Practice Questions Relate to Unit 1,2 and 3.

Computer Organisation and Design (Lovely Professional University)

TYPICAL QUESTIONS & ANSWERS

PART-I

OBJECTIVE TYPE QUESTIONS

Each Question carries 2 marks.

Choose the correct or best alternative in the following:

Q.1 In Reverse Polish notation, expression A*B+C*D is written as

(A) AB*CD*+ **(B)** A*BCD*+
(C) AB*CD+* **(D)** A*B*CD+

Ans: A

Q.2 SIMD represents an organization that _____.
(A) refers to a computer system capable of processing several programs at the same time.
(B) represents organization of single computer containing a control unit, processor unit and a memory unit.
(C) includes many processing units under the supervision of a common control unit
(D) none of the above.

Ans: C

Q.3 Floating point representation is used to store
(A) Boolean values **(B)** whole numbers
(C) real integers **(D)** integers

Ans: C

Q.4 Suppose that a bus has 16 data lines and requires 4 cycles of 250 nsecs each to transfer data. The bandwidth of this bus would be 2 Megabytes/sec. If the cycle time of the bus was reduced to 125 nsecs and the number of cycles required for transfer stayed the same what would the bandwidth of the bus?
(A) 1 Megabyte/sec **(B)** 4 Megabytes/sec
(C) 8 Megabytes/sec **(D)** 2 Megabytes/sec

Ans: D

Q.5 Assembly language

(A) uses alphabetic codes in place of binary numbers used in machine language
(B) is the easiest language to write programs
(C) need not be translated into machine language

(D) None of these

Ans: A

Q.6 In computers, subtraction is generally carried out by

(A) 9's complement **(B)** 10's complement
(C) 1's complement **(D)** 2's complement

Ans: D

Q.7 The amount of time required to read a block of data from a disk into memory is composed of seek time, rotational latency, and transfer time. Rotational latency refers to

(A) the time its takes for the platter to make a full rotation
(B) the time it takes for the read-write head to move into position over the appropriate track
(C) the time it takes for the platter to rotate the correct sector under the head
(D) none of the above

Ans: A

Q.8 What characteristic of RAM memory makes it not suitable for permanent storage?

(A) too slow **(B)** unreliable
(C) it is volatile **(D)** too bulky

Ans: C

Q.9 Computers use addressing mode techniques for _____.

(A) giving programming versatility to the user by providing facilities as pointers to memory counters for loop control
(B) to reduce no. of bits in the field of instruction
(C) specifying rules for modifying or interpreting address field of the instruction
(D) All the above

Ans: D

Q.10 The circuit used to store one bit of data is known as

(A) Register **(B)** Encoder
(C) Decoder **(D)** Flip Flop

Ans: D

Q. 11 $(2FAOC)_{16}$ is equivalent to

(A) $(195\ 084)_{10}$ **(B)** $(001011111010\ 0000\ 1100)_2$
(C) Both **(A)** and **(B)** **(D)** None of these
Ans: B

Q.12 The average time required to reach a storage location in memory and obtain its contents is called the

(A) seek time **(B)** turnaround time
(C) access time **(D)** transfer time

Ans: C

Q.13 Which of the following is not a weighted code?

(A) Decimal Number system **(B)** Excess 3-cod
(C) Binary number System **(D)** None of these

Ans: B

Q.14 The idea of cache memory is based

(A) on the property of locality of reference
(B) on the heuristic 90-10 rule
(C) on the fact that references generally tend to cluster
(D) all of the above

Ans: A

Q.15 _____ register keeps track of the instructions stored in program stored in memory.

(A) AR (Address Register) **(B)** XR (Index Register)
(C) PC (Program Counter) **(D)** AC (Accumulator)

Ans: C

Q.16 The addressing mode used in an instruction of the form ADD X Y, is

(A) Absolute **(B)** indirect
(C) index **(D)** none of these

Ans: C

Q.17 If memory access takes 20 ns with cache and 110 ns with out it, then the ratio (cache uses a 10 ns memory) is

(A) 93% **(B)** 90%
(C) 88% **(D)** 87%
Ans: B

Q.18 In a memory-mapped I/O system, which of the following will not be there?

(A) LDA **(B)** IN
(C) ADD **(D)** OUT

Ans: A

Q.19 In a vectored interrupt.
(A) the branch address is assigned to a fixed location in memory.
(B) the interrupting source supplies the branch information to the processor through an interrupt vector.
(C) the branch address is obtained from a register in the processor
(D) none of the above

Ans: B

Q.20 Von Neumann architecture is
(A) SISD **(B)** SIMD
(C) MIMD **(D)** MISD

Ans: A

Q. 21 The circuit used to store one bit of data is known as
(A) Encoder **(B)** OR gate
(C) Flip Flop **(D)** Decoder

Ans: C

Q.22 Cache memory acts between
(A) CPU and RAM **(B)** RAM and ROM
(C) CPU and Hard Disk **(D)** None of these

Ans: A

Q.23 Write Through technique is used in which memory for updating the data
(A) Virtual memory **(B)** Main memory
(C) Auxiliary memory **(D)** Cache memory

Ans: D

Q.24 Generally Dynamic RAM is used as main memory in a computer system as it
(A) Consumes less power **(B)** has higher speed
(C) has lower cell density **(D)** needs refreshing circuitary

Ans: B

Q.25 In signed-magnitude binary division, if the dividend is $(11100)_2$ and divisor is $(10011)_2$ then the result is

(A) $(00100)_2$ **(B)** $(10100)_2$
(C) $(11001)_2$ **(D)** $(01100)_2$

Ans: B

Q.26 Virtual memory consists of
(A) Static RAM **(B)** Dynamic RAM
(C) Magnetic memory **(D)** None of these

Ans: A

Q.27 In a program using subroutine call instruction, it is necessary
(A) initialise program counter
(B) Clear the accumulator
(C) Reset the microprocessor
(D) Clear the instruction register

Ans: D

Q.28 A Stack-organised Computer uses instruction of
(A) Indirect addressing **(B)** Two-addressing
(C) Zero addressing **(D)** Index addressing

Ans: C

Q.29 If the main memory is of 8K bytes and the cache memory is of 2K words. It uses associative mapping. Then each word of cache memory shall be
(A) 11 bits **(B)** 21 bits
(C) 16 bits **(D)** 20 bits

Ans: C

Q.30 A-Flip Flop can be converted into T-Flip Flop by using additional logic circuit
(A) $D = T \bullet Q_n$ **(B)** $D = \overline{T}$
(C) $D = T \cdot Q_n$ **(D)** $D = T \oplus Q_n$

Ans: D

Q.31 Logic X-OR operation of $(4ACO)_H$ & $(B53F)_H$ results
(A) AACB **(B)** 0000
(C) FFFF **(D)** ABCD

Ans: C

Q.32 When CPU is executing a Program that is part of the Operating System, it is said to be in
(A) Interrupt mode
(B) System mode
(C) Half mode
(D) Simplex mode

Ans: B

Q.33 An n-bit microprocessor has
(A) n-bit program counter
(B) n-bit address register
(C) n-bit ALU
(D) n-bit instruction register

Ans: D

Q.34 Cache memory works on the principle of
(A) Locality of data
(B) Locality of memory
(C) Locality of reference
(D) Locality of reference & memory

Ans: C

Q.35 The main memory in a Personal Computer (PC) is made of
(A) cache memory.
(B) static RAM
(C) Dynamic Ram
(D) both **(A)** and **(B)**.

Ans: D

Q.36 In computers, subtraction is carried out generally by
(A) 1's complement method
(B) 2's complement method
(C) signed magnitude method
(D) BCD subtraction method

Ans: B

Q.37 PSW is saved in stack when there is a
(A) interrupt recognised
(B) execution of RST instruction
(C) Execution of CALL instruction
(D) All of these

Ans: A

Q.38 The multiplicand register & multiplier register of a hardware circuit implementing booth's algorithm have (11101) & (1100). The result shall be
(A) $(812)_{10}$
(B) $(-12)_{10}$
(C) $(12)_{10}$
(D) $(-812)_{10}$

Ans: A

Q.39 The circuit converting binary data in to decimal is
(A) Encoder **(B)** Multiplexer
(C) Decoder **(D)** Code converter

Ans: D

Q.40 A three input NOR gate gives logic high output only when
(A) one input is high **(B)** one input is low
(C) two input are low **(D)** all input are high

Ans: D

Q.41 n bits in operation code imply that there are _____ possible distinct operators
(A) 2n **(B)** 2^n
(C) n/2 **(D)** n^2

Ans: B

Q.42 _____ register keeps tracks of the instructions stored in program stored in memory.
(A) AR (Address Register) **(B)** XR (Index Register)
(C) PC (Program Counter) **(D)** AC (Accumulator)

Ans: C

Q.43 Memory unit accessed by content is called
(A) Read only memory **(B)** Programmable Memory
(C) Virtual Memory **(D)** Associative Memory

Ans: D

Q.44 'Aging registers' are
(A) Counters which indicate how long ago their associated pages have been referenced.
(B) Registers which keep track of when the program was last accessed.
(C) Counters to keep track of last accessed instruction.
(D) Counters to keep track of the latest data structures referred.
Ans: A

Q.45 The instruction 'ORG O' is a
(A) Machine Instruction. **(B)** Pseudo instruction.
(C) High level instruction. **(D)** Memory instruction.
Ans: B

Q.46 Translation from symbolic program into Binary is done in
(A) Two passes. **(B)** Directly
(C) Three passes. **(D)** Four passes.

Ans: A

Q.47 A floating point number that has a O in the MSB of mantissa is said to have
(A) Overflow **(B)** Underflow
(C) Important number **(D)** Undefined

Ans: B

Q.48 The BSA instruction is
(A) Branch and store accumulator
(B) Branch and save return address
(C) Branch and shift address
(D) Branch and show accumulator

Ans: B

Q.49 State whether True or False.
(i) Arithmetic operations with fixed point numbers take longer time for execution as compared to with floating point numbers.
Ans: True.

(ii) An arithmetic shift left multiplies a signed binary number by 2.
Ans: False.

Q.50 Logic gates with a set of input and outputs is arrangement of
(A) Combinational circuit **(B)** Logic circuit
(C) Design circuits **(D)** Register

Ans: A

Q.51 MIMD stands for
(A) Multiple instruction multiple data
(B) Multiple instruction memory data
(C) Memory instruction multiple data
(D) Multiple information memory data

Ans: A

Q.52 A k-bit field can specify any one of
(A) 3^k registers **(B)** 2^k registers
(C) K^2 registers **(D)** K^3 registers

Ans: B

Q.53 The time interval between adjacent bits is called the
(A) Word-time **(B)** Bit-time
(C) Turn around time **(D)** Slice time

Ans: B

Q.54 A group of bits that tell the computer to perform a specific operation is known as
(A) Instruction code **(B)** Micro-operation
(C) Accumulator **(D)** Register

Ans: A

Q.55 The load instruction is mostly used to designate a transfer from memory to a processor register known as
(A) Accumulator **(B)** Instruction Register
(C) Program counter **(D)** Memory address Register

Ans: A

Q.56 The communication between the components in a microcomputer takes place via the address and
(A) I/O bus **(B)** Data bus
(C) Address bus **(D)** Control lines

Ans: B

Q.57 An instruction pipeline can be implemented by means of
(A) LIFO buffer **(B)** FIFO buffer
(C) Stack **(D)** None of the above

Ans: B

Q.58 Data input command is just the opposite of a
(A) Test command **(B)** Control command
(C) Data output **(D)** Data channel

Ans: C

Q.59 A microprogram sequencer
(A) generates the address of next micro instruction to be executed.
(B) generates the control signals to execute a microinstruction.
(C) sequentially averages all microinstructions in the control memory.
(D) enables the efficient handling of a micro program subroutine.

Ans: A

Q.60 A binary digit is called a
(A) Bit **(B)** Byte
(C) Number **(D)** Character

Ans: A

Q.61 A flip-flop is a binary cell capable of storing information of
(A) One bit **(B)** Byte
(C) Zero bit **(D)** Eight bit

Ans: A

Q.62 The operation executed on data stored in registers is called
(A) Macro-operation **(B)** Micro-operation
(C) Bit-operation **(D)** Byte-operation

Ans: B

Q.63 MRI indicates
(A) Memory Reference Information.
(B) Memory Reference Instruction.
(C) Memory Registers Instruction.
(D) Memory Register information

Ans: B

Q.64 Self-contained sequence of instructions that performs a given computational task is called
(A) Function **(B)** Procedure
(C) Subroutine **(D)** Routine

Ans: A

Q.65 Microinstructions are stored in control memory groups, with each group specifying a
(A) Routine **(B)** Subroutine
(C) Vector **(D)** Address

Ans: A

Q.66 An interface that provides a method for transferring binary information between internal storage and external devices is called
(A) I/O interface **(B)** Input interface
(C) Output interface **(D)** I/O bus

Ans: A

Q.67 Status bit is also called
(A) Binary bit **(B)** Flag bit
(C) Signed bit **(D)** Unsigned bit

Ans: B

Q.68 An address in main memory is called

(A) Physical address **(B)** Logical address
(C) Memory address **(D)** Word address

Ans: A

Q.69 If the value V(x) of the target operand is contained in the address field itself, the addressing mode is
(A) immediate. **(B)** direct.
(C) indirect. **(D)** implied.

Ans: B

Q.70 $(-27)_{10}$ can be represented in a signed magnitude format and in a 1's complement format as
(A) 111011 & 100100 **(B)** 100100 & 111011
(C) 011011 & 100100 **(D)** 100100 & 011011

Ans: A

Q.71 The instructions which copy information from one location to another either in the processor's internal register set or in the external main memory are called
(A) Data transfer instructions. **(B)** Program control instructions.
(C) Input-output instructions. **(D)** Logical instructions.

Ans: A

Q.72 A device/circuit that goes through a predefined sequence of states upon the application of input pulses is called
(A) register **(B)** flip-flop
(C) transistor. **(D)** counter.

Ans: D

Q.73 The performance of cache memory is frequently measured in terms of a quantity called
(A) Miss ratio. **(B)** Hit ratio.
(C) Latency ratio. **(D)** Read ratio.
Ans: C

Q.74 The information available in a state table may be represented graphically in a
(A) simple diagram. **(B)** state diagram.
(C) complex diagram. **(D)** data flow diagram.
Ans: B

Q.75 Content of the program counter is added to the address part of the instruction in order to obtain the effective address is called.
(A) relative address mode. **(B)** index addressing mode.
(C) register mode. **(D)** implied mode.

Ans: A

Q.76 An interface that provides I/O transfer of data directly to and form the memory unit and peripheral is termed as

(A) DDA. **(B)** Serial interface.
(C) BR. **(D)** DMA.

Ans: D

Q.77 The 2s compliment form (Use 6 bit word) of the number 1010 is

(A) 111100. **(B)** 110110.
(C) 110111. **(D)** 1011.

Ans: B

Q.78 A register capable of shifting its binary information either to the right or the left is called a

(A) parallel register. **(B)** serial register.
(C) shift register. **(D)** storage register.

Ans: C

Q.79 What is the content of Stack Pointer (SP)?

(A) Address of the current instruction
(B) Address of the next instruction
(C) Address of the top element of the stack
(D) Size of the stack.

Ans: C

Q.80 Which of the following interrupt is non maskable

(A) INTR. **(B)** RST 7.5.
(C) RST 6.5. **(D)** TRAP.

Ans: D

Q.81 Which of the following is a main memory

(A) Secondary memory. **(B)** Auxiliary memory.
(C) Cache memory. **(D)** Virtual memory.

Ans: C

Q.82 Which of the following are not a machine instructions

(A) MOV. **(B)** ORG.
(C) END. **(D) (B)** & **(C)**.

Ans: D

Q.83 In Assembly language programming, minimum number of operands required for an instruction is/are

(A) Zero. **(B)** One.
(C) Two. **(D)** Both **(B)** & **(C)**.

Ans: A

Q.84 The maximum addressing capacity of a micro processor which uses 16 bit database & 32 bit address base is

(A) 64 K. **(B)** 4 GB.
(C) both **(A)** & **(B)**. **(D)** None of these.

Ans: B

Q.85 The memory unit that communicates directly with the CPU is called the

(A) main memory **(B)** Secondary memory
(C) shared memory **(D)** auxiliary memory.

Ans: A

Q.86 The average time required to reach a storage location in memory and obtain its contents is called

(A) Latency time. **(B)** Access time.
(C) Turnaround time. **(D)** Response time.

Ans: B

State True or False

Q.87 A byte is a group of 16 bits.

Ans: False

Q.88 A nibble is a group of 16 bits.

Ans: False

Q.89 When a word is to be written in an associative memory, address has got to be given.

Ans: False

Q.90 When two equal numbers are subtracted, the result would be _____and not_____.

Ans: +ZERO, -ZERO.

Q.91 A _____development system and an _____are essential tools for writing large assembly language programs.

Ans: Microprocessor, assembler

Q.92 In an operation performed by the ALU, carry bit is set to 1 if the end carry C_8 is _____. It is cleared to 0 (zero) if the carry is _____ _____.

Ans: One, zero

Choose the correct alternative

Q.93 A successive A/D converter is

(A) a high-speed converter. **(B)** a low speed converter.
(C) a medium speed converter. **(D)** none of these.

Ans: C

Q.94 When necessary, the results are transferred from the CPU to main memory by

(A) I/O devices. **(B)** CPU.
(C) shift registers. **(D)** none of these.

Ans: B

Q.95 The gray code equivalent of $(1011)_2$ is

(A) 1101. **(B)** 1010.
(C) 1110. **(D)** 1111.

Ans: C

Q.96 A combinational logic circuit which sends data coming from a single source to two or more separate destinations is

(A) Decoder. **(B)** Encoder.
(C) Multiplexer. **(D)** Demultiplexer.

Ans: D

Q.97 In which addressing mode the operand is given explicitly in the instruction

(A) Absolute. **(B)** Immediate.
(C) Indirect. **(D)** Direct.

Ans: B

Q.98 A stack organized computer has

(A) Three-address Instruction. **(B)** Two-address Instruction.
(C) One-address Instruction. **(D)** Zero-address Instruction.

Ans: D

Q.99 A Program Counter contains a number 825 and address part of the instruction contains the number 24. The effective address in the relative address mode, when an instruction is read from the memory is

(A) 849. **(B)** 850.
(C) 801. **(D)** 802.

Ans: B

Q.100 A system program that translates and executes an instruction simultaneously is

(A) Compiler. **(B)** Interpreter.
(C) Assembler. **(D)** Operating system.

Ans: C

Q.101 The cache memory of 1K words uses direct mapping with a block size of 4 words. How many blocks can the cache accommodate.

(A) 256 words. **(B)** 512 words.
(C) 1024 words. **(D)** 128 words.

Ans: A

Q.102 A page fault

(A) Occurs when there is an error in a specific page.
(B) Occurs when a program accesses a page of main memory.
(C) Occurs when a program accesses a page not currently in main memory.
(D) Occurs when a program accesses a page belonging to another program.

Ans: C

DESCRIPTIVES

Q.1 Simplify the Boolean function F together with don't care conditions d in sum-of-products and product-of-sum forms

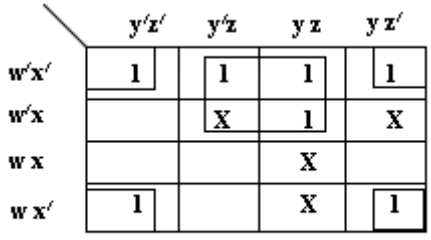
$$F(w,x,y,z) = \Sigma(0,1,2,3,7,8,10)$$
$$d(w,x,y,z) = \Sigma(5,6,11,15)$$ **(6)**

Ans:

Given function

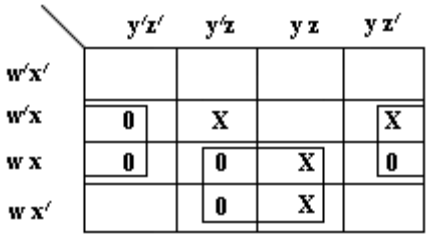
$F(w,x,y,z) = \Sigma(0,1,2,3,7,8,10)$
$d(w,x,y,z) = \Sigma(5,6,11,15)$

Sum of products



$F = w'z + x'z'$

Product of sums



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

Q.2 Represent the condition control statement by two register transfer statements with control functions.

If (P=1) then $(R_1 \leftarrow R_2)$ else if (Q=1) then $(R_1 \leftarrow R_3)$ **(6)**

Ans:

The two register transfer statements are

P: $R_1 \leftarrow R_2$

P′Q: $R_1 \leftarrow R_3$

Q.3 Explain the use of subroutine with the help of suitable example. **(4)**

Ans:
A subroutine is a self-contained sequence of instructions that performs a given computational task. During the program execution, a subroutine can be called many times to perform its operations. Each time a subroutine is called, a branch is executed to start executing its set of instruction and a branch is made back to the main program when the subroutine has been executed. The instruction that causes the transfer of program control to a subroutine is known by different names. Some common names are called subroutine, jump to subroutine or branch and save address.

The call subroutine instruction consists of an operation code and the address that specifies the beginning of subroutine. The instruction is executed by performing two operations (i) the address of the next instruction i.e. the return address in the program counter is stored at some temporary location, (ii) the control is transferred at the beginning of the subroutine. The last instruction of every subroutine, called as return from subroutine, transfers the return address stored in temporary location into the program counter. This return address helps in a transfer of program control to the instruction whose address was stored in temporary location. This temporary location can be the first memory location of the subroutine, or some fixed memory location or a processor register or can be a memory stack. But most efficient way is to store the return address in a memory stack.

Q.4 Justify the statement "Stack computer consists of an operation code only with no address field". **(5)**

Ans:
Stack-oriented machines do not contain any accumulator or general-purpose registers. Computers with stack organization have PUSH and POP instructions which requires an address field. Thus the instruction

PUSH X TOP ← M[X]

will push the data at address X to the top of the stack. The SP is automatically updated. The operation instruction does not contain any address field because the operation is performed on two top most operands of the stack. For example,

ADD

The instruction ADD consists of only operation code with no address field. This instruction pops the top two operands from the stack, add the numbers and then PUSH the result into the stack.

Q.5 Given that following are all the instructions related to AR, determine the control functions load(LD), clear(CLR) and increment(INC) for AR.

$R'T_0 : AR \leftarrow PC$

$R'T_2 : AR \leftarrow IR(0-11)$

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

$RT_0 : AR \leftarrow 0$

$D_5 T_4 : AR \leftarrow AR + 1$ **(6)**

Ans:

Given instructions are

$R'T_0 : AR \leftarrow PC$

$R'T_2 : AR \leftarrow IR(0-11)$

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

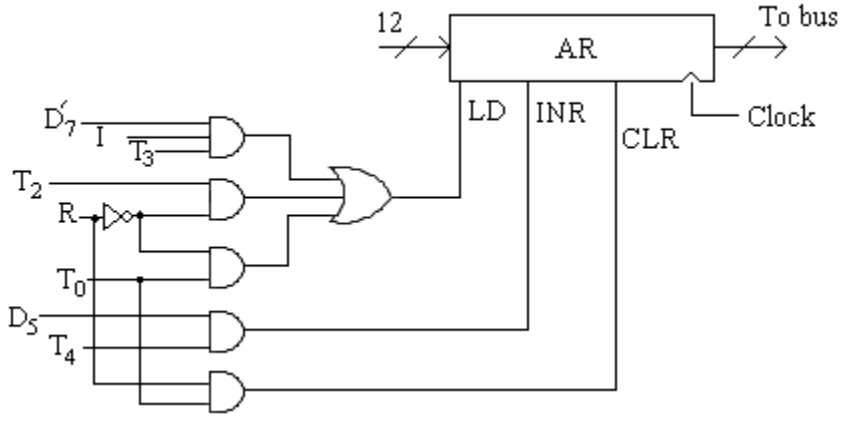
$RT_0 : AR \leftarrow 0$

$D_5 T_4 : AR \leftarrow AR + 1$

The first three statements specify transfer of information from a register or memory to AR. The content of the source register or memory is placed on the bus and the content of the bus is transferred into AR by enabling its LD control input. The fourth statement clears AR to 0. The last statement increments AR by 1. The control functions can be combined into three Boolean expressions as follows:

$$LD(AR) = R'T_0 + R'T_2 + D_7'IT_3$$
$$CLR(AR) = RT_0$$
$$INR(AR) = D_5T_4$$

Where LD(AR) is the load input of AR, CLR(AR) is the clear input of AR, and INR(AR) is the increment input of AR. The control gate logic associated with AR is shown in figure below.



Control gates associated with AR

Q.6 Explain indirect address mode and how the effective address is calculated in this case. **(5)**

Ans:

In indirect addressing mode the address field of the instruction gives the address where the operand is stored in the memory. Control fetches the instruction from memory and uses its address part to access memory again to read the effective address.

Opcode	Memory Address = Operand Address

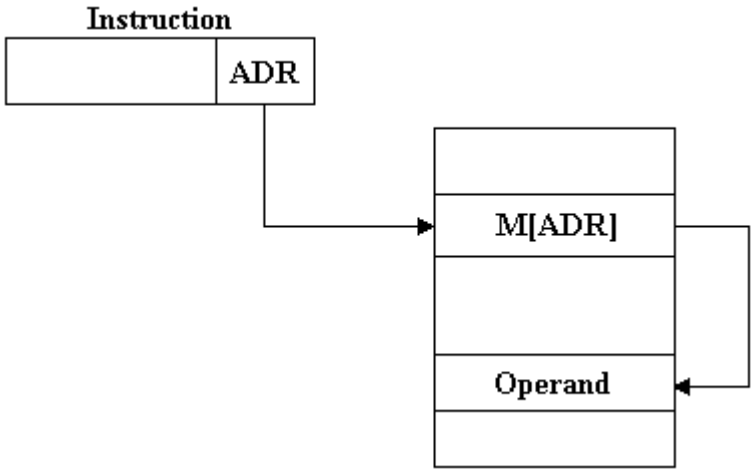
Indirect Addressing Mode

For Example,

Efffective address = M[ADR]

LD@ADR

AC ←M[M[ADR]]



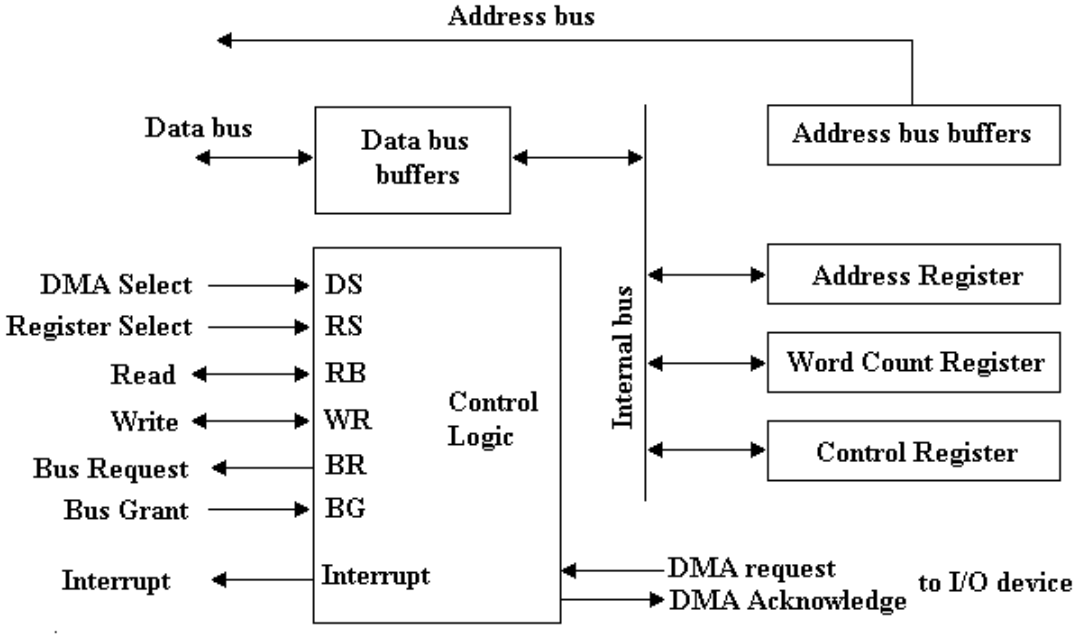
Q.7 Why is read and write control lines in a DMA controller bidirectional? Under what condition and for what purpose are they used as inputs? **(4)**

Ans:

The DMA controller consists of circuits of an interface to communicate with the CPU and I/O device. It also has an address register, a word count register, a set of address lines. The address register and address lines are used for direct communication with the memory. The word count register specifies the number of words that must be transferred. The figure shows the block diagram of 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. 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 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. The address register contains an address to specify the desired location in memory and the address bits goes into the address bus through address bus buffers. The address register is incremented after each word that is transferred from/to memory. The word count register holds the number of words to be transferred to memory. This register is decremented by one after each word transferred from/to memory and internally tested for zero. The control register specifies the mode of transfer. All registers in DMA appears to the CPU as I/O interface registers and hence the CPU can read from or write into the DMA registers.



Q.8 Explain the different types of mapping procedures in the organization of cache memory with diagram. **(12)**

Ans:

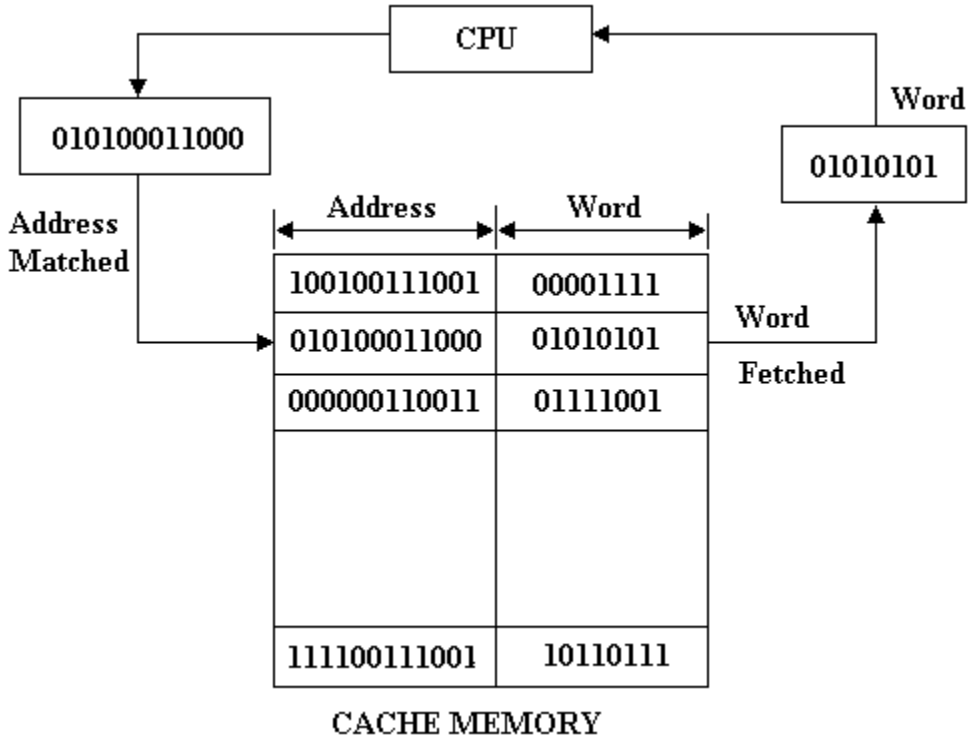
Associative Mapping

In case of associative mapping, the contents of cache memory are not associated with any address. Data stored in the cache memory are not accessed by specifying any address. Instead, data or part of data is searched by matching with the contents. In associative mapping method, both the word and the address of the word (in the main memory) are stored in the cache as shown in Figure. The address bits, sent by the CPU to search, are matched with addresses stored in the cache memory. If any address is matched, the corresponding word is fetched from the cache and sent to the CPU.

If not match is found in cache memory, the word is searched in the main memory. The word along with address is then copied from main memory into cache. If the cache is full, then the existing word along with its address must be removed to make room for the new word.

Associative mapping has the advantage that it is a very fast access method, but it has the disadvantage that it is very expensive and complicated because of complex logical circuits that ate required to implement data searching by content and not by address.

Due to the high cost associated with logic circuits required to implement associative mapping, other methods are used in which data is cache memory are accessed by address, like direct mapping and set associative mapping.



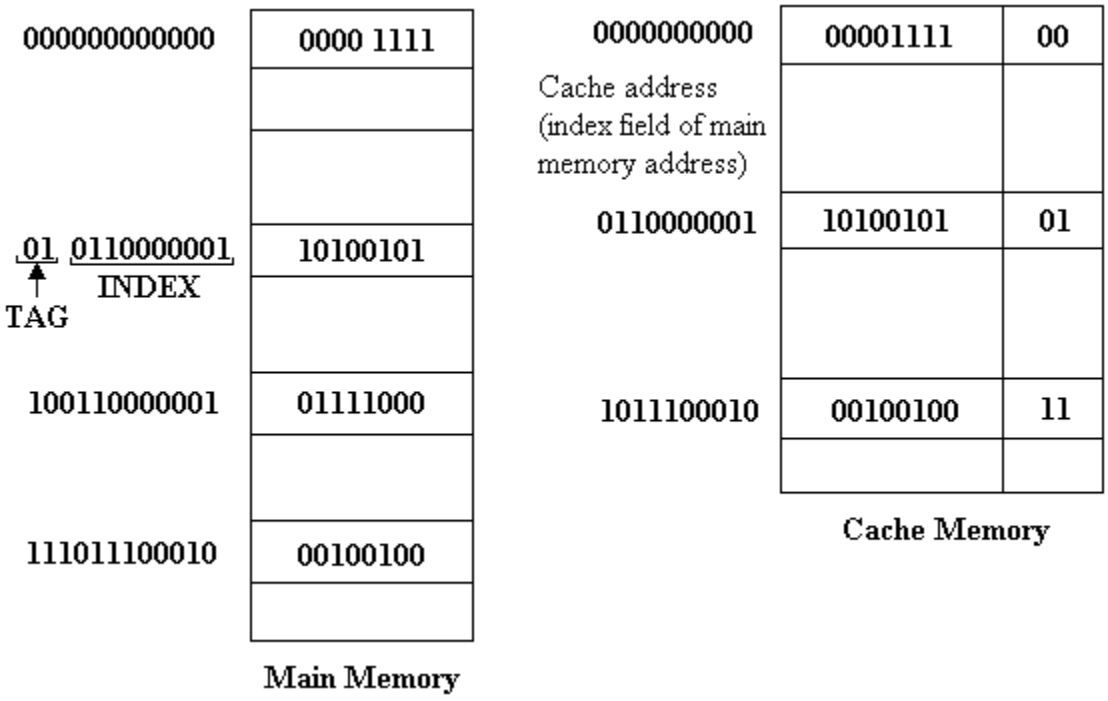
Direct Mapping

Suppose a computer has 4K main memory i.e. 4x1024 bytes and 1K cache memory. To address a word in the main memory, a 12 bit ($4K = 2^{12}$) address is required. Similarly, to address a word in the cache memory a 10-bit ($1K = 2^{10}$) address is required. Thus, a cache memory needs 10 bits to address a word and main memory requires 12 bits to address a word. In direct mapping method, the 12 bit address sent by CPU is divided into two parts called tag field and index field. The index field has the number of bits equal to the number of bits required to address a word in cache.

Thus, if a computer has main memory of capacity 2^m and cache memory of 2^n, then the address bits will be divided into n bits index field and (m-n) bits of tag field. The

tag field for above computer system will be of 2-bits and index field will have 10 bits.

In a direct mapping method, the cache memory stored the word as well as the tag field. The words will be stored at that location in the cache which is represented by the index fields of their addresses as shown in figure. When an address is sent by the CPU, the index part of the address is used to get a memory location in the cache. If the tag stored at that location matches the tag field of the requested address, the word is fetched. Else, if tag does not match, the word is searched in the main memory. When a word needs to be moved into cache memory from the main memory, its address I the main memory is divided into index and tag fields. The disadvantage of direct mapping is that the hit ratio can drop considerably if two or more words whose addresses have same index but different tags are accessed repeatedly.



Set Associative mapping

The disadvantage of direct mapping is that two words with same index but different tag cannot be stored into cache at the same time. As an improvement to this disadvantage of direct mapping, a third type of cache organization called set-associative mapping is used. In this mapping process, each word of a cache can store two or more words of main memory under the same index address. Each data word is stored along with its tag and the number of tag data pair in one word of cache is said to form a set. An example of set associative cache organization with set size of two is shown in figure.

INDEX	TAG	DATA	TAG	DATA
0000110010	00	10101010	01	10010110
0000111011	01	11000011	10	11000011

The words stored at address 000000110010 and 010000110010 of main memory are stored in cache memory at index address 0000110010. Similarly, the words stored at address 010000111011 and 100000111011 of main memory are stored in cache memory at index address 0000111011. When CPU generates a memory request, the word is searched into cache with the help of index addresses.

Q.9 What does the instruction field in the assembly language program specify? **(3)**

Ans:
The instruction field specifies a machine instruction or pseudo instruction. The instruction field in an assembly program may specify one of the following:
(i) A memory-reference instruction (MRI)
(ii) A register-reference or input-output instruction (non-MRI)
(iii) A pseudo instruction with or without an operand

A memory-reference instruction occupies two or three symbols separated by spaces. The first must be three letter symbol defining an MRI operation code such as AND, ADD, LDA, STA etc. the second is a symbolic address. The third symbol, which may or may not be present, is the letter I. If I is missing, the line denotes a direct address instruction. The presence of the symbol I denotes an indirect address instruction.

A non-MRI is defined as an instruction that does not have an address part. A non-MRI is recognized in the instruction field of a program by any one of the three-letter symbols for the register-reference and input-output instructions. A symbol address in the instruction field specifies the memory location of an operand. This location must be defined somewhere in the program by appearing again as a label in the first column.

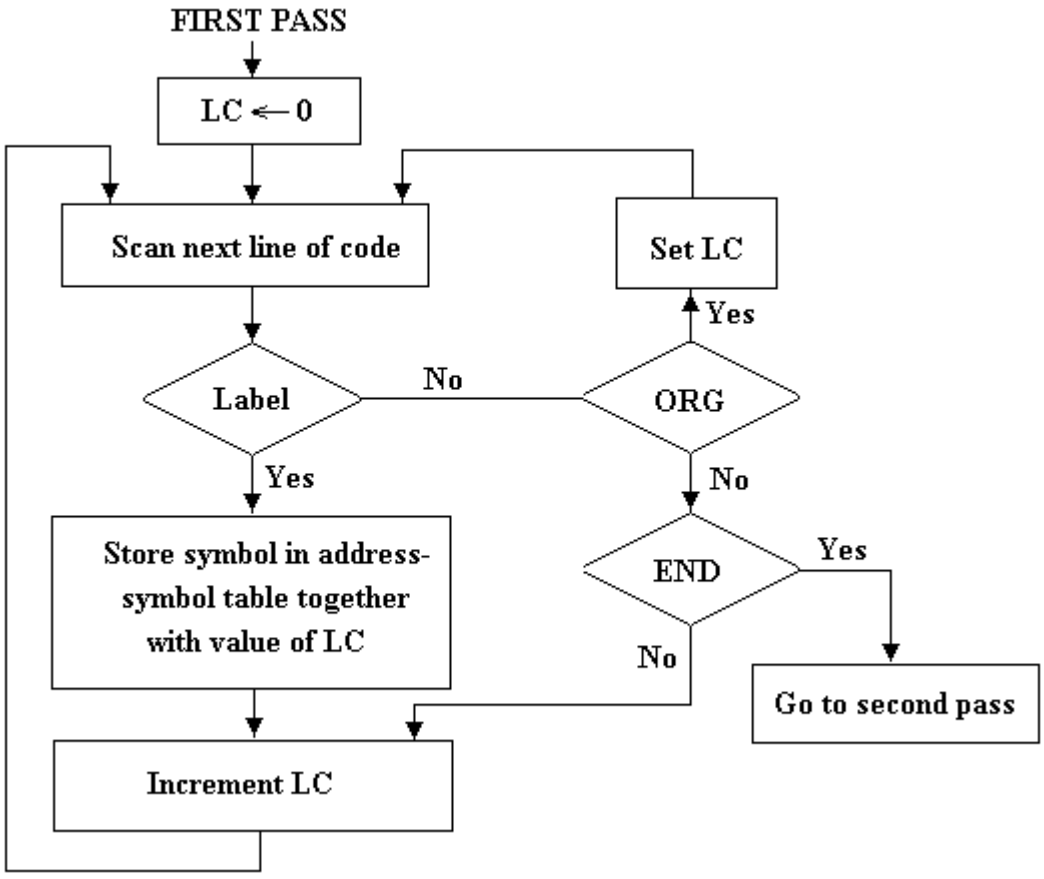
A pseudo instruction is not a machine instruction but rather an instruction to the assembler giving information about some phase of the translation. The ORG (origin) pseudo instruction informs the assembler that the instruction or operand is to be placed in the memory location specified by the number next to ORG. the END

symbol is placed at the end of the program to inform the assembler that the program is terminated.

Q. 10 Explain the working of first pass of assembler using flow chart. **(5)**

Ans:

The tasks performed by the assembler during first pass are explained in the flowchart. LC is initially set to 0. A line of symbolic code is analysed to determine if it has a label. If the line of code has no label, the assembler checks the symbol in the instruction field. If it contains an ORG pseudo instruction, the assembler sets LC to the number that follows ORG and goes back to process the next line. If the line has an END pseudo instruction, the assembler terminates the first pass and goes to the second pass. If the line of code contains a label, it is stored I the address symbol table together with its binary equivalent number specified by the content of LC. Nothing is stored in the table if no label is encountered. LC is then incremented by 1 and a new line of code is processed.



Flowchart for first pass of assembler

Q 11 Write a program in assembly language to subtract two integer numbers. **(8)**

Ans:

Assembly Language Program to Subtract Two Numbers

```
        ORG 100      /Origin of program is location 100
        LDA SUB      /Load subtrahend to AC
        CMA          /Complement AC
        INC          /Increment AC
        ADD MIN      /Add minuend to AC
        STA DIF      /Store difference
        HLT          /Halt computer
MIN,    DEC 83       /Minuend
SUB,    DEC -23      /Subtrahend
DIF,    HEX 0        /Difference stored here
        END          /End of symbolic program
```

Q.12 Explain the difference between vectored and non-vectored interrupt. Explain stating examples of each. **(8)**

Ans:
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 transfers to 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 as *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 to the computer. This information is called the interrupt vector. This can be first address of the I/O service routine or this address is an address that points to a location in memory where the beginning address of the I/O service routine is stored.

Q.13 Explain the various input output modes of data transfer. **(8)**

Ans:
There are three different ways by which the data can be transferred from input/output devices to the memory. They are
(i) Programmed I/O
(ii) Interrupt Initiated I/O
(iii) Direct Memory Access (DMA)

Programmed I/O
In this mode of data transfer the input/output instructions are written in the form of computer program. The instruction written in the program basically initiates the data transfer to and from a CPU register and Input/Output peripheral devices.

Programmed input/output requires that all input/output operations executed under the direct control of the CPU i.e. every data transfer operation involving an input/output devices requires the execution of an instruction by the CPU. Thus, in programmed input/output mode of transfer, the CPU stays in program loop until the input/output indicates that it is ready for data transfer. This is time-consuming process and keeps the processor busy. This mode of transfer is useful in small, low speed systems where hardware costs must be minimized.

Interrupt Initiated I/O

In interrupt-initiated I/O mode the transfer the CPU concentrate on some other program. This method uses interrupt signal to inform the CPU that the data are available from the peripheral device and the input-output flag is set to 1. When the flat is set the CPU deactivates its task to take care of the input-output transfer. After the transfer has been completed, the CPU returns to continue the previous program.

When the interrupt signal is generated, the CPU responds to it by storing the return address from program counter into the memory stack and the control branches to a service routine that processes the required the input/output transfer. There are two methods to choose the branch address of the service routine. One is called vectored interrupt and the other is called non-vectored interrupt. In vectored interrupt the source that interrupts, gives the branch information to the computer and this information is known as interrupt vector. In non-vectored interrupt, the branch address is assigned to a fixed location in memory.

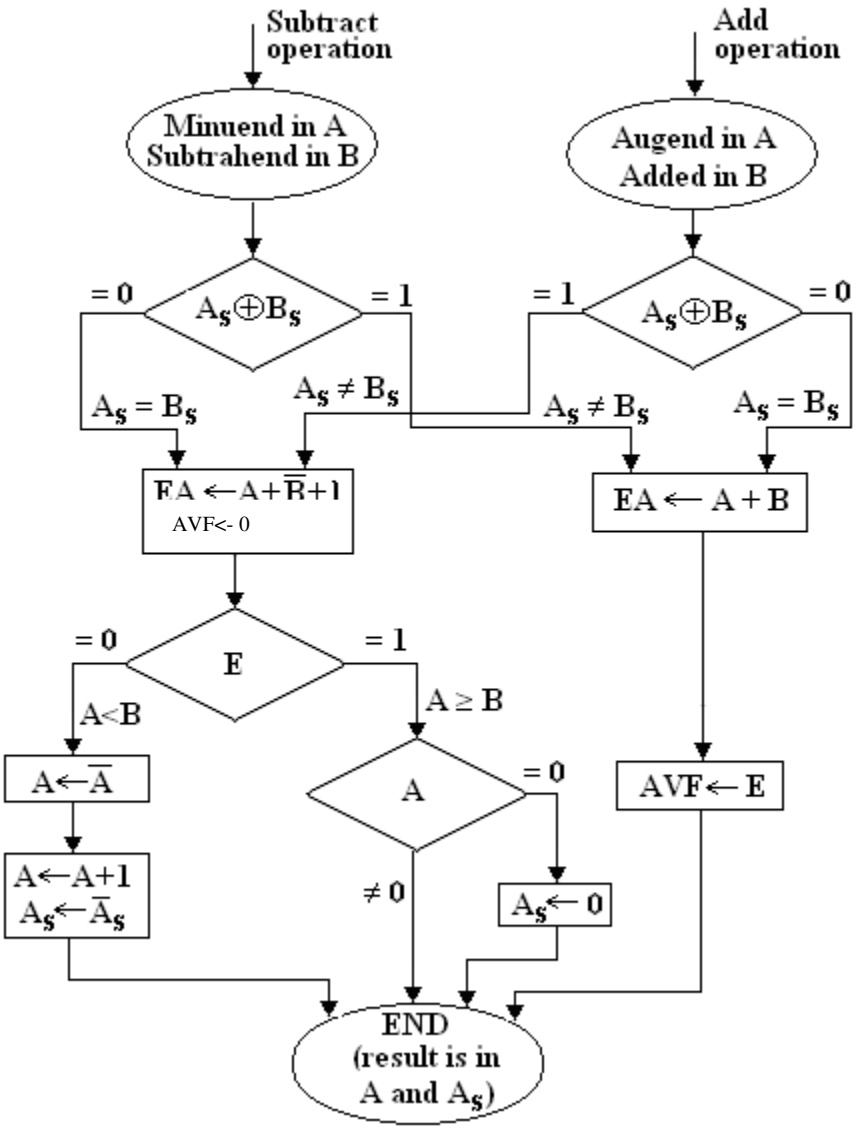
Direct Memory Access (DMA)

In DMA mode of transfer, the interface transfers data into and out of the memory unit through memory bus. The CPU initiates the transfer. It supplies the starting address and number of words to be transferred to the interface unit and then busy with other tasks. When the transfer is made, the DMA disables the bus request line. The transfer of data can be made in several ways. A block sequence consisting of a number of memory words can be transferred into a continuous burst. Alternative way

allows transferring one data word at a time, after which it must return control of the buses to the CPU.

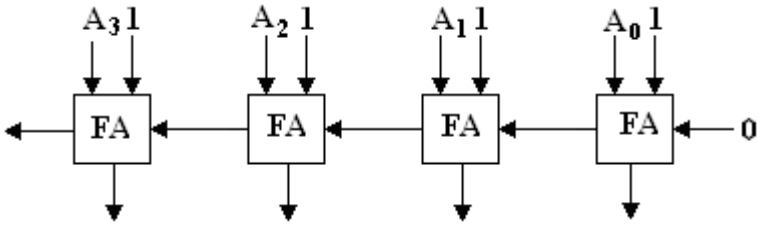
Q.14 Draw a flowchart for adding and subtracting two fixed point binary numbers where negative numbers are signed 1's complement presentation. **(8)**

Ans:



Q.15 Design a 4-bit combinational circuit decrementer using four full adders. **(8)**

Ans:
A – 1 = A + 2's complement of 1



Q.16 Differentiate between virtual and cache memory **(6)**

Ans:

Cache Memory is defined as a very high speed memory that is used in computer system to compensate the seed differential between the main memory access time and processor logic. A very high speed memory called a cache is used to increase the speed of processing by making current programs and data available to the CPU at a rapid rate. It is place between the CPU and the main memory. The cache memory access time is less than the access time of the main memory by a factor of 5 to 10. The cache is used for storing program segments currently being executed in the CPU and the data frequently used in the present calculations. By making programs and data available at a rapid rate, it is possible to increase the performance rate of a computer.

Virtual Memory is a concept used in some large computer system that permit the user to construct programs as though a large space were available, equal to the totality of auxiliary memory. Each address that is referenced by the CPU goes through an address mapping from the so-called virtual address to a physical address in main memory.

A virtual memory system provides a mechanism for translating program generated addresses into correct main memory locations. This is done dynamically, while programs are being executed in the CPU.

Q.17 Discuss Flynn's classification of Computer **(4)**

Ans:

The classification based on the multiplicity of instruction streams and data streams in a computer system is known a Flynn's Classification. Parallel processing can be classified in variety of ways. It can be considered from the internal organization of the processors, from the interconnection structure between processors, or from the flow of information through the system. One classification was introduced by M.J. Flynn, as "How do instructions and data flow in the system?" and this classification is known as Flynn's Classification. Flynn's classified parallel computers into four categories based on how instructions process data. These categories are:

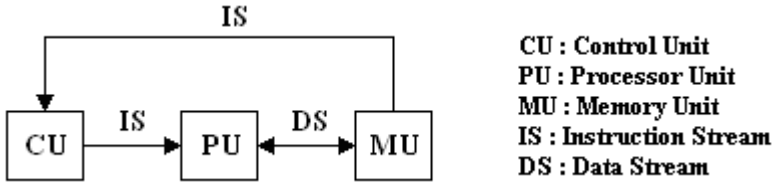
(i) Single Instruction Stream, Single Data Stream (SISD) Computer.
(ii) Single Instruction Stream, Multiple Data Stream (SIMD) Computer.
(iii) Multiple Instruction Stream, Single Data Stream (MISD) Computer.
(iv) Multiple Instruction Stream, Multiple Data Stream (MIMD) Computer.

The normal operation of a computer is to fetch instructions from memory and execute them in a processor. The sequence of instructions read from the memory constitutes an instruction stream. The operations performed on the data in the processor constitute a data stream. Parallel processing may occur in the instruction stream, in the data stream or in both.

Single Instruction Stream, Single Data Stream (SISD)

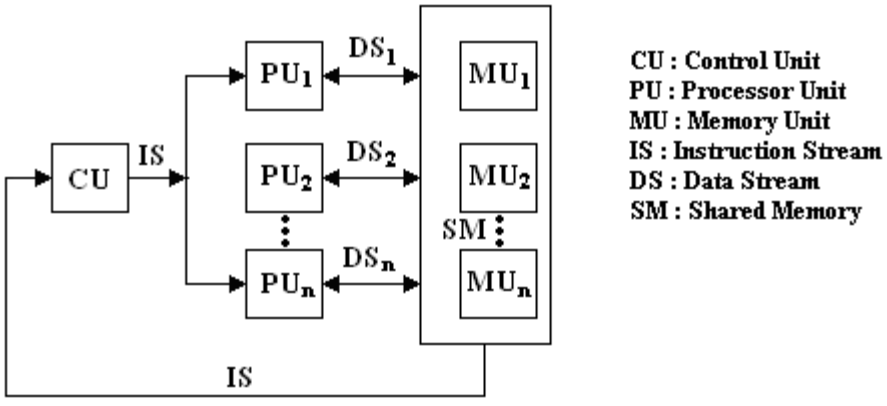
A computer with a single processor is called a Single Instruction Stream, Single Data Stream (SISD) Computer. It represents the organization of a single computer containing a control unit, a processor unit, and a memory unit. Instructions are executed sequentially and the system may or may not have internal parallel processing. Parallel processing may be achieved by means of a pipeline processing.

In such a computer a single stream of instructions and a single stream of data are accessed by the processing elements from the main memory, processed and the results are stored back in the main memory. SISD computer organization is shown in figure below.



Single Instruction Stream, Multiple Data Stream (SIMD)

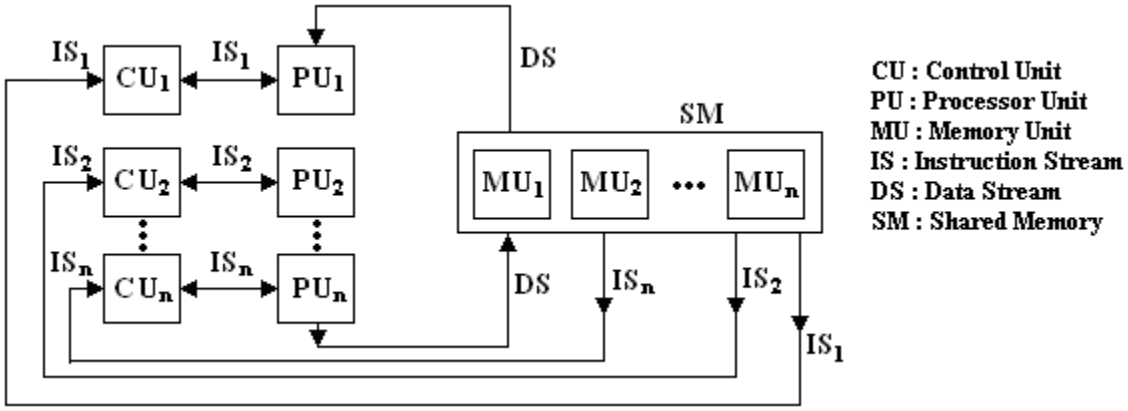
It represents an organization of computer which has multiple processors under the supervision of a common control unit. All processors receive the same instruction from the control unit but operate on different items of the data. SIMD computers are used to solve many problems in science which require identical operations to be applied to different data set synchronously. Examples are added a set of matrices simultaneously, such as $\sum_i \sum_k (a_{ik} + b_{ik})$. Such computers are known as array processors. SIMD computer organization is shown in figure below.



Multiple Instruction Stream, Single Data Stream (MISD)

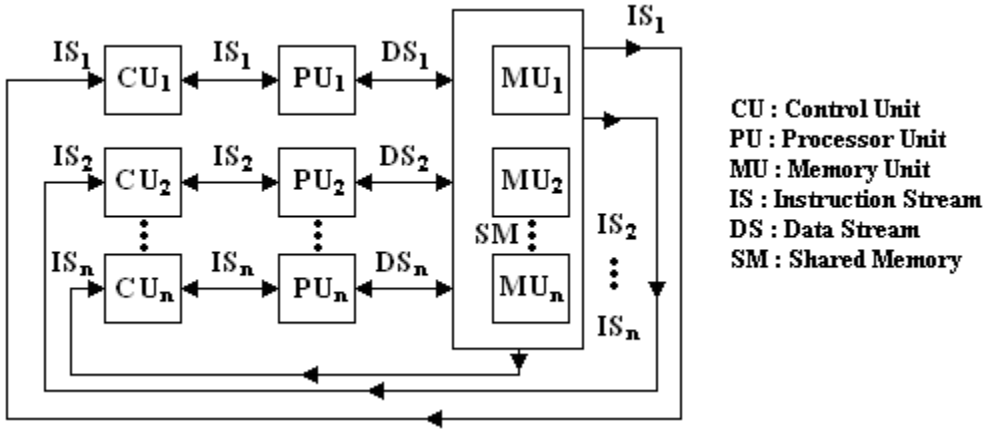
It refers to the computer in which several instructions manipulate the same data stream concurrently. In the structure different processing element run different

programs on the same data. This type of processor may be generalized using a 2-dimensional arrangement of processing elements. Such a structure is known as systolic processor. MISD computer organization is shown in figure below.



Multiple Instruction Stream, Multiple Data Stream (MIMD)

MIMD computers are the general purpose parallel computers. Its organization refers to a computer system capable of processing several programs at a same time. MIMD systems include all multiprocessing systems. MIMD computer organization is shown in figure below.



Q.18 A digital computer has a common bus system for 16 registers of 32 bits each. The bus is constructed with multiplexers.
(i) How many selection inputs are there in each multiplexer?
(ii) How many multiplexers are there in the bus? **(6)**

Ans:

Given, the common bus system consists of 16 registers and each register is of 32 bits.

(i) Since there are 16 registers i.e. 2^4 registers in the common bus system. Hence, 4 selection input lines are there to select one of 16 registers in each multiplexer.

(ii) Each register is of 32 bits, hence 32 multiplexers, one of each bit of the registers are there in the bus.

Q.19 Write a program to evaluate the arithmetic statement

$$Y = \frac{A - B + C}{G + H}$$

(i) Using an accumulator type computer with one address instruction.
(ii)Using a stack organized computer with zero-address instructions. **(8)**

Ans:

(i) Using one address instruction

Load A	AC ← M[A]
Sub B	AC ← AC – M[B]
Add C	AC ← AC + M[C]
Store T	M[T] ← AC
Load G	AC ← M[G]
Add H	AC ← AC + M[H]
Store Y	M[Y] ← AC
Load T	AC ← M[T]
Div Y	AC ← AC/M[Y]
Store Y	M[Y] ← AC

(ii) Using zero address instruction

Reverse polish notation of $Y = \frac{A - B + C}{G + H}$ is Y= AB–C+GH+/

Push A	TOS ← A
Push B	TOS ← B
Sub	TOS ← A – B
Push C	TOS ← C
Add	TOS ← A – B + C
Push G	TOS ← G
Push H	TOS ← H
Add	AC ← G + H
Div	TOS ← (A – B + C) / (G + H)
Pop Y	M[Y] ← TOS

Q.20 Explain the sequence that takes place when an interrupt occurs. **(8)**

Ans:
When an interrupt occur the three distinct actions as shown in Figure done automatically by the CPU is as follows:
(a) Saves the content and status of various CPU registers.

(b) Finding the cause of an interrupt.
(c) Branch the control to the Interrupt Service Routine (ISR).

Saving Content of CPU Registers

The interruption should not affect the result or logic of the current program. Hence, the CPU stores the content of various registers and the address of the next instruction to be executed so that once the interrupt has been serviced; the current program resumes its execution having the same condition. For saving the content and status of CPU registers the two operations can be:

(a) Certain registers in the CPU can be reserved for saving CPU status.
(b) Some location in the main memory can be reserved for saving CPU status.

If the internal registers of the CPU are used to save the status and content of different registers, the time taken for storing is less as compared to storing the CPU status at some locations in main memory. Generally, a part of memory is used as a stack for this purpose.

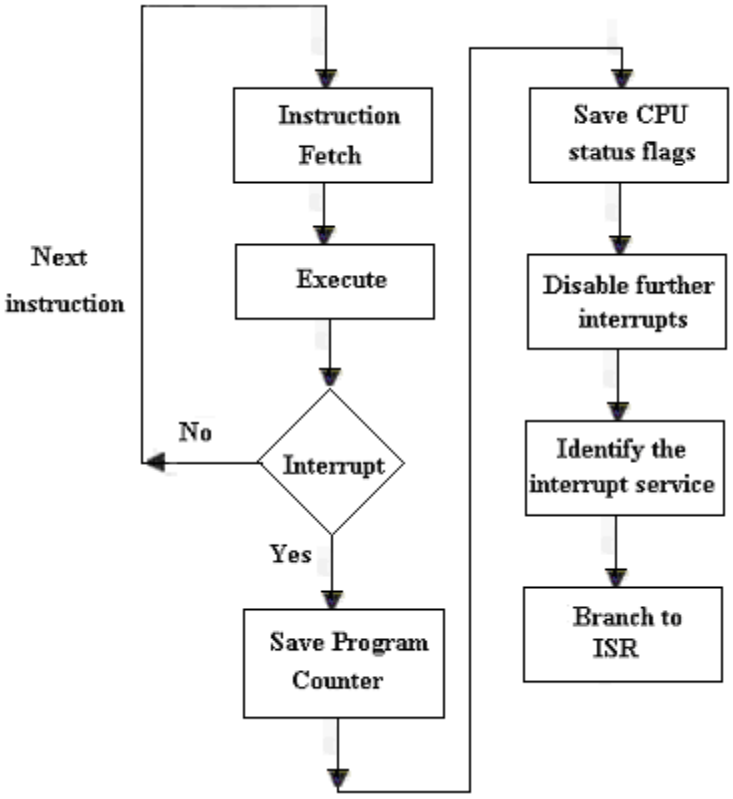
Identifying the interrupt source

For identifying the cause of interrupt or the source of interrupt, the CPU transmits interrupt acknowledgement (INTA) signal to the interrupt controller. The INTA signal conveys the following two messages to the interrupt controller:

(a) The CPU has sensed the interrupt signal issued by the interrupt controller.
(b) The interrupt controller should inform the CPU about the interrupt request should be serviced.

Branching to Interrupt Service Routine (ISR)

Once the source of the interrupt is found, the CPU should branch or transfer the control to the corresponding interrupt service routine (ISR) for taking necessary action. For this purpose, the CPU should have a mechanism to find the start addresses of the different ISRs. A common method is to store the ISRs of different interrupt in memory at different locations. But the start addresses of ISRs are stored at fixed location in memory one after another in form of a table. The CPU fetches the start address from the table and then loads the start address in program counter. And hence, the execution of the ISR is handled by CPU.



Interrupt Service Sequence

Q.21 Karnaugh Maps are useful for finding minimal implementations of Boolean expressions with only a few variables. Using the following K-Map:

(i) Find the minimal sum of products expression. Show your groupings.

(ii) Find the minimal sum of products expression. Show your groupings.

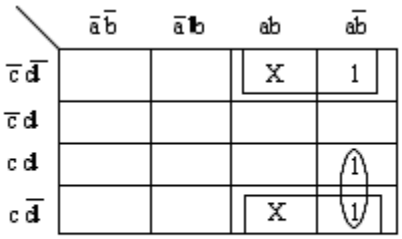
(iii) Are your solutions unique? If not, list and show the other minimal expression?

(iv) Does the minimal product of sums expression equal to minimal sum of products expression?

cd \ ab	00	01	11	10
00	0	0	X	1
01	0	0	0	0
11	0	0	0	1
10	0	0	X	1

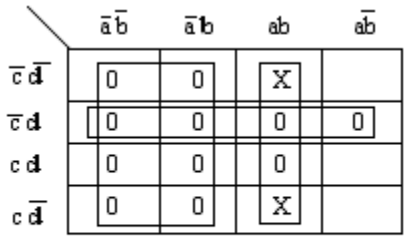
Ans:

i) Sum of Product



$F = \bar{d}a + c\,a\bar{b}$

ii) Product of Sum



$F' = a' + a\,b + c'\,d\,a$

$F = (a)(a'+b')(c+d'+a')$

iii) Our solution is unique. No other minimal expression.

iv) The minimal product of sums expression never equal to minimal sum of product expression

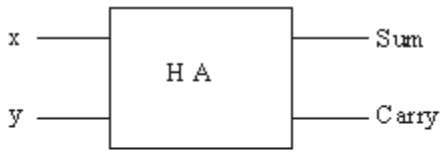
Q.22 A half-adder is a combinational logic circuit that has two inputs, x and y and two outputs, s and c that are the sum and carry-out, respectively, resulting from binary addition of x and y.

(i) Design a half-adder as a two-level AND-OR circuit.

(ii) Show how to implement a full adder, by using half adders.

Ans:

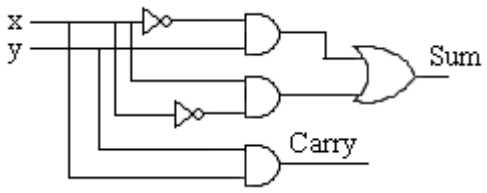
(i) Half-adder:-



Equations:-

$S = x'y + x\,y'$

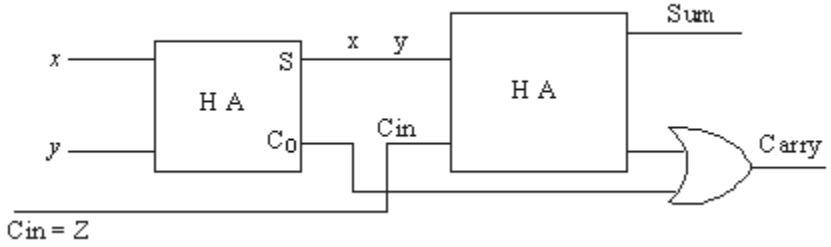
$C = x\,y$



Truth Table

INPUT		OUTPUT	
X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

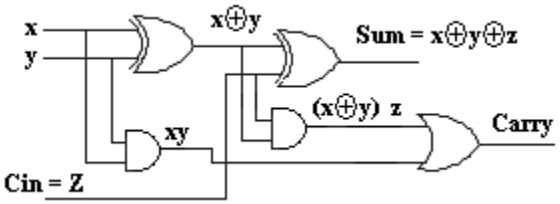
(ii) Full-Adder Circuit



Q.23 Show how to implement a full adder, by using half adders.

Ans:

Full-adder circuit by using two half-adder



Equations:

$S = \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xyz$

$C = xy + xz + yz$

Truth-Table

X	**Y**	**Z**	**C**	**S**
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1

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

Q.24 Simplify the following Boolean functions using four-variable maps.

(i) F (W, X, Y, Z) = ∑ (1, 4, 5, 6, 12, 14, 15)

(ii) F (A, B, C, D) = ∑ (0, 1, 2, 4, 5, 7, 11, 15)

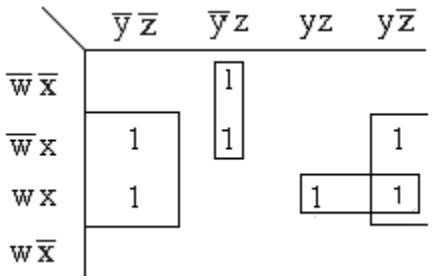
(iii) F (W, X, Y, Z) = ∑ (2, 3, 10, 11, 12, 13, 14, 15)

(iv) F (A, B, C, D) = ∑ (0, 2, 4, 5, 6, 7, 8, 10, 13, 15)

Ans:

(i) F (W, X, Y, Z) = ∑(1, 4, 5, 6, 12, 14, 15)

$F = x\bar{z} + \bar{w}\bar{y}z + wxy$

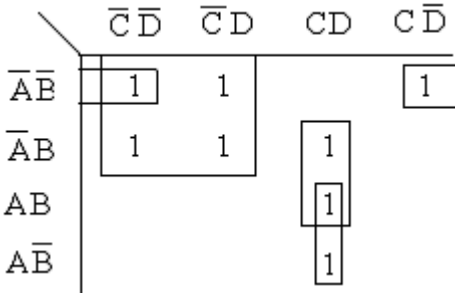


(ii) F (A, B, C, D) = ∑(0, 1, 2, 4, 5, 7, 11, 15)

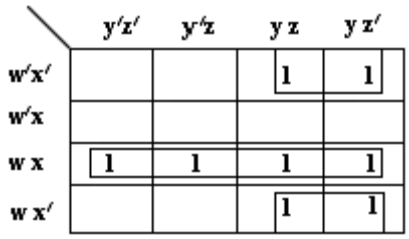
$\bar{A}B\bar{C}D + \bar{A}BCD + A\bar{B}CD + ABCD$

$F = \bar{A}\bar{C} + ACD + BCD$

$\bar{A}\bar{B}\bar{D}$

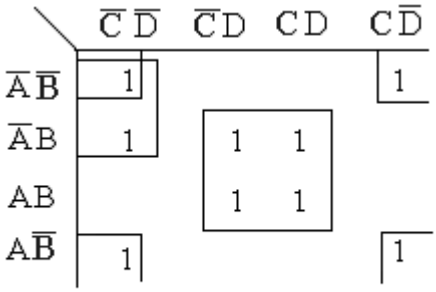


(iii) F (W, X, Y, Z) = ∑ (2, 3, 10, 11, 12, 13, 14, 15)



$F = wx + x'y$

(iv) $F(A, B, C, D) = \sum (0, 2, 4, 5, 6, 7, 8, 10, 13, 15)$



$F = BD + \bar{A}\bar{B} + \bar{B}\bar{D}$

Q.25 What is an instruction? With example explain three, two, one, zero address instructions.

Ans:

Instruction:-

A computer has a variety of instruction code format. It is the function of the control unit within the CPU to interpret each instruction code and provide the necessary control function needed to process the instruction.

Explanation:-

The format of an instruction is appear in memory words or in a control register. The bits of the instruction are divided into groups called fields. The most common fields found in instruction formats are:

1. An operation code field that specifies the operation to be performed.
2. An address field that designates a memory address or a processor register.
3. A mode field that specifies the way the operand or the effective address is determined.

Most computer fall into one of three types of CPU organizations:

1. Single accumulator organization.
2. General register organization.
3. Stack organization.

Accumulator-type organization is the basic computer presented. All operations are performed with an implied accumulator register. The instruction format in this type of computer uses one address field.

ADD X

The instruction format in general register type organization type of computer needs three register address fields.

ADD R1, R2, R3

In the stack-organized of CPU was presented. Would have PUSH and POP instructions.

PUSH x

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, $R1$and $R2$. 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

Zero-Addressing Instructions

A stack-organized computer does not use an address field for the instructions ADD and MUL. The PUSH and POP instructions, need an address field to specify the operand that communicates with the stack.

PUSH	A	TOS ← A
PUSH	B	TOS ← B
ADD		TOS ← (A + B)
PUSH	C	TOS ← C
PUSH	D	TOS ← D
ADD		TOS ← (C + D)
MUL		TOS ← (C + D) * (A + B)
POP	X	M[X] ← TOS

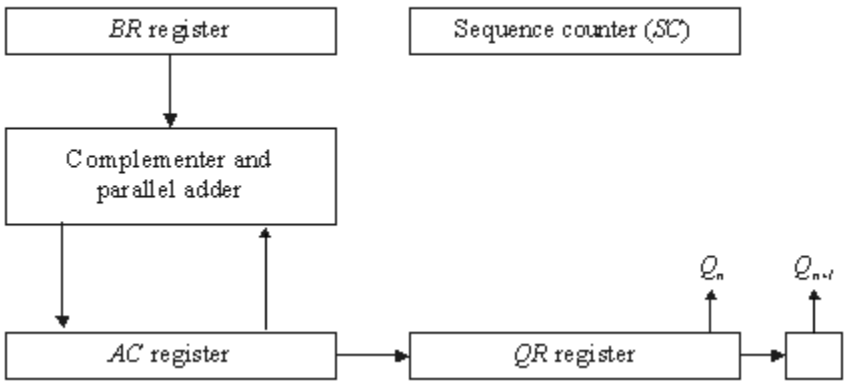
Q.26 Explain Booths algorithm for multiplying two signed binary numbers.

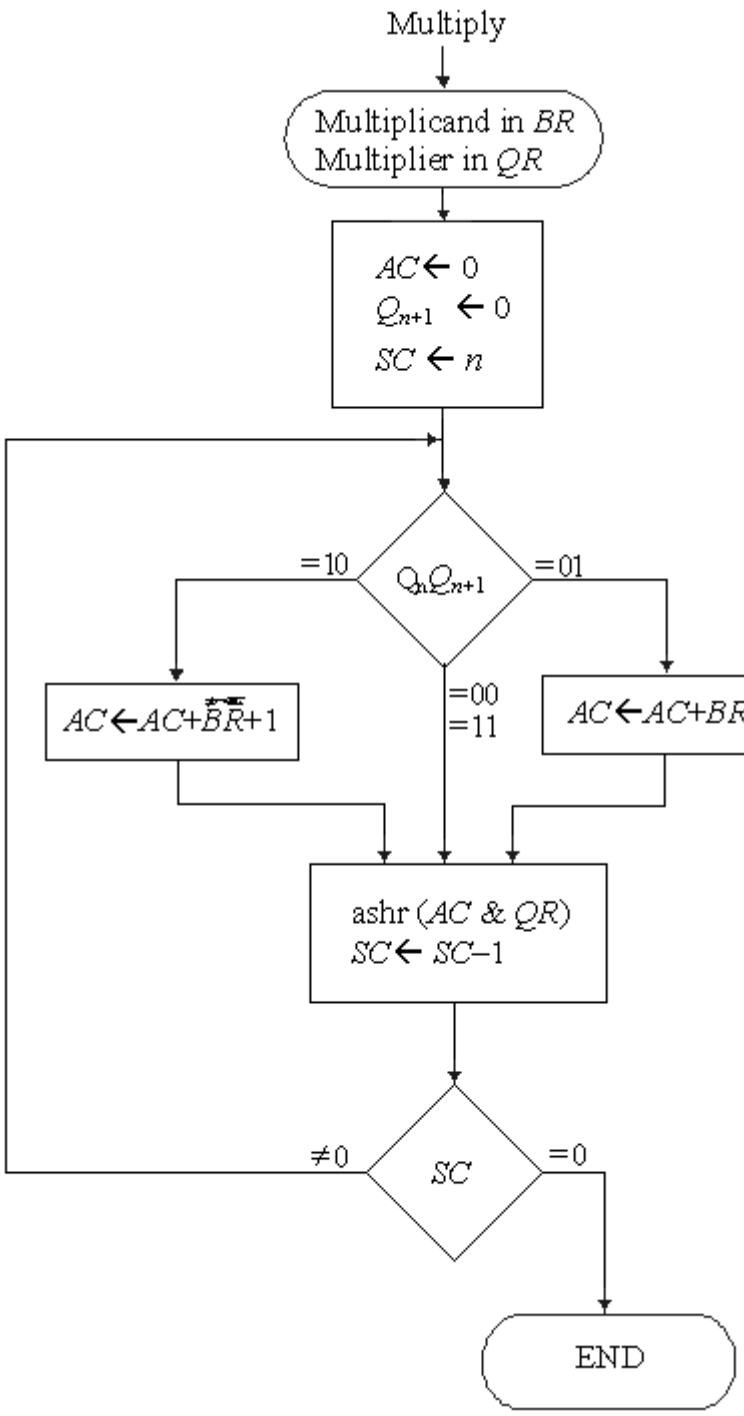
Ans:

Explanation:-

The hardware implementation of Booth algorithm requires the register configuration shown. The sign bits are not separted from the rest of the registers. Registers *A, B,* and *Q,* are named 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

Q_nQ_{n+1}	$\underline{BR}$ = 10111 $BR + 1 = 01001$	AC	QR	Q_{n+1}	SC
	Initial	00000	10011	0	101
1 0	Subtract BR	$\underline{01001}$ 01001			
	ashr	00100	110001	1	100
1 1	ashr	00010	01100	1	011

0 1	Add *BR*	10111			
		11001			
	ashr	11100	10110	0	010
0 0	ashr	11110	01011	0	001
1 0	Subtract *BR*	01001			
		00111			
	ashr	00011	10101	1	000

Q.27 A Computer uses a memory unit with 256 K words of 32 bits each. A binary instruction code is stored in one word of memory. The instruction has four parts: an indirect 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?

$256\ K = 2^8 \times 2^{10} = 2^{18}$

$64 = 2^6$

Ans:

(i) Memory unit = 256 K words = 2^{18} words

Hence, 18 bits is required to address one word of memory.

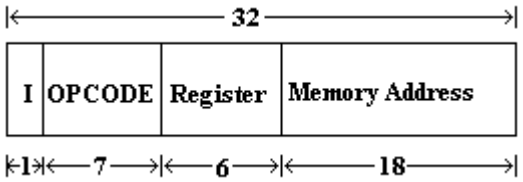
There are 64 registers i.e. 2^6 register

Hence, 6 bits are required to address one register.

1 bit is required for indirect bit.

Hence, 32 – (18+6+1) = 7 bits are required for operational code.

(ii)



(iii) Data : 32 bits;
address = 18 bits

Q.28 What is a micro-operation of list and explain the four categories of the most common micro-operations.

Ans:

A micro-operation 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 another register.

Common micro-operations

1) Arithmetic Microoperation
2) Logic Microoperation
3) Shift Microoperaztion
4) Arithmetic Logic Shift Unit

Explanation

1) Arithmetic microoperation - The basic arithmetic micro-operations are addition, subtraction, increment, decrement and shift. The arithmetic microoperation defined by the statement.

$$R_3 \leftarrow R_1 + R_2$$

2) Logic Micro-operation - Logic microoperation specify binary operations for strings of bits stored in registers. These operations consider each bit of the register separately and treat them as binary variables.

Example:- P : $R_1 \leftarrow R_1 \ R_2$

3) Shift Micro-operation - Shift microoperations are used for serial transfer of data. They are also used in conjunction with arithmetic, logic and other data - processing operations.

Example :- $R_1 \leftarrow \text{Shl } R_1$

$R_1 \leftarrow \text{Shl } R_1$

4) Arithmetic logic shift unit - To perform a microoperation the contents of specified registers are placed in the inputs of the common ALU. The ALU performs an operation and the result of the operation is then transferred to a destination register.

Q.29 The following transfer statements specify a memory explain the memory operation in each case.

(i) $R_2 \leftarrow M[AR]$ (ii) $M[AR] \leftarrow R_3$ (iii) $R_5 \leftarrow M[R_5]$

Ans:

i) Read memory word specified by the address in AR into register R_2.

ii) Write content of register R3 into the memory word specified by the address in AR.

iii) Read memory word specified by the address in R5 and Transfer content to R_5 (over writing the previous value).

Q.30 Discuss Direct Memory Access is detail.

Ans:

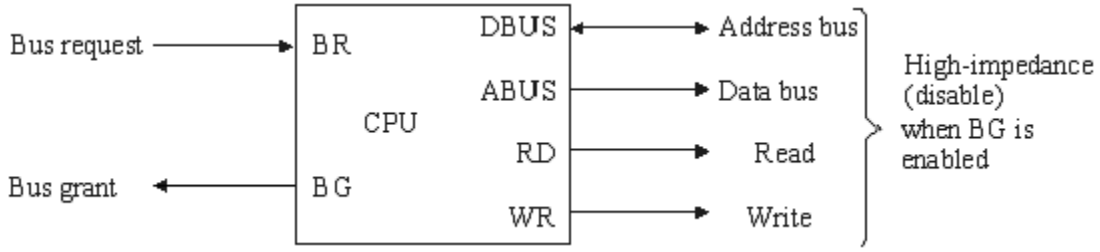
The transfer of data between a fast storage device such as magnetic disk and memory is often limited by the speed of the CPU. Removing the CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of

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.

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, takes control of the buses, and returns to its normal operation.
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



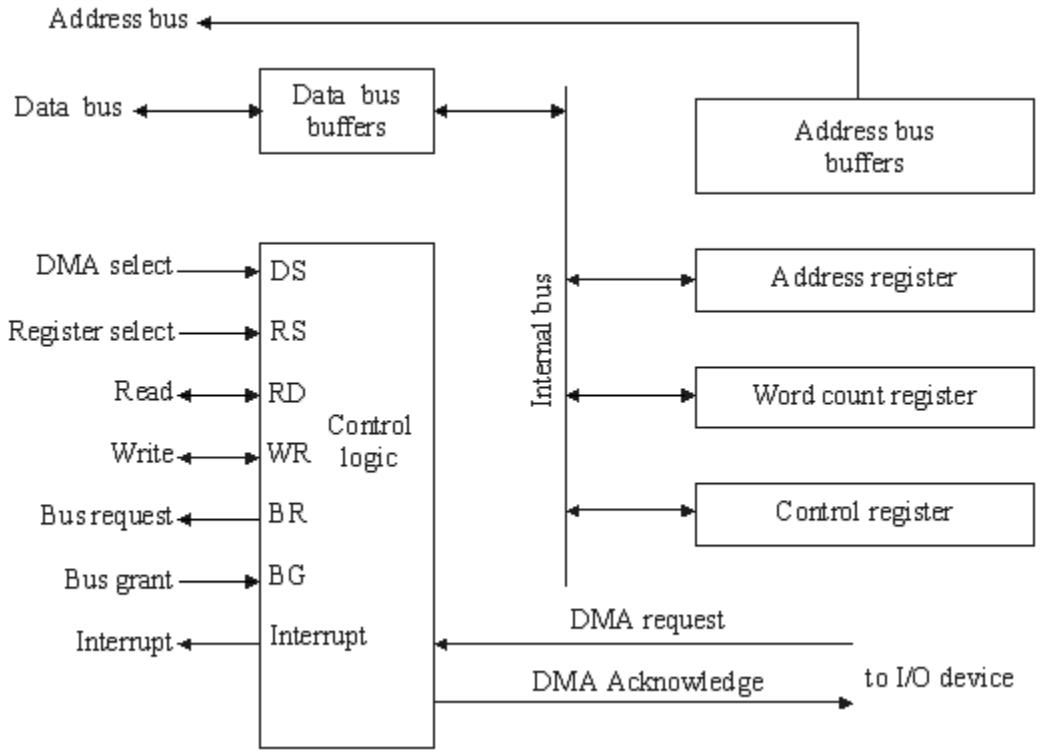
address bus by enabling the *DS* (DMA select) and *RS* (register select) inputs. The *RD* (read) and *WR* (write) inputs are bidirectional. When the *BG* (bus grant) input is 0, the CPU can communicate with the DMA registers through the data bus to read from or write to the DMA registers. When *BG* = 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.

DMA Transfer:-

Explanation:- 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 bidirectional. 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 registers. When *BG* = 1, the *RD* and *WR* are output lines from the 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.

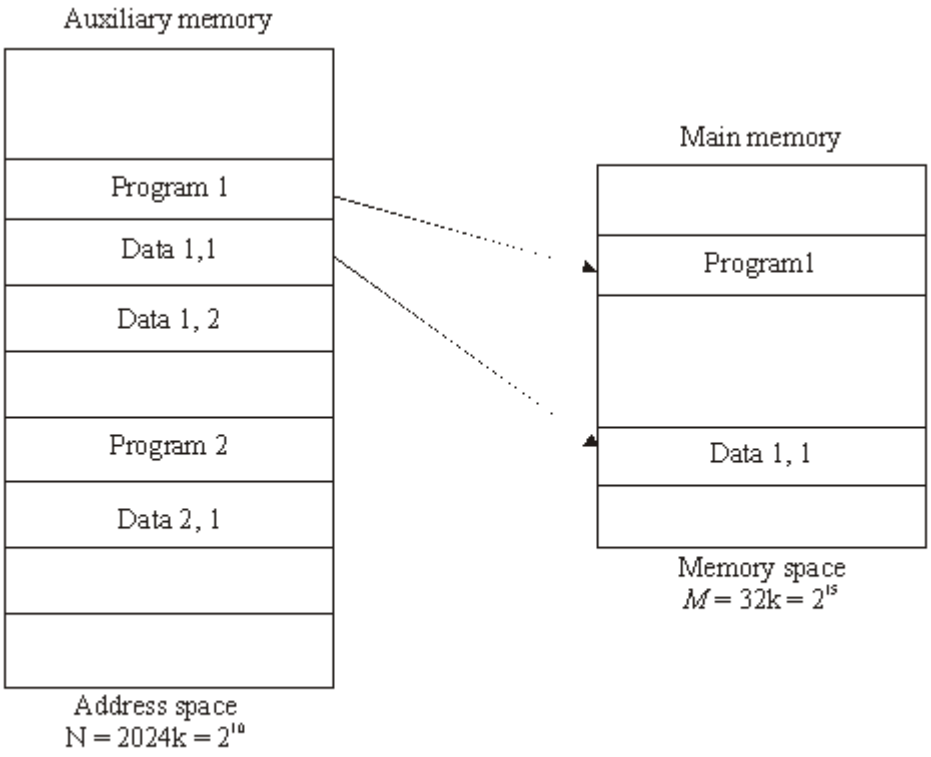
Q.31 Explain the concept of virtual memory with the help of diagram. Explain how virtual address in mapped to actual physical address.

Ans:

Virtual Memory:-

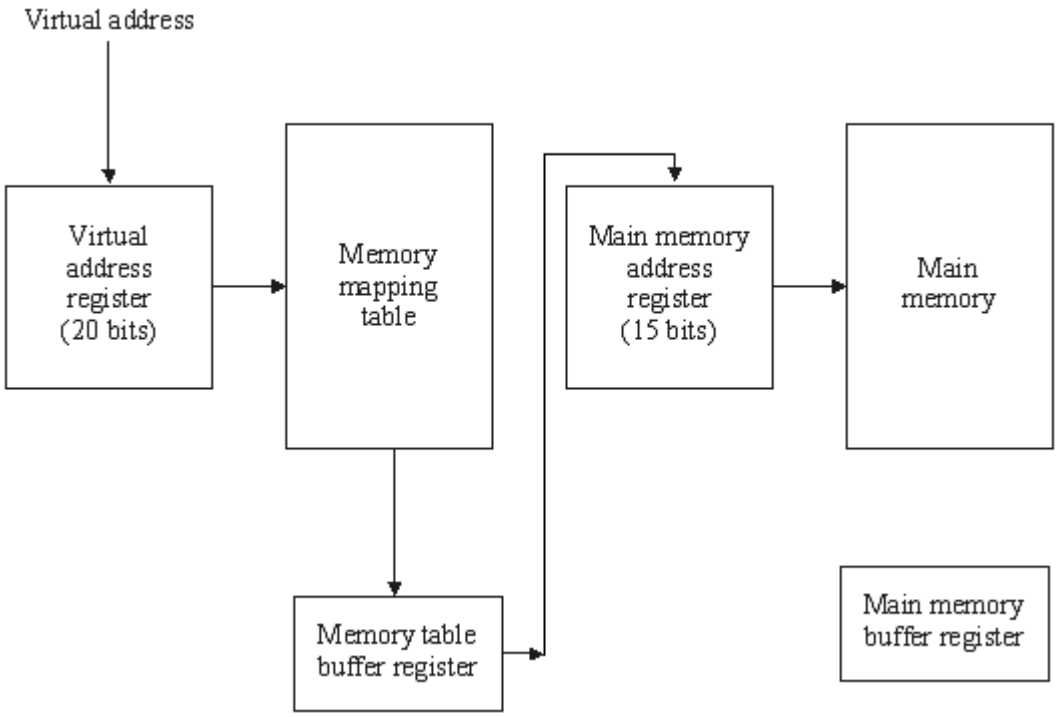
Virtual memory is a concept used in some large computer systems that permit the user to construct programs as though a large memory space were available, equal to the totality of auxiliary memory. Virtual memory is used to give programmers the illusion that they have a very large memory at their disposal, even though the computer actually has a relatively small main memory. A virtual memory system provides a mechanism for translating program generated address 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



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

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. To map a virtual address of 20 bits to a physical address of 15 bits we requires a table shown in figure below. The mapping is a dynamic operation, which means that every address is translated immediately as a word is referenced by CPU.



Q.32 Differentiate between:

(i) Isolated I/O and memory-mapped I/O

(ii) Source Initiated and Destination Initiated transfer using handshaking.

Ans:

(i) Isolated I/O and memory-mapped I/O.

Isolated I/O	Memory-Mapped I/O
(i) I/O transfer is made through separate read and write line.	(i) Memory transfer is made through separate read and write lines.
(ii) The I/O read and write control lines are enabled during the I/O are enabled during a memory transfer.	(ii) The memory read and memory write control lines are enabled during a memory transfer.
(iii) The isolated I/O configuration the CPU has distinct input and output instructions and each instruction is associated with the address of an interface register.	(iii) Computer with memory mapped I/O can use memory type instructions to access I/O data.

(iv)	The isolated I/O method isolates memory and I/O address so that memory address values are not affected by interface address assignment.	(iv)	In a memory mapped I/O organization theme are no specific input or output instructions.

(ii) Source Initiated and Destination Initiated transfer using handshaking.

Refer Page 395, 396, 397 from moris mano (3rd Edition)

Q. 33 Differentiate between hardwired control and micro programmed control. Draw the block diagram of a basic hardwired control organization with two decoders, a sequence counter and a number of control logic gates?

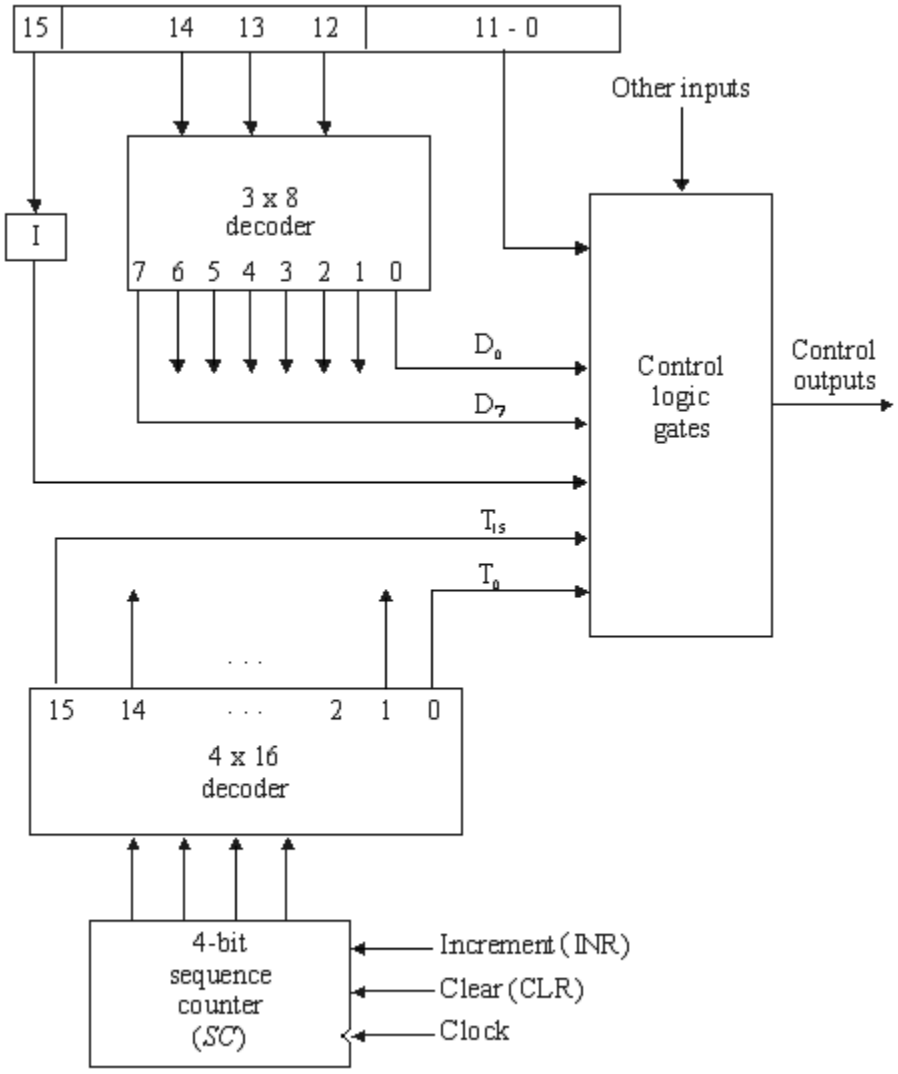
Ans:

The block diagram of the control unit is shown. It consists of two decoders, a sequence counter, and a number of control logic gates. An instruction read from memory is placed in the instruction register (IR). The position of this register in the common bus system is shown in figure. The instruction register is shown, where it is divided into three parts: the I bit, the operation code, and bit 0 through 11. The operation code in bits 12 through 14 are decoded with a 3 x 8 decoder. The eight outputs of the decoder are designated by the symbols D_0 through D_7. Bit 15 of the instruction is transferred to a flip-flop designated by the symbol I. Bits 0 through 11 are applied to the control logic gates. The 4-bit sequence counter can count in binary from 0 through 15. The outputs of the counter are decoded into 16 timing signals T_0 through T_{15}.

The sequence counter SC can be incremented or cleared synchronously, the timing diagram shows the time relationship of the control signals.

	Hardwired Control		**Microprogrammed control**
(i)	In hardwired organization, the control logic is implemented with gates, flip-flops, decoders and other digital circuits.	(i)	In microprogrammed organization, the control information is stored in a control memory. The control memory is programmed to initiate the required sequence of microoperations.

(ii) A hardwired control requires changes in the wired among the various components if the design has to be modified.	(ii) In microprogrammed control any required changes or modification can be done by updating the microprogram in control memory.



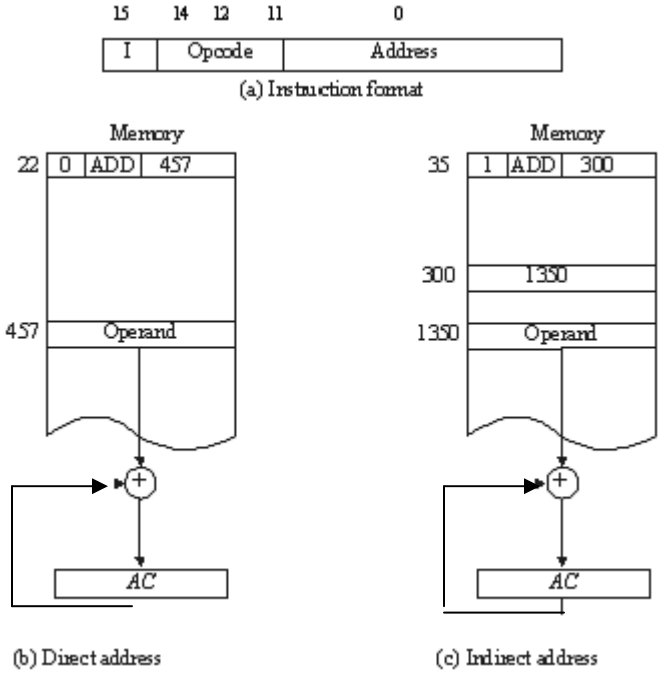
Control unit of basic computer

Q. 34 What is the difference between a direct and an indirect address instruction? How may references to memory are needed for each type of instruction to being an operand into a processor register.

Ans:

Consider the instruction code format shown in figure (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 1 for an indirect address. A direct address instruction is shown in figure (b) . It is placed in address 22 in memory. The I bit is 0,

so the instruction is recognized as a direct address instruction. The opcode 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 it to the content of AC. The instruction in address 35 shown in figure (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.



A direct address instruction needs two reference to memory:

i) Read instruction

ii) Read operand

An indirect address instruction needs three reference to memory:

i) Read instruction

ii) Read effective address

iii) Read operand.

Q. 35 List any two

(i) Register reference instruction

(ii) Memory reference instruction

(iii) Input-output instruction

Ans.

(i)

Register-Reference Instructions

$D_7I'T_3 = r$ (common to all register-reference instructions)			
$IR(i) = B$ (bit in IR (0-11) that specifies the operation)			
	r:	$SC \leftarrow 0$	Clear SC
CLA	rB_{11}:	$AC \leftarrow 0$	Clear AC
CLE	rB_{10}:	$E \leftarrow 0$	Clear E
CMA	rB_9:	$AC \leftarrow \overline{AC}$	Complement AC
CME	rB_8:	$E \leftarrow \overline{E}$	Complement E
CIR	rB_7:	$AC \leftarrow shr\,AC, AC(15) \leftarrow E, E \leftarrow AC(0)$	Complement E
CIL	rB_6:	$AC \leftarrow shl\,AC, AC(0) \leftarrow E, E \leftarrow AC(15)$	Circulate left
INC	rB_5:	$AC \leftarrow AC+1$	Increment AC
SPA	rB_4:	If (AC(15)=0) then ($PC \leftarrow PC+1$)	Skip if positive
SNA	rB_3:	If (AC(15)=1) then ($PC \leftarrow PC+1$)	Skip if negative
SZA	rB_2:	If (AC=0) then ($PC \leftarrow PC+1$)	Skip if AC zero
SZE	rB_1:	If (E=0) then ($PC \leftarrow PC+1$)	Skip if E zero
HLT	rB_0:	$S \leftarrow 0$ (S is a start-stop flip-flop)	Halt computer

Memory-Reference Instructions

Symbol	Operation decoder	Symbolic description
AND	D_0	$AC \leftarrow AC \wedge M[AR]$
ADD	D_1	$AC \leftarrow AC + M[AR], E \leftarrow Cout$
LDA	D_2	$AC \leftarrow M[AR]$
STA	D_3	$M[AR] \leftarrow AC$
BUN	D_4	$PC \leftarrow AR$
BSA	D_5	$M[AR] \leftarrow PC, PC \leftarrow AR+1$
ISZ	D_6	$M[AR] \leftarrow M[AR]+1$
		If $M[AR]+1=0$ then $PC \leftarrow PC+1$

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 instruction]			
	p:	SC ← 0	Clear SC
INP	pB_{11}:	AC(0-7) ← INPR, FGI ← 0	Input character
OUT	pB_{10}:	OUTAR ← 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.36. What is the function of interrupt facility in a multiprogram environment? Explain how a source routine is initiated for the input or output transfer. Lists the tasks which this service routine is supported to perform. **(8)**

Ans: Refer Program Interrupt from page 281,282 from Morris Mano, (3rd Edition)

Q. 37 Write an assembly language program to:

(i) Input a character & store it in memory

(ii) Add two numbers

Ans:

(i) Input a character & store it in memory

```
CIF,    SKI         /check input flag
        BUN CIF     /Flag=0, branch to check again
        INP         /Flag=1, input character
    STA CHR         /Store character
        HLT
CHR,    M           /Store character in memory
```

(ii) Add two numbers

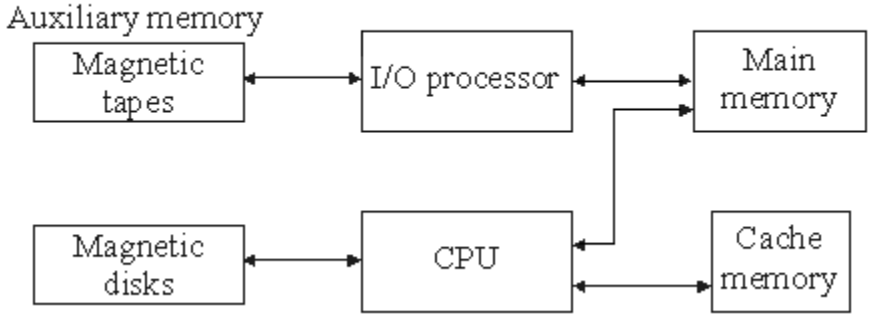
```
        ORG  0          /Origin of program is location O
        LDA  A          /Load operand from location A
        ADD  B          /Add operand from location B
        STA  C          /Store sum in location C
        HLT             /Halt Computer
A,      DEC  83         /Decimal operand
B,      DEC-23          /Decimal operand
C,      DEC  0          /Sum stored in location C
        END             /End of symbolic program
```

Q.38 Explain what is meant by a cache memory. What general principles are used to make effective use of cache memory?

Ans:

A special very-high-speed memory called a cache is sometimes used to increase the speed of processing by making current programs and data available to the CPU at a rapid rate. The cache memory is employed in computer systems to compensate for the speed differential between main memory access time and processor logic. CPU logic is usually faster than main memory access time, with the result that processing speed is limited primarily by the speed of main memory.

Memory hierarchy in a computer system



Cache Memory

The references to memory at any given interval of time tend to be confined within a few localized areas in memory. This phenomenon is known as the property of locality of reference.

If the active portions of the program and data are placed in a fast small memory, the average memory access time can be reduced, thus reducing the total execution time of the program. It is placed between the CPU and main memory as illustrated. The cache

memory access time is less than the access time of main memory by a factor of 5 to 10. The cache is the fastest component in the memory hierarchy and approaches the speed of CPU components.

The basic operation of the cache is as follows. When the CPU needs to access memory, the cache is examined. If the word is found in the cache, it is read from the fast memory. If the word addressed by the CPU is not found in the cache, the main memory is accessed to read the word.

The performance of cache memory is frequently measured in terms of a quantity called hit ratio. When the CPU refers to memory and finds the word in cache, it is said to produce a hit. If the word is not found in cache, it is in main memory and it counts as a miss. The ratio of the number of hits divided by the total CPU references to memory is the hit ratio.

The average memory access time of a computer system can be improved considerably by use of a cache. If the hit ratio is high enough so that most of the time the CPU accesses the cache instead of main memory, the average access time is closer to the access time of the fast cache memory.

The basic characteristic of cache memory is its fast access time. Therefore, very little or no time must be wasted when searching for words in the cache. The transformation of data from main memory to cache memory is referred to as a mapping process. Three types of mapping procedures are of practical interest when considering the organization of cache memory.

1. Associative mapping
2. Direct mapping
3. Set-associative mapping.

Q. 39 What is associative memory? Explain with the help of a block diagram. Also mention the situation in which associative memory can be effective utilized.

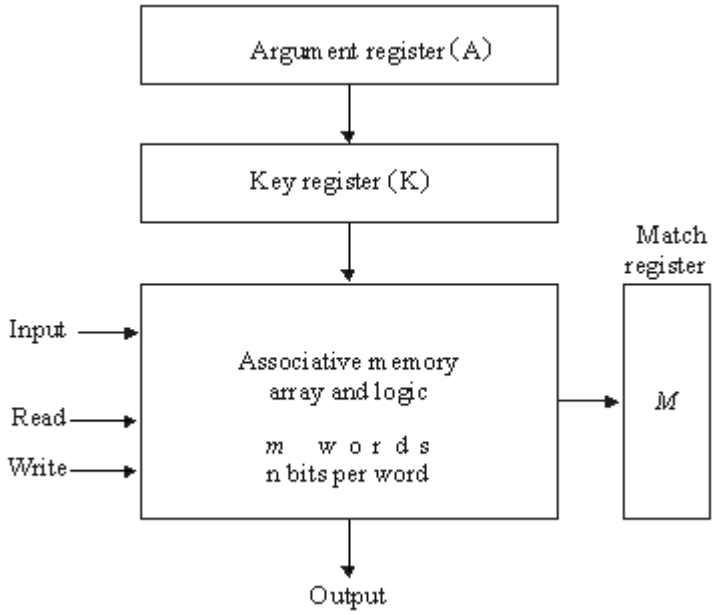
Ans:

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 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.

Hardwared Organization

The block diagram of an associative memory is shown. It consists of a memory array and logic for m words with *n* bits per word. The argument register A and key register K each have n bits, one for each bit of a word. Each word in memory is compared in parallel with the content of the argument register. The 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 match register.



Block diagram of associative memory

The key register provides a mask for choosing a particular field or key in the argument word. The entire argument is compared with each memory word if the key register contains all 1's. Otherwise, only those bits in the argument that have 1's in their corresponding position of the key register are compared.

A numerical example, suppose that the argument register A and the key register K have the bit configuration shown below. Only the three leftmost bits of A are compared with memory words because K has 1's these positons.

Word 2 matches the unmasked argument field because the three leftmost bits of the argument and the word are equal.

A	101 111100	
K	111 000000	
Word1	100 111100	no match
Word2	101 000001	match

Q. 40 With neat flow chart, explain binary division process.

Ans:

Hardware Algorithm:-

The flowchart is shown in the figure. 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. A constant is set into the sequence counter *SC* to specify the number of bits in the quotient.

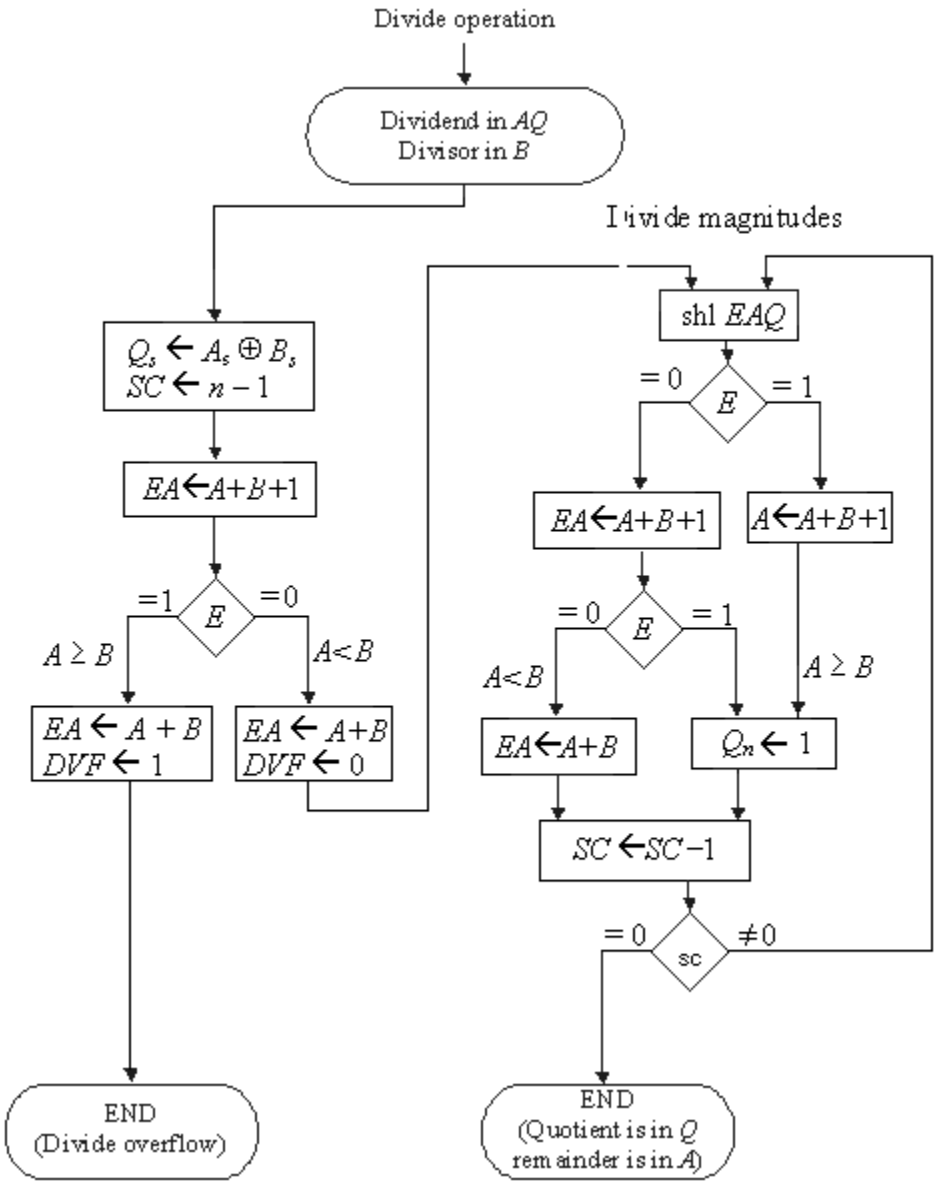
A divide-overflow condition is tested by subtracting the division in B from half of the bits of the dividend stored in A. If A $\geq$ B, the divide-overflow flip-flop *DVF* is set and

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, we know that EA > B because EA consists of a 1 followed by n - 1 bits while B consists of only n - 1 bits.

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*.

Flowchart for divide operation



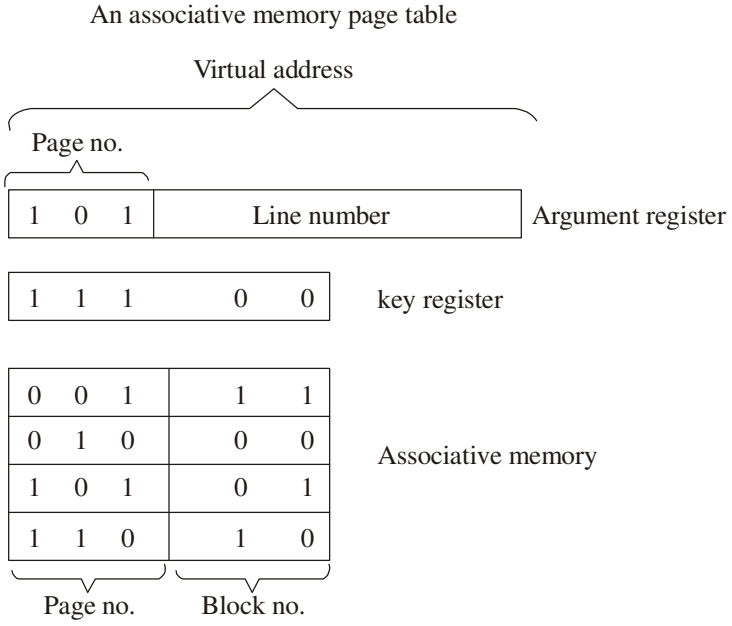
Q. 41 Describe through diagram how matched word can be read out from on associative memory.

Ans:

A system with *n* pages and *m* blocks would require a memory-page table of *n* locations of which up to *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. This method can be implemented by means of an associative memory with each word in memory 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 number is extracted.

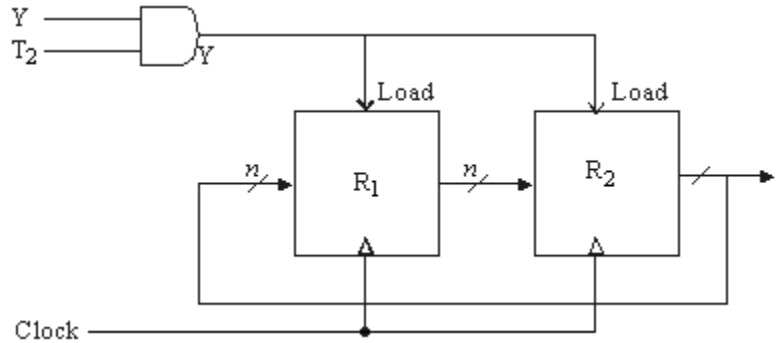
Each entry in the associative memory array consists of two fields. The first three bits specify a field for storing the page number. The last two bits constitute a field for storing the block number. The virtual address is placed in the argument register. The page number bits in the argument register 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. 42 Give the hardware implementation of flowing

$$YT_2 : R_2 \leftarrow R_1,\ R_1 \leftarrow R_2$$

Ans:

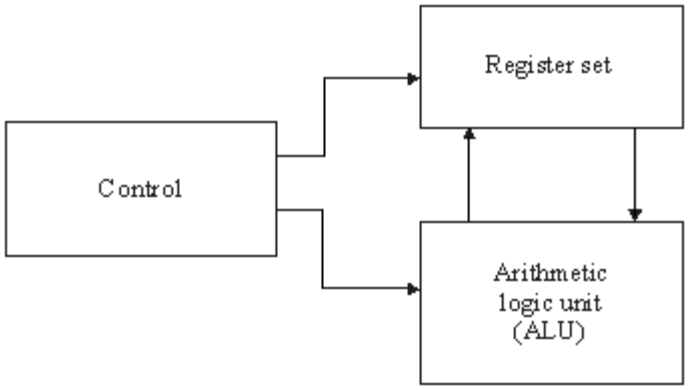


Q. 43 Briefly discuss the steps followed in designing a CPU.

Ans:

The part of the computer that performs the bulk of data processing operations is called the central processing unit and is referred to as the CPU. The register set stores intermediate data used during the execution of the instructions. The arithmetic logic unit (ALU) performs the required microoperations for executing the instructions. The control unit supervises the transfer of information among the registers and instructs the ALU as to which operation to perform.

The registers communicate with the ALU through buses and explain the operation of the memory stack.



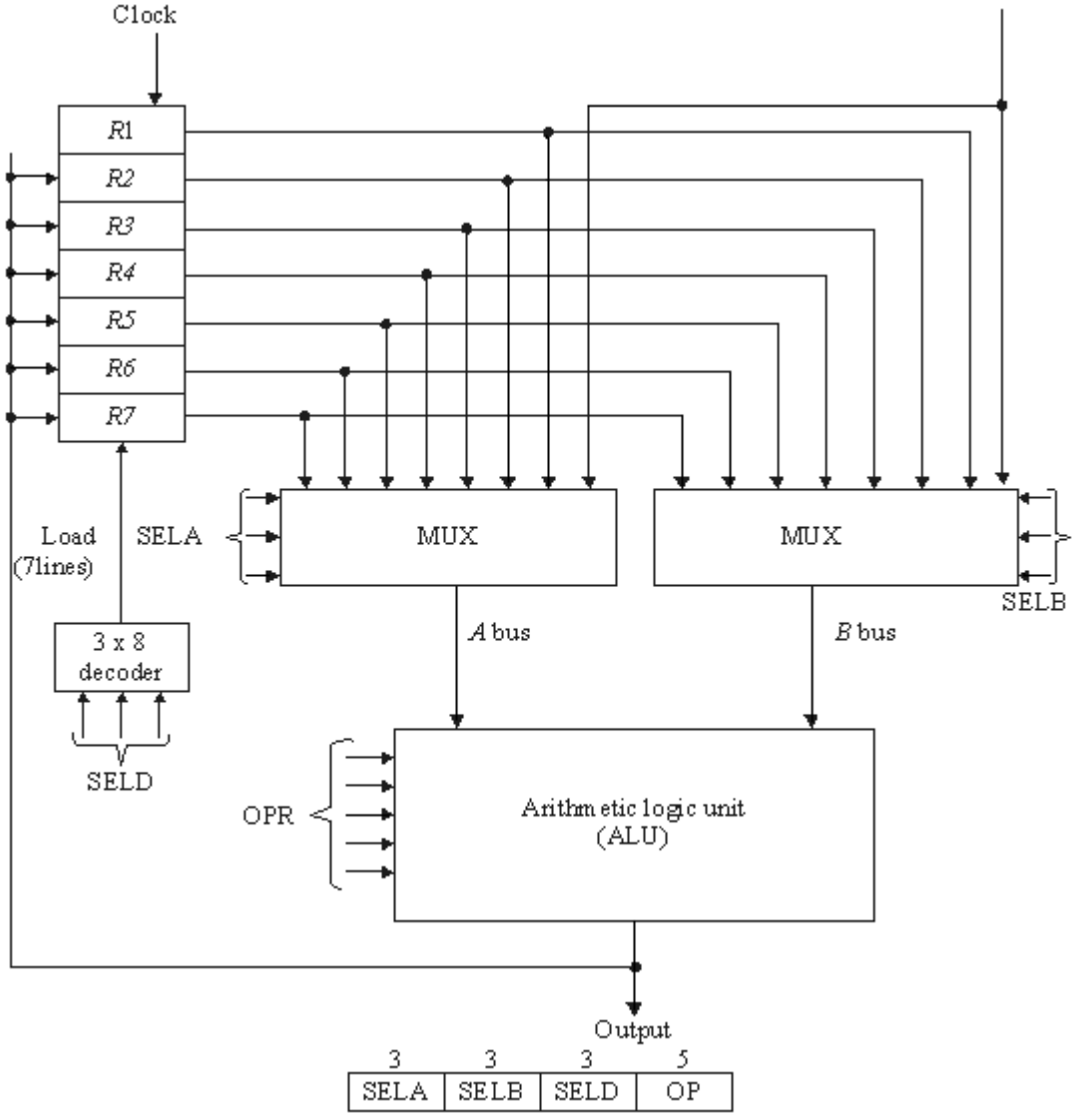
Major components of CPU

A large number of registers are included in the CPU, it is most efficient to connect them through a common bus system. The registers communicate with each other not only for direct data transfers, but also while performing various microoperations. Hence, it is necessary to provide a common unit that can perform all the arithmetic, logic, and shift microoperations in the processor.

A bus organization for seven CPU registers is shown. The output of each register is connected to two multiplexers (MUX) to form the two buses *A* and *B*.

There are 14 binary selection inputs in the unit, and their combined value specifies a *control word*. The 14-bit control word is defined. It consists of four fields. Three fields contain three bits each, and one field has five bits. The three bits of SELA select a source register for the *A* input of the ALU. The three bits of SELB select a register for the *B* input of the ALU. The three bits of SELD select a destination register using the decoder and its seven load outputs. The five bits of OPR select one of the operations in the ALU.

The ALU provides arithmetic and logic operations. In addition, the CPU must provide shift operations. The shifter may be placed in the input of the ALU to provide a preshift capability, or at the output of the ALU to provide postshifting capability. In some cases, the shift operations are included with the ALU.



(c) Control word

Register set with common ALU

Q. 44 What are the types of instructions that are included in an instruction set? Give two examples of each type.

Ans:
Refer Section 8.6 of Morris mano (3rd Edition)

Q. 45 The following program is stored in the memory unit of the basic computer. Show the contents of the AC, PC and IR (in hexa decimal), at the end, after each instruction is executed. All number listed below are in hexadecimal.

Location	Instruction
010	CLA
011	ADD 016
012	BUN 014
013	HLT
014	AND 017
015	BUN 013
016	C1A5
017	93C6

Ans:

		AC	PC	IR
010	CLA	0000	011	7800
017	ADD 016	C1A5	012	1016
012	BUN 014	C1A5	014	4014
013	HLT	8184	014	7001
014	AND 017	8184	015	0017
015	BUN 013	8184	013	4013
016	C1A5			
017	93C6			

$(C1A5)_{16}$	=	1100	0001	1010	0101	AND
$(93C6)_{16}$	=	1001	0011	1100	0110	
		1000	0001	1000	0100	
	=	$(8184)_{16}$				

Q. 46 What do you understand by subroutine call? What steps are carried out when a subroutine call is encountered? State the difference between subroutine call and interrupts.

Ans:

The procedure for branching to a subroutine and returning to the main program is referred to as a subroutine linkage. The BSA instruction performs an operation commonly called subroutine call. The procedure used in the basic computer for subroutine linkage is commonly found in computer with only one processor register.

Subroutine parameters and data linkage.

When a subroutine is called, the main program must transfer the data. The subroutine shifted the number and left is 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 to 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 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 200. The return to the main program is to the HLT instruction.

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.

Similarity between subroutine and program interrupt. The interrupt procedure is in principle guite similar to a subroutine call except for three variations.

1) The interrupt is usually invited by an internal and external signal rather than execution of an instruction.

2) The address of the interrupt service program is determined by the hardware rather than address field of an instruction.

3) An interrupt procedure usually stores all the information necessary to define the state of CPU rather than storing only the program counter.

Q. 47 With neat diagram, explain the strobe control data transfer method, state its disadvantage.

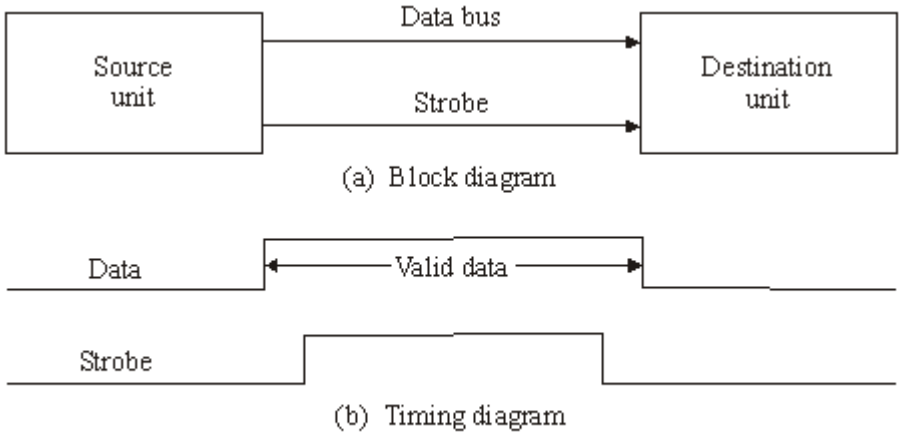
Ans:

Strobe Control:-

The strobe control method of asynchronous data transfer employs a single control line to time each transfer. The strobe may be activated by either the source or the destination unit.

The data bus carries the binary information from source unit to the destination unit. Typically, the bus has multiple lines to transfer an entire byte or word. The strobe is a single line that informs the destination unit when a valid data word is available in the bus.

In the timing diagram the source unit first places the data on the data bus. After a brief delay to ensure that the data settle to a steady value, the source activates the strobe pulse. The information on the data bus and the strobe signal remain in the active state for a sufficient time period to allow the destination unit to receive the data. The source removes the data from the bus a brief period after it disables its strobe pulse, which indicates that the data bus does not contain valid data. New valid data will be available only after the strobe is enabled again.

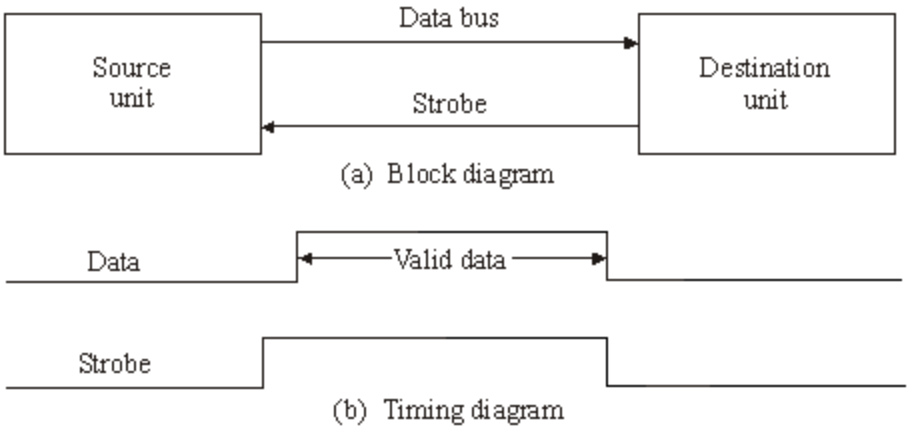


Source-initiated strobe for data transfer

Data transfer initiated by the desintation unit. The destination unit activates the strobe pulse, informing the source to provide the data. The source unit responds by placing the requested binary information on the data bus. The data must be valid and remain in the bus long enough for the destination unit to accept it.

Disadvantages:-

The disadvantage of the strobe method is that the source unit that initiates the transfer has no way of knowing whether the detination 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.



Destination-initiated strobe for data transfer

Q. 48 A virtual memory system has 6k words of address space and 3k words of memory space. Page references are made by CPU in following sequence:

3, 2, 0, 3, 4, 1, 2, 2, 0

Find out the pages that are available at the end if the replacement algorithm used is

(a) LRU (b) FIFO Assume the page and block size of 1k words.

Ans:

(a) LRU

3	2	0	3	4	1	2	2	0
3	3	3		3	3	2		2
	2	2		4	4	4		0
		0		0	1	1		1

(b) FIFO

3	2	0	3	4	1	2	2	0
3	3	3		4	4	4		0
	2	2		2	1	1		1
		0		0	0	2		2

Q. 49 What is the need of I/O interface?

Ans: Refer Section 11.2 of Morris Mano (3rd Edition)

Input-output interface provides a method for transferring information between internal storage and external I/O devices. Peripherals connected to a computer need special communication links for interfacing them with the central processing unit.

Q. 50 State the advantages of assembly language over machine language.

Ans:

Advantages

1) Assembly language programs are more understandable than machine language program.

2) Debugging is easy than machine language.

3) Assembly language is easy to write the program than machine language.

4) Assembly language uses assembler to generate object code.

5) Assembly language programs takes less computation time and computer's main memory.

Q. 51 State the advantages of memory mapped I/O over 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 and 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.

Q. 52 Write short notes on any two.

(i) Instruction pipeline.

(ii) DMA based data transfer.

(iii) Shift operation of data in a register.

Ans:

(i) Instruction pipeline:-

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. Whenever 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.

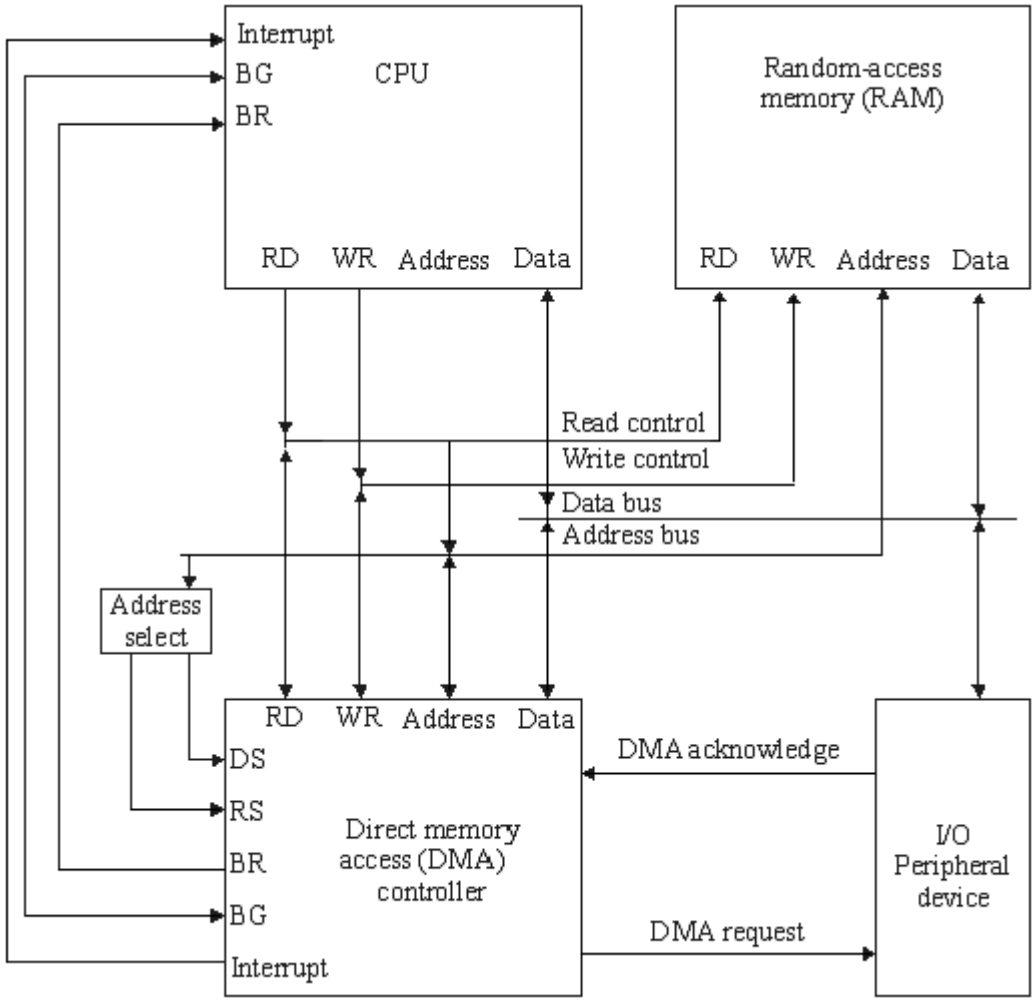
There are certain difficulties that will prevent the instruction pipeline from operating at the maximum rate. Different segments may take different time to operate on the incoming information. Some segments are skipped for certain operations. For example, a register mode instruction does not need an effective address calculation. Two or more segments may require memory access at the same time, causing our 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.

(ii) DMA based data transfer:-

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.

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. When this input is active, the CPU terminates the execution of the current instruction. The CPU activates the bus grant (BC) 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, takes control



of the buses, and returns to its normal operation.

When the DMA takes control of the bus system, it communicates directly with the memory.

(iii) Shift operation of data in a register:-

The contents of a register can be shifted to the left or the right. There are three types of shifts: logical, circular, and arithmetic.

A logical shift is one that transfers 0 through the serial input. We will adopt the symbols shl and shr for logical shift-left and shift-right microoperations. For example:

$R1 \leftarrow \text{shl}\, R1$

$R2 \leftarrow \text{shr}\, R2$

The circular shift circulates the bits of the register around the two ends without loss of information. This is accomplished by connecting the serial output of the shift

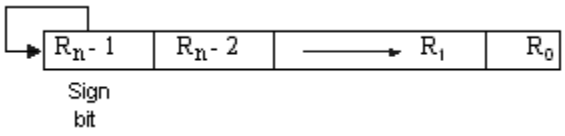
register to its serial input. We will use the symbols cil and cir for the circular shift left and right.

Shift Micro operations

Symbolic designation	Description
R ← shl R	Shift-left register R
R ← shr R	Shift-right register R
R ← cil R	Circular Shift-left register R
R ← cir R	Circular Shift-right register R
R ← ashl R	Arithmetic shift-left R
R ← ashr R	Arithmetic shift-right R

An arithmetic shift is a microoperation that shifts a signed binary number to the left or right. An arithmetic shift-left multiplies a signed binary number by 2. An arithmetic shift-right divides the number by 2. Arithmetic shifts must leave the sign bit unchanged because the sign of the number remains the same when it is multiplied or divided by 2. The leftmost bit in a register holds the sign bit, and the remaining bits hold the number. The sign bit is 0 for positive and 1 for negative. Negative numbers are in 2's complement form.

Arithmetic shift right

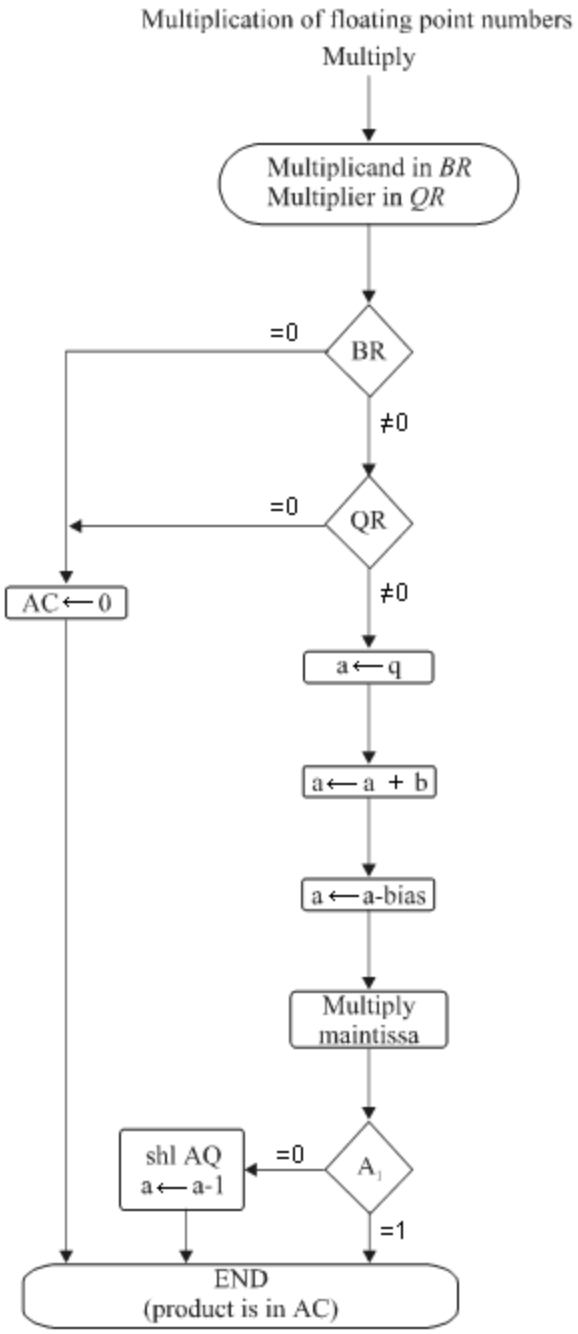


Q. 53 With neat flowchart, explain the process of multiplication of floating point numbers.

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.

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

The flowchart for floating-point multiplication is shown in figure. 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 multiplication of the mantissa is done as in the fixed point case with the product residing in *A* and *Q*. Overflow cannot occur during multiplication, so there is no need to check for it.

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. Note that only one normalization shift is necessary. The multiplier and multiplicand were originally normalized and contained fractions. The smallest normalized operand is 0.1, so the smallest possible product is 0.01. Therefore, only one leading zero may occur.

Although the low-order half of the mantissa is in *Q*, we do not use it for the floating point product. Only the value in the AC is taken as the product.

Q. 54 Divide dividend $(-53)_{10}$ by divisor $(-7)_{10}$ by using binary division algorithm Show all steps.

Ans:

Divisor B = 111011 B+1 = 000101

	E	A	Q	SC
Dividend		001011	000000	6
Shl EAQ	0	010110	000000	
add B+1		000101		
E = O	0	011011		
Set Q_0=1	0	011011	000001	5
Shl FAQ	0	110110	000010	
Add B+1		000101		
	0	111011		
E =);	0	111011		
Set Q_0=1	0	0111011	000011	4
Shl EAQ	1	110110	000110	
Add B+1		000101		
E = 1	1	111011	000111	3
Shl EAQ	1	110110	001110	
Add B+1		000101		
E = 1	1	111011		
leave Q_0=0	1	111011	001110	
Add B		111011		
E = O	0	110110		2
Shl EAQ	1	101100	011100	
Add B+1		000101		
E = 1	1	110001		
Set Q_0= 0	1	110001		
Shl EAQ	1	100011	11010	
Add B+1		000101		
E = 1	1	101000		
leave Q_0=0	1	101000	11010	
	1	111011		
Add B	0	100011	11010	0

Q. 55 What is wrong with the following register transfer statements?

(i) $xT : AR \leftarrow AR,\ AR \leftarrow 0$

(ii) $yT : R_1 \leftarrow R_2,\ R_1 \leftarrow 0$

(iii) $zT : PC \leftarrow AR,\ PC \leftarrow PC + 1$

Ans:

(i) Cannot complement and increment the same register at the same time.

(ii) Cannot transfer two different values (R_2 and R_3) to the same register (R_1) at the same time.

(iii) Cannot transfer a new value into a register (PC) and increment the original value by one at the same time.

Q.56 (i) How many 128 x 8 RAM chips are needed to provide a memory capacity of 2048 bytes?

(ii) How many lines of the address bus must be used to access 2048 bytes of memory? How many of there lines will be common to all chips?

(iii) How many lines must be decoded for chip select? Specify the size of the decoders.

Ans:

(i) Memory capacity = 2048 bytes

2^{11} = 2048 bytes

$$= \frac{2^{11}}{2^7} = \frac{2048}{128}$$

$= 2^4 =$ 16 chips RAM

(ii) Memory capacity = 2048 bytes

address bus = 11 (2^{11} = 2048)

$128 = 2^7$ 7 lines to address each chip and 4 lines to decoder for selecting 16 chips. Thus, 7 lines out of 11 will be common to all chips.

(iii) Remaining 4 lines must be decoded for chip select the size of decoder is 4 x 16.

Q. 57 Explain the following with example

(i) Memory-reference instructions.

(ii) Input-output instructions.

(iii) Program control instruction.

Ans:

(i) Memory reference instructions

Memory-Reference Instructions

Symbol	Operation decoder	Symbolic description
AND	D_0	$AC \leftarrow AC \wedge M[AR]$
ADD	D_1	$AC \leftarrow AC + M[AR], E \leftarrow Cout$
LDA	D_2	$AC \leftarrow M[AR]$
STA	D_3	$M[AR] \leftarrow AC$
BUN	D_4	$PC \leftarrow AR$
BSA	D_5	$M[AR] \leftarrow PC, PC \leftarrow AR+1$
ISZ	D_6	$M[AR] \leftarrow M[AR]+1$
		If $M[AR]+1=0$ then $PC \leftarrow PC+1$

(ii) Input-output instructions.

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 instruction]			
	p:	$SC \leftarrow 0$	Clear SC
INP	pB_{11}:	$AC(0-7) \leftarrow INPR, FGI \leftarrow 0$	Input character
OUT	pB_{10}:	$OUTAR \leftarrow AC(0-7), FGO \leftarrow 0$	Output character
SKI	pB_9:	IF(FGI=1) then ($PC \leftarrow PC+1$)	Skip on input flag
SKO	pB_8:	IF(FGO=1) then ($PC \leftarrow PC+1$)	Skip on Output flag
ION	pB_7:	$IEN \leftarrow 1$	Interupt enable on
IOF	pB_6:	$IEN \leftarrow 0$	Interupt enable off

(iii) Program control instruction.

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

Q.58 What do you mean by effective address of data? List any four addressing modes. How is effective address calculated for them?

Ans:

Effective Address :- The effective address is defined to be the memory address obtained from the computation dictated by the given addressing mode. The effective address is the address of the operand in a computational type instruction

Addressing modes:-

(i) Direct mode

(ii) Indirect mode

(iii) Register direct & register indirect mode

(iv) Immediate mode

(v) Implicit mode

(i) Direct Mode:- In this mode. The instruction includes a memory address

Ex. LDAC 5,

Data from memory location 5 is stored in accumulator.

(ii) Indirect Mode:- The address specified in the instruction is not the address of the operand. It is the address of a memory location that contains the address of the operand.

Ex. LDAC @ 5.

First it retrieve the content of memory location 5 say 10. Then the CPU goes to location 10, reads the contents of that location and loads the data into the CPU.

(iii) Register Direct & Register indirect modes:-

Register modes work the same as direct & indirect modes discussed above, except they do not specify a memory address. Instead they specify a register.

Example: LDAC R $\Rightarrow$ Content of Reg. 'R' is copied in accumulator

LDAC (R) $\Rightarrow$ The content of Reg. 'R' gives the memory address whose content is to be copied in to accumulator.

(iv) Immediate Mode:- The operand specified in the instruction is not an address, it is the actual data to be used.

Example: LDAC # 5, It moves the data value 5 to accumulator.

(v) Implicit Mode:- It does not explicitly specify an operand. The instruction implicitly specify the operand because it always applies to a specific register.

Example: CLAC $\Rightarrow$ Clear accumulator

CMC $\Rightarrow$ Complement accumulator.

Q. 59 Write a program by using two - addressing & one-addressing format to evaluate.

$$\frac{A-B+C * (D-E)}{C + G * H}$$

Ans:

MOV	R_1, D	$R_1 \leftarrow$ M[D]
SUB	R_1, E	$R_1 \leftarrow R_1$ - M[E]
MOV	R_2 C	$R_2 \leftarrow$ M[C]
MUL	R_2, R_1	$R_2 \leftarrow R_1 * R_2$
MOV	R_3, B	$R_3 \leftarrow$ M[B]
ADD	R_3, R_2	$R_3 \leftarrow R_3 + R_2$
MOV	R_4, A	$R_4 \leftarrow$ M[A]
SUB	R_4, R_3	$R_4 \leftarrow R_4$ - R_3
MOV	R_5, G	$R_5 \leftarrow$ M[G]
MUL	R_5, H	$R_5 \leftarrow R_5 *$ M[H]
ADD	R_5, C	$R_5 \leftarrow R_5$+M[C]
DIV	R_4, R_5	$R_4 \leftarrow R_4 \div R_5$
MOV	Y, R_4	M[y] $\leftarrow R_4$

One-addressing

$$Y = \frac{A-B+C * (D-E)}{C+G * H}$$

Load	D	AC $\leftarrow$ M[D]
SUB	E	AC $\leftarrow$ AC-M[E]
MUL	C	AC $\leftarrow$ AC+M[C]
ADD	B	AC $\leftarrow$ AC+M[B]
SUB	A	AC $\leftarrow$ M[A]- AC
Store	T	M[T]- AC
Load	G	AC $\leftarrow$ M[G]
MUL	H	AC $\leftarrow$ AC* M[H]
ADD	C	AC $\leftarrow$ AC+M[C]
DIV	T	AC $\leftarrow$ M[T] $\div$ AC
Store	Y	M[y] $\leftarrow$ AC

Q.60 Define interrupt. Why priority of interrupt is required? How it is restored?

Ans:

Interrupt:- An interrupt is a signal sent by an I/O interface to CPU when it is ready to send information to the memory or receive information from the memory.

interrupt — A priority interrupt is a system that establishes a priority over the various sources to determine which condition is to be serviced first when two or more requests arrive simultaneously. Higher - priority interrupt levels are assigned to requests which, if delayed or interrupted. Devices with high speed transfers such as magnetic disks are given high priority, and slow devices such as keyboards receive low priority. When two devices interrupt the computer at the same time, the computer services the devices, with the higher priority first. The task of the interrupt system is to identify the source of the interrupt. There is also the possibility that several sources will request service simultaneously. In this case the system most also decide which device to service first, than use the priority interrupt.

Establishing the priority of simultaneous interrupts can be done by software or hardware. A polling procedure is used to identify the highest-priority source by software means. In this method there is one common branch address for all interrupts. The program that takes care of interrupts begins at the branch address and polls the interrupt sources in sequence. The order in which they are tested determines the priority of each interrupt. The highest-priority source is tested first, and if its interrupt signal is on, control branches to a service routine from this source. Otherwise, the next-lower-priority source is tested, and so on. Thus the initial service routine for all interrupts consists of a program that tests the interrupt sources in sequence and branches to one of many possible service routines. The particular service routine reached belongs to the highest-priority device among all devices that interrupted the computer. The disadvantage of the software method is that is there are many interrupts, the time required to poll them can exceed the time available to service the I/O device. In this situation a hardware priority-interrupt unit can be used to speed up the operation.

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 in figure. The interrupt request line 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 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. This signal is received by device 1 at its *PI* (priority in) input. The acknowledge signal passes on to the next device through the *PO* (priority out) output only if device 1 is not requesting an interrupt. If device 1 has a pending interrupt, it block the acknowledge signal from the next device by placing a 0 in the *PO* output. It

then proceeds to insert its own interrupt vector address (VAD) into the data bus for the CPU to use during the interrupt cycle.

A device with a 0 in its *PI* input generates a 0 in its *PO* output to inform the next-lower-priority device that the acknowledge signal has been blocked. A device that is requesting an interrupt and has 1 in its *PI* input will intercept the acknowledge signal by placing a 0 in its *PO* output. If the devices does not have pending interrupts, it transmits the acknowledge signal to the next device by placing a 1 in its *PO* output. Thus the device with *PI*=1 and *PO*=0 is the one with the highest priority that is requesting an interrupt, and this device places its VAD on the data bus. The daisy chain arrangement gives the highest priority to the device that receives the interrupt acknowledge signal from the CPU. The farther the device is from the first position, the lower is its priority.

Figure shows the internal logic that must be included with each device when connected in the daisy-chaining scheme. The device sets its RF flip-flop when it wants to interrupt the CPU. The output of the *RF* flip-flop goes through an open-collector inverter, a circuit that provides the wired logic for the common interrupt line. If *PI*=0, both *PO* and the enable line to VAD are equal to 0, irrespective of the value of *RF*. If *PI* =1 and *RF*=0, the *PO*=1 and the vector address is disabled. This condition passes the acknowledge signal to the next device through *PO*. The device is active when *PI*=1 and *RF*=1. This condition places a 0 in *PO* and enables the vector address for the data bus. It is assumed that each device has its own distinct vector address. The *RF* flip-flop is reset after a sufficient delay to ensure that the CPU has received the vector address.

Q.61 Differentiate between synchronous and asynchronous data transfer method. **(6)**

Ans:

Synchronous transmission

(i) The two circuits share a common clock frequency and bits are transmitted continously.

(ii) In long distant serial transmission, each unit is driven by a separate clock of the same frequency.

(iii) In synchronous transmission bits must be transmitted continously to keep the clock frequency in both units synchronized with each other.

Asynchronous transmission

(i) In synchronous transmission employs special bits that are inserted at both ends of the character code.

(ii) Each character consists of three parts: a start bit, the character bits, and stop bits.

(iii) Asynchrnous transmission sent only binary information.

Q. 62 Write on ALP to find square-root of a number.

Ans:

Example:- Suppose that X is the square root of number N. To find a suitable equation to be used by the computer for iteration the following manipulation is done:

$$x^2 = N$$
$$2X^2 = N + X^2$$
$$X^2 = \frac{N + X^2}{2}$$
$$\frac{N + X^2}{2}$$

Address	Machine codes	Label	Mnemonics	Operands	Comments
2600	3E, X		MVI	A,X	X is the first approximation
2602	57	BACK	MOV	D,A	
2603	2A,00,25		LHLD	2500	N in H-L.
2606	CD,06,24		CALL	2406	Division Sub-routine, N/X in L.
2609	7E		MOV	A,D	X in A.
260A	85		ADD	L	(X+N/X)
260B	6F		MOV	L,A	
260C	26,00		MVI	H,00	
260E	3E,02		MVI	H,02	
2610	CD,06,24		CLL	2406	(N/X+X)/2 in L=X_{new}
2613	7D		MOV	A,L	
2614	BA		CMP	D	Compare X_{new} with X.
2615	CA;1B,26		JZ	BELOW	Jump to BELOW, if X_{new} = 0
2618	C3,02,26		JMP	BACK	Jump to Back, if X_{new} = X.
261B	32,50,25		BELOW	STA	2550
261A	76			HLT	

Q. 63 What is page fault and how page fault is handled by memory management software?

Ans:

Page fault: —

When a program starts execution, one or more pages are transferred into main memory and the page table is set to indicate their position. The program is executed from main memory until it attempts to reference a page that is still in auxiliary memory. This condition is called page fault.

Handle:—

Page fault occurs in a virtual memory system. It signifies that the page referenced by the CPU is not in main memory. A new page is than transferred from auxiliary memory to main memory. If main memory is full it would be necessary to remove a

page from a memory block to make room for the new page. So the replacement algorithm is used.

Two of the most common replacement algorithms used are the first - in - first - out and least recently used.

Performance:—

Want an algorithm which will result in minimum number of page faults. Same page may be brought into memory several times.

Q. 64 Discuss the different mapping techniques used for cache memory. What is the need of mapping techniques?

Ans:

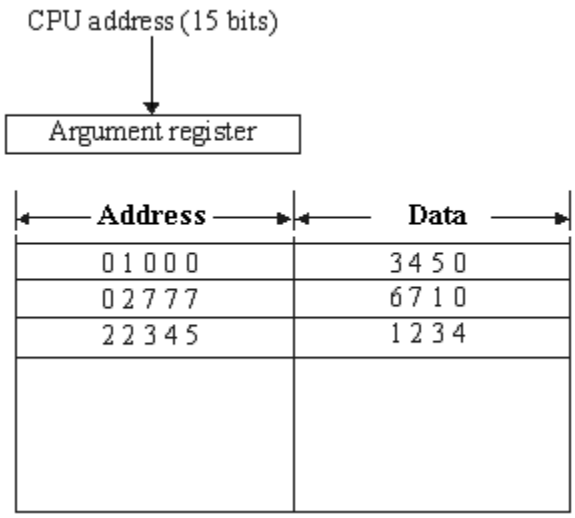
The basic characteristic of cache memory is its fast access time. The transformation of data from main memory to cache memory is referred to as a mapping process.

1) Associative mapping 2) Direct mapping
3) Set-associative mapping

Associative Mapping:-

The fastest and most flexible cache organization uses an associative memory. This 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

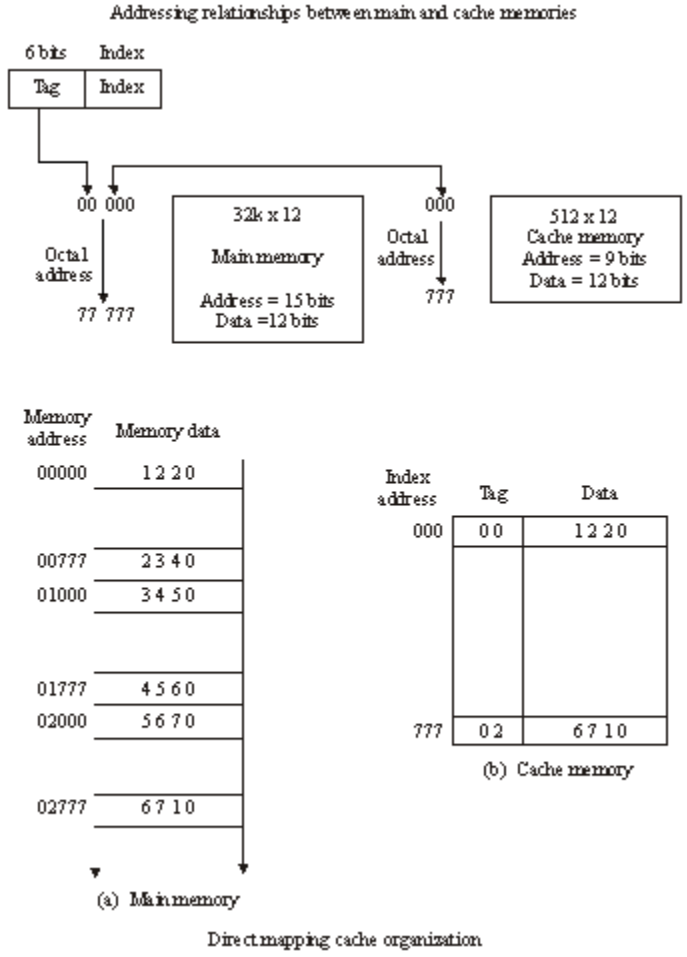


Associative mapping cache (all numbers in octal) number and its corresponding 12-bit word is shown as a four-digit octal number. A CPU address of 15 bits is placed in the argument register and the associative memory is searched for a matching address. If the address is found, the corresponding 12-bit 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 in round-robin order whenever a new word is requested from main memory. This constitutes a first-in first-out (FIFO) replacement policy.

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 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. 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 the 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.

Set-Associative Mapping:-

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 store 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

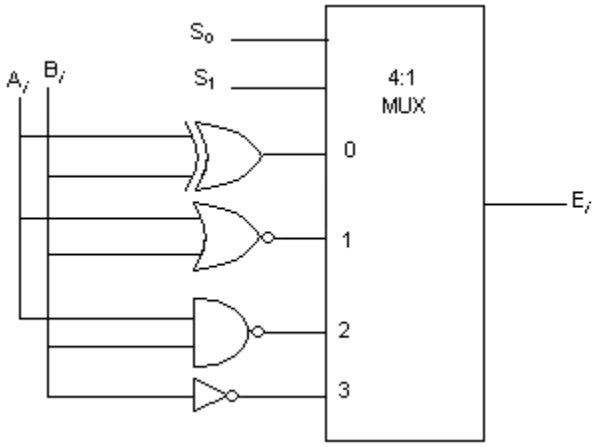
Index	Tag	Data	Tag	Data
000	0 1	3 4 5 0	0 2	5 6 7 0
777	0 2	6 7 1 0	0 0	2 3 4 0

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.

When a miss occurs in a set-associative cache and the set is full, it is necessary to replace one of the tag-data items with new value. The most common replacement algorithms used are: FIFO and LRU. The FIFO procedure selects for replacement the item that has been in the set the longest. The LRU algorithm selects for replacement the items that have 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. 65 Draw the logic diagram of a logic circuit capable of performing X - OR, NOR, NAND, complement operation of two bits A and B. During complement operation, the circuit should be capable of complementing B.

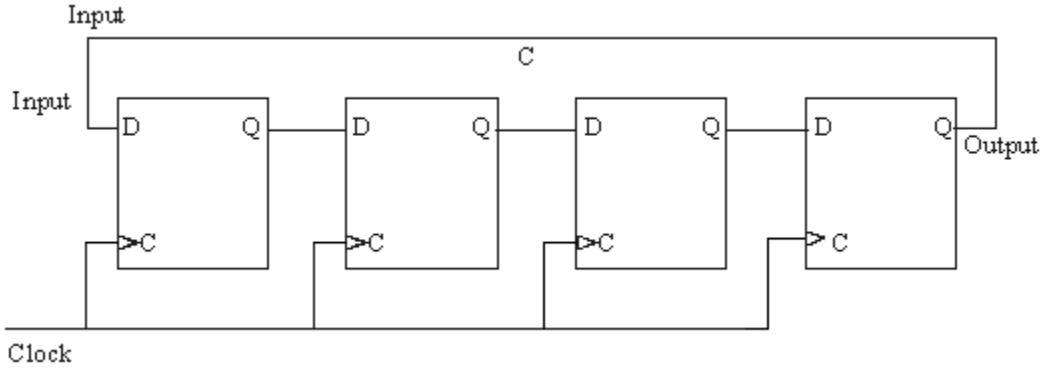
Ans:



S_1	S_0	output	operation
0	0	$E = A \oplus B$	XOR
0	1	$E = (A+B)'$	NOR
1	0	$E = (AB)'$	NAND
1	1	$E = A'$	Complement

Q. 66 Draw the logic circuit using D flip flop for implementing circular shift left operation and circular shift right operation order control input.

Ans.



Q. 67 Write short notes on any two

(i) Hardware polling

(ii) Arithmetic pipe lining

(iii) Common bus architecture

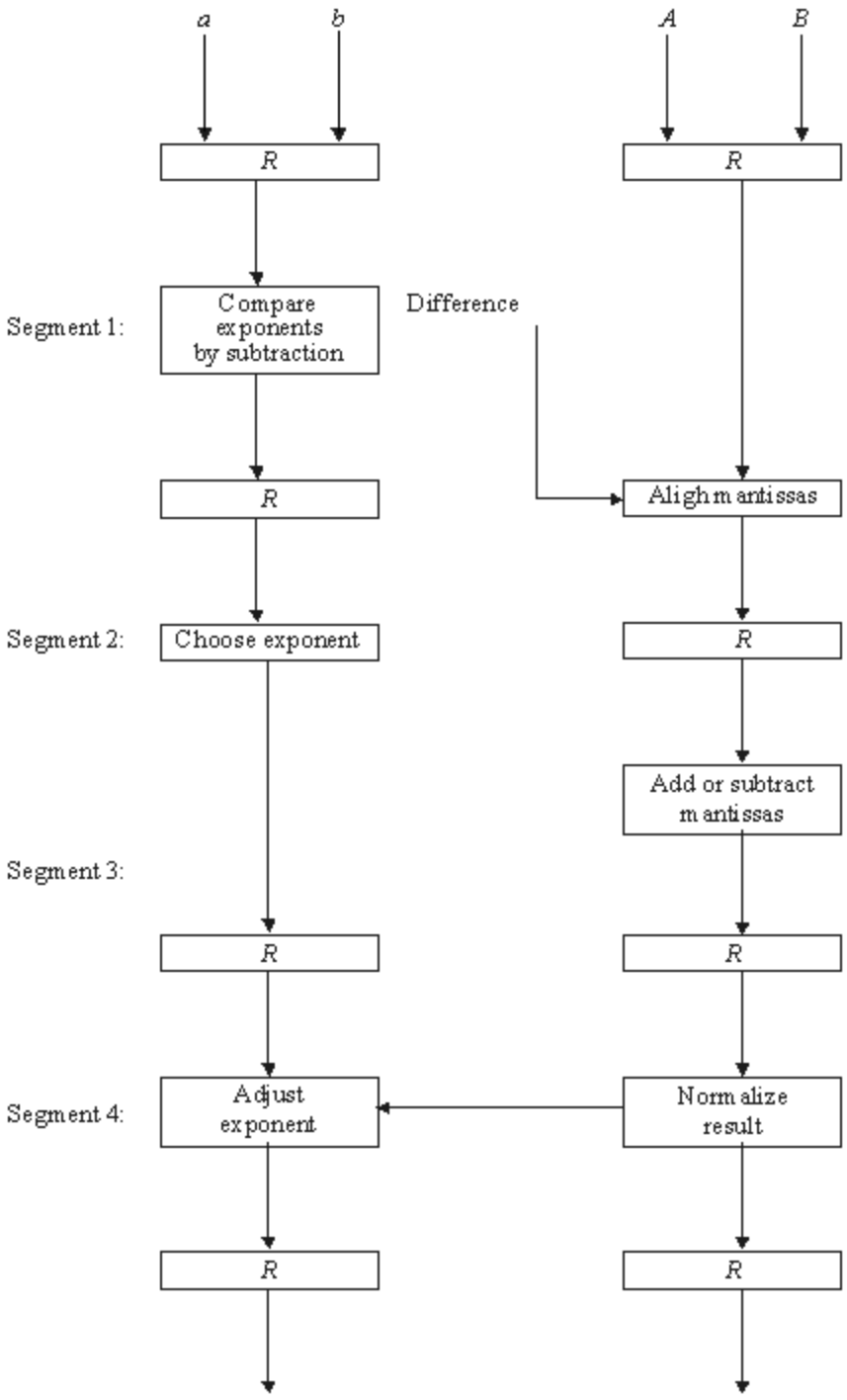
Ans:

(i) Hardware polling:-

A polling procedure is used to identify the highest-priority source by software means. In this method there is one common branch address for all interrupts. The program that takes care of interrupts begins at the branch address and polls the interrupt sources in sequence. The order in which they are tested determines the priority of each interrupt. The highest-priority source is tested first, and if its interrupt signal is on, control branches to a service routine for this source. Otherwise, the next-lower-priority source is tested, and so on. Thus the initial service routine for all interrupts consists of a program that tests the interrupt sources in sequence and branches to one of many possible service routines. The disadvantage of the software method is that if there are many interrupts, the time required to poll them can exceed the time available to service the I/O device. In this situation a hardware priority-interrupt unit can be used to speed up the operation.

A hardware priority-interrupt unit functions as an overall manager in an interrupt system environment. It accepts interrupt requests from many sources, determines which of the incoming requests has the highest priority, and issues an interrupt request to the computer based on this determination. To speed up the operation, each interrupt source has its own interrupt vector to access its own service routine directly. Thus no polling is required because all the decisions are established by the hardware priority-interrupt unit. The hardware priority function can be established by either a serial or a parallel connection of interrupt lines. The serial connection is also known as the daisy-chaining method.

(ii) Arithmetic Pipeline:-



Pipeline for floating-point addition and subtraction

Pipeline arithmetic units are usually found in very high speed computers. They are used to implement floating-point operations, multiplication of fixed-point numbers, and similar computations encountered in scientific problems.

The inputs to the floating-poing adder pipeline are two normalized floating-point binary numbers.

$X = A \times 2^a$

$Y = B \times 2^b$

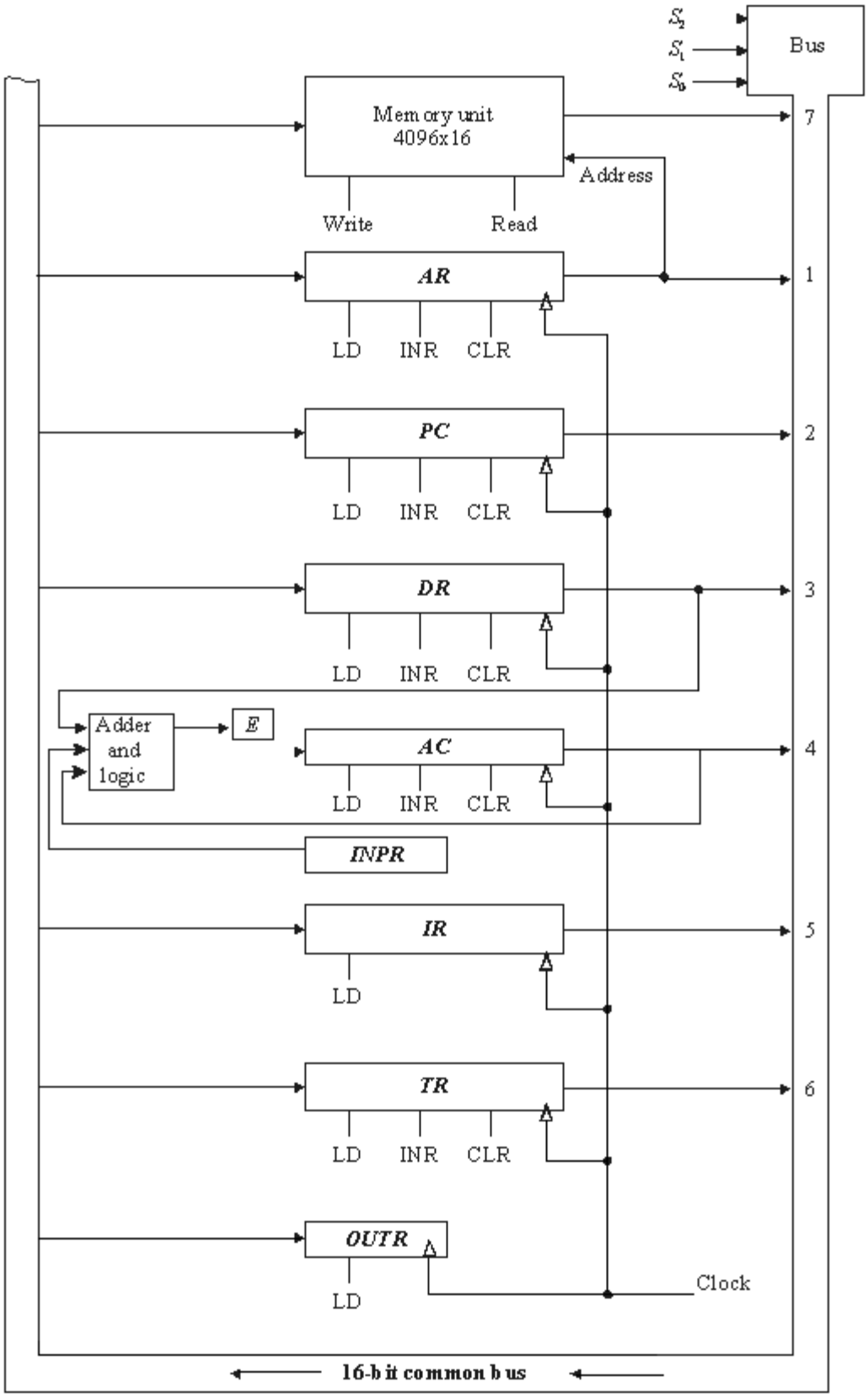
A and B are two fractions that represent the maintissas and a and b are the exponents. The floating-point addition and subtraction can be performed in four segments. The suboperations that are performed in the four segments are:

1) Compare the exponents.
2) Align the mantissas.
3) Add or subtract the mantissas.
4) The exponents are compared by subtracting them to determine their difference. The larger exponent is chosen as the exponent of the result. The exponent difference determines how many times the mantissa associated with the smaller exponent must be shifted to the right. This produces an alignment of the two mantissas. It should be noted that the shift must be designed as a combinational circuit to reduce the shift time. The two mantissas are added or subtracted in segment 3. The result is normalized in segment 4. When an overlow occurs, the mantissa of the sum or difference is shifted right and the exponent incremented by one. If an underflow occurs, the number of leading

zeros in the mantissa determines the number of left shifts in the mantissa and the number that must be subtracted from the exponent.

(iii) **Common Bus System:-**

The basic computer has eight registers, a memory unit, and a control unit .Paths must be provided to transfer information from one register to another and between memory and registers. The number of wires will be excessive if connections are made between the outputs of each register and the inputs of the other registers. A more efficient scheme for transferring information in a system with many registers is to use a common bus. The connection of the registers and memory of the basic computer to a common bus system is shown in figure.



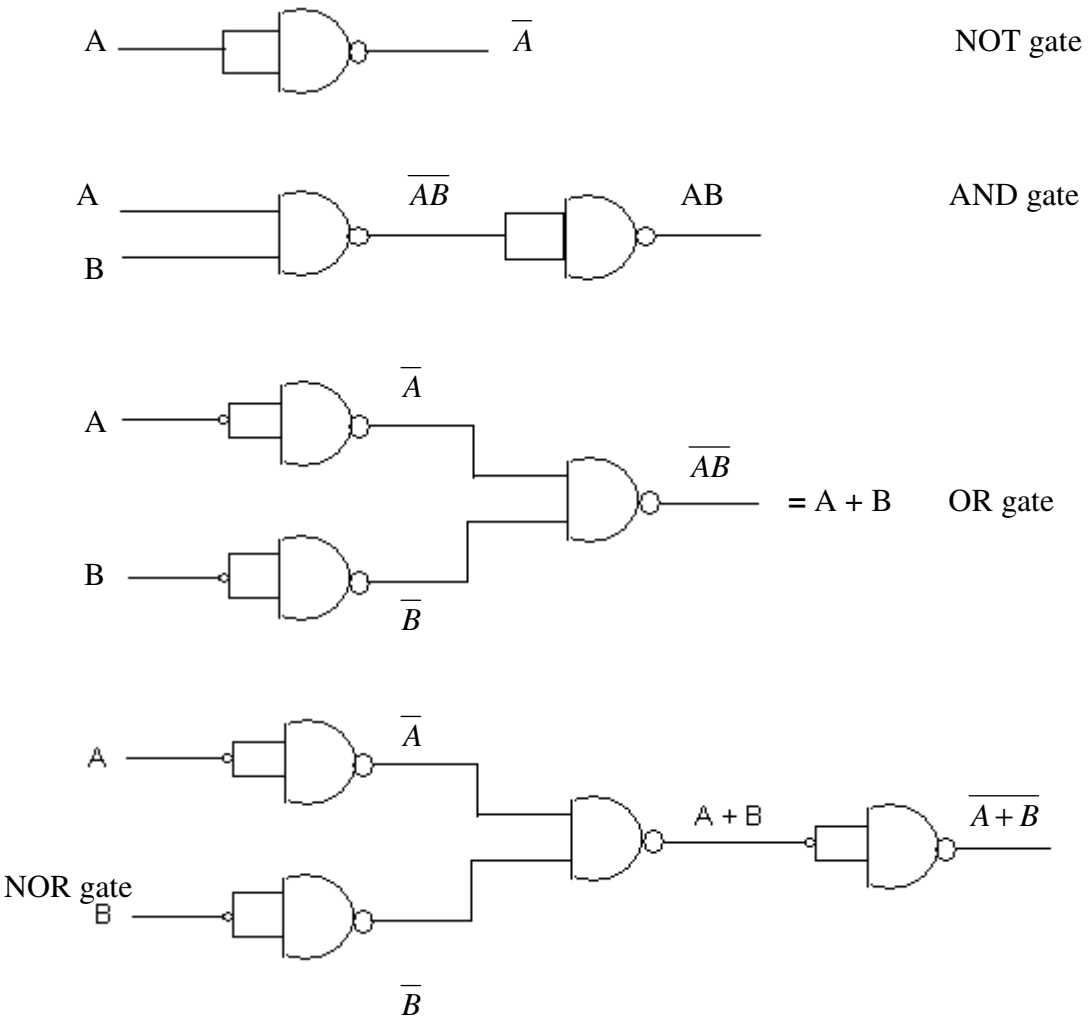
Basic computer registers connected to a common bus.

Q.68 What binary number does $5A34F_{16}$ represent? **(1)**

Ans: 0 1 01 1 0 1 0 0 0 1 1 0 1 0 0 1 1 1 1 $_2$

Q.69 Using NAND gate generate the NOT, AND, OR and NOR functions. **(2x4=8)**

Ans:

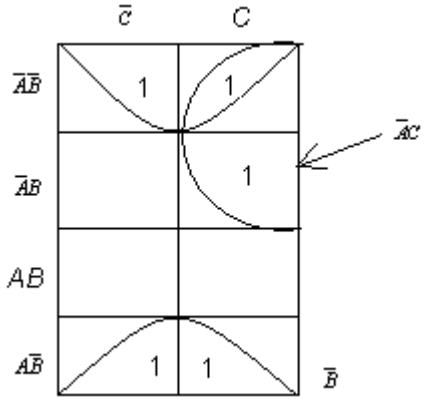


Q.70 Minimize the expression

$$X = A\overline{B}C + \overline{A}BC + \overline{AB}C + \overline{ABC} + A\overline{BC}$$

Using Karnaugh's map. **(4)**

Ans:



Four 1's is in adjacent group.
Remaining 1 is absorbed in overlapping gp – gp of four 1's produce a single variable terms $\bar{B}$. This is determined by observing that within the gp $\bar{B}$ is the only variable. That does not change from cell to cell. gp of 2 1's produce a 2 variable term $\bar{A}C$. To get minimized for the 2 terms are summed (OR ed) as

$$X = \bar{B} + \bar{A}C$$

Q.71 Subtract 1010100 – 1000011 using 2's complement. **(3)**

Ans:
X = 1 0 1 0 1 0 0
Y = 1 0 0 0 0 1 1
2's complement of Y = 0 1 1 1 1 0 1
X + Y = 1 0 0 1 0 0 0 1
Discard end carry 2 = - 10000000
X – Y = 0 0 1 0 0 0 1

Q.72 For the following memory units (specified by the number of words the number of bits per word), determine the number of address lines, input/output lines and the number of bytes that can be stored in the specified memory

(i) 64K x 8
(ii) 16M x 32
(iii) 4G x 64
(iv) 2K x 16 **(1 ½ x 4 = 6)**

Ans:
(i) 64K x 8
i/p, o/p lines = 8
Address lines = 16
Mem = 64K

(ii) 16M x 32
i/p, o/p lines = 32
Add = 24
Mem = 64M (16M x 4)

(iii) 4G x 64
i/p, o/p lines = 64
Add = 32
Mem = 32GB (4G x 8)

(iv) 2K x 16
i/p, o/p = 16
Add = 11
Mem = 4K

Q.73 What is a micro-operation? List and briefly explain the most commonly encountered arithmetic operations. **(1+4)**

Ans:
A micro-operation is an elementary operation performed with the data store in registers. The micro-operation most often encountered in digital computer are classified into four categories:

1. Register transfer micro-operations transfer binary information from register to another.
2. Arithmetic micro-operations perform arithmetic operation on numeric data stored in registers.
3. Logic micro-operations perform bit manipulation operation on numeric data stored in registers.
4. Shift micro-operations perform shift operations on data stored registers.

The basic arithmetic micro-operation are addition, subtraction increment, decrement, and shift. Arithmetic shifts are explained later in conjunction with the shift micro-operations. The arithmetic micro-operation defined by the statement

$$R3 \leftarrow R1 + R2$$

specifies an *add* micro-operation .It states that the contents of registerR1 added to the contents of register R3. Subtraction is most often implemented through complementation and addition. Instead of using the minimum operator, we can specify the subtraction by the following statement:

$$R3 \leftarrow R1 + \overline{R2} + 1$$

$\overline{R2}$ is the symbol for the 1's complement of R2. Adding 1 to the 1's complement produces the 2's complement. Adding the contents of R1to the 2's complement of R2 is equivalent to R1- R2.

Arithmetic Micro-operations

Symbolic designation	Description
$R3 \leftarrow R1 + R2$	Contents of R1 plus R2 transferred to R3
$R3 \leftarrow R1 - R2$	Contents of R1 minus R2 transferred to R3
$R2 \leftarrow \overline{R2}$	Complement the contents of R2 (1's complement)
$R2 \leftarrow \overline{R2} + 1$	2's complement the contents of R2 (negate)
$R3 \leftarrow R1 + \overline{R2} + 1$	R1 plus the 2's complement of R2 (subtraction)
$R1 \leftarrow R1 + 1$	Increment the contents of R1 by one
$R1 \leftarrow R1 - 1$	Decrement the contents of R1 by one

The increment and decrement micro-operations are symbolized by plus one and minus-one operation, respectively. These micro-operations are implemented with a combinational circuit or with or with a binary up-down counter.

Q.74 Discuss the different ways in which ROM can be programmed. **(5)**

Ans:

The required path in a ROM may be programmed in three different ways. The first, mask programming, is done by the semiconductor company during the last fabrication process of the unit. The procedure for fabricating a ROM requires that the customer fill out the truth table that he or she wishes the ROM to satisfy. The truth table may be submitted in a special form provided by the manufacturer makes the corresponding mask for the paths to produce the 1's and 0's according to the customer's truth table. This procedure is costly because the vendor charges the customer a special fee for custom masking the particular ROM. For this reason, mask programming is economical only if a large quantity of the same ROM configuration is to be ordered.

For small quantities it is more economical to use a second type of ROM called a programmable read-only memory or PROM. When ordered, PROM units contain all the fuses intact, giving all 1's in the bits of the stored words. The fuses in the PROM are blown by application of current pulses through the output terminals for each address. A blown fuse defines a binary 0 state, and an intact fuse gives a binary 1 state.

This allows users to program PROMs in their own laboratories to achieve the desired relationship between input addresses and words. Special instruments called PROM *programmers* are available commercially to facilitate this procedure. In any case, all procedures for programming ROMs are hardware procedures even through the word "programming" is used.

The hardware procedure for programming ROMs or PROMs is irreversible, and once programmed, the fixed pattern is permanent and cannot be altered. Once a bit pattern has been established, the unit must be discarded if the bit pattern is to be changed. A third type of ROM available is called *erasable* PROM or EPROM can be restructured to the initial value even through its fuses have been blown previously. When the EPROM is placed under a special ultraviolet light for a given period of time, the shortwave radiation discharges the internal gates that serve as fuses. After erasure, the EPROM returns to its initial state and can be reprogrammed to a new set of words. Certain PROMs can be erased with electrical signals instead of ultraviolet light. These PROMs are called *electrically erasable* PROM or EEPROM.

Q.75 Design a 4 bit arithmetic circuit which is capable of performing the following micro-operations:

Addition (with and without carry)

Subtraction (with and without borrow)

Increment

Decrement

and Transfer

Substantiate the circuit diagram with the help of an example. **(12)**

Ans:

The basic component of an arithmetic circuit is the parallel adder. By controlling the data inputs to the adder, it is possible to obtain different types of arithmetic operations.

The diagram of a 4-bit arithmetic circuit is shown in Fig 4. It has four full-adder circuits that constitute the 4-bit adder and four multiplexers for choosing different operations. There are two 4-bit inputs A and B and a 4-bit output D. The four inputs from A go

directly to the X inputs of the binary adder. Each of the four inputs from B are connected to the data inputs of the multiplexers. The multiplexers data inputs also receive the complement of B. The other two data inputs are connected to logic-0 and logic-1. Logic-0 is a fixed voltage value (0 volts for TTL integrated circuits) and the logic-1 signal can be generated through an inverter whose input is 0. The four multiplexers are controlled by two selection inputs, S_1 and S_0. The input carry C_{in} goes to carry input of the FA in the least significant position. The other carries are connected from one stage to the next.

The output of the binary adder is calculated from the following arithmetic sum:

$$D = A + Y + C_{in}$$

Where A is the 4-bit binary number at the X input and Y is the 4-bit binary number at the Y inputs of the binary adder. C_{in} is the input carry, which can be equal to 0 or 1. Note that the symbol + in the equation above denotes an arithmetic plus. By controlling the value of Y with the two selection inputs S and making C_{in} equal to 0 or 1, it is possible to generate the eight arithmetic micro-operations listed in Table 4.

TABLE 4: Arithmetic Circuit Function Table

Select		Input	Output	
S_1	S_0	C_{in}	Y	$D = A + Y + C_{in}$	Micro-operation
0	0	0	B	$D = A + B$	Add
0	0	1	B	$D = A + B + 1$	Add with carry
0	1	0	$\overline{B}$	$D = A + \overline{B}$	Subtractwith borrow
0	1	1	$\overline{B}$	$D = A + \overline{B} + 1$	Subtract
1	0	0	0	$D = A$	Transfer A
1	0	1	0	$D = A + 1$	Increment A
1	1	0	1	$D = A - 1$	Decrement A
1	1	1	1	$D = A$	Transfer A

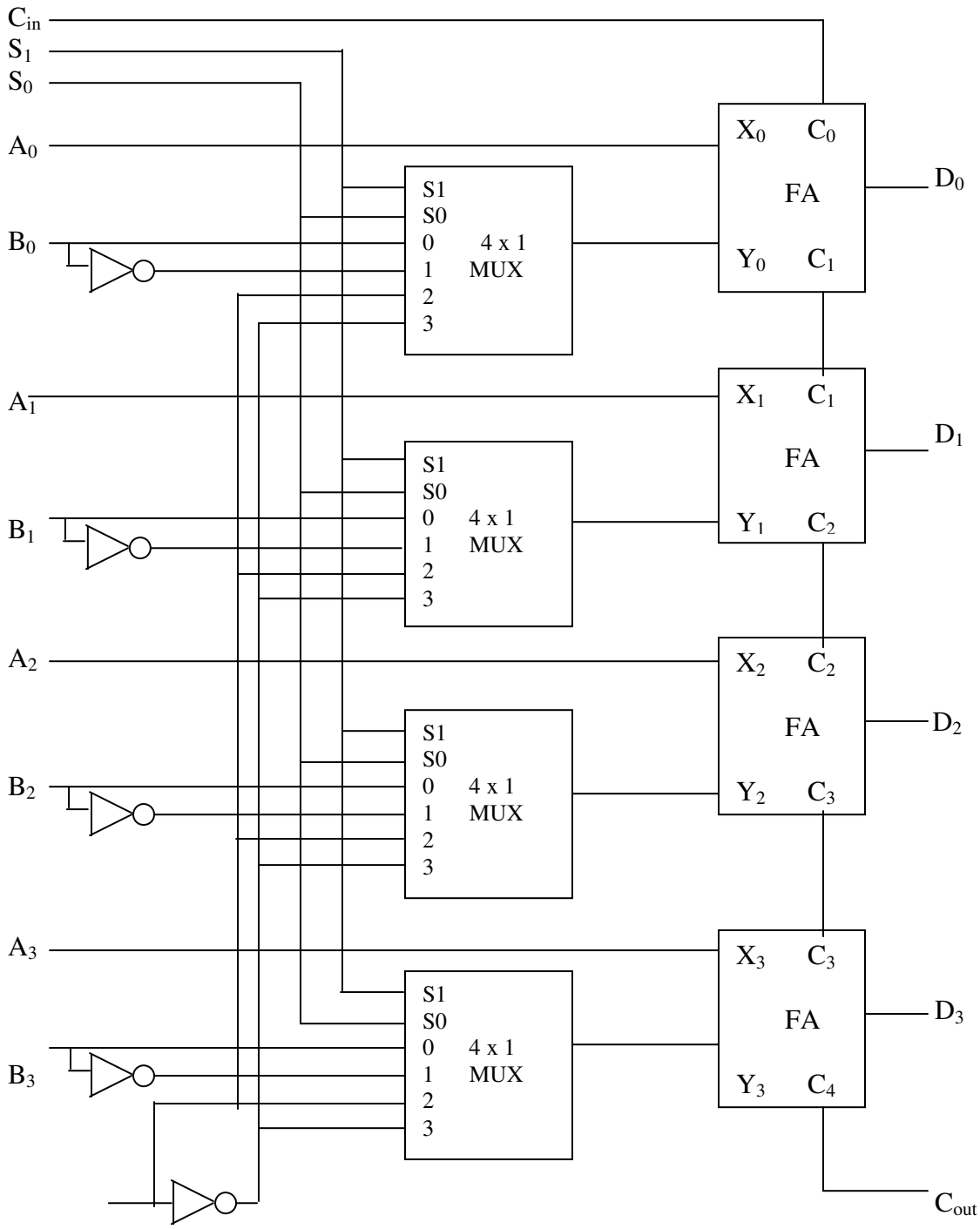


Fig 4 : 4 – bit arithmetic circuit

Q.76 Draw detailed flowchart of the instruction cycle. Indicate the conditions in which register-reference / memory-reference and input-output instructions are executed. Also include the interrupt cycle micro-operations in the flowchart. **(9)**

Ans:

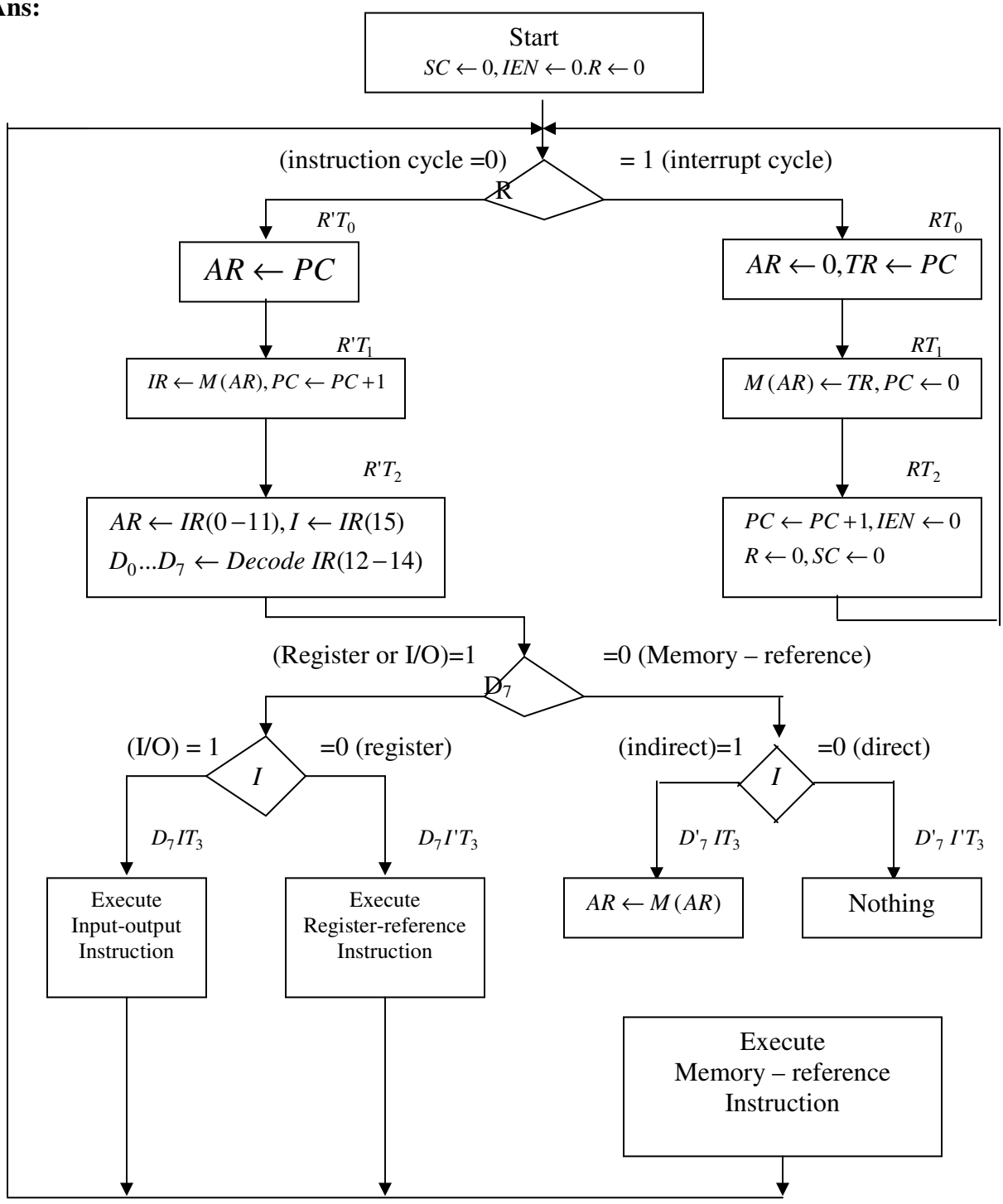


Fig 5 : Flowchart for computer operation.

Q.77 Give the cache access time as 10 ns memory access time as 100 ns and cache hit rate as 90%, calculate the effective memory access time. **(3)**

Ans: 0.90 * 10 + 0.10 * 100
= 19 ns Effective Mem Access Time.

Q.78 Explain the terms burst transfer and cycle stealing. **(5)**

Ans: Refer Section 11-6 page 418 from Morris mano (3rd Edition)
When the DMA takes control of the bus system, it communicates directly with the memory. The transfer can be made in several ways. In DMA burst transfer, a block sequence consisting of a number of memory words is transferred in a continuous burst while the DMA controller is master of the memory buses. This mode of transfer is needed for fast devices such as magnetic disks, where data transmission cannot be stopped or slowed down until an entire block is transferred. An alternative technique called cycle stealing allows the DMA controller to transfer 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 I/O transfer to "steal" one memory cycle.

Q.79 Explain how DMA controller communicates and transfers data between the peripherals devices and RAM. **(8)**

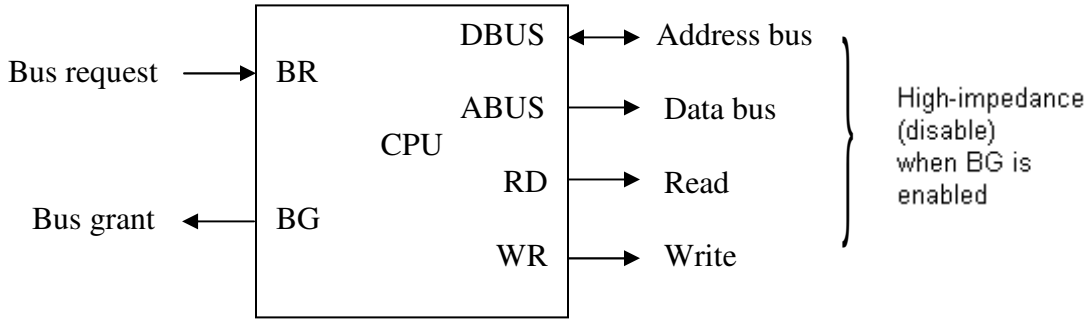
Ans:
During DMA transfer, the CPU is idle and has no control of the memory buses .A DMA controller takes over the buses to manage the transfer directly between the I/O device and memory.

The CPU may be placed in an idle state in variety of ways. One common method extensively used in microprocessor is to disable the buses through special control signals. 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. When this input is active, the CPU terminates the execution of the current instruction and places the address bus, the data bus, and the read and writes lines into a high- impedance state. The high-impedance state behaves like an open circuit, which means that the output is disconnect and does not have logic significance. 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 transfer without processor intervention. The CPU disables the bus grant takes control of the buses, and returns to its normal operation.

When the DMA takes control of the bus system, it communicates directly with the memory. The transfers can be made in several ways. In DMA burst transfer, a block sequence consisting of a number of memory words in transferred in a continuous burst while the DMA controller is master of the memory buses. This mode of transfer is needed for fast devices such as magnetic disk, where data transmission cannot be stopped or slowed down until an entire block is transferred.

The DMA controller needs the usual circuits of an interface to communicate with the CPU and I/O device. In addition, it needs an address register, a word count register, and a set of address lines. The address register and address lines.

Figure 7.1 CPU bus signals for DMA transfer.



The DMA is first initialized by the CPU. After that, the DMA starts and continues to transfer data between memory and peripheral unit until an entire block is transferred. The initialization process is essentially a program consisting of I/O instructions that include the address for selecting particular DMA registers. The CPU initializes the DMA by sending the following information through the data bus:

1. The starting address of the memory block where data are available (for read) or where data are to be stored (for write)
2. The word count, which is the number of words in the memory block.
3. Control to specify the mode of transfer such as read or write.
4. A control to start the DMA transfer.

The starting address is stored in the address register. The word count is stored in the word count register, and the control information in the control register. Once the DMA is initialized, the CPU stops communicating with the DMA unless it receives an interrupt signal or if it wants to check how many words have been transferred.

DMA Transfer

The position of the DMA controller among the other components in a computer system is illustrated in Fig 7.2. 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 and WR signal, and sends a DMA acknowledge to the peripheral device. Note that the RD and WR lines in the DMA controller are bidirectional .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 registers. When BG =1, the RD and WR are output lines from the

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.

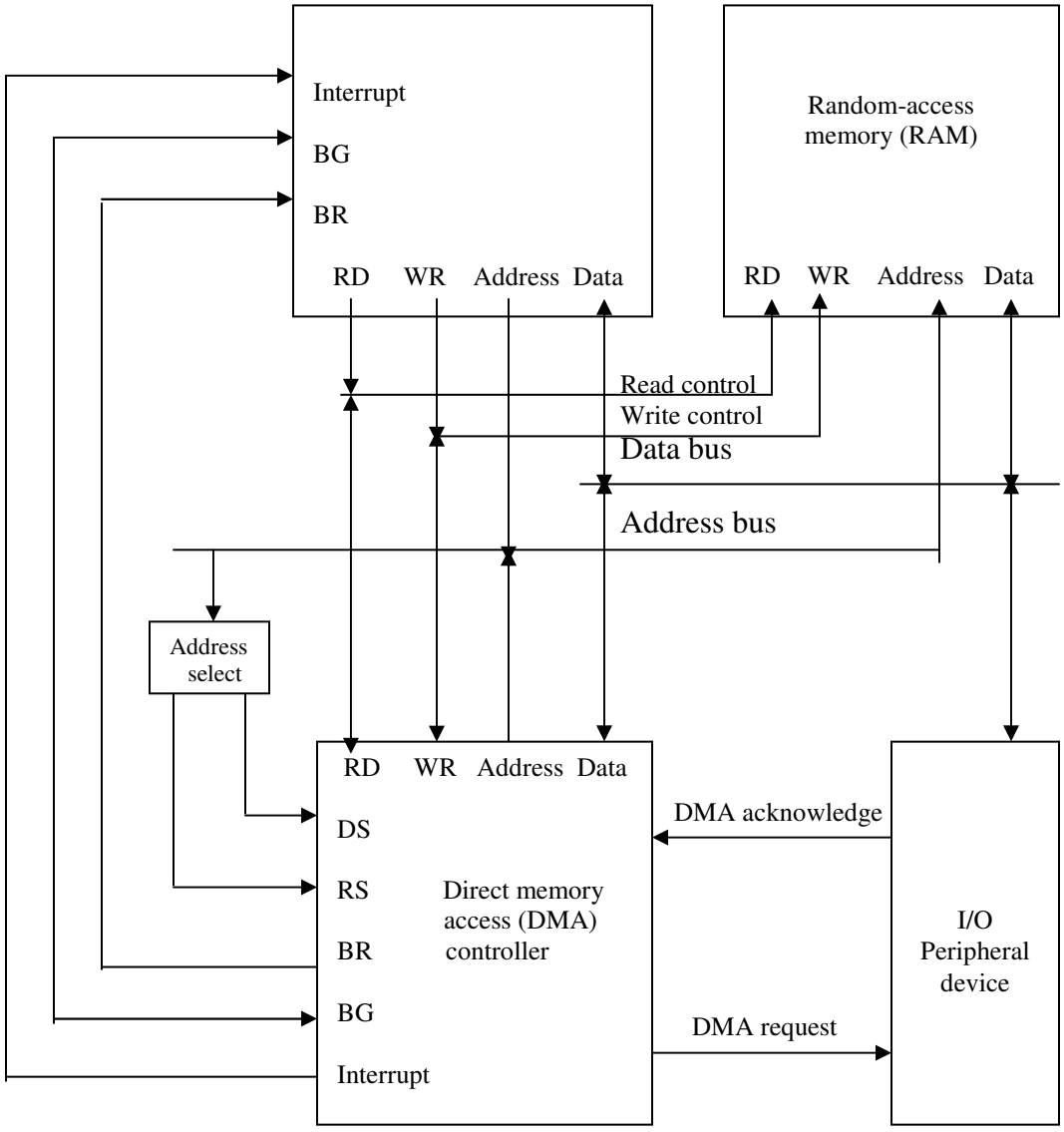


Fig 7.2 DMA transfer in a computer system.

For each word that is transferred, the DMA increment its address register 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.

If the word count register reaches zero, the DMA stops any further transfer and removes its bus request. It also informs the CPU of the termination by means of an interrupt. When the CPU responds to the interrupts, it reads the content of the words were transferred successfully. The CPU can read this register at any time to check the number of words already transferred.

A DMA controller may have more than one channel. In this case, each channel has a request and acknowledges pair of control signals which are connected to separate peripheral devices. Each channel also has its own address register and word count register within the DMA controller. A priority among the channels may be established so that channels with high priority are serviced before channels with lower priority.

Q.80 Differentiate between
(i) Isolated and Memory-Mapped I/O
(ii) Associated Mapping and Direct Mapping. **(8)**

Ans:

(i) Isolated versus Memory-Mapped I/O

Many computers use one common bus to transfer information between memory or I/O and the CPU. The distinction between a memory transfer and I/O transfer is made through separate read and write lines. The CPU specifies whether the address on the address line is for a memory word or for an interface register by enabling one of two possible read or write lines. The I/O read and memory write control lines are enabled during a memory write control lines are enabled during a memory transfer. This configuration isolates all I/O interface addresses from the addresses assigned to memory and is referred to as the isolated I/O method for assigning addresses in a common bus.

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 (for input) or I/O write (for output) control line. This informs the external components that are attached to the common bus that the address in the address lines is for an interface register and not for a memory word .On the other hand, 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. This informs the external components that the address is for a memory word and not for an I/O interface.

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 is the case in computers that employ only one set of read and write signals and do not distinguish between memory and I/O addresses. This configuration is referred to as memory-mapped I/O. The computer treats an interface register as being part of the memory system. The assigned addresses for interface registers cannot be used for memory words which reduce the memory address range available.

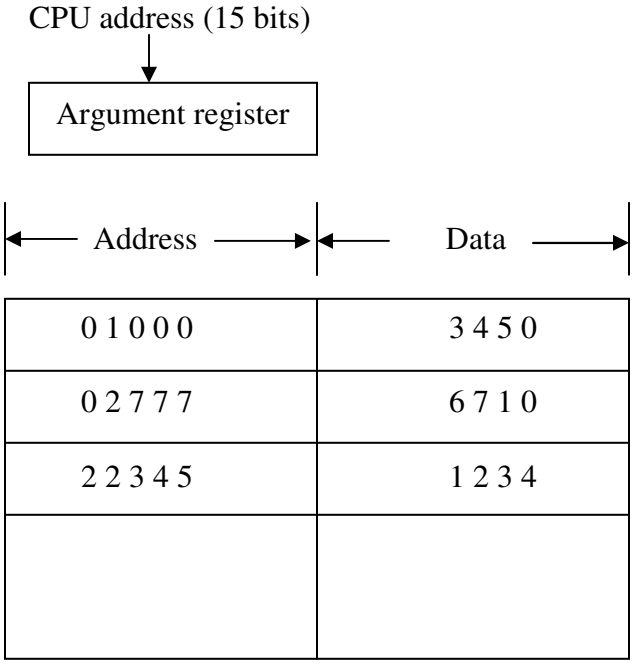
In a memory-mapped I/O organization there is no specific input or output instructions. The CPU can manipulate I/O data residing in interface registers with the same instructions that are used to manipulate memory words. Each interface is organized as a set of registers that respond to read and write requests in the normal address space. Typically, a segment of the total address space is reserved for interface registers, but in general, they can be located at any address as long as there is not also a memory word that responds to the same address.

Computers with memory-mapped I/O can use memory-type instructions to access I/O data. It allows the computer to use the same instructions for either input-output transfers or for memory transfers, 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.

(ii) **Associative Mapping**

The fastest and most flexible cache organization uses an associative memory. As in fig. 7.1 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-bites is shown as a five-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.

Fig 7.1 Associate mapping cache (all numbers in octal).



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 cell of the cache in round-robin order whenever a new word is requested from main memory. This constitutes a first-in first-out (FIFO) replacement policy.

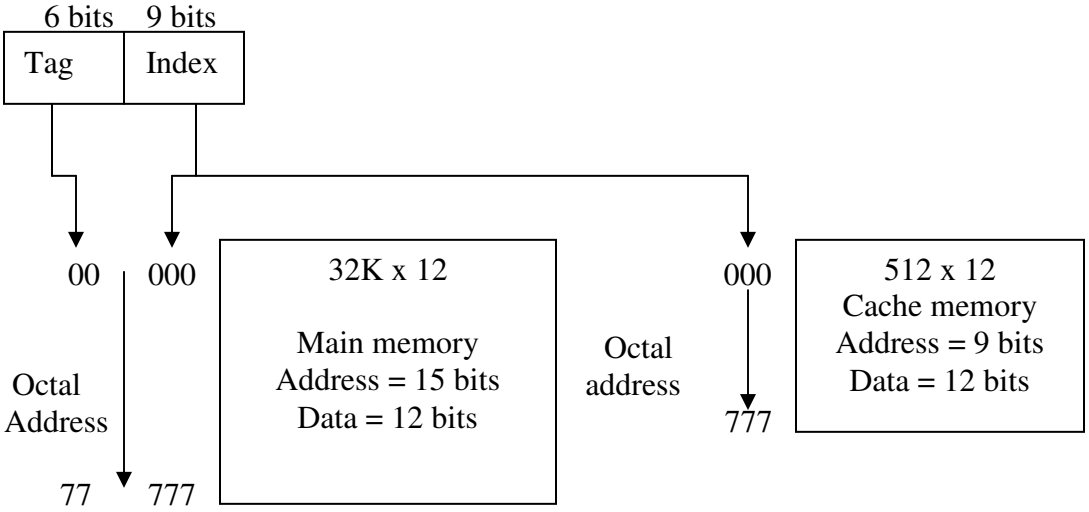
Direct Mapping

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 in Fig 7.2. The CPU address of 15 bits is divided into two fields. The nine least significant 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 $n - k$

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. 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 the cache.

Fig 7.2 Addressing relationships between main and cache memories.



Q.81 What is a page fault? What does a page fault signify? Explain the different page replacement algorithms which determine the page to be removed in case of full memory. **(6)**

Ans:

A virtual memory system is a combination of hardware and software techniques.

The memory management software system handles all the software operation for the efficient utilization of memory space. It must decide (1) Which page in main memory ought to be removed to make room for a new page, (2) When a new page is to be transferred from auxiliary memory to main memory, and (3) Where the page is to be placed in main memory. The hardware mapping mechanism and the memory management software together constitute the architecture of a virtual memory.

When a program starts execution, one or more pages are transferred into main memory and the page table is set to indicate their position. The program is executed from main memory until it attempts to reference a page that is still in auxiliary memory. This condition is called page fault. When page fault occurs, the execution of the present program is suspended until the required page is brought into main memory. Since loading a page from auxiliary memory to main memory is basically an I/O operation, the operating system assigns this task to the I/O processor. In the meantime, control is transferred to the next program in memory that is waiting to be processed in the CPU.

Later, when the memory block has been assigned and the transfer completed, the original program can resume its operation.

When a page fault occurs in a virtual memory system, it signifies that the page referenced by the CPU is not in main memory. A new page is then transferred from auxiliary memory to main memory. If main memory is full, it would be necessary to remove a page from a memory block to make room for the new page. The policy for choosing pages to remove is determined from the replacement algorithm that is used. The goal of a replacement policy is to try to remove the page least likely to be referenced in the immediate future.

Two of the most common replacement algorithms used are the first-in, first-out (FIFO) and the least recently used (LRU). The FIFO algorithm selects for replacement the page that has been in memory the longest time. Each time page is loaded into memory, its identification number is pushed into a FIFO stack. FIFO will be full whenever memory has no more empty blocks. When a new page must be loaded , the page least recently brought in is removed is easily determined because its identification number is at the top of the FIFO stack The FIFO replacement policy has the advantage of being easy to implement. It has the disadvantage that under certain circumstances pages are removed and loaded from memory too frequently.

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 loaded page as in FIFO. The LRU algorithm can be implemented by associating a counter with every page that is in main memory. When a page is referenced, its associated with all pages presently in memory are incremented by 1. The least recently used page is 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.82 What is the significance of initialising cache? How is it done? **(4)**

Ans:

One more aspect of cache organization that must be taken into consideration is the problem of initialization. The cache is initialized when power is applied to the computer or when the main memory is loaded with a complete set of programs from auxiliary memory. After initialization the cache is considered to be empty, but in effect it contains some non valid data. It is customary to include with each word in cache a valid bit to indicate whether or not the word contains valid data.

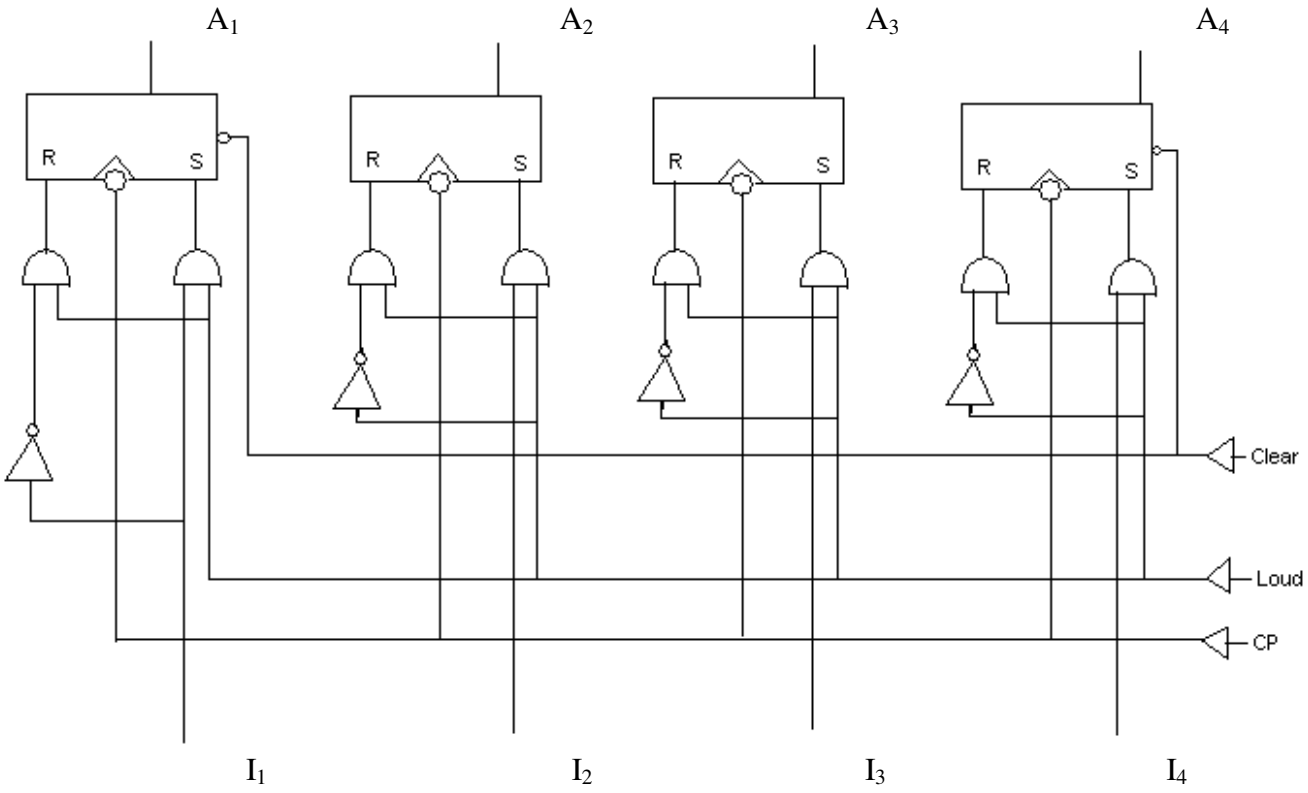
The cache is initialized by clearing all the valid bits to 0. The valid bit of a particular cache word is set to 1 the first time this word is loaded from main memory and stays set unless the cache has to be initialized again. The introduction of the valid bit means that

a word in cache is not replaced by another word unless the valid bit is set to 1 and a mismatch of tags occurs. `If the valid bit happens to be 0, the new word automatically replaces the invalid data. Thus the initialization condition has the effect of forcing misses from the cache until it fills with valid data.

Q.83 Draw the block diagram of four-bit register with parallel load. **(8)**

Ans:

Four – bit register with parallel load



Q.84 Define the following:

(i) Decoder

(ii) Multiplexers

(iii) Demultiplexers

Draw the block diagrams also. **(8)**

Ans:

(i) Decoder: A decoder is a digital function that converts binary information from one coded form to another.

(ii) Multiplexers: A Multiplexer is a digital function that receives binary information from 2^n lines and transmits information on single output line.

(iii)**Demultiplexers:** A demultiplexer is a digital function that receives information on a single line being selected is determined from the bit ambition of n selected lines.

Q.85 What is the sequence of external operations needed to store a word into memory? **(4)**

Ans: The sequence of entered operations needed to store a word into memory are:

1. Transfer the address bits of the required word into MAR.
2. Transfer the data bits of the word into the MBR.
3. Activate the write signal.

Q.86 Briefly explain fixed-point representation of numbers. What is the signed magnitude, 1's complement and 2's complement of -9? **(8)**

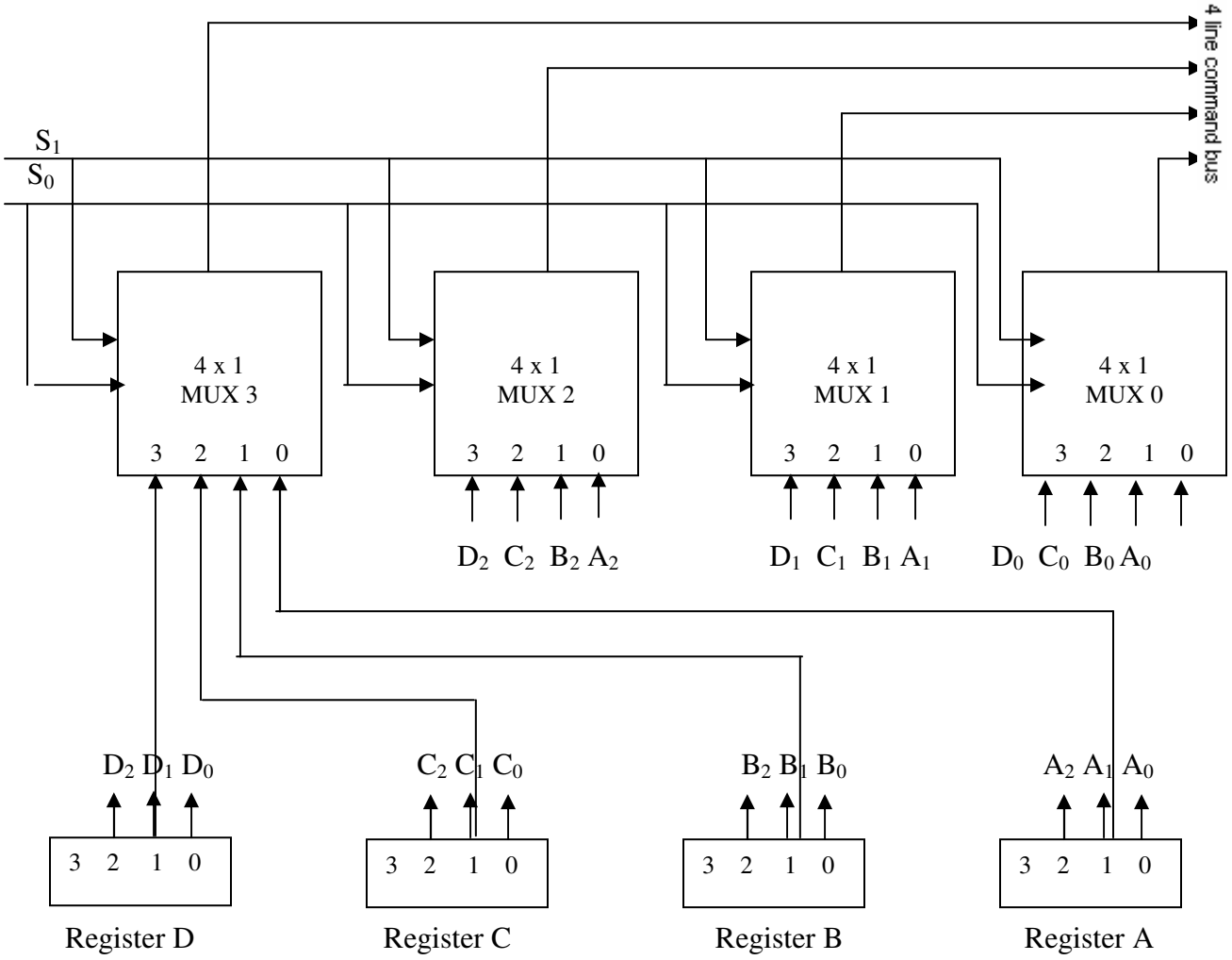
Ans: The fixed point method assumes that the decimal point is always fixed in one position. The two positions most widely used as (1) decimal point in the extreme left of the register to make the stored number a fraction & (2) a decimal point in the extreme right of the register to make the stored number an integer.

The signed magnitude representation of – 9 is obtained from +9 (0001001) by complementing only the sign bit. The signal -1's complement representation of -9 is obtained by complementing all the bits of 0001001 (+9), including the sign bit. The signed -2's complement designation is obtained by taking the 2's complement of the positive number, including its sign bit.

Q.87 Draw a diagram of bus system for four registers of 4-bits each. The bus is to be constructed with multiplexers. **(8)**

Ans:

Fig 4: Bus system for four registers.



Q.88 Explain the following:

(i) Memory-reference instruction.
(ii) Register-reference instruction
(iii) Input-output instruction. **(8)**

Ans:
(i) Refer Section 5-6 page 147 from Morris mano (3rd Edition)
(ii) Refer Section 5-5 page 145 from Morris mano (3rd Edition)
(iii) Refer Section 5-7 page 154 from Morris mano (3rd Edition)

Q.89 Write the instructions for multiplying two unsigned numbers in assembly language. **(8)**

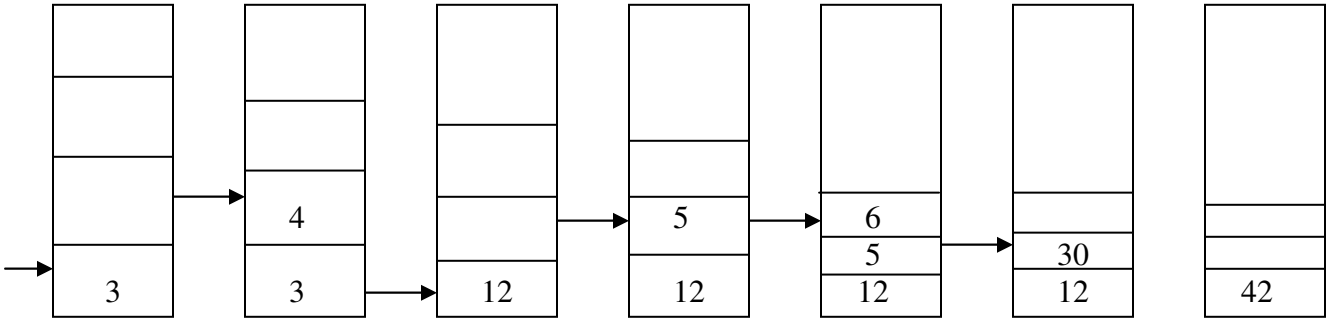
Ans:

```
L0P,    ORG 100
        CLE
```

```
                LDA Y
                CIR
                STA, Y
                SZE
                BUN ONE
                BUN ZR0

ONE,            LDA X
                ADD P
                STA P
                CLE

ZR0,            LDA X
                CIL
                STA X
                ISZ CTR
                BUN L0P
                HLT

CTR,            DEC -8
X,              HEX 000F
Y,              HEX 000B
P,              HEX 0
                END
```

Q.90 Give stack operations to evaluate 3*4+5*6. **(4)**

Ans:



3 4 * 5 6 * +

Stack operations to evaluate 3 * 4 + 5 * 6

Q.91 What is the main difference between implied and immediate modes of addressing. **(4)**

Ans:

Implied mode: In this mode the operands are specified implicitly in the definition of the instruction. For example, the instruction "complement accumulator" is an implied-

mode instruction because the operand in the accumulator register is implied 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.

Q.92 Discuss the Base Register Addressing Mode in detail. **(4)**

Ans:
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. This is similar to the indexed addressing mode except that the register is now called a base register instead of an index register. A base register is assumed to hold a base address and the address field of the instruction gives a displacement relative to this base address.Thus,
Effective address = address part of instruction + content of base register

Q.93 List in detail typical program control instructions of basic computer. **(6)**

Ans:

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

Refer Section 8-7 page 275-276 from Morris mano (3rd Edition)

Q.94 What is a cache memory? How is the performance of cache memory measured? **(6)**

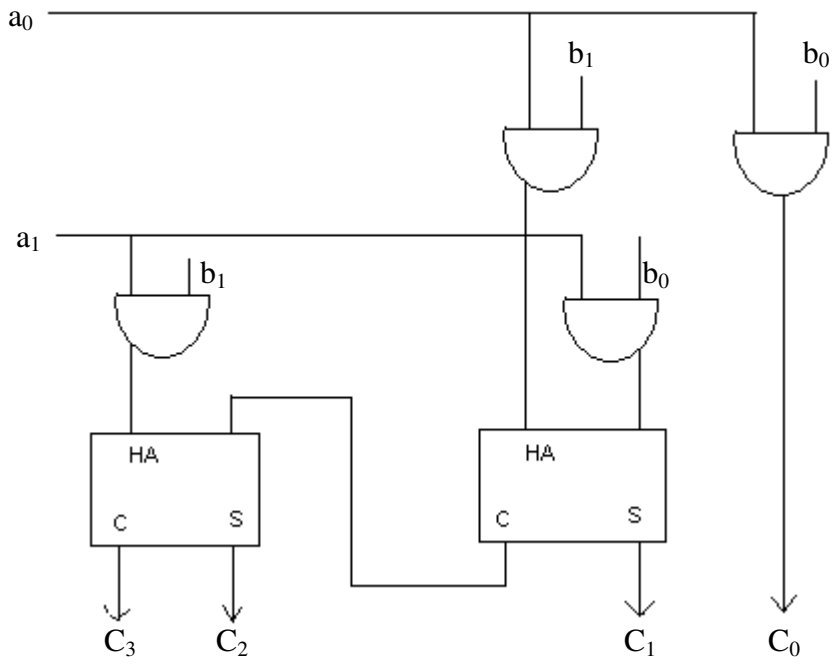
Ans:
When a program loop is executed, the CPU repeatedly refers to the set of instructions in memory that constitute the loop. Every time a given subroutine is called, its set of instructions are fetched from memory. Thus loops and subroutines tend to localize the references to memory for fetching instructions. To a lesser degree, memory references to data also tend to be localized. Table-lookup procedures repeatedly refer to that portion in memory where the table is stored. Iterative procedures refer to common memory locations and array of numbers are confined within a local portion of memory. The result of all these observations is the locality of reference property, which states that over a short interval of time, the addresses generated by a typical program refer to a few localized areas of memory repeatedly, while the remainder of memory is accessed relatively infrequently.

If the active portions of the program and data are placed in a fast small memory, the average memory access time can be reduced, thus reducing the total execution time of the program. Such a fast small memory is referred to as a *cache memory*. It is placed between the CPU and main memory. The cache memory access time is less than the access time of main memory by a factor of 5 to 10. The cache is the fastest component in the memory hierarchy and approaches the speed of CPU components.

The fundamental idea of cache organization is that by keeping the most frequently accessed instructions and data in the fast cache memory, the average memory access time of a computer system can be improved considerably by use of cache.
The performance of cache memory is frequently measured in terms of a quantity called *hit ratio.* When the CPU refers to memory and finds the word in cache, it is said to produce a *hit.* If the word is not found in cache, it is in main memory and it counts as a *miss.* The ratio of the number of hits divided by the total CPU references to memory (hits plus misses) is the hit ratio. The hit ratio is best measured experimentally by running representative programs in the computer and measuring the number of hits and misses during a given interval of time. Hit ratios of 0.9 and higher have been reported. This high ratio verifies the validity of the locality of reference property.

Q.95 Design an array multiplier that multiplies two 2-bit numbers. Use AND gates and half adders. **(8)**

Ans: Refer page 349 from Morris mano (3rd Edition)



2 by 2 binary array multiplier

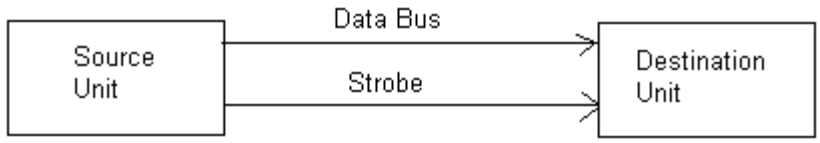
Q.96 Derive an algorithm in flowchart form to multiply two binary numbers in signed-2's complement representation. **(8)**

Ans: Refer fig 10-8 from page 347 from Morris mano (3rd Edition)

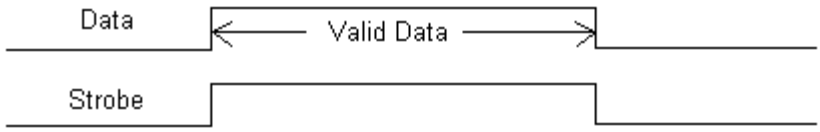
Q.97 What is strobe control? Draw a block diagram for source-initiated strobe for data transfer. **(8)**

Ans:
The strobe control method of asynchronous data transfer employs a single control line to time each transfer. The strobe may be activated by either the source or the destination unit.

Source-initiated strobe for data transfer



(a) Block Diagram



(b) Timing Diagram

Q.98 Write short notes on UART. **(4)**

Ans:
The transmitter accepts an 8-bit character from the computer 4 proceeds to send a serial 11-bit message into the printer line. The receiver accepts a serial 11-bit message from the keyboard line and forwards the 8-bit character code to computer. Integrated circuits are available which are specifically designed to provide the interface between computer and teletype or any other similar interactive terminal. Such circuit is called asynchronous communication interface or universal asynchronous receiver-transmitter (UART).

Q.99 State and verify the two DeMorgan's theorems by means of truth tables. **(7)**

Ans: DeMorgan's Law:

1) $\overline{AB} = \overline{A} + \overline{B}$

2) $\overline{A + B} = \overline{A} + \overline{B}$

1)

AB	A.B	$\overline{A.B}$	$\overline{A}$	$\overline{B}$	$\overline{A} + \overline{B}$
0 0	0	1	1	1	1
0 1	0	1	1	0	1
1 0	0	1	0	1	1
1 0	1	0	0	0	0

2)

AB	A+B	$\overline{A + B}$	$\overline{A}$	$\overline{B}$	$\overline{A}.\overline{B}$
0 0	0	1	1	1	1
0 1	1	0	1	0	0
1 0	1	0	0	1	0
1 1	1	0	0	0	0

Q.100 Obtain the truth table for the function F = AB' + B'C + A'C **(7)**

Ans:

A B C	$\overline{B}$	$A\overline{B}$	$\overline{B}C$	$\overline{A}$	$\overline{A}C$	$A\overline{B} + \overline{B}C$	$A\overline{B} + \overline{B}C + \overline{A}C$
0 0 0	1	0	0	1	0	0	0
0 0 1	1	0	1	1	1	1	1
0 1 0	0	0	0	1	0	0	0
0 1 1	0	0	0	1	1	0	1
1 0 0	1	1	0	0	0	1	1
1 0 1	1	1	1	0	0	1	1
1 1 0	0	0	0	0	0	0	0
1 1 1	0	0	0	0	0	0	0

Q.101 Obtain the simplified Boolean functions of the full-adder in sum of products form and draw the logic diagram using NAND gates. **(7)**

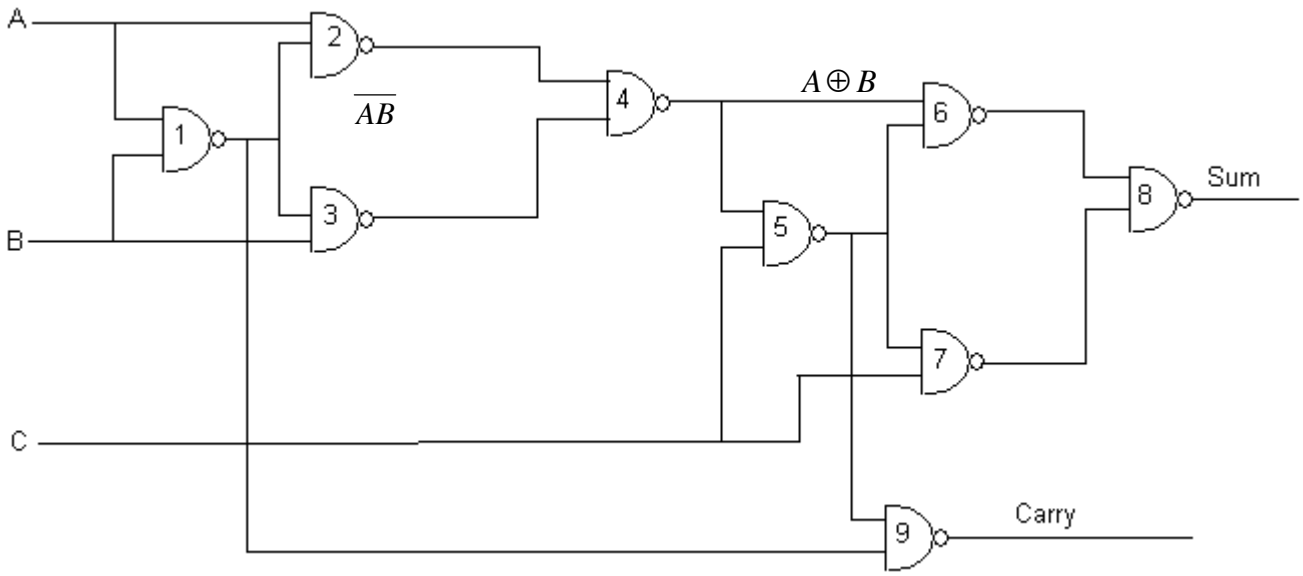
Ans:

$$\begin{aligned} A \oplus B &= \overline{A}B + A\overline{B} \\ &= \overline{\overline{A}B + A\overline{B}} \\ &= \overline{\overline{\overline{A}B}.\overline{A\overline{B}}} \\ &= \overline{A.\overline{\overline{A}B}.\overline{A\overline{B}}.B} = \text{X} \end{aligned}$$

This is realized by NAND gates 1, 2, 3 and 4 in Fig.3. Similarly, gates 5, 6, 7, 8 realize $X \oplus C = Sum$. Such a realization of exclusive OR function requires minimum number of NAND gates.

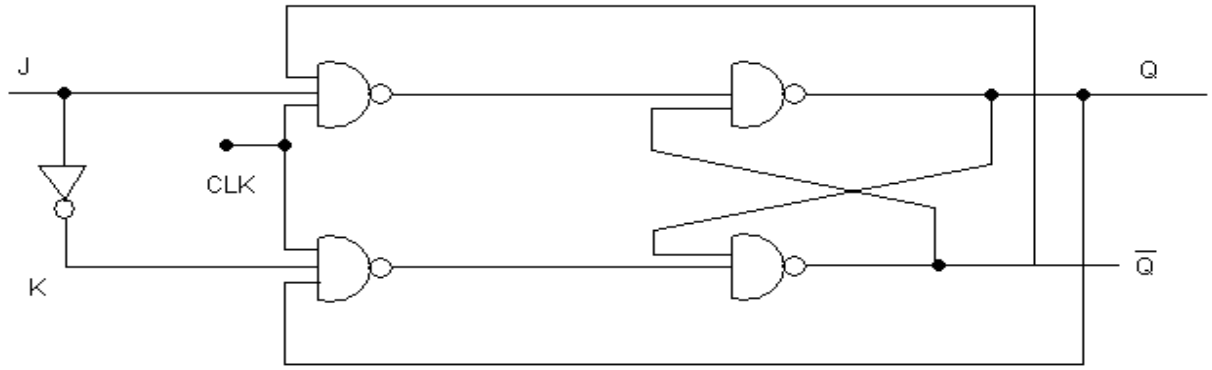
$$\begin{aligned} \text{Carry} &= \text{AB} + \text{BC} + \text{AC} \\ &= \text{AB} + \text{BC} + \text{AB} + \text{AC} \\ &= \text{B (A + C)} + \text{A (B + C)} \\ &= \text{B (A +} \overline{A}C\text{)} + \text{A (B +} \overline{B}C\text{)} \\ &= \text{AB +} \overline{A}BC + A\overline{B}C \\ &= \overline{AB + (A \oplus B).C} \\ &= \overline{\overline{AB + (A \oplus B).C}} \\ &= \overline{\overline{AB}.\overline{(A \oplus B).C}} \end{aligned}$$

Since $\overline{AB}$ and $\overline{(A \oplus B).C}$ are already available, carry can be implemented by an additional NAND gate no.9.



Q.102 Show that a JK flip-flop can be converted to a D flip-flop with an inverter between the J & K inputs. **(7)**

Ans:



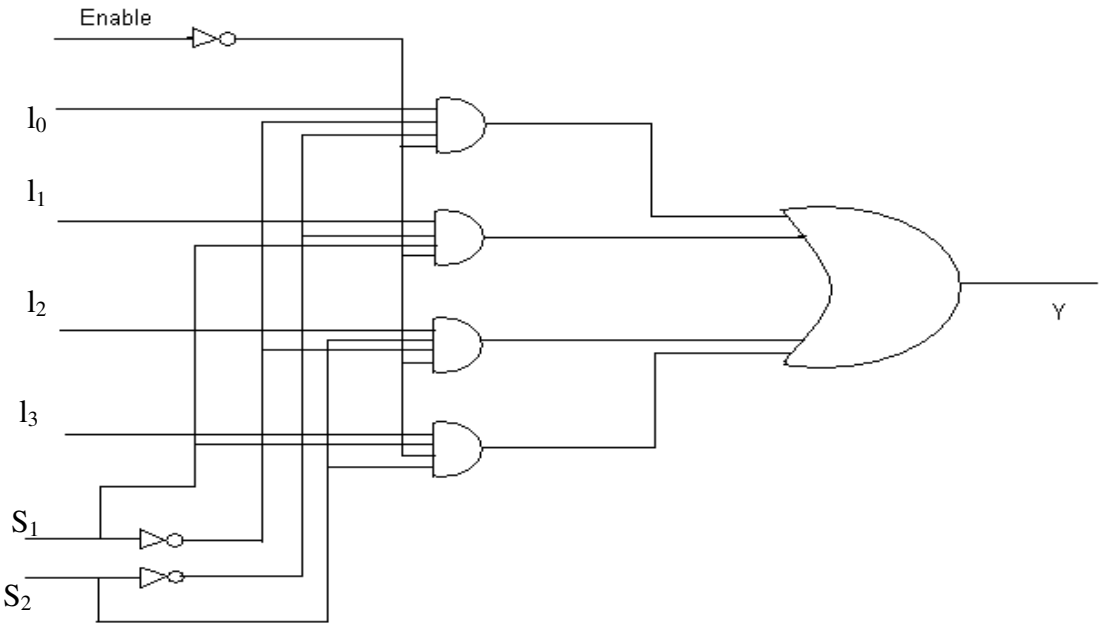
Initial Q = 1

$\overline{Q}$ = 0

CLK	J	Q	$\overline{Q}$
1	0	0	1
1	1	1	0

Q.103 Draw the logic diagram of a 4 by 1 multiplexer with an enable input. **(7)**

Ans: 4 : 1 MUX



Q.104 What are the two instructions needed in the basic computer to set the E register to 1? **(2)**

Ans:

1. Branch Instruction
2. Executing instruction

Q.105 What is the difference between an immediate, 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? **(7)**

Ans:

Immediate → directly supplying the operand value eg. MVI A 05 : 05H is stored in register A → 0 Mem Ref.

Direct → directly giving memory address of operand. Eg. LDA (addr) : contents of memory location (addr) are stored in register A. 1 Mem Ref.

Indirect → giving the address of location where address of operand is stored 2 Mem Ref.

8085 has "register indirect" instructions which fetches data from memory location whose address is presented in (BC). (DE) or (HL) register pairs. It requires one memory reference.

Q.106 List the differences between a subroutine call and an interrupt. **(5)**

Ans:

Subroutine call: A call subroutine instruction consists of an operation code together with an address that specifies the beginning of the subroutine. The instruction is executed by performing two operations:

(1) the address of the next instruction available in the program counter (the return address) is stored in a temporary location so the subroutine knows where to return, and

(2) control is transferred to the beginning of the subroutine. The last instruction of every subroutine, commonly called return from subroutine**,** transfers the return address from the temporary location into the program counter. This results in a transfer of program control to the instruction whose address was originally stored in the temporary location.

Program Interrupt: The concept of program interrupt is used to handle a variety of problems which arise out of normal program sequence. Program interrupt refers to the transfer of program control from a currently running program to another service program as a result of an external or internal generated request. Control returns to the original program after the service program is executed.

The interrupt procedure is, in principle, quite similar to a subroutine call except for three variations:

1. The interrupt is usually initiated by an internal or external signal rather than from the execution of an instruction
2. The address of the interrupt service program is determined by the hardware rather than from the address field of an instruction.

An interrupt procedure usually stores all the information necessary to define the state of the CPU rather than storing only the program counter.

Q.107 Convert the following arithmetic expressions into reverse polish notation:
(i) A + B + C
(ii) A * B/C + D.
Show the intermediate steps.

Ans:
i) A + B + C → AB+ + C
A B+ C +

ii) A*B/C + D = (A*B/C) + D →
(ABC/*) + D →
ABC / * D +

Q.108 What is the condition that must exist in an index-mode instruction to make it the same as a register indirect-mode instruction? **(2)**

Ans:
Index – Mode Instruction: In this mode the content of an index register is added to the address part of the instruction to obtain the effective address. The index register is special CPU register that contains an index value. The address field of the instruction defines the beginning address. The distance between the beginning address and the address of the operand is the index value stored in the index register. Any operand in the array can be accessed with the same instruction provided the index register contains the correct index value.

Register Indirect – Mode: In this mode the instruction specifies a register in the CPU whose contents give the address of the operand in memory. In other words, the selected register contents the address of the operand rather than the operand itself. If the memory address is obtained by adding address part of instruction to the contents of the index register (as in index-mode instruction) and is stored in a CPU register, then a register-indirect-mode instruction can be used to access the operand. A reference to the register is then equivalent to specifying a memory address. The advantage of a register – indirect mode instruction is that the address field of the instruction uses fewer bits to select a register than would have been required to specify a memory address directly. This approach combines the register – indirect and index-mode instructions.

Q.109 Describe the different categories of instructions available in the instruction set of the basic computer. **(6)**

Ans:

The basic computer has three different instruction code formats. The operation part of the instruction contains three bits; the meaning of the remaining thirteen bits depends on the operation code encountered.

A memory – reference instruction uses the last 12 bits to specify an address and the first bit to specify the mode as illustrated in Fig 1(a).

A register – reference instruction, shown in Fig 1(b), specifies an operation on or a test of the AC or E register. An operand from memory is not needed; therefore, the last 12 bits are used to specify the operation or test to be executed. A register – reference instruction is recognized by the operation code 111 with 0 in the first bit of the instruction.

Similarly, an input – output instruction does not need a reference to memory and is recognized by the operation code 111 with a 1 in the first bit of the instruction. The remaining 12 bits are used to specify the type of input – output operation or test performed. Note that the first bit of the instruction code is not used as a mode bit when the last 12 bits are not used to designate an address. The input-output instruction format is shown in Fig 1(c).

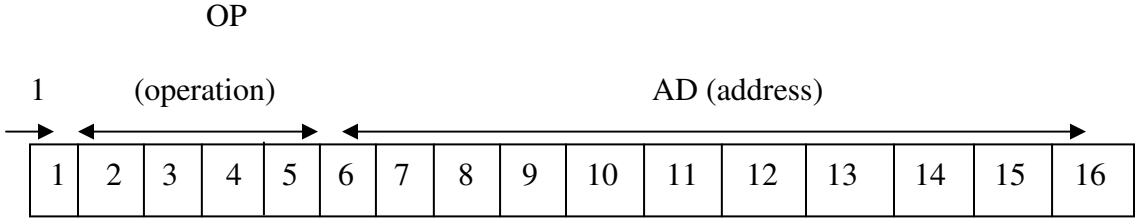


Fig 1(a) Memory – reference instruction

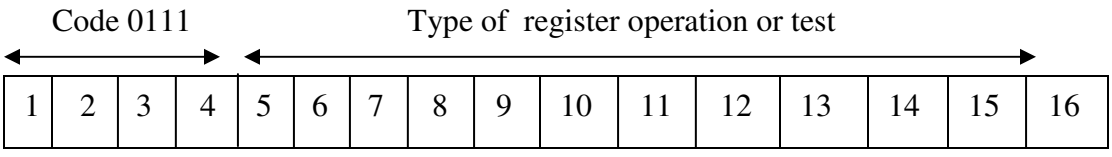


Fig 1(b) Register – reference instruction

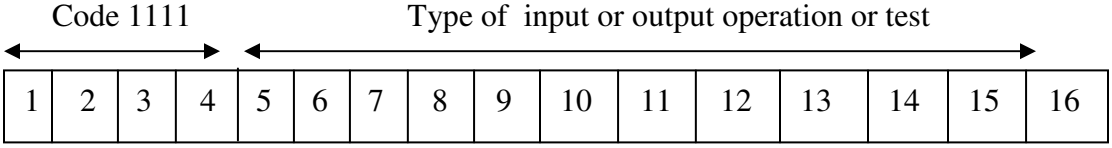


Fig 1(c) Input - output instruction

Q.110 List four alternatives for achieving a conditional branch operation in a control memory. **(7)**

Ans:

A computer that employs a microprogrammed control unit will have two separate memories: a main memory and a control memory. The main memory is available to the user for storing his programs. The contents of main memory may alter when the program is executed and every time the program is changed, the user's program in memory consists of machine instructions and data. The control memory holds a fixed microprogram that cannot be altered by the occasional user. The micoprogram consists of microinstructions that specify various internal control signals for execution of register micro – operations. Each machine instruction initiates a series of microinstructions in control memory. These microinstructions generate the micro – operations to

(1) fetch the instruction from main memory;
(2) evaluate the effective address of the operand;
(3) execute the operation specified by the instruction; and
(4) return control to the beginning of the fetch cycle to repeat the sequence again for the next instruction.

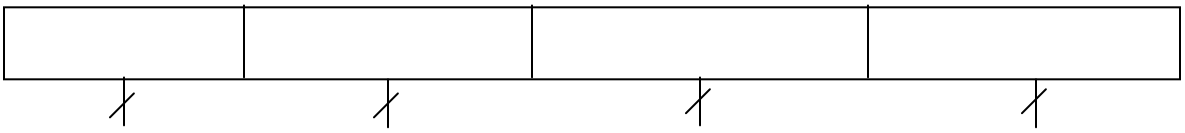
Mnemonic	**Branch condition**	**Tested condition**
BZ	Branch if zero	Z = 1
BNZ	Branch if not zero	Z = 0
BC	Branch if carry	C = 1
BNC	Branch if no carry	C = 0
BP	Branch if plus	S = 0
BM	Branch if minus	S = 1
BV	Branch if overflow	V = 1
BNV	Branch if no overflow	V = 0
Unsigned compare conditions (A – B)		
BHI	Branch if higher	A > B
BHE	Branch if higher or equal	A ≥ B
BLO	Branch if lower	A < B
BLOE	Branch if lower or equal	A ≤ B
BE	Branch if equal	A = B
BNE	Branch if not equal	A ≠ B

Signed compare conditions (A – B)		
BGT	Branch if greater than	A > B
BGE	Branch if greater or equal	A ≥ B
BLT	Branch if less than	A < B
BLE	Branch if less or equal	A ≤ B
BE	Branch if equal	A = B
BNE	Branch if not equal	A ≠ B

Q.111 Show how a microprogrammed control unit works. List three advantages over conventional control unit. **(7)**

Ans:
Microprogrammed Control unit uses a fast memory (called Control ROM) to store microinstructions. Each microinstruction generates control signals to cause one or more micro-operations. A set of microinstructions make a microprogram, which corresponds to an instruction, and is used to FETCH, DECODE and execute the instructions.

Data in a microinstruction of the Control ROM may be organized as shown below



To execute an instruction, following steps are involved

1. Requisite control signals are turned ON
2. If the jump condition is FALSE, next microinstruction in sequence is executed
3. If jump condition is TRUE, microprogram jumps to address specified in the last field.

Q.112 Describe either source-initiated or destination initiated data transfer using handshaking protocol. What advantage does handshaking provide over strobe based data transfer? **(7)**

Ans: Refer page 395-397 from Morris mano (3rd Edition)

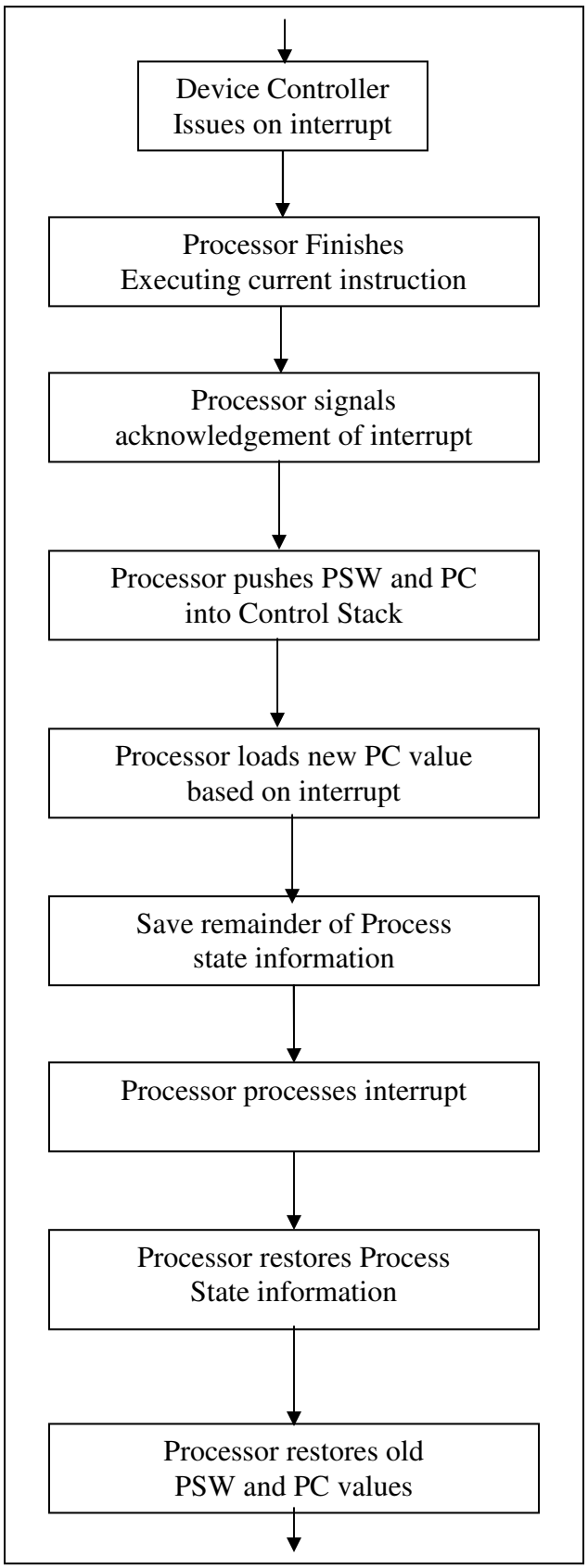
Q.113 Why does I/O interrupt make more efficient use of the CPU? **(7)**

Ans: I/O Interface:
Input – output (I/O) interface provides a method for transferring binary information between internal storage, such as memory and CPU registers, and external I/O devices. Peripherals connected on – line to a computer need special communication links for interfacing them with the central processor. The purpose of the interface and the communication link is to resolve the differences that exist between the central computer and each peripheral. The major differences are:

1. Peripherals are electromechanical devices and their manner of operation is different from the operation of the CPU and memory which are electronic devices.
2. The data transfer rate of peripherals is much slower then the transfer rate in the central computer.
3. The operation of the peripherals must be synchronized with the operation of the CPU and memory unit.
4. Data formats in peripherals differ from the word format in the central processor.
5. The operation of each peripherals must be controlled so as not to disturb the operation of the central computer and other peripherals connected to the system. The I/O interrupts the CPU only when it requires to transfer data, thus making an efficient use of CPU time.

Q.114 Explain the sequence of operations that take place in an interrupt driven I/O transfer. **(8)**

Ans:



Q.115 40 numbers are stored in memory starting from location START. Write a program in assembly language to count number of positive elements, negative elements and number of zeros. **(14)**

Ans:

```
lxi h, (START+44)h
mvi m,00h

lxi h, (START+49)h
mvi m,00h

lxi h, (START+53)h
mvi m,00h

lxi h,STARTh
mvi c,28h
mvi a,00h

loop:
dcr c
cmp c
jz ending
cmp m
jz zero
jn pos
jp neg

neg:
push h
lxi h, (START+44)h
inr m
pop h
inx h
jmp loop

pos:
push h
lxi h, (START+49)h
inr m
pop h
inx h
jmp loop

zero:
push h
lxi h, (START+53)h
inr m
```

```
pop h
inx h
jmp loop

ending:
hlt
```

Q.116 Give the differences between Demultiplexers and multiplexers. **(6)**

Ans:
Demultiplexers: A decoder with one or more enable inputs can function as demultiplexer. A demultiplexer is a digital function that receives information on a single line and transmits this information in one of $_2$n possible output lines.

Multiplexers: A multiplexer is a digital function that receives binary information 2n lines and transmits information on a single output line.

Q.117 Differentiate between serial and parallel transfer of information. **(3)**

Ans:
Parallel Transfer: Information transfer from one register to another can be performed either in parallel or in serial. Parallel transfer is simultaneous transfer of all bits from the source register to the destination register and is accomplished during on clock pulse.

Serial Transfer: For serial transfer, both the source and destination registers are shift-registers. The information is transferred one bit at a time by shifting the bits out of the source register into destination register.

Q.118 Define a bus. **(3)**

Ans:
Bus: A group of wires through which binary information is transferred among registers is called a bus.

Q.119 How is ring counter useful in timing sequence in digital computers? **(4)**

Ans:
A ring counter is a circular shift register with only one flip-flop being sent at any particular time, all others are cleared. The single bit is shifted from one flip-flop to the other to produce the sequence of timing signals.

Q.120 How many references to memory are required to bring an operand into a processor register? **(4)**

Ans:

No. of references to memory for obtaining the operand in to a processor register depends on the type of the instruction.

For immediate address instruction –One i.e for fetching operand part of the instruction.

For direct address instruction –Two

For indirect address instruction –Three

In all the cases, the first memory reference is that of fetching operand part of the instruction.

Q.121 Define the following:
(i) Logical shift (ii) Circular shift. **(6)**

Ans:
Refer Section 4-6 page 114 from Morris mano (3rd Edition)

Q.122 Define the following:

(i) Fetch Cycle. (ii) Indirect Cycle. **(8)**

Ans:
i) Fetch Cycle: An instruction is read from memory during the instruction fetch cycle. The fetch cycle is recognized by variable c_0. The four timing signals that occur during the cycle initiate the sequence of micro operations for the fetch cycle. The address, which is in PC, is transferred into MAR. The memory reads the instruction and places it in MBR. At the same time, the program counter is incremented by one to prepare it for the address of the next instruction.

ii) Indirect Cycle: The indirect cycle is recognized by the variable c_1. During this cycle, control reads the memory word where the address of the operand is to be found.

Q.123 Write down the three types of CPU organization with the help of examples. **(8)**

Ans:
Refer page 258,259 from Morris mano (3rd Edition)

Q.124 Explain the concept of program interrupt with suitable examples. **(8)**

Ans:
Refer page 281,282 from Morris mano (3rd Edition)

Q.125 What are pseudo-instructions and define some common symbols used in assembly language.

(8)

Ans:

Refer page 182 from Morris mano (3rd Edition)

Some common symbols used in assembly language are as follows:
AND
ADD
STA
CLA
INP
OUT etc.

Q.126 Explain Binary counters and Binary counter sequence in detail. **(8)**

Ans: Refer Section 2-6 page 55-56 from Morris mano (3rd Edition)

Q.127 Write down an algorithm for adding and subtracting numbers in signed – 2's complement representation. **(8)**

Ans:

Subtract	ADD
↓	↓
Minuend in AC	Augend in AC
Subtrahend in BR	Addend in BR
↓	↓
AC ← AC + BR + 1	AC ← AC + BR
.V ← overflow	V ← overflow
↓	↓
END	END

Q.128 Discuss how Booth's algorithm treats positive and negative multipliers uniformly. **(8)**

Ans:
Refer page 345-346 from Morris mano (3rd Edition)

Q.129 Give the difference between priority interrupt and daisy chain priority interrupt. **(8)**

Ans:
Refer page 410-411 from Morris mano (3rd Edition)

Q.130 Define the following:
(i) Control command. (ii) Test command.
(iii) Data-output command. (iv) Data-input command. **(8)**

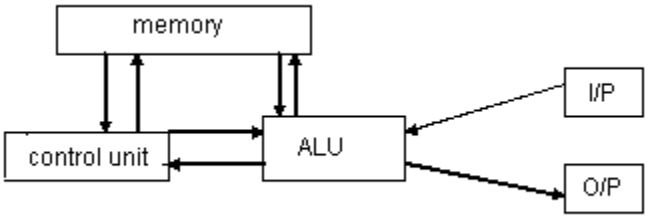
Ans:
Refer page 389 from Morris mano (3rd Edition)

Q.131 Explain Von Neumann Architecture. What are its drawbacks ? **(3+3)**

Ans:

The von Neumann architecture allows instructions and data to be mixed and stored in the same memory module and the contents of this memory are addressable by location only. The execution occurs in a sequential fashion.

The Von Neumann machine had five basic parts:- (i) Memory (ii) ALU (iii) Program control unit (iv) Input equipment & (v) output equipment.



Disadvantages:- The circuit speed can't be increased indefinitely. The most ironic aspect of the quest for even increasing speed is that most of the transistors are idle all the time. The traditional structure of a computer with a single CPU that issues sequential requests over a bus to a memory that responds to one request at a time has became known as the von-Neumann bottle neck.Modern computer predominantly follow the vonNeuman architecture, but use some elements of Harvard architecture .

Q.132 What is meant by Addressing Mode? Explain at least five different Addressing Modes with an example. **(2+6)**

Ans:

When a processor accesses memory, to either read or work data, it must specify the memory address it needs to access. An assembly language instruction may use one of several addressing modes to generate this address. These one –

(i) Direct mode
(ii) Indirect mode
(iii) Register direct & register indirect mode
(iv) Immediate mode
(v) Implicit mode
(vi) Relative mode
(vii) Index mode & Base address mode.

Direct Mode:-In this mode,the instruction includes a memory address
Ex. LDAC 5,
Data from memory location 5 is stored in accumulator.

Indirect Mode:-The address specified in the instruction is not the address of the operand. It is the address of a memory location that contains the address of the operand.

Ex. LDAC @ 5.

First it retrieve the content of memory location 5 say 10. Then the CPU goes to location 10, reads the contents of that location and loads the data into the CPU.

Register Direct & Register indirect modes:-

Register modes work the same as direct & indirect modes discussed above, except they do not specify a memory address. Instead they specify a register.

Example: LDAC R $\Rightarrow$ Content of Reg. 'R' is copied in accumulator

LDAC (R) $\Rightarrow$ The content of Reg. 'R' gives the memory address whose content is to be copied in to accumulator.

Immediate Mode:- The operand specified in the instruction is not an address, it is the actual data to be used.

Example:LDAC # 5, It moves the datavalue 5 to accumulator.

Implicit Mode:- It does not explicitely specify an operand.The instruction implicitely specify the operand because it always applies to a specific register.

Example: CLAC $\Rightarrow$ Clear accumulator

CMC $\Rightarrow$ Complement accumulator.

Q.133 Discuss the general characteristics of memory systems. What is the use of virtual memory and discuss its concept? **(2+2+6)**

Ans:

Memory is required in a computer system to store program and datas. The memory is of different types primary memory and secondary memory.

ROM, RAM, EPROM are belongs to primary memory. Where as the hard disk, magnetic tape, magnetic dram, floppy disk memories are known as secondary memory.

Primary memories are also learn as main memory. The memory unit that communicates directly with the CPU is called main memory. The devices that provides back up storage are called auxiliary memory / secondary memory.

The time required to access a memory unit for reading & writing is depends on the technology on which the memory unit is built. Primary memories are more factor than secondary memory. But they are costlier than secondary memory. To minimize the

cost and to improve the performance a large auxiliary memory is used along with a small quantity of main memory. When programs or data required by CPU is not available in main memory. Then the I/O processor brings those information's from auxiliary memory.

Virtual memory is a concept used in some large computer systems that permits the user to construct programs as though a large memory space were available equal to the totality of auxiliary memory. Each address that is referenced by the CPU goes through an address mapping from the so called virtual address to a physical address in main memory. Virtual memory is used to give programmer the illusion that they have a large memory at their disposal, even though the computer actually has a relatively small main memory. Virtual memory system provides a mechanism for translating program generated addresses in to correct main memory locations. This is done dynamically. The translation or mapping is handled automatically by the hardware by means of mapping table.

Q.134 How many 256×8 ROM chips are required to produce a memory capacity of 4000 bytes? How many address lines are required to access the 4000 bytes? How many of these addresses will be common to all these chips? **(1+1+2)**

Ans:

4000 bytes ≈ 4K bytes = $2^2\ 2^{10}$ bytes = 2^{12} bytes.

ROM size = 256 byte = 2^8 bytes.

No. of ROM chip req. = $\frac{2^{12}}{2^8} = 2^4 = 16.$

No. of add. Lines required to access 4 K bytes = 12.

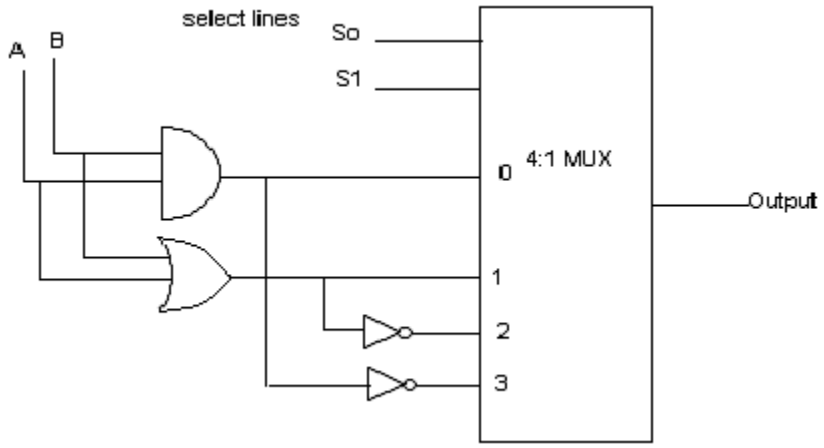
256 No. of addresses will be common to all these chips.

Q.135 Draw the block diagram of Arithmetic Logic Unit (ALU) to perform the following operations. (use MUX) (Assume the length of the data).

(i) AND (ii) OR

(iii) NOR (iv) NAND **(8)**

Ans:



Q.136 Show the function table for above design and explain. **(6)**

Ans:

Function Table

S_1	S_0	*O/p*	*Function realized*
0	0	$A \wedge B$	AND
0	1	$A \vee B$	OR
1	0	$\overline{A \vee B}$	NOR
1	1	$\overline{A \wedge B}$	NAND

Q. 137 What is meant by pipelining? Why do we require instruction pipelining? Explain its working procedure. Discuss the pipeline performance measures. **(8)**

Ans:

Pipe lining is a technique of decomposing a sequential process into sub-operations, with each sub process being executed in a special dedicated segment that operates competently with all other segments. It can be visualized as a collection of processing segments through which binary information flows. Each segment performs partial processing dictated by the way the task is partitioned. The result obtained from the computation in each segment is transferred to the next segment in the pipeline. The final result is obtained after the data have passed through all segments. The name

'Pipeline' implies a flow of information analogous to an industrial assembly line. Its characteristic is that several computations can be in progress in distinct segments at the same time.

An instruction pipeline operates on a stream of instruction by overlapping the fetch, decode & execute phases of the instruction cycle. The pipe line technique provides a factor operation over a pinely serial sequence even through the maximum theoretical speed is never fully achieved.

Working of Instructional Pipelining:

An instructional pipeline reads consecutive instructions from memory while previous instructions are being executed in other segments. This causes the instruction fetch & execute phases to overlap and perform simultaneous operations. When branch instruction is encountered, pipeline must be emptied and all the instructions that have been read from memory after the branch instruction must be discarded.

Instruction pipeline may be divided on to four segments such as-

1. Fetch the instruction from memory.
2. Decode instruction & calculate effective address.
3. Fetch operand from memory.
4. Execute instruction

Pipeline performance measure is in terms of time taken in executing a program. If a non-pipe line unit that performs a given task and takes a time equal to 't_n' to complete. The speed up of a pipe line processing over an equivalent non-pipe line processing is defined by the ratio: $S = \frac{nt_n}{(K + n - 1)t_p}$

Where K = No. of segments in pipe line.

T_p = Time taken by each segment to process a sub-operation.

n = No. of tasks.

Q.138 What are the different conflicts that will arise in pipeline (elaborate)? How do you remove the conflicts? **(3+3)**

Ans:

There are three major difficulties that causes the instruction pipe line to deviate from its normal operation.

1. Resource conflicts.
2. Data dependency
3. Branch difficulties.

Resource conflicts : It is caused by access to memory by two segments at the same time. Most of these conflicts can be resolved by using separate instruction & data memories.

Data Dependency: This conflicts arises when an instruction depends on the result of a previous instruction but this result is not yet available.

Branch difficulties : It arises from branch and other instructions that change the value of PC.

Address dependency : It may occur when an operand address can not be calculated because the information needed by the addressing mode is not available. For example, an instruction with register indirect mode cannot proceed to fetch the operand, if the previous instruction is loading the address into the register. So the operand access to memory must be delayed until the required address is available.

Handling data dependency :-

The method used for resolving data conflicts are

(i) Hardware Inter lock : It detects instructions whose source operands are destinations of instruction father up in the pipe line. Detention of this situation causes the instruction whose source is not available to be delayed by sufficient lock cycles to resolve the conflict.

(ii) Operand forwarding : It was special hardware to detect a conflict and then avoid it by routing the data through special paths between pipeline segments. For example, instead of transferring as ALU result into a destination register, the hardware checks the destination operand and it if is needed as a source in the next instruction, it passes the result directly into the ALU, by passing the register file.

(iii) Delayed Load : The responsibility for solving data conflicts problems is given to compiler that translates the high level programming language into a machine language program. The compiler for such computers is designed to detect a data conflict and re-order the instructions by inserting no-operation instructions.

Similarly for handling **branch address conflicts,** The following methods are used

(1) Pre-fetch target instruction.

(2) Branch target buffer

(3) Loop buffer

(4) Branch Prediction

(5) Delayed branch-Employed in most RISC processes.

Q.139 Explain how addition and subtraction of floating point operations are carried out with an example and show the algorithm / flow chart. **(4+4)**

Ans:

Floating point addition and subtraction are carried out in following steps.

(i) Check for zeros

(ii) Align the mantissas

(iii) Add or subtract the mantissas

(iv) Normalize the result.

Example:

Let one operand is in AC $\rightarrow 0.23 \times 10^2 = A \times 10^a$

The other operand is in BR $\rightarrow 0.15 \times 10^1 = B \times 10^b$

Ist Step:-

Check for zero.

AC is not zero

BR is also not zero

It AC = 0. Than the answer is BR in case of addition & is equal to BR with opposite sign in case of subtraction.

If BR = 0, Than the answer is AC for both addition & subtraction.

Step-2:- Allign the mantissas:-

This is done in order to make the exponent part of both operands same before addition or subtraction.

If a < b, than shift right 'A' and increment a till a = b.

If a > b, shift right 'B' and increment b till a = b.

In this case a = 2, b = 1, then a > b.

Shr B and b = b+1 gives B = 0.015, b = 2. so now a = b.

Step-3 Add or sub the mantissas.

Now A = 0.23 B = 0.015

For addition A = A+B = 0.245. For subtraction A = A-B = 0.215

Step-4:- Normalise the result.

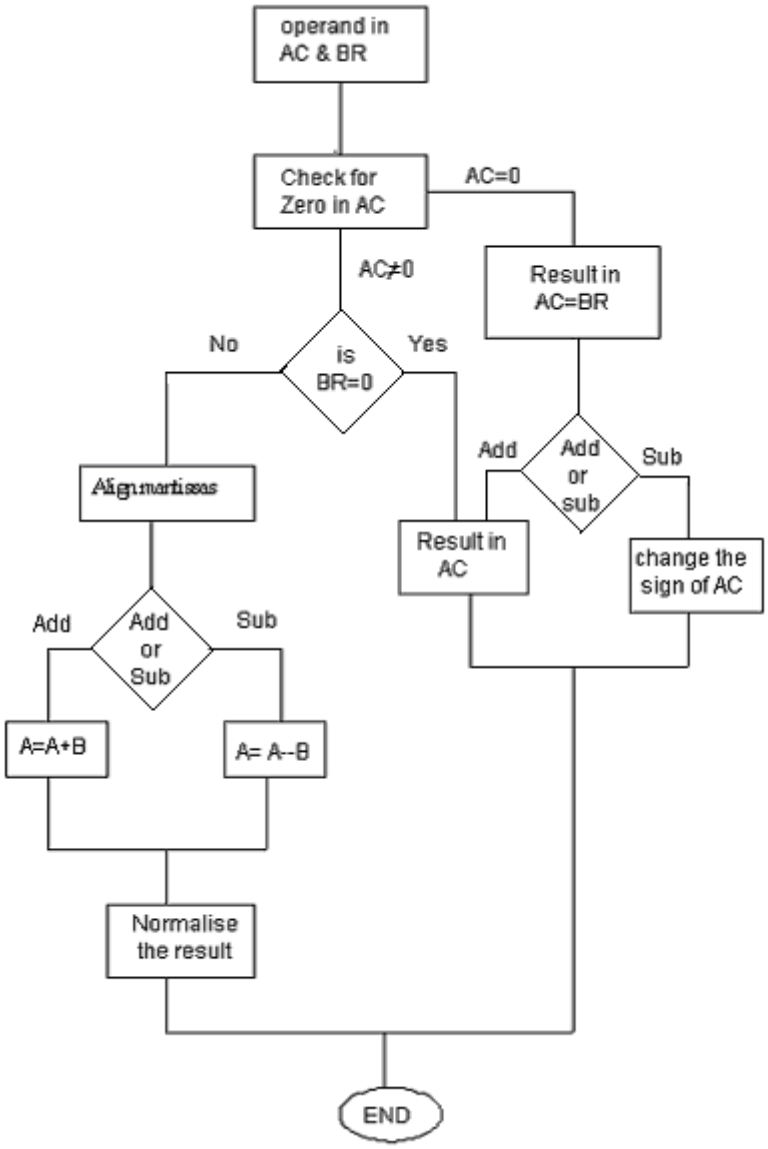
For addition the result is 0.245×10^2

For subtraction, the result is 0.215×10^2

Both are in normalized state. If not than mantissas needs to be shifted left & exponent is decremented until a non-zero digit appears in the first position of mantissas.

Example - If the result is 0.0035×10^5

Than after shifting left two times to mantissas & documenting exponent by e, we get the normalized result as 0.35×10^3.



Flow Chart:-

Q.140 What is meant by normalization? Why we do normalization of floating point numbers? **(6)**

Ans:

When two normalized mantissas are added, the sum may contain an over flow digit. An over flow can be corrected easily by shifting the sum once to the right and incrementing the exponent.

Example 0.35×10^5

$$\underline{+\ 0.73 \times 10^5}$$

1.08×10^5 over flow occur.

After normalization it becomes. 0.108×10^6.

Similarly when two normalized mantissas are subtracted, the result may contain most significant zero as shown below.

$$0.56 \times 10^5$$

$$\underline{-\ 0.53 \times 10^5}$$ under flow occurs.

0.03×10^5

To normalize the under flow condition, mantissas is shifted left and decrement the exponent until a non-zero digit appears in the first position.

$$0.03 \times 10^5 \xrightarrow{\text{normalised}} 0.3 \times 10^4$$

Normalization means to have the most significant digit of mantissa to be non-zero.

It is done in case of floating point addition and subtraction to over come overflow and underflow as discussed in above example.

Q.141 Explain the role of stacks for handling interrupts. **(4)**

Ans:

When an interrupt comes to the processor, the processor recognize it only if the interrupt enable FF is ON i.e. IEN = 1. The processor complete the execution of the instruction currently giving on and sends a acknowledgement signal the I/O device generating interrupt signal. On receiving the interrupt acknowledge signal from CPU.

The hardware for interrupt handling puts out the address of the memory where the interrupt service routine (ISR) is stored. If the program counter content of the processor w.r.t. its first program and other register contents associated with the first program along with flag register, accumulator are not stored properly, than the first program can't be continued after ISR being executed. So it is essential for CPU to save the contents of PC, AC & other processor register in some designated memory location. For this work, stack memory is used by most of the processor. As the stack works on LTFO principle, so it is convenient to save register contents before servicing the I/O device interrupting the CPU and also it is convenient to POP up all register information's sequentially after the processor returns back to main program at the end of ISR.

In this way stack memory is used in handling the interrupts by the processor in a interrupt drives circumstance.

Q. 142 What is the sequence of steps that will take place when an interrupt occurs? **(6)**

Ans:

When an interrupt is recognized the following steps are carried out by the processor-

1. Complete the execution of current instruction.
2. Document the stack pointer. SP ← SP-1.
3. Save the program counter content in to stack

   M[SP] ← PC.
4. Save the contents of processor registers.
5. Send interrupt acknowledgement signal to interrupt controller.
6. Transfer the vector address of the corresponding interrupt to program counter.
7. Disable the interrupt enable FF, so that other interrupt signal can't be recognized till the end of ISR.
8. Go to fetch the first instruction of ISR & execute the ISR.

When the processor came access the return instruction of ISR, it dies the following-

1. activate the interrupt enable FF to allow it to be interrupted.
2. restore contents of processor registers.
3. restore the return address saved in stack in to PC.
4. continue execution of main program.

Q.143 Define hit ratio and explain its significance. **(4)**

Ans:

When the CPU refers to memory, and finds the word in cache, it is said to produce a hit. If the word is not found in cache, it is in main memory and it counts as a Miss. The performance of cache memory is frequently measured in terms of a quality called hit ratio. The ratio of the number of hits divided by the total CPU reference to memory (hits + misses) is called hit ratio.

Say, the memory access time for main memory = A_m & cache memory access time = A_c, If the hit ratio = 0.9, than the average memory access time = $(0.1\ A_m + 0.9\ A_C)$.

The performance of cache memory improves if the hit ratio is high.

Q.144 Show the hardware to be used for addition and subtraction of signed 2s compliment representation. Explain how it operates and how an over flow will be handled. **(14)**

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.

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 (–6) – (–13) = +7.

$-6 \rightarrow 1111\,1010$

$+13 \rightarrow 0000\,1101$

$\textcircled{1}\ \overline{0000\,0111} = +7$

discarded

Overflow:- During addition / subtraction of signed 2's complement representation, if the result occupies (η+1) digits, than an over flow occurred

$+70 \rightarrow 0100\,0110$

$+80 \rightarrow 0101\,0000$

$\overline{1001\,0110}$

↓

Sign bit has been changed

$-70 \rightarrow 1011\,1010$

$+80 \rightarrow 1011\,0000$

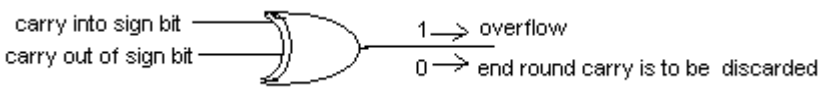
$\overline{-150}\quad \overline{1011\,0\,1010}$

↓

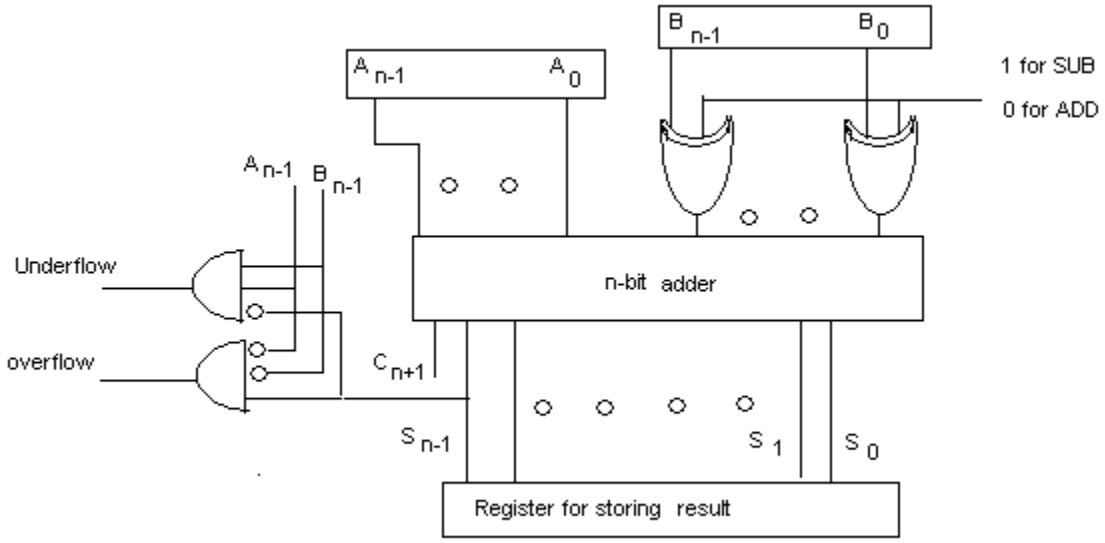
result is (η+1) bits.

In above two cases, overflow has occurred. It is not that the carry out of sign bit is to be discarded. Hence to distinguish the situation where end round carry is to be discarded & when it is an over flow, the following observations must be made.

If the carry in to the sign bit position is equal to the carry out of sign bit position than the end round carry is to be discarded & there is no overflow. Other wise if these two carry are not equal. Then it is considered to be overflow. Hence, the overflow condition can be checked as follows.



The hardware realization of signed 2's complement addition & subtraction is as given below.



Q.145 What is an instruction? What are the different parts of an instruction? Explain the significance of each part of an instruction with an example? **(8)**

Ans:

A computer instruction is a binary code that specifies a sequence of micro operations for the computer. Instruction codes together with data are stored in memory. The computer reads each instruction from memory and places it in a control register. The control then interprets the binary code of the instruction and proceeds to execute it by issuing a sequence of micro-operations.

op-code	Operand

→ Instruction code.

An instruction code consists of two ports.

(i) Opcode → operators part

(ii) Operands → Data / Address of data

The operation code of an instruction is a group of bits that define such operations as add, subtract shift, multiply etc.

The operand part of the instruction can be a data on which operation as specified by opcode is to be carried out or it can be a address of the memory where the data is available.

Example : **Add 450**, is an instruction where the opcode is **Add** i.e. the operation is to be carried out is addition of two operand. One operand is available in accumulator. The other one is available in the memory having address 450. The result of accumulator shall be stored in accumulator.

In case of indirect addressing mode, the instruction is **Add @ 450**. i.e. operand is in accumulatorhe other operand is to be obtained indirectly. The operand part of the instruction is 450, is the address of the memory location, where not the data but the address of the data is available. So after reading the memory location 450, it gets another address of memory location, where data is to be found.

If the Opcode part of the instructions is of n bits, than there can be 2^n no, of distinct operations in that computer. The bit length of operand part signifies the size of the memory. If the bit length of operand part is of K-bits, than the CPU can address 2^k memory words. So each part of the instruction has their own significance.

Q. 146 If a Computer has 128 operation codes and 512 k addresses, how many bits would be required for
(i) Single address instruction (ii) Two address instruction. **(6)**

Ans:

No. of Op codes = 128 = 2^7

Size of memory = 512 K = $2^8 \times 2^{10}$ bits.

(a) Size of single address instructions = 7 + 18 = 25 bits.

(b) Size of two address instruction = 7 + 2(18) = 7+36 = 43 bits.

Q. 147 What are the different modes of data transfer? Explain the DMA Controller with a block diagram. What is meant by block transfer? **(2+10+2)**

Ans:

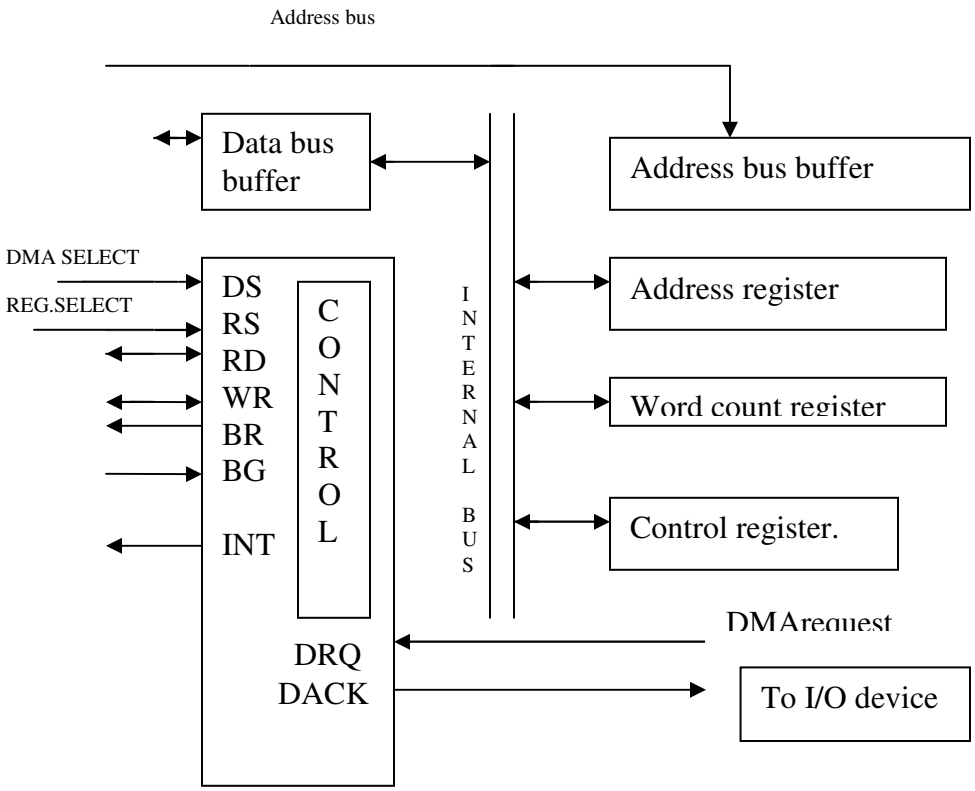
Different modes of data transfer are-

1. Programmed I/O
2. Interrupt initiated I/O
3. Direct memory access (DMA)

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 would improve the speed of transfer. This transfer technique is called direct memory access. Dining DMA controller takes over the buses to manage the transfer directly between I/O device & memory.

The block diagram of DMA controller is given below:-



DMA controller communicate 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 and RS inputs. BR input is used by the DMA controller to request the CPU to relinquish control of the buses. When this input is active, the CPU terminates the execution of the current instruction and places the address bus, data bus and the read and write lines in to a high impedance state. The CPU activates the bus grant (BG) output to inform the DMA controller that the buses are in the high impedance state. Now the DMA can conduct memory transfer without processor's intervention. The address register contains an address to specify the desired location of memory. The address register is incremented after each word that is transferred to memory.

The word count register holds the No. of words to be transferred. This register is decremented after each word transfer and is tested for zero internally.

The DMA controller is first initialized by CPU by sending the following information's through the data bas.

1. The starting address of memory block were data are available for DMA read operation or where data are to be stored for DMA write operation.

2. The word count, to word count register.

3. Control to specify the mode of transfer such as read or write.

4. A control to start DMA transfer.

The transfer of data between memory & I/O device can be in two ways-

1. Burst transfer

2. Cycle stealing

Burst Transfer :- A block sequence consisting of a number of memory words is transferred in a continuous burst while DMA controller is master of the memory bus. This mode of transfer is needed for fast I/O devices.

Q.148 Explain the different shift operations with examples. What is the status of flag bit in each case? **(7)**

Ans:

Shift micro operations are used for serial transfer of data. They are also used in conjunction with arithmetic, logic and other data processing operations. The content of the register can be shifted left or right. There are three types of shifts.

1. Logical shift
2. Circular shift
3. Arithmetic shift

Logical Shift:- In this shift a zero is transferred through the serial input.

Examples: $R_1 \rightarrow 11000011$

$ShrR_1 \rightarrow 01100001$, LSB is lost.

$Shl\ R_1 \rightarrow 10000110$, MSB is lost

Circular Shift:- Also known as rotate operation.

It circulate the bits of the register around the two ends without loss of intimation.

Let the register content of R_1 = 11000011

Cil → Circular shift left = 10000111

Cir → Circular shift right = 1110001

Arithmetic Shift:-This shift a signed binary number to left or right. An arithmetic shift lest multiplies a signed binary number b2 and an arithmetic shift right divides the number by 2. Arithmetic shift must leave the sign bit un changed. The left most bit in a register holds the sign bit.

Let the register contains 10111100 = (–68)

Then shift right operation gives 11011110 = (–34)

The sign bit is shifted right & also remain in its position, so as to get sign unchanged. The arithmetic shift right results the original number divided by 2. In case of arithmetic shift left operation, the original sign bit is lost. So in order to ensure that the sign bit remains same even after the shift left operation, the two MSB bits must be same. Other wise change of sign takes place is known as over flow. This overflow set a FF 'V_s' to be 1 to indicate that there is an overflow.

$V_S = R_{n-1} \oplus R_{n-2}$.

Example $R_1 = 11000011$, $R_{n-1} = R_{n-2}$.

ashl $R_1 = 10000110$, Sign bit remain unchanged.

Originally $R_1 = 11000011 = (-61)$

After ashl $R_1 = 10000110 = (-112)$ which is equal to two times of (-61).

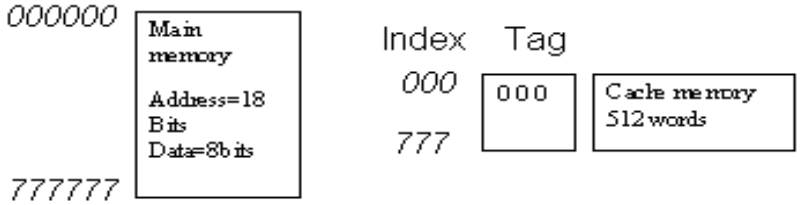
Q. 149 A personal computer has main memory of $32K \times 8$ bytes and cache memory of 512 words. The cache is directly mapped with block size of 4 words.
(i) How many bits are required in tag, index block and word fields of the address format?
(ii) Show the addressing format?
(iii) What are the advantages of direct addressing scheme? **(3+2+2)**

Ans:

Size of main memory = 32 K × 8 bytes = $2^5 \times 2^{10} \times 2^3$ bytes = 2^{18} bytes.

Size of cache memory = 512 words = 2^9 words.

Mapping method is direct mapping Block size = 4 words



(i) No. of index bits = 9, No. of tag bits = 18-9 = 9

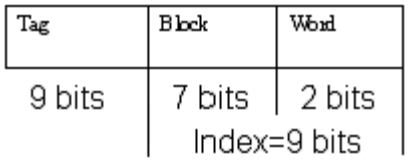
Total no. of bits in each word of cache = Data bits + Tag bits

= 8 + 9 =17 bits

No. of bits for representing block = 7

No. of bits for representing the word in the block = (9-7) = 2.

(ii) The addressing format for cache memory is by the index bits starting from (000) to (777) in octal representation.



(iii) Advantages of direct mapping scheme are-

(a) Less expensive than associative memory due to decrease in word size of cache memory.

(b) Mapping logic is simpler & less complex as only the tag field is to be matched instead of the whole address of the main memory.

Q.150 Simplify the Boolean function F together with the don't-care conditions d in (1) sum-of-products form and (2) product-of-sums form.

$$F(w, x, y, z) = \sum (0,1,2,3,7,8,10)$$

$$d(w, x, y, z) = \sum (5,6,11,15)$$

Draw the logic diagrams also. **(8)**

Ans:

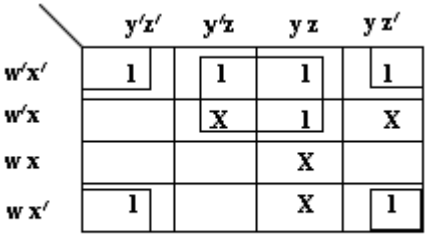
Given function

$$F(w, x, y, z) = \Sigma(0,1,2,3,7,8,10)$$

$$F(w, x, y, z) = \sum (0, 1, 2, 3, 7, 8, 10)$$

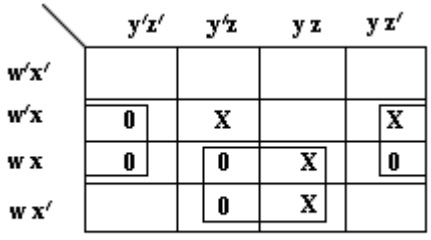
$$D(w, x, y, z) = \sum (5, 6, 11, 15)$$

Sum of products



$F = x'z' + w'z$

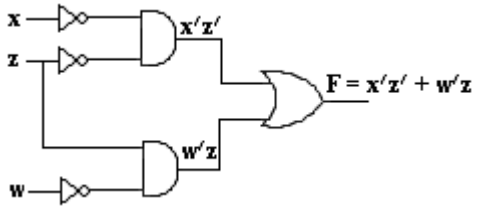
Product of sums



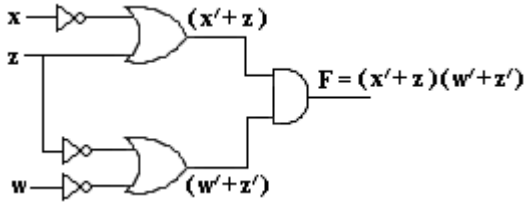
$F = (x' + z)(w' + z')$

Logic Diagram

SOP



POS



Q. 151 Represent the following decimal numbers in BCD, Ex - 3 and gray code.
(i) 56 (ii) 93

Ans:
(i) 56
Binary equivalent of 56 is 111000

BCD representation of 56 is 0101 0110

Er-3 representation of 56 is 10001001

Gray Code 56 = $(111000)_2$

Gray Code = (100100)

(ii) 93
Binary equivalent of 93 is 1011101

BCD representation of 93 is 1001 0011

Er-3 representation of 93 is 11000110

Gray Code 93 = $(1011101)_2$

Gray Code = (1110011)

Q. 152 Multiply $+.1001 \times 2^{-0011}$ and $-.1010 \times 2^{-0001}$ using binary multiplication algorithm. Show all steps. **(8)**

Ans: Let $X = +.1001 \times 2^{-0011}$

$Y = -.1010 \times 2^{-0001}$

Steps

1 : Add Exponents
(–0011) + (–0001)
= –0100

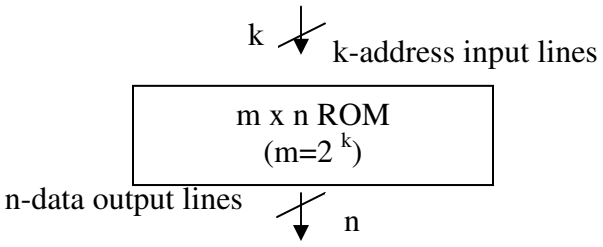
2 : Multiply mantissas
+ .1001 x –.1010
= .01011010

3 : Normalize and round the final product.
Hence, the final product is $1.011010 \times 2^{-0110}$

Q.153 Explain what do you understand by ROM ? **(4)**

Ans:

R.O.M.stands for (read-only-memory).it is a memory unit that performs the read operation only. It does not have a write capability. The binary information stored in it is made permanent during the hardware production. R.O.M. comes with special internal electronic fuses that can be programmed for a specific configuration by blowing out fuses of required position by ROM programmer.



A m x n ROM is an array of binary cells organized in to the m words of 'n' bits each as shown in the block diagram, it has K-address input lines to select one of the 2^k=m words of memory. As it has 'n' output lines, so the word length of each memory cell is of n-bits it is used for storing fixed programs that are not to be altered.

Example: design of control unit for stored-program computer.

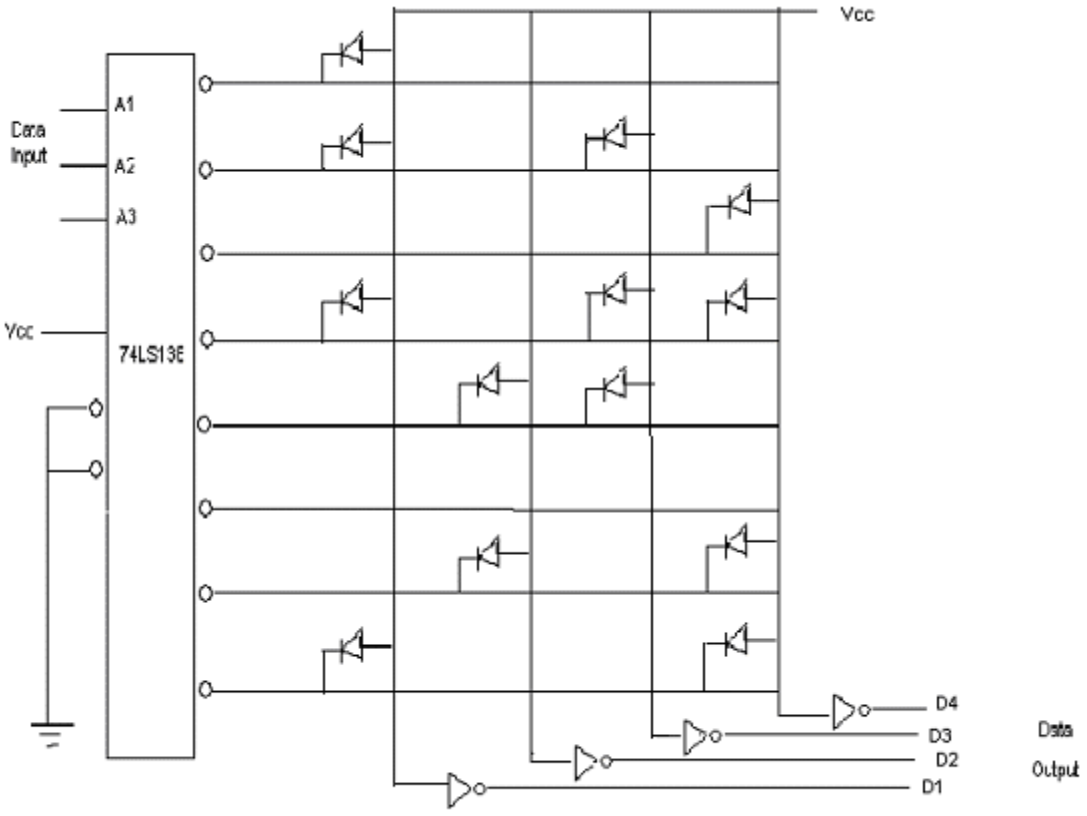
Q.154 With the help of a diagram using diodes in a matrix form, explain the concept underlying ROM ? **(6)**

Ans:

A ROM implementing the following truth table using diodes in a matrix form is shown below.

Truth table

Address inputs			output data			
A3	A2	A1	D4	D3	D2	D1
0	0	0	0	0	0	1
0	0	1	0	1	0	1
0	1	0	1	0	0	0
0	1	1	1	1	0	1
1	0	0	0	1	1	0
1	0	1	0	0	0	0
1	1	0	1	0	1	0
1	1	1	1	0	0	1



Diodes are used to connect the output lines of the decoder to the data output lines that corresponds to logic 1's in the truth table for example when the input address lines are 000, decoder output '0' is selected. According to the truth table, the required output is 0001, so data line D1 must be a '1'.

Q.155 Define PROM and EPROM. **(4)**

Ans:

PROM: is the programmable read only memory. The user programs it. In PROM, FET or BJT are used in place of diodes of a ROM. PROM have each address select line connected to a FET gate (or bipolar transistor base) with the source (emitter) grounded and the drain (collector) connected to the vertical data lines. A fuse link exists between grounds & the FET source (or bipolar transistor emitter). Because PROMS are programmed by the user, each address select & data lines intersection has its own fused FET or transistor. When the fuse is intact the memory cell is configured as logical '1', and when the fuse is blown (open circuit) the memory cell is logical'0'. Logical 0's are program by selecting the appropriate select line then driving the vertical data line with a pulse of high current.

EPROM: stands for erasable programmable read only memory. Instead of fused bi-polar or FET transistor switch as in PROM, it has a MOSFET that has two gates. The memory cell is configured like PROM cell, however no fuse is present. When an EPROM is programmed, a charge is placed on the floating gate, permitting channel current to flow & thus establishing the logic level of the memory cell .The floating gate to channel voltage is discharged when placing it under ultra violet light erases the device. It takes approximately 5 to 20 minute of exposure to the UV light to erase the device.

Q.156 Give the three methods by which the subtraction of two n-digit unsigned numbers M-N ($N \neq 0$) in base r can be carried out. **(3)**

Ans:

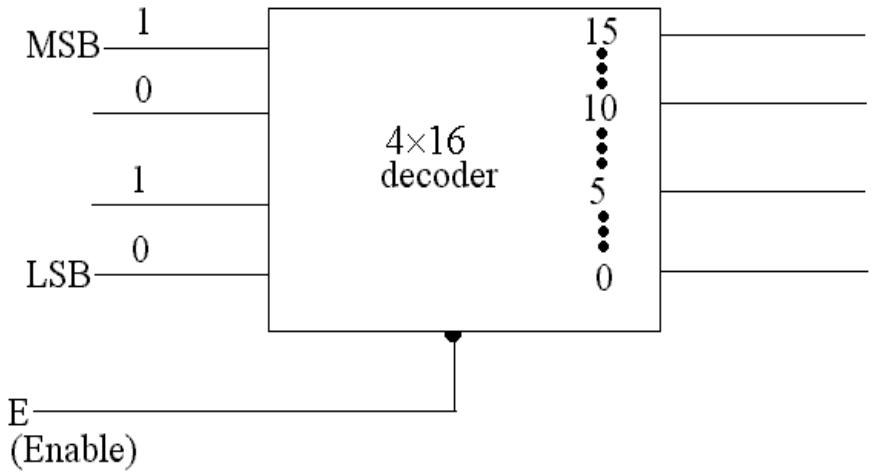
The three methods by which the subtraction of two n-digit unsigned numbers i.e.
M- N (where N is not equal to 0) in base r can be carried out are:

i. **Direct method of subtraction:** in which the subtraction in which we borrow a 1 from the higher significant position when the minuend digit is smaller then the corresponding subtrahend digit.

ii. **(r-1)'s complement method:** where the (r-1)s complement of subtrahend is added to minuend i.e. $M + (r^n - N)$

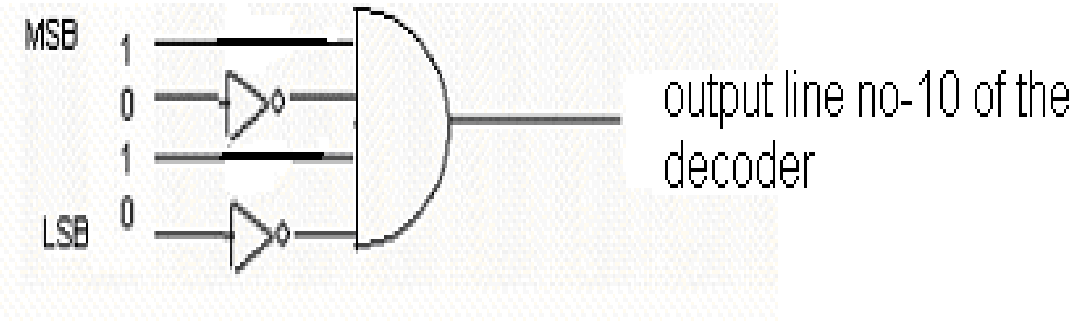
iii. **r's complement method:** where the r's complement of subtrahend is added to minuend i.e. $M+[(r^n - N)+1]$.

Q.157 In the 4 – to –16 decoder shown in the figure, mark the output line of the decoder which goes low, when the input to the decoder is as shown in this logic diagram. Give reasons for your answer. **(5)**

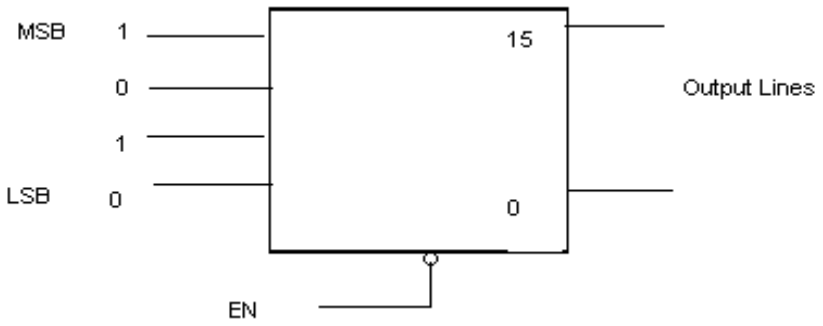


Ans:

When the input to (4x16) decoder is (1010), with enable active low, the output line marked with decimal '10' shall be active high. Because the AND gate corresponding to output line 10 shall produce a high output.



(If the output are buffered, than with the same configuration a zero shall be produced on the output line---10)

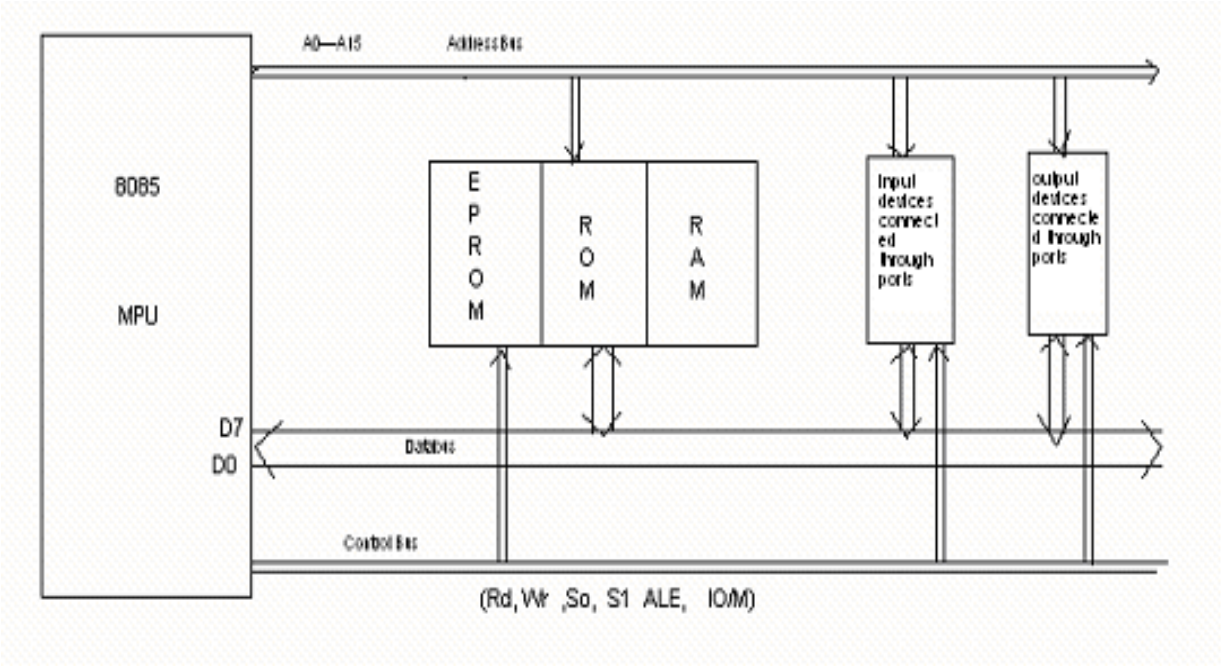


The (4x16) decoder consists of 16 AND gates which decodes all the 16 possible minterms, one of the AND gate out of 16 shall produce a high output depending on the input bits combination.

Q.158 With the help of a diagram, explain a microcomputer system having microprocessor 8085, EPROM and RAM memories, input and output, and buses linking all peripherals (memory and I/Os) to the microprocessor unit. **(6)**

Ans:

8085 microprocessor is a 8 bit microprocessor i.e. it can operate on the data of 8-bit at a time hence it has an 8-bit ALU to perform arithmetic operation like add, substraction and logical operation like ANDing, ORing, complementing etc….



It has got 16-bit address line (A0 toA15) where A_0 toA_7 are multiplexed with the data lines D0-D7. It can address up to 2^{16} =64k of memory locations. Further it has a control signal $IO/\overline{M}$ which is 1 for addressing to an I/O device & is '0' while addressing memory locations. In I/O mapped I/O, 8085 microprocessor can address up to 2^8 = 256 I/O devices.

The memory unit consists of EPROM, ROM & RAM

The monitor programs & other essential programs and data are stored in EPROM or ROM whereas the RAM memory is used by the user for storing the programs & data, which is not of permanent nature. Besides address lines & data lines, a microprocessor system has got control bus consisting of control signals. These control signals are generated by the control unit of 8085 microprocessor and are connected to all peripherals. Control signals controls the flow of operations sequentially. RD, WR, $IO/\overline{M}$, S0, S1, ALE are some of the control signals of 8085 microprocessor . MPU generates specific control signals for every operation such as memory read, memory write, I/O read, I/O write, DMA operation, interrupt cycles etc….

The data bus of the computer system is bi-directional, where as the address bus is unidirectional.

Input devices are connected to MPU through their respective interface circuits to feed data from external world to the computer system. The example of input devices are keyboard, mouse, joystick, etc….similarly, output devices are used to transfer data from the microprocessor to the outside world. They include devices such as light emitting diodes, CRT or video screen, a printer, a magnetic tape x-y plotter, D/A converter etc…

Q.159 Differentiate between :

(i) micro instructions and micro operations. **(2)**

(ii) mini computers and microcomputers. **(2)**

(iii) RAM and ROM chips. **(2)**

Ans:

(i) **Microinstructions:** A microinstruction specifies one or more micro-operation for the system and a sequence of microinstruction constitutes a micro program, which besides in control memory.

Micro operation: it is an elementary operation performed with the data stored in registers.

To perform a micro operation microinstruction generates necessary control signals in proper sequence.

ii) **Mini computer**: These are the computer of medium sized, which are slower, then the mainframes and also have smaller memory capacity. In earlier days these machines were used to meet the needs of manufacturing needs of small factories, data processing task of medium sized business & colleagues. Now a day, even the high end microcomputers are more powerful then earlier minicomputers.

Microcomputers: In earlier days 4bits & 8bit microprocessor oriented computer systems were called as microcomputers. But now a days due to invent of powerful microprocessors of 16, 32 and 64-bit the microcomputers are quite powerful with large memory capacity. The personal computers come under the category of microcomputers.

iii) **RAM:** it is a read- write semiconductor memory. In this memory, user can write data many a times and also can read it.
ROM: it is a read only memory. Although it also works on the principle of random access but it cannot be written more than once. As fusible links are burnt while writing this type of memory, so it can be written once and the data is of permanent nature.

Q.160 Write short notes on :
(i) Cache memory. **(4)**
(ii) Shift instructions. **(4)**

Ans:
(i) **Cache, memory:** it is a special high-speed memory called a cache, is used to increase the speed of processing by making current programs and data available to CPU at a rapid rate. The cache memory is employed in computer systems to compensate for the speed differential between main memory access time and processor logic. CPU logic is usually faster than main memory access time, with the result that processing speed is limited primarily by the speed of main memory.
To compensate for the mismatch in operating speed, extremely fast small cache between CPU & main memory, whose access time is close to processor logic clock cycle time, is used. Cache is primarily used for storing segments of programs currently being executed.
ii) **Shift instruction:** shift instructions are used for serial transfer of data in a register. These are also used in conjunction with arithmetic, logic & other data processing operations. The content of the register can be shifted left or right. There are three types of shift operations.
a) Logical shift: it transfers 0 through the serial input.
b) Circular shift: it does the rotate operation .it circulates the bits of the register around the two ends without loss of information.
c): arithmetic shift: it shifts a signed binary number to the left or right. An arithmetic shift left multiplies a signed binary number by 2 and an arithmetic shift right divides the number by 2. It leaves the sign bit unchanged.
All the above-mentioned shift instructions are of two types depending on the direction of shifting of bits in the register. i.e. left shift or right shift. The examples of shift instruction are SHR, SHL, SHRA, SHLA, ROR, ROL, RORC, & ROLC. Where RORC &ROLC are the instructions for shifting through carry.

Q.161 In connection with the assembly language, what do you understand by

(i) field
(ii) symbolic address
(iii) pseudo instructions **(9)**

Ans:

i) **Field:** The bits of the instruction are divided in to groups called fields. The most common fields found in instruction formats are: opcode field, address field & mode field. Other special fields are some times employed under certain circumstances. The group of bits of any particular field specifies various processing information to the processor. E.g.: ADD XXXX, where ADD represents the opcode field & specifies the operation and XXXX represents the address field where the operand is found and the other operand is always in accumulator.

(ii) **Symbolic address:** A symbolic address consists of one, two or three but not more than three alphanumeric characters. The first character must be a letter; the next two may be letters or numbers. The symbol can be chosen arbitrary by the programmer. A symbolic address in the label field is terminated by a comma (,), so that it will be recognized as a label by the assembler.

(iii) **Pseudo instruction:** These are not machine instructions, rather an instruction to the assembler giving information about some phase of the translation. e.g.: ORG(origin) informs the assembler that the instruction or operand in the following line is to be placed in a memory location specified by the number next to ORG. similarly END,DEC,HEX are also pseudo instructions which instruct the assembler about end of program, the number is decimal, the number is hexadecimal respectively.

Q.162 (i) Explain subroutine. **(2)**

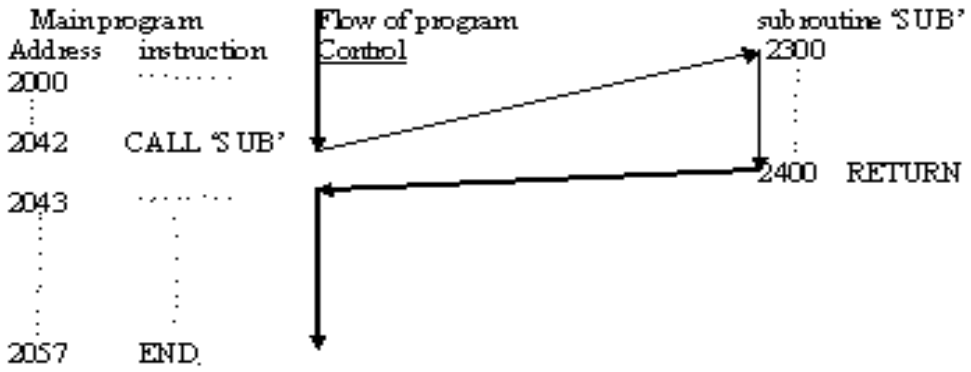
(ii) Demonstrate the use of subroutines with the help of suitable diagram. **(3)**

Ans:

(i) **Subroutine:** 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 is executed, a branch is made back to the main program. Subroutine consists of a self-contained sequence of instructions that caries out a given task.

(ii) A subroutine is called in a main program by using suitable CALL instruction. When this CALL instruction followed by the name of the subroutine is executed, the return address of the main program is stored in the stack memory or any specified register. The program counter holds the starting address of the subroutine program.

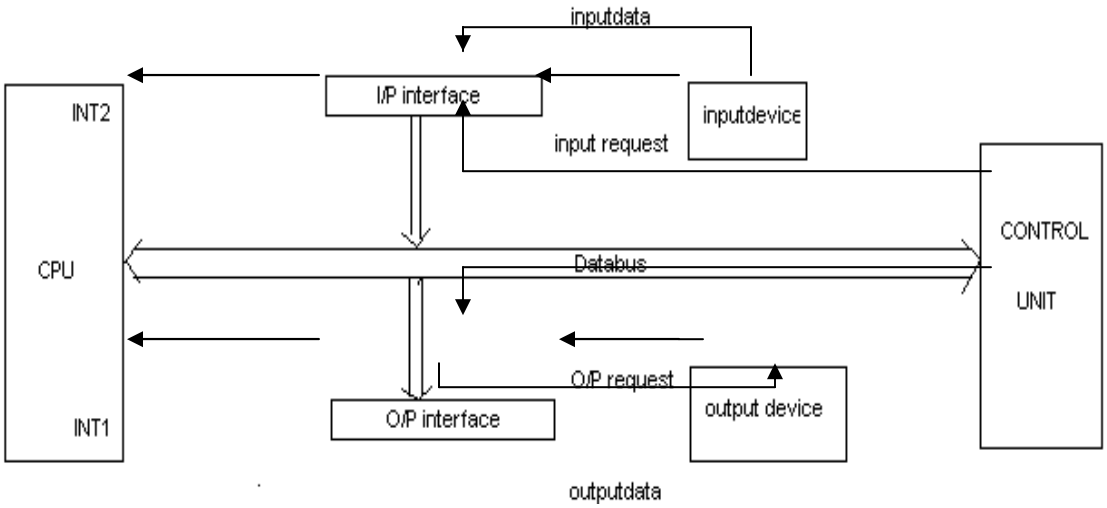
At the end of the subroutine there must be a RETURN instruction. This enables the processor to return the control of program flow again to main program. The return address that is stored in the stack is loaded in program counter and the execution of main program proceeds.



Q.163 What do you understand by the term 'Program Interrupt'. Explain with the help of suitable diagrams. **(8)**

Ans:

A computer system is used to communicate with the input and output devices. The data flow is from input device to computer and from computer to output device .Now the simplest way of transfer of data between I/O device and computer is called program control data transfer. In which the CPU keep on searching for the data buffer of input and output devices in order to receive a data or to send a data. Hence CPU does no useful work & CPU time is wasted in waiting for data transfer between I/O devices. To improve the efficiency of CPU, program interrupt technique is used. In which CPU does its own work without checking the input and output flag. Whenever an I/O device needs to communicate, it sends an interrupt signal. On receiving the interrupt, the normal processing of CPU gets stopped temporarily and it saves the return address of the program in stack and the program for data transfer between the CPU & I/O device is executed. Once this communication is over, again CPU comes back to the 1st program by obtaining the return address from stack. In this way CPU efficiency is improved. Hence, this program interrupt is applicable in multiprogramming environment that is when two or more program resides in memory at the same time. The block diagram arrangement of program interrupt technique is shown below.



When the input device needs to send a data to CPU, its interface circuit initiate a interrupt to CPU through INT2 line and similarly when O/P device need a data from CPU, its control circuit generates INT1 signal to CPU. Both input & output of data is carried out by generating interrupt signal to CPU Hence, in this scheme, interrupt is generated by external device only.

Q.164 Differentiate between the terms external, internal and software interrupts. **(6)**

Ans:

There are three major types of interrupt that causes a break in the normal execution of a program. They are:

(i)External interrupt: it comes from I/O devices on from any external sources. Any microprocessor has got same interrupt lines. For example 8085 has got RST 7.5,RST 6.5,RST 5.5,INTR & TRAP lines Any I/O device connected to these lines sends interrupt signal when need the service of microprocessor. As these interrupts come from an external I/O device, so it is called external interrupt. These are asynchronous with the program and independent of it.

(ii)Internal interrupt: it arises from illegal or erroneous use of an instruction or data. Examples of interrupts caused by internal error conditions are register overflow, attempt to divide by zero, an invalid operation code, stack overflow and protection violation. These error conditions usually occur as a result of premature termination of the instruction execution. The service program that process the internal interrupt determines the corrective measures to be taken hence the internal interrupt is initiated by some exceptional condition caused by the program itself rather than by an external event. These are synchronous with the program. If the program reruns, internal interrupts will occur in the same place each time.

(iii)Software interrupts: initiated by executing an instruction. Software interrupts are special call instructions that behave like an interrupt rather than a subroutine call. It can be used by the programmer to initiate an interrupt procedure at any desired point in the program.

Q.165 Write short notes on
(i) Conditional branching **(3)**
(ii) Flags **(3)**

Ans:

(i). **Conditional branching**: branching from main program to a subroutine or to another part of the main program when a particular condition is satisfied is called conditional branching. Hence before branching from normal flow of program execution, the condition is tested. If satisfied, then the program flow jumps to a new location of the program or to a subroutine. The conditions are checked with the status of flag bits. For example, jump on +ve is a conditional branching. It checks the sign flag associated with accumulator content and it branches to the new location or address as specified by the condition at jump instruction is not satisfied, then the normal flow of execution of the program continue. The other examples of conditional branching are jump on minus, jump on even parity, jump on carry, jump on no carry etc….

In case of conditional jump instructions, no return address is saved but in care of conditional call instruction, the return address is to be saved.

(ii). **Flags**: These are nothing but a single bit Flip-Flops, the value of the flip-flop can be zero or one depending on same information associated with accumulator content. For example sign flag is a single bit FF, when holds 1, if the sign of accumulator content is negative in case of 2's complement arithmetic otherwise it is zero. Similarly, parity flag, direction flag, zero flag, auxiliary carry flag, etc… the content of flag bits keeps on changing with the program flow depending on the accumulator content. Flags are tested during conditional branching instructions. The content of flag register & accumulator is called program status word.

Q.166 Explain 'Assemblers' and 'Cross Assemblers'. Give advantages of using them. **(8)**

Ans:

Assembler: the assembler is a program that translates source code or mnemonics in to binary code of the microprocessor and generates a file called object file. It performs various functions such as error checking & memory allocation.

Cross assembler: the cross assembler is a program that develop the assembly language program of a microprocessor running on a computer system which have some other microprocessor. For example we can use PCs based on Pentium microprocessor to develop the program for 8085 microprocessor. Hence, cross assembler is used in PC's to develop the program for other microprocessor.

If we want to develop assembly language program for the same processor, i.e. the processor of the system, then only assembler is required.

Advantage of using assembler & cross assembler: -

(i) Assembler translates mnemonics in to binary code with speed and accuracy, thus elementary human error.

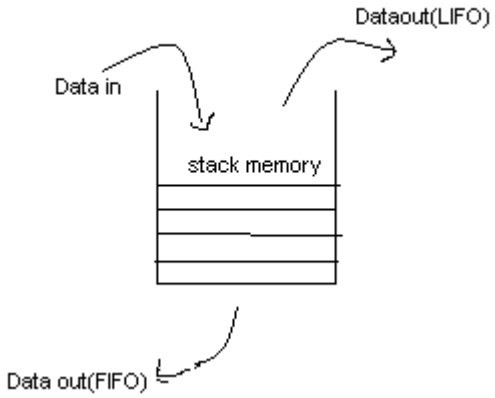
(ii) The assembler assigns appropriate values to the symbol used in the program, this facilitates specifying jump locations.

(iii) Insert/delete instruction in a program is made easy by using assembler. New memory location is also calculated by assembler.

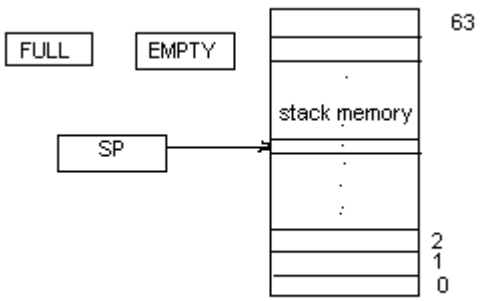
(iv) It can check syntax error.
(v) It can reserve memory for data or result.
(vi) It can provide files for documentation.

Q.167 Explain the principle of stack ; what are LIFO and FIFO operations.**(9)**

Ans:
A stack is a storage device that stores information in such a manner that the item stored can be retrieved in a sequential manner. The stack in digital computers is essentially a memory unit with an address register that can count. This register is called stack pointer. This needs initialization before using a portion of r/w memory as stack memory. The value of the stack pointer always points to the top item in the stack.
The insertion of data in to stack is called push operation and deletion of data from stack is called pop operation. When a push operation is carried out, SP is decremented & incremented with pop operation.
Stack memory deletion process can be of two types.
(i) LIFO (ii) FIFO



LIFO stands for last in first out i.e. the most recent data, which is put in to the stack, can be taken out first. Where as in case of FIFO principle, the data which was inserted first can be taken out first.
Let us see the principle of operation of a stack memory, where the stack pointer is of 6 bits. Hence the total number of memory words that this stack can hold is 2^6 = 64 words as shown in figure below,



To indicate the situation of stack, two flags named as FULL & EMPTY are used. When the stack is full, FULL flag is set to 1 and when the stack is empty of items, EMPTY flag is set to 1.

The micro-operations associated with insertion of data into stack is

SP ← SP+1
M[SP] ← DR where DR stands for data register.
If(SP=0), then (FULL ← 1)
EMPTY ← ◇
Similarly, the micro operations associated with POP is

DR ← M [SP]
SP ← SP-1
If (SP=0) then (EMPTY ← 1)
FULL ← 0

Stack memory is very useful for storing return address and other registers content by the processor when it encounters an interrupt or a call instruction. This stack memory can also be used for efficient evolution of arithmetic operations with reverse polish notation (RPN)

Q.168 List the instructions necessary for using stack. **(5)**

Ans:

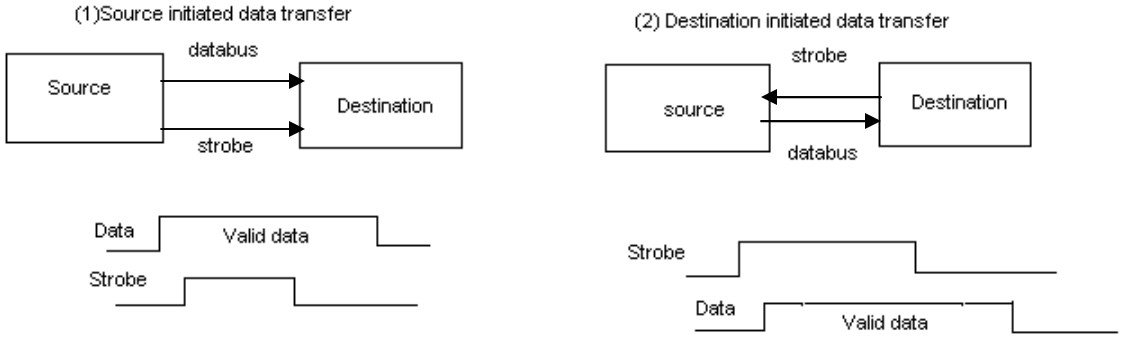
The instructions that are necessary for using stack are:
(I) PUSH X: it will PUSH the word at address X to the top of the stack. The stack pointer is updated automatically.
(ii) POP X: it will take out the word which is available at the top of the stack and the stack pointer is updated automatically.
In case of 8085 microprocessor, the other instructions used for stack is PUSH rp, POP rp, SPHL, XTHL
Where the instruction XTHL exchanges the content of H & L register with the top two data of the stack memory. The content of SP remains same.
The instruction SPHL, copies the content of H register in $(SP)_H$& the content of register L is copied in $(SP)_L$.

Q.169 Explain the Strobe Control method of Asynchronous data transfer. What are the disadvantages of this method? **(14)**

Ans:

When the two units under communication uses their own private clock for internal registers, then it is called asynchronous communication. In this mode of data transfer, it requires some control signals between the communicating devices to indicate the time at which data is being transmitted. One way of achieving this is by mean of Strobe pulse, supplied by one of the unit to indicate to the other unit when data transfer has to occur.
In the strobe control of data transfer, there is only one control line between the destination device & source device. There is no confirmation of receipt of data by destination unit. This is the disadvantage of strobe control method.
The timing diagram that shows the time relationship that must exist between the control signal & the data in the buses are different depending on whether the transfer is initiated by

the source or by the destination device. The strobe signal may be activated by the source or by destination device. If it is activated by source, then it is called source initiated data transfer, in other case it is called as destination initiated data transfer. In first case, data is placed in data bus by source, & than strobe signal is sent to destination to inform the availability of data on data on data bus. The destination unit uses the falling edge of the strobe pulse to transfer the contents of the data bus in to one of its internal registers. In the second case the destination units activates the strobe pulse, informing the source to provide the data. Source places the data on the data bus for acceptance of destination unit. After receiving data, destination unit then disable the strobe. The source removes the data from the bus after a predetermined time interval. The timing diagram is shown below



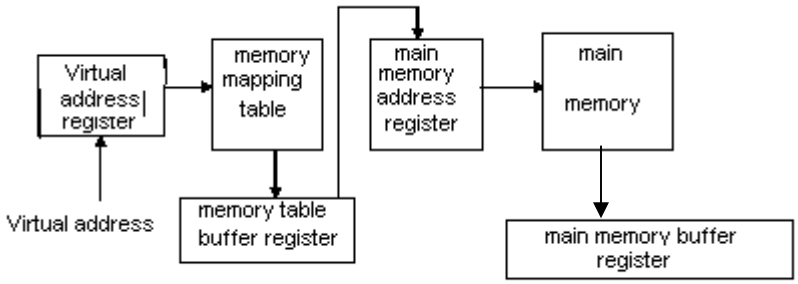
The disadvantage of the strobe method is that the source unit that initiates the transfer has no way of knowing whether the destination unit has actually received the data item that was placed in the bus. Similarly, a destination unit that initiates the transfer has no way of knowing whether the source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

Q.170 Explain the term Virtual Memory. **(7)**

Ans:

Virtual memory is a concept used in large computer system that permits the user to construct programs as though a large memory space is available. The user shall feel that the system has a main memory equal to the totally of auxiliary memory. Each address that is referenced by the CPU goes through an address mapping from the so called virtual address to a physical address in main memory. It gives the programmer the illusion that they have a very large memory at this disposal even though the computer actually has a relatively small main memory.

Say, for a PC with 256 MB RAM & 40 GB hard disk, the auxiliary memory is 40 GB, but the main memory is 256 Megabyte, but in virtual memory concept, the user shall have a feeling of 40Gbyte main memory. A virtual memory system provides a mechanism for translating program generated addresses into correct main memory locations. This done dynamically, while programs are being executed in the CPU. The translation or mapping of virtual address to physical address is handled by the hardware by the means of a mapping table. The address used by the programmer is called virtual address. The address in the main memory is called physical address. For the specification cited above, the address space is of 40×10^{12} and the memory space is of 256×10^{6} words. The memory table for mapping a virtual address is given below



Q.171 Construct an associative memory page table with the number of words equal to the number of blocks in the main memory. **(7)**

Ans:

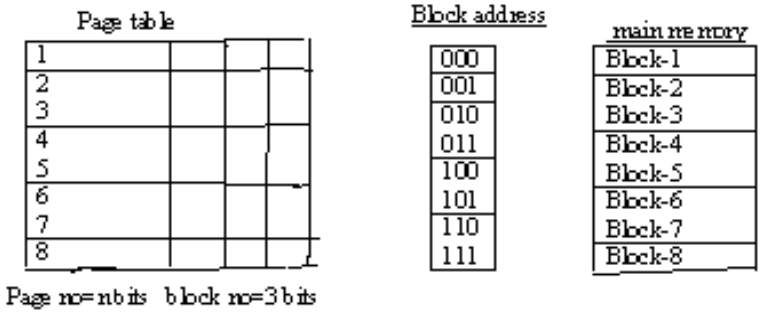
Given ; the no. of words of page table= no. of blocks in main memory.

Let the number of words of a page table=8

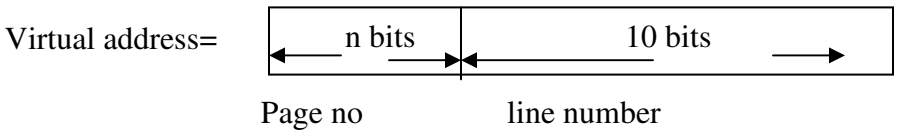
No. of blocks in main memory is 8.

Let each block is of 1k, i.e. 1024 words. So the size of each page shall be also 1 k. so main memory is of 8k. No. of pages= 2^n

Size of address space= $2^n \times 1024$ words.



The virtual address is of n bits plus 10 bits= (10+n) bits 10 bits are required to represent line number & n-bits are required to represent the page number.



The page no of virtual address is compared with the page no. stored in the page table of 8 words. If it matches, than the corresponding word of page table, consisting of (n+3) bits are read out and the block number contained in last three bits of the word gives the

location where this page is stored in main memory. The block numbers along with the line number (10 bits) constitute the physical address of the data, which is required by the program. In this method, the memory space used for storing the page table is utilized fully.

Q.172 Explain the term 'Booth's multiplication algorithm'.**(4)**

Ans:

Booths multiplication algorithm gives a procedure for multiplying binary integers in signed 2's complement representation. It operates on the fact that strings of 0's in a multiplier needs no addition but just shifting and a string of 1's in the multiplier from bit weight 2^k *to* 2^m can be treated as $2^{k+1} - 2^m$. For example, the binary number 001110(+14) has a string of 1's from 2^3 to 2^1 .i.e. m=1, k=3. Therefore the number can be represented as $2^{k+1} - 2^m = 2^4 - 2^1 = 16 - 2 = 14$.

Therefore the multiplicand 'M' is to be multiplied by 14.i.e. the multiplier can be taken as $M\ 2^4 - M\ 2^1$.This is equivalent to shifting the multiplicand 'M' to left for 4 bits and subtracting the multiplicand shifted to left by 1 bit.

Hence the booth's multiplication algorithms requires examination of the multiplier bits and shifting if the partial product. Prior to the shifting, the multiplicand may be added to the partial product, subtracted from the partial product or left unchanged according to the following rules.

(1) The multiplicand is subtracted from the partial product upon encountering the first least significant 1 in a string of 1's in the multiplier.

(2) The multiplicand is added to the partial product upon countering the first 0 (provided that there was a previous 1) in a string of 0's in the multiplier.

(3) The partial product does not change when the multiplier bits is identical to previous multiplier bit.

Q.173 Explain the working of a Half substractor with the help of a diagram. **(4)**

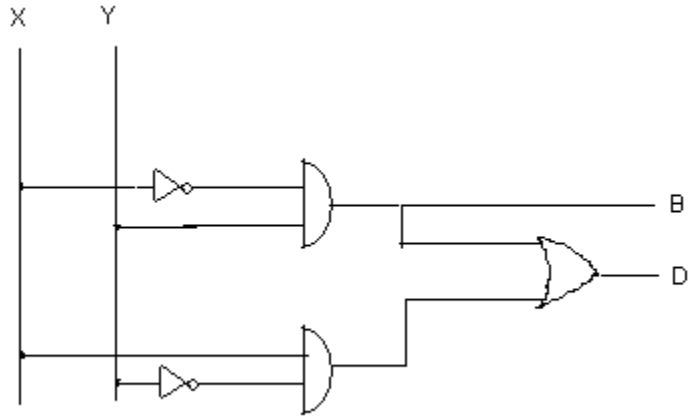
Ans:

Half substractor is a combinational circuit that is used for subtracting two single bits. The result is of two bits, known as borrow & difference. The truth table is given below.

X	Y	B (borrow)	D (difference)
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

$$B = \overline{x}\, y$$
$$D = x \oplus y$$

The realization of a half substractor is

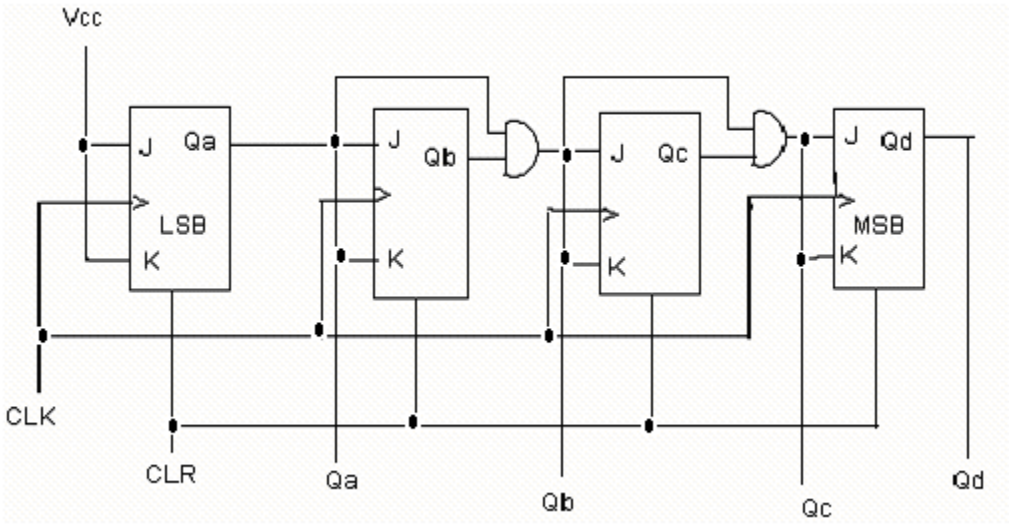


Half Substractor

Q.174 Explain with a neat diagram the working of a 4-bit synchronous binary counter. **(6)**

Ans:

4 bit synchronous counter is shown below.



To start the operation of the counter it is cleared to '0000, state by asynchronous clear input .With the clock pulse, only the LSB FF shall toggle to output 1, as its input J & k are tied to Vcc but the other FFs shall toggle only when the AND of the outputs of FFs preceding it are 1.

Hence the translation table shall be

Clk pulse	Qd	QC	Qb	Qa
CLR	0	0	0	0
↓	0	0	0	1
↓	0	0	1	0
↓	0	0	1	1
↓	0	1	0	0
.	0	1	0	1
.	0	1	1	0
.	0	1	1	1
.	1	0	0	0
.	1	0	0	1
.	.	.	.	.
	.	.	.	.
.	.	.	.	.
	.	.	.	.
↓	1	1	1	1
↓	0	0	0	0 (initial state)

So, the counter will proceed from 0000 to 1111 with the falling edge of the clock pulses. This counter is called synchronous counter as the clock input to all the FFs is connected to same signal source, which produces the clock signals for the counter.

Q.175 Implement the following Boolean function after simplifying using Karnaugh map technique. $f(A,B,C,D)=\sum m(1,2,4,15)+\sum d(0,3,14)$ **(8)**

Ans: $f(A,B,C,D)=\sum m(1,2,4,15)+\sum d(0,3,14)$

	$\bar{C}\bar{D}$	$\bar{C}D$	$C\bar{D}$	$C\bar{D}$
$\bar{A}\bar{B}$	X	1	X	1
$\bar{A}B$	1	0	0	0
AB	0	0	1	X
$A\bar{B}$	0	0	0	0

$$f=\bar{A}\,\bar{B}+ABC+\bar{A}\,\bar{C}\,\bar{D}$$

Q.176 Give the logic diagram and truth table of a full subtractor. Realize the full subtractor using NAND gates only. **(6)**

Ans:

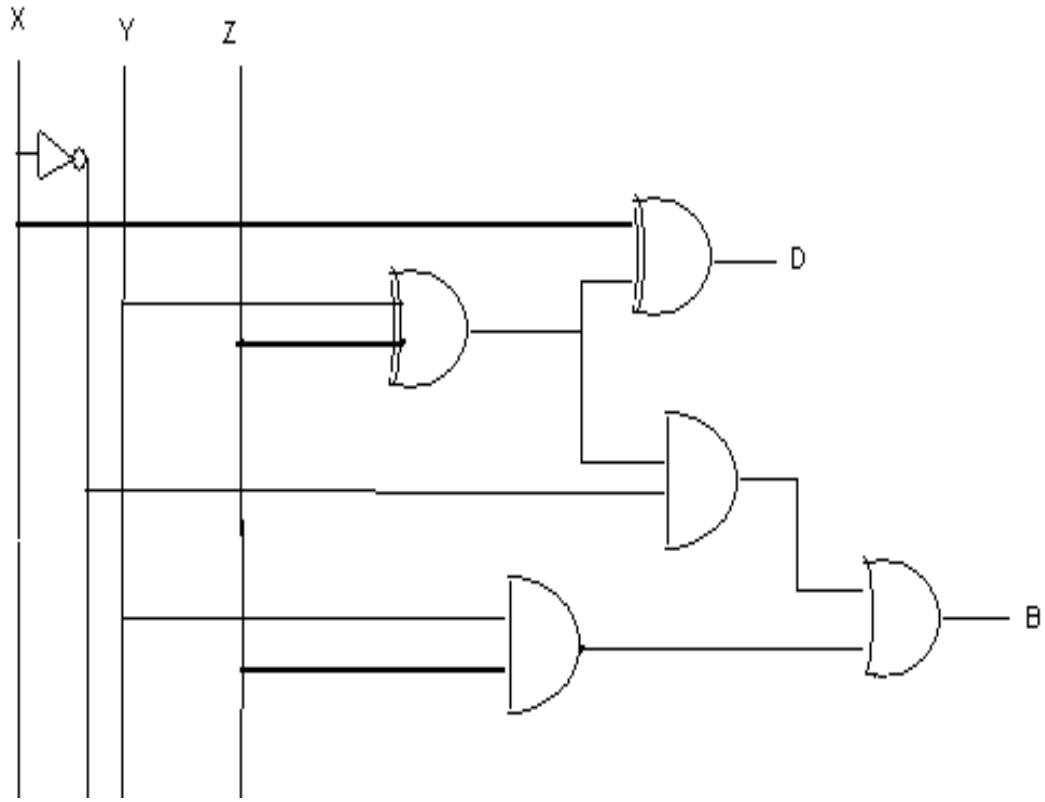
Full subtractor is a combinational logic circuit capable of doing substraction of a substrahend bit from minuend bit with borrow. The table is shown below:

← *input* →			← *outputs* →	
X	Y	Z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

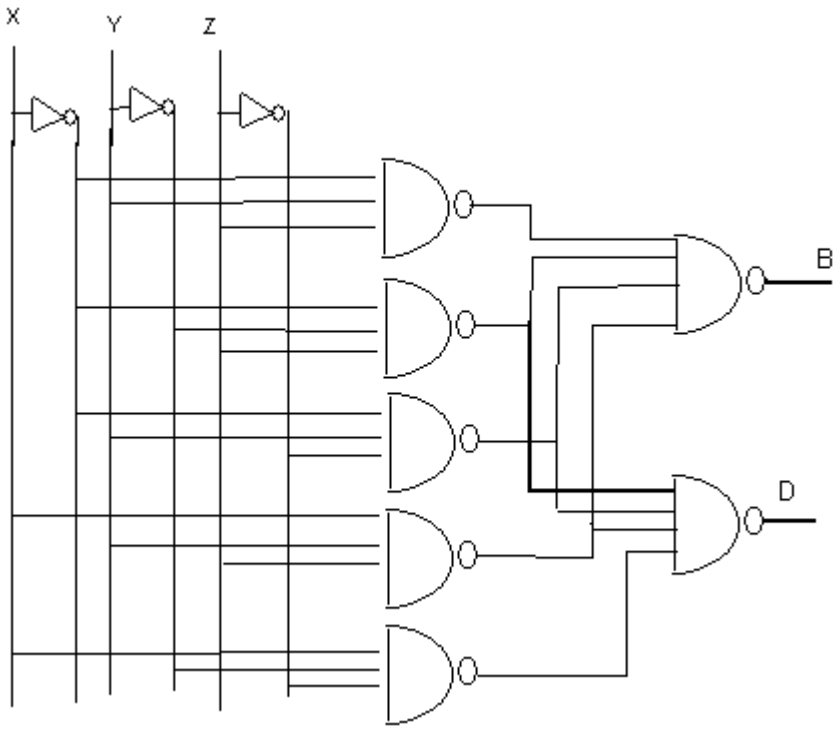
$$B = \bar{x}\bar{y}z + \bar{x}y\bar{z} + \bar{x}yz + xyz$$
$$= \bar{x}(\bar{y}z + y\bar{z}) + yz(\bar{x} + x)$$
$$= \bar{x}(y \oplus z) + yz$$

$$D = \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xyz$$
$$= \bar{x}\bar{y}z + xyz + \bar{x}y\bar{z} + x\bar{y}\bar{z}$$
$$= \bar{z}(\bar{x}\bar{y} + xy) + \bar{z}(\bar{x}y + x\bar{y})$$
$$= z\overline{(x \oplus y)} + \bar{z}(x \oplus y)$$
$$= x \oplus y \oplus z.$$

The logic diagram of the above eqation is given below:



The NAND gate realization of the above circuit is given below:



Q.177 What are sequential circuits? Compare them with combinational circuits. **(4)**

Ans:

Sequential circuits are a part of digital circuit using feed back paths. Due to presence of feed back path, the output of the circuit at any time depends on the present inputs well as previous states available in the circuit. Sequential logic expands combinational logic functions to include the ability to store & latter retrieve binary information memories, Registers, counters & flip flops belongs to sequential circuits.

comparison between sequential ckt & combinational ckt.

Sequential	***Combinational***
(i) Feedback path in circuit	No feed back.
(ii) O/p depends on previous state and present input.	O/p depends on only present inputs
(iii) clock input is required to sequence the operation in time.	Not required
(iv) Circuit behaviour is represented by transition table	Circuit behaviour is represented by truth table.
(v) Circuit design and analysis is complicated.	Circuit design and analysis is easy.

Q.178 Explain the various phases of instruction cycle in a basic computer. **(6)**

Ans:

The program is stored in the memory get executed one instruction after other in a sequential manner. The process carried out by the CPU for execution of each instruction is called instruction cycle. This instruction cycle is subdivided into a sequence of sub cycles or phases. These are

(i) Fetch an instruction from memory.

(ii) Decode the instruction

(iii) Read the effective address from memory if the instruction has an indirect address.

(iv) Execute the instruction.

Upon the completion of step (iv), the control goes back to steo-1 to fetch, decode and execute the next instruction. This process continuous indefinitely unless HALT instruction is encountered. The micro operation that takes place for fetch and decode cy le is as under :-

T_0 : AR ← PC

T_1 : IR ← M[AR}, PC ← PC+1

T_2 : D_0, D_1, ……D_7 ← Decode IR

The necessary control signals are generated by the control unit during T_3, T_4… , for execution of the instruction. The program counter gets incemented during T_1, so as to enable to fetch the next instruction from memory after completion of execution of present instruction.

Q.179 Perform (–35) – (+40) with negative numbers in signed 2's complement representation. **(6)**

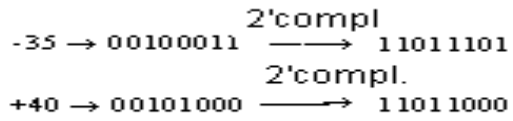
Ans:

-35 – (+40) in 2's complement representation is carried out as under :-

+ 35 = 010001

+ 40 = 0101000

In signed 2's complement representation, both the operands are represented as



As we have to substract (+40) from -35

So the result of substracton = Minuend in signed 2's complement from + 2's complement of substracted.

-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.180 Use restoring method to divide 10100011 by 1011. **(8)**

Ans:

The restoring method of division to divide 10100011 by 1011 is given below. Assume both numbers are positive

	E	A	Q	SC	B	Remark
	0	1010	0011	4	1011	
EA←A + $\overline{B}$+1	0	1111				E=0,so A < B
EA←AA+B	1	1010				
Shl EAQ	1	0100	0110			E=1, A←A+B+1
A←A+ $\overline{B}$+1		1001				
Q_N←1			0111			
SC ←SC-1				3		
	1	1001	0111	3	1011	
Shl EAQ	1	0010	1110			E=1
A←A + $\overline{B}$+1		0111				
Q_N←1			1111			
SC ←SC-1				2		
Shl EAQ	0	1111	1110			E=0
A←A + $\overline{B}$+1	1	0100				E=1
Q_N←1			1111			
SC ←SC-1				1		
Shl EAQ	0	1001	1110			E=0
A←A + $\overline{B}$+1	0	1110				E=0
Q_N←1	1	1001				
SC ←SC-1				0		END

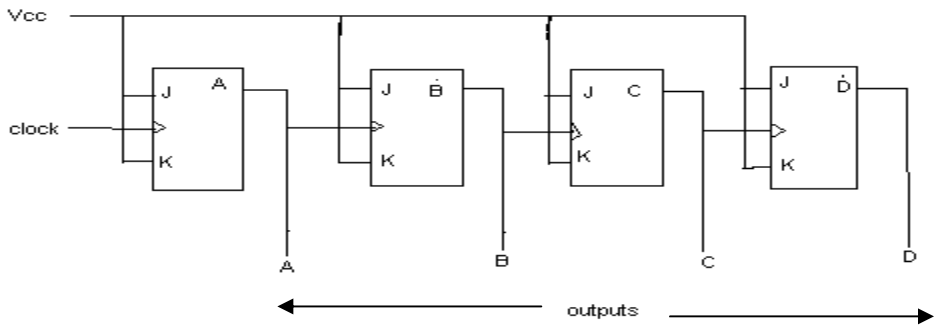
The quotient is in Q i.e $(1110)_2 = (14)_{10}$, Reminder is in a i.e $(1001)_2 = (9)_{10}$

and as we know $163 \div 11 = 14\frac{9}{11}$. Hence the restoring method is correct.

Q.181 Differentiate between synchronous and asynchronous counters. Explain the working of an asynchronous counter. **(6)**

Synchronous counter	***Asynchronous counter***
(i) The change of state of all FF_3 are synchronous with applied clock signal. (ii) As the state change takes place at one instant, so the propagation delay of FFs do not affect the performance. (iii) Design process is structured combinational circuit is designed to obtain required input for FFs. (iv) More reliable	(i) The change of state of all FFs are not synchronous with applied clock signal. The output of some FFs may act as an clock signal for other. (ii) Propagation delay of FFs may introduce some error in states (iii) Design process is not st4rucxtured one mainly combination circuit is designed to obtain the clock signal of FFs. (iv) less reliable.

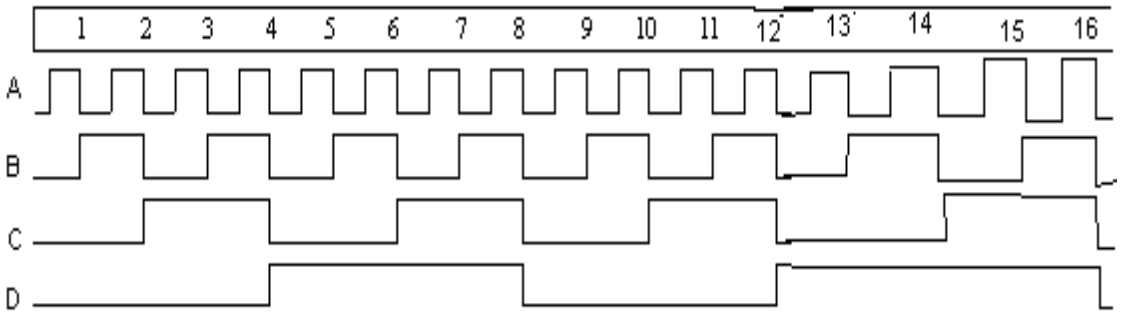
A four bit mod 16 asynchronous counter is shown below :-



clk	***Present state***				***Next state***			
	Q_D	Q_C	Q_B	Q_A	Q_D	Q_C	Q_B	Q_A
1	0	0	0	0	0	0	0	1
2	0	0	0	1	0	0	1	0
3	0	0	1	0	0	0	1	1
4	0	0	1	1	0	1	0	0
……	….	….	….	….	….	….	….	….
….	…	….	…..	….	….	….	….	….

15	1	1	1	0	1	1	1	1
16	1	1	1	1	0	0	0	0 (initial state)

Working Clock signal is connected to a clock input of LSB FF only. The J and K input terminals of each FFs are shorted.J & K are connected to V_{cc}. So the output Q_A toggles with negative transition of clock signal of the LSB FF.The output of A drives B, output of B drives C, and output of C drives D. When the output of a FF is used as clock input for the next flip-flop,we call the counter as asynchrous or ripple counter.In this type of counters all the flip-flops do not change their states at the same time.Because of this the overall propagation time is the sum of the induvisual delays.This counter gives the sequence of counting from 0000 to 1111.The timing diagram is given below:



Q.182 With the help of a neat diagram explain the working of a 4 – bit bidirectional shift register with parallel load. **(8)**

Ans:

A four bit bi-directional shift register with parallel load is shown above. Each state consist of a D-FF Q a 4x1 mux. Two selection inputs S_1l so select one of the multiplexer data input to the D FFs. The selection lines control the mode of operation of the register according to the function table given below :-

Mode control		Register Operation
S_1	S_0	
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load

Q.183 Give the list of microoperations for implementing a subroutine call and return from a subroutine call, using stack. **(6)**

Ans:

When a subroutine is called in the main programme, the processor stores the return address in some specified location. It may be in the stock or in some specified memory location or in some specified register. Then the address of the first instruction of the subroutine is loaded into the program counter and the execution continues till the instruction for return to main program is encountered. On execution of return instruction, the processor loads the return address stored in specified location back to OPC, so that the main program continues its execution.

To implement this subroutine call return from subroutine call using stack the following micro operations are executed

Let CALL be the subroutine call instruction and RET be return to main program instruction.

For CALL SR

T_0 : AR ←PC

T_1 : Fetch: IR ← M[AR], PC ← PC + 1

T_2 : Decode D_0 …. D_7 ← IR (12-15), AR ← IR(0-11)

T_3 : TR← AR , Add. Of subroutine is stored in temporary register.

T_4 : AR ← SP
T_5 : M[AR] ← P, C } Return address is stored in stack where stack is build in R / W memory

T_6 : PC←TR, SC←0, address of first instruction of subroutine is loaded in program counter sequence counter is made zero.

For RET

T_0 : AR← PC ; Program counter is loaded in Add. Register.

T_1 : IR←M[AR], PC← PC+1 ; opcode fetch

T_2 : D_0 …D_7 ← IR[12-15], Decode.

T_3 : AR← SP ; Stack pointer address is transferred to Add. Register

T_4 : DR←M[AR]; M[AR] is transferred to Data Register.

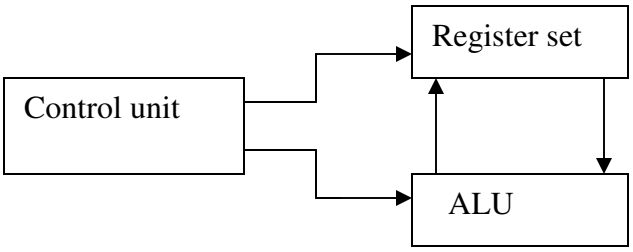
T_5 : PC ← DR, SC←0, Program counter is loaded with return address & sequence counter is made zero.

Q.184 Name the three types of CPU organizations. Explain about Accumulator organization with the help of examples. **(8)**

Ans:

The CPU controls the computer. It fetches instruction from memory, supplying the address and control signals needs by memory to access its data. The CPU decodes the instructions and controls the execution procedure. Internally CPU has three sections

(i) Register set **(ii)ALU** **(iii)Control unit**



Register Set : It includes a set of registers and a bus. The system address and data buses interact with this section of CPU. It also contains some special purpose registers not directly accessible by the programmer. The special purpose registers are – Instruction register, program counter, Address register, Data register, Stock pointer, Index register etc.

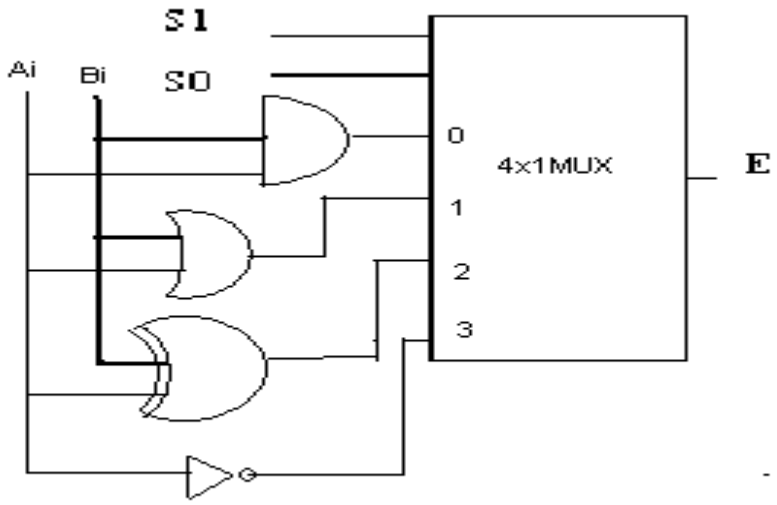
ALU : It performs moist arithmetic and logical operations such as add, subtract, complement, AND, OR, Ex-OR et. It receives its operand from register section of the CPU and stores its result back in register section.

Control Unit ; It controls all the operations of CPU. This unit generates the internal control signals that causes registers to load data, increment or clear their contents and output their contents, as well as cause the ALU to perform the correct function.

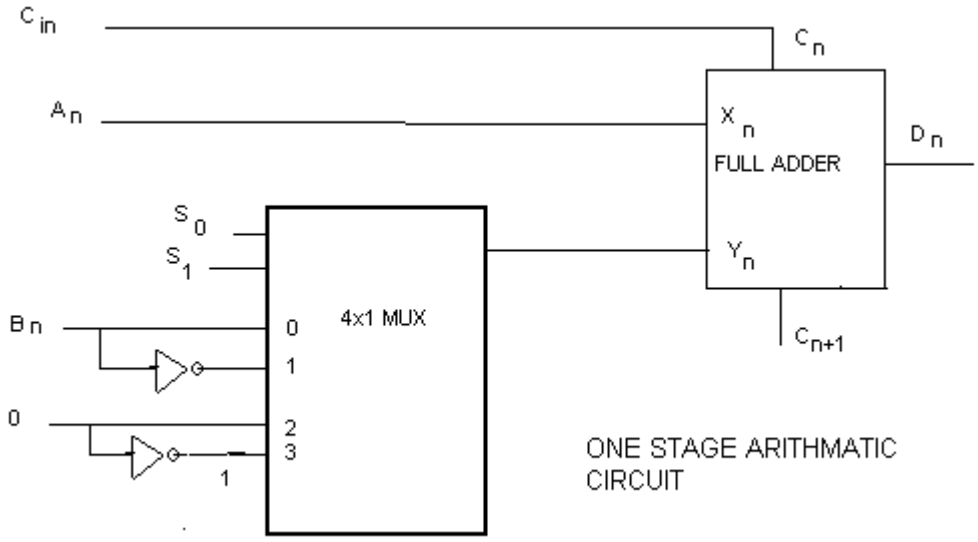
The control unit can be designed in two ways
(i) Hardwired control unit
(ii) Micro-programmed control unit

The hardware circuit for implementing logical operation is :-

S_1	S_0	*Output*	*Operation*
0	0	$E = A \wedge B$	AND
0	1	$E = A \vee B$	OR
1	0	$E = A \oplus B$	X OR
1	1	$E = \overline{A}$	Complement



The hard wave realization for implementing addition, substraction, increment, & decrement is given below (for one stage):



Function Table

S_1	S_0	C_{in}	*Output*	*Operation*
0	0	0	D = A + B	Add
0	0	1	D = A + B + 1	Add with carry
0	1	0	D = A + $\bar{B}$	Substract with borrow
0	1	1	D = A + $\bar{B}$ +1	Substract
1	0	0	D = A	Transfer A
1	0	1	D = A + 1	Increment A
1	1	0	D = A – 1	Decrement A
1	1	1	D = A	Transfer A

Q.185 What are pseudo operations? Name any three and explain about them. **(6)**

Ans:

Pseudo instructions are not assembly language instructions. These are instructions to the assembler. So the assembler does not convert these instructions to machine code. These are symbolic codes which specify the origin of the program, End of the program, type of the data like Decimal/Hex decimal etc.

Example: ORG, END, DEC, HEX

ORG: Its purpose is to specify on origin, that is, the memory location of the next instruction below it.

DEC: Decimal operands are specified following the symbol DCE, E, B, DEC-32, it means. The variable 'B' is a decimal number equal to -32,

END: It specifies the END of the assembly language program.

Q.186 A two-way set associative cache memory uses blocks of four words. The cache can accommodate a total of 2048 words from the main memory. The main memory size is $128\,K \times 32$.

(i) How many bits are there in the tag, index block and word fields of the address format?

(ii) What is the size of the cache memory? **(8)**

Ans:

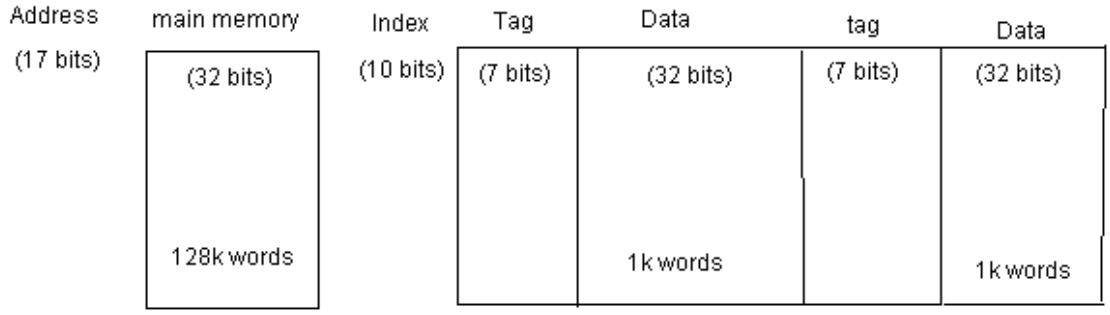
Size of cache memory = 2048 words = 2K words.

Size of main memory = $128\,K \times 32 = 2^7\,K \times 32$

Each word size of main memory = 32 bits.

Address length of main memory = 17 bits

Address length of cache memory = 10 bits (per set)



(i) Tag = 7 bits

Block index = (10-2) = 8 bit.

Word field = 2 bit as each block is of 4 words.

(ii) Word length of cache memory = 2(7+32) = 78 bits, size = (1024×78)

Q.187 What is pipelining? Name the two pipeline organizations. Explain about the arithmetic pipeline with the help of an example. **(6)**

Ans:

Pipe lining professing is an implementation technique where arithmetic sub-operations or the phases of a computer instruction cycle overlap are execution.

Pipe lining is a technique of decomposing a sequential process in to sub operations, with each sub process being executed is a special dedicated segment that operates concurrently with all other segments. A pipe line can be visualized as a collection of processing segments through which binary information flows. Each segment performs partial processing dictated by the way the task is partitioned the result obtained from the computation in each segment is transformed to the next segment in the pipe line. The final result obtained after the data have passed through all segments.

There are different types of pipe lining organizations

(i) Arithmetic pipe line

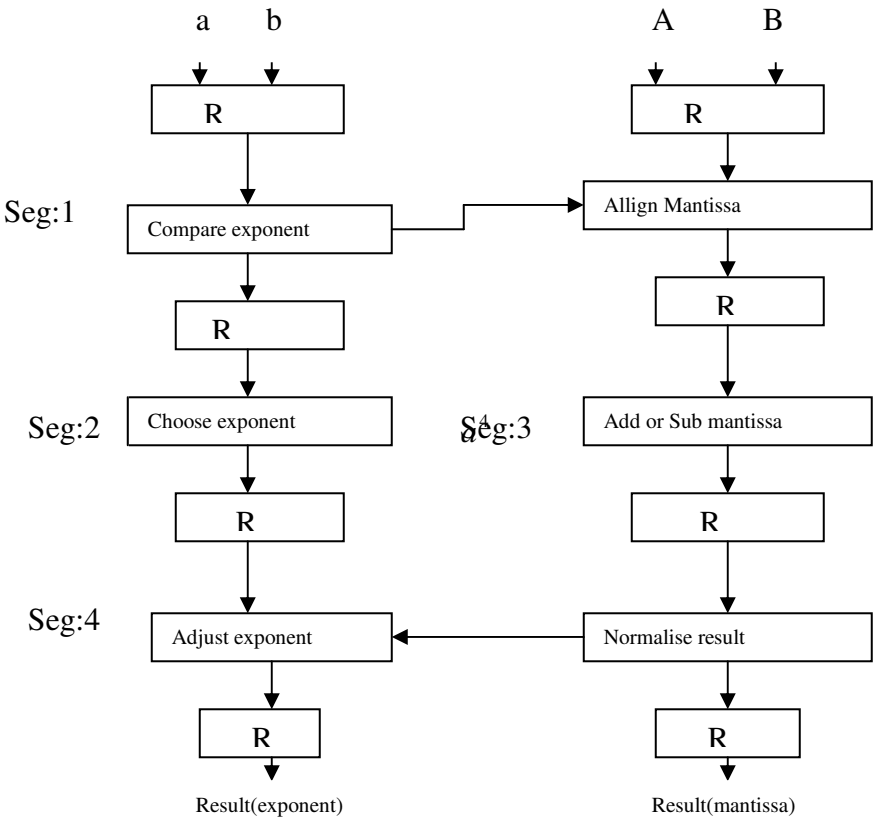
(ii) Instruction pipe line

Arithmetic Pipe Line

Let us consider arithmetic pipe line for floating point adder. Let two input are-

$$X = A \times 2^a$$
$$Y = B \times 2^b$$

Where A & B are two tractions that represents mantissas and a and b are the exponents. Arithmetic addition and subs traction can be performed in four segments.



. As per the flow chart

Segment-1: Two exponent are subtracted, to obtain

3-2 = 1

Segment-2: Longer exponent is selected.

Shift mantissa of Y to the night to obtain

$$X = 0.9504 \times 10^3$$
$$Y = 0.0820 \times 10^3$$

Segment-3: Addition of mantissa order same exponent results.

$$Z = 1.0324 \times 10^3$$

Segment -4: The sum is adjusted by normalizing the result.

$$Z = 0.10324 \times 10^4.$$

Q.188 What are the hazards of instruction pipelining? How are these taken care of. **(8)**

Ans:

There are three major difficulties that causes the instruction pipe line to deviate from its normal operation.

(i) Resource conflict: Caused by access to memory by two segments at the same time. Most of the conflict can be resolved by using separate instruction and data memories.

(ii) Data dependency: This conflict arises when an instruction depends on the result of a pervious instruction, but this result is not yet variable.

(iii) Branch difficulties: This arises from branch & other instructions that changes the value of PC.

The problem of data dependency can be solved through the followings.

Operand forwarding: The hardware avoid the conflict by routing the data through special paths between pipe line segments.

Through Compiler Programs: Insert the No. operation instruction in the program.

Handling branch difficulties: The methods used are –

(i) Prefetch target instructions

(ii) Use of branch target buffer

(iii) Use of loop buffer.

(iv) branch prediction

(v) Delayed branch.

Q.189 Explain the concept of parallel priority logic with four interrupt sources and give the logic diagram. **(8)**

Ans:

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 bits in the register. In addition to the interrupt register, the circuit may include a mark register whose purpose is to control the status of each interrupt request. The mark 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 interrupt registers individual bits are set by internal I_{10} devices requesting the service of CPU, and is cleared by program instructions. The I/O devices are origin

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.

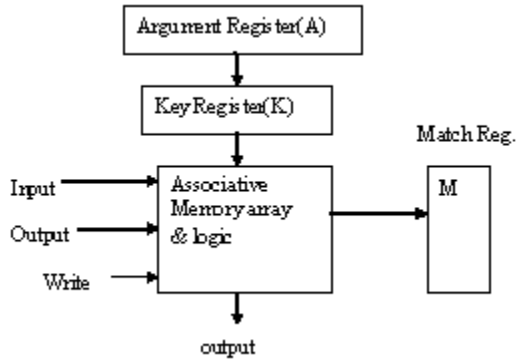
The mark register has same no. of bits as that of interrupt register. By means of program, it is possible to set or next any bit of the mark register. If its is 1, then the associated interrupt is recognized, other wise it is treated to be marked. Each interrupt bit along with its mark bit are applied to a AND gate to 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 executor of ISR darning 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. It $I_{EN} = 1$, than interrupt is recognized by CPU other wise not.

If IST = 1 & IEN = 1. Than the interrupt signal goes to CPU, in return, CPU bends interrupt acknowledgement signal, which enables the vector address register to place the vector address of ISR in to program computer.

Q.190 Draw the logic diagram of the cell of one word in associative memory including the read and write logic. **(6)**

Ans:

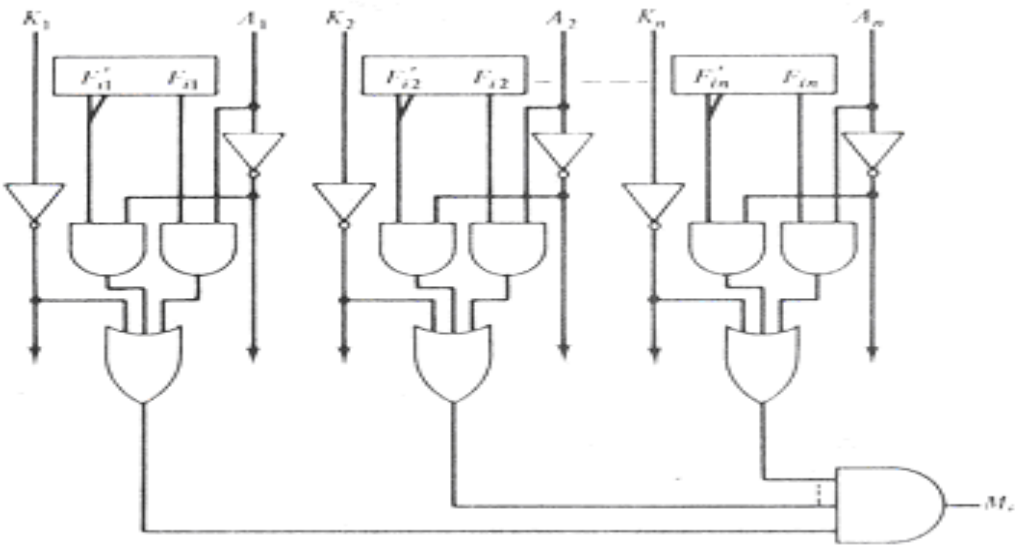
he logic diagram of the cell of one word in associative memory including the read and



write logic is sown 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 ward. The mach 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.

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



Match logic for one word of associative memory.

Q.191 Explain the working of a Microprogram Sequencer with the help of a neat diagram. **(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 wined . Micro programming is the second alterative.

The control function that specifies micro operations is a binary variable. These binary control variables are stored in memory is called a micro programmed control unit. A sequence of micro instructions constitutes a micro program. Each machine instruction initiates a service of micro instructions in control memory. These micro instructions generates the micro operations 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 fetch phase in order to repeat the cycle for new instruction. Control memory address register specifies the address of the micro interaction read from memory. The micro instruction contains a control word that specifies one or more micro operations for the data processes. Once these operations are excreted, the control must determine the next address. The location of the next micro instruction may be the one next in sequence or it may be located some where 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 conditions. The next address is computed by the circuit is called micro program sequences. The typical functions of a micro program sequences are

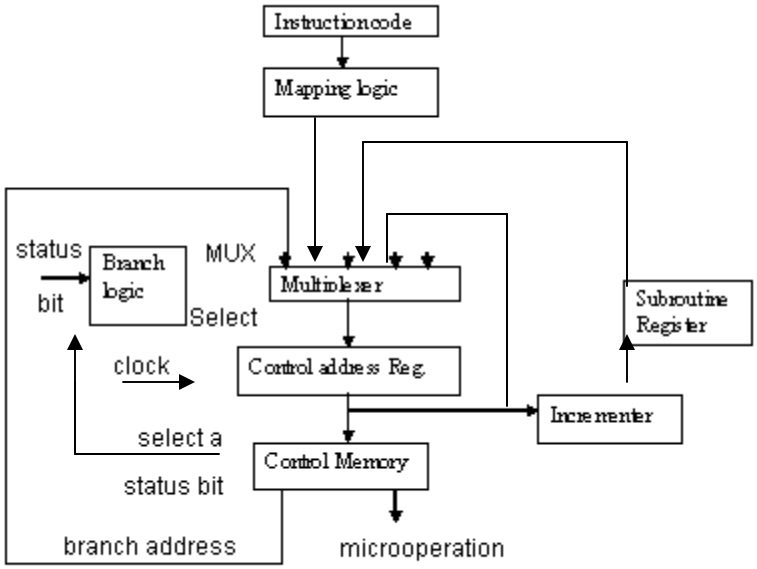
(i) Incrementing the control address registers by one.

(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.



Q.192 Name and explain the fields of a symbolic microinstruction. **(6)**

Ans:
The format of a micro operation is shown below:-

F1	F2	F_3	Condition for branching	Branch field	Address

Where, F_1, F_2, F_3 are micro operation field. The bit pattern of each filed are encoded to specify distinct micro operations. If there are three fields, them it means that no more than three micro operations can be chosen for a micro instruction.If fewer than three micro operations are used, then one or more of the fields will use the binary code for no operation.

Condition field

It consists of bit pattern, which is encoded to specify the status bit condition.

Branch field:

Its bit pattern provides an unconditional branch operation. The branch field bits are used in conjunction with the address field to choose the address of next micro instruction.

Q.193 Write an assembly language program to sort the given list in ascending order. The length of the series is given in the memory location 203FH and the series itself starts from 2040H onwards. **(10)**

Ans:

Length of the series is in memory location -203F H. The series starts from 2040_H on words.

Aim: To arrange the series in ascending order.

Program:-

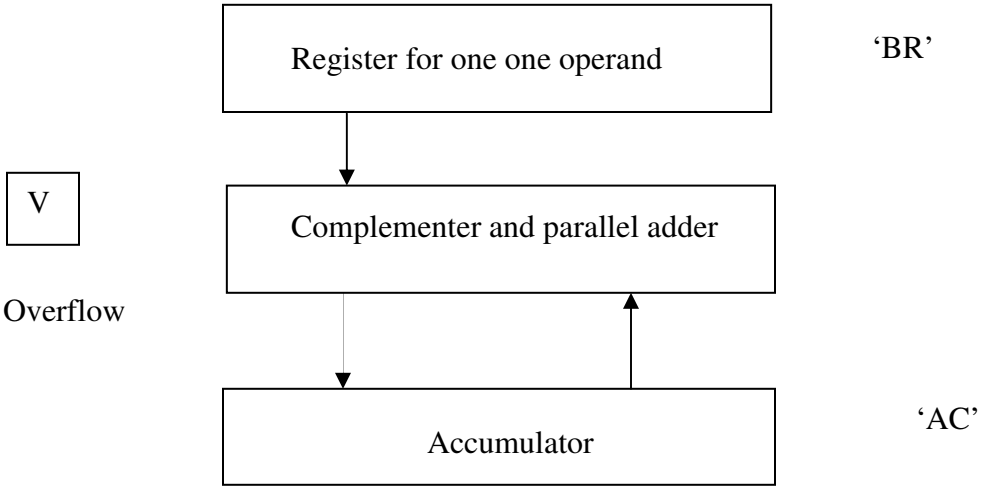
```
LXI  H,203F
MOV B,M        :Length of the series is in reg B
DCR B
INX H
LOOP1: MOV C,B      : (Length of the series -1) is in reg C
LOOP2: MOV A,M   :First element of the series goes to Accumulator
    INX H
    CMP M        :comparison between two elements
    JNC  LOOP3
    MOV D,M
     MOV M,A
     DCX H
     MOV M,D
     INX H
LOOP3:    DCR C
     JNZ  LOOP2
     LXI H,2040
     DCR B
     JNZ LOOP1
     HLT
```

This programme gives the series in ascending order in the same location.

Q.194 Give an algorithm for subtracting numbers in signed 2's complement form. **(4)**

Ans:

The hardware arrangement for signed 2's complement addition & subs-traction is given below.



The algorithm is shown in the chart