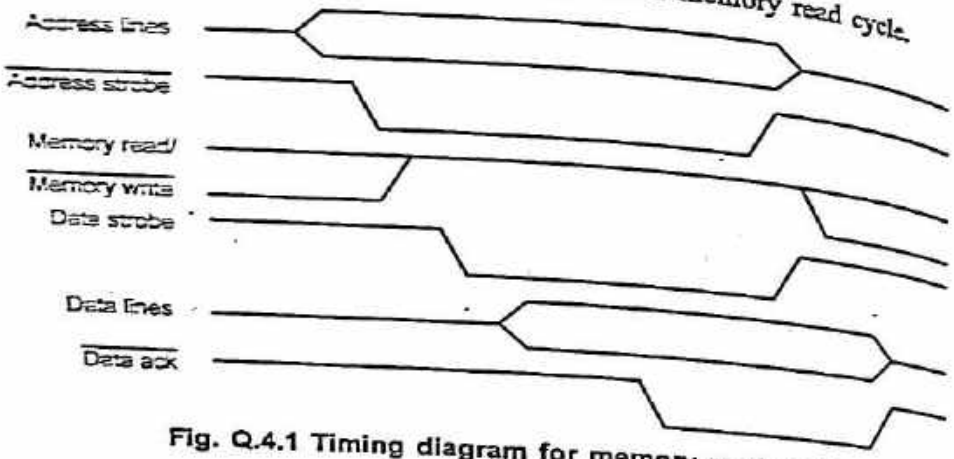


Fig. Q.4.1 Timing diagram for memory read cycle

- Processor initiates a memory read cycle by floating the address of the memory location on the address lines.
- Once the address lines are stable, the processor asserts the address strobe signal on the bus. The address strobe signals the validity of the address lines.
- Processor then sets the memory Read/$\overline{\text{Write}}$ signal to high, i.e. read cycle.
- Now the processor asserts the data strobe signal. This signals to the memory that the processor is ready to read data.
- The memory subsystem decodes the address and places the data on the data lines.
- The memory system then asserts the data acknowledge signal. This signals to the processor that valid data is available on the data bus..

- Processor latches in the data and negates the data strobe. This signals to the memory that the data has been latched by the processor.
- Processor negates the address strobe signal.
- Memory system then negates the data acknowledgement signal. This signals the end of the memory read cycle.

Q.5 Draw and explain the memory write cycle.

Ans. : Fig. Q.5.1 shows the timing diagram for memory write cycle.

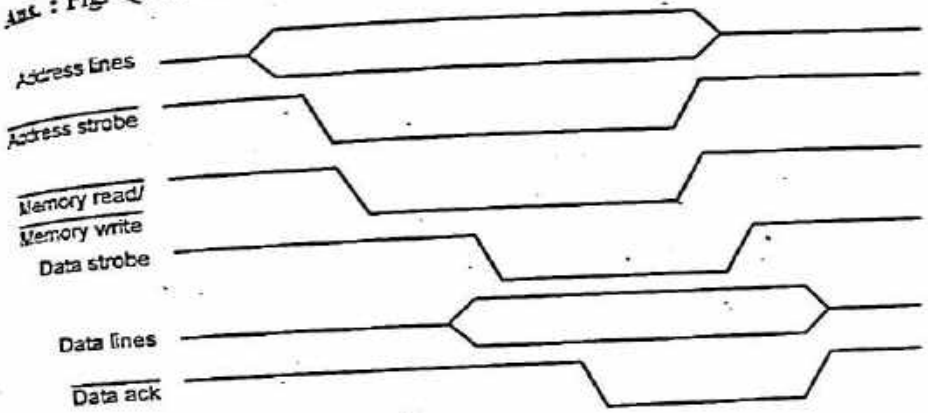


Fig. Q.5.1 Timing diagram for memory write cycle

1. Processor initiates a memory write cycle by floating the address of the memory location on the address lines.
2. Once the address lines are stable, the processor asserts the address strobe signal on the bus. The address strobe signals the validity of the address lines.
3. Processor then sets the memory Read/$\overline{\text{Write}}$ signal to low, i.e. write cycle.
4. The processor then places the data on the data lines.
5. Now the processor asserts the data strobe signal. This signals to the memory that the processor has valid data for the memory write operation.

6. The memory subsystem decodes the address and writes the data into the addressed memory location.
7. The memory system then asserts the data acknowledge signal. This signals to the processor that data has been written to the memory.
8. Then the processor negates the data strobe, signaling that the data is no longer valid.
9. Processor also negates the address strobe signal.
10. Memory system then negates the data acknowledgement signal, signaling an end to the memory write cycle.

13.5 : Characteristics of Semiconductor Memory ; SRAM, DRAM and ROM

Q.6 Write a short note on SRAM.

☞ [SPPU : May-06, 09, 11, 13, Dec.-06, Marks 6]

OR Explain DRAM with diagram. Also give advantages and disadvantages. Read write operation

☞ [SPPU : May-06, 09, Dec.-06, 07, Marks 6]

OR List the different types of internal memory explain any two in brief.

☞ [SPPU : June-15, Marks 6]

OR Draw DRAM cell and explain its read and write operation in detail.

☞ [SPPU : Dec.-15, Marks 7]

Ans. : • The internal memory is a semiconductor random access memory.

- The Fig. Q.6.1 shows the classification of semiconductor memory.
- The two basic forms of semiconductor read/write memories are :
  - Static RAM (SRAM)
  - Dynamic RAM (DRAM)

Static RAM : • Memories that consists of circuits capable of retaining their state as long as power is applied are known as static memories.

- These are Random Access Memory (RAM) and hence combinely called **static RAM memories.**

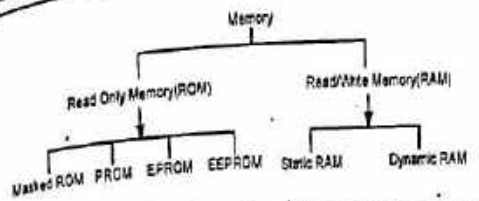


Fig. Q.6.1 Classification of semiconductor memory

Static RAM Cell :

- The Fig. Q.6.2 shows the implementation of static RAM cell. It consists of two cross-coupled inverters as a latch and two transistors T_1 and T_2 which act as a switches.

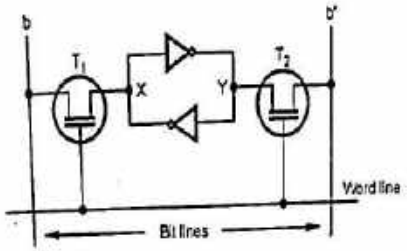


Fig. Q.6.2 Static RAM cell

- The latch is connected to two bit lines by transistors T_1 and T_2. The word line controls the opening and closing of transistors T_1 and T_2. When word line is at logic 0 level (Ground level), the transistors are off and the latch retains its state.

Read operation : • For read operation, word line is made logic 1 (high) so that both transistors are ON. Now if the cell is in state 1, the signal on bit line b is high and the signal on bit line b' is low. The opposite is true if the cell is in state 0. The b and b' are complements of each other. The sense/write circuits connected to the bit lines monitor the states of b and b' and set the output accordingly.

Write operation : • For write operation, the state to be set is placed on the line b and its complement is placed on line b' and then the word line is activated. This action forces the cell into the corresponding state and write operation is completed.

Dynamic RAMs : • Dynamic RAM stores the data as a charge on the capacitor. Fig. Q.6.3. shows the dynamic RAM cell.

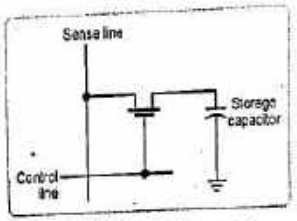


Fig. Q.6.3 Dynamic RAM

- A dynamic RAM contains thousands of such memory cells.

Write operation : • When COLUMN (Sense) and ROW (Control) lines go high, the MOSFET conducts and charges the capacitor.

- When the COLUMN and ROW lines go low, the MOSFET opens and the capacitor retains its charge. In this way, it stores 1 bit.

Read operation : • When ROW (Control) line goes high, the MOSFET conducts and the capacitor is connected to sence line. The level of charge on the memory cell capacitor determines whether that particular bit is a logical "1" or "0". The presence of charge in the capacitor indicates a logic "1" and the absence of charge indicates a logic "0".

- Since only a single MOSFET and capacitor are needed, the dynamic RAM contains more memory cells as compared to static RAM per unit area.

- The **disadvantage** of dynamic RAM is that it needs refreshing of charge on the capacitor after every few milliseconds. This complicates the system design, since it requires the extra hardware to control refreshing of dynamic RAMs.

Q.7 Compare SRAM Vs DRAM. [SPPU : Dec.-08, 12, Marks 6]

Ans. :

Sr. No.	Static RAM	Dynamic RAM
1.	Static RAM contains less memory cells per unit area.	Dynamic RAM contains more memory cells as compared to static RAM per unit area.
2.	It has less access time hence faster memories.	Its access time is greater than static RAMs.
3.	Static RAM consists of number of flip-flops. Each flip-flop stores one bit.	Dynamic RAM stores the data as a charge on the capacitor. It consists of MOSFET and the capacitor for each cell.
4.	Refreshing circuitry is not required.	Refreshing circuitry is required to maintain the charge on the capacitors after every few milliseconds. Extra hardware is required to control refreshing. This makes system design complicated.
5.	Cost is more.	Cost is less.

Q.8 Draw and explain the working of ROM cell.

Ans. : • We can't write data in Read Only Memories (ROM). It is non-volatile memory i.e. it can hold data even if power is turned off.

- Generally, ROM is used to store the binary codes for the sequence of instructions and data such as look up tables. This is because this type of information does not change.

- ROMs are also accessed randomly with unique addresses.

- The Fig. Q.8.1 shows the typical configuration of a ROM cell. It consists of a transistor T and switch P.

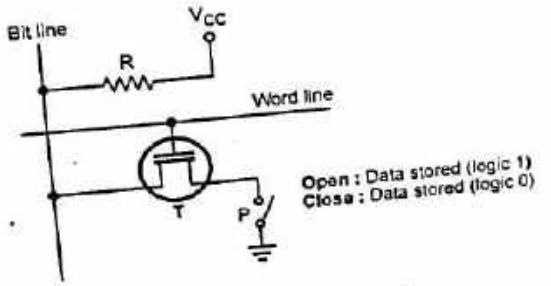


Fig. Q.8.1 ROM cell

- The transistor T is driven by the word line.

- The contents of cell can be read from the cell when word line is logic 1.

- A logic value 0 is read if the transistor is connected to ground through switch P. If switch P is open, a logic value 1 is read.

- The bit line is connected through a resistor to the power supply.

- A sense circuit at the end of the bit line generates the proper output value.

- Data is stored into a ROM when it is manufactured.

- There are four types of ROM : Masked ROM, PROM, EPROM and EEPROM or E^2PROM.

Q.9 Write short note on : EPROM

☞[SPPU : May-06, 13, Dec.-08, 12, Marks 4]

Ans. : EPROM (Erasable Programmable Read Only Memory)

- Erasable programmable ROMs use MOS circuitry. They store 1's and 0's as a packet of charge in a buried layer of the IC chip.
- EPROMs can be programmed by the user with a special EPROM programmer.
- The important point is that we can erase the stored data in the EPROMs by exposing the chip to ultraviolet light through its quartz window for 15 to 20 minutes, as shown in the Fig. Q.9.1.

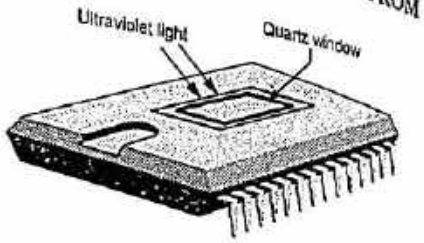


Fig. Q.9.1 EPROM

- It is not possible to erase selective information, when erased the entire information is lost.
- The chip can be reprogrammed.
- This memory is ideally suitable for product development, experimental projects and college laboratories, since this chip can be reused many times.

EPROM Programming : • When erased each cell in the EPROM contains 1. Data is introduced by selectively programming 0's into the desired bit locations. Although only 0's will be programmed, both 1's and 0's can be presented in the data.

- During programming address and data are applied to address and data pins of the EPROM. When the address and data are stable, program pulse is applied to the program input of the EPROM. The program pulse duration is around 50 ms and its amplitude depends on EPROM IC. It is typically 5.5 V to 25 V.

, In EPROM, it is possible to program any location at any time - either individually, sequentially or at random.

Q.10 Explain EEPROM. ☞[SPPU : May-06, 10, 11, Dec.-09, 10, Marks 4]

Ans. : • Electrically erasable programmable ROMs also use MOS circuitry very similar to that of EPROM.

- Data is stored as charge or no charge on an insulated layer or an insulated floating gate in the device.
- The insulating layer is made very thin (<200Å). Therefore, a voltage as low as 20 to 25V can be used to move charges across the thin barrier in either direction for programming or erasing.
- EEPROM allows selective erasing at the register level rather than erasing all the information since the information can be changed by using electrical signals.
- The EEPROM memory also has a special chip erase mode by which entire chip can be erased in 10 ms. This time is quite small as compared to time required to erase EPROM. It can be erased and reprogrammed with device right in the circuit.

Disadvantage : • EEPROMs are most expensive and the least dense ROMs.

13.6 : Cache Memory

Q.11 Explain the role of cache in memory organisation.

☞[SPPU : Dec.-15, Marks 3]

OR Explain need of cache memory and direct mapping cache organization technique.

☞[SPPU : Dec.-16, Marks 6]

Ans. : • In a computer system the program which is to be executed is loaded in the main memory (DRAM). Processor then fetches the code and data from the main memory to execute the program. The DRAMs which form the main memory are slower devices. So it is necessary to insert wait states in memory read/write cycles. This reduces the speed of execution.

- To speed up the process, high speed memories such as SRAMs must be used. But considering the cost and space required for SRAMs, it is not desirable to use SRAMs to form the main memory. The solution for this problem is come out with the fact that most of the microcomputer programs work with only small sections of code and data at a particular time.
- **Definition** : The part of program (code) and data that work at a particular time is usually accessed from the SRAM memory. This is accomplished by loading the active part of code and data from main memory to SRAM memory. This small section of SRAM memory added between processor and main memory to speed up execution process is known as **cache memory.**

Q.12 Explain the cache memory system.

OR Define cache miss, hit rate, cache slot and cache line.

Ans. : • Fig. Q.12.1 shows a simplest form of cache memory system.

- A cache memory system includes a small amount of fast memory (SRAM) and a large amount of slow memory (DRAM). This system is configured to simulate a large amount of fast memory.

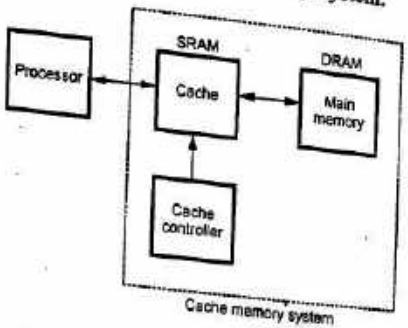


Fig. Q.12.1 Cache memory system

- Cache controller implements the cache logic. If processor finds that the addressed code or data is not available in cache - the condition referred to as **cache miss**, the desired memory block is copied from main memory to cache using cache controller. The cache controller decides which memory block should be moved in or out of the cache and in or out of main memory, based on the requirements. (The cache block is also known as **cache slot or cache line.**)

The percentage of accesses where the processor finds the code or data word it needs in the cache memory is called the **hit rate/hit ratio.** The hit rate is normally greater than 90 percent.

$$\text{Hit rate} = \frac{\text{Number of hits}}{\text{Total number of bus cycles}} \times 100\ \%$$

Q.13 The application program in a computer system with cache uses 1400 instruction acquisition bus cycle from cache memory and 100 from main memory. What is the hit rate ? If the cache memory operates with zero wait state and the main memory bus cycles use three wait states, what is the average number of wait states experienced during the program execution ?

Ans. : $\text{Hit rate} = \frac{1400}{1400+100} \times 100 = 93.3333\ \%$

$$\text{Total wait states} = 1400 \times 0 + 100 \times 3 = 300$$

$$\text{Average wait states} = \frac{\text{Total wait states}}{\text{Number of memory bus cycles}} = \frac{300}{1500} = 0.2$$

Q.14 What is principle of locality or locality of reference ?

Ans. : • We know that program may contain a simple loop, nested loops or a few procedures that repeatedly call each other. The point is that many instructions in localized area of the program are executed repeatedly during some time period and the remainder of the program is accessed relatively infrequently. This is referred to as **locality of reference.**

- It manifests itself in two ways : **temporal** and **spatial.**
- The temporal means that a recently executed instruction is likely to be executed again very soon.
- The spatial means that instructions stored near by to the recently executed instruction are also likely to be executed soon.
- The temporal aspect of the locality of reference suggests that whenever an instruction or data is first needed, it should be brought into the cache and it should remain there until it is needed again.
- The spatial aspect suggests that instead of bringing just one instruction or data from the main memory to the cache, it is wise to bring several instructions and data items that reside at adjacent address as well. We

use the term block to refer to a set of contiguous addresses of some size.

Q.15 Explain commonly used cache organizations.

Ans. : Two most commonly used system organizations for cache memory are :

- Look-aside and
- Look-through

Look-aside system organization

- The Fig. Q.15.1 shows look-aside system of cache organization. Here, the cache and the main memory are directly connected to the system bus.

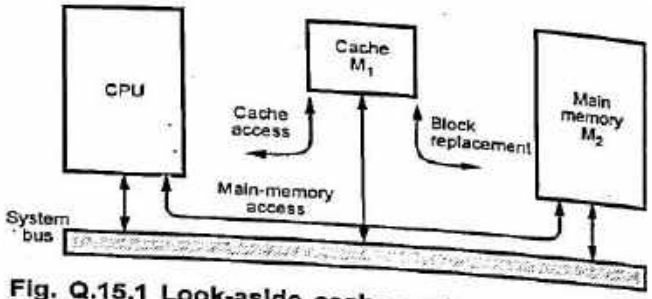


Fig. Q.15.1 Look-aside cache system organization

- In this system, the CPU initiates a memory access by placing a physical address on the memory address bus at the start of read or write cycle.
- The cache memory M_1 immediately compares physical address to the tag addresses currently residing in its tag memory. If a match is found, i.e., in case of **cache hit**, the access is completed by a read or write operation executed in the cache. The main memory is not involved in the process of read or write.
- If match is not found, i.e., in case of **cache miss**, the desired access is completed by a read or write operation directed to M_2. In response to cache miss, a block of data that includes the target address is transferred from M_2 to M_1. The system bus is used for this transfer and hence it is unavailable for other uses like I/O operations.

Look-through system organization

- Fig. Q.15.2 shows look-through system of cache organization. Here, the CPU communicates with cache via a separate (local) bus which is isolated from the main system bus. Thus during cache accesses, the system bus is available for use by other units, such as I/O controllers, to communicate with main memory.

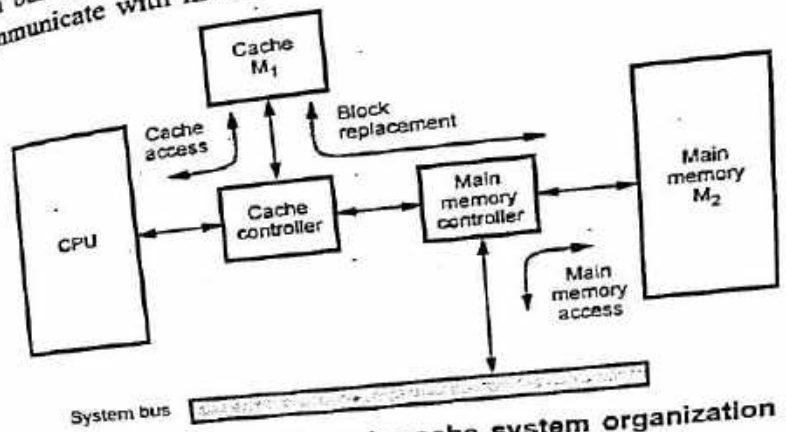


Fig. Q.15.2 Look-through cache system organization

- Unlike the look-aside system, look-through cache system does not automatically send all memory requests to main memory; it does so only after a cache miss.
- A look-through cache systems use wider local bus to link M_1 and M_2, thus speeding up cache-main-memory transfers (block transfers).
- Look-through cache system is faster.

Q.16 Explain the elements of cache design.

Ans. : • The cache design elements include cache size, mapping function, replacement algorithm write policy, block size and number of caches.

- **Cache size :** The size of the cache should be small enough so that the overall average cost per bit is close to that of main memory alone and large enough so that the overall average access time is close to that of the cache alone.
- **Mapping function :** The cache memory can store a reasonable number of blocks at any given time, but this number is small compared to the

total number of blocks in the main memory. Thus we have to use mapping functions to relate the main memory blocks and cache blocks. There are two mapping functions commonly used : direct mapping and associative mapping.

- **Replacement algorithm :** When a new block is brought into the cache, one of the existing blocks must be replaced, by a new block.
- There are four most common replacement algorithms :
  - Least-Recently-Used (LRU)
  - First-In-First-Out (FIFO)
  - Least-Frequently-Used (LFU)
  - Random
- Cache design change according to the choice of replacement algorithm.
- **Write policy :** It is also known as cache updating policy. In cache system, two copies of the same data can exist at a time, one in cache and one in main memory. If one copy is altered and other is not, two different sets of data become associated with the same address. To prevent this, the cache system has updating systems such as : write through system, buffered write through system and write-back system. The choice of cache write policy also change the design of cache.
- **Block size :** It should be optimum for cache memory system.
- **Number of caches :** When on-chip cache is insufficient, the secondary cache is used. The cache design changes as number of caches used in the system changes.

Q.17 What are the different cache mapping techniques ? Explain any one with neat diagram. ☞[SPPU : Dec.-08, Marks 8]

OR Explain direct cache mapping techniques along with its merits and demerits. ☞ [SPPU : Dec.-06, May-10, Marks 6]

OR Explain set associative cache mapping techniques along with its merits and demerits. ☞[SPPU : Dec.-06, 10, Marks 6]

OR Draw and explain :

i) Direct cache mapping ☞[SPPU : Dec.-16]

ii) Associative cache mapping ☞[SPPU : June-16]

iii) Set associative cache mapping techniques along with its merits and demerits. ☞[SPPU : Dec.-07, 09, May-11, 12, Marks 14]

OR Explain 2-way set associative cache organization. ☞[SPPU : May-14, Marks 8]

OR Explain direct mapping technique used in cache memory. ☞[SPPU : June-15, Marks 6]

OR Explain any one type of cache mapping technique with diagram. ☞[SPPU : June-17, Marks 6]

Ans. : • The mapping techniques are classified as :

1. Direct-mapping technique
2. Associative-mapping technique

• Fully-associative • Set-associative techniques.

Direct-Mapping : • It is the simplest mapping technique.

- In this technique, each block from the main memory has only one possible location in the cache organization.
- This means that to determine whether requested word is in the cache, only tag field is necessary to be compared. This needs only one comparison.
- The main drawback of direct mapped cache is that if processor needs to access same memory locations from two different pages of the main memory frequently, the controller has to access main memory frequently. Since only one of these locations can be in the cache at a time. For example, if processor want to access memory location 100 H from page 0 and then from page 2, the cache controller has to access page 2 of the main memory. Therefore, we can say that direct-mapped cache is easy to implement, however, it is not very flexible.

Associative-Mapping (Fully-Associative Mapping) :

- Fig. Q.17.1 shows the associative-mapping technique. In this technique, a main memory block can be placed into any cache block position. As there is no fix block, the memory address has only two fields : word and tag.

- This technique gives complete freedom in choosing the cache location in which to place the memory block. Thus, the memory space in the cache can be used more efficiently.

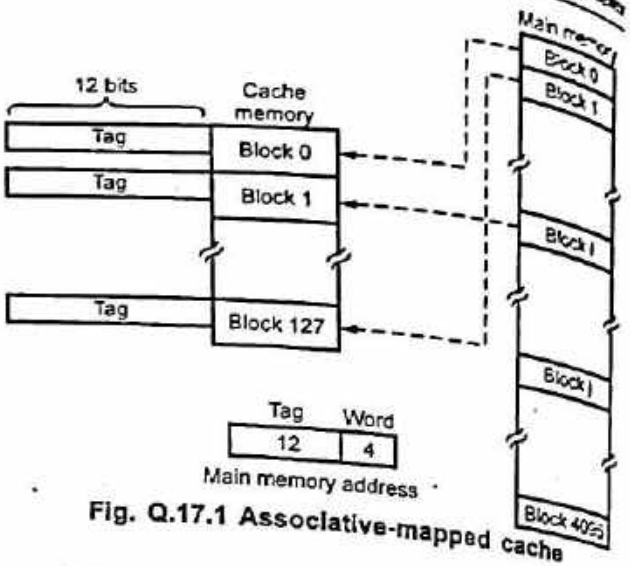


Fig. Q.17.1 Associative-mapped cache

- A new block that has to be loaded into the cache has to replace (remove) an existing block only if the cache is full.
- In such situations, it is necessary to use one of the possible replacement algorithm to select the block to be replaced.
- **Disadvantage :** In associative-mapped cache, it is necessary to compare the higher-order bits of address of the main memory with all tags corresponding to each block to determine whether a given block is in the cache. This is the main disadvantage of associative-mapped cache.

Set-Associative Mapping

- The set-associative mapping is a combination of both direct and associative mapping.
- It contains several groups of direct-mapped blocks that operate as several direct-mapped caches in parallel.
- A block of data from any page in the main memory can go into a particular block location of any direct-mapped cache. Hence the contention problem of the direct-mapped technique is eased by having a few choices for block placement.

- The required address comparisons depend on the number of direct-mapped caches in the cache system. These comparisons are always less than the comparisons required in the fully-associative mapping.
- In two-way set-associative cache, each block from main memory has two choices for block placement.
- As there are two choices, it is necessary to compare address of memory with the tag bits of corresponding two block locations of particular set. Thus for two-way set-associative cache, we require two comparisons to determine whether a given block is in the cache.
- Since there are two direct-mapped caches, any two bytes having same offset from different pages can be in the cache at a time. This improves the hit rate of the cache system.
- To implement set-associative cache system, the address is divided into three fields, Tag, set and word.

Q.18 Consider a cache consisting of 256 blocks of 16 words each, for a total of 4096 (4 K) words and assume that the main memory is addressable by a 16-bit address and it consists of 4 K blocks. How many bits are there in each of the TAG, BLOCK/SET and word fields for different mapping techniques ?

☞[SPPU : May-07, Dec.-12, Marks 12]

Ans. : We know that memory address is divided into three fields. We will now find the exact bits required for each field in different mapping techniques.

a) Direct-mapping : Word bits : We know that each block consists of 16 words. Therefore, to identify each word we must have ($2^4 = 16$) four bit reserved for it.

Block bits : The cache memory consists of 256 blocks and using direct-mapped technique, block k of the main memory maps onto block k modulo 256 of the cache. It has one to one correspondence and requires unique address for each block. To address 128 block we require ($2^8 = 256$) eight bits.

Tag bits : The remaining 4 (16 – 4 – 8) address bits are tag bits which stores the higher address of the main memory.

The main memory address for direct-mapping technique is divided as shown below :

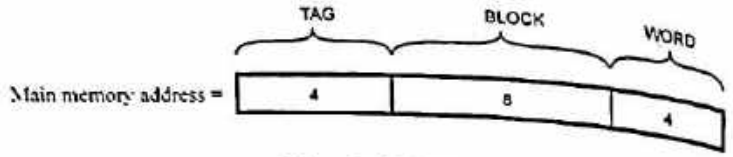


Fig. Q.18.1 (a)

b) Associative-mapping : Word bits : The word length will remain same i.e. 4 bits.

- In the associative-mapping technique, each block in the main memory is identified by the tag bits and an address received from the CPU is compared with the tag bits of each block of the cache to see if the desired block is present. Therefore, this type of technique does not have block bits, but all remaining bits (except word bits) are reserved as tag bits.

Block bits : 0 : Tag bits : To address each block in the main memory (2^{12}= 4096) 12 bits are required and therefore, there are 12 tag bits.

The main memory address for direct mapping technique is divided as shown below :

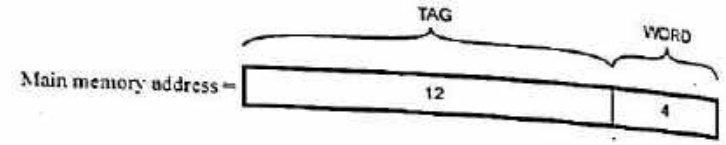


Fig. Q.18.1 (b)

c) Set-associative mapping : Let us assume that there is a 2-way set-associative mapping. Here, cache memory is mapped with two blocks per set. The set field of the address determines which set of the cache might contain the desired block.

Word bits : The word length will remain same i.e. 4 bits

Set bits : There are 128 sets (256/2). To identify each set (2^7 =128) seven bits are required.

Tag bits : The remaining 5 (16 – 4 – 7) address bits are the tag bits which stores higher address of the main memory.

The main memory address for 2-way set associative mapping technique is divided as shown below :

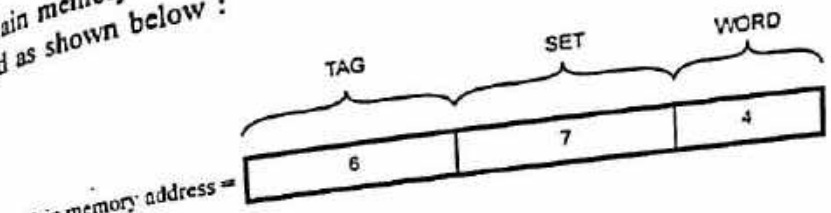


Fig. Q.18.1(c)

Q.19 A block set-associative cache consists of 64 blocks divided into 4 block sets. The main memory contains 4096 blocks, each consists of 128 words of 16 bits length :

i) How many bits are there in main memory ?

ii) How many bits are there in each of the TAG, SET and WORD fields ? ☞[SPPU : May-05, Dec.-10, Marks 10]

Ans. : i) Number of bits in main memory :

= Number of blocks × Number of words per block × Number of bits per word

= 4096 × 128 × 16

= 8388608 bits

ii) **Number of bits in word field :** There are 128 words in each block. Therefore, to identify each word (2^7 = 128) 7 bits are required.

iii) **Number of T bits in set field :** There are 64 blocks and each set consists of 4 blocks.

Therefore, there are 16 (64/4) sets. To identify each set (2^4 = 16) four bits are required.

iv) Number of bits in tag field : The total words in the memory are :

4096 × 128 = 524288.

To address these words we require (2^{19} = 524288) 19 address lines. Therefore, tag bits are eight (19 – 7 – 4).

Q.20 A direct mapped cache has the following parameters : cache size = 1 K words, Block size = 128 words and main memory size is 64 K words. Specify the number of bits in TAG, BLOCK and WORD in main memory address. ☞ [SPPU : May-10, Dec.-18, Marks 4]

Ans. : Word bits = $\log_2 128 = 7$-bits

$$\text{Number of blocks} = \frac{\text{Cache size}}{\text{Words in each block}} = \frac{1K}{128} = 8$$

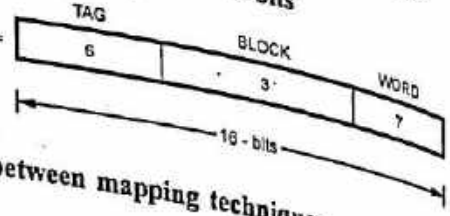
$\therefore$ Number of block bits = $\log_2 8 = 3$-bits

Number of address bits to address main memory

$= \log_2 64K = \log_2 2^{16} = 16$ - bits

$\therefore$ Tag bits = $16 - 3 - 7 = 6$-bits

Main memory address =



Q.21 Give comparison between mapping techniques.

Ans. :

Sr. No.	Direct-mapping	Associative-mapping	Set-associative-mapping
1.	Each block from the main memory has only one possible location in the cache.	A block of data from main memory can be placed into any cache block position.	A block of data from main memory can go into a particular block location of any direct-mapped cache.
2.	Needs only one comparison.	Needs comparison with all tag bits.	Needs number of comparisons equal to number of blocks per set.
3.	Cache hit ratio decreases if processor needs to access same memory location from two different pages of the main memory frequently.	Cache hit ratio has no effect if processor needs to access same memory location from two different pages of the main memory frequently.	The effect of reduction in cache hit ratio in case of frequent access to the two different pages of the main memory is reduced.
4.	Main memory address is divided into three fields : TAG, BLOCK and WORD.	Main memory address is divided into two fields : TAG and WORD.	Main memory address is divided into three fields : TAG, SET and WORD.
5.	Searching time is less.	Searching time is more.	Searching time increases with number of blocks per set.

Table Q.21.1 Comparison between mapping techniques

Q.22 Write a note on cache coherency.

Ans. :

- In a single CPU system, two copies of same data, one in cache memory and another in main memory may become different. This data inconsistency is called as **cache coherence problem.**
- Cache updating systems eliminates data inconsistency in the main memory caused by **cache write operations.**
- In multiprocessor systems, another bus master can take over control of the system bus. This bus master could write data into a main memory blocks which are already held in the cache of another processor. When this happens, the data in the cache no longer match those held in main memory creating inconsistency.
- There are four different approaches to prevent data inconsistency, that is to protect cache coherency :
  1. Bus watching (snooping)
  2. Hardware transparency
  3. Non-cacheable memory
  4. Cache flushing.

- **Bus watching :** In bus watching, cache controller invalidates the cache entry, if another master writes to a location in shared memory which also resides in the cache memory.
- **Hardware transparency :** In hardware transparency, accesses of all devices to the main memory are routed through the same cache or by copying all cache writes both to the main memory and to all other caches that share the same memory.
- **Non-cacheable memory :** The processor can partition its main memory into a cacheable and non-cacheable memory.. By designing shared memory as non-cacheable memory cache coherency can be maintained, since shared memory is never copied into cache.
- **Cache flushing :** To avoid data inconsistency, a cache flush writes any altered data to the main memory and caches in the system are flushed before a device writes to shared memory.

Q.23 Write a detail note on MESI protocal.

Ans. : • The MESI protocol makes it possible to maintain the coherence in cached systems. It is based on the four states that a block in the cache memory can have. These four states are the abbreviations for MESI : **modified, exclusive, shared and invalid.** States are explained below.

- **Modified :** The line (block) in the cache, has been modified, i.e. it is different from main memory is modified and this line (block) is available only in this cache.
- **Exclusive :** The line in the cache is same as that in main memory and it is not present in any other cache.
- **Shared :** The line (block) in the cache is same as that in main memory and the same line may be present in one or more other caches.
- **Invalid :** The line (block) in the cache does not contain valid data.

- The cache coherence mechanism receives requests from both the processor and the bus, and responds to these based on the type of request such as read/write and hit/miss in the cache, and the state of the cache block specified in the request.
- The Fig. Q.23.1 (a) shows the state diagram for MESI protocol.

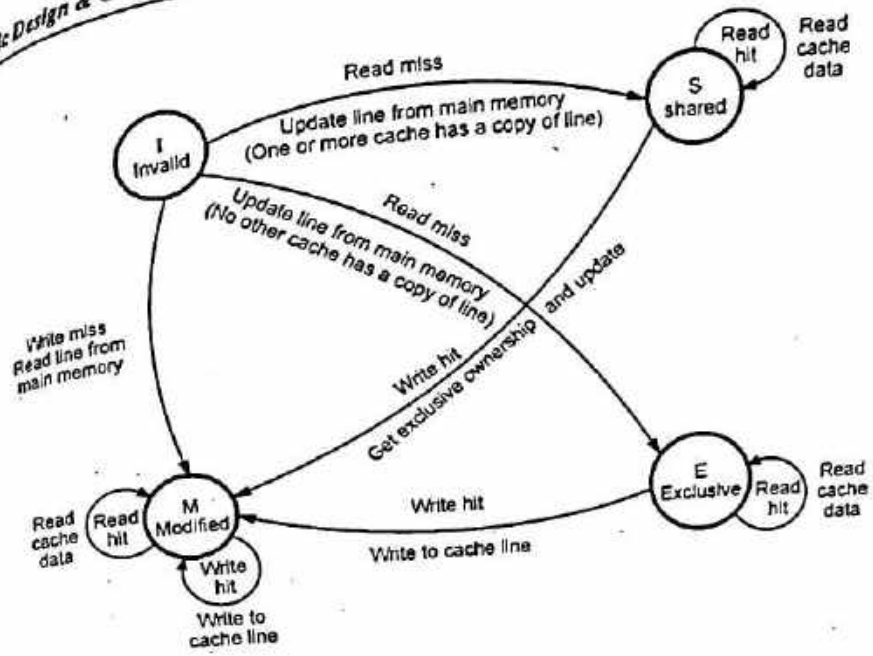


Fig. Q.23.1 (a) Transitions initiated by the processor

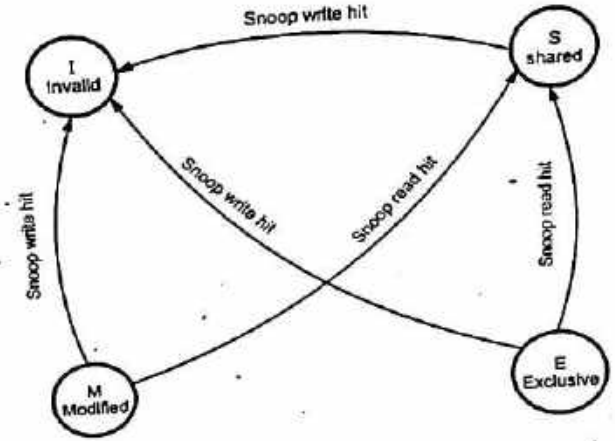


Fig. Q.23.1 (b) Transition initiated from common bus

- Each line (block) of cache has its own state bits.
- Fig. Q.23.1 (a) shows the transitions that occur due to actions initiated by the processor and Fig. Q.23.1 (b) shows the transitions that occur due to events that are snooped on the common bus.

Read Miss : • When a read miss occurs in the local cache, the processor reads the line of main memory which contains the missing address. The snoopy cache controller sends read miss command on the bus. This alerts all other processors to snoop the transaction.

Read Hit : • When a read hit occurs on a block which is currently in the local cache, the processor reads the block in its cache. There is no change in its state. The state remains modified, shared or exclusive.

Write Miss : • When a write miss occurs in the local cache, the processor gives a command on the bus to read the block of main memory containing the missing address. When the block is loaded, its status is marked as modified.

Write HIT : • When a write hit occurs on a block which is currently in the local cache, the action which is to be performed depends on the current state of that line in the local cache.

Q.24 List and explain write policies used with cache memory.

☞ [SPPU : May-12, Marks 4]

Ans. : • In a cache system, two copies of same data can exist at a time, one in cache and one in main memory. If one copy is altered and other is not, two different sets of data become associated with the same address. To prevent this, the cache system has updating systems such as : **write through system, buffered write through system** and **write-back system.**

Write through Systems : • The cache controller copies data to the main memory immediately after it is written to the cache. Due to this, main memory always contains a valid data and thus any block in the cache can be overwritten immediately without data loss.

- The write through is a **simple approach.**
- This approach **requires time to write data** in main memory with increase in bus traffic.
- This in effect **reduces the system performance.**

Buffered Write through System : • In buffered ... the processor can start a new cycle before the write cycle to the ... memory is completed. This means that the write accesses to the main memory are buffered.

- In such systems, read access which is a "cache hit" can be performed simultaneously when main memory is updated.
- However, two consecutive write operations to the main memory or read operation with cache "miss" require the processor to wait.

Write-Back System : • In a write-back system, the **alter** (update) bit in the tag field is used to keep information of the new data. If it is set, the controller copies the block to main memory before loading new data into the cache.

- Due to one time write operation, number of write cycles are reduced in write-back system. But this system has following disadvantages:
  - Write-back cache controller logic is more complex.
  - It is necessary that, all altered blocks must be written to the main memory before another device can access these blocks in main memory.
- In case of power failure, the data in the cache memory is lost, so there is no way to tell which locations of the main memory contain old data. Therefore, the main memory as well as cache must be considered volatile and provisions must be made to save the data in the cache.

Q.25 What are the different replacement algorithms ? Explain LRU algorithm in detail.

☞ [SPPU : Dec.-06, May-19, Marks 6]

OR Describe any cache replacement algorithm in short.

☞ [SPPU : Dec.-15, Marks 4]

Ans. : • When a new block is brought into the cache, one of the existing blocks must be replaced, by a new block.

- In case of direct-mapping cache, we know that each block from the main memory has only one possible location in the cache, hence there is no choice. The previous data is replaced by the data from the same memory location from new page of the main memory.
- For associative and set-associative techniques, there is a choice of replacing existing block. The choice of replacement of the existing block should be such that the probability of accessing same block must

be very less. The replacement algorithms do the task of selecting the existing block which must be replaced.

- There are four most common replacement algorithms :
  - Least-Recently-Used (LRU)
  - First-In-First-Out (FIFO)
  - Least-Frequently-Used (LFU)
  - Random
- **Least-Recently-Used :** In this technique, the block in the set which has been in the cache longest with no reference to it, is selected for the replacement. Since we assume that more-recently used memory locations are more likely to be referenced again. This technique can be easily implemented in the two-way set-associative cache organization.
- **First-In-First-Out :** This technique uses same concept that stack implementation uses in the microprocessors. In this technique, the block which is first loaded in the cache amongst the present blocks in the cache is selected for the replacement.
- **Least-Frequently-Used :** In this technique, the block in the set which has the fewest references is selected for the replacement.
- **Random :** Here, there is no specific criteria for replacement of any block. The existing blocks are replaced randomly. Simulation studies have proved that random replacement algorithm provides only slightly inferior performance to algorithms just discussed.

Q.26 Calculate the average access time of memory for a computer with cache access time of 100 ns, a main memory access of 1000 ns and a hit ratio is 0.9.

Ans. : The average access time is given by,

$t_A = t_{A1} + (1 - h)\, t_{A2} = 100 \text{ ns} + (1 - 0.9) \times 1000 \times 10^{-9} = 200 \text{ ns}$

Q.27 Suppose a cache is 10 times faster than main memory and suppose that the cache can be used 90 % of the time. How much speed up do we gain by using the cache ?

Ans. : Given : Hit ratio = 90 % = 0.9.

END...Z

Unit 6

14 Input / Output Systems

14.1 : I/O Module

Q.1 Why does I/O devices can not be connected directly to the system bus ?

OR What is I/O module ?

Ans. : • I/O devices (peripherals) cannot be connected directly to the system bus. The reasons are discussed here.

1. A variety of peripherals with different methods of operation are available. So it would be impractical to incorporate the necessary logic within the CPU to control a range of devices.
2. The data transfer rate of peripherals is often much slower than that of the memory or CPU. So it is impractical to use the high speed system bus to communicate directly with the peripherals.
3. Generally, the peripherals used in a computer system have different data formats and word lengths than that of CPU used in it.

- So to overcome all these difficulties, it is necessary to use a module in between system bus and peripherals, called **I/O module** or **I/O system** or **I/O interface.**

Q.2 State the functions performed by an I/O Interface.

Ans. : The functions performed by an I/O interface are :

1. Handle data transfer between much slower peripherals and CPU or memory.
2. Handle data transfer between CPU or memory and peripherals having different data formats and word lengths.
3. Match signal levels of different I/O protocols with computer signal levels.
4. Provides necessary driving capabilities - sinking and sourcing currents.

Ans. : difference between peripherals and CPU.

Sr. No.	Peripherals	CPU
1.	These are electro-mechanical and electromagnetic devices.	It is an electronic device.
2.	Data transfer rate is slower than that of the CPU.	Data transfer rate is faster than that of peripherals.
3.	Data is in form of codes.	Data is in word format.

Q.4 Draw and explain the block diagram of I/O module.

Ans. : Important blocks necessary in any I/O module are shown in Fig. Q.4.1.

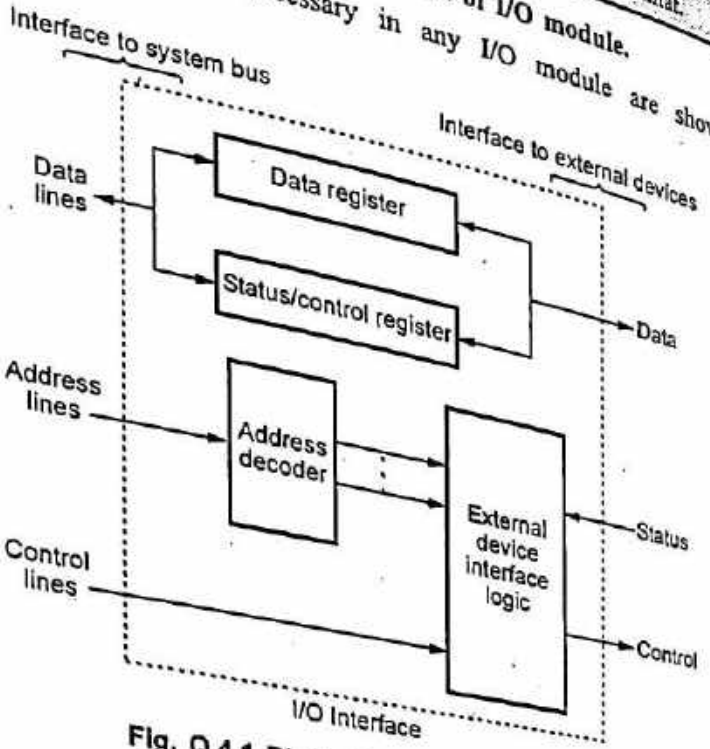


Fig. Q.4.1 Block diagram of I/O module

As shown in the Fig. Q.4.1, I/O module consists of data register, status/control register, address decoder and external device interface logic.

- The data register holds the data being transferred to or from the processor.
- The status/control register contains information relevant to the operation of the I/O device. Both data and status/control registers are connected to the data bus.
- Address lines drive the address decoder. The address decoder enables the device to recognize its address when address appears on the address lines.
- The external device interface logic accepts inputs from address decoder, processor control lines and status signal from the I/O device and generates control signals to control the direction and speed of data transfer between processor and I/O devices.

Q.5 Explain I/O interfacing techniques.

Ans. : • I/O devices can be interfaced to a computer system I/O in two ways, which are called interfacing techniques,

- Memory mapped I/O
- I/O mapped I/O

Memory mapped I/O :

- In this technique, the total memory address space is partitioned and part of this space is devoted to I/O addressing as shown in Fig. Q.5.1.
- When this technique is used, a memory reference instruction that causes data to be fetched from or stored at address specified, automatically becomes an I/O instruction if that address is made the address of an I/O port.

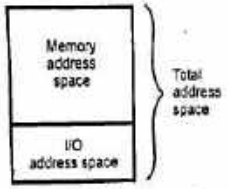


Fig. Q.5.1 Address space

Advantage : • The usual memory related instructions are used for I/O related operations. The special I/O instructions are not required.

Disadvantage : • The memory address space is reduced.

I/O mapped I/O : • If we do not want to reduce the memory address space, we allot a different I/O address space, apart from total memory space which is called I/O mapped I/O technique as shown in Fig. Q.5.2.

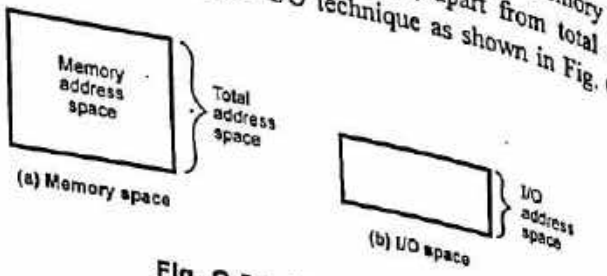


Fig. Q.5.2 Address space

Advantage : • The advantage is that the full memory address space is available.

Disadvantage : • The memory related instructions do not work. Therefore, processor can only use this mode if it has special instructions for I/O related operations such as I/O read, I/O write.

Q.6 Give comparison between memory mapped I/O and I/O mapped I/O.

Ans. :

☞ [SPPU : June-22, Marks 4]

Sr. No.	Memory mapped I/O	I/O mapped I/O
1.	Memory and I/O share the entire address range of processor.	Processor provides separate address range for memory and I/O devices.
2.	Usually, processor provides more address lines for accessing memory. Therefore more decoding is required for control signals.	Usually, processor provides less address lines for accessing I/O. Therefore, less decoding is required.
3.	Memory control signals are used to control read and write operations.	I/O control signals are used to control read and write I/O operations.

Q.7 Give comparison between memory and I/O bus.

Ans. :

Sr. No.	Memory bus	I/O bus
1.	Memory address bus shares entire address range.	I/O bus shares only I/O address range.
2.	Memory address bus width is greater than I/O address bus width	I/O address bus width is smaller than memory address bus width.
3.	Memory bus includes data bus, address bus and control signals to access memory.	I/O bus includes data bus, address bus and control signals to access I/O.

Q.8 State and explain different data transfer techniques.

☞ [SPPU : May-22, Marks 8]

Ans. : • In I/O data transfer, the system requires the transfer of data between external circuitry and the processor. Different ways of I/O data transfer are :

1. Program controlled I/O or polling control.
2. Interrupt program controlled I/O or interrupt driven I/O.
3. Hardware controlled I/O.
4. I/O controlled by handshake signals.

Program controlled I/O or polling control

- In program controlled I/O, the transfer of data is completely under the control of the processor program. This means that the data transfer takes place only when an I/O transfer instructions executed. In most of the cases it is necessary to check whether the device is ready for data transfer or not. To check this, processor polls the status bit associated with the I/O device.

Interrupt program controlled I/O or interrupt driven I/O

- In interrupt program controlled approach, when a peripheral is ready to transfer data, it sends an interrupt signal to the processor. This

indicates that the I/O data transfer is initiated by the external I/O device.

- When interrupted, the processor stops the execution of the program and transfers the program control to an interrupt service routine.
- This interrupt service routine performs the data transfer.
- After the data transfer, it returns control to the main program at the point it was interrupted.

Hardware controlled I/O

- To increase the speed of data transfer between processors memory and I/O, the hardware controlled I/O is used. It is commonly referred to as **Direct Memory Access (DMA).** The hardware which controls this data transfer is commonly known as **DMA controller.**
- The DMA controller sends a HOLD signal to the processor to initiate data transfer. In response to HOLD signal, processor releases its data, address and control buses to the DMA controller. Then the data transfer is controlled at high speed by the DMA controller without the intervention of the processor.
- After data transfer, DMA controller sends low on the HOLD pin, which gives the control of data, address, and control buses back to the processor.
- This type of data transfer is used for large data transfers.

I/O control by handshake signals

- The handshake signals are used to ensure the readiness of the I/O device and to synchronize the timing of the data transfer. In this data transfer, the status of handshaking signals are checked between the processor and an I/O device and when both are ready, the actual data is transferred.

14.2 : Programmed I/O and Interrupt Driven I/O

Q.9 What do you mean by modes of transfer ? Explain the following : i) Programmed I/O ii) Interrupt Initiated I/O.

☞ [SPPU : Dec.-05,09,12, May-08, June-16, Marks 7]

Q.9 Write short notes on : a) Programmed I/O b) Interrupt driven I/O.

Ans. : Programmed I/O

- I/O operations will mean a data transfer between an I/O device and memory or between an I/O device and the processor.
- If in any computer system I/O operations are completely controlled by the processor, then that system is said to be using **'programmed I/O'.**
- When such a technique is used, processor executes programs that initiate, direct and terminate the I/O operations, including sensing device status; sending a read or write command and transferring the data.
- It is the responsibility of the processor to periodically check the status of the I/O system until it finds that the operation is complete.

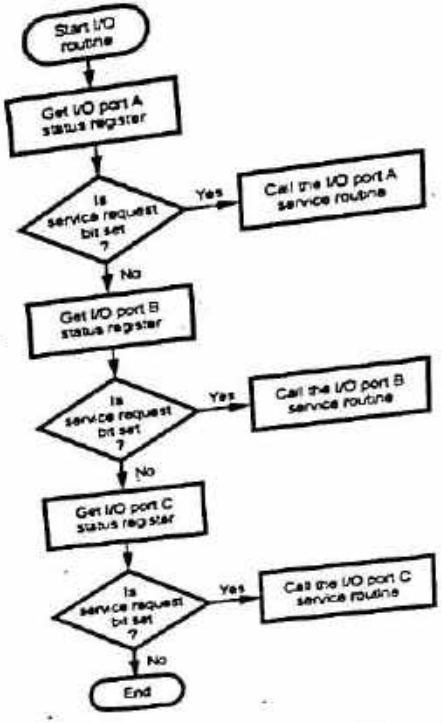


Fig. Q.9.1 Flowchart for I/O service routine

Interrupt-initiated I/O

- This method provides an external asynchronous input that would inform the processor that it should complete whatever instruction that is currently being executed and fetch a new routine (Interrupt Service Routine) that will service the requesting device.
- After this servicing is completed, the processor would resume exactly where it left off. The event that causes the interruption is called **interrupt** and the special routine executed to service the interrupt is called **Interrupt Service Routine (ISR).**

Q.10 What are hardware interrupt and software interrupts.

Ans. : • An interrupt caused by an external signal is referred as a **hardware interrupt.**

• Conditional interrupts or interrupts caused by special instructions are called **software interrupts.**

Q.11 What do you mean by maskable and non maskable interrupts ?

Ans. : • In the processor those interrupts which can be masked under software control are called **maskable interrupts.**

• The interrupts which cannot be masked under software control are called **non-maskable interrupts.**

Q.12 What do you mean by vector interrupt ? Explain.

Ans. : • When the external device interrupts the processor (interrupt request), processor has to execute interrupt service routine for servicing that interrupt.

• If the internal control circuit of the processor produces a CALL to a predetermined memory location which is the starting address of interrupt service routine, then that address is called **vector address** and such interrupts are called **vector interrupts.**

• For vector interrupts fastest and most flexible response is obtained since such an interrupt causes a direct hardware-iruplemented transition to the correct interrupt-handling program. This technique is called **vectoring.**

• When processor is interrupted, it reads the vector address and loads it into the PC.

• There are two ways to support vector interrupts :
  ▪ Fixed vector address,
  ▪ Programmable vector address.

• The processors which support fixed vector address approach have default vector address for each interrupt. The programmer cannot change this address.

• The processor which supports programmable vector address approach maintain the table in memory called the **interrupt vector table.**

• The Interrupt Vector Table (IVT) is an array of memory locations which holds the vector addresses of interrupts supported by the processor.

• When interrupt occurs the processor reads corresponding vector address given in the interrupt vector table and proceed for execution of interrupt service routine.

• The programmers are allowed to change the vector addresses stored in the interrupt vector table. Thus this approach is known as **programmable vector address.**

Q.13 What are nested interrupts.

Ans. : A system of interrupts that allows an interrupt service routine to be interrupted is known as **nested interrupts.**

Q.14 Compare programmed I/O and interrupt driven I/O.

Ans. :

Sr. No.	Programmed I/O	Interrupt driven I/O
1.	In programmed I/O, processor has to check each I/O device in sequence and in effect 'ask' each one if it needs communication with the processor. This checking is achieved by continuous polling cycle and processor cannot execute other instructions in sequence.	External asynchronous input is used to tell the processor that I/O device needs its service and hence processor does not have to check whether I/O device needs it service or not.
2.	During polling processor is busy and therefore, have serious and decremental effect on system throughput.	In interrupt driven I/O, the processor is allowed to execute its instructions in sequence and only stop to service I/O device when it is told to do so by the device itself. This increases system throughput.
3.	It is implemented without interrupt hardware support.	It is implemented using interrupt hardware support.

	depend on interrupt status.	s interrupt driven I/O.
5.	It does not need initialization of stack.	It needs initialization of stack.
6.	System throughput decreases as number of I/O devices connected in the system increases.	System throughput does not depend on number of I/O devices connected in the system.

Q.15 State the drawbacks of programmed I/O and interrupt driven I/O.

Ans. : 1.The I/O transfer rate is limited by the speed with which the CPU can test and service a device.

2. The time that the CPU spends testing I/O device status and executing a number of instructions for I/O data transfers can often be better spent on other processing tasks.

14.3 : Direct Memory Access (DMA)

Q.16 Explain the working of DMA controller.

☞ [SPPU : June-22, Marks 8]

Ans. :

- DMA controlled data transfer is used for large data transfer. For example to read bulk amount data from disk to main memory.
- To read a block of data from the disk processor sends a series of commands to the disk controller device telling it to search and read the desired block of data from the disk.
- When disk controller is ready to transfer first byte of data from disk, it sends DMA request DRQ signal to the DMA controller.
- Then DMA controller sends a hold request HRQ, signal to the processor HOLD input. The processor responds this HOLD signal by floating its buses and sending out a hold acknowledge signal HLDA, to the DMA controller.
- When the DMA controller receives the HLDA signal, it takes the control of system bus.
- When DMA controller gets control of the buses, it sends the memory address where the first byte of data from the disk is to be written. It

also sends a DMA acknowledge, DACK signal to the disk controller device telling it to get ready to output the byte.

- Finally, it asserts both the I/O read and memory write signals on the control bus. Asserting the I/O read signal enables the disk controller to output the byte of data from the disk on the data bus and asserting the memory write signal enables the addressed memory to accept data from the data bus. In this technique data is transferred directly from the disk controller to the memory location without passing through the processor or the DMA controller.
- Thus, the CPU is involved only at the beginning and end of the transfer.
- After completion of data transfer, the HOLD signal is deasserted to give control of all buses back to the processor.

Q.17 Give comparison between I/O program controlled transfer and DMA transfer.

Ans. :

Sr. No.	I/O program controlled transfer	DMA transfer
1.	It is software controlled data transfer.	Hardware controlled data transfer.
2.	Data transfer speed is low.	Data transfer speed is high.
3.	CPU is involved in the transfer.	CPU is not involved in the transfer.
4.	Extra hardware is not required.	DMA controller is required to carry-out data transfer.
5.	During data transfer data is routed through processor	During data transfer data does not routed through processor.

Q.18 Draw and explain the block diagram of typical DMA controller.

OR Write a short note on direct memory access (DMA).

OR What is direct memory access ? Explain. Give block diagram of circuitry required for direct memory access. ☞ [SPPU : June-22, Marks 8]

Ans. : DMA stands for Direct Memory Access. It is a hardware controlled data transfer. DMA controller is used to carry-out data transfer. During data transfer data is not routed through processor.

For performing the DMA operation, the basic blocks required in a DMA channel/controller are shown in Fig. Q.18.1.

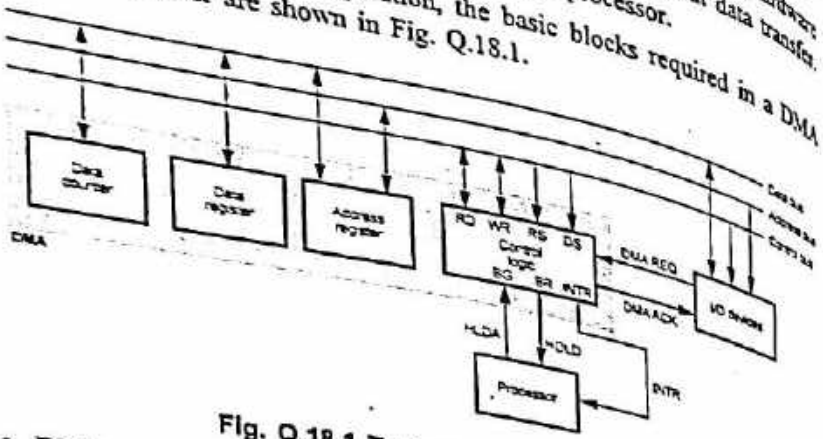


Fig. Q.18.1 Typical DMA block diagram

- DMA controller communicates with the CPU via the data bus and control lines.
- The registers in DMA are selected by the CPU through the address bus by enabling the DS (DMA select) and RS (Register select) inputs.
- The RD (Read) and WR (write) inputs are bidirectional.
- When the BG (bus grant) input is 0, the CPU can communicate with the DMA registers through the data bus to read from or write the DMA registers RD and WR signals are input signals for DMA.
- When BG = 1, the CPU has relinquished the buses and the DMA can communicate directly with the memory by specifying an address in the address bus and activating the RD or WR signals (RD and WR are now output signals for DMA). DMA consists of data count, data register, address register and control logic.
- Data counter register stores the number which gives the number data transfers to be done in one DMA cycle. It is automatically decremented after each word transfer.

- Data register acts as buffer whereas address register initially holds the starting address of the device. Actually, it stores the address of the next word to be transferred. It is automatically incremented or decremented after each word transfer.
- After each transfer, data counter is tested for zero. When the data count reaches zero, the DMA transfer halts.
- The DMA controller is normally provided with an interrupts capability, in which case it sends an interrupt to processor to signal the end of the I/O data transfer.

END...

SOLVED MODEL QUESTION PAPER (In Sem)

Logic Design and Computer Organization

S.E. (IT) Semester - III (As Per 2019 Pattern)

Time : 1 Hour] [Maximum Marks : 30

N.B. : *i) Attempt Q.1 or Q.2, Q.3 or Q.4.*

ii) Neat diagrams must be drawn wherever necessary.

iii) Figures to the right side indicate full marks.

iv) Assume suitable data, if necessary.

Q.1 a) *With neat circuit diagram explain the operation of two-input TTL NAND gates.* **(Refer Q.3 of Chapter - 1)** [4]

b) *Convert the following binary numbers to octal then to decimal. Show the steps of conversions.*

i) 11011100.101010 *ii) 01010011.010101* *iii) 10110011*

(Refer Q.13 of Chapter - 2) [6]

c) *Minimize the following function using K-map and implement using basic logic gates* $f(A,B,C,D) = \Sigma\, m(0,1,2,4,8,9,12,13) + d(3,6,7)$. **(Refer Q.18 of Chapter - 3)** [5]

OR

Q.2 a) *Explain with neat diagram two input CMOS NAND gate.* **(Refer Q.10 of Chapter - 1)** [5]

b) *Represent following number in single precision format :* $(0.625)_{10}$. **(Refer Q.63 of Chapter - 2)** [5]

c) *Find 1's complement of* $(11010100)_2$. **(Refer Q.20 of Chapter - 2)** [5]

Q.3 a) *Design a logic circuit to convert the 8421 BCD to Excess-3 code.* **(Refer Q.2 of Chapter - 4)** [6]

b) *What is half adder and full adder ?* **(Refer Q.5 of Chapter - 4)** [4]

c) *Explain the working of n-bit binary adder.* **(Refer Q.14 of Chapter - 4)** [5]

OR

Q.4 a) *Implement full adder using two half adders.* **(Refer Q.9 of Chapter - 4)** [4]

b) *Design and implement 8 : 1 MUX using two 4 : 1 MUX and implement given function* $F(X, Y, Z) = \sum m\,(1,3,4,7)$. **(Refer Q.10 of Chapter - 5)** [6]

c) *Design an 8-bit BCD adder using 4-bit binary adder.* **(Refer Q.31 of Chapter - 5)** [5]

END...

JUNE-2022 (END SEM) [5869] - 282

Course 2019 | Solved Paper

Time : $2\frac{1}{2}$ Hours] [Maximum Marks : 70

Instructions to the candidates :

1) Answer Q.1 or Q.2, Q.3 or Q.4, Q.5 or Q.6, Q.7 or Q.8.
2) Assume suitable data, if necessary.
3) Neat diagrams must be drawn wherever necessary.
4) Figures to the right indicate full marks.

Q.1 a) *Differentiate between combinational circuit and sequential circuit.* **(Refer Q.2 of Chapter - 6)** [9]

b) *Explain with diagram, how a D flip flop can be converted to T flip - flop.* **(Refer Q.19 of Chapter - 6)** [9]

OR

Q.2 a) *Define register. Draw and explain various types of shift register.* **(Refer Q.2 and Q.3 of Chapter - 7)** [9]

b) *Delineate modulus of counter. Design MOD - 81 counter using decade counter IC - 7490.* [9]

Ans. : Total number of counts or stable states a counter can indicate is called modulus.

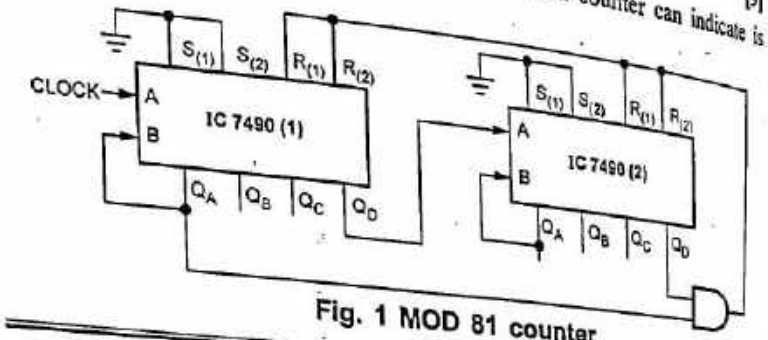


Fig. 1 MOD 81 counter

Q.3 a) *Explain following terms in brief.*
i) ALU signals ii) ALU functions iii) ALU types.
(Refer Q.4 of Chapter - 9) [9]

Ans. : ALU Types : Combinational ALU and sequential ALU

- The simplest combinational ALU combines the functions of a two's-complement adder-subtractor with the logic circuit that generates logic functions.
- The sequential ALU consists of one-word registers to store operands, parallel adder and logic circuit, flag register and control unit.

b) *Write in brief about fetch cycle with registers involved and sequence of micro instructions to carry out those operations.* **(Refer Q.9 of Chapter - 10)** [8]

OR

Q.4 a) *What is a register in CPU ? List various registers commonly used in CPU. Write short note on Flag register.* [9]

Ans. : Register is a storage place in the CPU. Registers are used to provide input to the ALU or they can be used to store the result of the ALU.

These register can be grouped as :

General purpose

- Data
- Address
- Condition codes

- **General-purpose registers :** These registers are used as simple storage area, mainly these are used to store intermediate results of the operation. They are also used for addressing functions (e.g. register indirect, displacement) to access operands. Getting the operand from the

general purpose registers is more faster than from memory so it is better to have sufficient number of general purpose register in the processor.

- **Data registers :** These registers are used only to hold data.
- **Address registers :** These registers are used for addressing purpose. They may be dedicated to a particular addressing mode.
  - **Segment pointers :** In a processor that supports segmented addressing, a segment register holds the address of the base of the segment.
  - **Index registers :** These are used for indexed addressing. There contents can be added to the address operand to give the effective address. Incrementing the index register then allows the program to access the next location in memory and so on, making it very useful for working with arrays or blocks of memory.
  - **Stack pointer :** A stack pointer is a register that stores the address of the last program request in a stack and its purpose is to keep track of a call stack.

Condition codes (Flag) register : Condition codes are bits set by the CPU hardware as the result of operations. For example, an arithmetic operation may produce a positive, negative, zero or overflow result. The condition code register is also referred to as status register. ALU operations and certain register operations may set or reset one or more bits in the **status register**. Status bits lead to a new set of CPU instructions.

Flag registers are also known as status registers. Status registers in most CPUs which store additional information about the results of machine instructions, e.g. comparisons. It usually consists of several independent flags such as carry, overflow and zero. It is chiefly used to determine the outcome of conditional branch instructions or other forms of conditional execution.

The contents of status register is also known as the Program Status Word (PSW). The PSW typically contains following condition codes or flags.

- **Sign :** It contains the sign bit of the result of the last arithmetic operation. Sign bit is one negative numbers and zero for positive numbers.
- **Zero :** It is set when the result is 0.
- **Carry :** It is set if an operation resulted in a carry (addition) into or borrow (sub-traction) out of a high-order bit. It is used for multiword arithmetic operations.
- **Equal :** It is set if a logical compare result is equality.
- **Overflow :** It is used to indicate arithmetic overflow.
- **Interrupt enable/disable :** It is used to enable or disable interrupts.
- **Supervisor :** It indicates whether the processor is executing in supervisor or user mode. In supervisory mode, processor is allowed to execute privileged instructions and access certain critical areas of memory.

b) *Explain and draw a basic structure of VON Neumann architecture. Write the difference between Harvard and Von Neumann architecture with diagrams.* **(Refer Q.13 and Q.19 of Chapter - 9)** [8]

Q.5 a) *What is a machine instruction ? Explain elements of machine instruction. Describe 0-1-2-3 address formats of machine instruction.* **(Refer Q.1, Q.2 and Q.9 of Chapter - 11)** [9]

b) *Draw and explain SISD, SIMD, MISD and MIMD architectures.* **(Refer Q.35 of Chapter - 12)** [9]

OR

Q.6 a) *What are the various data operands on which instruction operates ? Give examples of each data type.* **(Refer Q.14 of Chapter - 11)** [5]

b) *What is meant by multicore architecture ? What are its advantages ? List the typical features of multicore intel core i7.* **(Refer Q.62 and Q.66 of Chapter - 12)** [5]

Q.7 a) *Along with suitable diagram, explain fully associative cache mapping technique.* **(Refer Q.17 of Chapter - 13)** [5]

b) *Compare :* [5]

i) SRAM and DRAM. **(Refer Q.7 of Chapter - 13)**

ii) Memory mapped I/O and I/O mapped I/O.

(Refer Q.6 of Chapter - 14)

OR

Q.8 a) *Explain in brief the characteristics of memory system. Draw memory hierarchy. What is the objective of organizing different memories at the different hierarchy levels ?* **(Refer Q.1 and Q.2 of Chapter - 13)** [5]

b) *What is DMA ? Along with suitable diagram, explain how DMA is used for data transfer.* **(Refer Q.16 and Q.18 of Chapter - 14)** [5]

END...

Notes