TYPICAL QUESTIONS & ANSWERS

PART - I

OBJECTIVE TYPE QUESTIONS

Each Question carries 2 marks.

Choose correct or the best alternative in the following:

Q.1 In a virtual memory system, the addresses used by the programmer belongs to
(A) memory space. (B) physical addresses.
(C) address space. (D) main memory address.

Ans: C
An address used by programmers in a system supporting virtual memory concept is called virtual address and the set of such addresses are called address space.

Q.2 The method for updating the main memory as soon as a word is removed from the Cache is called
(A) Write-through (B) write-back
(C) protected write (D) cache-write

Ans: B
In this method only cache location is updated during write operation.

Q.3 A control character is sent at the beginning as well as at the end of each block in the synchronous-transmission in order to
(A) Synchronize the clock of transmitter and receiver.
(B) Supply information needed to separate the incoming bits into individual character.
(C) Detect the error in transmission and received system.
(D) Both (A) and (C).

Ans B
As the data are sent continuously as a block of data at the rate dictated by the clock frequency, so the receiver should be supplied with the same function about the same bit length in order to interrupt the information.

Q.4 In a non-vectored interrupt, the address of interrupt service routine is
(A) Obtained from interrupt address table.
(B) Supplied by the interrupting I/O device.
(C) Obtained through Vector address generator device.
(D) Assigned to a fixed memory location.

Ans: D
The source device that interrupted the processor supply the vector address which helps processor to find out the actual memory location where ISR is stored for the device.

Q.5 Divide overflow is generated when
(A) Sign of the dividend is different from that of divisor.
(B) Sign of the dividend is same as that of divisor.
(C) The first part of the dividend is smaller than the divisor.
(D) The first part of the dividend is greater than the divisor.

Ans: B
If the first part of the dividend is greater than the deviser, then the result should be of greater length, then that can be hold in a register of the system. The registers are of fixed length in
any processor.

Q.6 Which method is used for resolving data dependency conflict by the compiler itself?
(A) Delayed load. (B) operand forwarding.
(C) Pre fetch target instruction. (D) loop buffer.

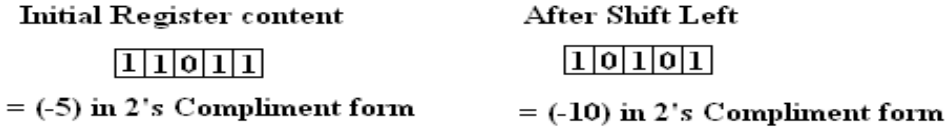
Ans: A
In case of delayed load technique the complier detects the data conflict and reorder the instruction as necessary to delay the loading of the conflicting data by inserting no operation instructions.

Q.7 Stack overflow causes
(A) Hardware interrupt.
(B) External interrupt.
(C) Internal interrupt.
(D) Software interrupt.

Ans: C
Stack overflow occurs while execution of a program due to logical faults. So it is a program dependent, hence interrupt activated.

Q.8 Arithmetic shift left operation
(A) Produces the same result as obtained with logical shift left operation.
(B) Causes the sign bit to remain always unchanged.
(C) Needs additional hardware to preserve the sign bit.
(D) Is not applicable for signed 2's complement representation.

Ans: A
If the register hold minus five in two's compliment form than in arithmetic shift left the contents of the register shall be



It is found that the register contents multiplied by two after logical shift left operation. Hence arithmetic shift left operation is same as logical shift operation.

Q.9 Zero address instruction format is used for
(A) RISC architecture.
(B) CISC architecture.
(C) Von-Neuman architecture.
(D) Stack-organized architecture.

Ans: D
In stack organized architecture push and pop instruction is needs a address field to specify the location of data for pushing into the stack and destination location during pop operation but for logic and arithmetic operation the instruction does not need any address field as it operates on the top two data available in the stack.

Q.10 Address symbol table is generated by the
(A) memory management software.
(B) assembler.
(C) match logic of associative memory.
(D) generated by operating system

Ans: B
During the first pass of assembler address symbol table is generated which contains the label used by the programmer and its actual address with reference to the stored program.

Q.11 The ASCII code for letter A is
(A) 1100011 (B) 1000001
(C) 1111111 (D) 0010011

Ans. (B)

Q.12 The simplified expression of $\overline{(A+B)} + \overline{C}$ is
(A) (A + B)C (B) A(B + C)
(C) (C+A + B) (D) None of these

Ans. (A)

Q.13 The negative numbers in the binary system can be represented by
(A) Sign magnitude (B) I's complement
(C) 2's complement (D) All of the above

Ans. (C)

Q.14 ABCD - seven segment decoder / driver in connected to an LED display. Which segments are illuminated for the input code DCBA = 0001.
(A)b, c (B) c, b
(C)a, b, c (D) a, b, c, d

Ans. (A)

Q.15 How many flip-flops are required to produce a divide-by-32 device?
(A)4 (B) 6

(C)5 (D) 7

Ans. (C)

Q.16 The content of a 4-bit register is initially 1101. The register is shifted 2 times to the right with the serial input being 1011101.
What is the content of the register after each shift?
(A)1110, 0111 (B) 0001, 1000
(C)1101, 1011 (D) 1001, 1001

Ans. (A)

Q.17 How many different addresses are required by the memory that contain 16K words?
(A)16,380 (B) 16,382
(C)16,384 (D) 16,386

Ans. (C)

Q.18 What is the bit storage capacity of a ROM with a 512' 4-organization?
(A) 2049 (B) 2048
(C) 2047 (D) 2046

Ans. (B)

Q.19 DMA interface unit eliminates the need to use CPU registers to transfer data from
(A) MAR to MBR (B) MBR to MAR
(C) I/O units to memory (D) Memory to I/O units

Ans. (D)

Q.20 How many 128 x 8 RAM chips are needed to provide a memory capacity of 2048 bytes?
(A) 8 (B) 16
(C) 24 (D) 32

Ans. (B)

Q.21 Which of the following is a self complementing code?
(A) 8421 code (B) 5211
(C) Gray code (D) Binary code

Ans. (A)

Q.22 Which gate can be used as anti-coincidence detector?
(A) X-NOR (B) NAND
(C) X-OR (D) NOR
Ans. (C)

Q.23 Which of the following technology can give high speed RAM?
(A) TTL (B) CMOS
(C) ECL (D) NMOS

Ans. (C)

Q.24 In 8085 microprocessor how many I/O devices can be interfaced in I/O mapped I/O technique?
(A) Either 256 input devices or 256 output devices.
(B) 256 I/O devices.
(C) 256 input devices & 256 output devices.
(D) 512 input-output devices.

Ans. (C)

Q.25 After reset, CPU begins execution of instruction from memory address
(A) 0101_H (B) 8000_H
(C) 0000_H (D) $FFFF_H$

Ans. (C)

Q.26 Which is true for a typical RISC architecture?
(A) Micro programmed control unit.
(B) Instruction takes multiple clock cycles.
(C) Have few registers in CPU.
(D) Emphasis on optimizing instruction pipelines.

Ans. (A)

Q.27 When an instruction is read from the memory, it is called
(A) Memory Read cycle (B) Fetch cycle
(C) Instruction cycle (D) Memory write cycle

Ans. (B)

Q.28 Which activity does not take place during execution cycle?
(A) ALU performs the arithmetic & logical operation.
(B) Effective address is calculated.
(C) Next instruction is fetched.
(D) Branch address is calculated & Branching conditions are checked.
Ans. (D)

Q.29 A circuit in which connections to both AND and OR arrays can be programmed is called
(A) RAM (B) ROM
(C) PAL (D) PLA
Ans. (A)

Q.30 If a register containing data $(11001100)_2$ is subjected to arithmetic shift left operation, then the content of the register after 'ashl' shall be

(A) $(11001100)_2$ (B) $(1101100)_2$

(C) $(10011001)_2$ (D) $(10011000)_2$

Ans. (D)

Q.31 Which logic is known as universal logic?

(A) PAL logic. (B) NAND logic.

(C) MUX logic. (D) Decoder logic.

Ans. (B)

Q.32 The time for which the D-input of a D-FF must not change after the clock is applied is known as

(A) Hold time. (B) Set-up time.

(C) Transition time. (D) Delay-time.

Ans. (A)

Q.33 How many memory chips of (128 x 8) are needed to provide a memory capacity of 4096 x 16?

(A)64 (B) A B

(C)32 (D) None

Ans. (A)

Q.34 In addition of two signed numbers, represented in 2' s complement form generates an overflow if

(A) A. B = 0 (B) A = 0

(C) $A \oplus B = 1$ (D) A + B = 1

Ans. (C)

Where A is the carry in to the sign bit position and B is the carry out of the Sign bit position.

Q.35 Addition of $(1111)_2$ to a 4 bit binary number 'a' results:-

(A) Incrementing A (B) Addition of $(F)_H$

(C) No change (D) Decrementing A

Ans. (C)

Q.36 In a microprocessor system, suppose. TRAP, HOLD, RESET Pin got activated at the same time, while the processor was executing some instructions, then it will first respond to

(A) TRAP (B) HOLD

(C) RESET (D) None

Ans. (D)

Q.37 Pseudo instructions are
(A) Machine instructions (B) Logical instructions
(C) Micro instructions (D) instructions to assembler.

Ans. (A)

Q.38 An attempt to access a location not owned by a Program is called
(A) Bus conflict (B) Address fault
(C) Page fault (D) Operating system fault

Ans. (B)

Q. 39 Dynamic RAM consumes _____ Power and _____ then the Static RAM.
(A) more, faster **(B)** more, slower
(C) less, slower **(D)** less, faster

Ans. (C)

Q.40 The flag register content after execution of following program by 8085 microprocessor shall be

Program
SUB A
MVI B, $(01)_H$
DCR B
HLT

(A) $(54)_H$ **(B)** $(44)_H$
(C) $(45)_H$ **(D)** $(55)_H$

Ans. (A)

Q.41 Which flag of the 8085's flag register is not accessible to programmer directly?
(A) Zero flag
(B) Carry flag
(C) Auxiliary carry flag
(D) Parity flag

Ans. (C)

Q.42 Cache memory works on the principle of
(A) Locality of data.
(B) Locality of reference
(C) Locality of memory
(D) Locality of reference & memory

Ans. (B)

Q.43 Which of the following is a Pseudo instruction?
(A) SPHL (B) LXI

(C) NOP (D) END

Ans. (D)

Q.44 A demultiplexer can be used as
(A)Encoder (B)Decoder
(C)Multiplexer (D)None of the above

Ans. (B)

Q.45 Excess-3 equivalent representation of $(1234)_H$ is
(A) $(1237)_{Ex-3}$ (B) $(4567)_{Ex-3}$
(C) $(7993)_{Ex-3}$ (D) $(4663)_{Ex-3}$

Ans. (B)

Q.46 Which of the memory holds the information when the Power Supply is switched off?
(A) Static RAM (B) Dynamic RAM
(C) EEROM (D) None of the above

Ans. (C)

Q.47 Minimum no. of NAND gate required to implement a Ex-OR function is
(A)2 (B)3
(C)4 (D)5

Ans. (C)

Q.48 Which of the following interrupt is maskable?
(A)INTR (B)RST 7.5
(C)TRAP (D)Both (A) and (B)

Ans. (B)

Q.49 Which of the following expression is not equivalent to x?
(A) x NAND x (B) x NOR x
(C) x NAND 1 (D) x NOR 1

Ans. (D)

Q.50 Word 20 contains 40
Word 30 contains 50
Word 40 contains 60
Word 50 contains 70
Which of the following instructions does not, load 60 into the Accumulator
(A) Load immediate 60
(B) Load direct 30
(C) Load indirect 20

(D) both (A) & (C)

Ans. (B)

Q.51 An interrupt for which hardware automatically transfers the program to a specific memory location is known as
(A) Software interrupt
(B) Hardware interrupt
(C) Maskable interrupt
(D) Vector interrupt

Ans. (B)

Q.52 Synchronous means _____
(A) At irregular intervals
(B) At same time
(C) At variable time
(D) None of these

Ans. (B)

Q.53 'n' Flip flops will divide the clock frequency by a factor of

(A) n^2 (B) n

(C) 2^n (D) log (n)

Ans. (B)

Q.54 In DMA the data transfer is controlled by

(A) Microprocessor (B) RAM

(C) Memory (D) I/O devices

Ans. (D)

Q.55 The number of instructions needed to add a numbers an store the result in memory using only one address instruction is

(A) n (B) n - 1

(C) n +1 (D) Independent of n

Ans. (D)

Q.56 Negative numbers cannot be represented in
(A) Signed magnitude form
(B) I's complement form
(C) 2's complement form
(D) 8-4-2-1 code

Ans. (C)

Q.57 Which of the following architecture is/are not suitable for realizing SIMD

(A) Vector Processor (B) Array Processor

(C) Von Neumann (D) All of the above

Ans. (C)

Q.58 In Boolean expression A+BC equals
(A) (A+B)(A+C) (B) (A'+B)(A'+C)
(C) (A+B)(A'+C) (D) (A+B)C

Ans. (A)

Q.59 A JK flip-flop can be implemented using D flip-flop connected such that
(A) $D=J\overline{Q}+\overline{K}Q$ (B) $D=\overline{J}Q+K\overline{Q}$
(C) $D=\overline{J}\overline{Q}+KQ$ (D) $D=J\overline{Q}+K\overline{Q}$

Ans. (A)

Q.60 An effective solution to the power consumption problem lies in using _____ transistors to implement ICs.
(A) NMOS (B) TTL shottky
(C) PMOS (D) both NMOS & PMOS

Ans. (D)

Q.61 Memory interleaving technique is used to address the memory modules in order to have
(A) higher average utilization
(B) faster access to a block of data
(C) reduced complexity in mapping hardware
(D) both (A) & (B)

Ans. (C)

Q.62 In a multiprogramming system, which of the following is used
(A) Data parallelism (B) Paging concept
(C) L1 cache (D) None of the above

Ans. (B)

Q.63 Cycle stealing technique is used in
(A) Interrupt based data transfer
(B) Polled mode data transfer
(C) DMA based data transfer
(D) None of these

Ans. (C)

Q.64 Manipulation of individual bits of a word is often referred to as
(A) Bit twidding (B) Bit swapping
(C) Micro-operation (D) None of these

Ans. (A)

Q.65 Which of the following is not a characteristic of a RISC architecture.

(A) Large instruction set (B) One instruction per cycle
(C) Simple addressing modes (D) Register-to-register operation

Ans. (A)

Q.66 When CPU is not fully loaded, which of the following method of data transfer is preferred

(A) DMA (B) Interrupt
(C) Polling (D) None of these

Ans. (D)

Q.67 Associative memory is some times called as

(A) Virtual memory (B) Cache memory
(C) Main memory (D) Content addressable memory

Ans. (D)

Q.68 BCD equivalent of Two's complement is

(A) nine's complement (B) ten's complement
(C) one's complement+1 (D) none of these
Ans. (C)

Q.69 PAL circuit consists of

(A) Fixed OR & programmable AND logic
(B) Programmable OR & Fixed AND Logic
(C) Fixed OR & fixed AND logic
(D) Programmable OR & programmable AND logic
Ans. (A)

Q.70 8085 microprocessor carryout the subtraction by

(A) BCD subtraction method
(B) Hexadecimal subtraction method
(C) 2's complement method
(D) Floating Point subtraction method
Ans. (C)

Q.71 CPU checks for an interrupt signal during

(A) Starting of last Machine cycle
(B) Last T-State of instruction cycle
(C) First T-State of interrupt cycle
(D) Fetch cycle

Ans. (B)

Q.72 During DMA acknowledgement cycle, CPU relinquishes

(A) Address bus only **(B)** Address bus & control bus
(C) Control bus & data bus **(D)** Data bus & address bus

Ans. (D)

Q.73 If the clock input applied to a cascaded Mod-6 & Mod-4 counter is 48KHz. Than the output of the cascaded arrangement shall be of
(A) 4.8 KHz **(B)** 12 KHz
(C) 2 KHz **(D)** 8 KHz

Ans.(C)

Q.74 If the stack pointer is initialised with $(4FEB)_H$, then after execution of Push operation in 8085 microprocessor, the Stack Pointer shall be
(A) 4FEA **(B)** 4FEC
(C) 4FE9 **(D)** 4FED

Ans. (D)

Q.75 A more efficient way to organise a Page Table is by means of an associative memory having
(A) Number of words equal to number of pages
(B) Number of words more than the number of pages
(C) Number of words less than the number of pages
(D) Any of the above

Ans. (A)

Q.76 If there are four ROM ICs of 8K and two RAM ICs of 4K words, than the address range of Ist RAM is (Assume initial addresses correspond to ROMs)
(A) $(8000)_H$ to $(9FFF)_H$ **(B)** $(6000)_H$ to $(7FFF)_H$
(C) $(8000)_H$ to $(8FFF)_H$ **(D)** $(9000)_H$ to $(9FFF)_H$

Ans. (C)

Q.77 $A \oplus B \oplus C$ is equal to $A \odot B \odot C$ for
(A) A=0, B=1, C=0 **(B)** A=1, B=0, C=1
(C) A=1, B=1, C=1 **(D)** All of the above

Ans. (D)

Q.78 Gray code equivalent of $(1000)_2$ is
(A) $(1111)_G$ **(B)** $(1100)_G$
(C) $(1000)_G$ **(D)** None of these

Ans. (A)

PART- II

DESCRIPTIVES

Q.1 Use K-maps to find the simplest Sum of Products (SOP) form of the function
F = f . g, where

$$f = wx\bar{y} + \bar{y}z + \bar{w}y\bar{z} + \bar{x}y\bar{z}$$

$$g = (w + x + y' + z')(x' + y' + z) \quad (7)$$

Ans:

$$f = wx\bar{y} + \bar{y}z + \bar{w}y\bar{z} + \bar{x}y\bar{z}$$

$$= wxy'(z + z') + (x + x')y'z(w + w') + w'(x + x')\ yz' + (w + w')x'yz'$$

$$= wxy'z + wxy'z' + wxy'z + w'xy'z + wx'y'z + w'x'y'z + w'xyz' + wx'y'z$$

$$wx'yz' + w'x'yz'$$

$$= m13 + m12 + m13 + m5 + m9 + m1 + m6 + m12 + m10 + m2$$

$$= m1 + m2 + m5 + m6 + m9 + m10 + m12 + m13$$

$$now, g = (w + x + y' + z')(x' + y' + z)\ (w' + y + z')$$

$$= (w + x + y' + z')(w' + x' + y' + z)\ (w + x' + y' + z)(w' + x' + y' + z)\ (w' + x + y + z')$$

$$= M3M14M6M13M9$$

The K map for f and g are given below

f

wx \ yz	y'z'	y'z	yz	yz'
w'x'	0	1	0	1
w'x	0	1	0	1
wx	1	1	0	0
wx'	0	1	0	1

g

wx \ yz	y'z'	y'z	yz	yz'
w'x'	1	1	0	1
w'x	1	1	1	0
wx	1	0	1	0
wx'	1	0	1	1

Therefore the K-map for F=fg is given below

wx \ yz	y'z'	y'z	yz	yz'
w'x'	0	1	0	1
w'x	0	1	0	0
wx	1	0	0	0
wx'	0	0	0	1

F= m1+m2+m5+m10+m12

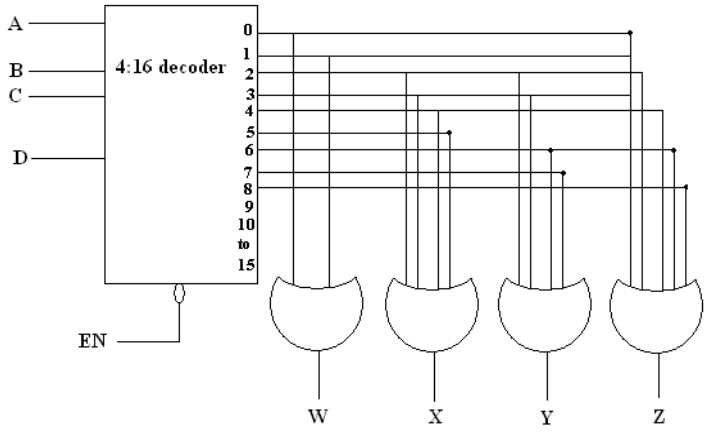
Simplification of above K-map gives-
F = w'y'z + x'yz' + wxy'z'

Q.2 Design a combinational circuit that generates 9's complement of a BCD digit. (6)

Ans:

Decimal No.	BCD input				output			
	A	B	C	D	W	X	Y	Z
0	0	0	0	0	1	0	0	1
1	0	0	0	1	1	0	0	0
2	0	0	1	0	0	1	1	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	0	1
5	0	1	0	1	0	1	0	0
6	0	1	1	0	0	0	1	1
7	0	1	1	1	0	0	1	0
8	1	0	0	0	0	0	0	1
9	1	0	0	1	0	0	0	0

w=m0+m1
x=m2+m3+m4+m5
y=m2+m3+m6+m7
z=m0+m2+m4+m6+m8



Q.3 Show that the dual of EX-OR is also its complement. (3)

Ans:
Logic equation for EX-OR operation is: $A \oplus B = A'B + AB'$
dual of (A ⊕ B)=dual of [A'B+AB']
=(A+B')(A'+B)
=A.A'+A.B+A'.B'+B.B'
=AB+A'B'=(A ⊕ B)
Hence dual of EX-OR is also its complement.

Q.4 Explain with the help of an example, the use of hamming code as error detection and correction code. (7)

Ans:

Hamming code is generated by adding k- parity bits to n - bit data word, forming the new word of (n+k) bits. The bit positions are numbered from l to (n + k) from left to right. Those positions numbered as a power of 2 are reserved for the parity bit. The remaining bits are data bits.

The relation between k & n are as:-

$2^k - 1 - k \geq n$,

If n = 4 then k = 3 and for n = 8 then k =4

Let's consider 8 bit data word 11000100, for which parity generation, error detection and correction capability of Hamming code shall be discussed. For n = 8, k = 4, therefore n+k =12

Bit position:

1	2	3	4	5	6	7	8	9	10	11	12
p1	p2	1	p3	1	0	0	p4	0	1	0	0

The 4 parity bits, p1, p2 p3 and p4 are in position 1, 2, 4 and 8 respectively. The 8-bit data word is in remaining positions. Each parity bit is calculated as follows:-

P1 = XOR of bits (3. 5, 7, 9, 11) = 1+1+0+0+0=0

P2 = XOR of bits (3, 6, 7, 10, 11) =1 + 0 + 0 + 1 + 0 = 0

P4 = XOR of bits (5, 6, 7, 12) = 1 + 0 + 0 + 0 = 1

PS = XOR of bits (9, 10, 11, 12) = 0 +1 + 0 + 0 = 1

Therefore the code generated is : - 0 0 1 1 1 0 0 1 0 1 0 0

Bit position 1 2 3 4 5 6 7 8 9 10 11 12

This new code is transmitted and at receiver side parity is checked over the same combination including parity bit. the four check bits are so generated as follows -

Cl = XOR of bits {l, 3, 5, 7, 9, 11}

C2 = XOR of bits {2, 3, 6, 7, 10, 11}

C4 = XOR of bits {4, 5, 6, 7, 12)

C8 = XOR of bits (8, 9, 10, 11, 12)

If the result C = C8 C4 C2 C 1 = 0000, it indicates there is no error in received data. However, if C is not equal to zero, then the binary number formed by the check bits C8 C4 C2 C1 gives the position of the erroneous bit. Consider the following three cases for error detection:-

Received data

Bit Position	1 2 3 4 5 6 7 8 9 10 11 12	C8	C4	C2	C1	Remarks
A	0 0 1 1 1 0 0 1 0 1 0 0	0	0	0	0	No error
B	1 0 1 1 1 0 0 1 0 1 0 1	0	0	0	1	Error in bit position 1
C	0 0 1 1 0 0 0 1 0 1 0 0	0	1	0	1	Error in bit position 5

The error can be corrected by complementing the bit in the position as dictated by C8 C4 C2 C1.

Hence, in this way hamming code can detect and correct one bit error only.

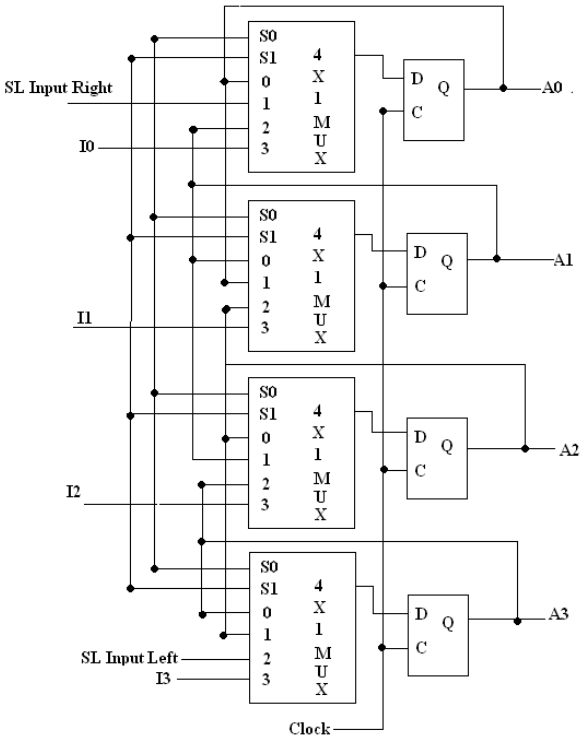
Q.5 With the help of a neat sketch, explain the working of a 4-bit universal shift register. (6)

Ans:

A register that can shift the data in both directions and has the capacity of parallel load is called Universal shift register. All the possible operation lo the register can be made with it. It can work as serial in parallel out, serial in serial out, parallel in parallel out and parallel in serial out fashion.

A 4- bit universal shift register is shown below:-

Each stage consists of a D- FF and a 4x1 Multiplexer. The two select in puts SO, Sl select one of the multiplexer data input for the DFF's. The selection lines control the mode of operation of the register according to the function table given below



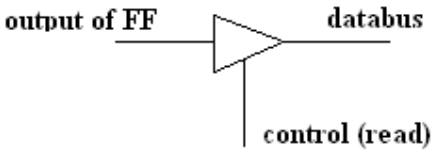
Function Table

Mode control		Register Operation
S1	SO	
0	0	No Change '-
0	1	Shift Right
1	0	Shift Left
1	1	Parallel Load

For Sl SO = 00, data input 0 of each multiplexer is selected. This condition forms a path from the output of each FF into the input of the same FF. The clock pulse transfers the binary value it held previously and no change of state occurs.

When S 1 SO = 01, this cases a shift right operation. The data available at serial input (Shift Right) enters into the First FF and the output of one FF is transferred to the other next to it causing the data stored, in the register to shift right by one bit with

each clock pulse. If we want that the data stored in register should rotate right, then the output A3 can be connected to the serial input (Shift Right).
When Sl SO = 10, shift left operation takes place. Here also in order to rotate the data in left direction by one bit with each clock pulse, the output AO can be connected to serial input (Shift Left) terminal.
When Sl SO = 11, the data available on the line 10, 11, 12, 13 get loaded in to the register with one clock pulse.
Further, the output of the register can also be available at AO, Al, A2 and A3 lines. These lines can be connected to with an tri- state buffer so as to read the data of the register only when the control input is 1.



Q.6 State the condition in which overflow occurs in case of addition & subtraction of two signed 2's complement number. How is it detected? (3)

Ans:
Overflow may occur when two n-bits number of same sign are added or when two n-bit I the time of the numbers of different sign are subtracted. If the result of addition / Subtraction is of (n + l) bit or out of permissible range than it is said to be overflow. For Example:-
+70-----01000110
+80----- 01010000
+150----- 10010110
Here the result of addition is in 8-bit only, but as +150 is out of the range that a number can be represented by 8-bit signed 2's Compliment form. So, overflow has occurred giving the wrong result. Now

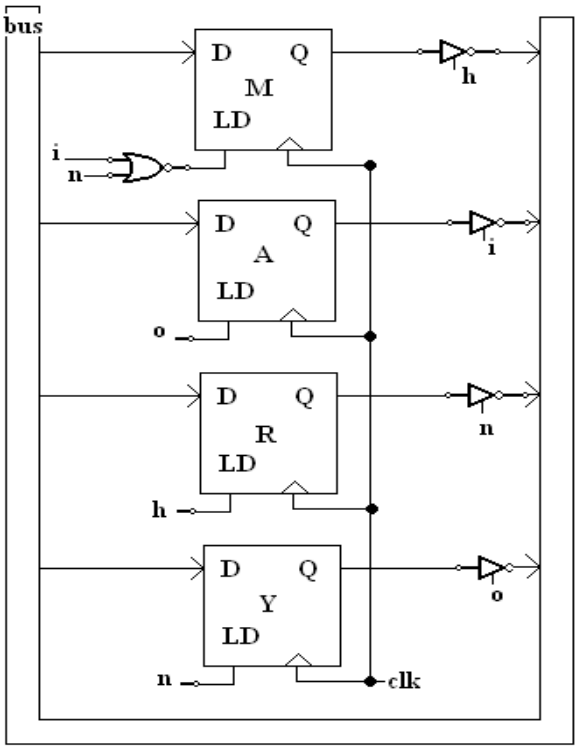
-70 2's 10111010
-————-»
-80 2's . 10110000
———>>
-150 101101010

Here the result of nine bits, clearly shows the over flow, as the largest negative number that can be represented by 8 signed 2'compliments number is -128 only.
An overflow condition can be detected by observing the carry in to the sign bit position and the carry out of sign bit position. If these two carries are not equal, then over flow is produced. If these two carries applied to an XOR gate, an overflow will be detected when the output of the gate is equal to 1.

Q.7 Implement the following RTL code, using common bus and tri-state buffers.
J : M←A
o: A←Y
b: R← M

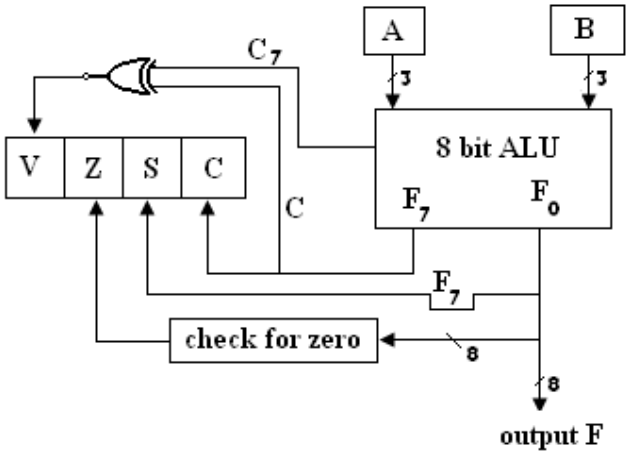
n: Y ←R, M ←R
Assume M, A, R and Y are to be one bit D – flip- flop. (6)

Ans:
The hardware realisation of R.TL behavior is shown below by using common bus and tri state buffer. All the FFs get the same clock pulse. Depending on the control signal of tri state buffer, the source FF is selected, and depending on control signals connected to 'LD' of FF's the destination FF is selected.



Q.8 What do you mean by program control instructions? With a neat diagram, explain how the status register containing overflow, zero, sign and carry flags works with the status of the accumulator content obtained from ALU. (3+4)

Ans:
Program control type instructions, when executed by the processor, may change the address value of the Program Counter and cause type flow of control to be altered. Program control instruction specifies conditions for altering the content of Program Counter. This causes break in the sequence of instruction execution. This Instruction also gives the capability for branching to different Program segments.
Examples - Branch., Jump, Skip, Call, Return etc.
Status bits are set or reset depending on the result of a logical or arithmetic manipulation of accumulator data. So status bits are called condition - code bits or flag bits. These status bits constitute status register.
The hardware realization of status register containing overflow, Zero, Sign, Carry flag is shown below –



(i) Bit 'c' (carry) is *set* to 1. if the end carry c8 is 1. It is cleared to zero if the carry is 0
(ii) Bit's' (sign) is set to one if' the highest order bit F7 is 1. It is set zero, if thebitF7=0
(iii) Bit 'z'(zero) is set to 1. if the output of 'ALU' contains all zeros. Otherwise it is set to zero.
(iv) Bit 'v' (overflow) is set to 1, if the Ex – OR of the last two carries i.e. c7 & c8 is equal to 1, and cleared to 0 otherwise. This is the condition for an overflow when negative numbers are in 2's complement form.

Q.9 What are interrupts? Explain different types of interrupts. (6)

Ans:
In a microprocessor system, there are three major types of interrupt that cause a break in the normal executing of a program. These are -
(i) External Interrupt: interrupt signal came from input-output devices connected external to processor. These interrupts depend on external conditions that are independent of the program being executed at the time. The examples that causes external Interrupt are - I/0 device requesting transfer of data, elapsed time of an event, power failure. time out mechanism for a program etc.
(ii) Internal Interrupt: Cause due to illegal or erroneous use of an instruction or data. Internal interrupts are also called traps. Internal interrupts are initiated due to some exceptional condition caused by the program itself rather than by an external event. If the program is rerun, the internal interrupts will occur in the same place each time. Example of cause of internal interrupts are - attempt to divide by zero, stack overflow, Invalid opcode, protection violation etc.
(iii) Software interrupts: It is initiated by executing an instruction. These are special call instructions that behaves like an interrupt rather than subroutine call. These can be used by the programmer to initiate an interrupt procedure at any designed point of the program. These interrupts arc usually used for switching to supervisor made from user mode.

Q.10 How stack is implemented in a general microprocessor system. (3)

Ans:

In a general microprocessor system, there is a special register known as stack pointer, which holds the address of the top of the stack. In some microprocessor, register stack is provided. In order to indicate the stack full condition and stack empty condition, two flags are used. These two flags are known as EMPTY flag & FULL flag. The empty flag is set when the stack is completely empty. Full flag is set only when all the stack locations are filled with data. Stack is essential for implementing subroutine call and interrupts.

Stacks operate in two principles

(1) LIFO i.e. Last in First Out

(2) FIFO i.e. first in first out.

These principles of operation depends on stack architecture. Most of general purpose processor use LIFO principle for their stack.

If the stack is organized in R/W memory, than the stack pointer is loaded with same address to initialize. The memory stack grow down word i.e. with each Push operations, stack pointer is decremented. The situation is just reverse on register stack.

Q.11 What are the advantages of assembly language? How is it different from high-level language? (6)

Ans:

Writing program for a computer consists of specifying, directly or indirectly, a sequence of machine instructions- The machine instruction stored in RAM of the computer is in binary format. This binary format is very difficult to use and to – troubleshoot. So programs are written by user by using English like symbols of the alpha-numeric character set, which is known as assemble language. The assembler converts these assembly language programs to binary form.

Advantages of assemble language program is it is easy to use. It is easy to troubleshoot, it is fast to execute than high level language program.

A programming language is defined by a set of rules. Users must conform to all format rules of the assembly language if it is to be translated correctly. Each microprocessor has its own assembly language format. The assembly language use predefined rules that specify the symbols that can be used & how they may be combined to form a line of code.

Some of the common rules are

(i) The label field may be empty or it may specify a symbolic address.

(ii) The instruction field specify a machine instruction or a Pseudo instructions.

(iii) The comment field may be empty or it may include a comment.

(iv) The symbolic address consists of up to four alphanumeric characters.

(v) Symbolic address in the label field is terminated by a comma so that it will be recognized as a label by the assembler.

(vi) The comment field is preceded by a slash foe assembler to recognize the beginning of a comment field.

Q.12 What is vertical micro code? State the design strategy of a vertical micro coded control unit. (6)

Ans:

In vertical microcode, the micro-operators are grouped in to fields. Each micro-operation is assigned a unique encoded value in this field. For example, 16 micro operations could be encoded using four bits, with each microoperation assigned with a unique binary field value from 0000 to 1111, These 16 micro-operation also include the 'NOP" i.e. No operation. Vertical micro instructions require fewer bits than their equivalent horizontal micro instructions, however the micro sequencer incorporate a decoder for each microoperation field to generate the actual micro-operation signals.

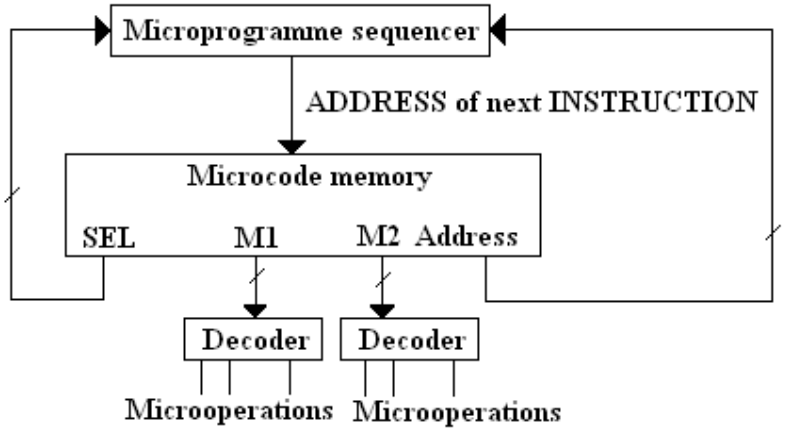
The design strategy used for vertical microcode is as follows:

Whenever two microoperations occur during the same state, assign them to different fields.

(i) Include NOP in each field if necessary.

(ii) Distribute the remaining micro-operations to make the best use of the micro-operation field bits.

(iii) Group together micro-operation that modify the same registers in the same field.



Q.13 What is a microprogram sequencer? With block diagram, explain the working of microprogram sequencer. (2+8)

Ans:

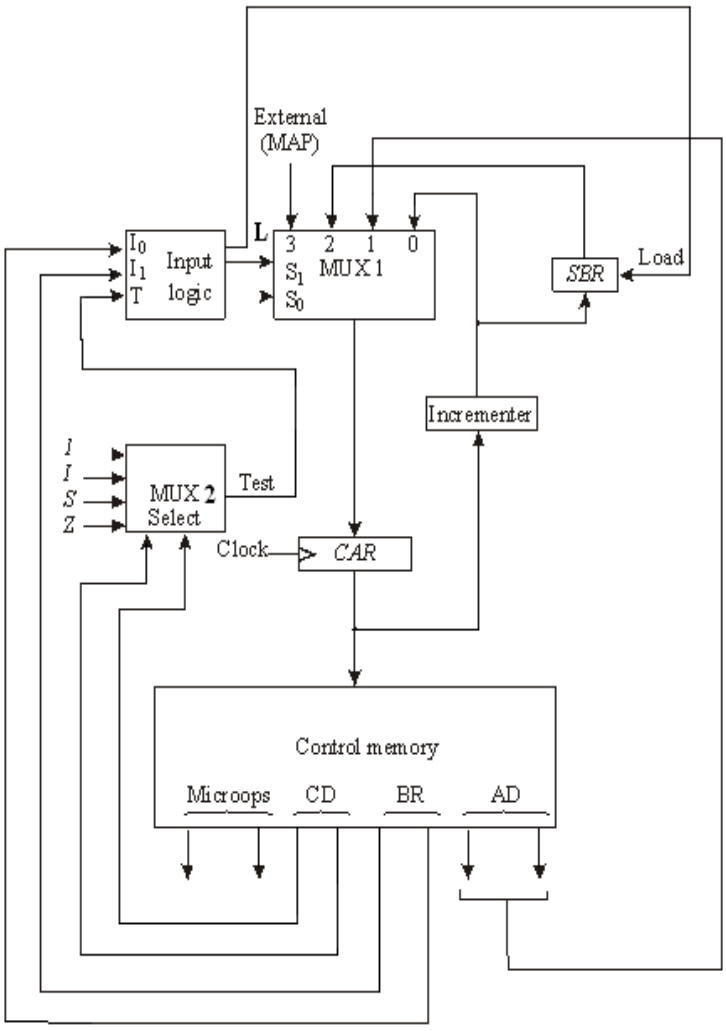
The function of control unit in a digital computer is to initiate sequences of micro-operations. When the control signals are generated by hardware using conventional logic design techniques, the control unit is said to be hard wired. Microprogramming is the second alterative. The control function that specifies microoperation is a binary variable. These binary control variables are stored in memory is called a microprogrammed control unit. A sequence of microinstructions constitutes a microprogram. Each machine instruction initiates a search of micro instruction in control memory. These microinstructions generates the microoperations to fetch the instruction from main memory, to evaluate the effective address, to execute the operation specified by the instruction and to return control to the fetch phase, in order to repeat the cycle for new instruction. Control memory address register specifies the address of the microinstruction to be read from memory. The micro instruction contains a control word that specifics one or more micro operations for the data processes. Once these operations are executed, the

control must determine the new address. The location of the next microinstruction may be the one next in sequence or it may be located somewhere else in the control memory. For this reason it is necessary to use some bits of the present micro instruction to control the generation of the address of the next micro instruction. The next address may also be a function of external input condition. The next address is computed by the circuit is called microprogram sequencer. The typical functions of a micro program sequencer are

(i) Incrementing the control address registers by one.
(ii) Loading into the control address register an address from control memory
(iii) Transferring an external address loading an initial address to start control operations.

The sequencer should also have a facility for subroutine call and return. This is shown in the following diagram:-

Fig. Microprogram sequencer for a control memory

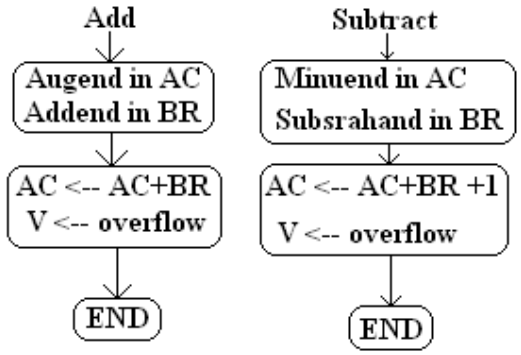


Q.14 Give the flow chart for add and subtract operation of two signed 2's complement data. Explain the logic of each operation. (4+6)

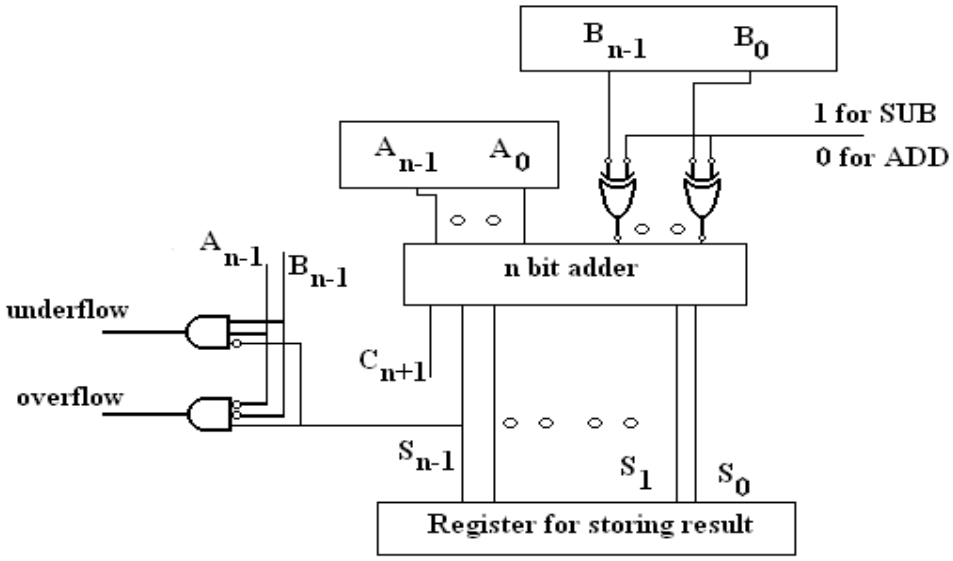
Ans:

In signed 2's complement representation, the left most bit of a binary number represents the sign bit. '0' for + ve & 1 for – ve. If the sign bit is 1, the entire number is represented in 2's complement form.

Flow chart:



Addition: - both the operands are added up along with the sign bit. A carry out of the sign bit position is discarded.
Subtraction:-Take 2's complement form of the subtracted including the sign bit and add it to the minuend including the sign bit. A carry out of the sign bit position is discarded.



Example: If we want to carry out -35 - (+40) in signed 2's complement representation then ,the binary representation of + 35 = 010001 and + 40 = 0101000 In signed 2's complement representation; both the operands are represented as

$-35 \rightarrow 0010011 \xrightarrow{2'\text{scompl}} 11011101$
$+40 \rightarrow 00101000 \xrightarrow{2'\text{scompl}} 11011000$
As we have to subtract (+40) from-35

So the result of substraction = Minuend in signed 2's complement from plus 2's | complement of subtrahend.

-35-(+40) = 11011101
11011000
10110101

The overflow carry is neglected. So the answer is (10110101) which is in signed 2's complement from, which is equal to (-75).

Q.15 Explain different methods used for establishing the priority of simultaneous interrupts. (6)

Ans:

To establish the priority of simultaneous interrupts can be done by software or hardware. Polling procedure is a software method. It is used for identifying the highest-priority source by executing a program. In this method there is one common branch address for all interrupts. The programme polls the interrupt sources in sequence. The order in which the sources are polled determines the priority of each interrupt. Thus the initial service routine for all interrupts consist of a program that tests the interrupts sources in sequence and to branch to one of many possible service routines.

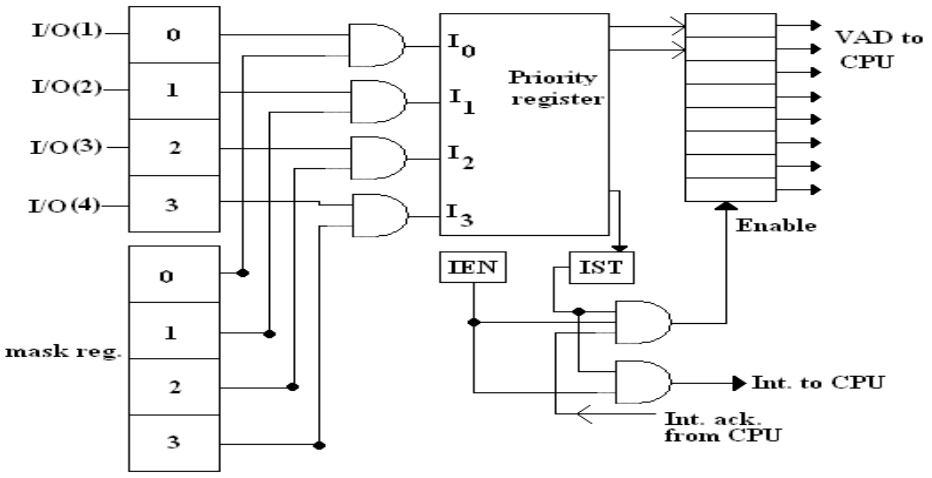
There are two hardware methods for establishing priority. These are

(i) Daisy- chaining priority and (ii) Parallel priority interrupts.

Daisy chaining is a hardware implementation of polling procedure, whereas parallel priority method uses a priority encoder and is the fastest method for establishing the priority of interrupt sources.

The parallel priority interrupt method uses a register whose bits are set separately by the interrupt signal from each device. Priority is established according to the position of the bus in the register. In addition to the interrupt register, the circuit may include a mask register whose purpose is to control the status of each interrupt request. The mask register can be programmed to disable lower priority interrupts while a higher priority device is being serviced. It can also provide a facility that allows a high priority device to interrupt the CPU, while a lower priority device is being serviced.

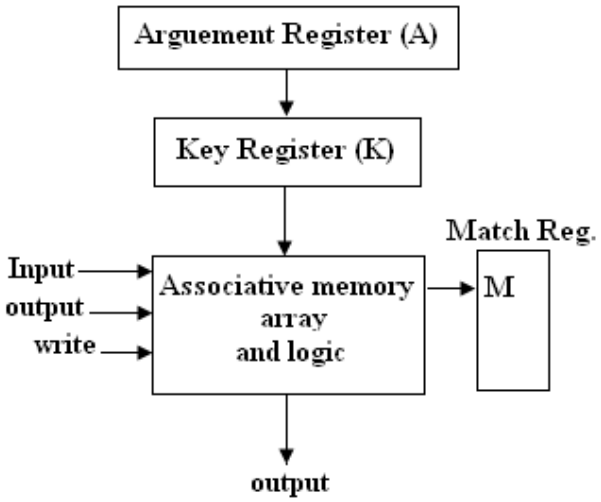
The priority logic for a system of four interrupt sources is shown below.



in the interrupt register, individual bits are set by internal I/O device requesting the service of CPU and is cleared by program instructions. The I/O devices are given with some priority value depending on their nature of devices and the services rendered by them. For example magnetic disk may get higher priority than a printer.
The mask register has same no. of bits as that of interrupt register. By means of program, it is possible to set or reset any bit of the mask register. If it is 1, then the associated interrupt is recognized, otherwise it is treated to be masked. Each interrupt bit along with its mask bit are applied to a AND gate lo produce four inputs to a priority encoder.
In this way the interrupts are recognized by CPU. The priority encoder output decides the vector address of the interrupt service subroutine. (ISR), which is to be loaded in to PC for execution of ISR during interrupt cycle. Another output of priority encoder sets an interrupt status flip-flop (IST FF). When an interrupt is recognized, the interrupt enable FF(IEN) can be set or cleared by the program to provide an overall control over the interrupt system. If $I_{EN} = 1$, then interrupt is recognised by CPU otherwise not. If IST= 1 & IEN = 1, then the interrupt signal goes to CPU, in return CPU sends interrupt acknowledgement signal, which enables the vector address register to place the vector address of ISR into program computer.

Q.16 Give the hardware organization of associative memory. Why associative memory is faster than other memories. Deduce the logic equation used to find the match in the associative memory. Explain how four-bit argument register is realized. (3+2+5)

Ans:
The hardware organization of the cell of one word in associative memory including the read and write logic is shown below:-



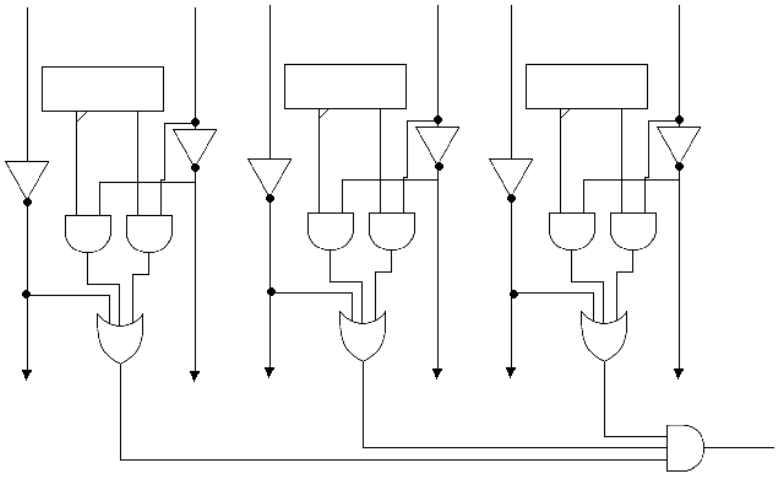
This consists of a memory array of logic for 'M' words with n-bits per word. The argument register A. The key register K, each has n-bits, one for each bit of a word. The match register 'M' has m bits, one for each memory word. Each word in the memory is compared in parallel with the content of the argument register. Words that match the bits of the argument register set a corresponding bit in the match register. After the matching process, those bits in the match register that have been set indicate

the fact that their corresponding words have been matched. As the identification and search of the data is done parallel by the hardware circuit so it is faster than other mapping logic.
Let A1, A2 .. ... An are n-bits of arguments register and K1, K2 „ . .Kn are n- bits of key register.
Let there be m-words in the memory, each of n- bits arrange in the matrix term having row 1 from 1 to m and column from 1 to n.
The output of comparison of each bit in a particular row i is given by xj.
Than Xj +kj' = if kj =1= 1if kj =0
The match logic register bits be M1, M2, Mn. As there is one bit in match register for each word .
Mi = (x1+k1') (x2 + k2')...(xn + kn')

$$Mi = \prod_{j=1}^{n}(Aj + Fij + Aj'Fij' + Kj')$$

As xj = AjFij+Aj'Fij'

The circuit for match for one word of associative memory is given below-



Q.17 Why page-table is required in a virtual memory system. Explain different ways of organizing a page table. (4+2)

Ans:
In any computer the address space is larger than memory space i.e. secondary memory is larger than the main memory, physically available to processor for execution of program. So programs and data are transferred to and from auxiliary memory and main memory based on demand imposed by the CPU. As the address of virtual memory is of larger bit then that of main memory, so mapping technique is required. To obtain the actual main memory address of the data from its virtual memory address. For this purpose a page table is required which holds the page number of virtual memory and the block number of the main memory. Further, each word of page table also has 'presence bit' to donate whether this page is presently available in main memory or not.
The different ways of organizing a page table are:
(i) In the R/W memory: it is called memory page table. But it is inefficient w.r.t. storage utilization and it required two main memory references to read a data,

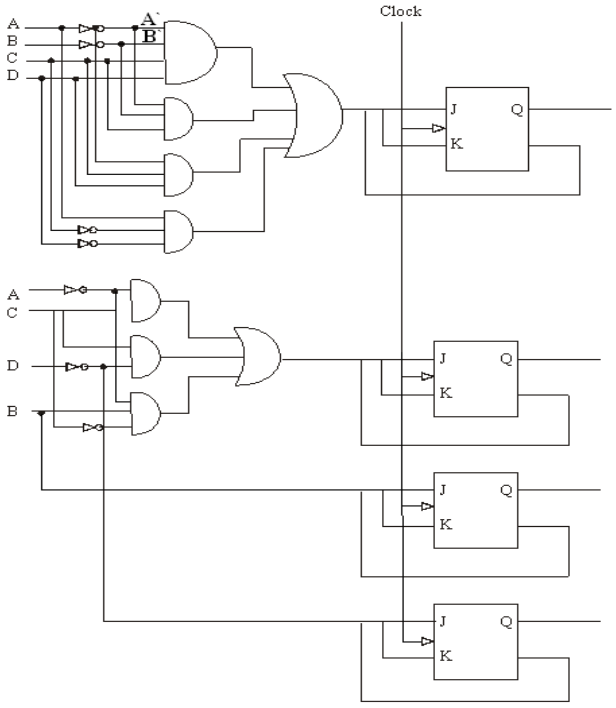
thus reducing the speed of execution of program.

(ii) By using associative logic: It is more efficient way to organize the page table, as it can be constructed with no. of words equal to no. of blocks in main memory.

Q. 18 Design a sequential circuit with JK flip-flop to satisfy the following state equations.

A(t+1) = A'B'CD + A'B'C + ACD +AC'D'
B(t+1) = A'C +CD' + + A'BC'
C(t+1) = B
D (t+1) = D'

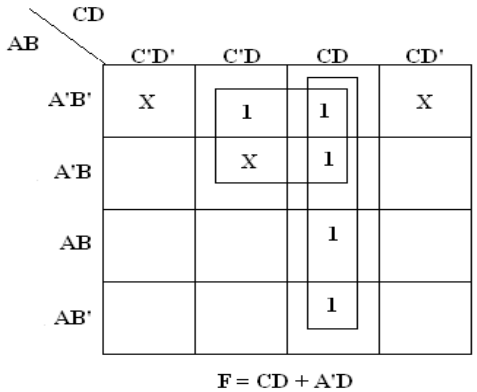
Ans.



Q.19 Simplify the Boolean function F together with don't care condition.

F (A,B, C, D) = m (1, 3, 7,11,15) + d(0,2,5)

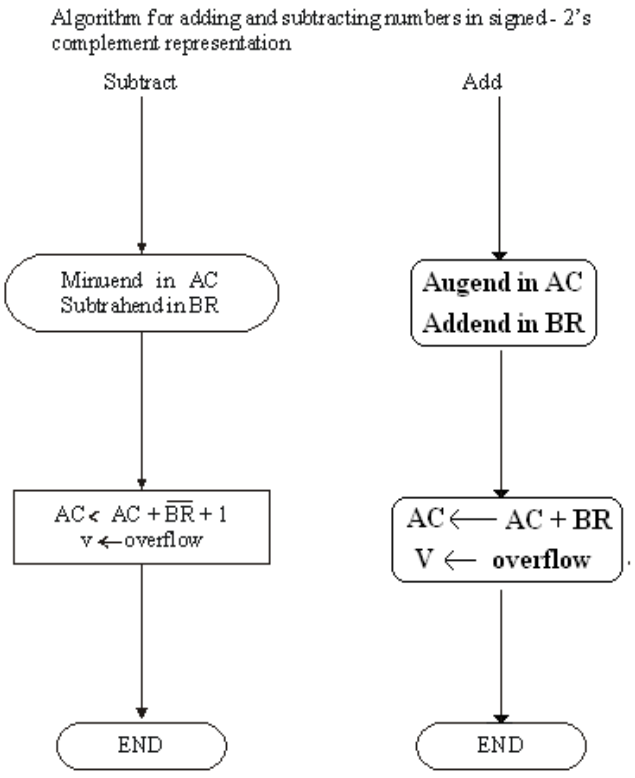
Ans.



F = CD + A'D

Q. 20 Explain the Adder-Subtractor with the help of 2's complement.

Ans.
The addition of two numbers in signed 2's complement form consists of adding the numbers with the sign bits treated the same as the other bits of the number. A carry-out of the sign-bits position is discarded. The subtraction consists of first taking the 2's complement of the subtrahend and then adding it to the minuend. When two numbers of a



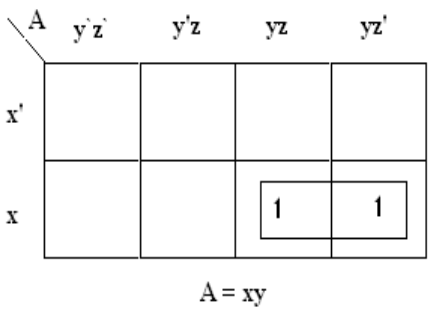
n digits each are added and the sum occupies n. + 1 digits then an overflow occurred. An overflow can be detected by inspecting the last two carries out of the addition. When the two carries are applied to an exclusive OR gate, the overflow is detected when, the output of the gate is equal to 1.
The sum is obtained by adding the contents of AC and BR (including their sign bits). The overflow bit V is set to 1 if the exclusive-OR of the last two carries is 1, and it is cleared to 0 otherwise. The subtraction operation is accomplished by adding the content of AC to the 2's complement of BR. Taking the 2's complement of BR has the effect of changing a positive number to negative, and vice versa. An overflow must be checked during this operation because the two numbers added could have the same sign. The programmer must realized that if an overflow occurs, there with be an erroneous result in the AC register.

Q.21 Design a combinational circuit using a ROM. The circuit accepts a 3-bit number and generates an output in binary number equal to the square of the input number.

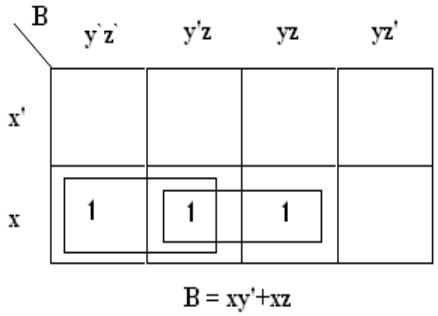
Ans.

The three bit number can represent 8 number of variables combination from 0-7. For this type of circuit, we need six bits of output. The truth table for the circuit is shown below:

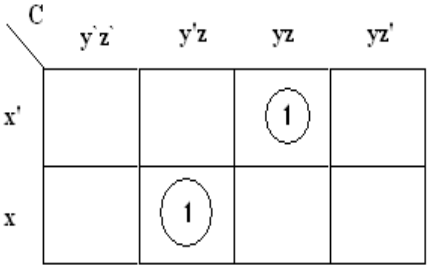
Input			Output					
X	Y	Z	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1
0	1	0	0	0	0	1	0	0
0	1	1	0	0	1	0	0	1
1	0	0	0	1	0	0	0	0
1	0	1	0	1	1	0	0	1
1	1	0	1	0	0	1	0	0
1	1	1	1	1	0	0	0	1



A = xy



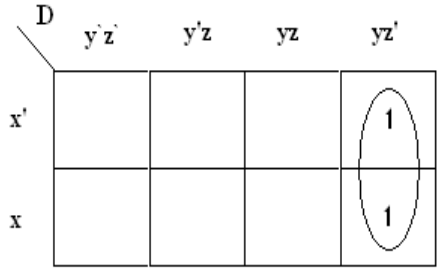
B = xy'+xz



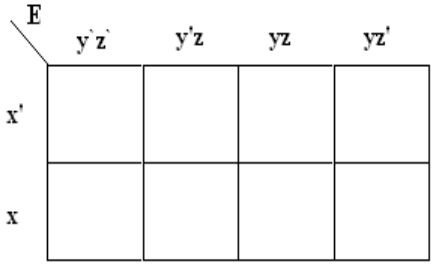
C = x'yz + xy'z

= z (x'y + xy')

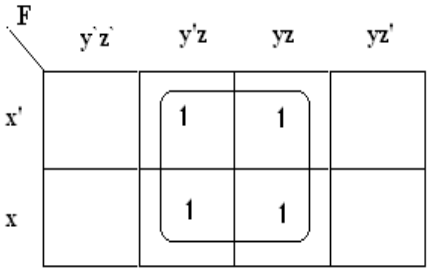
= z (x⊕y)



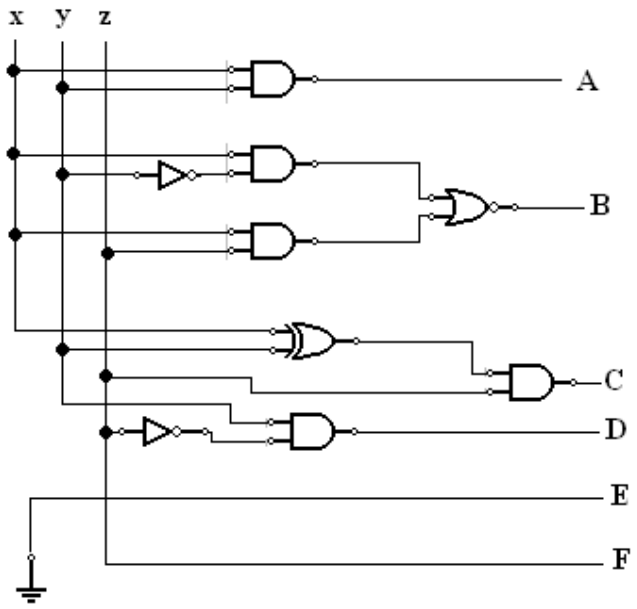
D = yz'



E = 0



F = Z



Q.22 Represent microinstructions for a microprogram of LD r_1, (r_2) instruction at control memory addresses a_j to a_{j+5}. How will be program counter increment f 9 for the next, instruction?

Ans.

(Z_{DR} = 1 if DR = 0 ; Z_{AC} = 1 if AC = 0)

INR (PC) = R' T1 + RT_2 + D_6 T_6 Z_{DR} + PB_9 (FGI) + PB_8 (FGO)
+ rB_4 (AC_{15}) ' rB_3 (AC_{15}) + rB_2 Z_{AC} +RB_1 E'

LD (PC) = D_4 T_4 + D_5 T_5

CLR (PC) = RT_1.

Q.23 Evaluate the arithmetic statement X = (A+B)*(C+D) using a general register computer with three address, two address and one address instruction format

Ans.

Three-Address Instructions

Computers with three-address instruction formats can use each address field to specify either a processor register or a memory operand. The program in assembly language that evaluates *X* = evaluates *X* = (A+B) * (C+D) is shown below, together with comments that explain the register transfer operation of each instruction.

ADD	R1, A, B	R1 M[A] + M[B]
ADD	R2, C, D	R2 M[C] + M[D]
MUL	X, R1, R2	M[X] R1 * R2.

It is assumed that the computer has two processor registers, *R*1and *R*2. The symbol *M*[A] denotes the operand at memory address symbolized by *A*.

Two-Address Instructions

Two-address instructions are the most common in commercial computers. Here again each address field can specify either a processor register or a memory word. The program to evaluate $X = (A + B) * (C + D)$ is as follows:

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	X, R1	M[X] <- R1

The MOV instruction moves or transfers the operands to and form memory and processor registers.

One-Address Instructions

One-address instructions use an implied accumulator (*AC*) register for all data manipulation. For multiplication and division there is a need for a second register. However, here we will neglect the second register and assume that the *AC* contains the result of all operations. The program to evaluate

$X = (A + B) * (C + D)$ is

LOAD	A	AC <-M[A}
ADD	B	AC<-A[C] + 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	X	M[X] <-AC

Q. 24 Write an assembly language program to convert a digital string into respective exactly opposite value.

Ans.

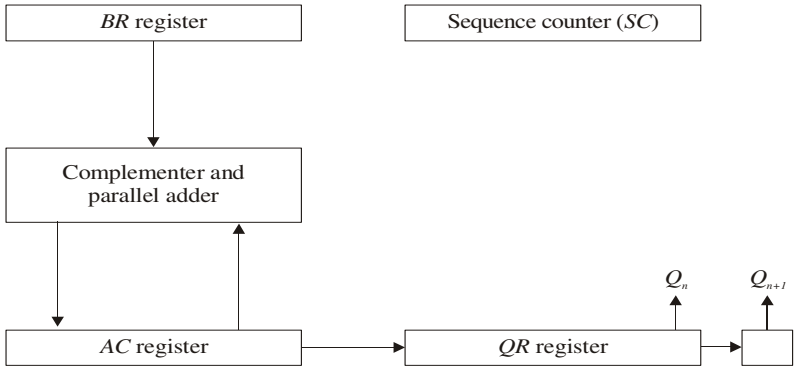
Address	Machine Code	Labels	Memories	Operands	Comments
2000	11,01,26		LXI	$D_1$2500H	Memory location for storing result.
2003	21,0025		LXI	$H_1$2500H	Address for count in H-L pair.
2006	46		MOV	B_1M	Ist number is accumulator
2007	21,0025	LOOP	INX	H	Decrement Court.
2008	DA, 14,20		JC	AHEAD	Yes, next digit in accumulator go to AHEAD
2009	00	AHEAD	DCR	C	Decrement Court.
2010	12		STAX	D	Store the result.

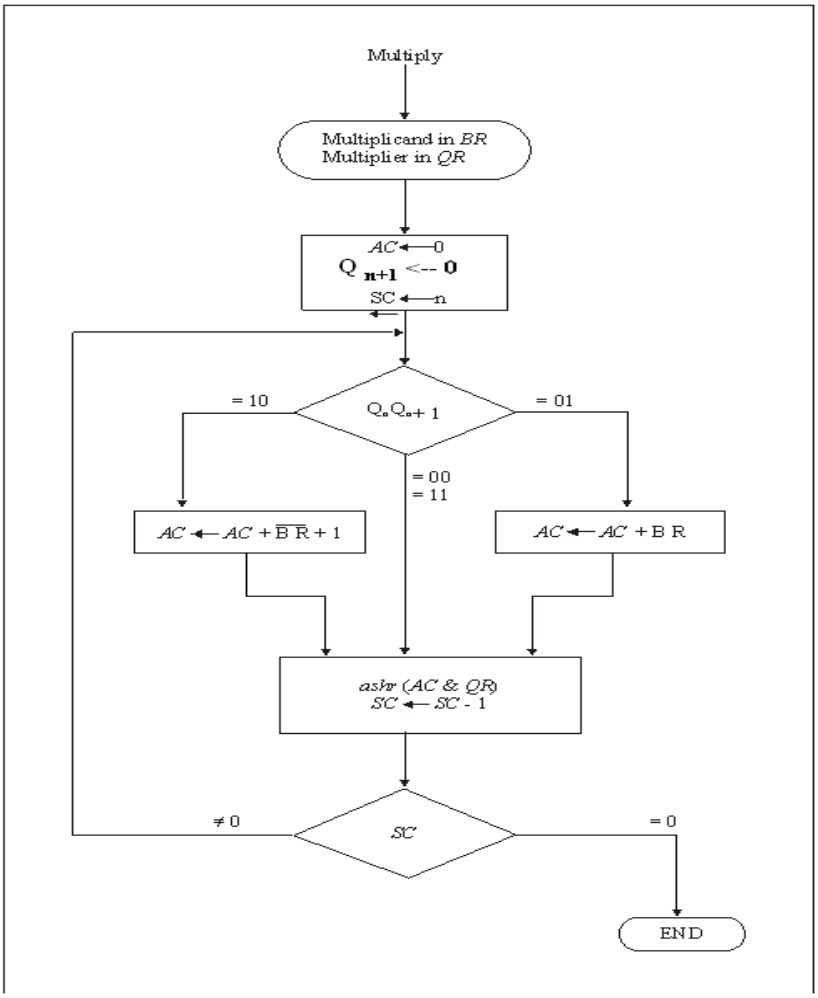
Q.25 Explain Booth's multiplications algorithm through an example. Give an example of multiplicant and multiplier for which this algorithm takes the maximum time.

Ans.

The hardware implementation of Booth algorithm requires the register configuration shown. The sign bits are not separated from the rest of the registers. Registers *A, B,* and *Q,* as *AC*, *BR,* and *QR,* respectively. Q_n designates the least significant bit of the multiplier in register QR. An extra flip-flop Q_{n+1} is appended to *QR* to facilitate a double bit inspection of the multiplier. The flowchart for Booth algorithm is shown.

Hardware for Booth algorithm





Example: Refer table 10-3 from page 348, Morris mano (3rd Edition)

Q. 26 Using 8-bit 2's complement representation of negative numbers, perform the following computations: .

(i) - 35 + (-11) (ii) 19 - (- 4)

Ans.

35 = 00100011
-35 = 11011100
11011101
11 = 00001011
-11 = 11110100
= 11110101
-35+ (-11) = 11011101
+ 11110101

111010010

(ii) 19 = 0001 0011
4 = 00000100
-4 = 11111100
19 - (- 4) = 19 + 4
00010011
00000100

00010111

Q. 27 Consider a cache (M_1) and memory (M_2) hierarchy with the following characteristics:

M_1 : 16 K words, 50 ns access time

M_2 : 1 M words, 400 ns access time

Assume 8 words cache blocks and a set size of 256 words with set associative mapping.

(i)Show the mapping between M_2 and M_1.

(ii)Calculate the Effective Memory Access time with a cache hit ratio of h = .95.

Ans.

(i) Main Memory = 1M words.
$= 2^{20}$ words.
Block size = 8 words.

main memory = $\frac{2^{20}}{8} = 2^{17}$ blocks

Cache memory = 16 k words.

Therefore Cache memory = $\frac{16K}{8} = \frac{2^{14}}{2^3} = 2^{11}$ blocks.

Set size = 265 words.

$= \frac{256}{8} = \frac{2^8}{2^3} = 2^5$ blocks

Tag = 17-11+5 = 11 bits

Set = 11-5 = 6 bits

word = 3 bits.

(ii) Given, t_c = 50ns tm = 400 ns

h= 0.95

Memory access time = $ht_c + (1- h)(t_c + t_m)$

$= 0.95 \times 50 + (1-0.95)(50+400)$

$= 47.5 + 22.5$

$= 70$ns

Q. 28 Write short notes on the following:

(i) Flip-Flops.

(ii) Multiplexer.

Ans.

(i) **Flip-Flops**:-

Flip-Flops is another name for a bistable multivibrator. A flip-flop is capable of storing 1 bit of binary data. It has two stable states- 'one' and 'zero'. The output stays low or high, to change it, the circuit must be drived by an input called trigger. Until the trigger arrives, the output voltage remains low or high indefinitely.

Edge Triggered Flip-flops:- An edge triggered flip-flop responds only during the brief instant the clock switches from one voltage level to another. When the triggering occurs on the positive going edge of the clock, it is called positive-edge triggering. Sometimes, triggering on the negative edge is better suited to the application. This means the trailing edge of the clock activates the gates, allowing data to be recognized. This is called negative-edge triggering.

Preset and Clear :- When power is first applied, flip-flops come up in random states. To get some computers started, an operator has to push a reset button. This sends a reset or CLEAR signal to all flip-flops. Also, it is necessary in some digital system to PRESET (synonymous with set) certain flip-flops.

In a clocked flip-flop PRESET and CLEAR inputs are called asynchronous, because they activate the flip-flops independently of the clock.

Types of Flip-flops

1)R-S. Flip-Flop

2)Clocked R-S Flip-FLop

3)D Flip-FLop.

4)J-K Flip-Flop

5)Master-Slave Flip-Flop

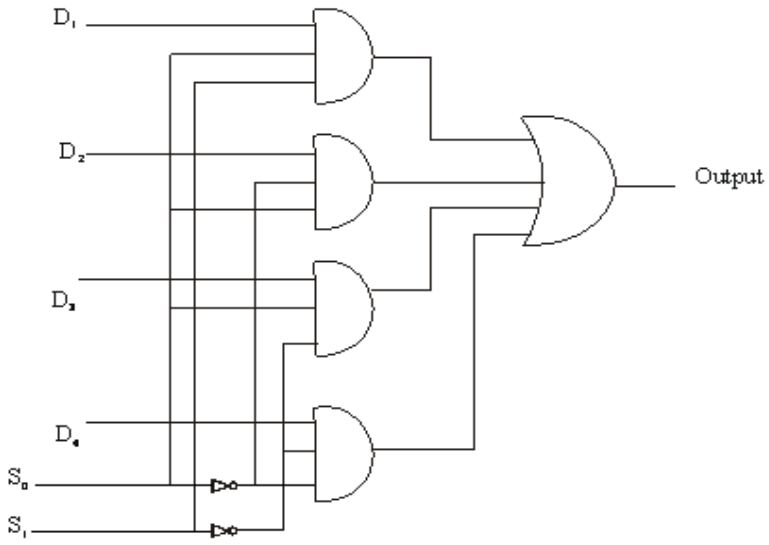
6)T-Flip-Flop

(ii) **Multiplexer**

The multiplexer (MUX) is a combinational logic circuit that selects binary information from one of the multiple input lines (D_{n-1}.....D_1, D_0) and directs it to an output line according to a received select code ($S = S_{n-1}$..... S_1 S_0) and directs it. A block diagram of a four input multiplexer is shown in Fig. it's truth table given below.

Truth Table		
S_0	S_1	O/P
0	0	D_4
0	1	D_a
1	0	D_2
1	1	D_1

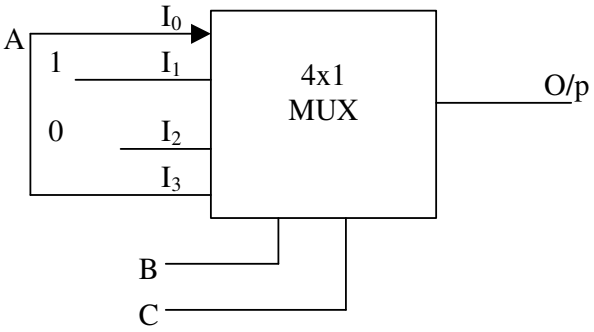
Output = $D_4 S_0 S_1 + D_3 S_0 S_1 + D_2 S_0 S_1 + D_1 S_0 S_1$



Q.29 Implement the following by using 4:1 multiplexer
$P = \Pi(M_0, M_1, M_5, M_7)$.

Ans.

	I_0	I_1	I_2	I_3
A´	0	1	2	3
A	4	5	6	7
	A´	1	0	A



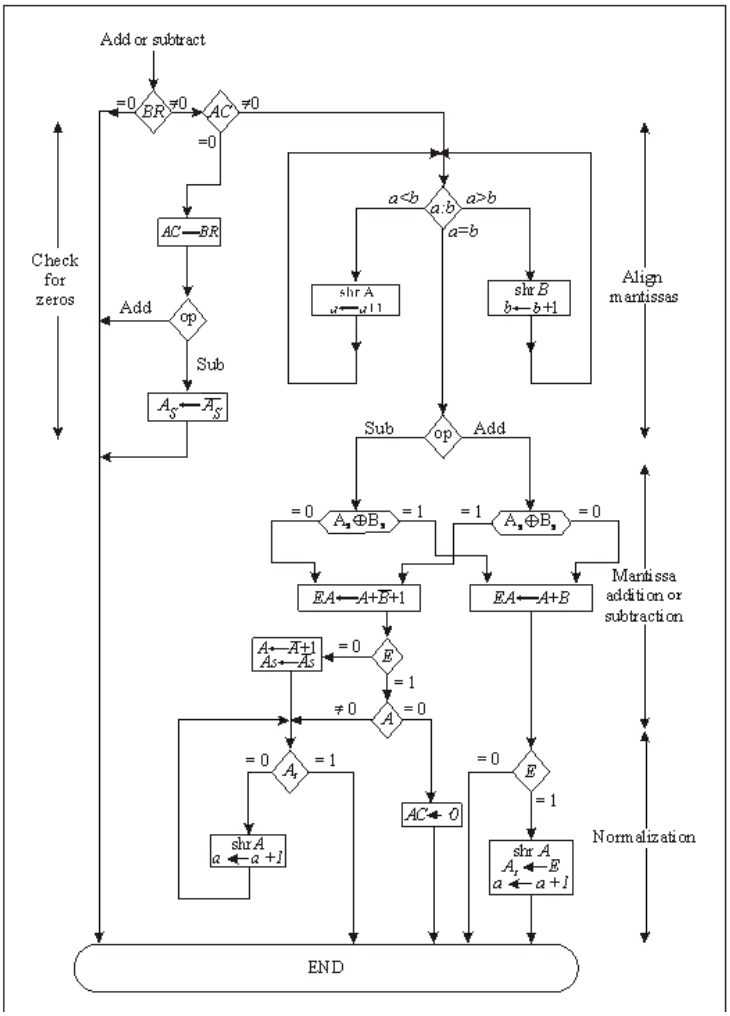
B and C are the selection lines.

Q. 30 Explain with neat flow chart the addition and subtraction of floating point numbers.

Ans:

In addition and subtraction the two floating point operands are in AC and BR. The sum or difference is formed in AC. The algorithm can be divided into four parts.

Flow Chart



(1) Check for zeros.
(2) Align the mantissas
(3) Add or subtract the mantissas
(4) Normalize the result.

The flowchart for adding or subtracting two floating point binary numbers is shown in fig. If BR is equal to zero, the operation is terminated, with the value in the AC being the result. If AC is equal to zero, we transfer the content of BR into AC and also complement its sign if the numbers are to be subtracted. If neither number is equal to zero, we proceed to align the mantissas.

The magnitude comparator attached to exponents a and b provides three outputs that indicate their relative magnitude. If the two exponents are equal, then perform the

arithmetic operation. If the exponents are not equal, the mantissa having the smaller exponent is shifted to the right and its exponent incremented. This process repeated until the two exponents are equal.

Q.31 Multiply $(-7)_{10}$ with $(3)_{10}$ by using Booth's multiplication. Give the flow table of the multiplication.

Ans.
Binary equivalent of 7 = 0111, -7 = 1001
Binary equivalent of 3 = 0011.

Q_n	Q_{n+1}	BR=1001 $\overline{BR}$+1=0111	AC	QR	Q_{n+1}	SE
		Initial	0000	0011	0	100
1	0	Subtract BR	0111 0111			
		ashr	0011	1001	1	011
1	1	ashr	0001	1100	1	010
0	1	Add BR	1001 1010			
		ashr	1101	0110	0	001
0	0	ashr	1110	1011	0	000
Final product = 11101011						

Q.32 Design a hardware circuit by using common bus architecture to implement the following Register Transfer Languages.

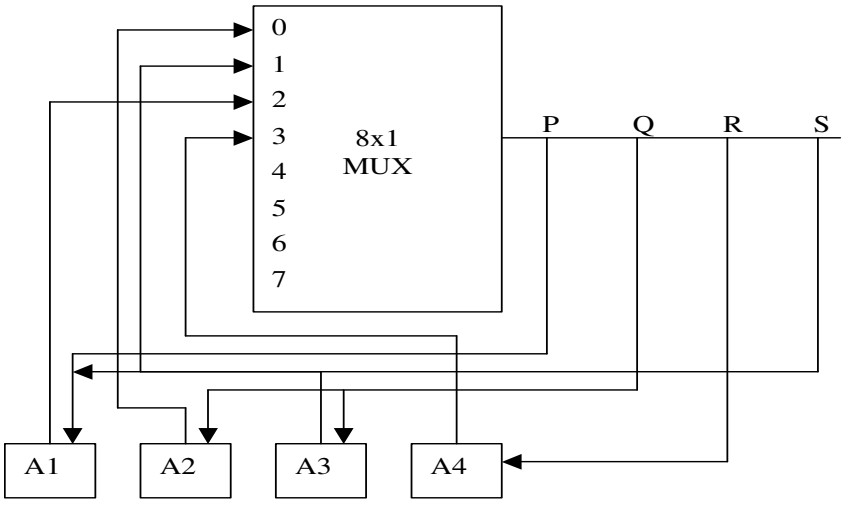
P: $A_1 \leftarrow A_2$

Q: $A_2 \leftarrow A_3$

R: $A_4 \leftarrow A_1$

S: $A_3 \leftarrow A_4$, $A_1 \leftarrow A_4$

Where A_1, A_2, A_3, A_4 are one bit register



Input			Output
S_2	S_1	S_1	
0	0	0	P
0	0	1	Q
0	1	0	R
0 1	1 0	1 0	S_1 S_2]s
X	X	X	X
X	X	X	X
X	X	X	X
X	X	X	X

Q.33 Explain hardware polling method for data transfer.

Ans.

In a bus system that uses polling, the bus grant signal is replaced by a set of lines called poll lines which are connected to all units. These lines are used by the bus controller to define an address for each device connected to the bus. The bus controller sequences through the addresses in a prescribed manner. When a processor that requires access recognizes its address, it activates the bus busy line and then accesses the bus. After number of bus cycles, the polling process continues by choosing a different processor. The polling sequence is normally programmable, and as a result, the selection priority can be altered under program control.

There are so many ways to boot alternative OSs to common PCs these days. There are many ways to run multiple operating systems on a single piece of hardware. Below we count the pros & cons of the three most popular methods: re-partitioning, emulation and virtualization.

1. Partitioning

With the re-partitioning method the user must manually re-partition his hard drive and then install the OSs one after the other on the right partition and then use a boot manager to boot between OSs.

Pros:

- OSs run full speed
- Full access to the hardware
- Partition resizing is possible (depending on the file system used)

Cons:

- Manually partitioning can be tricky for newbies
- Frustrating if you 'lose' your boot manager after an OS update
- Requires a full reboot to run another OS

2. Virtualization

Virtualization is the new kid on the block and it's gaining ground very fast. To the eyes of a simple user it looks a lot like straight emulation, but in reality it's not. The virtualizer "shares" more hardware resources with the host OS than an emulator does.

Pros:

- Slower than the re-partitioning method but much faster than emulation
- Support for all host hardware, including 3D support
- Virtual clustering made-easy

Cons:
- Requires enough RAM
- Only runs on the same architecture as the host OS

3. Emulation

Emulators will completely emulate the target CPU and hardware (e.g. sound cards, graphics cards, etc). Emulators are the "old way" of running multiple OSs on a single computer. Emulation on PCs these days is only good for non-OS usages (e.g. game consoles, embedded systems) or specific OS/CPU development purposes.

Pros:
- Best solution for embedded/OS development
- Doesn't interfere with the underlying host OS
- Can be ported to any architecture

Cons:
- Can be very slow
- No 3D or other exotic PC hardware support
- Requires enough RAM

Being a traditional geek chick myself, I still prefer the manual re - partioning method for my PCs (I like the clean nature of it), but for a MacTel I would much prefer Boot Camp's special portioning scheme (if Vista, Linux are supported properly — otherwise, Virtualization is my next best option on MacTels). Tell us what's your preferred method is below

Q. 34 Explain with an example, how effective address is calculated in different types of addressing modes.

Ans.

To explain the difference between the various modes, the two word instruction at address 200 and 201 is a "load to AC" instruction with an address field equal to 500. The first word of the instruction specifies the operation code and mode, and the second word specifies the address part. PC has the value 200 for fetching this instruction. The content of processor register R1 is 400, and the content of an index register XR is 100. AC receives the operand after the instruction is executed. The figure lists a few pertinent addresses and shows the memory content at each of these addresses for each possible mode. We calculate the effective address and the operand that must be loaded into AC. In the direct address mode the effective address is the address part of the instruction 500 and the operand to be loaded into AC is 800. In the immediate mode the second word of the instruction is taken as the operand rather than an address, so 500 is loaded into AC. In the indirect mode the effective address is stored in memory at address 500. Therefore the effective address is 800 and the operand is 300. In the Index mode the effective address is XR + 500 = 100 + 500 = 600 and the operand is 900. In the register mode the operand is in R1 and 400 is loaded into AC.

The Autoincrement mode is the same as the register indirect mode except that R1 is incremented to 401 after the execution of the instruction. The Autodecrement mode decrements R1 to 399 prior to the execution of the instruction. In the relative mode the effective address is 500+202=702 and the operand is 325.

In the register indirect made the effective address is 400, equal to the content of R1 and the operand loaded into AC is 700.

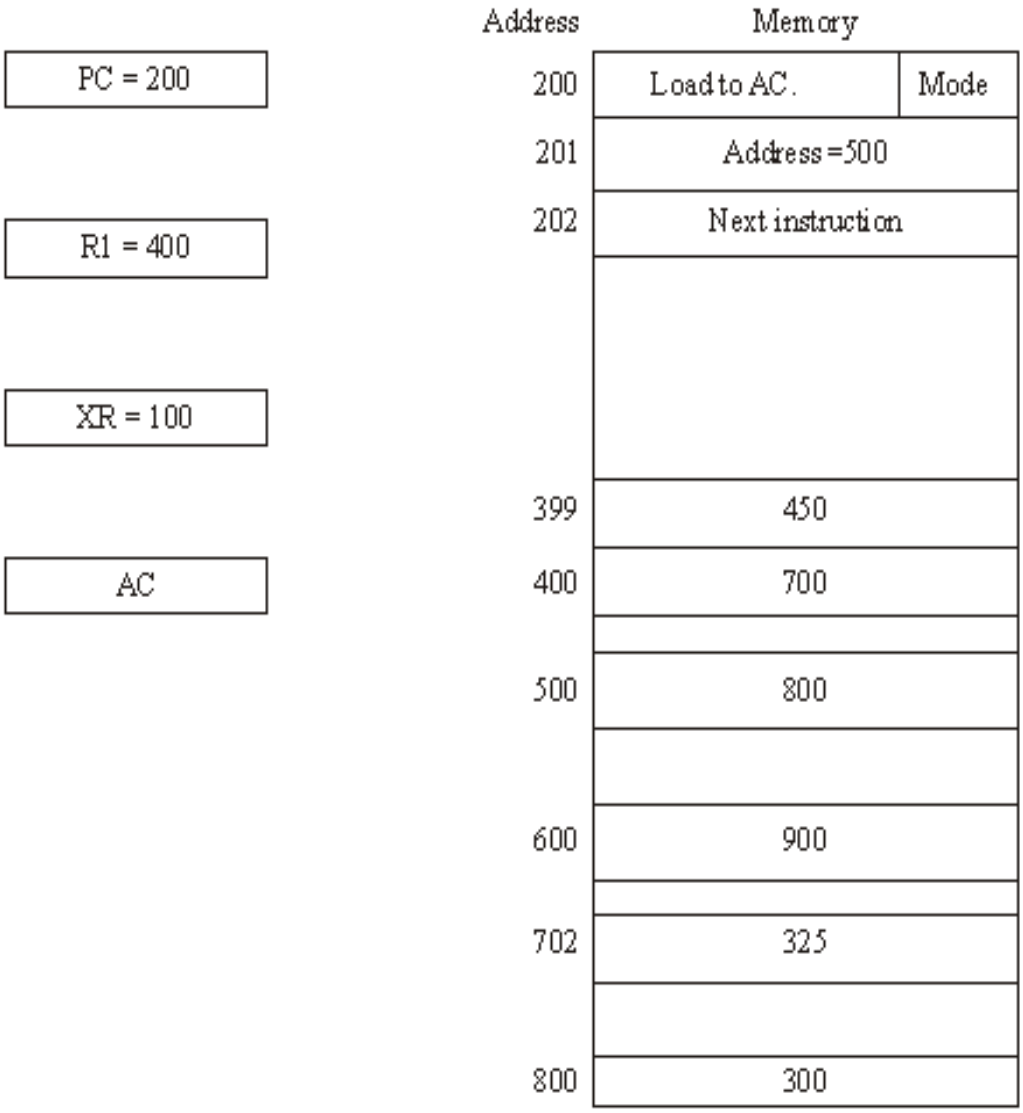


Fig. Numerical example for addressing modes

Q.35 How an interrupt is recognized? Explain the interrupt cycle.

Ans.

In the parallel priority interrupt method uses a register whose bits are get separately by the interrupt signal from each device. Priority is established according to the position of the bits in the register. In the interrupt register the circuit may include a mask register whose purpose is to control the status of each interrupt request. The mask register can be programmed to disable low priority interrupts while a higher priority device is being carried.

The mask register has the same number of bits as the interrupt register. Each interrupt bit and its corresponding mask bit are applied to an AND get to produce the four inputs to a priority encoder. In this way an interrupt is recognized only if its corresponding mask bit is get to 1 by the program.

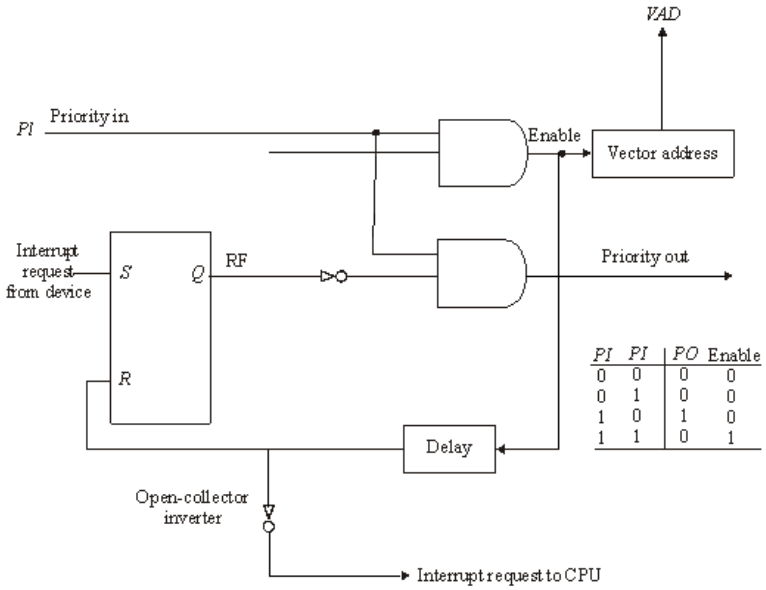


Fig. One stage of the daisy-chain priority arrangement

Interrupt Cycle:-

The interrupt enable flip-flop *IEN* shown it can be set or cleared by program instructions. When *IEN* is cleared, the interrupt request coming from *IST* is neglected by the CPU. The program-controlled *IEN* bit allows the programmer to choose whether to use the interrupt facility. If an instruction to clear *IEN* has been inserted in the program, it means that the user does not want his program to be interrupted. An instruction to set *IEN* indicates that the interrupt facility will be used while the current program is running. Most computers include internal hardware that clears *IEN* to 0 every time an interrupt is acknowledged by the processor. At the end of each instruction cycle the CPU checks *IEN* and the interrupt signal from *IST*. If either is equal to 0, control continues with the next instruction. If both *IEN* and *IST* are equal to 1, the CPU goes to an interrupt cycle. During the interrupt cycle the CPU performs the following sequence of micro-operations:

SP	*SP* –1	Decrement stack pointer
M[*SP*]	*PC*	Push *PC* into stack
INTACK	1	Enable interrupt acknowledge
PC	*VAD*	Transfer vector address to *PC*
IEN	0	Disable further interrupts

Go to fetch next instruction.

The CPU pushes the return address from *PC* into the stack. It then acknowledges the interrupt by enabling the *INTACK* line. The priority interrupt unit responds by placing a unique interrupt vector into the CPU data bus. The CPU transfers the vector address into *PC* and clears *IEN* prior to going to the next fetch phase. The instruction read from memory during the next fetch phase will be the one located at the vector address.

Q. 36 Compare assembly language with high level language. Write a program using assembly language of 8085 microprocessor to check whether a given number is

odd or even. If the given number is even then display '1' on its SOD line. Give the flow chart also.

Ans.

High Level Language	Assembly Language
(1) Programs developed in High level language are most understandable	(1) Program are less under-stand able than high level language but more than machine language
(2) Program are portable	(2) Not portable, portable to the processor of same architecture only
(3) Debugging is easier	(3) Debugging is more complex
(4) Most suited for software development	(4) Not good for large programs
(5) Program are not machine dependent	(5) Program are machine dependent
(6) Provides flexible construct for program development	(6) Does not provide flexible construct for development
(7) Programs are translated using compiler and/or interpreter to generate object code	(7)Uses assembler to generate object code

```
DATA SEGMENT
NUMBER DB 11
EVE DB      ' ENTERED NUMBER IS EVEN'
ODD DB      'ENTERED NUMBER IS ODD'

 DATA ENDS
 CODE SEGMENT
 ASSUME CS: CODE, DS: DATA
                START : MOV    DX, DATA
                        MOV    DS, DATA
                        MOV    AL, NUMBER
                        SHR    AL, 1
                        JL     LABEL 1
                 MOV    DX, OFFSET EVE
                      INC      LABEL 2

   LABEL1 :   MOV DX, OFFSET ODD
   LABEL 2 :  MOV AH, 09H
              INT   21H
              MOV  AH, 4CH
              INT    21H
   CODE       ENDS
   END        START
```

Q.37 Compare horizontal microcode with vertical microcode. State the advantage of micro programmed control unit. **(6)**

Ans.

Horizontal Microcode	**Vertical Micro-code**
(1) Control signal directly in micro-code (2) All control signals always there. (3) Lots of signals many bits in micro-instruction	(1) Each action encoded density. (2) Actions need to be decoded to signal at execution time. (3) Takes less space but may be slower.

Advantage of micro programmed control unit is that once the hardware configuration is established. There should be no need for further hardware or wiring changes. If establish a different control sequence for the system, is specify a different set of micro instructions for control memory.

Q. 38 Explain in detail the different mappings used for cache memory. Compare them.

Ans.
Three types of mapping procedures used for cache memory:
(i) Associative mapping
(ii) Direct mapping
(iii) Set-associative mapping

(i) **Associative mapping:-**

The fastest and most flexible cache organization uses an associative memory. The organization is illustrated. The associative memory stores both the address and content (data) of the memory word. This permits any location in cache to store any word from main memory. The diagram shows three words presently stored in the cache. The address value of 15 bits is shown as a five-digit octal number and its corresponding 12-bit word is shown as a four-digit octal number and its corresponding 12-bit word is shown as a four-digit octal number. A CPU address of 15 bits is placed in the argument register and the associative memory is searched for a matching address. If the address is found, the corresponding 12-bit data is read and sent to the CPU. If no match occurs, the main memory is accessed for the word. The address-data pair is then transferred to the associative cache memory. If the cache is full, an address-data pair must be displaced to make room for a pair that is needed and not presently in the cache. The decision as to what pair is replaced is determined from the replacement algorithm that the designer chooses for the cache. A simple procedure is to replace cells of the cache is round-robin order whenever a

new word is requested from main memory. This constitutes a first-in first-out (FIFO) replacement policy.

Fig. Associative mapping cache (all numbers in octal)

CPU address (15 bits)

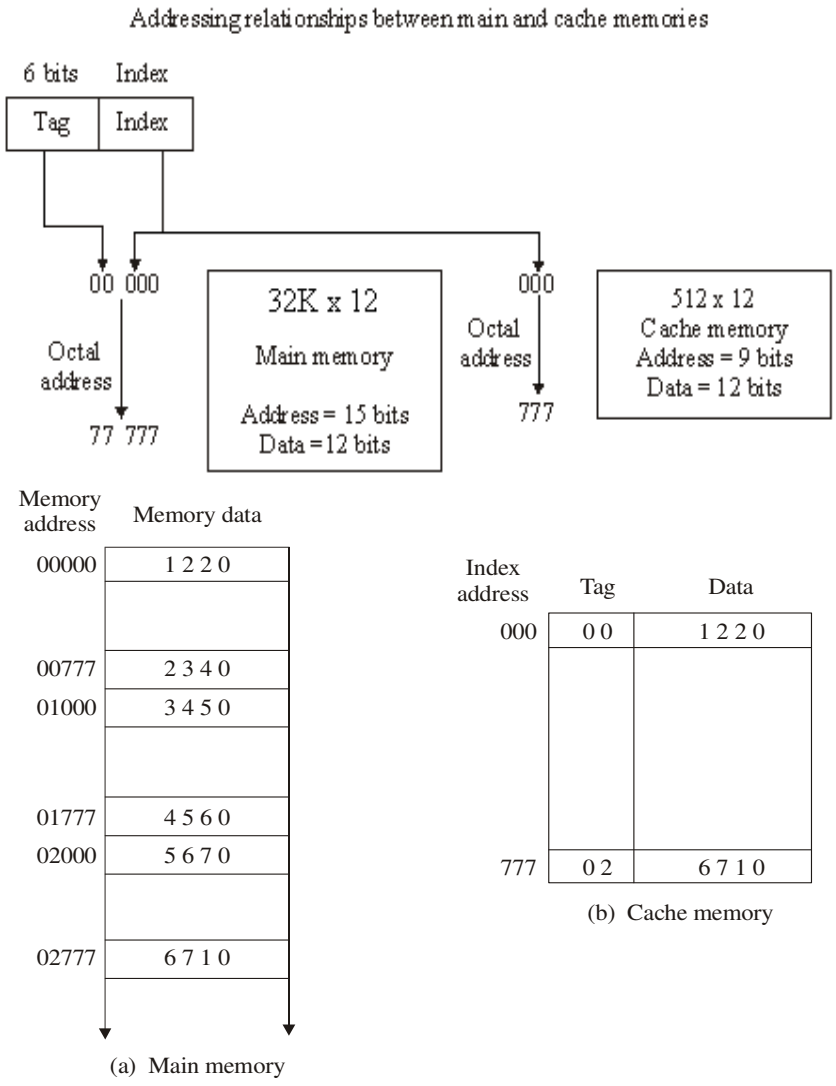
↓

Argument register

← Address →	← Data →
01000	3450
02777	6710
22345	1234

(ii) **Direct Mapping** :-

Associative memories are expensive compared to random-access memories because of the added logic associated with each cell. The possibility of using a random-access memory for the cache is investigated. The CPU address of 15 bits is divided into two fields. The nine least significant bits constitute the *index* field and the remaining six bits form the *tag* field. The figure shows that main memory needs an address that includes both the tag and the index bits. The number of bits in the index field is equal to the number of address bits required to access the cache memory. In the general case, there are 2^k words in cache memory and 2^n words in main memory. The n bit memory address is divided into two fields: k bits for the index field and the *n*-k bits for the tag field. The direct mapping cache organization uses the *n*-k bits for the tag field. The direct mapping cache organization uses the *n*-bit address to access the main memory and the k-bit index to access the cache. The internal organization of the words in the cache memory is as shown. Each word in cache consists of the data word and its associated tag. When a new word is first brought into the cache, the tag bits are stored alongside the data bits. When the CPU generates a memory request, the index field is used for the address to access that cache. The tag field of the CPU address is compared with the tag in the word read from the cache. If the two tags match, there is a hit and the desired data word is in cache. If there is no match, there is a miss and the required word is read from main memory. It is then stored in the cache together with the new tag, replacing the previous value. The disadvantage of direct mapping is that the hit ratio can drop considerably if two or more words whose addresses have the same index but different tags are accessed repeatedly. However, this possibility is minimized by the fact that such words are relatively far apart in the address range.



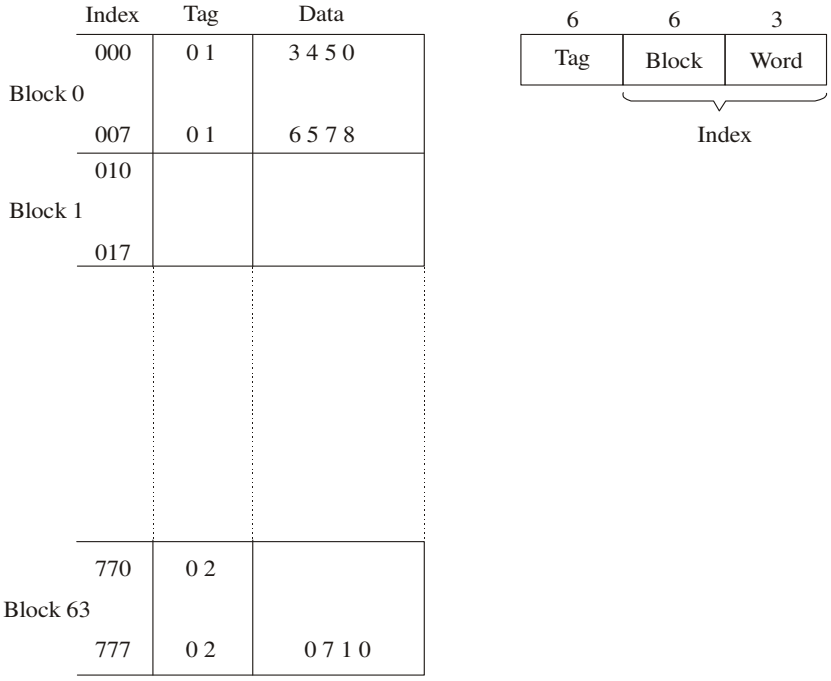
Direct mapping cache organization

To see how the direct-mapping organization operates, consider the numerical example shown. The word at address zero is presently stored in the cache (index = 000, tag = 00, data = 1220). Suppose that the CPU now wants to access the word at address 02000. The index address is 000, so it is used to access the cache. The two tags are then compared. The cache tag is 00 but the address tag is 02, which does not produce a match. Therefore, the main memory is accessed and the data word 5670 is transferred to the CPU. The cache word at index address 000 is then replaced with a tag of 02 and data of 5670.

The direct-mapping example just described uses a block size of one word. The same organization but using a block size of 8 words is shown.

The index field is now divided into two parts: the block field and the word field. In a 512-word cache there are 64 blocks of 8 words cache, since 64 x 8 = 512. The block number is specified with a 6-bit field and the word within the block is specified with a 3-bit field. The tag field stored within the cache is common to all eight words of the same block. Every time a miss occurs, an entire block of eight words must be transferred from main memory to cache memory. Although this takes extra

time, the hit ratio will most likely improve with a larger block size because of the sequential nature of computer programs.



Direct mapping cache with block size of 8 words

(ii) **Set-Associative Mapping** :-

It was mentioned previously that the disadvantage of direct mapping is that two words with the same index in their address but with different tag values cannot reside in cache memory at the same time. A third type of cache organization, called set-associative mapping, is an improvement over the direct-mapping organization in that each word of cache can stored two or more words of memory under the same index address. Each data word is stored together with its tag and the number of tag-data items in one word of cache is said to form a set. An example of a set-associative cache organization for a set size of two is shown. Each index address refers to two data words and their associated tags. Each tag requires six bits and each data words has 12 bits, so the word length is 2(6+12) = 36 bits. An index address of nine bits can accommodate 512 words. Thus the size of cache memory is 512 x 36. It can accommodate 1024 words of main memory since each word of cache contains two data words. In general, a set-associative cache of set size k will accommodate *k* words of main memory in each word of cache.

The octal numbers listed are with reference to the main memory contents illustrated in the fig. The words stored at addresses 01000 and 02000 of main memory are stored in cache memory at index address 000. Similarly, the words at addresses *02777* and *00777* are stored in cache at index address *777*. When the CPU generates a memory request, the index value of the address is used to access the cache. The tag field of the CPU address is then compared with both tags in the cache to determine if a match occurs. The comparison logic is done by an associative search of the tags in the set similar to an associative memory search: thus the name "set-associative." The

Index	Tag	Data	Tag	Data
000	0 1	3 4 5 0	0 2	5 6 7 0
777	0 2	6 7 1 0	0 0	2 3 4 0

hit ratio will improve as the set size increases because more words with the same index but different tags can reside in cache. However, an increase in the set size increases the number of bits in words of cache and requires more complex comparison logic. When a miss occurs in a set-associative cache and the set if full, it is necessary to replace one of the tag-data items with a new value. The most common replacement algorithms used are: random replacement, first-in, first-out (FIFO), and least recently used (LRU). With the random replacement policy the control chooses one tag-data item for replacement at random. The FIFO procedure selects for replacement the item that has been in the set the longest. The LRU algorithm selects for replacement the item that has been least recently used by the CPU. Both FIFO and LRU can be implemented by adding a few extra bits in each word of cache.

Q.39 State how different policies of writing into cache are implemented.

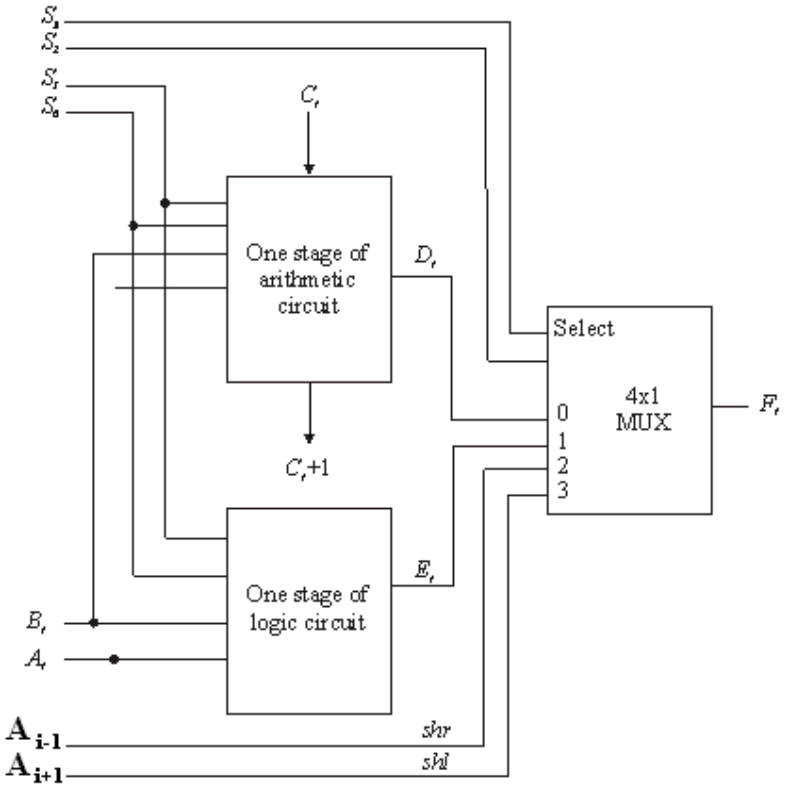
Ans.

The simplest and most commonly used procedure is to update main memory with every memory write operation, with cache memory being updated in parallel if it contains the word at the specified address. This is called the write-through method. This method has the advantage that main memory always contains the same data as the cache. This characteristic is important in systems with direct memory access transfers. It ensures that the data residing in main memory are valid at all times so that an I/O device communicating through DMA would receive the most recent updated data. The second procedure is called the write-back method. In this method only the cache location is updated during a write operation. The location is then marked by a flag so that later when the word is removed from the cache it is copied into main memory. The reason for the write-back method is that during the time a word resides to the cache, it may be updated several times; however, as long as the word remains in the cache, it does not matter whether the copy in main memory is out of date, since requests from the word are filled from the cache. It is only when the word is displaced from the cache that an accurate copy need be rewritten into main memory.

Q. 40 Design a hardware circuit to implement logical shift, arithmetic shift and circular shift operations. State your design specifications.

Ans.

In computer systems a number of storage registers connected to a common operational unit called an arithmetic logic unit (ALU). To perform a micro-operation, the contents of registers are placed in the inputs of a common ALU. The ALU performs an operation and the result of the operation is then transferred to a destination register. The ALU is a combination circuit so that the entire register transfer operation from the source register through the ALU and into the destination register can be performed during one clock pulse period. The shift micro-operation are often performed in a separate unit, but sometimes the shift unit is made part of the overall ALU. The arithmetic, logic and shift circuit can be combined into one ALU with common selection variables. One stage of an arithmetic logic and shift unit is shown in fig.



Function Table for Arithmetic Logic Shift Unit

Operation select						
S_3	S_2	S_1	S_0	c_{in}	Operation	Function
0	0	0	0	0	$F = A$	Transfer A
0	0	0	0	1	F = A+1	Increment A
0	0	0	1	0	F = A + B	Addition
0	0	0	1	1	F = A + B + 1	Add with carry
0	0	1	0	0	F = A + $\overline{B}$	Subtract with borrow
0	0	1	0	1	F = A + $\overline{B}$ +1	Subtraction
0	0	1	1	0	F = A - 1	Decrement A
0	0	1	1	1	F = A	Transfer A
0	1	0	0	x	F = A^ B	AND
0	1	0	1	x	F = A v B	*OR*
0	1	1	0	x	F = A $\oplus$ B	*XOR*
0	1	1	1	x	F = $\overline{A}$	Complement A
1	0	x	x	x	F = shr A	Shift right A into F
1	1	x	x	x	F = shl A	Shift left A into F

(i) Input A_i and Bi are applied to both the arithmetic and logic units. A particular micro-operation is selected with inputs S_1 and S_0.

(ii) A 4x1 MUX at the output chooses between an arithmetic output in D_i and a logic output in E_i.

(iii) The data inputs to the multiplexer are selected with inputs S_3 and S_2.

(iv) The other two data inputs to the MUX receive inputs A_{i-1} for the shift right operation and A_{i+1} for the shift left operation.

(v) C_{in} is the selection variable for the arithmetic operation.

(vi) The circuit provides eight arithmetic operation, four logic operations and two shift operations. Each operation is selected with the five variables S_3, S_2, S_1, S_0 and C_{in}.

(vii) The table lists the 14 operations of the ALU. The first eight are arithmetic operation and are selected with S_3 S_2 = 00. The next four are logic operation and are selected with S_3 S_2 = 01 and last two operation are shift operation and are selected with S_3 S_2 = 10 and 11.

Q.41 Discuss different techniques used for interfacing I/O units with the processor.

Ans.

Isolated I/O:- In the isolated I/O configuration the CPU has distinct input and output instructions, and each of these instructions is associated with the address of an interface register. When the CPU fetches and decodes the operation code of an input or output instruction, it places the address associated with the instruction into the common address lines. At the same time, it enables the I/O read or I/O write control line. When the CPU is fetching an instruction or an operand from memory, it places the memory address on the address lines and enables the memory read or memory write control line. The isolated I/O method isolates memory and I/O

addresses so that memory address values are not affected by interface address assignment since each has its own address space.

Memory Mapped I/O :- In computers that employ only one set of read and write signals and do not distinguish between memory and I/O addresses. The configuration is referred to as memory-mapped I/O. In a memory-mapped I/O organization there are no specific input or output instructions. The CPU can manipulate I/O data residing in interface registers with the same instructions that are used to manipulate memory words. The advantage is that the load and store instructions used for reading and writing from memory can be used to input and output data from I/O registers.

Q. 42 Write short notes on:-

(i) Sequential circuit.
(ii) Priority encoder.
(iii) Virtual memory.
(iv) Program control instructions.

Ans.

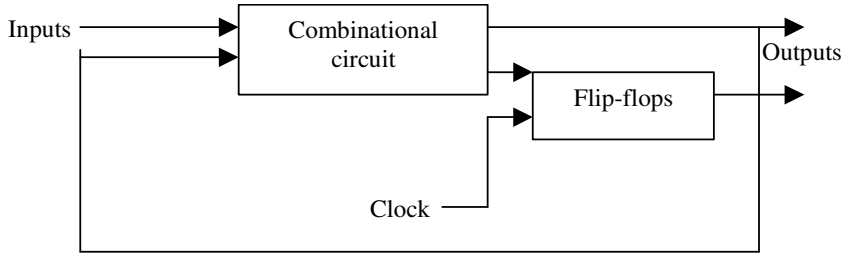
(i) **Sequential circuit**:-

A sequential circuit is an interconnection of flip-flops and gates. The gates by themselves constitute a combinational circuit, but when included with the flip-flops, the overall circuit is classified as a sequential circuit. The block diagram of a clocked sequential circuit is shown in Figure below. It consists of a combinational circuit and a number of clocked flip-flops. In general, any number or type of flip-flops may be included. In the diagram, the combinational circuit block receives binary signals from external inputs and from the outputs of flip-flops. The gates in the combinational circuit determine the binary value to be stored in the flip-flops after each clock transition. The outputs of flip-flops, in turn, are applied to the combinational circuit inputs and determine the circuit's behavior. The next state of flip-flops is also a function of their present state and external inputs. Thus a sequential circuit is specified by a time sequence of external inputs, external outputs, and internal flip-flop binary states.

Block diagram of a clocked synchronous sequential circuit

(ii) **Priority encoder**:-

The priority encoder is a circuit that implements the priority



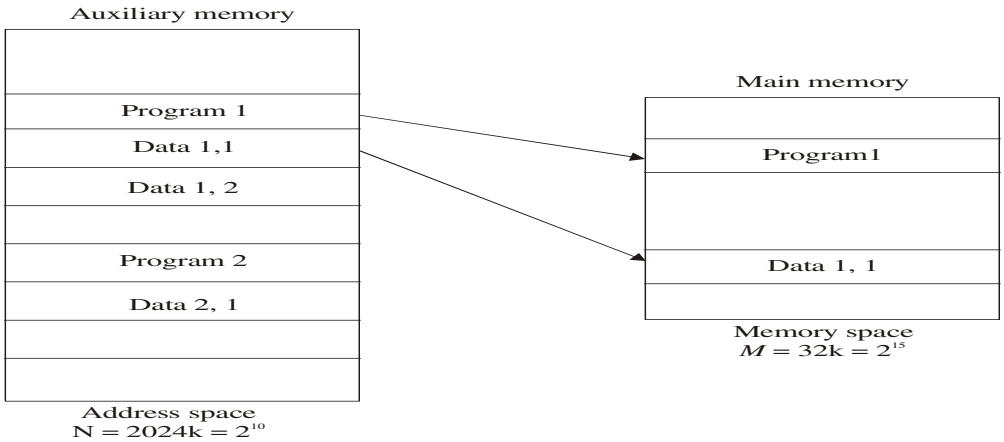
function. The logic of the priority encoder is such that if two or more inputs arrive at the same time, the input having the highest priority will take precedence. The truth table of a four-input priority encoder is given in Table below. The X's in the table designate don't care conditions. Input *Io* has the highest priority; so regardless of the values of other inputs, when this input is 1, the output generates an output

xy = 00. I_1 has the next priority level. The output is 01 if $I_{1=}$ 1 provided that $I_o = 0$ regardless of the values of the other two lower - priority inputs. The output for I_2 is generated only if higher-priority inputs are 0, and so on down the priority level. The interrupt status IST is set only when one or more inputs are equal to 1.The interrupt status IST is cleared to 0 and the other outputs of the encoder are not used, so they are marked with don't-care conditions. This is because the vector address is not transferred to the CPU when IST = 0. The output of the priority encoder is used to form part of the vector address for each interrupt source. The other bits of the vector address can be assigned any value.

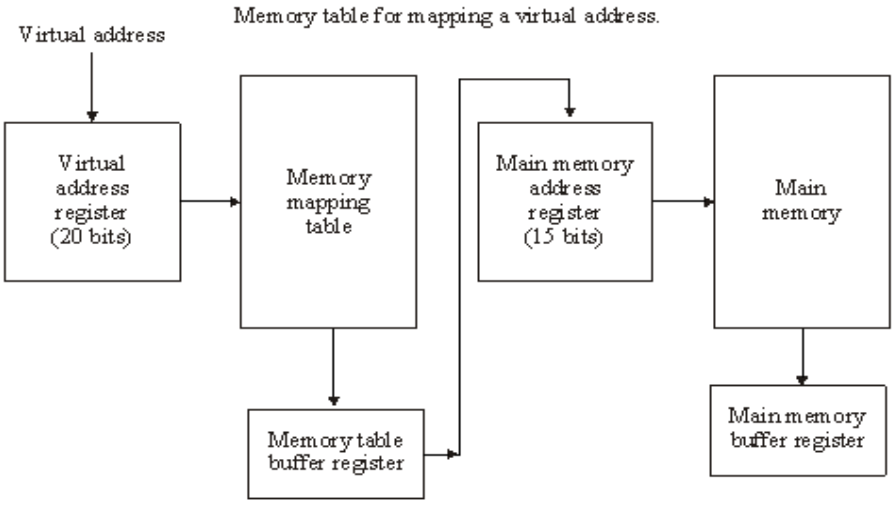
Inputs				Output			Boolean function
I_0	I_1	I_2	I_3	X	Y	IST	
1	X	X	X	0	0	1	
0	1	X	X	0	1	1	$x=I_0'I_1'$
0	0	1	X	1	0	1	$y=I_0'I_1+I_0'I_2'$
0	0	0	1	1	1	1	$(IST)=I_0+I_1+I_2+I_3$
0	0	0	0	X	X	0	

(iii) **Virtual memory**:- Virtual memory is a concept used in some large computer that permit the user to construct programs as through a large memory space were available, equal to the totality of auxiliary memory. Virtual memory is used to give programmes the illusion that they have a very large memory at their disposal, even through the computer actually has a relatively small main memory. A virtual memory system provides a mechanism for translating program generated addresses into correct main memory locations.

In a virtual memory system, programmes are told that they have the total address space at their disposal. The address field of the instruction code has a sufficient number of bits to specify all virtual address. In our example, the address field of an instruction code will consist of 20 bits but physical memory addresses must be specified with only 15 bits. Thus CPU will reference instructions and data with a 20-bit address, but the information at this address must be taken from physical memory because access to auxiliary storage for individual words will be prohibitively long.



Relation between address and memory space in a virtual memory system.



(iv) **Program Control Instruction**: – A program control type of instruction, when executed, may change the address value in the program counter and cause the flow of control to be altered. In other words, program control instructions specify conditions for altering the content of the program counter, while data transfer and manipulation instructions specify conditions of data-processing operations. Some typical program control instructions are list in table below. The branch and jump instructions are used interchangeably to mean the same thing. It is written in assembly language as BR ADR, where ADR is a symbolic name for an address.

Name	Mnemonic
Branch	BR
Jump	JMP
Skip	SKP
Call	CALL
Return	RET
Compare (by subtraction)	CMP
Test (by ANDing)	TST

Branch and jump instructions may be conditional or unconditional. An unconditional branch instruction causes a branch to the specified address without any condition. The conditional branch instruction specifies a condition such as branch, if positive or branch if zero. If the condition is meet, the program counter is loaded with the branch address and the next instruction is taken from this address.
The skip instruction does not need an address field and is therefore a zero-address instruction. A conditional skip instruction will skip the next instruction if the condition is met. If the condition is not met, control proceeds with the next instruction in sequence where the programmer inserts an unconditional branch instruction. The call and return instruction are used in conjunction with subroutines. The compare and text instructions do not change the program sequence directly. The compare instruction performs a subtraction between two operands, the result of the operation is not retained. Similarly, the test instructions performs the

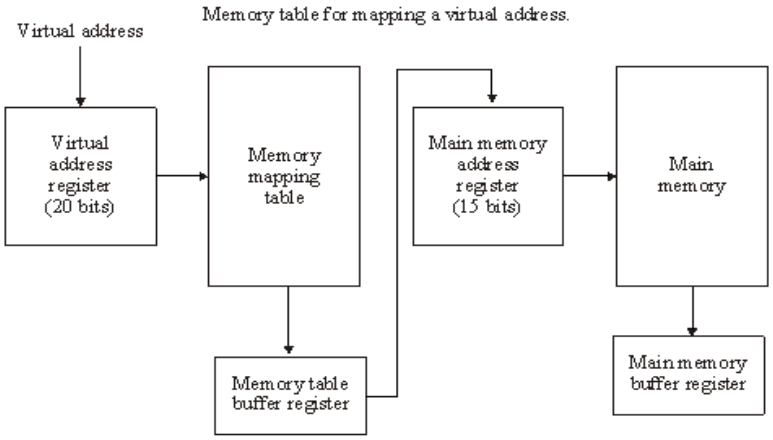
logical AND of two operands and updates certain status bits without retaining the result or changing the operands.

Q.43 Discuss the main features of associative memory Page Table. How does it work in mapping the virtual address into Physical memory address? **(7)**

Ans.

An address generated by user program is called virtual address and the set of virtual addresses make the virtual address space. A main memory address is called a location or physical address and set of such locations are called memory space or physical address space. However, in a system that uses a virtual memory, the size of virtual address space is usually longer than the available physical address space. Consider a computer of main – memory capacity of 32 K words. Since 32K = 2^{15} , 15- bits will be needed to specify a physical address. Suppose the computer has available auxiliary memory for strong 2^{20} words. Let N the address space and M be the memory space. Thus, N = 1024K and M=32K.

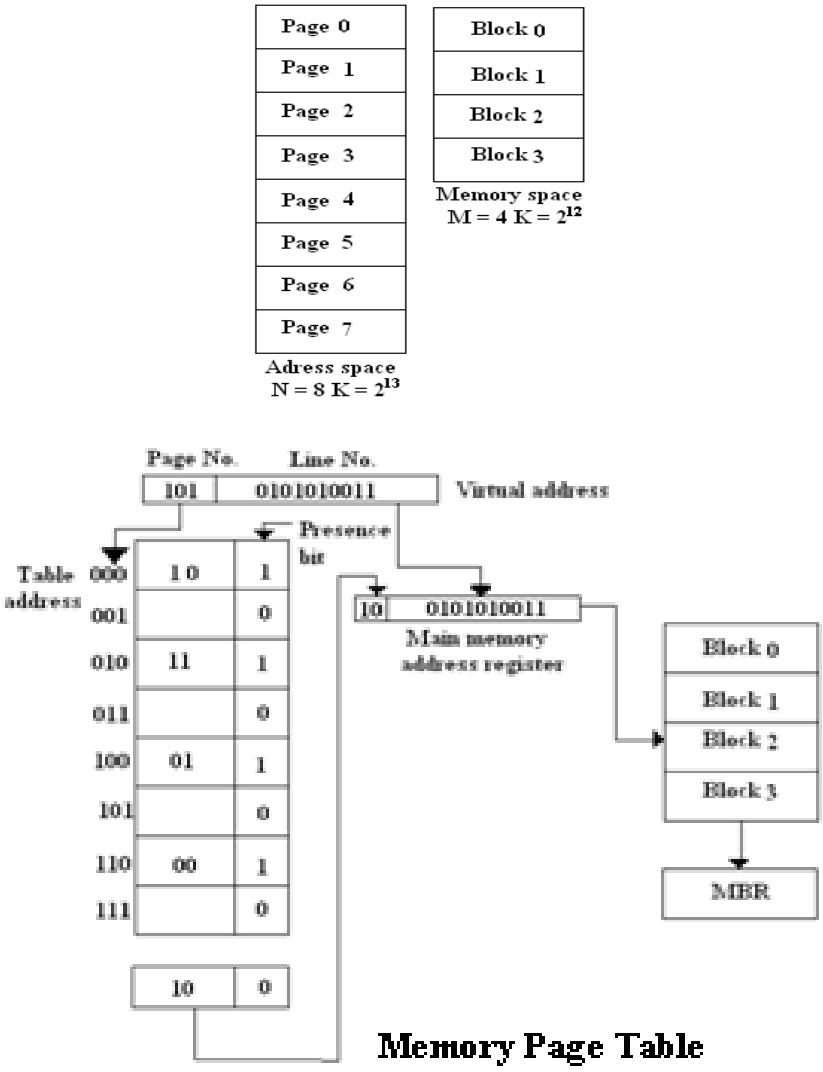
The address bit of the instruction code will consist of 20 bits put physical memory addresses must be specified using only 15 bits. Thus, the CPU will reference instructions and data with 20- bit addresses, but the information at this address must be taken from physical memory rather than auxiliary memory. Thus, it is required to map a virtual address of 20 bit to a physical address of 15 bit. For this a memory mapping table is needed which is shown in the fig. below. This mapping is a dynamic operation and every address is translated immediately as a word is referenced by CPU. The mapping table can be stored in main memory.



Address mapping can be further simplified if the information in address space and memory space can be divided into groups of equal size. The address space is broken into groups of equal size known as page and the memory space is broken into groups of same size known as blocks. These blocks can range from 64 to 4096 words. If a page or block consists of 1K words than the address space of 1024K consists of 1024 pages and the memory space of 32K consists of 32 blocks. Consider a computer with an address space of 8K and memory space of 4K. Thus, if the group of 1K words, we

have 8 pages and 4 blocks. This division of address space and memory space is shown in figure.

Every address generated by the CPU is divided into two parts: a page number address and a line within the page. In a computer with 2^p words per page, p bits are used for line-address and remaining high- order bits of the virtual address specify page number. Virtual address is 13 bits as in fig. Each page consists of 1K words i.e. 2^{10} words. Thus, 10 bits will be used to specify line number address and three high – order bits of the virtual address will specify one of the eight pages of the address space. The line address in address space and memory space is same and hence only mapping required is from a page number to a block number.



The mapping table in a paged system is shown in Figure. The memory page table consists of eight words. The table address denotes the page number and content of words give the block number where that page is stored in main memory. Thus this shows that pages 0, 2, 4 and 6 are stored in main memory in blocks 2, 3, 1 and 0, respectively. The present bit signifies that whether the page is in main memory or not. If presence bit is 1 the page is available in main memory and if it is 0 the page is not available in main

memory. When a CPU references a word in memory with virtual address of 13 bits, the lower – order 10 bits specifies the line number and three high – order specifies a page number which is also used as an address of the memory-page table. Then the content of the word at this address is read-out into the memory table buffer register along with the presence bit. If presence bit is 1, the content thus read (the block number) is transferred into main memory address register and the line number is also transferred into the main memory address register as 10 lower order bits. A read signal thus transfers the content to the main memory buffer register, to be used by the CPU. If the presence bit is 0, the content of the word referenced by the CPU does not reside in main memory. Then a call to operating system is generated to fetch the required page from auxiliary memory to main memory before resuming computation.

Q.44 A Virtual memory has a Page Size of 1K words. There are eight Pages and four blocks. The associative memory page table contains the following entries.

Page	Block
6	0
1	1
4	2
0	3

Give the list of virtual addresses in decimal that will cause a Page fault if used by CPU. **(6)**

Ans.
The pages which are not in main memory are:

Page	Address	Address that will cause fault
2	2K	2048-3071
3	3K	3072-4095
5	5K	5120-6143
7	7K	7168-8191

Q.45 How LRU technique is implemented ? **(3)**

Ans.
The LRU policy is more difficult to implement but has been more attractive on the assumption that the least recently used page is a better candidate for removal than the least recently used page is a better candidate for removal than the least recently loaded page as in FIFO. The LRU algorithm can be implemented by associating a counter with every page that is in the main memory. When a page is referenced, its associated counter is set to zero. At fixed interval of time, the counters associated with all pages presently in memory are incremented by 1. The least recently used page in the page with the highest count. The counters are often called aging registers, as their count indicates their age, that is how long ago their associated pages have been referenced.

Q.46 What is cycle stealing DMA operation? **(3)**

Ans.

Cycle Stealing: In this method, the DMA controller transfers one data word at a time, after which it must return control of the buses to the CPU. The CPU merely delays its operation for one memory cycle to allow the direct memory input/output transfer to 'Steal' one memory cycle.

Q.47 What do you understand by the term micro-operation. Explain Register and Arithmetic types of micro-operation. Show the hardware realization of decrement micro-operation.

i.e. $T1: X \leftarrow X - 1$ **(6)**

Ans.

A miocrooperation is an elementary operation performed with the data stored in registers. The operations executed on data stored in registers are called microoperations. A microoperation is an elementary operation performed on the information stored in one or more registers. The result of the operation may replace the previous binary information of a register or may be transferred to some another registers. Examples are clear, shift, count, load etc. A bidirectional shift register is capable of performing the shift right and shift left microoperations.

The microoperations must often encounter in digital computers are classified into following categories:

(1) Register transfer microoperations transfer binary information from one register to another.

(2) Arithmetic microoperations perform arithmetic operations on numeric data stored in registers.

(3) Logic microoperations perform bit manipulation operations on non numeric data stored in registers.

(4) Shift microoperations perform shift operations on data stored in registers.

The register transfer microoperation does not change the information content when the binary information moves from source register to destination register.

Arithmetic microoperations are those microoperations that are used to perform arithmetic operations. The basic arithmetic microoperations are addition, subtraction, increment, decrement and shift.

Let the arithmetic microoperations defined by the statement

$$R3 \leftarrow R1+R2$$

specifies an add microoperation. It states that add the content of register R1 and that of register R2 and stored the result in register R3. Thus to implement this statement with hardware we need three registers and the digital component that performs the addition operation.

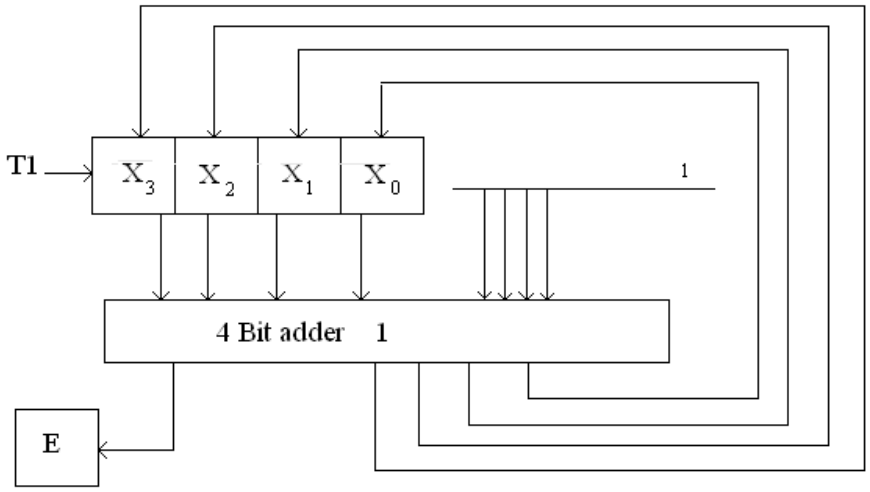
Subtraction is basically implemented through complementation and addition and can be specified by the statement

$$R \leftarrow A+B'+1$$

where B' is the 1's complement of B. When we add 1 to the 1's complement it will give 2's complement of B and adding A to the 2's complement of B gives A minus B(A-B). The increment and decrement microoperations are implemented with the help of combinational circuit or with a binary up-down counter. They are symbolized by plus-one or minus-one operation executed on the contents of a register.

The multiplication operation and division are valid arithmetic operations. Multiplication operation is basically implemented with a sequence of subtract and shift microoperations.
Hardware realization of decrement microoperation

T1: X ← X -1



Q.48 A Register 'A' holds on 8-bit binary number 11011001. Determine the operand 'B' and the logic micro-operation to be performed in order to change the value of 'A' to
(i) 01101101
(ii) 11111101 **(4)**

Ans.

A	=	11011001
B	=	10110100
A ⊕ B		01101101
A	=	11011001
B	=	11111101
A ∨ B		11111101

Q.49 Give the flow chart of division of two signed magnitude data. Discuss the logic of the flow chart. **(10)**

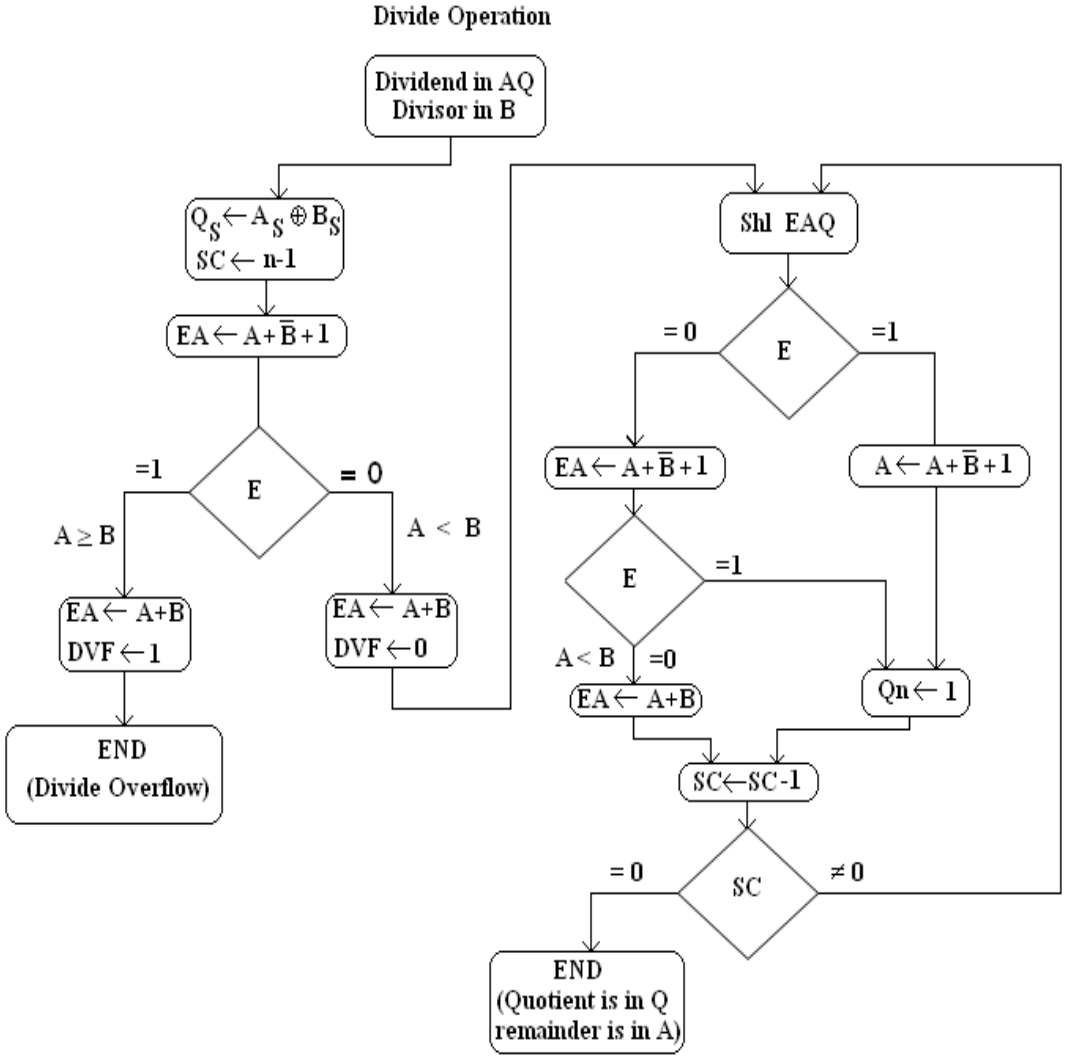
Ans.

The dividend is in A and Q and the divisor in B. The sign of the result is transferred into Q_s to be part of the quotient. The operands are transferred to registers from a memory unit that has words of n bits. Since an operand must be stored with its sign, one bit of the word will be occupied by the sign and the magnitude will consist of n-1 bits. A divide overflow condition is tested by subtracting the divisor in B from half of the bits of the dividend stored in A. If A ≥ B, the divide overflow flip flop DVF is set and the operation is terminated prematurely. If A<B, no divide overflow occurs so the value of the dividend is restored by adding B to A.

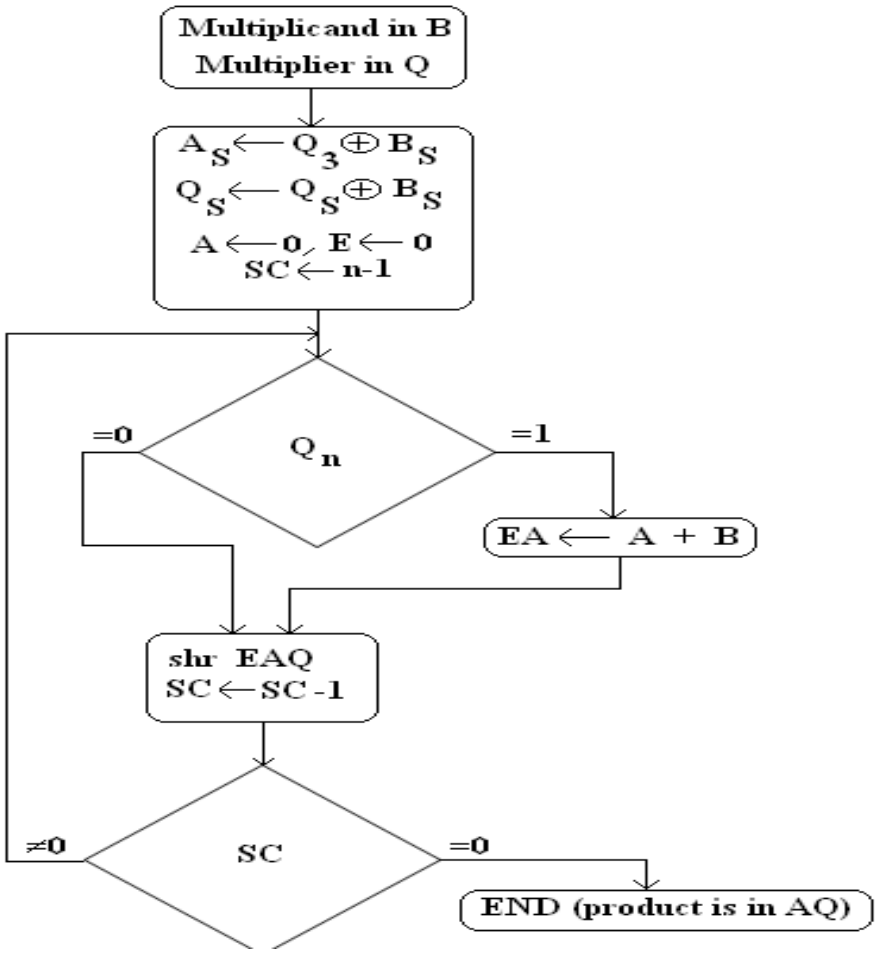
The division of the magnitudes starts by shifting the dividend in AQ to the left with the high order bit shifted into E. If the bit shifted into E is 1, then EA>B because EA consists of a 1 followed by n-1 bits while B consists of only n-1 bits. In this case B must be subtracted from EA and 1 inserted into Q_n for the quotient bit. Since register A is missing the high order bit of the dividend (which is in E), its value EA -2^{n-1}. Adding to this value the 2's complement of B results in

$$(EA-2^{n-1})+(2^{n-1}-B)=EA-B$$

The carry from this addition is not transferred to E if we want E to remain a 1.
If the shift left operation inserts a 0 into E, the divisor is subtracted by adding its 2's complement value and the carry is transferred into E. If E = 1, it signifies that $A \geq B$; therefore Q_n is set to 1. If E=0, it signifies that A<B and the original number is restored by adding B to A. This process is repeated again and again with register A holding the partial remainder. After n-1 times, the quotient magnitude is formed in register Q and the remainder is found in register A. The quotient sign is in Q_S and the sign of the remainder in A_S is the same as the original sign of the dividend.



Flow chart for divide operation.



Multiplication of (-3) with (+4)

Multiplicand (-3) B=101	E	A	Q	SC
Multiplier in Q	0	000	100	11
Q_n=0; shift right EAQ	0	000	010	10
Q_n=0; shift right EAQ	0	000	001	01
Q_n=1; Add B		101		
First partial product		101		
Shift right EAQ	0	010	100	00

Final product in AQ = 10100

Q.50 Convert the following arithmetic expression from reverse polish notation to infix notation:
ABXYZ+*–/
Write a program using three address instruction to evaluate the same. **(4)**

Ans.
The given expression is ABXYZ + * - /
ABXYZ + * - / = A / (BXYZ+*-)
= A / [B-(XYZ+*)]

= A / [B-{X*(YZ+)}]
= A / [B – {X* (Y+Z)}]
The arithmetic expression is
$\delta = A / \{X * (Y + Z)\}]$
PROGRAM USING THREE ADDRESS INSTRUCTIONS
ADD R1, Y, Z R1← M[Y]+M[Z]
MUL R1, X, R1 R1← M[X] * R1
SUB R1, B, R1 R1 ← M[B]-R1
DIV δ, A, R1 M[δ] ← M[A] / R1

Q.51 What are the interrupts? Explain different types of interrupts. **(8)**

Ans.
An interrupt signal is a signal sent by an input output interface to CPU when it is ready to send information to the memory or receive information from the memory.
The word interrupt is used from any exceptional event that causes the CPU to temporarily transfer the control from its current program to another program, an interrupt handler which will service the interrupt. This program is known as Interrupt Service Routine (ISR). Interrupts are the primary means by which input/ output devices obtain the services of the CPU. Various sources, internal and external to the CPU can generate interrupts. Input output interrupts are external requests to the CPU to initiate or terminate an input output operation, such as data transfer with a hard disk.
Interrupts are produced by hardware or by software error-detection circuits that invoke error-handling routines within the operating system. An attempt by an instruction to divide by 0 is an example of software generated interrupt. A power-supply failure can generate an interrupt that requests interrupt handler to save critical data about the system's state.
The interrupt is initiated by a signal generated by an external devices or a signal generated externally by the CPU. When CPU receives an interrupt signal from peripherals it stops executing the current program, saves the content or statue of various registers in the stack and then CPU executes a sub routine in order to perform the specific task requested by the interrupt.
The interrupt generated by special instructions are called software interrupts and they are used to implement system services.
In general, interrupts can be classified in the following three ways:

- Hardware and Software interrupt
- Vectored and Non-vectored interrupts
- Maskable and Non-maskable interrupts.

Hardware interrupt is a type of interrupt generated either externally by the hardware devices such as input output ports, key board, and disk drives etc or internally by the microprocessor. External hardware interrupts are used by devices to request attention from CPU. Internal hardware interrupts are generated by the CPU to control events.
The software interrupts are program instructions. These instructions are inserted at desired location in a program. A program generated interrupt, also called trap, which stops current processing in order to request a service provided by the CPU. While running a program, if software interrupt instruction is encountered the CPU initiates

an interrupt. For example, a program might generate a software interrupt to read input from keyboard.

When interrupt signal is generated the CPU responds to the interrupt signal by storing the return address from program counter into memory stack and then control is transfer or branches to the service routine that processes the interrupt. The processor chooses the branch address of the service routine in two different ways. One is called vectored interrupt and the other is called non-vectored interrupt.

In non-vectored interrupt, the branch address is assigned to a fixed location in memory. In vectored interrupt, the source that initiated the interrupt supplies the branch information. This information is called the interrupt vector.

In certain situations it may be desired that some of the several interrupts should not occur while CPU is busy in performing some important task. This can be done by masking. The interrupt that can be masked off is called maskable interrupt. When interrupts are masked, they are not withdrawn. They remain pending. Once the masking is removed, interrupt takes place.

Certain interrupts have to be serviced without delay; else something serious damage may be caused to program, data or results. Thus the CPU must have means to distinguish between urgent and non-urgent interrupts. Such interrupts are known as non-maskable interrupts and CPU does not mask (ignore) them. For example, memory failure, that must be serviced immediately.

Q.52 Explain the significance of different fields of an instruction with an example. **(4)**

Ans.

An instruction is a command given to a computer to perform a specified operation on some given data and the format in which the instruction is specified is known as Instruction format. The most common fields found in the instruction are:-

(i) An operation code field that specifies the operation to be performed. It is known as opcode field.

(ii) An address field that designates the registers address and/or a memory addresses.

(iii) A mode field that specifies the way the operands or the effective address is determined.

For example,

ADD R1, R0, ADD is the opcode and R1, R0 are the address field. Operations specified by computer instructions are executed on some data stored in memory or some registers. Operands residing on memory are specified by register address. The instruction may be of several different lengths containing different number of addresses. The number of address fields in the instruction format of a computer system depends on the internal architecture/organization of registers.

Q.53 The 8-bit registers A, B, C & D are loaded with the value $(F2)_H$, $(FF)_H$, $(B9)_H$ and $(EA)_H$ respectively.

Determine the register content after the execution of the following sequence of micro-operations sequentially.

(i) $A \leftarrow A + B$, $C \leftarrow C + Shl(D)$

(ii) $C \leftarrow C \wedge D$, $B \leftarrow B + 1$.

(iii) $A \leftarrow A - C$.

(iv) $A \leftarrow Shr(B) \oplus Cir(D)$ **(8)**

Ans.

$A = (F2)_H = (11110010)_2$
$B = (FF)_H = (11111111)_2$
$C = (B9)_H = (10111001)_2$
$D = (EA)_H = (11101010)_2$

(i) $A \leftarrow A + B, C \leftarrow C + shl(D)$

A+B = 11110010
+11111111

11110001 $= (F1)_H$

shl(D) = shl(11101010) = 11010100 $= (D4)_H$

C + shl (D) = 10111001
+11010100

10001101 $= (8D)_H$

After these microoperations the content of A, B, C, and D are $(F1)_H$, $(FF)_H$, $(8D)_H$ and $(D4)_H$ respectively.

(ii) $C \leftarrow C \wedge D, B \leftarrow B+1$

C ^ D = 10001101
^11010100

10000100 $= (84)_H$

B+1 = 11111111
+ 1

00000000 $= (00)_H$

After these microoperations the content of A, B, C, and D are $(F1)_H$, $(00)_H$, $(84)_H$ and $(D4)_H$ respectively.

(iii) $A \leftarrow A - C$

A – C = 11110001
-10000101

01101100 $= (6D)_H$

After this microoperation, the content of A, B, C, and D are (6D)H, (00)H, (84)H and (D4)H respectively.

(iv) $A \leftarrow shr(B) \oplus Cir(D)$

shr(B) = shr(00000000) = 00000000 $= (00)_H$

Cir(D) = Cir (11010100) = (01101010) $= (6A)_H$

shr(B) ⊕ Cir(D) = 00000000
⊕ 01101010

01101010 $= (6A)_H$

After this microoperation, the content of A, B, C, and D are $(6A)_H$, $(00)_H$, $(84)_H$ and $(6A)_H$ respectively.

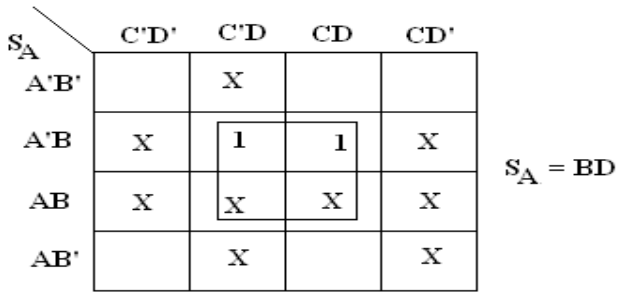
Q.54 Design a synchronous self starting counter using S-R flip flops for counter the sequence 0, 2, 3, 5, 8, 7, 15, 12, 11, 10 & repeat. **(8)**

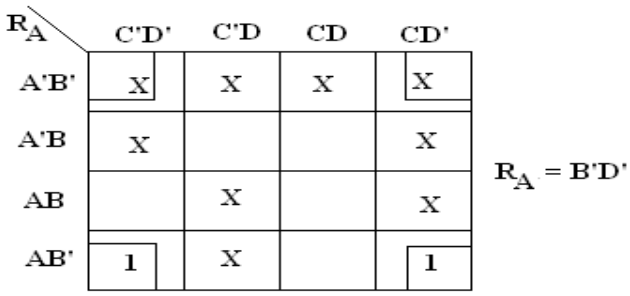
Ans.

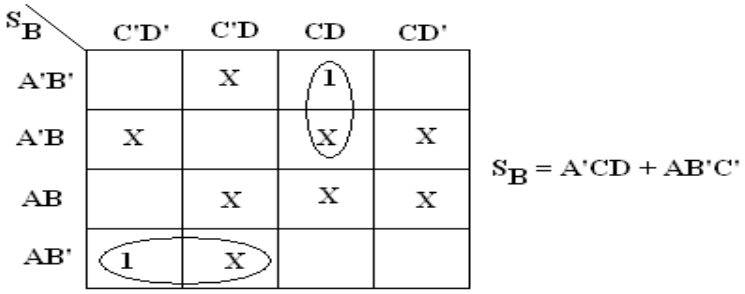
Excitation table for counter using S-R Flip Flop

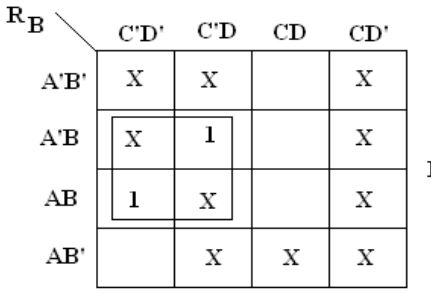
A	B	C	D	S_A	R_A	S_B	R_B	S_C	R_C	S_D	R_D
0	0	0	0	0	x	0	x	1	0	0	X
0	0	1	0	0	x	0	x	x	0	1	0
0	0	1	1	0	x	1	0	0	1	x	0
0	1	0	1	1	0	0	1	0	x	0	1
1	0	0	0	0	1	1	0	1	0	1	0
0	1	1	1	1	0	x	0	x	0	x	0
1	1	1	1	x	0	x	0	0	1	0	1
1	1	0	0	x	0	0	1	1	0	1	0
1	0	1	1	x	0	0	x	x	0	0	1
1	0	1	0	0	1	0	x	0	1	0	x

The K – maps are as follows:-



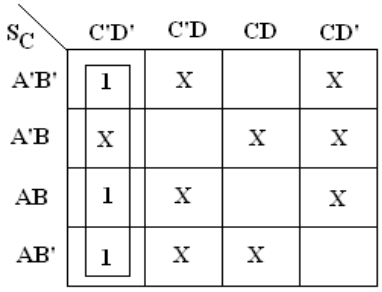






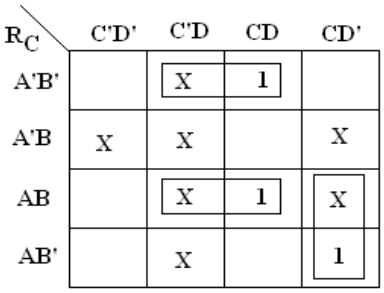
R_B	C'D'	C'D	CD	CD'
A'B'	X	X		X
A'B	X	1		X
AB	1	X		X
AB'		X	X	X

$R_B = BC'$



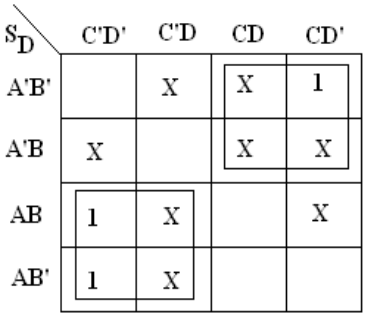
S_C	C'D'	C'D	CD	CD'
A'B'	1	X		X
A'B	X		X	X
AB	1	X		X
AB'	1	X	X	

$S_C = C'D'$



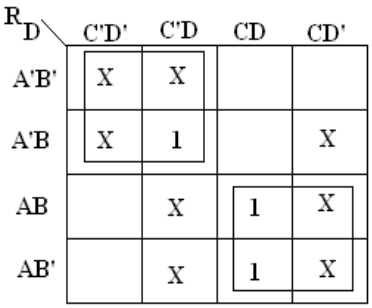
R_C	C'D'	C'D	CD	CD'
A'B'		X	1	
A'B	X	X		X
AB		X	1	X
AB'		X		1

$R_C = A'B'D$
$= + A B D + A C D'$
$= (A \odot B) D + A C D'$



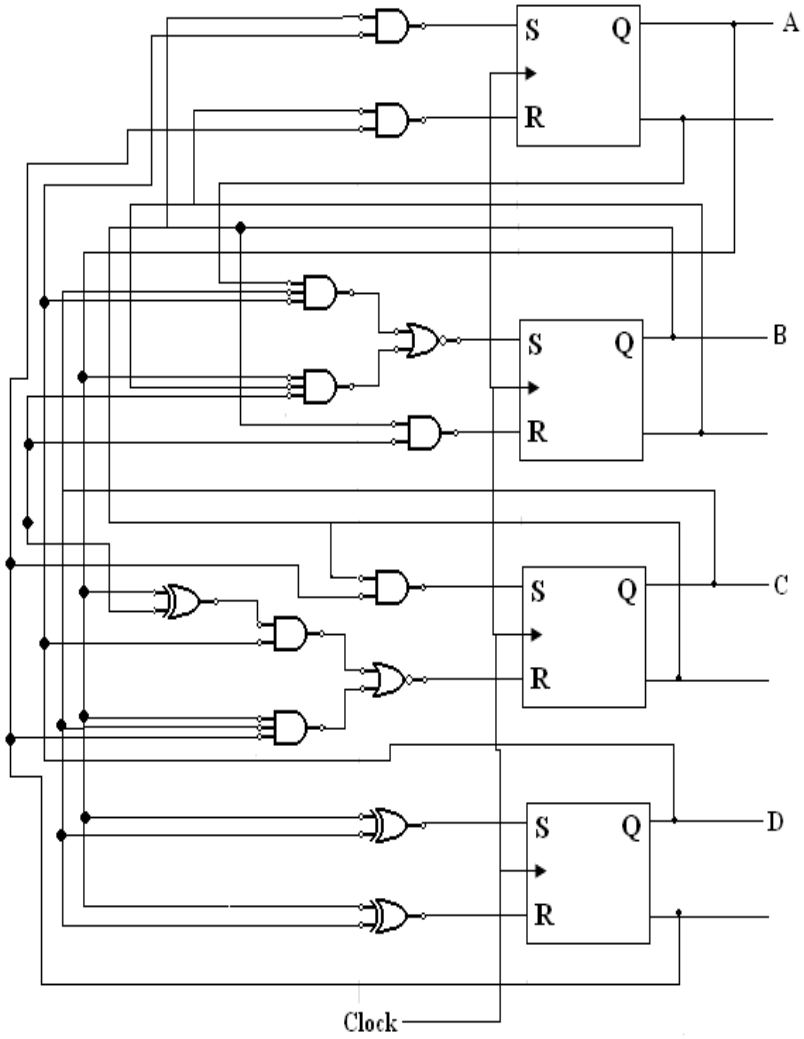
S_D	C'D'	C'D	CD	CD'
A'B'		X	X	1
A'B	X		X	X
AB	1	X		X
AB'	1	X		

$S D = AC' + A'C$
$= A \oplus C$



R_D	C'D'	C'D	CD	CD'
A'B'	X	X		
A'B	X	1		X
AB		X	1	X
AB'		X	1	X

$RB = A'C' + AC$
$= A \odot C$



Logic Circuit

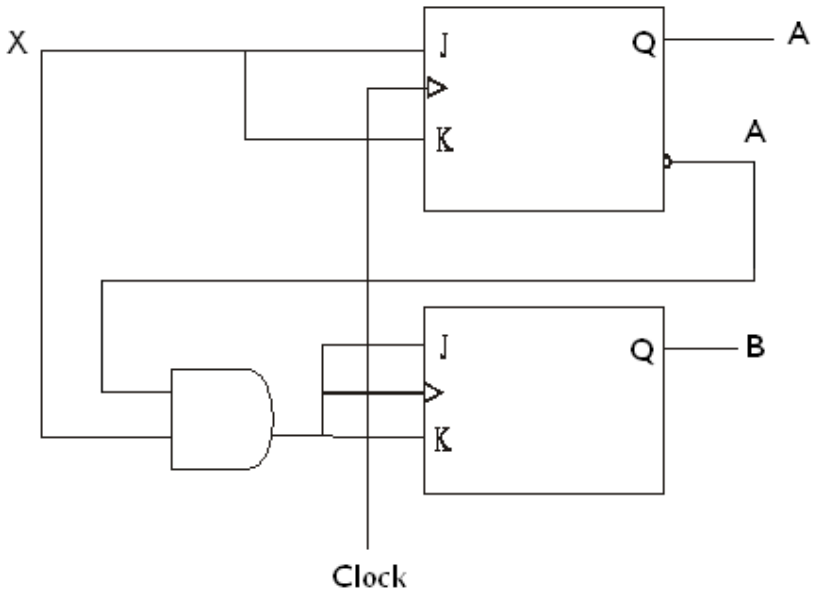
Given, f (b, a, c) = Σm (1,3,5,6,7,11,13,14) and don't care M_4, M_9, M_{10}

	a'c'	a'c	ac	ac'
d'b'		1	1	
d'b	X	1	1	1
db		1		1
db'		X	1	X

F = a'c + d'c + db'a + bac'

Q. 55 Design a two bit countdown counter with two flip-flop and one input x.
When x = 0, the state of the flip-flop does not change. When x = 1, the state sequence is 11, 10. 01. 00, 11 and repeats.

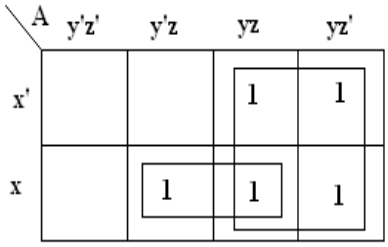
Ans.
$J_A = K_A = x$
$J_B = K_B = a'x$
Two bit countdown counter



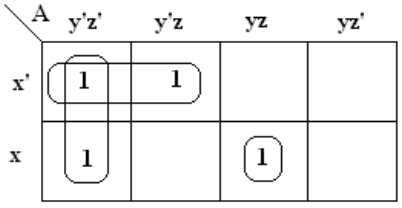
Q. 56 Design a combinational circuit with three inputs x, y, z and three outputs A, B, C. When the binary input is 0, 1, 2, or 3 the binary output is two greater than the output. When the binary input is 4, 5, 6 and 7, the binary output is one less than the input.

Ans.

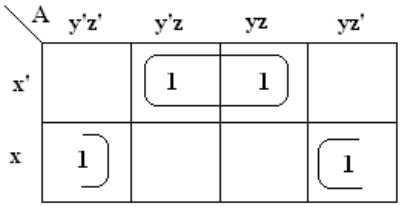
x	y	z	A	B	C
0	0	0	0	1	0
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	1	0	1
1	0	0	0	1	1
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	1	1	0



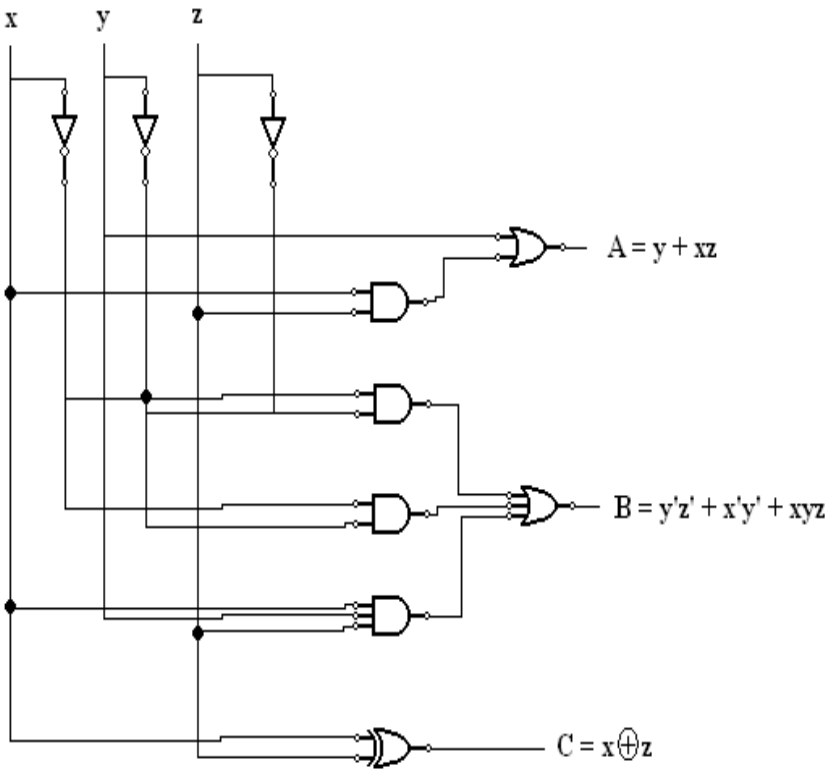
A = y + xz



B = y'z' + x'y' + xyz

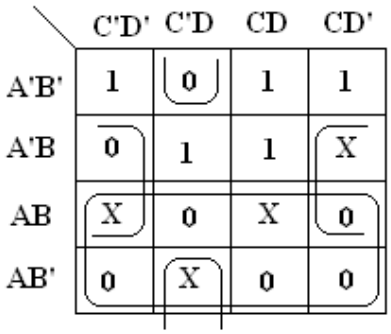


C = x'z + xz' = x⊕z



Q. 57 If $Y = (M_0, M_2, M_3, M_5, M_7) + (M_6, M_9, M_{12}, M_{15})$ where 'd' stands for don't care. Express the Boolean expression is product of sum form and also show the k-map for that product of sum form.

Ans.



F = A'(B'+D) (B + C + D')

Q. 58 A RAM chip 4096 x 8 bits has two enable lines. How many pains are needed for the integrated circuit package of Draw a block diagram and label all input and outputs pins of the RAM. What is the main feature of random access memory?

Ans.
RAM chip is of 4096 x 8. Hence, 12 bits for inputs. 2 for enable lines and 8 bits for outputs. Hence, total 22 pins are needed.

MAIN MEMORY:-

The main memory is the central storage unit in a computer system. It is a relatively large and fast memory used to store programs and data during the computer operation. The principal technology used for the main memory is based on semiconductor integrated circuits. Integrated circuit RAM chips are available in two possible operating modes, *static* and *dynamic.*

The static RAM consists essentially of internal flip-flop that store the binary information. The stored information remains valid as long as power is applied to be unit. The dynamic RAM stores the binary information in the form of electric charges that are applied to capacitors. The capacitors are provided inside the chip by MOS transistors. The stored charge on the capacitors tend to discharge with time and the capacitors must be periodically recharged by refreshing the dynamic memory. Refreshing is done by cycling through the words every few milliseconds to store the delaying charge. The dynamic RAM offers reduced power consumption and large storage capacity in a single memory chip.

The static RAM is easier to use and has shorter read and write cycles.

Most of the main memory in a general-purpose computer is made up of RAM integrated circuit chips, but a portion of the memory may be constructed with ROM chips. Originally, RAM was used to refer to a random-access memory, but now it is used to designate a read/write memory to distinguish it from a ready-only memory, although ROM is also random access. RAM is used for storing the bulk of the programs and data that are subject to change. ROM is used for storing programs

that are permanently resident in the computer and for tables of constants that do not change in value once the production of the computer is completed.

Q.59 The RAM IC as described above is used in a microprocessor system, having 16 bit address line and 8-bit data line. It's enable-1 input is active when A_{15} and A_{14} bits are 0 & 1 and enable-2 input is active when A_{13}, A_{12} bits are 'X' and 'O'. What shall be the range of addresses that is being used by the RAM. **(4)**

Ans.

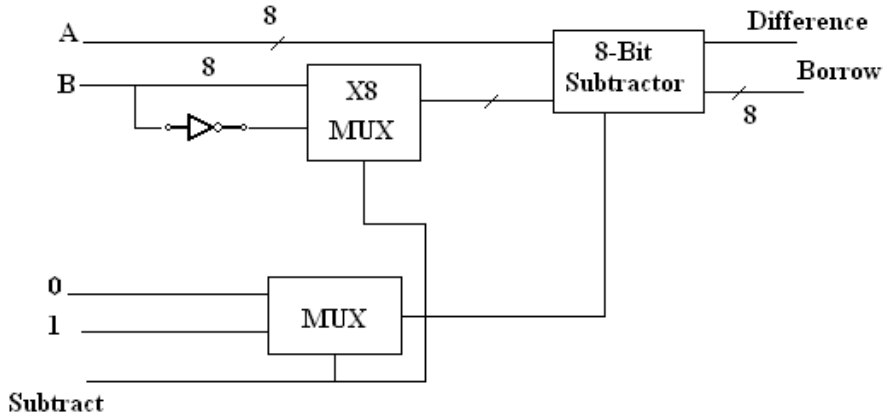
A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	**1**	**x**	**0**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**

Thus, its range of address being used by RAM is 4000 - 6PFF.

Q. 60 Implement a full subtractor logic by using multiplexer.

Ans.

Full subtractor circuit

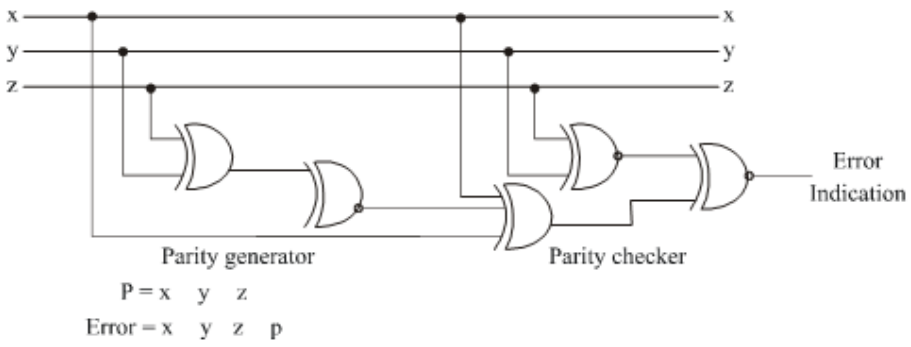


$$\text{Difference} = A \oplus B \oplus P$$

$$\text{Borrow} = B.P + A'.(B + P)$$

A	B	P	Difference	Borrow
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

Q. 61 Derive the circuit for a 3-bit parity generator and 4-bit parity checker using an even parity bit.



Q. 62 What is microoperation? Give suitable examples of some four types of microoperations.

Ans.

A microoperation is an elementary operation performed with the data stored in registers.

1) Register transfer microoperation transfer binary information from one register to another.
2) Arithmetic microoperations perform arithmetic operation on numeric data stored in registers.
3) Logic micro operation performs bit manipulation operation on numeric data stored in register.
4) Shift microoperation performs shit operation on data stored in registers.

Example:-

Arithmetic microoperation

$R_3 \longleftarrow R_1 + R_2$

Subtract microoperation

$R_3 \leftarrow R_1 + \overline{R_2} + 1$

Logic microoperation

$P : R_1 \leftarrow R_2 \oplus R_3$

$R_4 \leftarrow R_5 \vee R_6$

Shift Microoperation

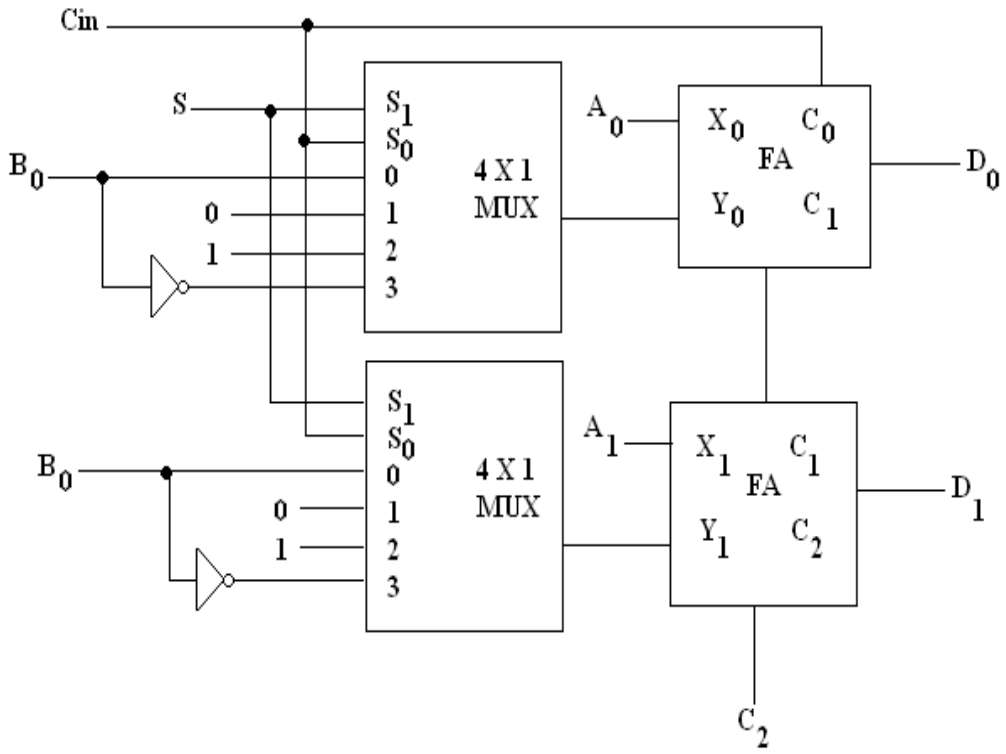
$R_1 \leftarrow ShlR_1$

$R_2 \leftarrow ShrR_2$

Q. 63 Give the hardware realization of 4-bit arithmetic circuit capable of doing addition, subtraction, increment, decrement etc. Give the function table and explain its operation.

Ans.
Arithmetic Circuit

S	Cin	X	Y	
0	0	A	B	(A+B)
0	1	A	0	(A+1)
1	0	A	1	(A-1)
1	1	A	B	(A-B)



Q. 64 Give the comparison between & examples of hardwired control unit and micro programmed control unit.

Ans.
Comparison between Hardwired and microprogrammed control unit

Characteristics	Hardwired Control	Microprogrammed Control
(1) Speed	Fast	Slow
(2) Implementation	Hardware	Software
(3) Ability to handle large/complex instruction sets	Somewhat difficult	Easier
(4) Design process	Difficult for more operation	Easy
(5) memory	No memory used	Control memory used.
(6) Flexibility	No flexibility	More flexibility

Q.65 What do you mean by Fetch cycle, instruction cycle, machine cycle, interrupt acknowledgement cycle.

Ans.
The execution of an instruction may itself involve a number of steps. The two stages of fetch and execution as follows.
The instruction fetch is a common fraction instruction from location is memory. The instruction execution may involve several operations and depends on the nature of the instructions. The instruction cycle is referred to as the fetch cycle and execute cycle. Interrupt acknowledgement cycle that I_1 regardless of the values of the other two lower-priority inputs. The output for I_2 is generated only if higher-priority inputs are 0_1 and on down the priority level.

Inputs				Outputs			
I_0	I_1	I_2	I_3	x	y	IST	Boolean Function
1	x	x	x	0	0	1	
0	1	x	x	0	1	1	$x = I'_0 I'_1$
0	0	1	x	1	0	1	$y = I'_0 I_1 + I'_0 I_2$
0	0	0	1	1	1	1	$(IST) = I'_0 + I_1 + I'_2 I_3$

The interrupt status IST is set only when one or more inputs are equal to 1. If all inputs are I_0 IST is cleared to 0 and the other outputs of the encoder are not used, so they are marked with don't care conditions. This is because the vector address is not transferred to the CPU when IST = 0.
The output of the priority encoder is used to form part of the vector, address for each interrupt source. The other bits of the vector address can be assigned any value.

Interrupt Cycle:-
The interrupt enable flip-flop *IEN* shown in Fig. 11-14 can be set or cleared by program instructions. When *IEN* is cleared, the interrupt request coming from IST is neglected by the CPU. The program-controlled *IEN* bit allows the programme to choose whether to use the interrupt facility. If an instruction to clear *IEN* has been inserted in the program, it means that the user does not want his program to be interrupted. An instruction to set *IEN* indicates that the interrupt facility will be used while the current program is running. Most computers include internal hardware that clears *IEN* to 0 every time an interrupt is acknowledged by the processor.
At the end of each instruction cycle the CPU checks *IEN* and the interrupt signal from *IST*. If either is equal to 0, control continues with the next instruction. It both *IEN* and *IST* are equal to 1; the CPU goes to an interrupt cycle. During the interrupt cycle the CPU performs the following sequence of microoperations:

SP ← SP – 1	Decrement stack pointer
M [SP] ← PC	Push PC into stack
INTACK ← 1	Enable interrupt acknowledge
PC ← VAD	Transfer Vector address to PC
IEN ← 0	Disable further interrupts

The CPU pushes the return address from PC into the stack. It then acknowledges the interrupt by enabling the *INTACK* line. The priority interrupt unit responds by placing a unique interrupt vector into the CPU data bus. The CPU transfers the vector address into *PC* and clears *IEN* prior to going to the next fetch phase. The instruction read from memory during the next fetch phase will be the one located at the vector address.

Q. 66 Explain in brief how a digital computer system works in a interrupt driven input-output programming.

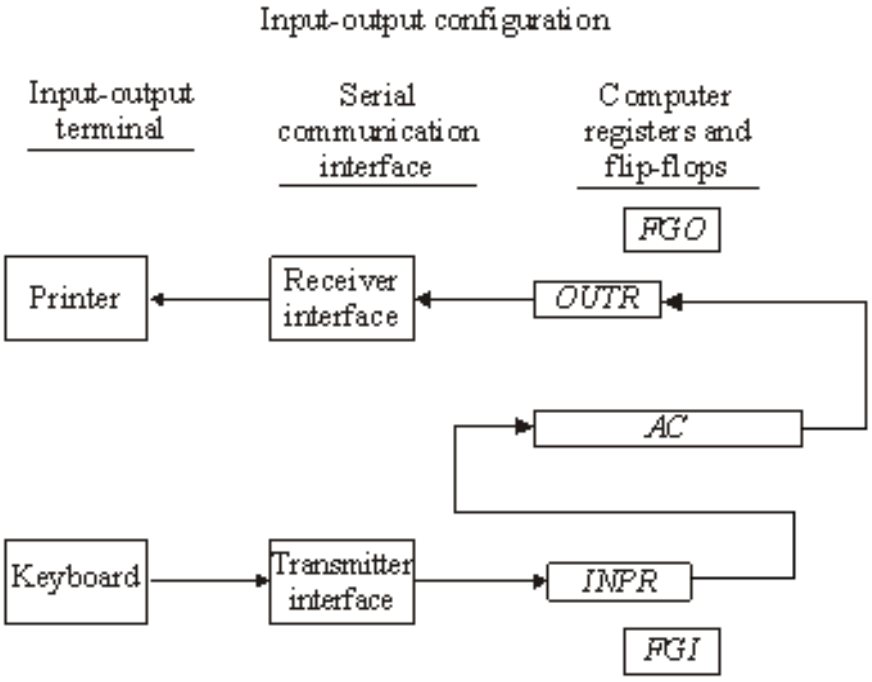
Ans.

A computer can serve no useful purpose unless it communicates with the external environment. Instructions and data stored in memory must come from some input device. Computational results must be transmitted to the user through some output device. Commercial computers include many types of input and output devices. To demonstrate the most basic requirements for input and output communication, we will use as an illustration a terminal unit with a keyboard and printer.

Input-Output Configuration:-

The terminal sends and receives serial information. Each quantity of information has eight bits of an alphanumeric code. The serial information from the keyboard is shifted into the input register *INPR*. These two registers communicate with a communication interface serially and with the AC in parallel. The input-output configuration is shown in. The transmitter interface receives serial information from the keyboard and transmits it to *INPR*. The receiver interface receives information from OUTR and sends it to the printer serially. The input register INPR consists of eight bits and holds alphanumeric input information. The 1-bit input flag *FGI* is a control flip-flop. The flag bit is set to 1 when new information is available in the input device and is cleared to 0 when the information is accepted by the computer.

Output register:-



The output register *OUTR* works similarly but the direction of information flow is reversed. Initially, the output flag *FGO* is set to 1. The computer checks the flag bit; if it is 1, the information from *AC* is transferred in parallel to *OUTR* and *FGO* is cleared to 0. The output device accepts the coded information, prints the corresponding character, and when the operation is completed, it sets *FGO* to 1. The computer does not load a new character into *OUTR* when *FGO* is 0 because this condition indicates that the output device is in the process of printing the character.

Input-Output Instructions. Input and output instructions are needed for transferring information to and from *AC* register, for checking the flag bits, and for controlling the interrupt facility. Input-output instructions have an operation code 1111 and are recognized by the control when $D_7 = 1$ and I = 1. The remaining bits of the instruction specify the particular operation. The control functions and microoperations for the input-output instructions are listed. These instructions are executed with the clock transition associated with timing signal T_3. Each control function needs a Boolean relation D_7IT_3, which we designate for convenience by the symbol *p*. The control function is distinguished by one of the bits in *IR* (6-11). By assigning the symbol B_i to bit i of *IR,* all control functions can be denoted by pBi for i=6 through 11. The sequence counter SC is cleared to O when $p=D_7IT_3 = 1$

Input-Output Instructions

$D_7IT_3 = p$ (common to all input-output instructions)			
IR (i) = B_i [bit in *IR*(6-11) that specifies the instructions]			
	p:	SC ← 0	Clear SC 0
	INP	pB_{11}: AC(07) ← INPR, FGI ← 0	Input character
	OUT	pB_{10}: OUTR ← AC(0-7), FGO ← 0	Output character
	SKI	pB_9: If (FGI=1) then (PC ← PC+1)	Skip on input flag
	SKO	pB_8: If(FGO=1) then (PC ← PC+1)	Skip on output flag
	ION	pB_7: IEN ← 1	interrupt enable on
	IOF	pB_6: IEN ← 0	interrupt enable off

Q.67 Design a CPU that meets the following specifications:
It can access 64 words of memory, each word being 8-bit long. The CPU does this by outputing a 6-bit address on its output pins A [5.....0] and reading in the 8-bit value from memory on inputs D [7.....0]. It has one 8-bit accumulator, 8-bit address register, 6-bit program counter, 2-bit instruction register, 8-bit data register. The CPU must realize the following instruction set.

Instruction	**Instruction code**	**operation**		
AND	00AAAAAA	AC ←AC+M(AAAAAA)		
JMP	01AAAAAA	Go to AAAAAA		
ADD	10AAAAAA	AC ←AC + M(AAAAAA)		
INC	11xxxxxx	AC ←AC+1		
Label	**Microoperation**	**CD**	**BR**	**AD**
AND	ORG 16			

	MOP	I	CALL	INDRCT
	READ	U	JMP	NEXT
ANDOP	AND	U	JMP	FETCH
ADD	ORGO			
	MOP	I	CALL	INDRCT
	READ	U	JMP	NEXT
	ADD	U	JMP	FETCH
STORE	ORG8			
	NOP	I	CALL	INDRCT
	ACTRD	U	JMP	NEXT
	WRITE	U	JMP	FETCH
COMPLEMENT	NOP	I	CALL	INDRCT
	READ	U	JMP	NEXT
	COM	U	JMP	FETCH

Q.68 What do you mean by software of hardware interrupts? How there are used in a microprocessor system?

Ans.

Software and hardware interrupt:

The software interrupts are program instructions. These instructions are inserted at desired location in a program. A program generated interrupt also called trap, which stops current processing in order to request a service provided by the CPU. While running a program, if software interrupt instruction is encountered the CPU initiates an interrupt. For example a program might generate a software interrupt to read input from keyboard. Hardware interrupt is a type of interrupt generated either externally by the hardware devices such as input/ output ports, keyboard and disk drive etc or internally by the microprocessor. External hardware interrupts are used by device to request attention from CPU. Internal hardware interrupts are generated by the CPU to control events.

Q.69 What are the reasons of Pipe-Line conflicts is a Pipe Lined processor? How are they resolved?

Ans.

There are three major difficulties that cause the instruction pipeline to deviate form its normal operation.

1) Resource conflicts caused by access to memory by two segments at the same time. Most of these conflicts can be resolved by using separate instruction and data memories.

2) Data dependency conflicts arise when an instruction depends on the result of a previous instruction, but this result is not available.

3) Branch difficulties arise from branch and other instructions that change the value of PC. In computer, for solving conflicts problems to the compiler that translates the high level programming language into a machine language program. The compiler for such computer is designed to detect a data conflict and re order the instructions, to delay the loading of the conflicting data by inserting no-operation instructions. This method is referred to as delayed load.

Q. 70 Explain the difference between a subroutine & macro.

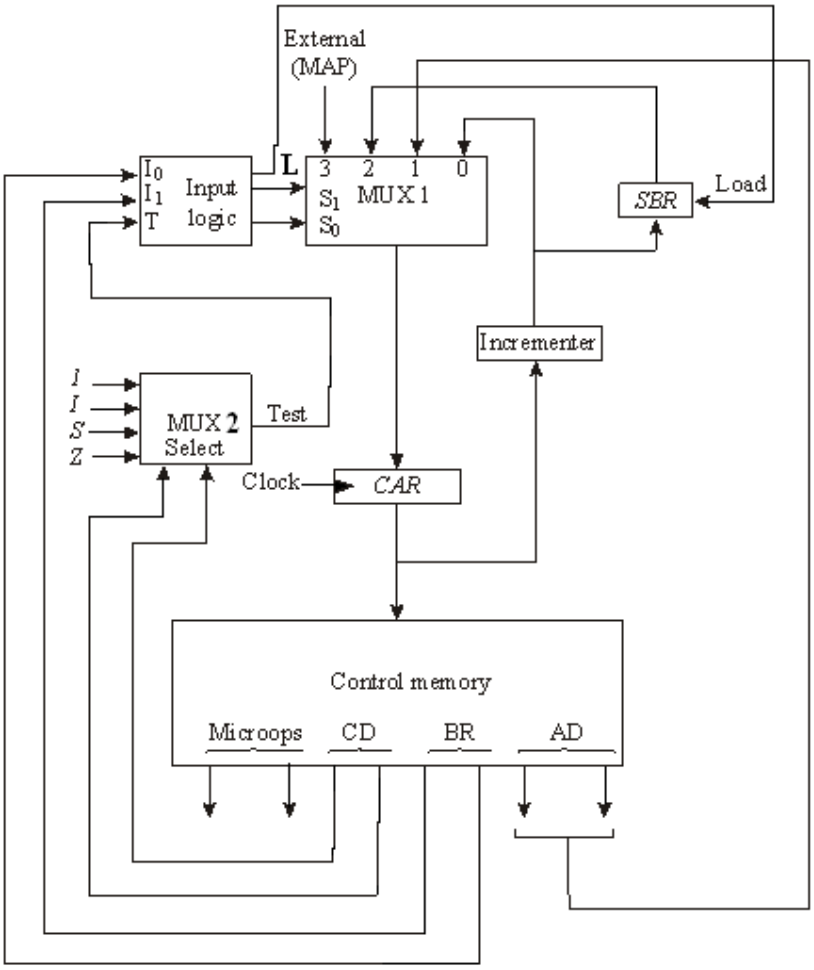
Ans.

It is inefficient to have to write code for standard routines. For example reading a character form the keyboard or saving a block of data to disk. Standard routines are available called library routines. They may be called up macros and many of the most useful routines are available as operating system calls. A call to a macro is a single command, which can be replaced by many commands that set put into the programme where the macro name is encountered. A set of common instructions that can be used in a program in called a subroutine.

Q. 71 With neat block diagram explain the working of a microprogram sequencer for control memory.

Ans.

Block Diagram of Micro program sequencer



The input logic circuit has three inputs I_0, I_1, and T, and three outputs, S_0, S_1, and L. Variables S_0, and S_1, select one of the source addresses for CAR. Variable L enables the load input in SBR. The binary values of the two selection variables

determine the path in the multiplexer. For example, with S_1 S_0 = 0, multiplexer input number 2 is selected and establishes a transfer path from *SBR* to *CAR*. Note that each of the four inputs as well as the output of MUX 1 contains a *7*-bit address. The truth table for the input logic circuit is shown. Inputs I_1 and I_0 are identical to the bit values in the BR field. The bit values for $S_1$and S_0 are determined from the stated function and the path in the multiplexer that establishes the required transfer. The subroutine register is loaded with the incremented value of *CAR* during a call microinstruction (BR = 01) provided that the status bit condition is satisfied (*T* = 1).

$$S_1 = I_1$$
$$S_0 = I_1 I_0 + I'_1 T$$
$$L = I'_1 I_0 T$$

Input Logic Truth Table for Microprogram Sequence

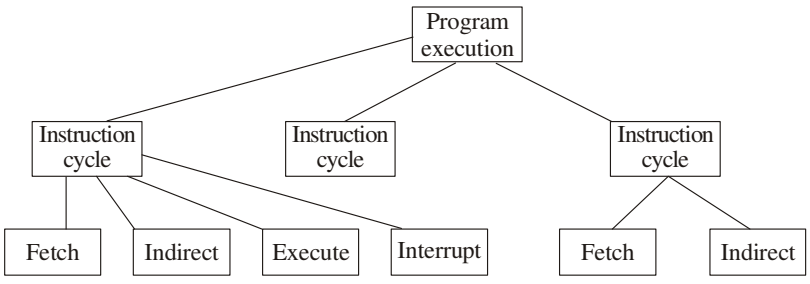
BR Field		Input I_1	I_0	T	Mux 1 S_1	S_0	Load *SBR* L
0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0
0	1	0	1	0	0	0	0
0	1	0	1	1	0	1	1
1	0	1	0	x	1	0	0
1	1	1	1	x	1	1	0

Q. 72 Discuss different method used for specifying micro operation in microoperation field of microcode. State their merits and demerits.

Ans.

The function of a computer is to execute program. We have the operation of a computer, in executing a program consists of a sequence of instruction cycles with one machine instruction per cycle. When we are referring to here is the execution time sequence of instructions. We have further seen that each instruction cycle can be considered to be made up a number of smaller writs.

Micro-operation - The prefix micro refers to the fact that each step is very simple and accomplish very little. The execution of program consists of the sequential execution of instructions. The performance



of each sub cycle involves one or more shorter operations, that is micro-operations. Micro-operation are the functional, or atomic, operation of a CPU.

The fetch cycle - The fetch cycle, which occurs at the beginning of each instruction cycle and causes an instruction to be fetched from memory.

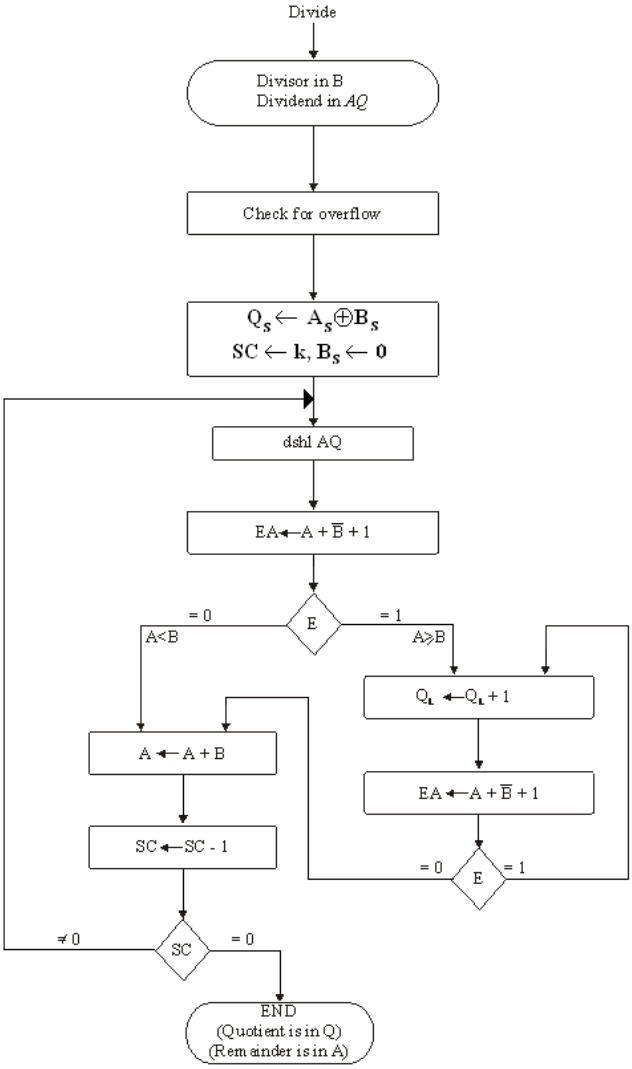
Instruction cycle:- The instruction cycle can be decomposed into a sequence of elementary micro-operations. There is one sequence each for the fetch, indirect and interrupt cycle and for execute cycle, there is one sequence of micro-operations for each opcode.

Micro - operations :-

- A computer executes a program
- Fetch/ execute cycle
- Each cycle has a number of steps are pipelining
- Called micro-operation
- Each step does very little

Q. 73 With neat flow chart, explain the procedure for division of floating point numbers carried out in a computer.

Ans:



Flowchart for decimal division

Decimal division is similar to binary division except of course that the quotient digits may have any of the 10 values from 0 to 9. In the restoring division method, the divisor is subtracted from the dividend or partial remainder as many times as necessary until a negative reminder results. The correct remainder is then restored by adding the divisor. The digit in the quotient reflects the number of subtractions up to but excluding the one that caused the negative difference. The decimal division algorithm is shown. It is similar to the algorithm with binary data except for the way the quotient bits are formed. The dividend (or partial remainder) is shifted to the left, with its most significant digit placed in A_e .The divisor is then subtracted by adding its 10's complement value. Since B_e is initially cleared, its complement value is 9 as required. The carry in E determines the relative magnitude of A and B, If E=0, it signifies that A<B. In this case the divisor is added to restore the partial remainder and Q_L stays at 0 (inserted there during the shift). If E=1, it signifies that A≥B. The quotient digit in Q_L is incremented once and the divisor subtracted again. This process is repeated until the subtraction results in a negative difference which is recognized by E being 0. When this occurs, the quotient digit is not incremented but the divisor is added to restore the positive remainder. In this way, the quotient digit is made equal to the number of times that the partial remainder "goes" into the divisor. The partial remainder and the quotient bits are shifted once to the left and the process is repeated k times to form k quotient digits. The remainder is then found in register A and the quotient is to register Q. The value of E is neglected.

Q.74 Give the flow table for register contents used in implementing booth's algorithm for the multiplier = - 6 and multiplicant = + 5.

Ans.

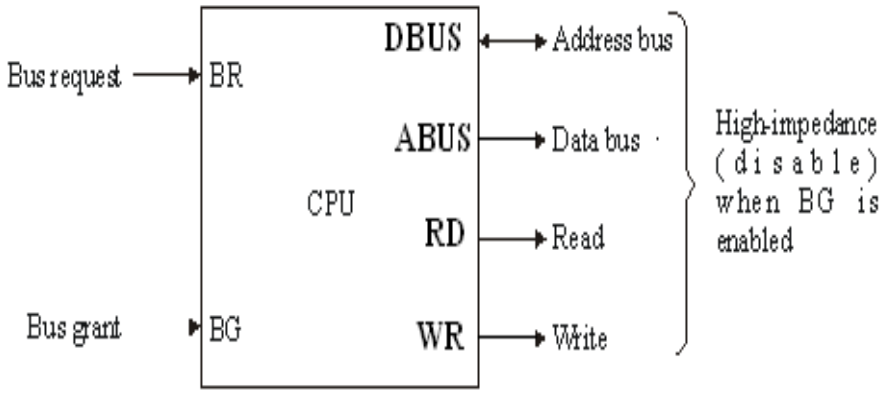
		BR = 0101				
Q_n	Q_{n+1}	$\overline{BR}+1$ = 1011	AC	QR	Q_{n+1}	SC
		Initial	0000	1010	0	100
0	0	Ashr	0000	0101	0	011
1	0	Subtract BR	$\frac{1011}{1011}$			
		ashr	1101	1010	1	010
0	1	Add BR	$\frac{0101}{0010}$			
		ashr	0001	0101	0	001
1	0	Subtract BR	$\frac{1011}{1100}$			
		ashr	1110	0010	1	000

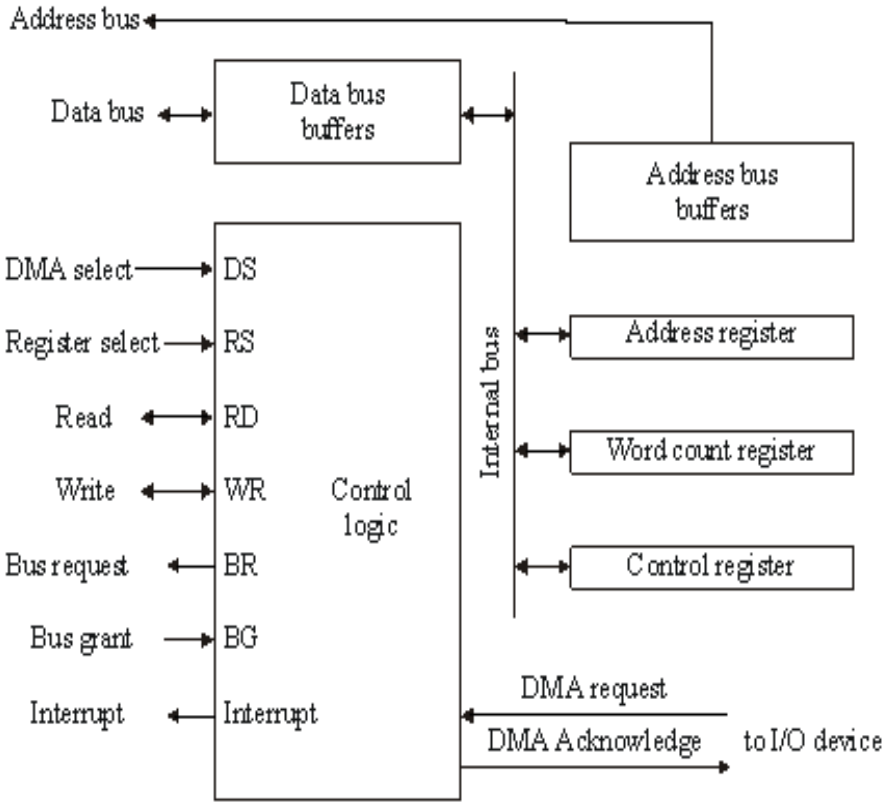
Final product is 11100010

Q.75 What do you mean by initialization of DMA controller? How DMA controller works? Explain with suitable block diagram.

Ans.
Figure below shows two control signals in the CPU that facilitate the DMA transfer. The bus request (BR) input is used by the DMA controller to request the CPU to

relinquish control of the buses. The CPU activates the bus grant (BG) output to inform the external DMA that the buses are in the high-impedance state. The DMA that originated the bus request can now take control of the buses to conduct memory transfers without processor intervention. When the DMA terminates the transfer, it disables the bus request line. The CPU disables the bus grant.





When the DMA takes control of the bus system, it communicates directly with the memory. Figure shows the block diagram of a typical DMA controller. The unit communicates with the CPU via the data bus and control lines. The registers in the 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 to the DMA registers. 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* control.

The DMA controller has three registers: an address register, a word count register, and a control register. For each word that is transferred, the DMA increments its address registers and decrements its word count register. If the word count does not reach zero, the DMA checks the request line coming from the peripheral. For a high-speed device, the line will be active as soon as the previous transfer is completed. A second transfer is then initiated, and the process continues until the entire block is transferred. If the peripheral speed is slower, the DMA request line may come somewhat later. In this case the DMA disables the bus request line so that the CPU can continue to execute its program. When the peripheral requests a transfer, the DMA requests the buses again.

Q.76 The access time of a cache memory is 120 ns and that of main memory 900 ns. It is estimated that 80% of the memory requests are for read and remaining 20% for write. The hit ratio for read access only is 0.9. A write-through procedure is used.

(i) What is the average access time of the system considering only memory real cycles?
(ii) What is the hit ratio taking in to consideration the write cycle?
(iii) What is the average access time of the system for both read and write requests.

Ans.

i) 0.9 x 120 + 0.1 x 11000 = 108 + 110 = 218 nsec.
cache access memory access

ii) 0.2 x 900 + 0.8 x 200 = 180 + 100 = 280 nsec.
write access

iii) Hit ratio = 0.8 x 0.9 = 0.72

Q.77 Write short notes an any two of the followings.

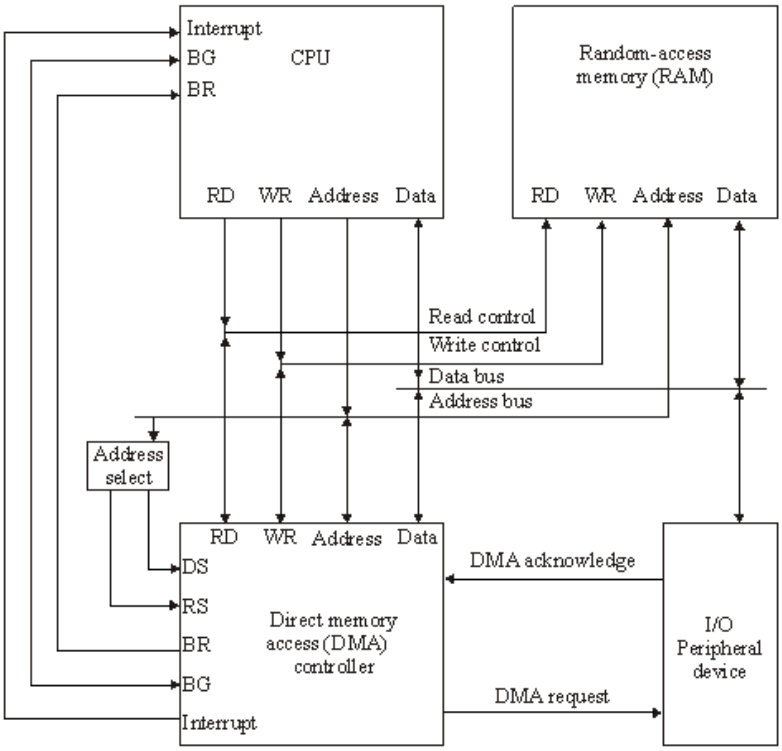
(i) DMA data transfer.
(ii) Handshaking method of data transfer
(iii) Isolated Vs memory mapped I/O.
(iv) RISC architecture.

Ans.

i) **DMA data transfer**:-

The position of the DMA controller among the other components in a computer system is shown in figure. The CPU communicates with the DMA through the address and data buses as with any interface unit. The DMA has its own address, which activates the *DS* and *RS* lines. The CPU initializes the DMA through the data bus. Once the DMA receives the start control command, it can start the transfer between the peripheral device and the memory. When the peripheral device receives a DMA acknowledge, it puts a word in the data bus (for write) or receives a word

from the data bus (for read). Thus the DMA controls the read or write operations and supplies the address for the memory. The peripheral unit can

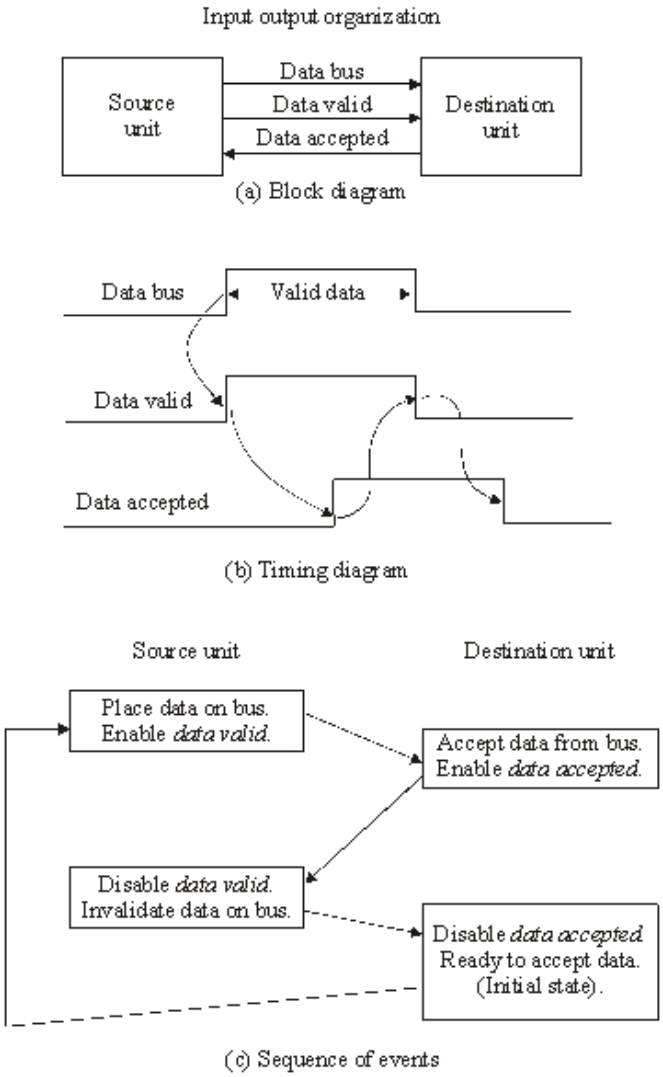


then communicate with memory through the data bus for direct transfer between the two units while the CPU is momentarily disabled. For each word that is transferred, the DMA increments its address registers and decrements its word count register. If the word count does not reach zero, the DMA checks the request line coming from the peripheral. For a high-speed device, the line will be active as soon as the previous transfer is completed. A second transfer is then initiated, and the process continues until the entire block is transferred. If the peripheral speed is slower, the DMA request line may come somewhat later. In this case the DMA disables the bus request line so that the CPU can continue to execute its program. When the peripheral requests a transfer, the DMA can continue to execute its program. When the peripheral requests a transfer, the DMA requests the buses again.

DMA transfer is very useful in many applications. It is used for fast transfer of information between magnetic disks and memory. It is also useful for updating the display in an interactive terminal. Typically, an image of the screen display of the terminal is kept in memory which can be updated under program control. The contents of the memory can be transferred to the screen periodically by means of DMA transfer.

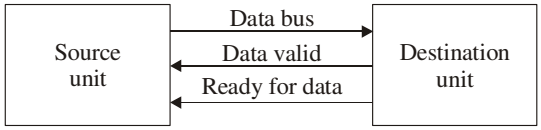
ii) **Handshaking method of data transfer**:-

The disadvantage of the strobe method is that the source unit that initiates the transfer has no way of knowing whether the destination unit has actually received the data item that was placed in the bus. Similarly, a destination unit that initiates the transfer has no way of knowing whether the source unit has actually placed the data on the bus.

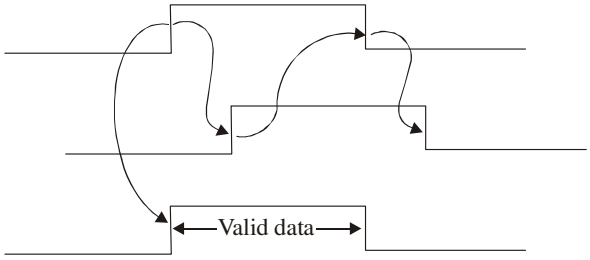


The two handshaking lines are *data valid*, which is generated by the source unit, and *data accepted,* generated by the destination unit. The timing diagram shows the exchange of signals between the two units. The sequence of events listed in part (C) shows the four possible states that the system can be at any given time. The source unit initiates the transfer by placing the data on the bus and enabling its *data valid* signal. The *data accepted* signal is activated by the destination unit after it accepts the data from the bus. The source unit then disables its *data valid* signal, which invalidates the data on the bus. The destination unit then disables its *data accepted* signal and the system goes into its initial state. The source does not send the next data item until after the destination unit shows its readiness to accept new data by disabling its *data accepted* signal. This scheme allows arbitrary delays from one state to the next and permits each unit to respond at its own data transfer rate. The rate of transfer is determined by the slowest unit.
The destination-initiated transfer using handshaking lines is shown. Notes that the name of the signal generated by the destination unit

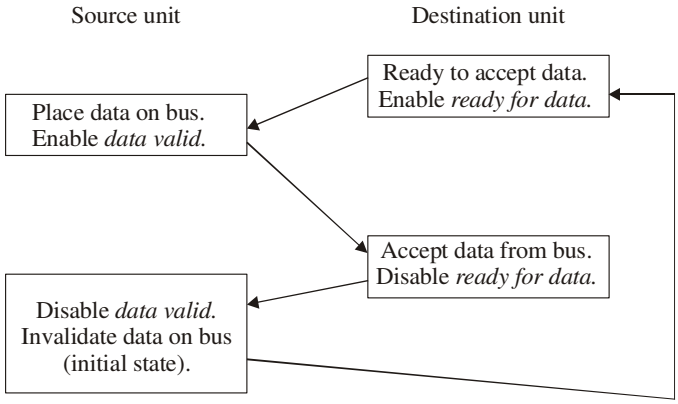
Destination-initiated transfer using handshaking.



(a) Block diagram



(b) Timing diagram



(c) Sequence of events

has been changed to *ready for data* to reflect its new meaning. The source unit in this case does not place data on the bus until after it receives the *ready for data* signal from the destination unit. From there on, the handshaking procedure follows the same pattern as in the source-initiated case. Note that the sequence of events in both cases would be identical if we consider the *ready for data* signal as the complement of *data accepted*. In fact, the only difference between the source-initiated and the destination-initiated transfer is in their choice of initial state.

iii) **Isolated vs memory mapped I/O:-**

In the isolated I/O configuration, the CPU has distinct input and output instructions, and each of these instructions is associated with the address of an interface register. When the CPU fetches and decodes the operation code of an input or output instruction, it places the address associated with the instruction into the common address lines. The isolated I/O method isolates memory and I/O addresses so that memory address values are not affected by interface address assignment since each has its own address space. The other alternative is to use the same address space for

both memory and I/O. This configuration is referred to as *memory-mapped I/O*. In a memory-mapped I/O organization there is no specific input or output instructions. The CPU can manipulate I/O data residing in interface registers with the same instructions that are used to manipulate memory words. Computers with memory-mapped I/O can use memory-type instructions to access I/O data. The advantage is that the load and store instructions used for reading and writing from memory can be used to input and output data from I/O registers. In a typical computer, there are more memory-reference instructions than I/O instructions. With memory-mapped I/O all instructions that refer to memory are also available for I/O.

iv) **RISC architecture**:-

The concept of RISC architecture involves an attempt to reduce execution time by simplifying the instruction set of the computer. The major characteristics of a RISC processor are:

1. Relatively few instructions.
2. Relatively few addressing modes.
3. Memory access limited to load and store instructions.
4. All operations done within the registers of the CPU.
5. Fixed-length, easily decoded instruction format.
6. Single-cycle instruction execution.
7. Hardwired rather than microprogrammed control.
8. Faster execution.

Other characteristics attributed to RISC architecture are:

1. A relatively large number of registers in the processor unit.
2. Use of overlapped register windows to speed-up procedure call and return.
3. Efficient instruction pipeline.
4. Compiler support for efficient translation of high-level language programs into machine language programs.

A large number of registers is useful for storing intermediate results and for optimizing operand references. The advantage of register storage as opposed to memory storage is that registers can transfer information to other registers much faster than the transfer of information to and from memory. Thus register-to-memory operations can be minimized by keeping the most frequent accessed operands in registers. Studies that show improved performance for RISC architecture do not differentiate between the effects of the reduced instruction set and the effects of a large register file.

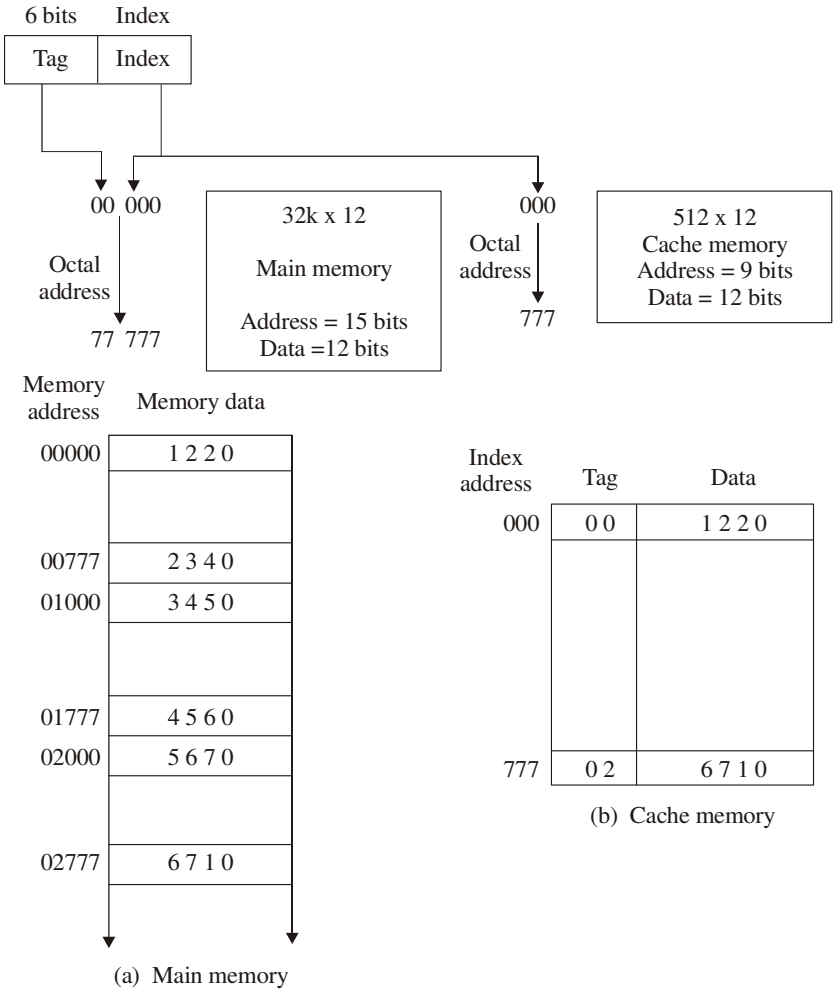
Q. 78 Explain direct mapping of cache memory system.

Ans.

Associative memories are expensive compared to random-access memories because of the added logic associated with each cell. The possibility of using a random-access memory for the cache is investigated. The CPU address of 15 bits is divided into two fields. The nine least significant bits constitute the *index* field and the remaining six bits form the *tag* field. The figure shows that main memory needs an address that includes both the tag and the index bits. The number of bits in the index field is equal to the number of address bits required to access the cache memory. In the general case, there are 2^k words in cache memory and 2^n words in

main memory. The n bit memory address is divided into two fields: k bits for the index field and the *n*-k bits for the tag field. The direct mapping cache organization uses the *n*-bit address to access the main memory and the k-bit index to access the cache. The internal organization of the words in the cache memory is as shown. Each word in cache consists of the data word and its associated tag. When a new word is first brought into the cache the tag bits are stored alongside the data bits. When the CPU generates a memory request, the index field is used for the address to access that cache. The tag field of the CPU address is compared with the tag in the word read from the cache. If the two tags match, there is a hit and the desired data word is in cache. If there is no match, there is a miss and the required word is read from main memory. It is then stored in the cache together with the new tag, replacing the previous value. The disadvantage of direct mapping is that the hit ratio can drop considerably if two or more words whose addresses have the same index but different tags are accessed repeatedly. However, this possibility is minimized by the fact that such words are

Addressing relationships between main and cache memories



Direct mapping cache organization

Relatively far apart in the address range. To see how the direct-mapping organization operates, consider the numerical example shown. The word at address

zero is presently stored in the cache (index = 000, tag = 00, data = 1220). Suppose that the CPU now wants to access the word at address 02000. The index address is 000, so it is used to access the cache. The two tags are then compared. The cache tag is 00 but the address tag is 02, which does not produce a match. Therefore, the main memory is accessed and the ata word 5670 is transferred to the CPU. The cache word at index address 000 is then replaced with a tag of 02 and data of 5670.

Q.79 What do you mean by locality of reference?

Ans.
The references to memory at any given internal of time tend to be confined within a few localized areas in memory. This phenomenon is known as the property of locality of reference. The locality of reference property, that over a short internal of time, the addresses generated by a typical program refer to a few localized area of memory repeatedly, while the remainder of memory is accessed relatively infrequently.

Q.80 A virtual memory system has an address space of 8k words, memory space of 4k words and Page & Block size of 1k words. The following page reference changes occur during a given time interval.

4, 2, 0, 1, 2, 6, 1, 4, 0, 1, 0, 2, 3, 5, 7

Determine the four pages that are resident in main memory after each Page reference change if the replacement algorithm used is (i)FIFO (ii) LRU.

Ans.
An address space of 8K and a memory of 4K words and page Block size of 1K words. Four pages of address space may reside in main memory in any one of the four blocks.

Page 0	
Page 1	
Page 2	Block 0
Page 3	Block 1
Page 4	Block 2
Page 5	Block 3
Page 6	
Page 7	

Address Space
$N = 8K = 2^{13}$

Memory Space
$M = 4K = 2^{12}$

(1) **FIFO**
String 4, 2, 0, 1, 2, 6, 1, 4, 0, 1, 0, 2, 3, 5, 7

4	2	0	1	2	6	1	4	0	1	0	2	3	5	7
4	4	4	4		6		6				6	6	5	5
	2	2	2		2		4				4	4	4	7
		0	0		0		0				2	2	2	2
			1		1		1				1	3	3	3

Page fault by FIFO = 10

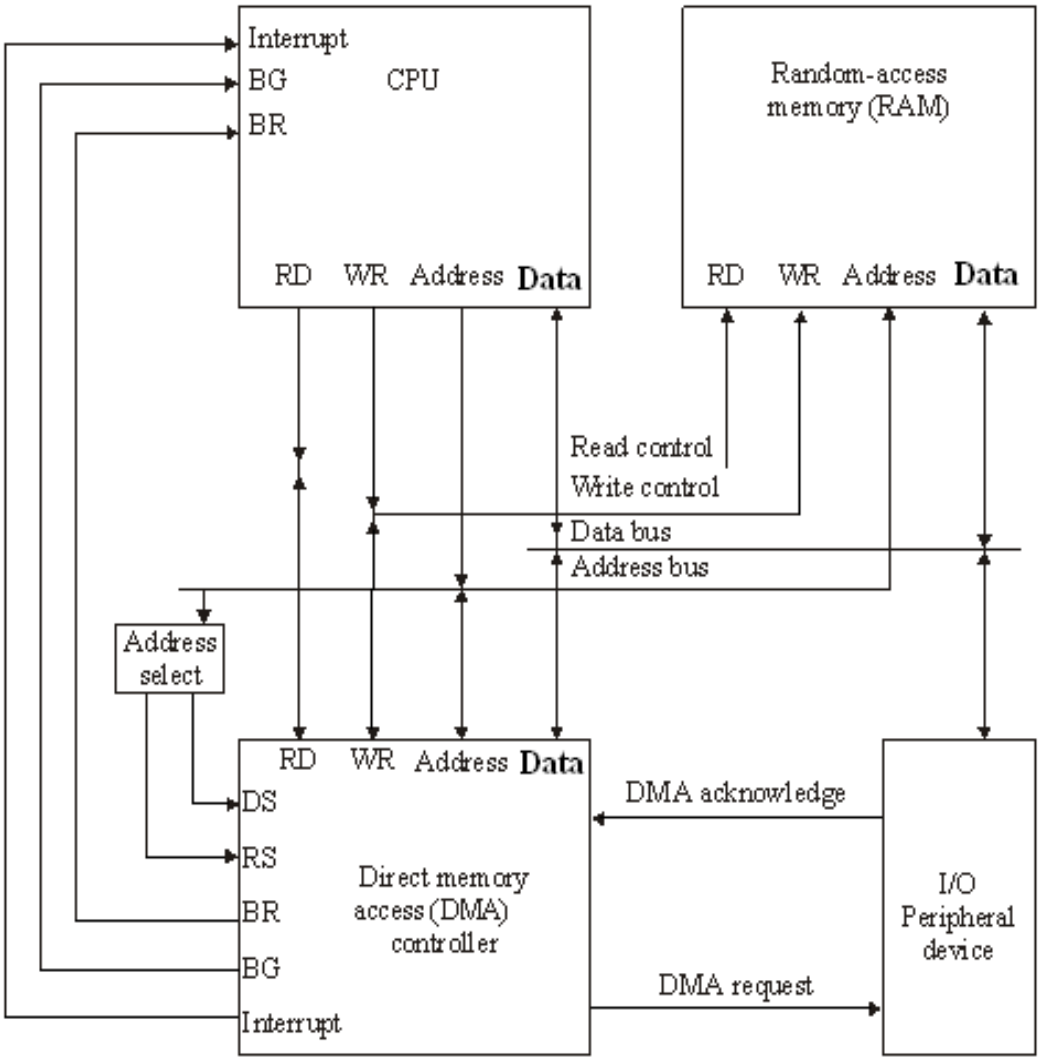
(2) **LRU**

4	2	0	1	2	6	1	4	0	1	0	2	3	5	7
4	4	4	4		6		6	6			2	2	2	2
	2	2	2		2		2	0			0	0	0	7
		0	0		0		4	4			4	3	3	3
			1		1		1	1			1	1	5	5

Page fault by LRU = 11

Q.81 With neat block diagram, explain how DMA controller is initialized for DMA data transfer.

Ans.



The position of the DMA controller among the other components in a computer system is illustrated in Fig. The CPU communicates with the DMA through the address and data buses as with any interface unit. The DMA has its own address, which activates the DS and RS lines. The CPU initializes the DMA through the data bus. Once the DMA receives the start control command, it can start the transfer between the peripheral device and the memory. When the peripheral device sends a DMA request, the DMA controller activates the BR line, informing the CPU to relinquish the buses. The CPU responds with its BG line, informing the DMA that its buses are disabled. The DMA then puts the current value of its address register into the address bus, initiates the RD or WR signal, and sends a DMA acknowledge to the peripheral device. Note that the RD and WR lines in the DMA controller are bi-directional. The direction of transfer depends on the status of the BG line. When BG = 0, the RD and WR are input lines allowing the CPU to communicate with the internal DMA controller to the random-access memory to specify the read or write operation for the data. When the peripheral device receives a DMA acknowledge, it puts a word in the data bus (for write) or receives a word from the data bus (for read). Thus the DMA controls the read or write operations and supplies the address for the memory. The peripheral unit can then communicate with memory through the data bus for direct transfer between the two units while the CPU is momentarily disabled.

Q.82 How data is transmitted in synchronous serial communication system?

Ans.

Synchronous transmission does not use start-stop bits to frame characters and therefore makes more efficient. In synchronous transmission, where an entire block of character is transmitted, each character has a parity bit for the receiver to check. After the entire block is sent the transmitter sends one more character that constitutes a parity over the length of the message. This character is called a longitudinal redundancy check (LRC) and is the accumulation of the exclusive OR of all transmitted character. The receiving station calculates the LRC as it receives characters and compares it with the transmitted LRC. The calculated and received LRC should be equal for error-free messages. If the receiver finds an error in the transmitted block, it inform the sender to retransmit the same block once again.

Q.83 How many characters per second can be transmitted over a 1200 baud line in asynchronous serial transmission in following modes – assume a character code is of eight bits?

(i) Synchronous Serial transfer
(ii) Asynchronous Serial Transfer with 2 stop bits
(iii) Asynchronous Serial Transfer with one stop bit.

Ans.

Baud Rate = 1200
Character Code = 8 bits
(i) Transmitted Characters per second in Synchronous Serial transmission
Transfer = 1200/8=150 Character per second
(ii) Asynchronous Serial Transfer with 2 stop bits
total number of bits = 1 start bit + 8 information bits + 2 stop bits

= 1+8+2 = 11 bits
baud rate = 1200
Transmitted Character per second = 1200/11
= 109 Character per second
(iii) Asynchronous Serial Transfer with one stop bit.
Total no. of bits = 1 start bit + 8 information bits + 1 stop bit
= 10 bits
band rate = 1200
►Transmitted Character per second = 1200/10 = 120 Character per second

Q.84 A Computer uses a memory unit with 256K words of 32 bits each. A binary instruction code is stored in one word of memory. The instruction has four parts: an indirect bit, an operation code, a register code part to specify one of 64 registers and an address part.

(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) For a memory unit with 256K words of 32 bits each we need 18 bits to specify an address.
Operation Code = 7 bits
32-25 = 7 bits for opcode
Register Code = 6 bits
> $2^6 = 64$

(ii) Instruction Word Format

	1	7	6	18 = 32 bits
	I	Opcode	Register Code	Address

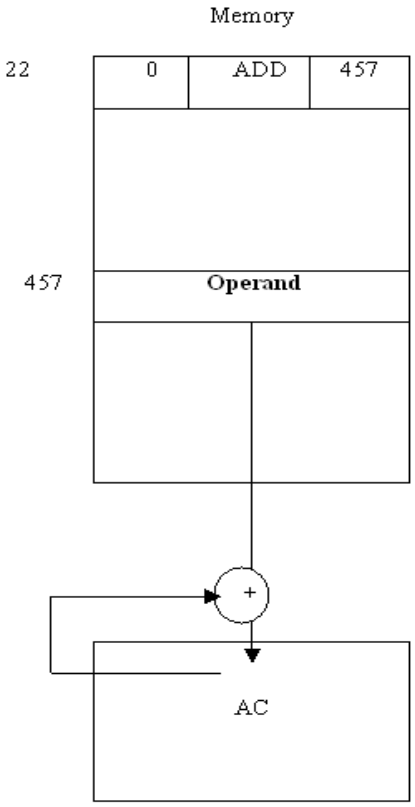
(iii) data and address inputs of the memory
data = 32 bits
address = 18 bits

Q.85 What is difference between a direct and an indirect address instruction? How many references to memory are needed for each type of instruction to bring an operand into a processor register?

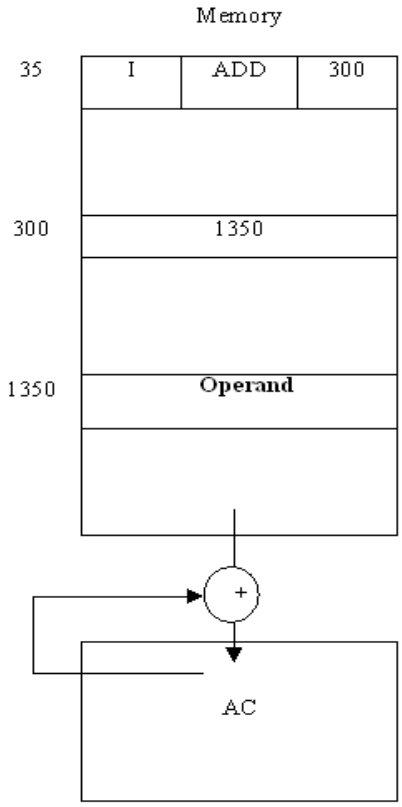
Ans.

15	14	12	11
1	OP Code	Address	

Instruction format fig. a



Direct Address
Fig.b



Indirect address
Fig.c

Direct address:-

Instruction code format shown in fig. a. It consists of a 3-bit operation code. A 12 bit address, and an indirect address mode bit designated by I. The mode bit is 0 for a direct address and I for an indirect address. A direct address instruction is shown in fig. b. It is placed in address 22 in memory. The I bit is 0, so the instructors is recognized as a direct address instruction. The operation code specifies an ADD instruction, and the address part is the binary equivalent of 457. The control finds the operand in memory at address 457 and adds is to the content of AC.

Indirect address:-

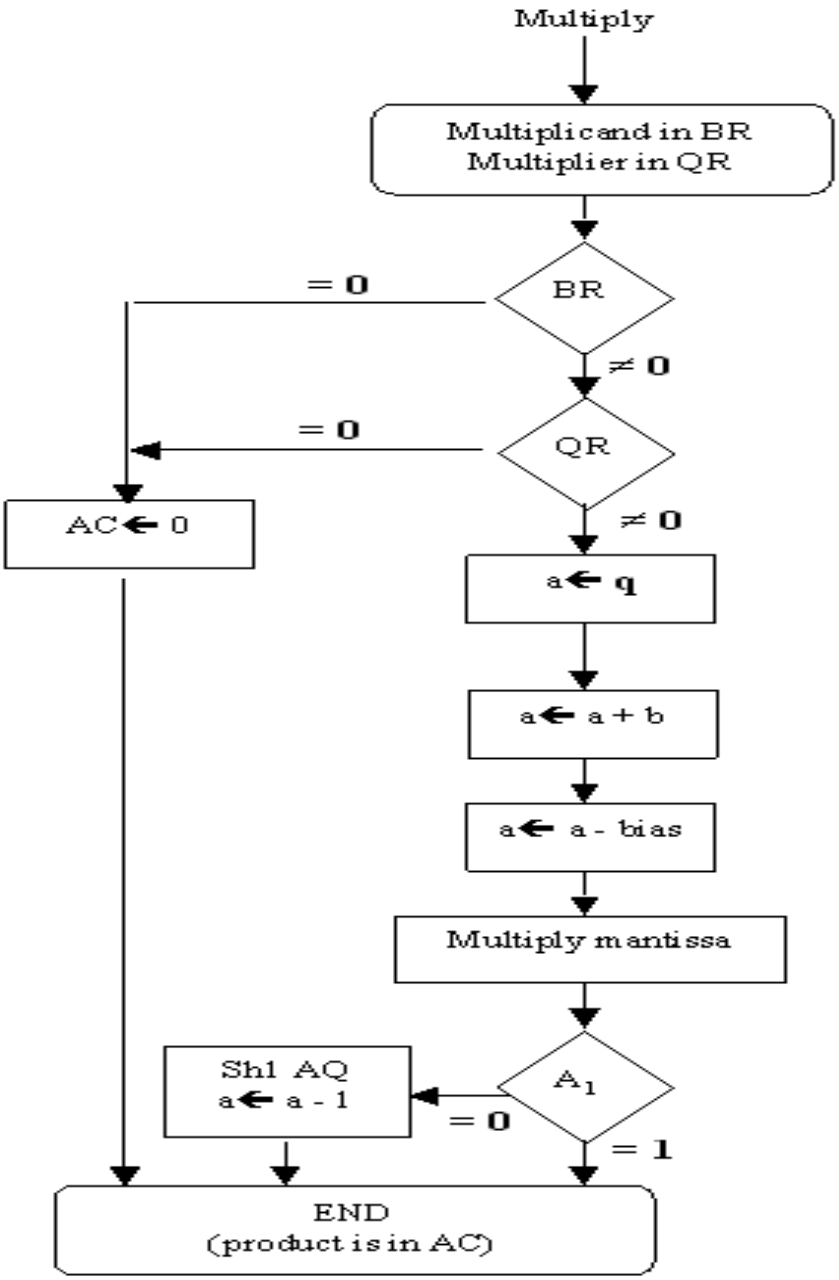
The instruction in address 35 shown in fig. c has a mode bit I=1. Therefore, it is recognized as an indirect address instruction. The address part is the binary equivalent of 300. The control goes to address 300 to find the address of the operand. The address of the operand in this case is 1350. The operand found in address 1350 is then added to the content of AC. The indirect address instruction needs two references to memory to fetch and operand. The first reference is needed to read the address of the operand the second is for the operand itself. The effective address to be the address of the operand in a computation type instruction or the target address in a branch-type instruction.

(1) ADD to AC
(2) ADD to AC
(3) LDA: Load to AC
(4) STA: Store AC

(5) BUN: Branch Unconditionally
(6) BSA: Branch and Save Return Address
(7) ISZ: Increment and Skip to Zero

Q.86 With neat flow chart discuss the procedure for floating point multiplication. Explain with the help of an example.

Ans.



The multiplication of two floating-point numbers requires that we multiply the mantissas and add the exponents.
The multiplication algorithm can be subdivided into four parts:

1. Check for zeros.
2. Add the exponents.
3. Multiply the mantissas.
4. Normalize the product.

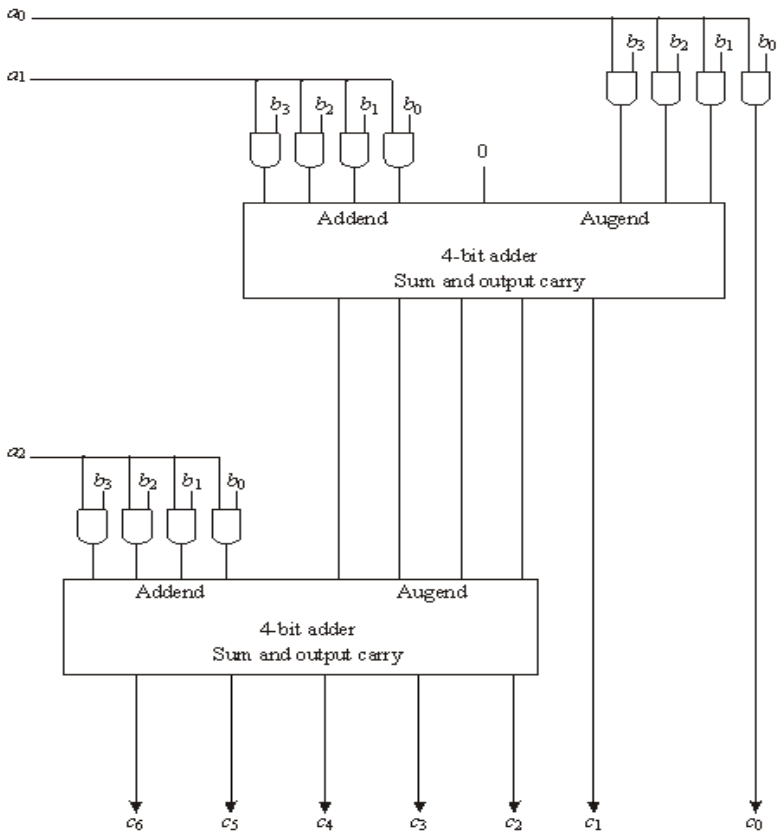
Steps 2 and 3 can be done simultaneously if separate adders are available for the mantissas the exponents.

The flowchart for floating-point multiplication is shown in Fig. The two operands are checked to determine if they contain a zero. If either operand is equal to zero, the product in the AC is set to zero and the operation is terminated. If neither of the operands is equal to zero, the process continues with the exponent addition.

The exponent of the multiplier is in q and the adder is between exponents a and b. It is necessary to transfer the exponents from q to a, add the two exponents, and transfer the sum into a. Since both exponents are biased by the addition of a constant, the exponent sum will have double this bias. The correct biased exponent for the product is obtained by subtracting the bias number from the sum. The mantissas are then multiplied as in the fixed – point case with the product residing in A and Q. Overflow cannot occur during multiplication, the product may have an underflow, so the most significant bit in A is checked. If it is a 1, the product is already normalized. If it is a 0, the mantissa in AQ is shifted left and the exponent decremented.

Q. 87 Design a Three-bit array multiplier. Use AND gates and binary address.

Ans.



Q.88 Write a program to evaluate the arithmetic statement.

$$P=\frac{(X-Y+Z)*(M*n-o)}{Q+R*S}$$

By using

(i) Two address instructions
(ii) One address instructions
(iii) Zero address instructions

Ans.

(i) Two Address Instruction

MOV	R1 X	R1 ← M [X]
ADD	R1, Z	R1 ← R1 + M[Z]
SUB	R1, Y	R1 ←R1 - M[Y]
MOV	R2, m	R2 ← M[m]
MUL	R2, n	R2 ← R2 * M[n]
SUB	R2, o	R2 ←R2 – M[o]
MUL	R1, R2	R1 ← R1 x R2
MOV	R3, R	R3 ← R3 * M[R]
MUL	R3, S	R3 ← R3 * M[S]
ADD	R3, Q	R3 ← R3 + M[Q]
DIV	R1, R3	R1← R1 ¸ R3
MOV	P, R1	M[P] ← R1

(ii) One address Instruction

$$P=\frac{(X-Y+Z)*(M*n-o)}{Q+R*S}$$

LOAD	X	AC←M [X]
ADDZ		AC←AC + M[Z]
SUB	Y	AC← AC - M[Y]
STORE	T	M [T] ←AC
LOAD	M	AC←M[m]
MUL	N	AC←AC x M[n]
SUB	O	AC←AC – M[o)
MUL	T	AC←AC x M[T]
STORE	U	M[U] ← AC
LOAD	R	AC ← M [R]
MUL	S	AC← AC x M[S)
ADD	Q	AC ← AC + M [Q]
DIV	V	AC ← M[v]/ AC
STORE	P	M[P] ← AC

(iii) Zero address Instruction

$$P=\frac{(X-Y+Z)*(M*n-o)}{Q+R*S}$$

PUSH	X	TOS←X
PUSH	Y	TOS←Y

SUB		TOS ← X-Y
PUSH	Z	TOS ← Z
ADD		TOS ← (X-Y+Z)
PUSH	M	TOS ← M
PUSH	N	TOS ← N
MUL		TOS ← M*N
PUSH	O	TOS ← O
SUB		TOS ← (M*N-O)
MUL		TOS ← (X-Y+Z) x (M*N-O)
PUSH	R	TOS ← R
PUSH	S	TOS ← S
MUL		TOS ← R*S
PUSH	Q	TOS ← Q
ADD		TOS ← Q-R*S
DIV	O	
POP	P	M [P] ← TOS

Q.89 Convert the following arithmetic expression from infix notation to RPN.
A*B + B* (B*D+C*E)

Ans.
B*D+C*E
In RPN
BD*CE*+
Infix A*B+B*(BxD+CxE)
RPN AB*BBD*CE*+*+

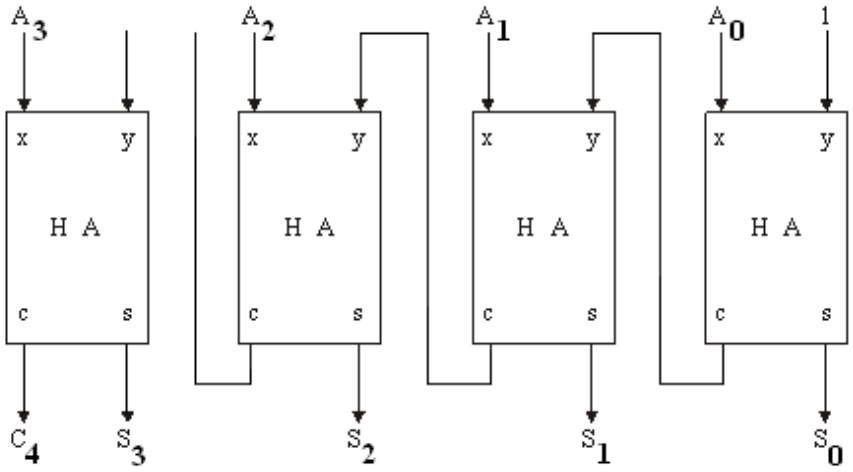
Q.90 What is subroutine? How it is executed by the processor? What is the importance of subroutine parameters and data linkage? How is it established?

Ans.
Subroutines:-
Frequently, the same piece of code must written over again in many different parts of a program. Instead of repeating the code every time it is needed, there is an obvious advantage if the common instructions are written only once. A set of common instructions that can be used in a program many times is called a *subroutine.* Each time that a subroutine is used in the main part of the program, a branch is executed to the beginning of the subroutine. After the subroutine has been executed, a branch it is made back to the main program.
Subroutine Parameters and Data Linkage:-
When a subroutine is called, the main program must transfer the data. The subroutine shifted the number and left it there to be accepted by the main program. It is necessary for the subroutine to have access to data from the calling program and to return results to that program. The accumulator can be used for a single input parameter and a single output parameter.
Consider a subroutine that performs the logic OR operation. *Two* operands must be transferred to the subroutine and the subroutine must return the result of the operation. The accumulator can be used to transfer one operand and to receive the result. The other operand is inserted in the location following the BSA instruction.

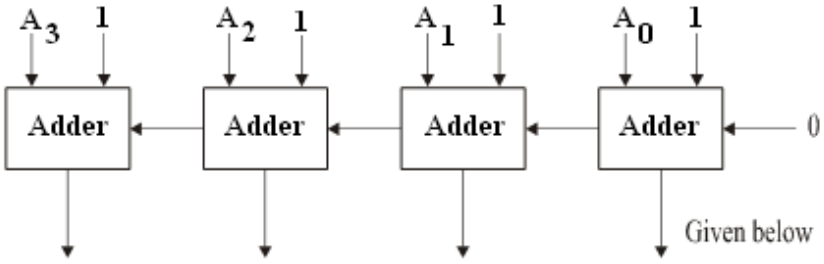
The subroutine must increment the return address stored in its first location for each operand that it extracts from the calling program. Moreover, the calling program can reserve one or more locations for the subroutine to return results that are computed. The first location in the subroutine must be incremented for these locations as well, before the return. If there is a large amount of data to be transferred , the data can be placed in a block of storage and the address of the first item in the block is then used as the linking parameter. A subroutine that moves a block of data starting at address into a block starting with address. The length of the block is 16 words. The first introduction is a branch to subroutine MVE. The items are retrieved from their blocks by the use of two pointers. The counter ensures that only 16 items are moved. When the subroutine completes its operation, the data required is in the block starting address. The return to the main program is to the HLT instruction.

Q.91 Design a 4-bit combinational incrementer and decrementer circuit.

Ans.



Incrementer Circuit



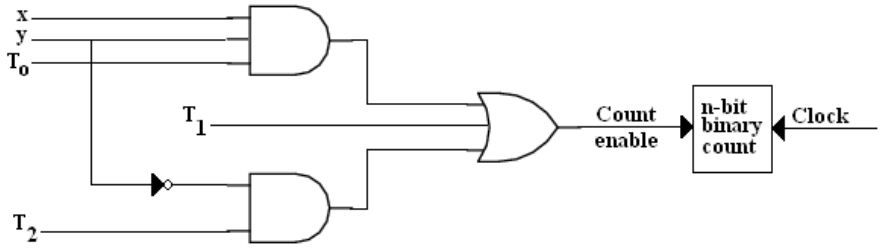
Decrementer Circuit

Q.92 Show the hardware implementation of following statement:

$xyT_0+T_1+yT_2: AR \leftarrow AR+1$

Where x, y are control functions and T_0, T_1, T_2 are T - State

Ans.

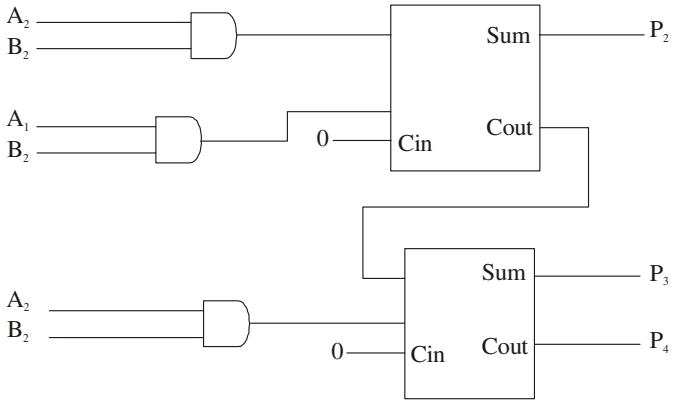


Q.93 Represent the given conditional control statement by two register transfer Statements with Control functions.
If (P = 1), Than $R_1 \leftarrow R_2$ else if (Q = 1) Than $R_1 \leftarrow R_3$. **(2)**

Ans: The two resistor transfer statements are :
P: $R_1 \leftarrow R_2$
P' Q: $R_1 \leftarrow R_3$

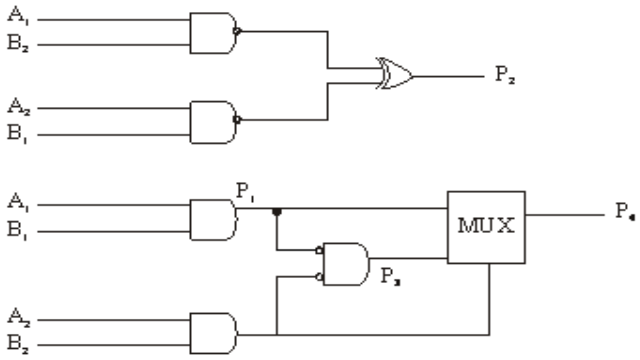
Q.94 Implement a 2 - bit multiplier circuit by using multipliers.

Ans.



2 bit multiplier circuit

By using multiplier



Q. 95 If $P = \sum(m_0, m_3, m_4, m_8, m_{12}, m_{13}, m_{15})$ and $Q = \Pi(M_1, M_4, M_5, M_{12}, M_{14})$, Then find the expression for $X = P \oplus Q$ in SOP & POS. **(8)**

Ans.

$P = \sum(m_0, m_3, m_4, m_8, m_{12}, m_{13}, m_{15})$

$Q = \pi (M_1, M_4, M_5, M_{12}, M_{14})$

$M_o = 0000$, $M_3 = 0011$, $M_4 = 0100$

$M_8 = 1000$, $M_{12} = 1100$, $M_{13} = 1101$, $M_{15} = 1111$

$M_1 = 0001$, $M_4 = 0100$, $M_5 = 0101$,

$M_{12} = 1100$, M14 = 1110

K - Map for SOP

CD \ AB	A'B'	A'B	AB	AB'
C'D'	1	0	1	X
C'D	1	0	X	X
CD	1	1	1	0
CD'	1	X	X	X

F = A'B' + CDB + C'D'A

F = C'D + C'A'B + DB'

= (C + 0')(C + A + B')(0' + B)

	A'B'	A'B	AB	AB'
C'D'	1	0	1	X
C'D	0	0	X	0
CD	0	1	1	0
CD'	1	X	X	X

F = (D' + B) (C + A + B')

Q. 96 What is excitation table? List the excitation table for SR-FF, JK-FF D-FF and T-FF.

Ans.

Flip-flop specifies the next state when the input and the present state are known. During the design of sequential circuits, the required transition form present state to next state and to find the FF input conditions that will cause the required transition. For this reason we need a table that lists the required input combinations for a given change of state. Such a table is called a flip-flop excitation table.

SR Flip-flop			
Q(t)	Q(t+1)	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

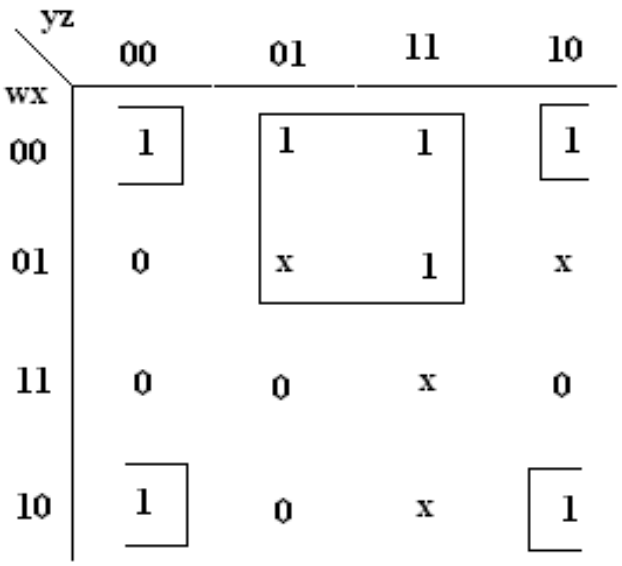
D Flip-flop		
Q(t)	Q(t+1)	DR
0	0	0
0	1	1
1	0	0
1	1	1

JK flip-flop			
Q(t)	Q(t+1)	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

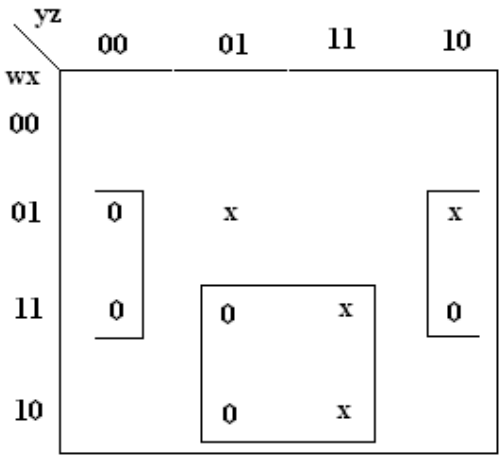
T flip-flop		
Q(t)	Q(t+1)	DR
0	0	0
0	1	1
1	0	1
1	1	0

Q. 97 Simplify the Boolean function F together with don't care condition D in
(i) Sum of Product (ii) Product of sums
F(w, x, y, z) = Σ (0, 1, 2,3, 7, 8, 10)
D(w, x, y, z) = Σ (5, 6, 11, 15)

Ans. (i)



.(ii)

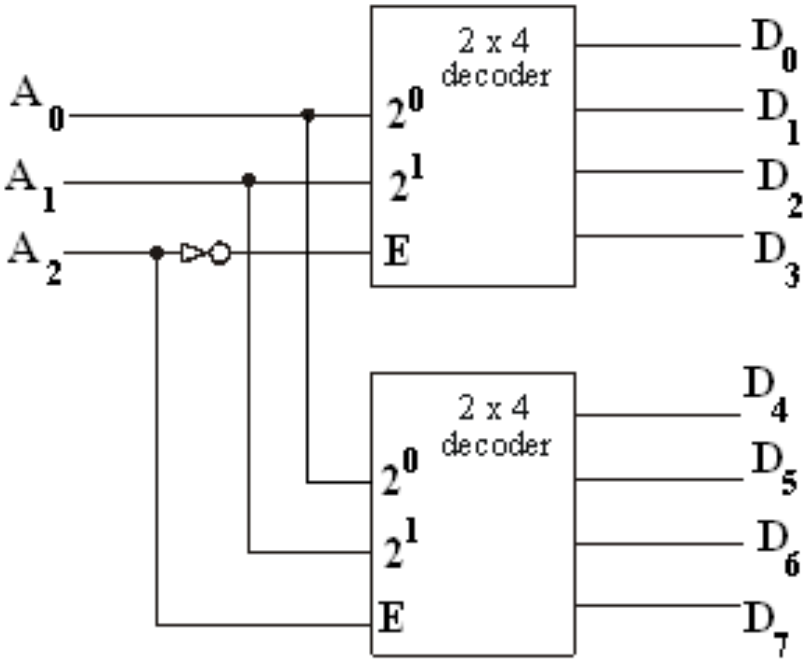


$F = (w' + z')\ (x' + z)$

Q. 98 Design a 3x8 decoder with the help of two 2x4 decoders.

Ans.

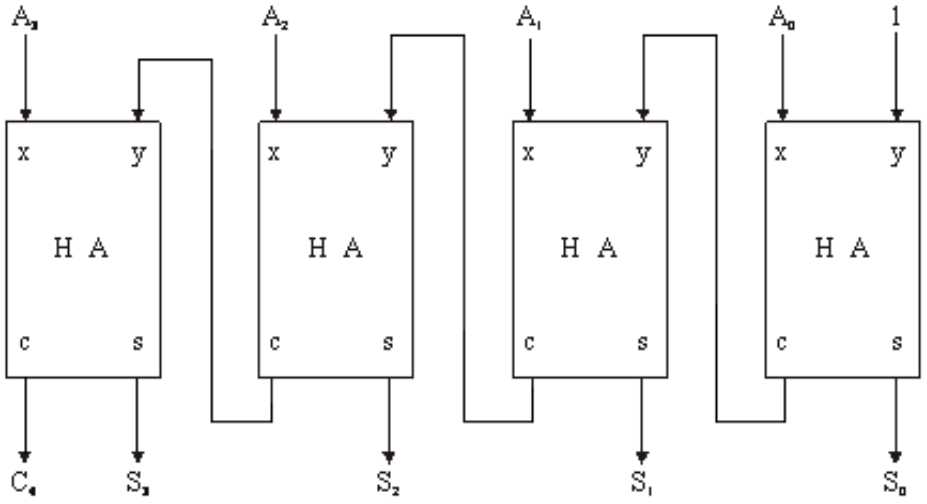
Decoder circuit



Q. 99 Design a binary Incrementer and binary Decrementer.

Ans.

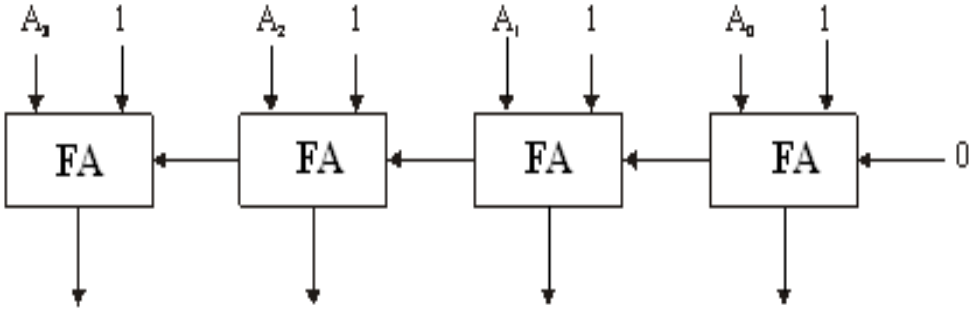
Incrementer circuit



4 bit binary incrementer

Decrementer circuit

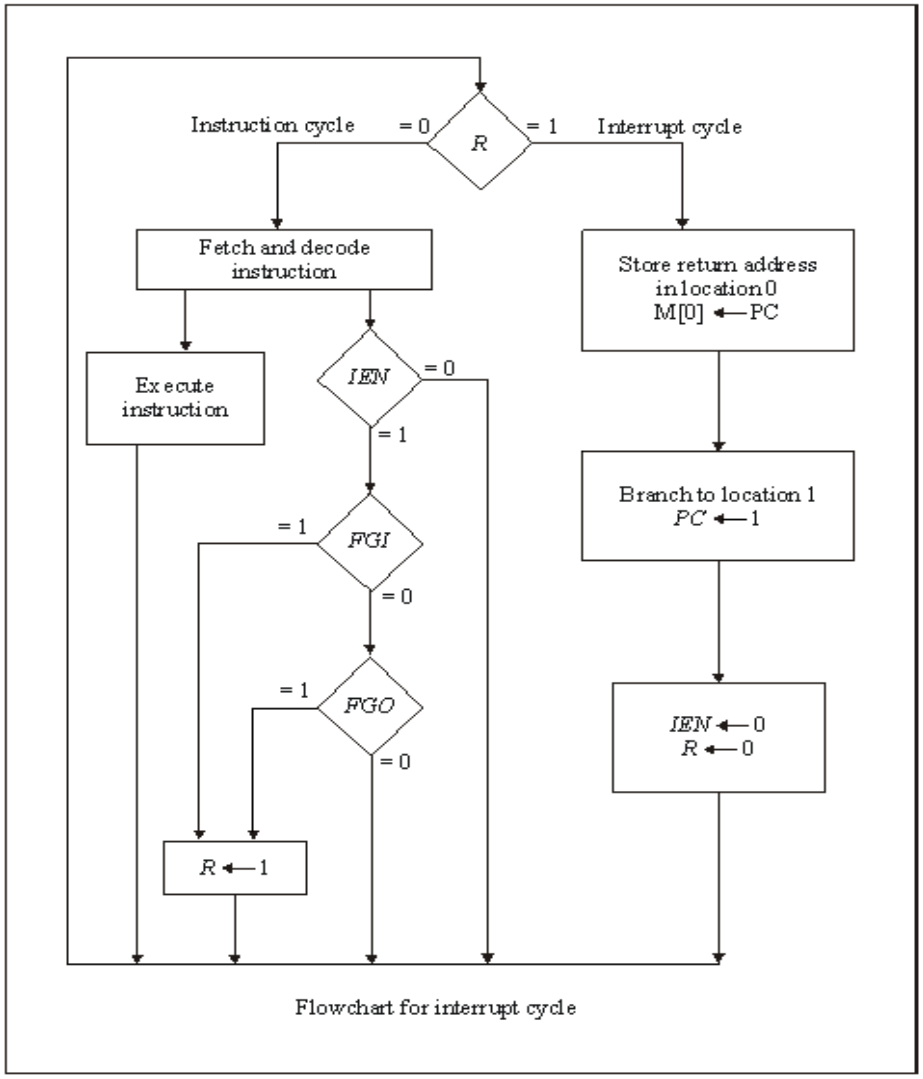
A – 1 = A + 2's complement of 1 = A + 1111



Given below

Q. 100 How does a basic computer handle an interrupt? Explain what happens during the interrupt with the help of an example. Also, give the register transfer statements.

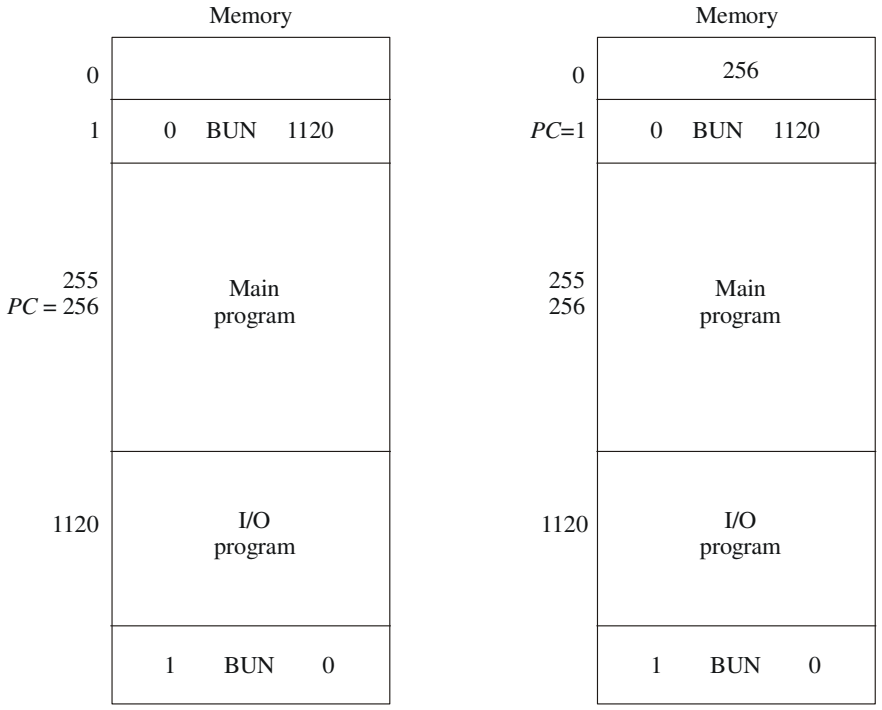
Ans.



Flowchart for interrupt cycle

The interrupt is handled by the computer can be explained by means of the flowchart. An interrupt flip-flop R is included in the computer. When R = 0, the computer goes through an instruction cycle. During the execute phase of the instruction cycle IEN in checked by the control. If it is 0, it indicates that the programmer does not want to use the interrupt, so control continues with the next instruction cycle. If IEN is 1, control checks the flag bits. If both flags are 0, it indicates that neither the input nor the output registers are ready for transfer of information. If either flag is set to 1 while IEN = 1, flip- flop R is set to 1. At the end of the execute phase, control checks the value of R, and if it is equal to 1, it goes to an interrupt cycle instead of an instruction cycle.

An example that shows, during the interrupt cycle is shown in fig. that an interrupt occurs and R is set to 1 while the control is executing the instruction at address 255. The return address 256 is in PC. The programmer has previously placed an input-output service program in memory starting from address 1120 and a BUN 1120 instruction at address 1.



Register transfer statement:-

The interrupt cycle is initiated after the last execute phase if the interrupt flip-flop is equal to 1. This flip-flop is set to 1 if IEN = 1 and either FG1 or FG0 are equal to 1. When timing signal T_0 T_1 or T_2 are active. This can be expressed with the following register transfer statements.

$$T_0'T_1'T_2'(IEN)(FGI+FGO):R \leftarrow 1$$

Q.101 Explain all the phases of instruction cycle.

Ans.

Instruction Cycle:-

A program residing in the memory unit of the computer consists of a sequence of instructions. The program is executed in the computer by going through a cycle for each instruction. Each instruction cycle in turn is subdivided into a sequence of sub cycles or phases. In the basic computer each instruction cycle consists of the following phases:

1. Fetch an instruction from memory.
2. Decode the instruction
3. Read the effective address from memory if the instruction has an indirect address.
4. Execute the instruction.

Fetch and Decode:-

The program counter PC is loaded with the address of the first instruction in the program. The sequence counter SC is cleared to 0, providing a decoded timing signal T_0. After each clock pulse, SC is incremented by one, so that the timing

signals go through a sequence T_0, T_1, and so on. The microoperation for the fetch and decode phases can be specified by the following register transfer statements.

T_0 : AR ← PC

T_1 : IR ← M[AR], PC ← PC + 1

T_2 : $D_0 \ldots D_7$ ← Decode IR(12-14), AR ← IR (011),
I ← IR(15)

To provide the data path for the transfer of PC to AR we must apply timing signal T_0 to achieve the following connection:

1. Place the content of PC onto the bus by making the bus selection inputs S_2 S_1 S_0 equal to 010.
2. Transfer the content of the bus to AR by enabling the LD input of AR.

T_1 : IR ← M[AR], PC ← PC + 1

The next check transition initiates the transfer from PC to AR since $T_0 = 1$. In order to implement the second statement T_1: IR ← M[R], PC ← PC +1
It is necessary to use timing signal T1, to provide the following connections in the bus system.

1. Enable the read input of memory.
2. Place the content of memory onto the bus by making
   S_2 S_1 S_0 = 111.
3. Transfer the content of the bus to IR by enabling the LD input of IR.
4. Increment PC by enabling the INR input of PC.

The three instruction types are subdivided into four separate paths. The selected operation is activated with the clock transition associated with timing signal T_3.

This can be symbolized as follows:

$D_7'I'T_3$: AR M[AR]

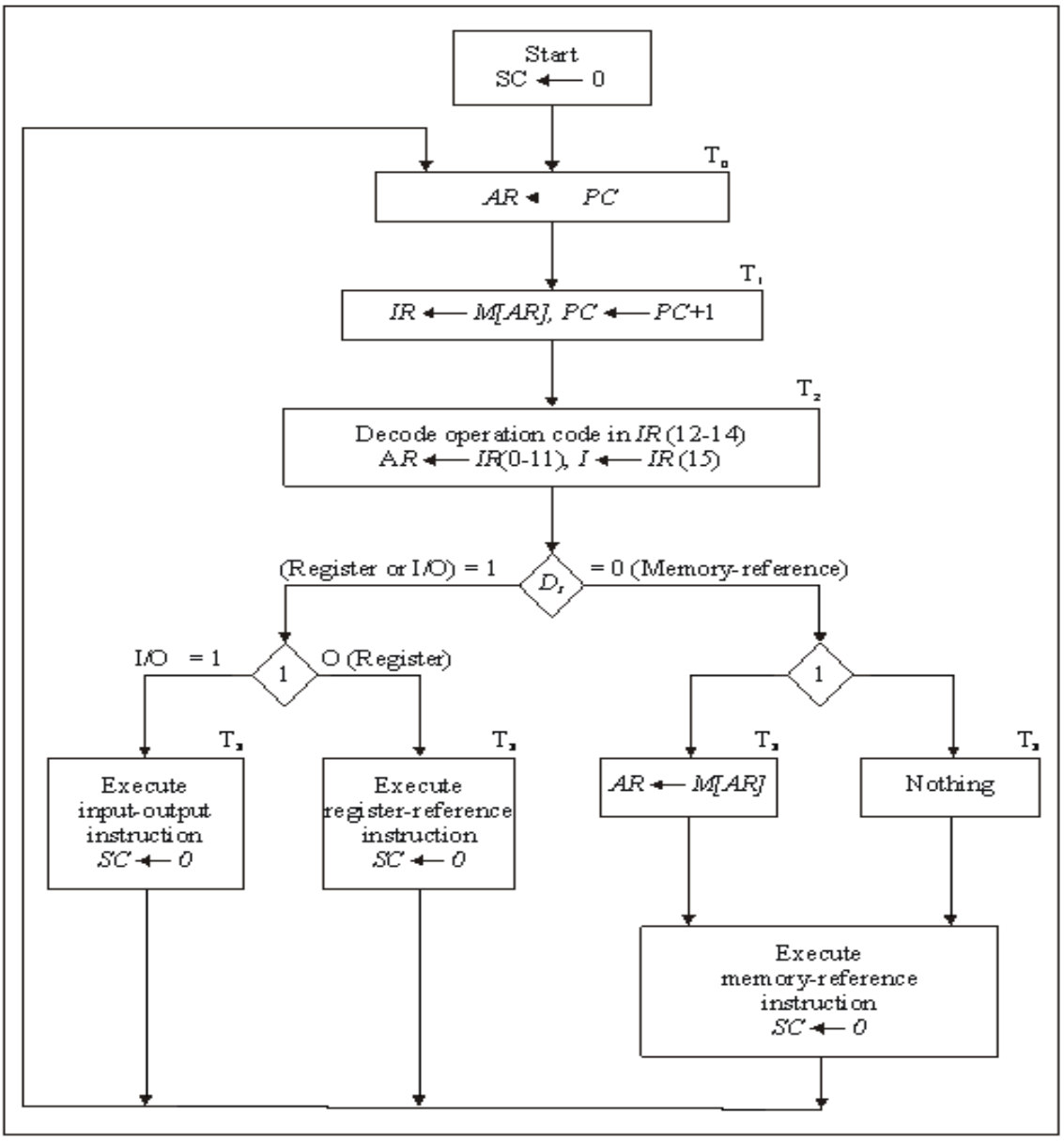
$D_7'I'T_3$: Nothing

$D_7'I'T_3$: Execute a register-reference instruction

$D_7'I'T_3$: Execute an input-output instruction

When a memory-reference instruction with I = 0 is encountered, it is not necessary to do anything since the effective address is already in AR. However, the sequence counter SC must be incremented with $D_7'I'T_3 = 1$, so that the execution of the memory-reference instruction can be continued with timing variable T_4. A register-reference or input-output instruction can be executed with the clock associated with timing signal T_3. After the instruction is executed, SC is cleared to 0 and control returns to the fetch phase with $T_o = 1$.

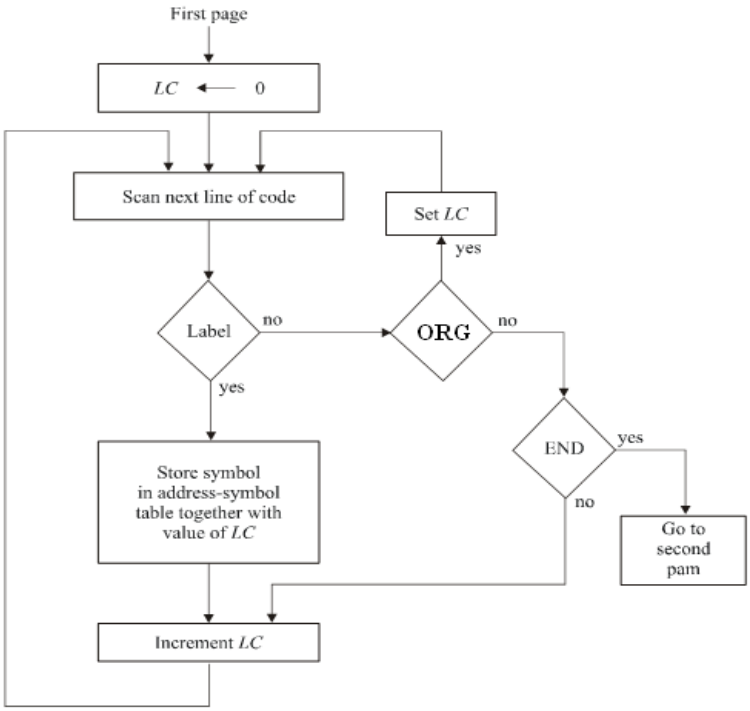
Note that the sequence counter SC is either incremented or cleared to 0 with every positive clock transition. We will adopt the convention that if SC is incremented, we will not write the statement SC ← SC + 1, but it will be implied that the control goes to the next timing signal is sequence. When SC is to be cleared, we will include the statement SC ← 0.

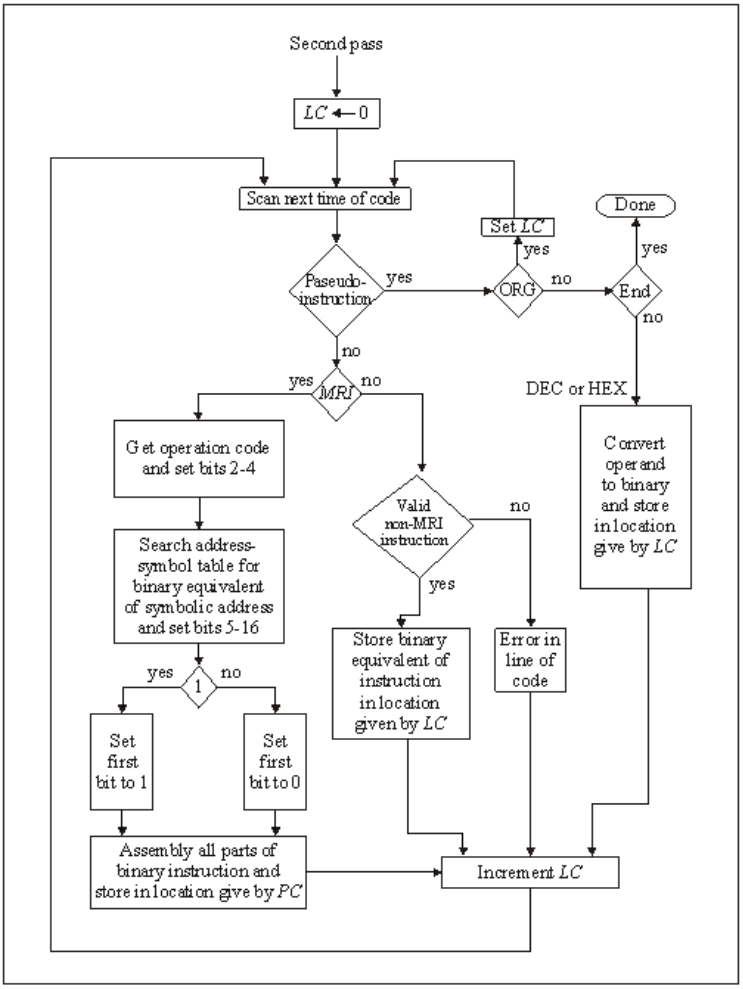


Q. 102 Explain working of two pass assembler. (Explain both pass 1 and pass 2 with flow chart).

Ans.

LC is initially set to 0. Lines of code are then analyzed one at a time. Labels are neglected during the second pass, so the assembler goes immediately to the instruction field and proceeds to check the first symbol encountered. It first checks the pseudo instruction table. A match with ORG sends the assembler to a subroutine that sets LC to an initial value. A match with END terminates the translation process. An operand pseudo instruction causes a conversion of the operand into binary. This operand is placed in the memory location specified by the content of LC. The location counter is then incremented by 1 and the assembler continues to analyze the next line of code.


First page
LC ← 0
Scan next line of code
Set LC
yes
Label
no
ORG
no
yes
END
yes
Store symbol in address-symbol table together with value of LC
no
Go to second pam
Increment LC


Second pass
LC ← 0
Scan next time of code
Set LC
yes
Done
yes
Paseudo-instruction
yes
ORG
no
End
yes
no
no
yes
MRI
no
DEC or HEX
Get operation code and set bits 2-4
Valid non-MRI instruction
no
Convert operand to binary and store in location give by LC
Search address-symbol table for binary equivalent of symbolic address and set bits 5-16
yes
Store binary equivalent of instruction in location given by LC
Error in line of code
yes
I
no
Set first bit to 1
Set first bit to 0
Assembly all parts of binary instruction and store in location give by PC
Increment LC

Q. 103 Write an assembly language program to multiply two positive numbers by a repeated addition method. For example to multiply 7*4 the program evaluated the product by adding 7 four times.

Ans.
Assembly language programme to multiply two positive numbers.

```
        ORG 100
LOP     CLE             /Clear
        LDA Y           /Load multiplier
        CIR             /Transfer multiplier bit to E
        STAY            /Store shifted multiplier
        SZE             /Check if bit is zero
        BUN ONE         /Bit is one; go to ONE
        BUN ZRO         /Bit is one; go to ZRO
ONE,    LDAX            /Load multiplicand
        ADD P           /Add to partial product
        STA P           /Store partial product
        CLE             /Clear E
ZRO,    LDA X           /Load multiplicand
        CIL             /Shift left
        STA X           /Store shifted multiplicand
        ISZ CTR           /Increment counter
        BUN LOP           /Counter not zero; repeat loop
        HLT               /Counter is zero; halt
CTR,    DEC – 8           /This location serves as a counter
X,      HEX 000F          /Multiplicand stored here
Y,      HEX 000B          /Multiplier stored here
P,      HEX 0             /Product formed here
```

Q. 104 Differentiate between the following:
(i) Autoincrement and Autodecrement addressing mode.
(ii) Program interrupt and subroutine call & return.

Ans.
Autoincrement and Autodecrement mode :-
The register is incremented or decremented after (or before) its value is used to access memory. The address stored in the register refers to a label of data in memory, it is necessary to increment or decrement the register after every access. This is achieved by using the increment or decrement instruction.

Subroutine call and Return :-
A subroutine call is a self contained sequence of instructions that performs a given computational task. During the execution of program, a subroutine may be called to perform its function many times at various points in the main program.
The BSA instruction performs the function usually referred to as a subroutine call. The indirect BUN instruction at the end of the subroutine performs the function referred to as a subroutine return. In most commercial computers, the return address associated with a subroutine is stored in either a processor register or in a portion of memory called a stack. When a subroutine is called, the main program must transfer the data. The subroutine shifted the number and left it there to be accepted by the

main program. It is necessary for the subroutine to have access to data from the calling program and to return results to that program. The accumulator can be used for a single input parameter and a single output parameter consider a subroutine that performs the logic OR operation. Two operands must be transferred to the subroutine and the subroutine must return the result of the operation.

Q. 105 What is a microinstruction? Write a micro instruction code format and explain all the fields in it.

Ans.
Each word in control memory contains within it a microinstruction. The microinstruction specifies one or more micro-operations for the system. A sequence of microinstruction constitutes a microprogram. It is an instruction stored in control memory.

OP code								
Computer Instruction		0	1	1	1			address
Mapping Bits	0	x	x	x	x	0	0	
Microinstruction address	0	1	0	1	1	0	0	

Instruction code to microinstruction address

A special type of branch exist when a microinstruction specifies a branch to the first word in control memory where a microprogram routine for an instruction is located. The status bits for this type of branch are the bits in the operation code part of the instruction. Micro instruction format is 20 bits in length. It divided into four functional parts. The three fields F_1, F_2 and F_3 specify micro operations for the computer. The CD field selects status bits condition:
* the BR field specifies the type of Branch to be used.
* the AD field contains a branch address. The address field is even bits wide, since the control memory has $128 = 2^7$ words.

F_1, F_2, F_3 : Micro operation fields

3	3	3	2	2	7
F1	F2	F3	CD	BR	AD

CD : Condition for branching
BR : Branch field
AD : Address field

* Micro operations are sub-divided into three fields of three bits each. These bits in each field are encoded to specify seven distinct micro operations. This gives 21 micro operations. If fewer than three micro operations are used, one or more of the fields will use the binary code 000 for no operation.

Symbols and Binary Code for Micro Instruction Fields

F_1	Micro operation	Symbol
000	None NOP	
001	AC ← AC + DR	ADD
010	AC ← 0	CLRAC
011	AC ← AC + 1	INCAC
100	AC ← DR	DRTAC
101	AR ← DR (0-10)	DRTAR
110	AR ← PC	DCTAR
111	M [AR] ← DR	WRITE

F_2	Micro Operation	Symbol
000	None NOP	
001	AC ← AC-DR	SUB
010	AC ← AC ∨ DR	OR
011	AC ← AC ∧ DR	AND
100	DR ← M[AR]	READ
101	DR ← AC	ACTDR
110	DR ← DR+1	INCDR
111	DR(0-10) ← PC	PCTDR

F_3	Micro Operation	Symbol
000	None NOP	
001	AC ← AC ⊕ DR	XOR
010	AC ← AC COM	
011	AC ← shl AC	SHL
100	AC ← shr AC	SHR
101	PC ← PC+1	INCPC
110	PC ← AR	ARTPC
111	Reserved	

CD	Condition	Symbol	Comments
00	Always=1	U	Unconditioned branch
01	DR(15)	I	Indirect address bit
10	AC(15)	S	Sign bit of AC
11	AC=0	Z	Zero value in AC

BR	Symbol	Function
00	JMP	CAR ← AD, if condition=1 CAR ← CAR+1, if condition=0
01	CALL	CAR ← AD, SBR ← CAR+1, if condition=1 CAR ← CAR+1, if condition=0
10	RET	CAR ← SBR (return from subroutine)
11	MAP	CAR(2-5) ← DR(11-14), CAR(0, 1, 6) ← 0

Q. 106 What is a microprogram? Write a microprogram for the fetch routine.

Ans.

A sequence of microinstructions constitutes a microprogram. The use of a micro program involves placing all control variables in words of ROM for use by the control unit through successive record operation.

Binary Microprogram for Control Memory (Partial)

Micro Routine	Address Decimal	Address Binary	Binary Microinstruction F1	F2	F3	CD	BR	AD
	0	0000000	000	000	000	01	01	1000011
	1	0000001	000	100	000	00	00	0000010
	2	0000010	001	000	000	00	00	1000000
	3	0000011	000	000	000	00	00	1000000
BRANCH	4	0000100	000	000	000	10	00	0000110
	5	0000101	000	000	000	00	00	1000000
	6	0000110	000	000	000	01	01	1000011
	7	0000111	000	000	110	00	00	1000000
STORE	8	0001000	000	000	000	01	01	1000011
	9	0001001	000	101	000	00	00	0001010
	10	0001010	111	000	000	00	00	1000000
	11	0001011	000	000	000	01	01	1000011
EXCHANGE	12	0001100	000	000	000	01	01	1000011
	13	0001101	001	000	000	00	00	0001110
	14	0001110	100	101	000	00	00	0001111
	15	0001111	111	000	000	00	00	1000000
FETCH	64	1000000	110	000	000	00	00	1000000
	65	1000001	000	100	101	00	00	1000010
	66	1000010	101	000	000	00	11	0000000
INDRCT	67	1000011	000	010	000	00	00	1000100
	68	1000100	101	000	000	00	10	0000000

Q. 107 Formulate a four segment instruction pipeline for a computer. Specify the operation to be performed in each segment.

Ans.

Instruction pipelines :- An instruction pipeline reads consecutive instructions from memory while previous instructions are being executed in other segments. This causes the instruction fetch and execute phases to overlap and perform simultaneous operations. The instruction fetch segment can be implemented by means of a first-in, first-out (FIFO) buffer. This is a type of unit that forms a queue rather than a stack. The execution unit is not using memory, the control increments the program counter and uses its address value to read consecutive instructions from memory. An instruction stream can be placed in a queue, waiting for decoding and processing by the execution segment. The instruction stream queuing mechanism provides an efficient way for reducing the average access time to memory for reading instructions. The computer needs to process each instruction with the following sequence of steps.

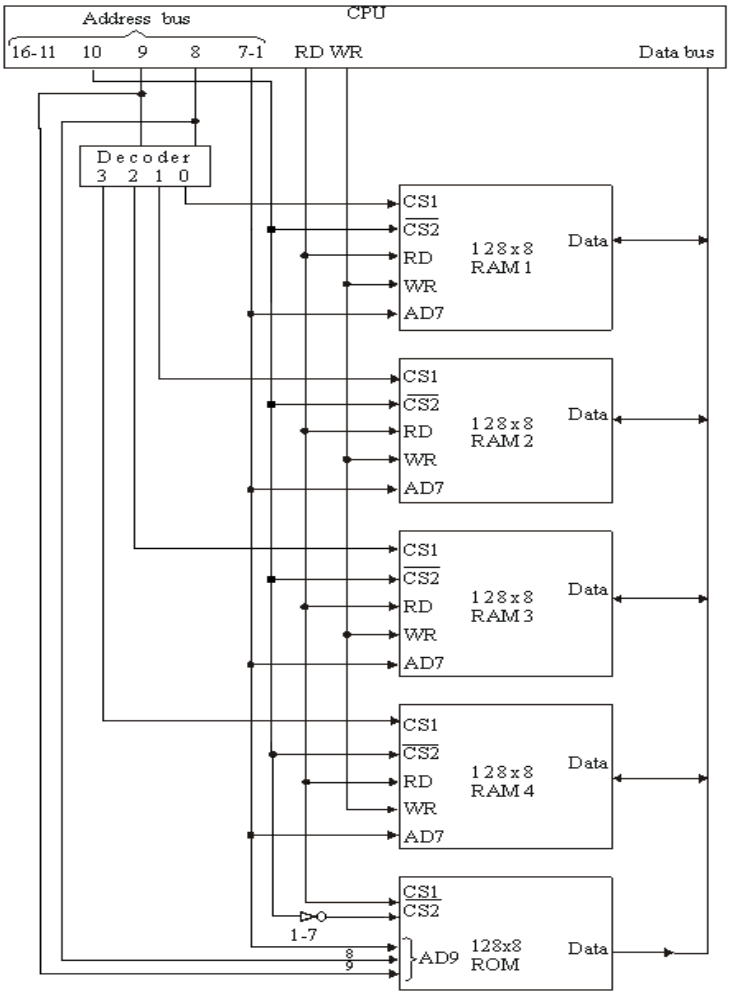
1. Fetch the instruction from memory.
2. Decode the instruction.

3. Calculate the effective address.
4. Fetch the operands from memory.
5. Execute the instruction.
6. Store the result in the proper place.

There are certain difficulties that will prevent the instruction pipeline from operating at its maximum rate. Different segments may take different times to operate on the incoming information. Some segments are skipped for certain operations. For example, a register mode instruction does not need an effective address calculation. Two or more segments may require memory access at the same time, causing one segment to wait until another is finished with the memory.
The design of an instruction pipeline will be most efficient if the instruction cycle is divided into segments of equal duration. The time that each step takes to fulfill its function depends on the instruction and the way it is executed.

Q. 108 Show the memory organization (1024 bytes) of a computer with four 128x8 RAM Chips and 512x8 ROM Chip. How many address lines are required to access memory.

Ans.
Memory organization:-



Address lines:-

128 = 128 = 2^7

512 = 2^9

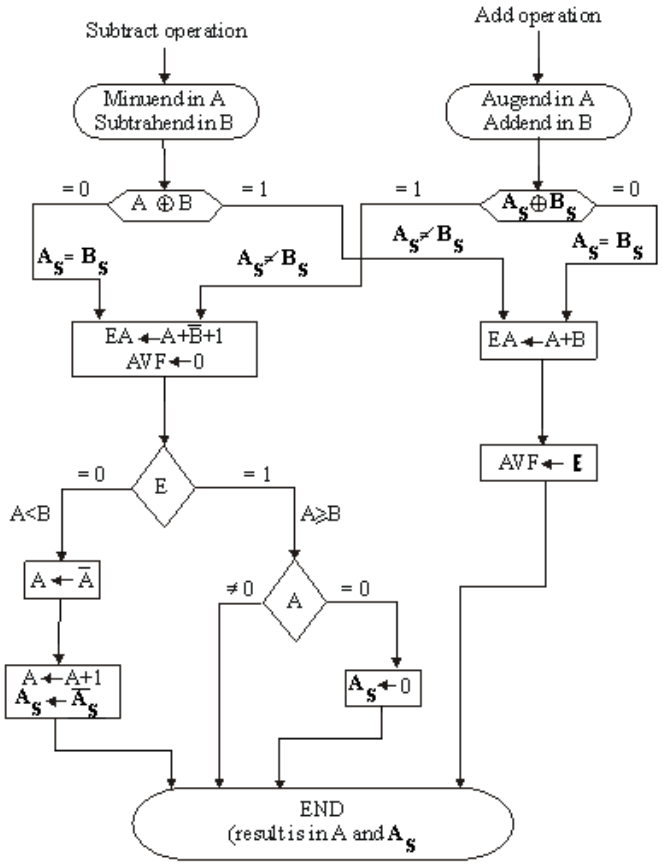
Address lines = 16

Q. 109 Write a general algorithm and flow chart for addition and subtraction of two signed magnitude Numbers.

Ans.

Hardware Algorithm:-

The flowchart for the hardware algorithm is presented. The two signs A_s, and B_s are compared by an exclusive-OR gate. If the output of the gate is 0, the signs are identical; if it is 1, the signs are different. For an add operation, identical signs dictate that the



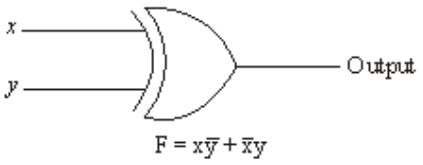
magnitudes are added. For a subtract operation, different signs dictate that the magnitudes be added. The magnitudes are added with a microoperation EA A + B, where EA is a register that combines E and A. The carry in E after the addition constitutes an overflow if it is equal to 1. The value of E is transferred into the add-overflow flip-flop AVF.

Q.110 Ram wants to purchase a bicycle. The bicycle must have brakes. The bicycle which has either a hand brake or foot brake, No bicycle has both type of brakes. Implement the same using basic gates.

Ans.

Suppose hand break = 0 = x
foot break = 1 = y
Truth table obtain by EX - OR gate

x	y	Output
0	0	0
1	0	1
0	1	1
1	1	0

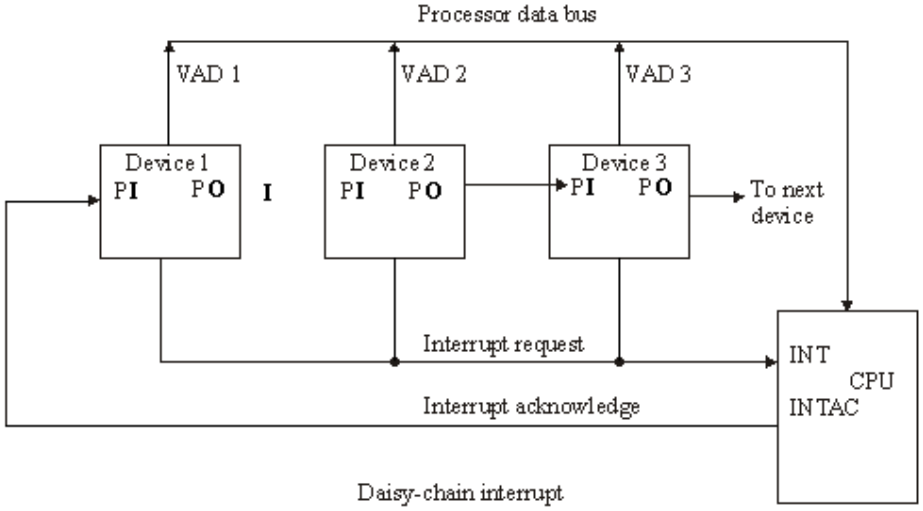


Q. 111 Write short notes on followings
(i) Daisy chaining priority.
(ii) Direct Memory Access.
(iii) Handshaking method for data transfer.
(iv) Associative Memory

Ans.

(i) **Daisy chaining priority**:-

The daisy-chaining method of establishing priority consists of a serial connection of all devices that request an interrupt.



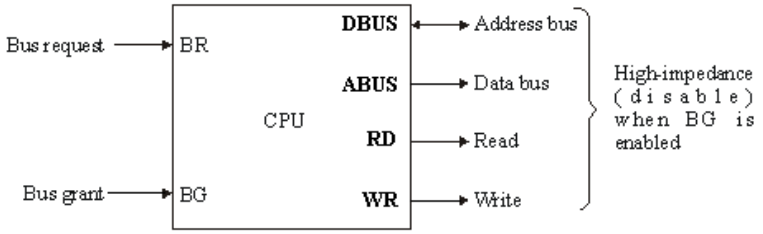
The device with the highest priority is placed in the first position, followed by lower-priority devices up to the device with the lowest priority, which is placed last in the chain. This method of connection between three devices and the CPU is shown. The interrupt request lines is common to all devices and forms a wired logic connection. If any device has its interrupt signal in the low-level state, the interrupt line goes to the low-level state and enables the interrupt input in the CPU. When no interrupts are pending, the interrupt line stays in the high-level state and no interrupts are recognized

by the CPU. This is equivalent to a negative logic OR operation. The CPU responds to an interrupt request by enabling the interrupt acknowledge line.

(ii) **Direct Memory Access**:-

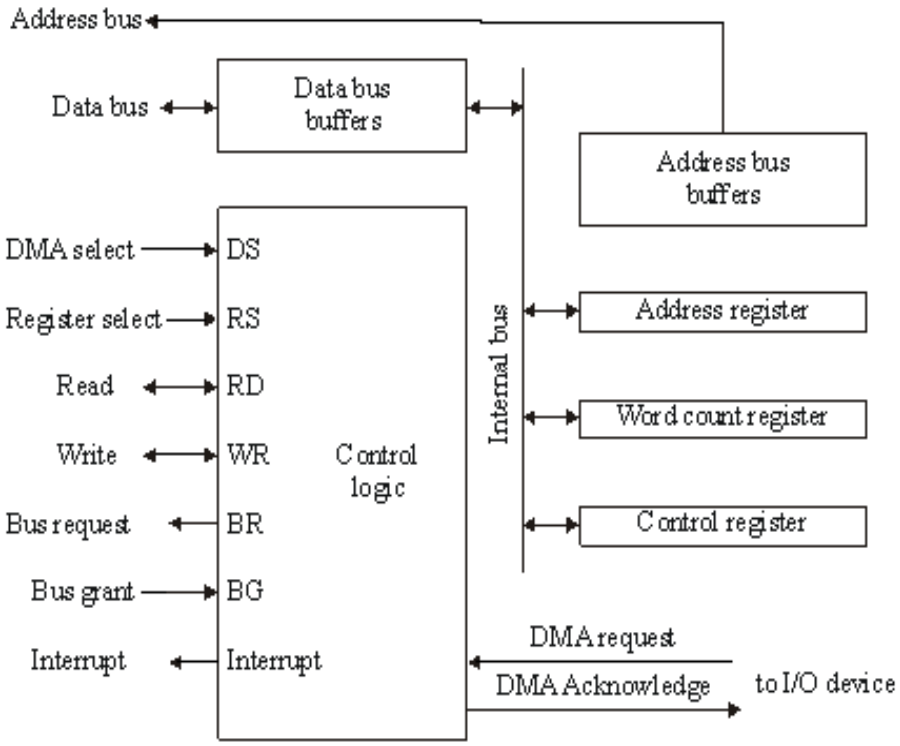
The transfer of data between a fast storage device such as magnetic disk and memory is often limited by the speed of the CPU. Removing the CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of transfer. This transfer technique is called direct memory access (DMA). During DMA transfer, the CPU is idle and has no control of the memory buses.

CPU bus signals for DMA transfer



The figure shows two control signals in the CPU that facilitate the DMA transfer. The bus request (BR) input is used by the DMA controller to request the CPU to relinquish control of the buses. The CPU activates the bus grant (BG) output to inform the external DMA that the buses are in the high-impedance state. The DMA that originated the bus request can now take control of the buses

Block diagram of DMA controller

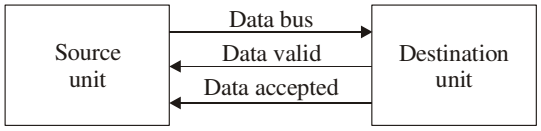


to conduct memory transfers without processor intervention. When the DMA terminates the transfer, it disables the bus request line. The CPU disables the bus grant. When the DMA takes control of the bus system, it communicates directly with the memory.

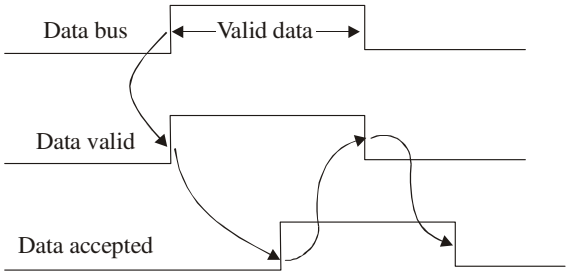
(iii) Handshaking method for data transfer:-

Source-initiated data transfer:- The disadvantage of the strobe method is that the source unit that initiates the transfer has no way of knowing whether the destination unit has actually received the data item that was placed in the bus. Similarly, a destination unit

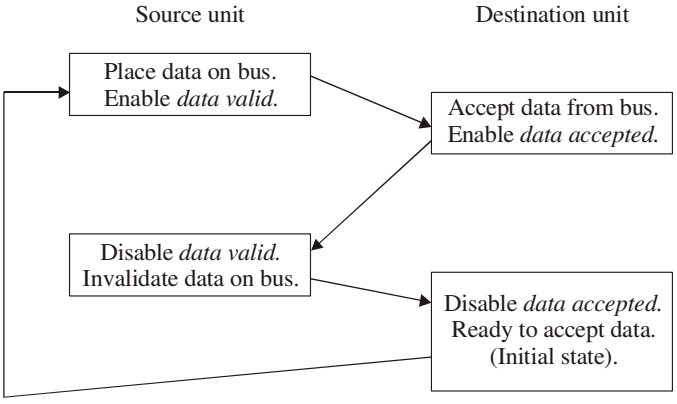
Input output organization



(a) Block diagram



(b) Timing diagram



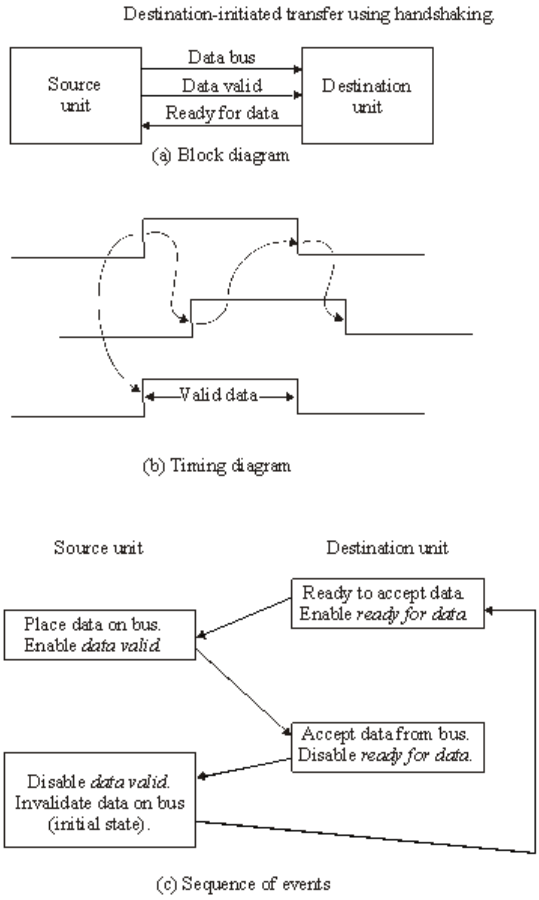
(c) Sequence of events

that initiates the transfer has no way of knowing whether the source unit has actually placed the data on the bus. The two handshaking lines are *data valid,* which is generated by the source unit, and *data accepted,* generated by the destination unit. The timing diagram shows the exchange of signals between the two units. The sequence of events listed in part (c) shows the four possible states that the system can be at any given time. The source unit initiates the transfer by placing the data on the bus and enabling its *data valid* signal. The *data accepted* signal is activated by the destination unit after it accepts the data from the bus. The source unit then

disables its *data valid* signal, which invalidates the data on the bus. The destination unit then disables its *data accepted* signal and the system goes into its initial state. The source does not send the next data item unit after the destination unit shows its readiness to accept new data by disabling its *data accepted* signal. This scheme allows arbitrary delays from one state to the next and permits each unit to respond at its own data transfer rate. The rate of transfer is determined by the slowed unit.

Destination initiated data transfer:-

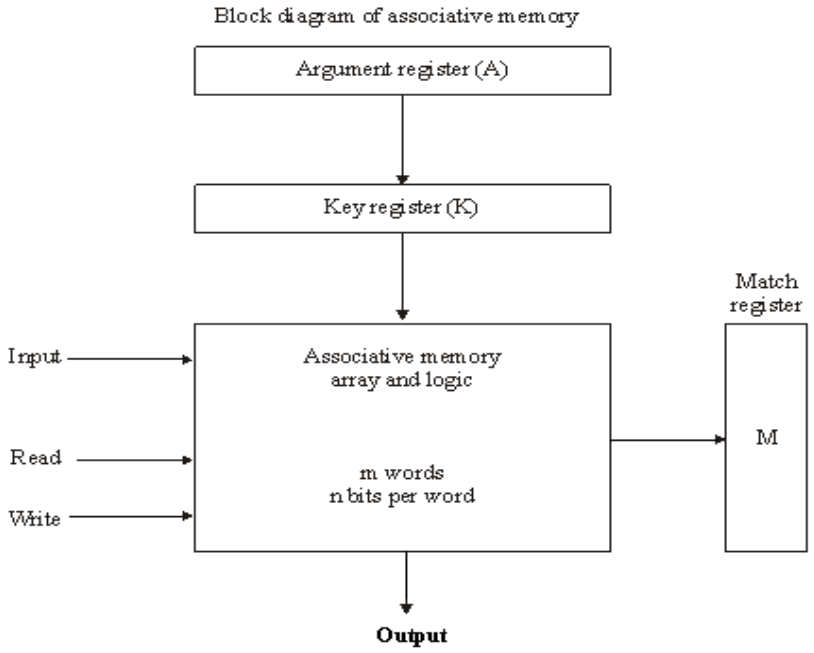
The destination-initiated transfer using handshaking lines. Note that the name of the signal generated by the destination unit has been changed to *ready for data* to reflect its new meaning. The source unit in this case does not place data on the bus until after it receives the *ready for data* signal from the destination unit. From there on, the handshaking procedure follows the same pattern as in the source-initiated case. Note that the sequence of events in both cases would be identical if we consider the *ready for data* signal as the complement of *data accepted*. In fact, the only difference between the source-initiated and the destination-initiated transfer is in their choice of initial state.



This disadvantage of the strobe method is that the source unit that initiates the transfer has no way of knowing whether the destination unit has actually received the data item that was placed in the bus. Similarly, a destination unit that initiates the transfer has no way of knowing whether the source unit has actually placed the data on the bus.

(iv) **Associative Memory**:-

The time required to find an item stored in memory can be reduced considerably if stored data can be identified for access by the content of the data itself rather than by an address. A memory unit access by the content of the data itself rather than by an address. A memory unit accessed by content is called an associative memory or content addressable memory (CAM) . This type of memory is accessed simultaneously and in parallel on the basis of data content rather than by specific address or location.



Q.112 a. Show the Truth Table's for the Following functions:-

i) $f(w, x, y, z) = w + x + y + z$

ii) $f(w, x, y, z) = wx + xz + \bar{y}$

Ans. (i) $f(w, x, y, z) = w + x + y + z$

w	x	y	z	output
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
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

(ii) $f(w, x, y, z) = wx + xz + \bar{y}$

w	x	y	z	Output= $wx + xz + \bar{y}$
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

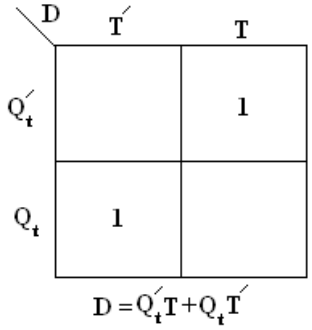
Q. 113 Construct a T flip flop using a
i) D - FF ii) J-K FF

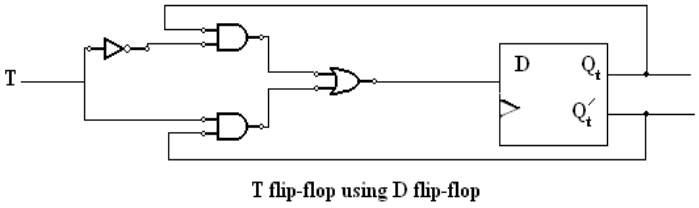
Ans. (i)
Conversion of D flip-flop to T flip-flop

Truth table of T		
Q_t	**T**	**Q_{t+1}**
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**0**

Excitation table of D		
Q_t	**Q_{t+1}**	**D**
0	**0**	**0**
0	**1**	**1**
1	**0**	**0**
1	**1**	**1**

Conversion table			
Q_t	**T**	**Q_{t+1}**	**D**
0	**0**	**0**	**0**
0	**1**	**1**	**1**
1	**0**	**1**	**1**
1	**1**	**0**	**0**





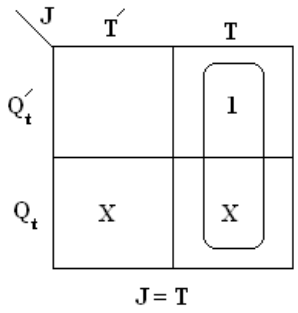
T flip-flop using D flip-flop

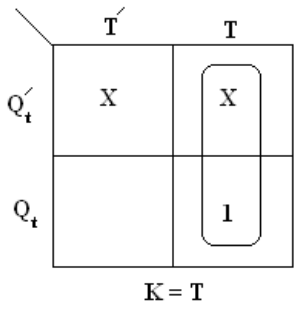
(ii)
Conversion of JK flip- flop to T flip-flop

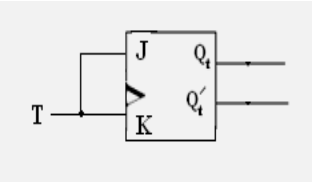
Truth table of T		
Q_t	T	Q_{t+1}
0	0	0
0	1	1
1	0	1
1	1	0

Excitation table of JK			
Q_t	Q_{t+1}	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Conversion table				
Q_t	T	Q_{t+1}	J	K
0	0	0	0	x
0	1	1	1	x
1	0	1	x	0
1	1	0	x	1



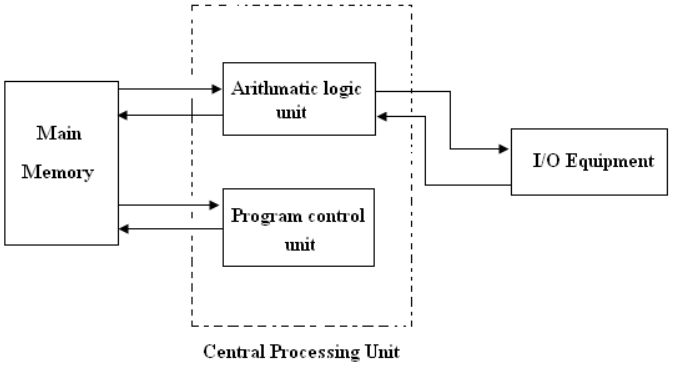




Q. 114 Explain Ven Neumann architector and stored program concept.

Ans.

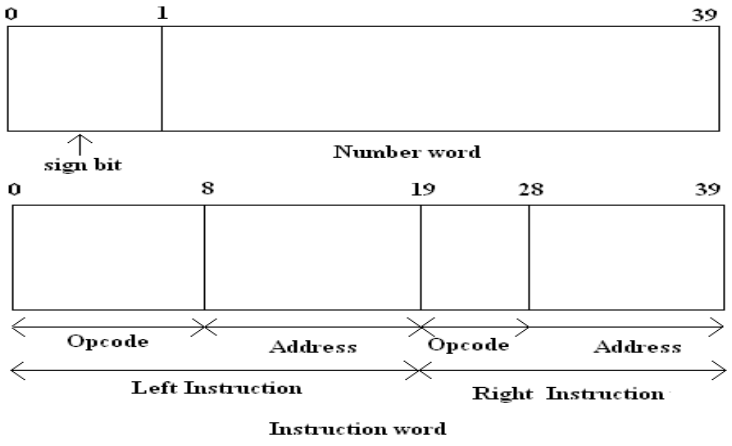
In 1946, John Von Neumann and his colleagues began the design of a new stored – program computer referred to as IAS computer. The general structure of IAS computer is



It consists of the following:

(a) A main memory, which stores both data and instructions.
(b) An arithmetic and logic unit (ALU) capable of operating on binary data.
(c) A control unit, which interprets the instructions in memory and causes them to be executed.
(d) Input and output (I/O) equipment operated by the control unit.

The memory of the IAS consists of 1000 storage locations, called words of 40 binary digits (bits) each. Both data and instructions are stored there. Thus the numbers must be represented in binary form, and each instruction also has to be in a binary code, in their respective formats as below.



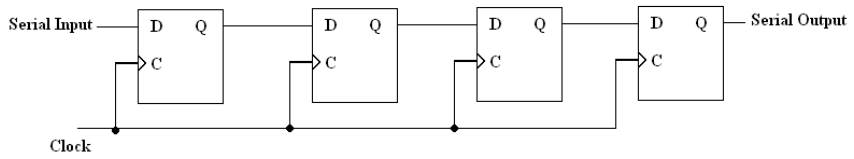
Each number is represented by a sign bit and a 39- bit value. A word may also contain two 20- bit instruction, with each instruction of an 8-bit operation code (opcode) specifying the operation to be performed and a 12-bit address designated one of the words in memory. John Von Neumann gave the idea of storing 'Programme' and 'Data' in the same memory. Storing of programs in memory helps in executing series of instructions repetitively. It makes the operation of computer automatic.

Q. 115 Show the hardware to implement the following micro-operations.

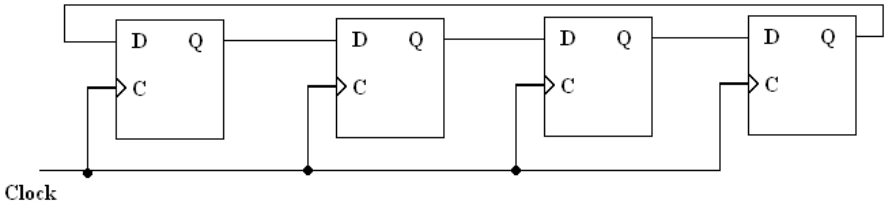
(i) L : α shl (x) (ii) α : cir (x)

'x' consist of four D-FFs.

Ans. (i) L : α shl (x)



(ii) α : cir (x)



Q. 116 Discuss the properties of an ideal instruction set computer.

Ans.

(i) A computer has a set of instruction so that the user can construct machine language programs to evaluate any function.

(ii) Input and output instructions are needed for communication between the computer and the user.

(iii) Programs and data transferred into memory and results of computations transferred back to the user.

Q. 117 Explain instruction cycle. Implement the RTLs of fetch phase.

Ans.

A program residing in the memory unit of the computer consists of a sequence of instructions. The program is executed in the computer by going through a cycle for each instruction. In the basic computer each instruction cycle consists of the following phases:

(1) Fetch an instruction from memory

(2) Decode the instruction

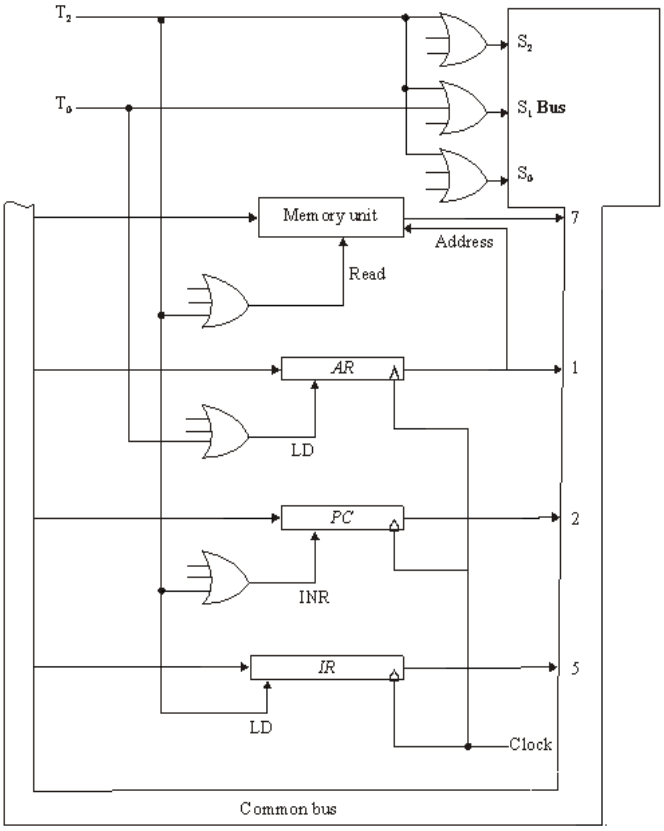
(3) Execute the instruction

Fetch

The program counter PC is loaded with the address of the first instruction in the program. The sequence counter SC is cleared to 0, providing a decoded tining signal T_o. After each clock pulse, SC is incremented by one.

T_o : AR ← PC

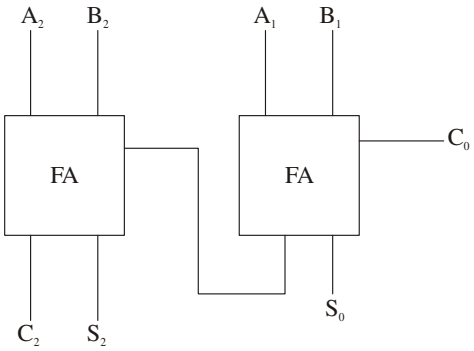
T_1 : IR ← M[AR], PC ← PC+1



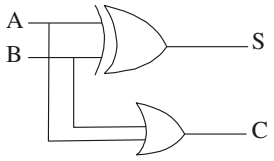
Q. 118 Design a 2-bit adder and logic circuit capable of perfering AND ADD, complement and shift left operation.

Ans.

2 bit adder

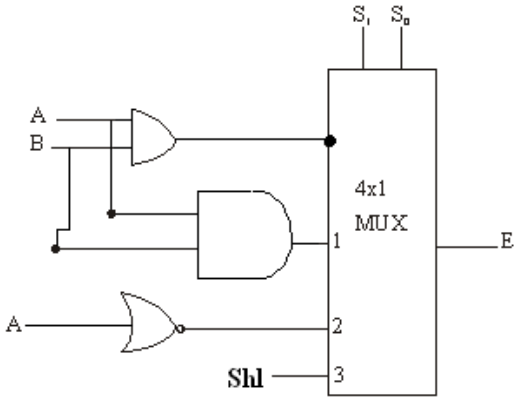


Logic Circuit



A	B	S	C
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

S1	S0	Output	Operation
0	0	E=A^B	AND
0	1	E=A ∨ B	ADD
1	0	E=A	Complement
1	1	Shl	



Q. 119 Discuss the different addressing modes of an instruction.

Ans.

In this mode the operands are specified implicitly in the definition of the instruction.

Immediate Mode:- In this mode the operand is specified in the instruction itself. In other words, an immediate-mode instruction has an operand field rather than an address field. The operand field contains the actual operand to be used in conjunction with the operation specified in the instruction.

Register mode:- In this mode the operands are in registers that reside within in CPU. The particular register is selected from a register field in the instruction. A k-bit field can specify any one of 2^k registers.

Register Indirect mode:- In this mode the instruction specifies register in CPU whose contents give the address of the operand in memory. Before using a register indirect mode instruction, the programmer must ensure that the memory address of the operand is placed in the processor register with a previous instruction.

Autoincrement or Autodecrement Mode:- This is similar to the register indirect mode except that the register is incremented or decremented after for before) its value is used to access memory. When the address stored in the register refers to a table of data is memory. It is necessary to increment or decrement the register after every access to the table.

Direct Address Mode:- In this mode the effective address is equal to the address part of the instruction.

Indirect Address Mode:- In this mode the address field of the instruction gives the address where the effective address is stored in memory. Control fetches the instruction from memory and uses its address part to access memory again to read the effective address.

effective address = address part of instruction + content of CPU register

Relative Address Mode:- In this mode the content of the program counter is added to the address part of the instruction in order to obtain the effective address. When this number is added to the content of the program counter, the result produces an

effective address whose position in memory is relative to the address of the next instruction.

Indexed Addressing Mode:- In this mode the content of an index register is added to the address paid of the instruction to obtain the effective address.

Base Register Addressing Mode:- In this mode the content of a base register is added to the address part of the instruction to obtain the effective address.

Q.120 What is the significance of program status word.

Ans.

The collection of all status bit condition in the CPU is called a program status word or PSW. The PSW is stored in a separate hardware register and contains the status information that characterizes the state of the CPU. The CPU does not respond to an interrupt until the end of an instruction execution. Before going to the next fetch phase, control checks for any interrupt signals.

If an interrupt is pending, control goes to hardware interrupt cycle. Desiring this cycle, the contents of PC and PSW are pushed onto the stack. The PSW is transferred to the status register and the return address to the program counts. The CPU state is restored and the original program continues executing.

Q.121 What is software interrupt? State it use.

Ans.

A software interrupt is initiated by executing an instruction. Software interrupt is a special call instruction that behaves like an interrupt rather than a subjective call. The most common use of software interrupt is associated with supervision call instruction. This instruction provides means for switching from a CPU user mode to the supervision mode. A software interrupt that steers the old CPU state and brings is new PSW that belongs to the supervisor mode.

Q. 122 What is the difference between 1's complement subtraction and 2's complement subtraction of binary numbers? Show it by example.

Ans.

One's complement representation:-

In a binary number, each 1 is replaced by 0 and 0 by 1, the resulting number is known as the one's complement of the first number. If one of these numbers is positive then the other number will be negative with the some magnitude and vice-versa.

Two's complement Representation :-

If 1 added to 1's complement of a binary number, the resulting number is known as the two's complement of the binary number. For example, 2's complement of 0101 is 1011. In this representations, if the MSB is 0 the number is positive, whereas if the MSB is 1 the number is negative. For an n bit number the maximum positive number is $(2^{n-1}-1)$ and the maximum negative number is -2^{n-1}

Q. 123 Show the step-by-step multiplication process using booth's algorithm, when +14 is multiplied by -14 Assume 5-bit registers that hold signed numbers.

Ans.

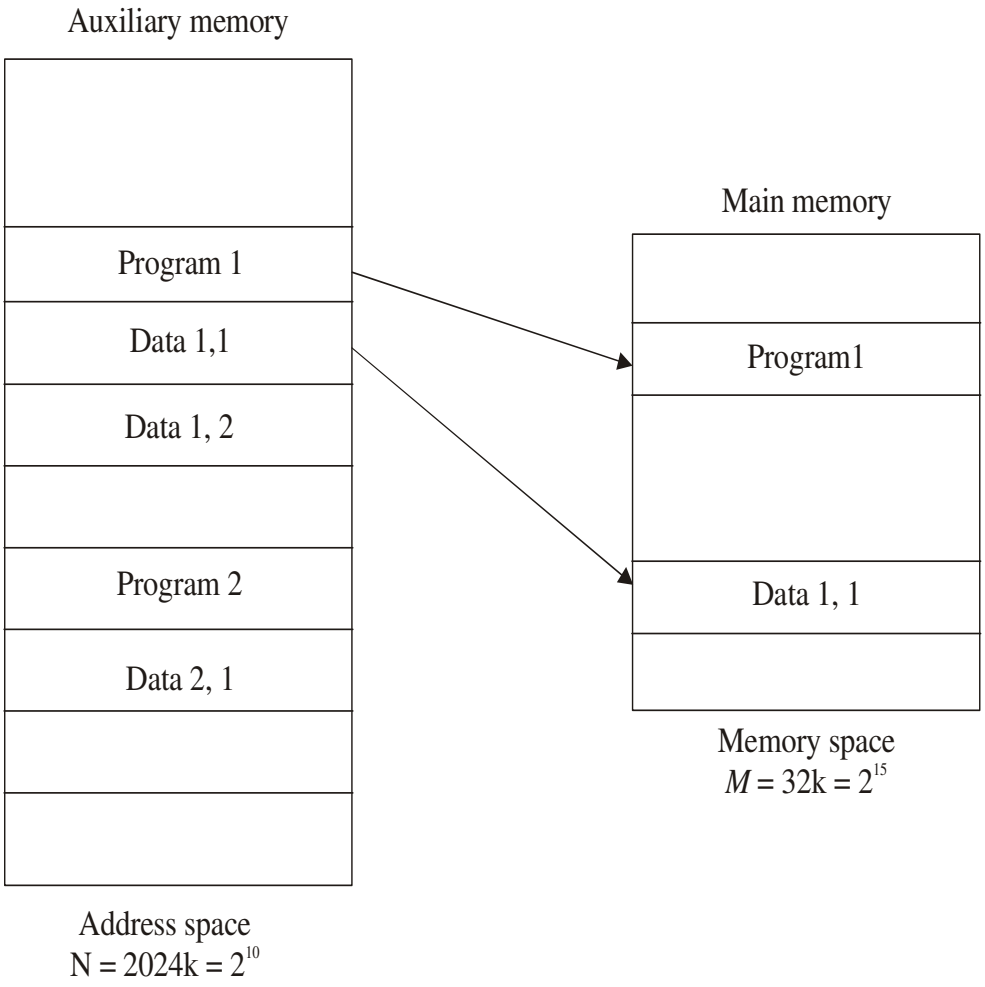
Q_n	Q_{n+1}	BR=01110 $\overline{BR}+1=10010$	AC	QR	Q_{n+1}	SC
		Initial	00000	10010	0	101
0	0	ashr	00000	01001	0	100
1	0	Subtract BR	10010 10010			
		ashr	11001	00100	1	011
0	1	Add BR	01110 00111			
		ashr	00011	10010	0	010
0	0	ashr	00001	11001	0	001
1	0	Subtract BR	10010 10011			
		ashr	11001	11100	1	000
Final Product = 1100111100						

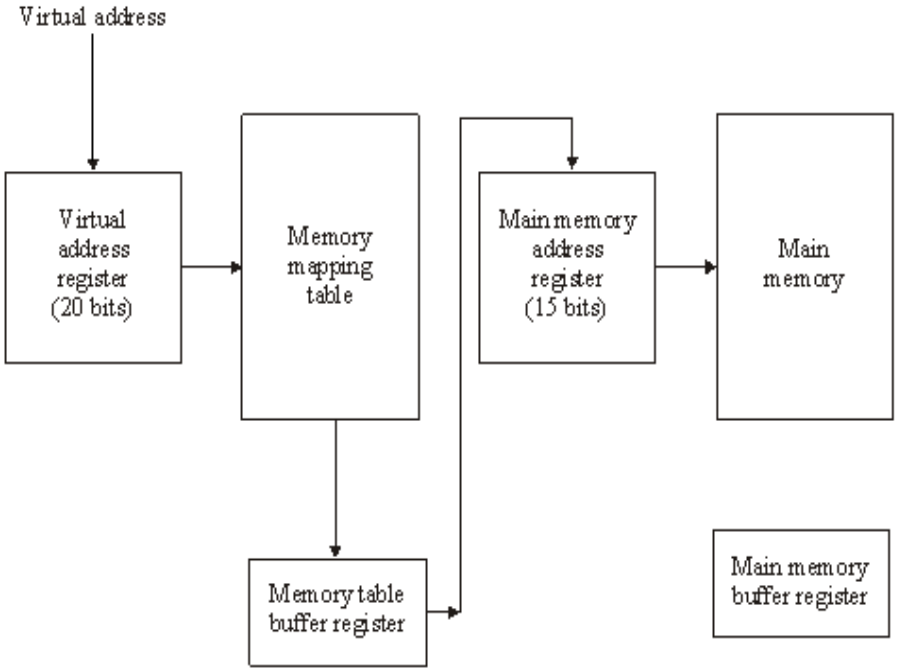
Q.124 Explain the use of time out mechanism in handshaking data transfer scheme.

Ans.
The handshaking scheme provides a hight degree of flexibility and reliability because the successful completion of a data transfer relies on active participation by both units. If one unit is faulty, the data transfer will not be completed. An error can be detected by means of a timeout mechanism, which produces an alarm if the data transfer is not completed within a predetermined time. The timeout signal used for interrupt the processor and hence executes a service routine that takes appropriate error recover action.

Q.125 Explain virtual memory & its mapping scheme.

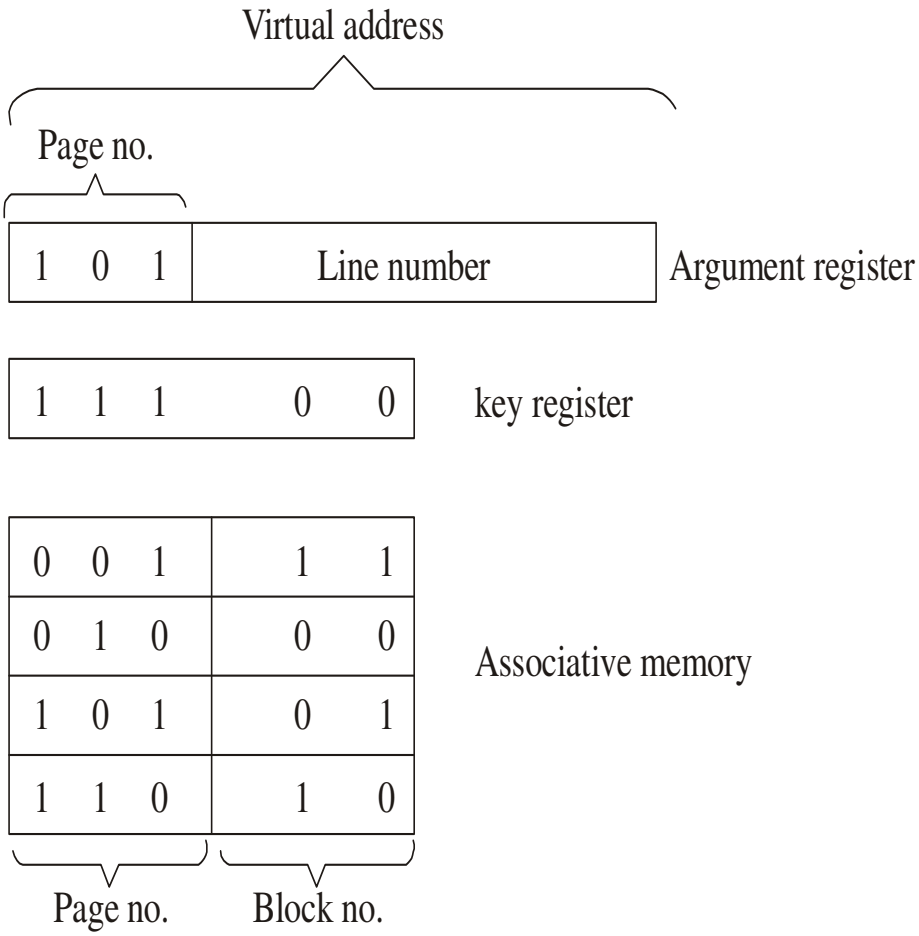
Ans.
Virtual memory is a concept used in some large computer systems that permit the user to construct programs as though a large memory space were available, equal to the totality of auxiliary memory. Virtual memory is used to give programmers the illusion that they have a very large memory at their disposal, even though the computer actually has a relatively small main memory. A virtual memory system provides a mechanism for translating program-generated addresses into correct main memory locations. In a virtual memory system, programmers are told that they have the total address space at their disposal. Moreover, the address field of the instruction code has a sufficient number of bits to specify all virtual addresses. In our example, the address field of an instruction code will consist of 20 bits but physical memory addresses must be specified with only 15 bits. Thus CPU will reference instructions and data with a 20-bit address, but the information. at this address must be taken from physical memory because access to auxiliary storage for individual words will be prohibitively long.


Auxiliary memory
Program 1
Data 1,1
Data 1, 2
Program 2
Data 2, 1
Address space
N = 2024k = 2^10
Main memory
Program1
Data 1, 1
Memory space
M = 32k = 2^15


Virtual address
Virtual address register (20 bits)
Memory mapping table
Main memory address register (15 bits)
Main memory
Memory table buffer register
Main memory buffer register

to map a virtual address of 20 bits to a physical address of 15 bits. The mapping is a dynamic operation, which means that every address is translated immediately as a word is reference by CPU an address space of 8k and a memory space of 4k. If we split each into groups of 1k words we obtain eight pages and four blocks as shown in four pages of address space may reside in main memory in any one of the four blocks. A virtual address has 13 bits. Since each page consists of 2^{10} = 1024 words, the high-order three bits of a virtual address will specify one of the eight pages and the low-order 10 bits give the line address within the page. A system with *n* pages and *m* blocks will be marked with block numbers and all others will be empty. A more efficient way to organize the page table would be to construct it with a number of words equal to the number of blocks in main memory. In this way the size of the memory is reduced and each location is fully utilized. The method can be implemented by means of an associative memory with each word in memory

An associative memory page table



containing a page number together with its corresponding block number. The page field in each word is compared with the page number in the virtual address. If a match occurs, the word is read from memory and its corresponding block. The page field in each word is compared with the page number in the virtual address. If a match occurs, the word is read from memory and its corresponding block number is

extracted. Each entry in the associative memory array consists of two fields. The first three bits specify a field for storing the page number. The last two bits constitute a field for storing the block number. The virtual address is placed in the argument register. The page number bits in the argument register are compared with all page numbers in the page field of the associative memory. If the page number is found, the 5-bit word is read out from memory. The corresponding block number, being in the same word, is transferred to the main memory address register. If no match occurs, a call to the operating system is generated to bring the required page from auxiliary memory.

Q.126 State the advantages of cache memory.

Ans.

Advantages –

(1) The average memory access time of a computer system can be improved by use of a cache.
(2) The fast access time of cache memory.
(3) Very little or no time wasted when searching for words in the cache.
(4) Cache memory is a fast & small memory.
(5) Program segment and data frequently needed by CPU are stored in cache memory and hence fast processing.

Q.127 Compare memory mapped I/O vs I/o mapped I/O.

Ans.

1) Memory mapped I/O use memory type instructions to access I/O data. It allows the computer to use the same instructions for either input-output transfers.
2) The load and store instructions are for reading and writing from memory can be used to input and output data from I/O registers.
3) Memory mapped I/O all instructions that refer to memory are also available for I/O.
4) In I/O mapped I/O configuration, the CPU has distinct input and output instructions, and each of these instructions is associated with the address of an interface register. When the CPU fetches and decodes the operation code of an input or output instruction, it places the address associated with the instruction in the common address lines and memory address values are not affected by interface assignment.

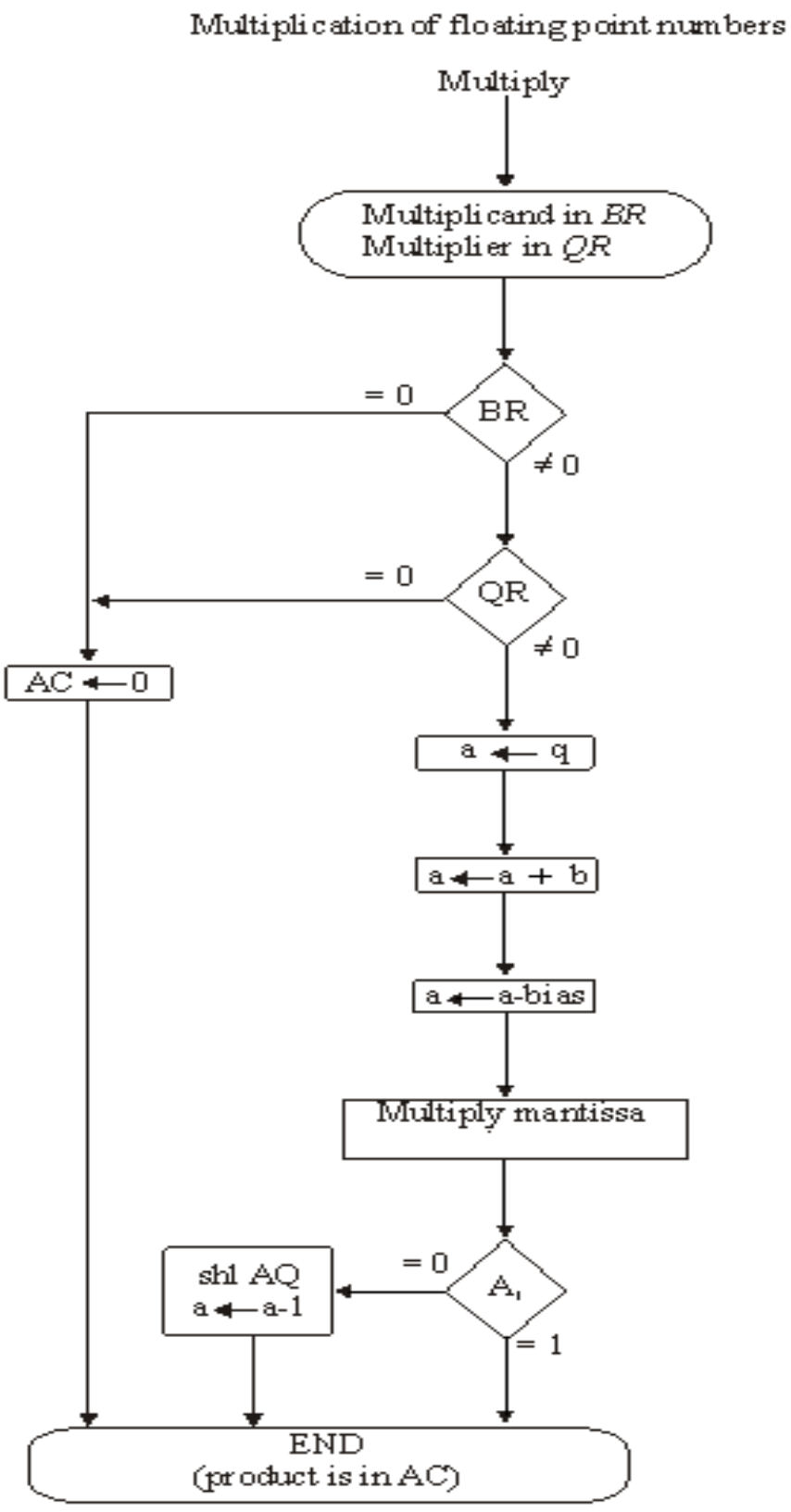
Q.128 Give the flow chart for multiplication of two floating-point numbers.

Ans.

Multiplication:-

The multiplication of two floating-point numbers requires that we multiply the mantissas and add the exponents. The multiplication algorithm can be subdivided into four parts.

1. Check for zeros.
2. Add the exponents.
3. Multiply the mantissas.
4. Normalize the product.


Multiplication of floating point numbers
Multiply
Multiplicand in BR
Multiplier in QR
BR
= 0
≠ 0
QR
= 0
≠ 0
AC ← 0
a ← q
a ← a + b
a ← a-bias
Multiply mantissa
A1
= 0
= 1
shl AQ
a ← a-1
END
(product is in AC)