UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: **Economics for Engineers** Subject Code: **HU501**
Year: **3rd Year** Semester: **Fifth**

Module Number	Topics	Number of Lectures
1	1. Economic Decisions Making – Overview, Problems, Role, Decision making process. 2.EngineeringCosts&Estimation– Fixed, Variable, Marginal & Average Costs, Sunk Costs ,Opportunity Costs, Recurring And Non recurring Costs, Incremental Costs, Cash Costs vs Book Costs, Life-Cycle Costs; Types Of Estimate, Estimating Models-Per-Unit Model, Segmenting Model, Cost Indexes, Power-Sizing Model, Improvement &Learning Curve, Benefits.	**2L** 5L 3L
2	3. Cash Flow, Interest and Equivalence: Cash Flow Diagrams, Categories & Computation, Time Value of Money, Debtre payment, Nominal & Effective Interest. 4. Cash Flow & Rate Of Return Analysis–Calculations, Treatment of Salvage Value, Annual Cash Flow Analysis, Analysis Periods; Internal Rate Of Return, Calculating Rate of Return, Incremental Analysis; Best Alternative Choosing An Analysis Method, Future Worth Analysis, Benefit-Cost Ratio Analysis, Sensitivity And Break even Analysis. Economic Analysis In The Public Sector – Quantifying And Valuing Benefits & drawbacks.	**2L** 2L 2L
3	5.Inflation And Price Change Definition, Effects, Causes, Price Change with Indexes, Types of Index, Composite vs Commodity Indexes, Use of Price Indexes In Engineering Economic Analysis, Cash Flows that inflate at different Rates. 6. Present Worth Analysis: End-Of Year Convention, View point Of Economic Analysis Studies, Borrowed Money View point, Effect Of Inflation & Deflation, Taxes, Economic Criteria, Applying Present Worth Techniques, Multiple Alternatives. 7. Uncertainty In Future Events-Estimates and Their Use in Economic Analysis, Range Of Estimates, Probability, Joint Probability Distributions, Expected Value, Economic Decision Trees, Risk, Risk vs Return, Simulation, Real Options.	**2L** 4L
4	8. Depreciation - Basic Aspects, Deterioration & Obsolescence, Depreciation And Expenses, Types Of Property, Depreciation Calculation Fundamentals, Depreciation And Capital Allowance Methods, Straight-Line Depreciation Declining Balance Depreciation, Common Elements Of Tax Regulations For Depreciation And Capital Allowances. 9. Replacement Analysis- Replacement Analysis Decision Map, Minimum Cost Life of a New Asset, Marginal Cost, Minimum Cost Life Problems.	**4L** 4L

	10. Accounting–Function, Balance Sheet, Income Statement, Financial Ratios Capital Transactions, Cost Accounting, Direct and Indirect Costs, Indirect Cost Allocation.	
	TOTAL NO. OF HOURS= 36L	

UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

Lecture-wise Plan

Subject Name: Design & Analysis of Algorithm **Subject Code-CS501**

Year: 3rd Year **Semester: Fifth**

Module Number	Topics	Number of Lectures
	Complexity Analysis:	**5L**
	1. Time and Space Complexity, Different Asymptotic notations – their mathematical significance,.	2
1	2. Solve recursive function with different methods	3
	Recursion Techniques:	**2L**
	1. Recursion – definition, use and limitations	1
2	2. Examples – Tower of Hanoi problem, Tail recursion	1
	Algorithm Design Techniques:	
	Divide and Conquer:	**3L**
3.	1. Basic method, use, Examples – Binary Search, Merge Sort and their complexities	2
	2. Quick Sort and itscomplexity	1
	Priority Queue:	**1L**
4	1. Definition, Heap Sort and its complexity.	1
	Dynamic Programming:	**4L**
5	1. Basic method, use, Examples – Matrix Chain Manipulation it's complexity	1
	2. All pair shortest paths(Floyd-Warshall algorithm), single source shortest path(Bellman-ford algorithm), Travelling Salesman Problemand their complexities	3
	Backtracking:	**2L**
6	1. Basic method, use, Examples – 8 queens problem	1
	1. Graph colouring problem.	1
	Greedy Method:	**6L**
	1. Basic method, use, Examples – Knapsack problem, Job sequencing with deadlines, Activity selection problem	3
7	2. single source shortest path (Dijkstra algorithm), Minimum cost spanning tree by Prim's and Kruskal's algorithm, their complexities	3
8	**Lower Bound Theory:**	**1L**
	1. O(nlgn) bound for comparison sort	1
	Disjoint set manipulation:	**1L**
9	1. Set manipulation algorithm like UNION-FIND, union by rank, path compression	1
	Graph traversal algorithm:	**2L**

10	1. Breadth First Search (BFS) and Depth First Search (DFS) – complexity and comparison with different edges	2
11	**Network Flow:**	**2L**
	1. Ford Fulkerson algorithm, Max-Flow Min-Cut theorem (Statement and Illustration)	2
12	**String matching problem:**	**3L**
	1. Different techniques of string matching problem Naïve, String matching using finite automata, Knuth Morris and Pratt algorithm,their complexities.	3
13	**Amortized Analysis:**	**1L**
	1. Aggregate, Accounting, and Potential Method.	1
14	**Matrix Manipulation Algorithm:**	**3L**
	1. Strassen's matrix manipulation algorithm and its application to solution os simultaneous linear equations using LUP decomposition.	2
	2. Inversion of matrix and Boolean matrix multiplication.	1
15	**Notion of NP-completeness:**	**4L**
	1. P class, NP class, NP hard class, NP complete class	1
	2. Their interrelationship, Satisfiability problem, Cook's theorem (Statement only), Clique decision problem, vertex cover problem, Hamiltonian cycle problem	3
16	**Approximation Algorithms:**	**3L**
	1. Necessity of approximation scheme, performance guarantee, polynomial time approximation schemes,	2
	2. Vertex Cover problem, travelling salesman problem	1
Total Number Of Hours = 42		

Faculty In-Charge

HOD, CSE Dept.

Assignment:

Module-1(Complexity analysis):

1. Distinguish between big oh and small oh, big omega and small omega.
2. Solve the recurrence relation:
   a)T(n)=T(n-1)+root(n)
   b)T(n)=2T(n-1)+nlogn

Module-2 (Recursion technique):

1. Write a function of fibbonacci series using tail recursion.
2. Write down the time complexity of tower of hanoi

Module-3(Divide and conquer):

1. Write down the algorithm ofunsuccessful binary search and find out the time complexity.
2. ModifytheMerge sortalgorithmsothattheinput array AisdividedintoKparts instead of2.Analyzeyouralgorithm.AssumeK>1.

Module-4(Priority queue):

1. Write analgorithmtosortanelementinadescendingorder using heapsort.
2. Findoutthetimecomplexityofheapsort.

Module-5(Dynamic programming):

1. Explain bellman ford algorithm with example.
2. Considerthefollowingfivematrices:
   A1=2X3,A2=3X4,A3=4X6,A4=6X2,A5=2X7.
   (i) How manyparenthesizationarepossibletomultiplythese matrices?
   (ii) Give aparenthesizedexpressionfor the order in which this optimalnumberof multiplicationsisachieved.
   (iii) Findtheoptimalcostofthesolution

Module-6(Backtracking):

1. Write down the complexity of N queens problem.
2. Write down the algorithm of graph coloring problem.

Module-7(Greedy approach):

1. Find out the time complexity of prime algorithm and dijkstra algorithm
2. What are the features are present in any greedy algorithm?

Module-8(Lower bound theory):

1. Prove that for any comparison sort the lower bound is O(nlogn)
2. Construct the decision tree for binary search

Module-9(Disjoint set manipulation):

1. Define union and find algorithm
2. How kruskal algorithm follow this method explain with example

Module-10(Graph traversal algorithm):

1. Find out the time complexity of BFS.
2. Write an algorithm to find the graph contain any back edge or not

Module-11(Network flow):

1. Define max flow min cut theorem
2. Explain Ford Fulkerson algorithm with example

Module-12(String matching problem):

1. Explain KMP algorithm with example.
2. Why KMP is better than Naïve and string matching with finite automata.

Module-13(Amortize analysis):

1. Define Aggregate Method Accounting Method and Potential Method**.**

Module-14(Matrix manipulation algorithm):

1. "Strassens'matrix multiplicationisbetter than thenormalmatrix multiplication"-Justify youranswer.
2. Discuss the procedure for strassen's matrix multiplication to evaluate the product of n matrices find the recurrence relation for the same and analyse its time complexity. Is this method an improvement over the conventional matrix multiplication method.

Module-15(Notion of NP completeness):

1. Define P,NP,NP hard, NP complete.
2. Show all NPC problem are solvable in polynomial time

Module-16(Approximation algorithm):

1. What is vertex-cover?
2. Prove that approx-vertex-cover is 2-approximation algorithm

UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

Lecture-wise Plan

Subject Name: Microprocessors & Microcontrollers Subject Code- CS502
Year: 3rd Year Semester: Fifth

Module Number	Topics	Number of Lectures
1	**INTRODUCTION OF:**	**2L**
	1. Review of Digital Electronics	1L
	2. Applications and basic concept of MP and MP based system	1L
2	**8085 ARCHITECTURE & PINS & SIGNALS:**	**4L**
	1. 8085 MP Architecture	1L
	2. Registers# Flags# Stack and Stack pointer	1L
	3. Timing and control unit	1L
	4. 8085 Pins & Signals	1L
3	**ADDRESSING MODES, TIMING DIAGRAMS, INSTRUCTION SET OF 8085:**	**5L**
	1. Sample one byte, two bytes and three byte Instructions and their timing diagram	1L
	2. I/O mapped I/ O & Memory mapped I/ O and Timing diagram of IN and OUT instruction	1L
	3. 8085 Addressing Modes with examples	1L
	4. 8085 Instructions set (data transfer and arithmetic group)	1L
	5. 8085 Instructions set (Logical, jump and machine control)	1L
4	**8085 PROGRAMMING:**	**4L**
	1. Arithmetic programming	1L
	2. Logical programming	1L
	3. Branching and shifting programming	1L
	4. Stack and subroutine	1L
5	**COUNTER AND TIME DELAYCALCULATION OF 8085:**	**3L**
	1. Introduction of counter and time delay	1L
	2. Programming for counter	1L
	3. Programming for delay	1L
6	**8085 INTERRUPTS:**	**3L**
	1. Introduction of various type of Interrupts	1L
	2. Concept of EI, DI, SIM, RIM instructions and examples	1L
	3. Hardware Interrupts including INTR	1L

	(Handshake Interrupt) and INA	
7	**MEMORY INTERFACING:**	**2L**
	1. Memory Chips (27 series and RAM chips)	1L
	2. Memory interfacing	3L
8	**INTERFACING CHIPS:**	**3L**
	1. Programmable peripheral Interface 8255	1L
	2. Programmable peripheral Interface 8259	1L
	3. Programmable peripheral Interface 8237	1L
9	**16-bit PROCESSOR 8086**:	**5L**
	1. Architecture of 8086	1L
	2. Pinout diagram of 8086	1L
	3. Addressing mode with examples	1L
	4. Instruction sets with examples	1L
	5. Interrupts of 8086	1L
10	**8051 FAMILY OF MICROCONTROLLER:**	**6L**
	1. Introduction and Overview of 8051 family	1L
	2. Architecture, Register Banks & SFRs	1L
	3. Pins & signals of 8051	1L
	4. Memory organization & External memory access	1L
	5. Overview of 8051 instructions & sample programs	1L
	6. Timers and counters	1L

Faculty In-Charge HOD, ECE Dept.

Assignment:

Module-1:

1. What is a difference between microprocessor and microcontroller?
2. What is a difference between latch and flipflop?
3. Discuss the evolution tree of general purpose processor.
4. What is a tri state buffer? Explain briefly.
5. Why microprocessor is called a “Micro” processor?
6. Discuss the operation of RAM and ROM with proper diagram.

7. Describe the general architecture of microprocessor.

Module-2:

1. Discuss about the Flag register of 8085 microprocessors.
2. Describe each function of every general-purpose register.
3. Draw the architectureof 8085 microprocessors and explain?
4. Using 74LS138 draw and explain the interfacing of memory and I/O device.
5. What are the function of ALE, HOLD, READY, s0, s1 and Interruptpin?

Module-3:

1. Draw the timing diagram of IN and OUT instruction and explain.
2. What is a difference between memory mapped I/O and peripheral mapped I/O?
3. What is a difference between absolute and partial decoding?
4. Describe the addressing mode of 8085 microprocessors.
5. How to optimize the instruction format of 8085?

Module-4:

1. Write an assembly language program to add two 8-bit numbers.
2. Write an assembly language program to add two 8-bit BCD number.
3. Write an assembly language program to add two 8-bit BCD Number without using DAA instruction.
4. Write an assembly language program to subtraction two 8-bit number without using SUB instruction.
5. Write an assembly language program to add two 16 bit numbers.
6. Write an ALP of 8085 to arrange the six 8 bits random numbers in ascending order by using subroutine.

Module-5:

1. Write an ALP to generate 1 sec delay.
2. Write an ALP to generate a 20 khz square wave.
3. Write an ALP to generate a 20 khz triangular wave.
4. The following sequences of instructions are executed by 8085 microprocessor:

```
C000   LXI SP, D050H
C003   POP H
C004   XRA A
C005   MOV A, H
C006   ADD L
C007   MOV H, A
C008   PUSH H
C009   PUSH PSW
C00A   HLT
```

D050	05
D051	40
D052	52
D053	03

D054	XX

What are the contents of Stack Pointer, Program Counter, Accumulator and HL pair?

5. The following sequence of instructions are executed by an 8085 microprocessor:

```
C000   LXI SP, D7FFH
C003   CALL C008H
C006   POP D
C008   POP H
```

What are the contents of the SP and HL register pair after execution the above program?

Module-6:

1. What is interrupt? Why interrupt is very important in 8085 microprocessors?

2. What is a different between maskable and non maskable interrupt?

3. Draw the timing diagram of RESTART instruction.
4. Explain the operation of RIM and SIM instruction.
5. What is the vector and non-vector interrupt?
6. Describe the interrupt process of 8085 up.

Module-8:

1. What do you mean by Mode 0, Mode 1 and Mode 2 operation of 8255 PPI?
2. Discuss the control word format in the BSR Mode of 8255 PPI.
3. In Mode 1 operation of 8255 PPI, what are the control signals when port A and B acts as output ports? Discuss the control signals.
4. Discuss about the DMA data transfer scheme of 8085.
5. What is pulling device? Why it is very important?
6. Explain the function of 8259 programable interrupt controller.

Module-9:

1. What are the main functions of BIU and EU of 8086? How does the separation in units speed up the processing?

2. Discuss the addressing mode of 8086 microprocessors.

3. Draw the architecture of 8086 and explain the function of it’s all registers.

4. How does 8086 follow the pipeline architecture?

5. How does 8086 generate physical address?

Module-10:

1. Draw the architecture of 8051microcontroller and explain it.

2. Discuss the addressing mode of 8051.
3. Explain the Flag register of 8051 microcontroller with example.
4. What is difference between shot jump and long jump of 8051?
5. What is difference between ACALL and SCALL instruction?
6. Write an assembly language program to add two 8-bit numbers.
7. Write an assembly language program to subtraction two 8-bit numbers.

UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

Lecture-wise Plan

Subject Name: Discrete Mathematics **Subject Code-CS503**

Year: 3rd Year **Semester: Fifth**

Module Number	Topics	Number of Lectures
1.	**Introduction to Propositional Calculus:**	**10L**
	Propositions, Logical Connectives,Conjunction, Disjunction, Negation and their truth table	2
	Conditional Connectives, Implication, Converse, Contrapositive, Inverse, Biconditional statements with truth table	2
	Logical. Equivalence, Tautology, Normal forms-CNF, DNF; Predicates and Logical Quantifications of Propositions and related examples.	6
2.	**Theory of Numbers:**	**10L**
	Well Ordering Principle, Divisibilitytheory and properties of divisibility	2
	Fundamental theorem of Arithmetic;Euclidean Algorithm for finding G.C.D and some basic properties of G.C.D with simple examples	2
	Order, Relation and Lattices: POSET, Hasse Diagram, Minimal , Maximal, Greatest and Least elements in a POSET, Lattices and its properties, Principle of Duality, Distributive and Complemented Lattices	6
3.	**Counting Techniques:**	**10L**
	Permutations, Combinations, Binomial coefficients, Pigeon- hole Principle	2
	Principles of inclusion and exclusions; Generating functions, Recurrence Relations and their solutions using generating function, Recurrence relation of Fibonacci numbers and it's solution	4
	Divide-and-Conquer algorithm and its recurrence relation and its simple application in computer.	4
4.	**Graph Coloring:**	**6L**
	Chromatic Numbers and its bounds, Independence and Clique Numbers	1
	Perfect Graphs-Definition and examples, Chromatic polynomial and its determination, Applications of Graph Coloring.	2
	Matchings: Definitions and Examples of Perfect Matching, Maximal and Maximum Matching, Hall's Marriage Theorem (Statement only) and related problems	3

Assignment:
Module-1:

1. Represent as propositional expressions:
   Tom is a math major but not computer science major
   P: Tom is a math major
   Q: Tom is a computer science major

Use De Morgan's Laws to write the negation of the expression, and translate the negation in English

2. Let
   P = "John is healthy"
   Q = "John is wealthy"
   R = "John is wise"
   Represent:
   John is healthy and wealthy but not wise: $P \land Q \land \neg R$
   John is not wealthy but he is healthy and wise: $\neg Q \land P \land R$
   John is neither healthy nor wealthy nor wise: $\neg P \land \neg Q \land \neg R$
   .
3. Translate the sentences into propositional expressions:
   "Neither the fox nor the lynx can catch the hare if the hare is alert and quick."
4. Given a conditional statement in English,
   i. translate the sentence into a logical expression
   ii. write the negation of the logical expression and translate the negation into English
   iii. write the converse of the logical expression and translate the converse into English

Module-2:

1. Prove that $\mathbf{3^n > n^2}$ for n = 1, n = 2 and use the mathematical induction to prove that $3^n > n^2$ for n a positive integer greater than 2.
2. Prove that for any positive integer number n , $n^3 + 2n$ is divisible by 3

3. Use mathematical induction to prove that

   $1^3 + 2^3 + 3^3 + ... + n^3 = n^2 (n + 1)^2 / 4$

   for all positive integers n.
4. In a room of 50 people whose dresses have either red or white color, 30 are wearing red dress, 16 are wearing a combination of red and white. How many are wearing dresses that have only white color?

Module-3:

1. Out of 7 consonants and 4 vowels, how many words of 3 consonants and 2 vowels can be formed?
2. In a group of 6 boys and 4 girls, four children are to be selected. In how many different ways can they be selected such that at least one boy should be there?
3. From a group of 7 men and 6 women, five persons are to be selected to form a committee so that at least 3 men are there on the committee. In how many ways can it be done?
4. In how many different ways can the letters of the word 'OPTICAL' be arranged so that the vowels always come together?
5. In how many different ways can the letters of the word 'CORPORATION' be arranged so that the vowels always come together?

Module-4:

1. Prove that any planar graph has an edge coloring of at most three colors in which adjacent edges of the same color are allowed but cycles of edges of the same color are not.
2. If G is a graph and H is any subgraph of G, $\chi(G) \geq \chi(H)$.
3. State and prove four coloring theorem.

UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

Lecture-wise Plan

Subject Name: Circuit Theory & Network **Subject Code- CS504A**
Year: 3rd Year **Semester: Fifth**

Module Number	Topics	Number of Lectures
1	**Introduction:**	**4L**
	1. Introduction todifferent types of signals-continuous and discrete, different types of systems .	1
	2. Introduction to linear , non -linear, lumped, distributed, passive, active, lateral, bi-lateral elements , networks	1
	3. Assumptions made circuit theory and network and general explanation of KCL, KVL	1
	4. Explaining current divider, voltage divider rule, method of solving network using KCL, KVL	1
2	**Magnetically coupled circuits**	**3L**
	1. Magnetic coupling, Polarity of coils, Polarity of inducedvoltage, Concept of Self and Mutual inductance, Coefficient of coupling. Modeling of coupled circuits,	1
	2. Concept of self and mutual inductance, co-efficient of coupling	1
	3. Modelling of coupled circuits	1
3.	**Laplace transform:**	**5L**
	1. Significance of Laplace transform. 2. Analysis of Impulse, Step & Sinusoidal response of RL, RC, and RLC circuits with respect to Laplace transform.	2
	3. Transient analysis of different electrical circuits with and without initialconditions.	3
4	**Fourier method and waveform analysis**	**8L**
	1. Significance of Fourier series and Fourier transform	1
	2. Difference of Fourier and Laplace transform	1
	3. Application of Fourier series in different types signals.	3
	4. Application of Fourier transform to solve circuit theory problems	3
5	**Network theorems**	**9L**
	1. Formulation of network equations, Source transformation,Loop variable analysis, Node variable analysis. Assumptions made in solving Network problems	1
	2. Problems with DC & AC sources involving:	
	A. Superposition Theorem	1

	B. Thevenin and Norton theorem.	2
	C. Maximum Power transfer theorem.	1
	D. Millman theorem	1
	E. Tellegen theorem	1
	Additional problem solving involving Network theorems	2
6	**Graph theory:**	**5L**
	1. Concept of Tree, Branch, Tree link	1
	2. Incidence matrix, Tie-set matrix and loop currents	2
	3. Cut set matrix and node pair Potentials, network equilibrium equations	2
7	**Two port network analysis:**	5L
	1. Open circuit Impedance & Short circuit Admittance parameter, Z-parameter, Y-parameter	**1**
	2. Transmission line parameters	1
	3. Hybrid parameters	1
	4. Inter-relations between the parameters, Driving point impedance and admittance	2
8	**Filter Circuits**	**6L**
	1. Analysis and synthesis of filters, general properties and types of filters.	2
	1. Low pass, High pass, Band pass, Band 2. reject,	2
	3. Active filters	2
Total Number Of Hours = 45		

Faculty In-Charge HOD, EE Dept.

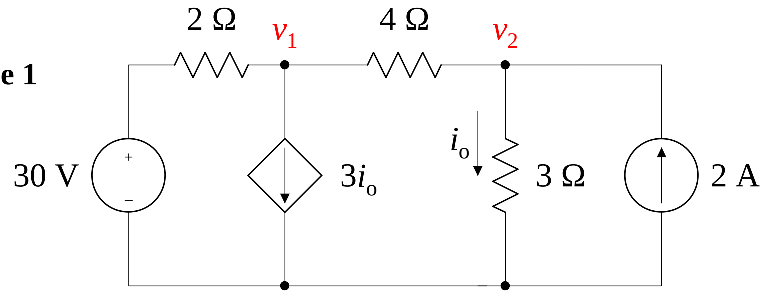
Lecture-wise Plan

Assignment:1

Module-1

1. In the circuit of Figure 1, find the current i_o using nodal analysis.

Figure 1



2. **Explain invertible system. Why it is important to have an inverse of a system?**
3. **What are the conditions for a system to be a linear system?**

Module-2:

1. **In the following circuit as shown in Fig.2, k=1 find I_1 and I_2**

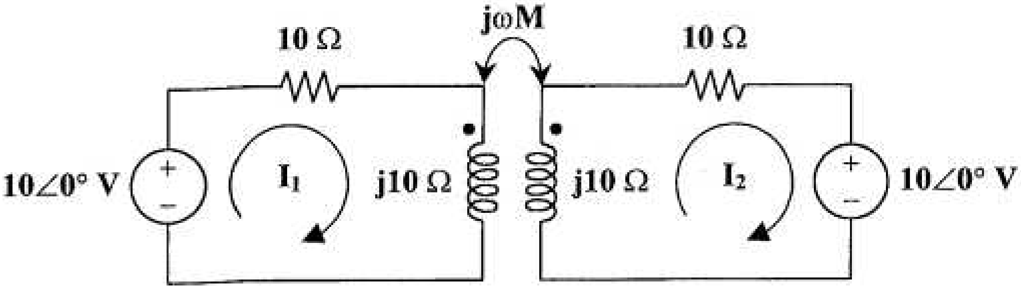


Figure 2

2. **For the three coupled coils (Figure 3) calculate the total inductance**

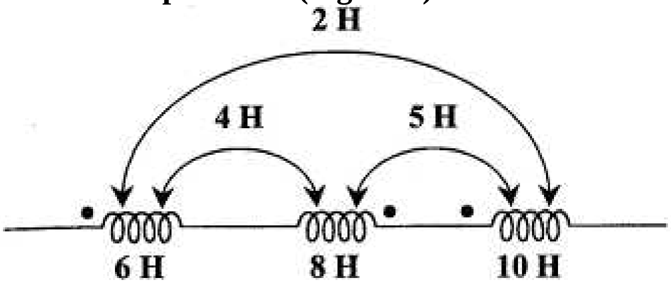
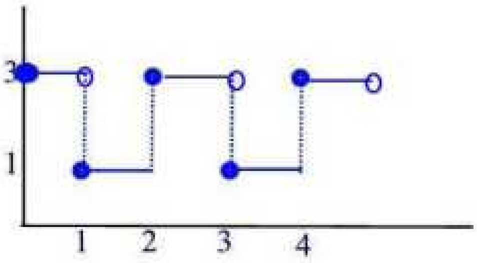


Figure 3

Module-3:

1. Find the Laplace transform of the given function , Fig.4



2. The circuit was initially in steady state with the switch in position ‘a’. At t=0+ it goes from ‘a’ to ‘b’, find the expression for voltage $V_o(t)$ at t>0.

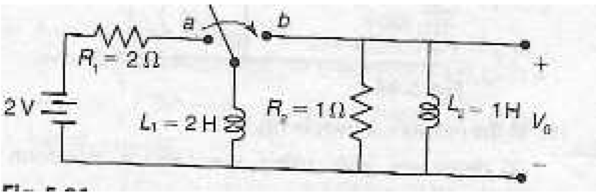


Fig.5

Module-4:

1. Find the Fourier for a train of pulses given by the equation
   V(t) = V; 0<t<T/2
   =0; T/2<t<T

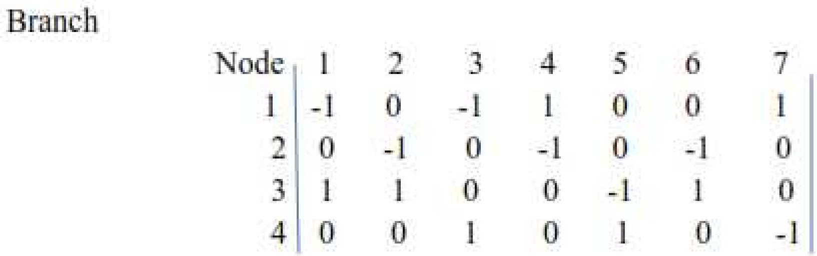
2. When a complex wave is applied to a pure inductor, the current wave has lesser harmonics than the applied voltage . Explain, why?

Module-5:

1. Explain the applications and limitations of Millmans’ theorem.
2. Mention the salient features of Tellegens’ theorem.

Module-6:

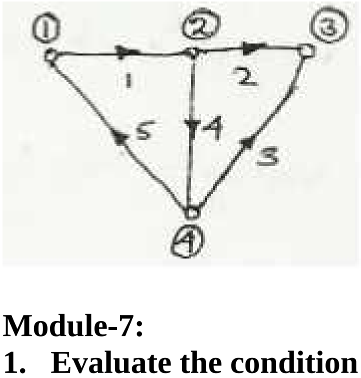
1. Draw the directed graph from the following incidence matrix



Branch

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

2. **Determine the tie-set matrix of the following graph. Also find the equation of the branch current and voltages.**



Module-7:

1. **Evaluate the condition of reciprocity for a two port network in terms of**
   (a) Z- parameters
   (b) Y-parameters
   (c) ABCD parameters
   (d) Hybrid parameters

Module-8:

1. **Design a band pass filter with cut off frequencies of 160 Hz and 8 kHz. The load for the circuit is 1 MΩ.**

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Data Communication — Subject Code-CS504B
Year: 3rd Year — Semester: Fifth

Module Number	Topics	Number of Lectures
1	**Data Communication Fundamentals:**	**10L**
	Layered Network Architecture; Data and Signal; Guided Transmission Media; Unguided Transmission Media; Transmission Impairments and Channel Capacity; Transmission of Digital Signal; Analog Data to Analog Signal; Digital Data to Analog Signal; Multiplexing of Signals: The telephone system and DSL technology; Cable MODEM and SONET	10
2	**Data Link control:**	**6L**
	Interfacing to the media and synchronization; Error Detection and Correction; Flow and Error control; Data Link Control.	6
3	**Switching Communication Networks:**	**8L**
	Circuit switching; Packet switching; Routing in packet switched networks; Congestion control in packet switched networks; X.25; Frame Relay; Asynchronous Transfer Mode Switching (ATM).	8
4	**Broadcast communication networks:**	**13L**
	Network Topology; Medium Access Control Techniques; IEEE CSMA/CD based LANs; IEEE Ring LANs; High Speed LANs – Token Ring Based; High Speed LANs – CSMA/CD based; Wireless LANs; Bluetooth; Cellular Telephone Networks; Satellite Networks.	10
	Network Security: Cryptography; Secured Communication; Firewalls.	3
Total Number Of Hours =37		

Assignments:
Module-1:

1. Write down the functions of OSI Layers
2. What will be SNR value in case of noiseless channel?
3. Define Bandwidth? Create the relationship between Bit Rate and Baud Rate?
4. Write down the names of network impairments?
5. Write down the features and basic components of a computer network

6. What kind of topology is well suited for university or college environment?
7. Why we need layered architecture?
8. What will be the channel capacity of a noisy channel having SNR value= 20dB and Bandwidth=3 KHz?

Module-2:

1. What is the significance of sequence number in Stop & Wait ARQ protocol?
2. Discuss Stop & Wait ARQ with 010101 bit sequence?
3. In Selective-Repeat ARQ, sender window size > 2m–1." Is it correct? Justify.
4. Suppose a sender is using sliding window protocol of window size 15. What will be the window status for the following occurrence? Sender has sent packets 0 to 11 and has received NAK 6.

Module-3:

1. Differentiate between circuit switching and packet switching.
2. Write short notes on the following topic:
   A. Frame Relay
   B. X.25
   C. ATM
3. Why packet switching is connection less?

Module-4:

1. Discuss CSMA/CA with the help of a flowchart.
2. Why CSMA/CD is not implemented in WLAN?
3. Describe 802.3 header formats. Why padding is required?
4. Describe Bluetooth Architecture.
5. Differentiate between Token Ring and Token Bus.
6. What do you understand by data privacy? How can authentication, integrity and non-repudiation be implemented by Digital Signature?
7. Define Firewall? Discuss all types of Firewall.

UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

Lecture-wise Plan

Subject Name: **Digital Signal Processing** **Subject Code-CS504C**

Year: 3rdYear, **Semester: Fifth**

Module Number	Topics	Number of Lectures
1.	**Discrete-time signals:**	**6L**
	1. Concept of discrete-time signal, basic idea of sampling and reconstruction of signal	2
	2. sampling theorem, sequences – periodic, energy, power	2
	3. unit-sample, unit-step, unit-ramp, real & complex exponentials, arithmetic operations on sequences.	2
2.	**LTI Systems:**	**6L**
	1. Definition, representation, impulse response, derivation for the output sequence, concept of convolution	2
	2. graphical, analytical and overlap-add methods to compute convolution supported with examples and exercises	2
	3. properties of convolution, interconnections of LTI systems with physical interpretations, stability and causality conditions, recursive and non-recursive systems	2
3.	**Z-Transform**	**6L**
	1. Definition, mapping between s-plane and z-plane, unit circle, convergence and ROC, properties of Z-transform	2
	2. Z-transform on sequences with examples and exercises, characteristic families of signals along with ROCs, convolution, correlation and multiplication using Z-transform, initial value theorem, Perseval's relation,	2
	3. inverse Z-transform by contour integration, power series & partial-fraction expansions with examples and exercises	2
4.	**Discrete Fourier Transform:**	**6L**
	1. Concept and relations for DFT/IDFT, Twiddle factors and their properties, computational burden on direct DFT, DFT/IDFT as linear transformations, DFT/IDFT matrices	2
	2. computation of DFT/IDFT by matrix method, multiplication of DFTs, circular convolution, computation of circular convolution by graphical, DFT/IDFT and matrix methods,	2
	3. linear filtering using DFT, aliasing error, filtering of long data sequences – Overlap-Save and Overlap-Add methods with examples and exercises	2
5.	**Fast Fourier Transform:**	**6L**
	1. Radix-2 algorithm, decimation-in-time, decimation-in-frequency algorithms	2
	2. signal flow graphs, Butterflies, computations in one place, bit reversal,	2
	3. examples for DIT & DIF FFT Butterfly computations and exercises	2
6.	**Filter Design**	**6L**
	1. Basic concepts of IIR and FIR filters, difference equations	2
	2. design of Butterworth IIR analog filter using impulse invariant and bilinear transforms,	2
	3. design of linear phase FIR filters, no. of taps, rectangular, Hamming and Blackman windows.	2

7.	**Digital Signal Processor**	**6L**
	1. Elementary idea about the architecture and important instruction sets of TMS320C 5416/6713 processor	2
	2. writing of small programs in Assembly Language, FPGA:Architecture, different sub-systems,	2
	3. design flow for DSP system design, mapping of DSP algorithms onto FPGA	2
Total Number Of Hours = 42		

Faculty In-Charge HOD, CSE Dept.

Assignment :

Module -1 (Discrete-time signals)

Module -2 (LTI Systems)

Module -3 (Z-Transform)

Module -4 (Discrete Fourier Transform)

Module -5 (Fast Fourier Transform)

Module -6 (Filter Design)

Module -7 (Digital Signal Processor)

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Object Oriented Programming Using Java **Subject Code-CS504D**
Year: 3rd Year **Semester: Fifth**

Module Number	Topics	Number of Lectures
1	**Introduction:**	**6L**
	1. Concepts of object oriented programming language, Major and minor elements, Object, Class.	2
	2. Relationships among objects, aggregation, links.	2
	3. Relationships among classes association, aggregation, using, instantiation, meta-class, grouping constructs.	2
2	**Object oriented concepts:**	**3L**
	1. Difference between OOP and other conventional programming – advantages and disadvantages.	1
	2. Class, object, message passing, inheritance, encapsulation, polymorphism.	2
3.	**Class & Object proprieties:**	**11L**
	1. Basic concepts of java programming – advantages of java, byte-code & JVM.	2
	2. Data types, access specifiers, operators, control statements & loops, array, creation of class, object, constructor, finalize and garbage collection.	2
	3. Use of method overloading, this keyword, use of objects as parameter & methods returning objects, call by value & call by reference	2
	4. Static variables & methods, garbage collection, nested & inner classes	2
	5. Basic String handling concepts- String (discuss charAt() , compareTo(), equals(), equalsIgnoreCase(), indexOf(), length() , substring(), toCharArray() , toLowerCase(), toString(), toUpperCase() , trim() , valueOf() methods) & StringBuffer classes (discuss append(), capacity(), charAt(), delete(), deleteCharAt(), ensureCapacity(), getChars(), indexOf(), insert(), length(), setCharAt(), setLength(), substring(), toString() methods).	2
	6. Concept of mutable and immutable string, command line arguments, basics of I/O operations – keyboard input using BufferedReader & Scanner classes.	1
	Reusability properties	**4L**

4	1. Super class & subclasses including multilevel hierarchy, process of constructor calling in inheritance.	1
	2. Use of super and final keywords with super() method, dynamic method dispatch.	1
	3. Use of abstract classes & methods, interfaces.	1
	4. Creation of packages, importing packages, member access for packages.	1
5	**Exception handling & Multithreading**	**10L**
	1. Exception handling basics.	1
	2. Different types of exception classes, use of try & catch with throw, throws & finally, creation of user defined exception classes.	3
	3. Basics of multithreading, main thread, thread life cycle, creation of multiple threads, thread priorities.	3
	4. Thread synchronization, inter-thread communication, deadlocks for threads, suspending & resuming threads.	3
6	**Applet Programming (using swing)**	**7L**
	1. Basics of applet programming, applet life cycle, difference between application & applet programming	1
	2. Parameter passing in applets, I/O in applets, use of repaint(), getDocumentBase(), getCodeBase() methods.	1
	3. Concept of delegation event model and listener, layout manager (basic concept), creation of buttons (JButton class only) & text fields.	5

Faculty In-Charge HOD, CSE Dept.

Assignment:

Module-1(Introduction):

1. Explain different properties of object oriented programming language.

Module-2 (Object oriented concepts):

1. Advantages and disadvantages of java over C and C++.
2. Explain with examples: encapsulation, polymorphism.

Module-3(Class & Object proprieties):

1. Explain different steps of java source code compilation and execution.
2. Why java is called platform independent programming language.
3. Explain with examples different access specifiers of java.
4. Explain finalize and garbage collection of java.
5. Explain the significant of static keyword.
6. String vs StringBuffer class.

Module-4(Reusability properties):

1. Explain different inheritance with examples.
2. Explain uses of this, this(), super, super().
3. Abstract class vs interface.
4. Member access for packages.

Module-5(Exception handling & Multithreading):

1. Different ways of exception handling.
2. Different ways of implementing concept of multithreading.
3. Discus problems in multithreading and their solutions.

Module-3(Applet Programming (using swing)):

1. Benefits of applet.
2. Different programs with applet.
3. Different components of swing.
4. Different event handling and layouts in swing.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Design & Analysis of Algorithm Lab
Course Code: CS591
L-T-P scheme: 0-0-3 **Course Credit: 2**

Objectives:

1. To learn Evaluation ofalgorithm: Computational complexity, order notations, recurrences.
2. To learn Sorting:Insertion sort, merge sort, Quick sort, Linearsort, priority queue.
3. To learn greedy method: Single-source shortest path problem and minimum spanning tree.
4. To learn dynamic programming techniques: Fibonacci, single and all pair shortest paths, knapsack.
5. To learn Graph: representation and algorithms, Breadth-first search(BFS), Depth-first search(DFS), topological sorting.

Learning Outcomes: The students will have a detailed knowledge of the concepts of different type of algorithm techniques, and the analysis of algorithm. Upon the completion of design and analysis of algorithm practical course, the student will be able to:

- Understand which algorithm is used in an applicationto Job scheduling in OS, applicationto Online gamesChess, Sudoku and application to string matching in word processor etc.
- Analyse which algorithm is used to Routing in network,to Range assignment, TSP.

Course Contents:
Exercises that must be done in this course are listed below:
Exercise No.1:
>Implement Binary Search using Divide and Conquer approach
> Implement Merge Sort using Divide and Conquer approach
Exercise No.2**:**
>Implement Quick Sort using Divide and Conquer approach
> Find Maximum and Minimum element from an array of integer using Divide and Conquer approach
Exercise No.3**:**
>Find the minimum number of scalar multiplication needed for chain of matrix
Exercise No.4**:**
>Implement all pair of Shortest path for a graph (Floyed- WarshallAlgorithm)
>Implement Single Source shortest Path for a graph (Bellman Ford Algorithm)
Exercise No.5**:**
>Implement 15 Puzzle Problem
Exercise No.6**:**
>Implement 8 Queen problem
>Graph Coloring Problem
Exercise No.7**:**
>Knapsack Problem orJob sequencing with deadlines
>Implement Single Source shortest Path for a graph (Dijkstra Algorithm)
Exercise No.8**: (implement any one of the following problem):**
>Minimum Cost Spanning Tree by Prim's Algorithm
>Minimum Cost Spanning Tree by Kruskal's Algorithm
Exercise No.9**: (implement any one of the following problem):**
>Implement Breadth First Search (BFS)
>Implement Depth First Search (DFS)
Exercise No.10**:**
>Implement Naïve algorithm for string matching.

Text Book:

1. T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, “Introduction to Algorithms”, 3rd edition, PHI.

2. E.Horowitz and Shani “Fundamentals of Computer Algorithms”, 2nd edition, Orient Black Swan.

1. Intel based desktop PC with minimum of 166 MHZ or faster processor with at least 64 MB RAM and 100 MB free disk space.
2. Turbo C or TC3 complier in Windows XP or Linux Operating System.

Exercise No.1:
>Implement Binary Search using Divide and Conquer approach
> Implement Merge Sort using Divide and Conquer approach

Description:

Divide and Conquer Method: Divide-and-conquer paradigm consists of following major phases:

- Breaking the problem into several sub-problems that are similar to the original problem but smaller in size,
- Solve the sub-problem recursively (successively and independently), and then combine these solutions into sub problems to create a solution to the original problem.

Aim: Implement Binary Search with using recursion

Algorithm:

```
 BinarySearch_Right(A[0..N-1], value, low, high) {
     // invariants: value >= A[i] for all i < low
                   value< A[i] forall i > high
     if(high < low)
        returnlow
     mid = low +((high – low) / 2)
     if(A[mid] > value)
        returnBinarySearch_Right(A, value, low, mid-1)
     else
        returnBinarySearch_Right(A, value, mid+1, high)
  }
```

Program:

```
#include<stdio.h>
#include<conio.h>
voidbinarysearch(int x[],intlb,intub,a);
void main()
{
          int x[20],n,i,a;
          clrscr();
          printf("Enter the value of n and enter the no of elements in sorted      manner:\n");
          scanf("%d",&n);
          for(i=0;i <n;i++)
          {
                    scanf(“%d”,&x[i]);
          }
          printf("The sorted list is:");
          for(i=0;i <n;i++)
          {
                    printf("%d ", x[i]);
          }
          printf("Enter the searching element:\n");
```

```
        getch();
}

voidbinarysearch(int x[20],intlb,intub,a)
{
        int mid=(lb+ub)/2;
        if(lb<=ub)
        {
                if(x[mid]==a)
                {
                        printf("Elements is found at position %d",(mid+1));
                }
                else if(x[mid]<a)
                {
                        return(binarysearch(x,mid+1,ub,a);
                }
                else
                {
                        return(binarysearch(x,lb,mid-1,a);
                }
        }
        else
                {
                        printf("Elements is not found");
                }
}
```

Output:

```
Enter the value of n and enter the no of elements in sorted manner:
4
1 2 3 4
The sorted list is:
1 2 3 4
Enter the searching element:
4
Elements is found at position 4
```

Explanation:

The worst-case and average-case time complexity for binary search is O(log n). The best-case is O(1).

Aim:Implement Merge Sort using Divide and Conquer approach

Algorithm:

```
MERGE-SORT (A, p, r)
IF p < r                                   // Check for base case
    THEN q = FLOOR[(p + r)/2]              // Divide step
        MERGE (A, p, q)                    // Conquer step.
        MERGE (A, q + 1, r)                // Conquer step.
        MERGE (A, p, q, r)                 // Conquer step.
```

```
#include<stdio.h>
#include<conio.h>
void merge(int [],int ,int ,int );
void part(int [],int ,int );
int main()
{
 int arr[30];
 int i,size;
 printf("\n\t------- Merge sorting method -------\n\n");
 printf("Enter total no. of elements : ");
 scanf("%d",&size);
 for(i=0; i<size; i++)
 {
   printf("Enter %d element : ",i+1);
   scanf("%d",&arr[i]);
 }
 part(arr,0,size-1);
 printf("\n\t------- Merge sorted elements -------\n\n");
 for(i=0; i<size; i++)
 printf("%d ",arr[i]);
 getch();
 return 0;
}

void part(int arr[],int min,int max)
{
 int mid;
 if(min<max)
 {
   mid=(min+max)/2;
   part(arr,min,mid);
   part(arr,mid+1,max);
   merge(arr,min,mid,max);
 }
}

void merge(int arr[],int min,int mid,int max)
{
  int tmp[30];
  int i,j,k,m;
  j=min;
  m=mid+1;
  for(i=min; j<=mid && m<=max ; i++)
  {
     if(arr[j]<=arr[m])
     {
         tmp[i]=arr[j];
         j++;
     }
     else
     {
         m++;
     }
  }
  if(j>mid)
  {
     for(k=m; k<=max; k++)
     {
         tmp[i]=arr[k];
         i++;
     }
  }
  else
  {
     for(k=j; k<=mid; k++)
     {
         tmp[i]=arr[k];
         i++;
     }
  }
  for(k=min; k<=max; k++)
     arr[k]=tmp[k];
}
```

OUTPUT:

```
Enter the no of elements:7
7 8 9 4 5 3 1
The unsorted list is: 7 8 9 4 5 3 1
The sorted list is
1 3 4 5 7 8 9
```

EXPLANATION:

Merge sort is based on Divide and conquer method. It takes the list to be sorted and divide it in half to create two unsorted lists. The two unsorted lists are then sorted and merged to get a sorted list. The two unsorted lists are sorted by continually calling the merge-sort algorithm; we eventually get a list of size 1 which is already sorted. The two lists of size 1 are then merged.

Best case – When the array is already sorted O(nlogn) .

Worst case – When the array is sorted in reverse order O(nlogn).

Average case – O(nlogn).Extra space is required, so space complexity is O(n) for arrays and O(logn) for linked lists.

Exercise No.2:
>Implement Quick Sort using Divide and Conquer approach
> Find Maximum and Minimum element from an array of integer using Divide and Conquer approach

Aim:Implement Quick Sort using Divide and Conquer approach

Algorithm

```
quick( x, lb,ub)
{
if(lb<ub)
        {
                j=partition(x,lb,ub);
                quick(x,lb,j-1);
                quick(x,j+1,ub);
        }
}
```

Program:

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#define MAX 6000

void quick(int x[],intlb,intub);
int partition(int x[],intlb,intub);

void main()
{
        inti,n,x[MAX];
        time_tstart,end;
        clrscr();
        printf("Enter the number of elements: ");
        scanf("%d",&n);

        for(i=0;i<n;i++)
                x[i]=rand();

        printf("\nEntered array is \n");
        for(i=0;i<n;i++)
                printf("%d ",x[i]);

        start=time(NULL);
        quick(x,0,n-1);
        end=time(NULL);
        printf("Sorted array is as shown:\n");
        for(i=0;i<n;i++)
                printf("%d ",x[i]);
        printf("\nTIME for %d elements : %f", n, difftime(end,start));
        getch();

}

void quick(int x[],intlb,intub)
{
        int j;
        if(lb<ub)
        {
                j=partition(x,lb,ub);
                quick(x,lb,j-1);
                quick(x,j+1,ub);
        }
}

int partition(int x[],intlb,intub)
{
        inta,down,up,temp;
        a=x[lb];
        up=ub;
        down=lb;
        while(down<up)
        {
                while(x[down]<=a&&down<ub)
                        down++;
                while(x[up]>a)
                        up--;
                if(down<up)
                {
                        temp=x[down];
                        x[down]=x[up];
                        x[up]=temp;
                }
        }
        x[lb]=x[up];
        x[up]=a;
        return up;
}
```

OUTPUT:

```
Enter the number of elements:5
    Entered array is
       41   18467     6334     26500     19169
    Sorted array is as shown
       41   6334       18467     19169     26500
    Time for 5 elements : 0.000000
```

EXPLANATION:

Worst-case: $O(N^2)$

This happens when the pivot is the smallest (or the largest) element.
Then one of the partitions is empty, and we repeat recursively the procedure for N-1 elements.
Best-case: O(NlogN**)**

The best case is when the pivot is the median of the array, and then the left and the right part will have same size.There are logN partitions, and to obtain each partitions we do **N** comparisons
(and not more than N/2 swaps). Hence the complexity is O(NlogN)

Average-case : O(NlogN).

Aim:Find Maximum and Minimum element from a array of integer using Divide and Conquer approach

Algorithm:

```
{
if (i == j)
{
min = a[i];
max = a[j];
  }
else if (j == i + 1)
 {
if (a[i] > a[j])
   {
min = a[j];
max = a[i];
    }
else
   {
min = a[i];
max = a[j];
    }
  }
else
{
mid = (i + j) / 2;
minmax(a, i, mid, &lmin, &lmax);
minmax(a, mid + 1, j, &rmin, &rmax);
min = (lmin>rmin) ? rmin :lmin;
max = (lmax>rmax) ? lmax :rmax;
  }
}
```

Program:

```
#include<stdio.h>
voidminmax (int* a, int i, int j, int* min, int* max) {
intlmin, lmax, rmin, rmax, mid;
if (i == j)
{
    *min = a[i];
    *max = a[j];
  }
else if (j == i + 1)
 {
if (a[i] > a[j])
   {
      *min = a[j];
      *max = a[i];
    }
else
   {
      *min = a[i];
*max = a[j];
    }
  }
{
mid = (i + j) / 2;
minmax(a, i, mid, &lmin, &lmax);
minmax(a, mid + 1, j, &rmin, &rmax);
   *min = (lmin>rmin) ?rmin :lmin;
   *max = (lmax>rmax) ?lmax :rmax;
  }
}
void main ()
{
int a [] = {3, 4, 2, 6, 8, 1, 9, 12, 15, 11};
int min, max;
minmax (a, 0, 9, &min, &max);
printf ("Min : %d, Max: %d\n", min, max);
getch();
}
```

OUTPUT

```
Min : 1,Max : 15
```

EXPLANATION:

To find the maximum and minimum element using divide and conquer method, we divide the array into two sub array. Every time we search the maximum and minimum element within both sub array. Atlast we combine the array and find out the maximum and minimum element.

Exercise No.3:
>Find the minimum number of scalar multiplication needed for chain of matrix

Description:

Dynamic Programming:

A dynamic programming algorithm will examine the previously solved sub problems and will combine their solutions to give the best solution for the given problem.

Aim: Write a program to find optimal solution for matrix multiplication

Algorithm:

```
for (L=2; L<n; L++)
   {
      for (i=1; i<n-L+1; i++)
      {
         j = i+L-1;
         m[i][j] = INT_MAX;
         for (k=i; k<=j-1; k++)
         {
            // q = cost/scalar multiplications
            q = m[i][k] + m[k+1][j] + p[i-1]*p[k]*p[j];
            if (q < m[i][j])
                m[i][j] = q;
         }
      }
   }
```

Program:

```
#include<stdio>
#include<conio>
#define INFY 999999999
longint m[20][20];
int s[20][20];
int p[20],i,j,n;

voidprint_optimal(inti,int j)
{
              if (i == j)
                               printf(" A%d ",i);
                else
                {
                               printf(" ( ");
                               print_optimal(i, s[i][j]);
                               print_optimal(s[i][j] + 1, j);
                               printf(" ) ");
                }
}

intMatrixChainOrder(int p[], int n)
{

    /* For simplicity of the program, one extra row and one
       extra column are allocated in m[][].  0th row and 0th
       column of m[][] are not used */
    int m[n][n];

    int i, j, k, L, q;

    /* m[i,j] = Minimum number of scalar multiplications needed
       to compute the matrix A[i]A[i+1]...A[j] = A[i..j] where
       dimension of A[i] is p[i-1] x p[i] */

    // cost is zero when multiplying one matrix.
    for (i=1; i<n; i++)
        m[i][i] = 0;

    // L is chain length.
    for (L=2; L<n; L++)
    {
        for (i=1; i<n-L+1; i++)
        {
            j = i+L-1;
            m[i][j] = INT_MAX;
            for (k=i; k<=j-1; k++)
            {
                // q = cost/scalar multiplications
                q = m[i][k] + m[k+1][j] + p[i-1]*p[k]*p[j];
                if (q < m[i][j])
                    m[i][j] = q;
            }
        }
    }

    return m[1][n-1];
}
}
int main()
{
    intarr[] = {1, 2, 3, 4};
```

```
   printf("Minimum number of multiplications is %d ",
                    MatrixChainOrder(arr, size));

   getchar();
   return 0;
}
```

Output:
Minimum number of multiplications is 18

Exercise No.4:
>Implement all pair of Shortest path for a graph (Floyed- WarshallAlgorithm)
>Implement Single Source shortest Path for a graph (Bellman Ford Algorithm or dijkstraalgorithm)

Aim:Implement Single Source shortest Path for a graph (Bellman Ford Algorithm using dynamic method) or Implement all pair of Shortest path for a graph (Floyd- Warshall Algorithm)

ALL PAIR SHORTEST PATH:
The all pairs shortest path can be formulated informally as: If you have a weighted graph find for all pairs of vertices in the graph, the shortest path between these vertices. Formally, let G = (V, E) be a graph of vertices (V) and edges (E).

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
int max(int,int);
voidwarshal(int p[10][10],int n)
{
inti,j,k;
for(k=1;k<=n;k++)
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
p[i][j]=max(p[i][j],p[i][k]&&p[k][j]);
}
int max(inta,int b)
{                                                    ;
if(a>b)
return(a);
else
return(b);
}
int main()
{
int p[10][10]={0},n,e,u,v,i,j;
printf("\n Enter the number of vertices:");
scanf("%d",&n);
printf("\n Enter the number of edges:");
scanf("%d",&e);
for(i=1;i<=e;i++)
 {
printf("\n Enter the end vertices of edge %d:",i);
scanf("%d%d",&u,&v);
p[u][v]=1;
 }
printf("\n Matrix of input data: \n");
for(i=1;i<=n;i++)
 {
for(j=1;j<=n;j++)
printf("\n");
 }
warshal(p,n);
printf("\n Transitive closure: \n");
for(i=1;i<=n;i++)
 {
for(j=1;j<=n;j++)
printf("%d\t",p[i][j]);
printf("\n");
 }
getch();
return 0;
}
```

EXPLANATION:

Using all pair shortest path we find the minimum distance from every node to another node.The time complexity of all pair shortest path is o(n^3).

Exercise No.5:

>Implement 15 Puzzle Problem

Description:

Branch and Bound method:

Branch and Bound is a general method for finding optimal solutions to problems, typically discrete problems. A branch-and-bound algorithm searches the entire space of candidate solutions, with one extra trick: it throws out large parts of the search space by using previous estimates on the quantity being optimized.

Aim:Implement 15 Puzzle Problem using Branch and Bound method

15 Puzzle Problem:

PROGRAM

```
#include<stdio.h>
#include<conio.h>
int m=0,n=4;
intcal(int temp[10][10],int t[10][10])
{
        inti,j,m=0;
        for(i=0;i<n;i++)
                for(j=0;j<n;j++)
                {
                        if(temp[i][j]!=t[i][j])
                        m++;
                }
        return m;
}
int check(int a[10][10],int t[10][10])
{
        for(i=0;i<n;i++)
                for(j=0;j<n;j++)
                        if(a[i][j]!=t[i][j])
        f=0;
        return f;
}
void main()
{
        intp,i,j,n=4,a[10][10],t[10][10],temp[10][10],r[10][10];
        int m=0,x=0,y=0,d=1000,dmin=0,l=0;
        clrscr();
        printf("\nEnter the matrix to be solved,space with zero :\n");
        for(i=0;i<n;i++)
                for(j=0;j<n;j++)
                        scanf("%d",&a[i][j]);

        printf("\nEnter the target matrix,space with zero :\n");
        for(i=0;i<n;i++)
                for(j=0;j<n;j++)
                        scanf("%d",&t[i][j]);
        printf("\nEntered Matrix is :\n");
        for(i=0;i<n;i++)
        {
                for(j=0;j<n;j++)
                        printf("%d\t",a[i][j]);
                printf("\n");
        }
        printf("\nTarget Matrix is :\n");
        for(i=0;i<n;i++)
        {
                for(j=0;j<n;j++)
                        printf("%d\t",t[i][j]);
                printf("\n");
        }
        while(!(check(a,t)))
        {
                l++;
                d=1000;
                for(i=0;i<n;i++)
                        for(j=0;j<n;j++)
                        {
                                if(a[i][j]==0)
                                {
                                        x=i;
                                        y=j;
                                }
                        }

                //To move upwards
                                temp[i][j]=a[i][j];

                if(x!=0)
                {
                        p=temp[x][y];
                        temp[x][y]=temp[x-1][y];
                        temp[x-1][y]=p;
                }
                m=cal(temp,t);
                dmin=l+m;
                if(dmin<d)
                {
                        d=dmin;
                        for(i=0;i<n;i++)
                        for(j=0;j<n;j++)
                        r[i][j]=temp[i][j];
                }

                //To move downwards
                for(i=0;i<n;i++)
                        for(j=0;j<n;j++)
                                temp[i][j]=a[i][j];
                if(x!=n-1)
                {
                        p=temp[x][y];
                        temp[x][y]=temp[x+1][y];
                        temp[x+1][y]=p;
                }
                m=cal(temp,t);
                dmin=l+m;
                if(dmin<d)
                {
                        d=dmin;
                        for(i=0;i<n;i++)
                        for(j=0;j<n;j++)
                        r[i][j]=temp[i][j];
                }

                //To move right side
                for(i=0;i<n;i++)
                        for(j=0;j<n;j++)
                                temp[i][j]=a[i][j];
                if(y!=n-1)
                {
                        p=temp[x][y];
                        temp[x][y]=temp[x][y+1];
                        temp[x][y+1]=p;
                }
                m=cal(temp,t);
                dmin=l+m;
                if(dmin<d)
```

```
                        d=dmin;
                        for(i=0;i<n;i++)
                        for(j=0;j<n;j++)
                        r[i][j]=temp[i][j];
                }

                //To move left
                for(i=0;i<n;i++)
                        for(j=0;j<n;j++)
                                temp[i][j]=a[i][j];
                if(y!=0)
                {
                        p=temp[x][y];
                        temp[x][y]=temp[x][y-1];
                        temp[x][y-1]=p;
                }
                m=cal(temp,t);
                dmin=l+m;
                if(dmin<d)
                {
                        d=dmin;
                        for(i=0;i<n;i++)
                        for(j=0;j<n;j++)
                        r[i][j]=temp[i][j];
                }
                printf("\nCalculated Intermediate Matrix Value :\n");
                for(i=0;i<n;i++)
                {
                        for(j=0;j<n;j++)
                        printf("%d\t",r[i][j]);
                        printf("\n");
                }
                for(i=0;i<n;i++)
                        for(j=0;j<n;j++)
                        {
                        a[i][j]=r[i][j];
                        temp[i][j]=0;
                        }
                printf("Minimum cost : %d\n",d);
        }
        getch();
}
```

OUPUT:

```
Enter the matrix to be solved,space with zero:
1
2
3
4
5
```

```
8
9
10
7
11
13
14
15
12
Enter the target matrix,space with zero :
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
0

Entered Matrix is :
1       2       3       4
5       6       0       8
9       10      7       11
13      14      15      12
Target Matrix is :
1       2       3       4
5       6       7       8
9       10      11      12
13      14      15      0
Calculated Intermediate Matrix Value :
1       2       3       4
5       6       7       8
9       10      0       11
13      14      15      12
Minimum cost : 4
Calculated Intermediate Matrix Value :
1       2       3       4
5       6       7       8
9       10      11      0
13      14      15      12
Minimum cost : 4
Calculated Intermediate Matrix Value :
1       2       3       4
```

```
9       10      11      12
13      14      15      0
Minimum cost : 3
```

EXPLANATION:

Found

Exercise No.6:
>Implement 8 Queen problem
>Graph Coloring Problem

Description:

BACKTRACKING:

Backtracking is a technique used to solve problems with a large search space, by systematically trying and eliminating possibilities.

A standard example of backtracking would be going through a maze.

Implement N queens problem using back tracking method

Nqueens problem

Program:

```
#include<conio.h>
#include<stdio.h>
#include<math.h>
int board[10];
int n;

voidPrintBoard()
{
int i, k;
staticintsolno=1;

printf("\nSolution %d:\n", solno++);
for (i = 0; i < n; i++)
    {
for (k = 0; k < board[i]; k++)
printf("  ");
printf("%2d\n", board[i]+1);
    }
}
intIsPlaceSafe(int row, int col)
{
int i;
for(i=0; i<row; i++)
    {
if( (board[i] == col) || (abs(board[i]-col) == abs(i-row)) )
return 0;
    }
return 1;
}
voidNQueens(int row)
```

```
int col;
for(col=0; col<n; col++)
    {
if(IsPlaceSafe(row, col))
        {
board[row] = col;
if(row == n-1)
PrintBoard();
else
NQueens(row+1);
        }
    }
}
int main()
{
int i;
clrscr();
printf("Enter board size: ");
scanf("%d", &n);
if(n>10) exit(1);
NQueens(0);
getch();
return 0;
}
```

Output:

2 4 1 3

EXPANATION:

Complexity of backtracking algorithm for 8 queens problem will be O(n!).

Aim:ImplementGraph coloring problem using back tracking method

Program:

```
#include<conio.h>
#include<stdio.h>
#include<math.h>
int board[10];
int n;
intadj[3][3]={0,1,1
                        1,0,1
                        1,1,0}

voidPrintBoard()
{
int i, k;
staticintsolno=1;

printf("\nSolution %d:\n", solno++);
for (i = 0; i < n; i++)
    {
for (k = 0; k < board[i]; k++)
printf("  ");
printf("%(3d)\n", board[i]+1);
}
intIsPlaceSafe(int row, int col)
{
int i;
for(i=0; i<row; i++)
    {
if( (board[i] == col) || adj[i]==1)
return 0;
    }
return 1;
}
voidcolorgraph(int row)
{
int col;
for(col=0; col<n; col++)
    {
if(IsPlaceSafe(row, col))
        {
board[row] = col;
if(row == n-1)
PrintBoard();
else
colorgraph(row+1);
        }
    }
}
int main()
{
int i;
clrscr();
printf("Enter board size: ");
scanf("%d", &n);
if(n>10) exit(1);
NQueens(0);
getch();
return 0;
}
```

Output:

```
Enter board size 3
1 2 3
```

Exercise No.7:

>Knapsack Problem

>Job sequencing with deadlines

Description:

GREEDY METHOD:

A greedy algorithm is an algorithm that follows the problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum.

Aim:Knapsack Problem,Job sequencing with deadlines using greedy method

Knapsack Problem

PROGRAM

```
#include <conio.h>
void knapsack(int n, float weight[], float profit[], float capacity)
     {
      float x[20], tp = 0;
      int i, j, u;
      u = capacity;

         for (i = 0; i < n; i++)
         x[i] = 0.0;

         for (i = 0; i < n; i++)
{
          if (weight[i] > u)
           break;
            else {
         x[i] = 1.0;
         tp = tp + profit[i];
         u = u - weight[i];
      }
   }

   if (i < n)
      x[i] = u / weight[i];

   tp = tp + (x[i] * profit[i]);

   printf("\nThe result vector is:- ");
   for (i = 0; i < n; i++)
      printf("%f\t", x[i]);

   printf("\nMaximum profit is:- %f", tp);

}

int main()
{
   float weight[20], profit[20], capacity;
   intnum, i, j;
   float ratio[20], temp;

   printf("\nEnter the no. of objects:- ");
   scanf("%d", &num);

   printf("\nEnter the wts and profits of each object:- ");
   for (i = 0; i <num; i++) {
      scanf("%f %f", &weight[i], &profit[i]);
 }

   printf("\nEnter the capacityacity of knapsack:- ");
   scanf("%f", &capacity);

   for (i = 0; i <num; i++)
{
      ratio[i] = profit[i] / weight[i];
 }

   for (i = 0; i <num; i++)
 {
      for (j = i + 1; j <num; j++)
      {
         if (ratio[i] < ratio[j])
         {
            temp = ratio[i];
          ratio[i] = temp;

          temp = weight[j];
          weight[j] = weight[i];
          weight[i] = temp;

          temp = profit[j];
          profit[j] = profit[i];
          profit[i] = temp;
        }
     }
  }

  knapsack(num, weight, profit, capacity);
getch( );
  return(0);
}
```

OUTPUT :

```
   Enter the  no of objects :- 7
   Enter  the weights and profits of each object:
      2   10
      3    5
      5   15
      7    7
        1       6
        4    18
       1       3
Enter the capacity of knapsack: 15
  The result vector is:- 1.000000        1.000000        1.000000        1.000000

                                                   1.000000        0.666667        0.000000


   Maximum profit is:- 55.333332
```

EXPLANATION :

If the items are already sorted into decreasing order of $v_i / w_{i,}$ then
the while-loop takes a time in *O(n)*;
Therefore, the total time including the sort is in *O(n log n)*.

If we keep the items in heap with largest v_i/w_i at the root. Then

creating the heap takes *O(n)* time

while-loop now takes *O(log n)* time .

Implement Single Source shortest Path for a graph (Dijkstra using greedy method)

Single Source shortest Path:

Dijkstra Algorithm

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
int cost[8][8],dist[8][8],n;
void main()
{
        inti,j,k,m,u,INF=100;
        printf("\nEnter the no. of vertices: ");
        scanf("%d",&n);
        printf("\nEnter the adjacency matrix(Enter 100 if there is no edge present)\n");
        for(i=1;i <= n;i++)
                for(j=1;j <= n;j++)
                {
                        scanf("%d",&cost[i][j]);
                }
        for(i=1;i <= n;i++)
                dist[i][1]=0;
        printf("\nDistance Matrix:\n");
        for(i=2;i <= n;i++)
        {
                dist[1][i]=cost[1][i];
                printf("%d    ",dist[1][i]);
        }
        printf("\n");
        printf("\n");
        for(k=2;k <n;k++)
        {
                for(u=2;u <= n;u++)
                {
                        m=min(k,u);
                        if(dist[k-1][u] > m)
                                dist[k][u]=m;
                        else
                                dist[k][u]=dist[k-1][u];
                        printf("%d    ",dist[k][u]);
                }
                printf("\n");
                printf("\n");
        }
        for(i=1;i <= n;i++)
        {
                printf("\nDistance of 1 to %d is ",i);
                printf("%d    ",dist[n-1][i]);
        }
        getch();
}
int min(intk,int u)
{
        int min1,i;
        min1=dist[k-1][1]+cost[1][u];
```

```
                        if(min1 > (dist[k-1][i]+cost[i][u]))
                                    min1=dist[k-1][i]+cost[i][u];
            return min1;
}
```

OUTPUT:

```
Enter the no. of vertices: 7
Enter the adjacency matrix(Enter 100 if there is no edge present)
0       6       5       5       100     100     100
100     0       100     100     -1      100     100
100     -2      0       100     1       100     100
100     100     -2      0       100     -1      100
100     100     100     100     0       100     3
100     100     100     100     100     0       3
100     100     100     100     100     100     0

Distance Matrix:
6    5    5    100    100    100
3    3    5    5      4      100
1    3    5    2      4      7
1    3    5    0      4      5
1    3    5    0      4      3
1    3    5    0      4      3
Distance of 1 to 1 is 0
Distance of 1 to 2 is 1
Distance of 1 to 3 is 3
Distance of 1 to 4 is 5
Distance of 1 to 5 is 0
Distance of 1 to 6 is 4
Distance of 1 to 7 is 3
```

EXPLANATION:

Dijkstraalgorithm is used for finding the single-source shortest paths in a graph.It also can work with negative edge.

Complexity is O(|v|.|e|).

Exercise No.8: (implement any one of the following problem):
>Minimum Cost Spanning Tree by Prim's Algorithm
>Minimum Cost Spanning Tree by Kruskal's Algorithm

Minimum Cost Spanning Tree:
A minimum spanning tree is a spanning tree of a connected, undirected graph. It connects all the vertices together with the minimal total weighting for its edges. A single graph can have many different spanning trees.

Aim:Minimum Cost Spanning Tree by Prim's Algorithm,Minimum Cost Spanning Tree by Kruskal's Algorithm using greedy method(any one)

Prims algorithm:
PROGRAM

```
#include<stdio.h>
#include<conio.h>

int n, cost[10][10];
void prim()
{
        inti,j,k,l,x,nr[10],temp,min_cost=0,tree[10][3];
        /* For first smallest edge */
        temp=cost[0][0];
        for(i=0;i <n;i++)
        {
                for(j=0;j <n;j++)
                {
                        if(temp > cost[i][j])
                        {
                                temp=cost[i][j];
                                k=i;
                                l=j;
                        }
                }
        }
        /* Now we have fist smallest edge in graph */
        tree[0][0]=k;
        tree[0][1]=l;
        tree[0][2]=temp;
        min_cost=temp;
        /*      Now we have to find min dis of each
                vertex from either k or l
                by initialising nr[] array
        */
        for(i=0;i<n;i++)
        {
                if(cost[i][k]< cost[i][l])
                        nr[i]=k;
                else
                        nr[i]=l;
        }
        /* To indicate visited vertex initialise nr[] for them to 100 */
        nr[k]=100;
        nr[l]=100;
        /* Now find out remaining n-2 edges */
        {
                for(j=0;j<n;j++)
                {
                if(nr[j]!=100 && cost[j][nr[j]] < temp)
                  {
                        temp=cost[j][nr[j]];
                          x=j;
                  }
                }
                /* Now i have got next vertex */
                tree[i][0]=x;
                tree[i][1]=nr[x];
                tree[i][2]=cost[x][nr[x]];
                min_cost=min_cost+cost[x][nr[x]];
                nr[x]=100;
                /* Now find if x is nearest to any vertex
                than its previous near value */
                for(j=0;j<n;j++)
                {
                        if(nr[j]!=100 && cost[j][nr[j]] > cost[j][x])
                                nr[j]=x;
                }
                temp=99;
        }
        /* Now i have the answer, just going to print it */
        printf("\n The min spanning tree is:- \n");
        for(i=0;i< n-1;i++)
        {
                for(j=0;j < 3;j++)
                printf("%d\t", tree[i][j]);
                printf("\n");
        }
        printf("\n Min cost:- %d", min_cost);
}

void main()
{
        inti,j;
        clrscr();
        printf("\n Enter the no. of vertices:- ");
        scanf("%d", &n);
        printf ("\n Enter the costs of edges in matrix form:- ");
        for(i=0;i<n;i++)
                for(j=0;j<n;j++)
                        scanf("%d",&cost[i][j]);
        printf("\n The matrix is:- \n");
        for(i=0;i<n;i++)
        {
                for(j=0;j<n;j++)
                        printf("%d\t",cost[i][j]);
                printf("\n");
        }
        prim();
        getch();
}
```

Output:

```
Enter the no. of vertices:- 3
```

```
99 2 3
2 99 5
3 5 99

The matrix is:-
99      2       3
2       99      5
3       5       99

The min spanning tree is:-
0  1  2
2  0  3

Min cost:- 5
```

EXPLANATION:

The time complexity of prim's algorithm is O(VlogV + ElogV) = O(ElogV **).**

Kruskal Algorithm:
PROGRAM

```
#include<stdio.h>
#include<conio.h>
#define INF 1000
char vertex[10];
intwght[10][10];
intspan_wght[10][10];
int source;
struct Sort
{
                int v1,v2;
```

```
}que[20];

intn,ed,f,r;
int cycle(ints,int d)
{
        intj,k;
        if(source==d)
        return 1;
        for(j=0;j <n;j++)
        if(span_wght[d][j]!=INF && s!=j)
        {
                if(cycle(d,j))
                        return 1;
        }
        return 0;
}
voidbuild_tree()
{
        inti,j,w,k,count=0;
        for(count=0;count <n;f++)
        {
                i=que[f].v1;
                j=que[f].v2;
                w=que[f].weight;
                span_wght[i][j]=span_wght[j][i]=w;
                source=i;
                k=cycle(i,j);
                if(k)
                        span_wght[i][j]=span_wght[j][i]=INF;
                else
                        count++;
        }
}
void swap(int *i,int *j)
{
        int t;
        t=*i;
        *i=*j;
        *j=t;
}
int main()
{
        inti,j,k=0,temp;
        int sum=0;
        clrscr();
        printf("*** SPANNING TREE USING KRUSKAL'S ***\n");
        printf("Enter the No. of Nodes : ");
        scanf("%d",&n);
        for(i=0;i <n;i++)
        {
                printf("Enter %d value : ",i+1);
                scanf("%c",&vertex[i]);
                for(j=0;j <n;j++)
                {
                        wght[i][j]=INF;
                        span_wght[i][j]=INF;
                }
        }
printf("Getting Weight\n");
for(i=0;i <n;i++)
        for(j=i+1;j <n;j++)
        {
```

```
        scanf("%d",&ed);
        if(ed>= 1)
        {
            wght[i][j]=wght[j][i]=ed;
            que[r].v1=i;
            que[r].v2=j;
            que[r].weight=wght[i][j];
            if(r)
            {
                for(k=0;k <r;k++)
                if(que[k].weight >que[r].weight)
                {
                        swap(&que[k].weight,&que[r].weight);
                        swap(&que[k].v1,&que[r].v1);
                        swap(&que[k].v2,&que[r].v2);
                    }
                }
                r++;
            }
        }
    printf("\nORIGINAL GRAPH WEIGHT MATRIX");
    printf("\nweight matrix\n");
    for(i=0;i <n;i++,printf("\n"))
        for(j=0;j <n;j++,printf("\t"))
            printf("%d",wght[i][j]);
    build_tree();
    printf("\nMINIMUM SPANNING TREE");
    printf("\nLIST OF EDGES");
    for(i=0;i <n;i++)
        for(j=i+1;j <n;j++)
        if(span_wght[i][j]!=INF)
        {
            printf("\n%c ------ %c = %d ",vertex[i],vertex[j],span_wght[i][j]);
            sum+=span_wght[i][j];
        }
    printf("\nTotal Weight : %d ",sum);
    getch();
}
```

Output:

```
      KRUSKAL'S ALGORITHM TO FIND THE ALGORITHM
        Enter the no of nodes:5
Enter  1 value  1
Enter  2 value  2
Enter  3  value 3
Enter  4 value  4
Enter  5 value  5
Enter  0 if  path does not exist between 1 and 2: 5
Enter  0 if  path does not exist between 1 and 3: 0
Enter  0 if  path does not exist between 1 and 4: 2
Enter  0 if  path does not exist between 1 and 5: 4
Enter  0 if  path does not exist between 2 and 3: 1
Enter  0 if  path does not exist between 2 and 4: 0
Enter  0 if  path does not exist between 2 and 5: 9
Enter  0 if  path does not exist between 3 and 4: 0
Enter  0 if  path does not exist between 3 and 5: 1
Enter  0 if  path does not exist between 4 and 5: 8
         ORIGINAL GRAPH  WEIGHT MATRIX
          Weight matrix:
          1000     5    1000      2      4
           5       1000    1     1000   9
         1000       1     1000   1000  1
```

```
          4         9        1      8    1000
          MINIMUM SPANNING TREE
        LIST OG EDGES
            1-----4=2
            1-----5=4
            2-----3=1
            3-----5=1
Total weight : 8
```

EXPLANATION:

Time Complexity:
O(ElogE) or O(ElogV). Sorting of edges takes O(ELogE) time. After sorting, we iterate through all edges and apply find-union algorithm. The find and union operations can take atmostO(LogV) time. So overall complexity is O(ELogE+ ELogV) time. The value of E can be atmost V^2, so O(LogV) are O(LogE) same. Therefore, overall time complexity is O(ElogE) or O(ElogV).

Exercise No.9: (implement any one of the following problem):
>Implement Breadth First Search (BFS)
>Implement Depth First Search (DFS)

Graph traversal algorithm(BFS and DFS)
Some algorithms require that every vertex of a graph be visited exactly once. The order in which the vertices are visited may be important, and may depend upon the particular algorithm. The two common traversals: - depth-first - breadth-first.

Implement Breadth First Search (BFS):

```
 # include<stdio.h>

#include<conio.h>
int a[20][20],q[20],visited[20],n,i,j,f=0,r=-1;
voidbfs(int v)
{
 for(i=1;i<=n;i++)
  if(a[v][i] && !visited[i])
   q[++r]=i;
 if(f<=r)
 {
  visited[q[f]]=1;
  bfs(q[f++]);
 }
}
void main()
{
 int v;
 clrscr();
 printf("\n Enter the number of vertices:");
 scanf("%d",&n);
 for(i=1;i<=n;i++)
 {
  q[i]=0;
  visited[i]=0;
 }
 printf("\n Enter graph data in matrix form:\n");
 for(i=1;i<=n;i++)
  for(j=1;j<=n;j++)
   scanf("%d",&a[i][j]);
 scanf("%d",&v);
 bfs(v);
 printf("\n The node which are reachable are:\n");
 for(i=1;i<=n;i++)
  if(visited[i])
   printf("%d\t",i);
  else
   printf("\n Bfs is not possible");
 getch();
}
```

OUTPUT:

```
Enter the number of vertices: 4
Enter  graph  data  in  matrix form:
               2    4   5   7
               2    5   8   9
               1    4   7    5
               3    6    9    1

             Enter the starting  vertex  : 2
             The node which are  reachable are:
                  1     2      3    4
```

EXPLANATION:

Time complexity to go over each adjacent edges of a vertex is say O(N), where N is number of adjacent edges. So for V number of vertices time complexity becomes O(V*N) = O(E), where E is the total number of edges in the graph. Since removing and adding a vertex from/to Queue is O(1), why it is added to the overall time complexity of BFS as O(V+E).

Implement Depth First Search (DFS)

```
#include<stdio.h>

#include<conio.h>

int a[20][20],reach[20],n;
voiddfs(int v)
{
 int i;
 reach[v]=1;
 for(i=1;i<=n;i++)
  if(a[v][i] && !reach[i])
  {
   printf("\n %d->%d",v,i);
   dfs(i);
  }
}
void main()
{
 inti,j,count=0;
 clrscr();
 printf("\n Enter number of vertices:");
 {
  reach[i]=0;
  for(j=1;j<=n;j++)
   a[i][j]=0;
 }
 printf("\n Enter the adjacency matrix:\n");
 for(i=1;i<=n;i++)
  for(j=1;j<=n;j++)
   scanf("%d",&a[i][j]);
 dfs(1);
 printf("\n");
 for(i=1;i<=n;i++)
 {
  if(reach[i])
   count++;
 }
 if(count==n)
  printf("\n Graph is connected");
 else
  printf("\n Graph is not connected");
 getch();
}
```

OUTPUT:

```
              Enter number of vertices :4
              Enter the adjacency matrix:
                                  5      5  6  1
                                  2     3   4  1
                                  1     5   7   9
                                  4     2    8   4

                  1->2
                  2->3
                  3->4
Graph  is connected.
```

EXPLANATION:

The maximum number of possible edges in the graph G if it does not have cycle is |V| − 1. If G has a cycles, then the number of edges exceeds this number. Hence, the algorithm will detects a cycle at the most at the V^{th} edge if not before it. Therefore, the algorithm will run in O(V) time.

Exercise No.10:

>Implement Naïve algorithm for string matching.

The problem of finding occurrence(s) of a pattern string within another string or body of text. There are many different algorithms for efficient searching.
Example:Naive algorithm,

Naive algorithm:

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int match(char st[100],char pat[100]);
void main()
{
        charst[100],pat[100];
        int status;
        clrscr();
        printf("*** Naive String Matching Algorithm ***\n");
        printf("Enter the String.\n");
        gets(st);
        printf("Enter the pattern to match.\n");
        gets(pat);
        status=match(st,pat);
        if(status==-1)
        printf("\nNo match found");
        else
        printf("Match has been found on %d position.",status);
        getch();
}
int match(char st[100],char pat[100])
{
        intn,m,i,j,count=0,temp=0;
        n=strlen(st);
        m=strlen(pat);
        for(i=0;i<=n-m;i++)
        {
                temp++;
                for(j=0;j<m;j++)
                {
                        if(st[i+j]==pat[j])
                        count++;
                }
                if(count==m)
                return temp;
                count=0;
         }
         return -1;
}
```

OUTPUT:

```
Enter the pattern to match
JAI
match has been found in 5 position.
```

Explanation:
Required time is O(n).

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Microprocessors & Microcontrollers Lab
Course Code: CS592
L-T-P scheme: 0-0-3 **Course Credit: 2**

Objectives:

The course is intended to create an appreciation for contemporary concepts in high performance mutli core super scalar architectures and appreciate their implementation in modern multi processors.

Learning Outcomes:

Upon successful completion of the course, a student will have:

1. An ability to define and explain the principles of computer architecture and the interfacing between its Hardware and software components
2. An ability to write assembly programs and understand its machine code equivalent
3. An in-depth understanding of architectural blocks involved in computer arithmetic, both integer and Floating point.
4. An in-depth understanding of the data path inside a processor, its control and handling of exceptions
5. An in depth understanding of pipelining for 32-bit architectures
6. An ability to understand and analyze computer memory hierarchy, at all levels of its organization, and the interaction between caches and main memory
7. An ability to understand multi-processor architectures

Course Contents:

Exercises that must be done in this course are listed below:

Exercise No.1: Introduction to 8085 Microprocessor.
Exercise No.2: a) Addition of 2 - 8 bit numbers
b) Subtraction of 2 - 8 bit numbers
Exercise No.3: a) Addition of 2 - 16 bit numbers
b) Subtraction of 2 – 16 bit numbers
Exercise No.4: a) Multiplication of 2 - 8 numbers
b) Division of 2 - 8 bit numbers
Exercise No.5: a) Ascending order
b) Descending order
Exercise No.6: Factorial of Given Numbers
Exercise No.7: To write an assembly language program to displace Fibanocci Series.

Text Book:

Recommended Systems/Software Requirements:

1. 8085 kit.

.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR
Lab Manual

Experiment 1:-

Aim

To study the microprocessor 8085

Architecture of 8085 Microprocessor

a) General purpose register

It is an 8 bit register i.e. B,C,D,E,H,L. The combination of 8 bit register is known as register pair, which can hold 16 bit data. The HL pair is used to act as memory pointer is accessible to program.

b) Accumulator

It is an 8 bit register which hold one of the data to be processed by ALU and stored the result of the operation.

c) Program counter (PC)

It is a 16 bit pointer which maintain the address of a byte entered to line stack.

d) Stack pointer (Sp)

It is a 16 bit special purpose register which is used to hold line memory address for line next instruction to be executed.

e) Arithmetic and logical unit

It carries out arithmetic and logical operation by 8 bit address it uses the accumulator content as input the ALU result is stored back into accumulator.

f) Temporary register

It is an 8 bit register associated with ALU hold data, entering an operation, used by the microprocessor and not accessible to programs.

g) Flags

Flag register is a group of fire, individual flip flops line content of line flag register will change after execution of arithmetic and logic operation. The line states flags are

i) Carry flag (C)

ii) Parity flag (P)

iii) Zero flag (Z)

iv) Auxiliary carry flag (AC)

v) Sign flag (S)

h) Timing and control unit

Synchronous all microprocessor, operation with the clock and generator and control signal from it necessary to communicate between controller and peripherals.

i) Instruction register and decoder

Instruction is fetched from line memory and stored in line instruction register decoder the stored information.

j) Register Array

These are used to store 8 bit data during execution of some instruction.

PIN Description

Address Bus

1. The pins Ao – A15 denote the address bus.
2. They are used for most significant bit

Address / Data Bus

1. AD0 – AD7 constitutes the address / Data bus
2. These pins are used for least significant bit

ALE : (Address Latch Enable)

1. The signal goes high during the first clock cycle and enables the lower order address bits.

IO / M

1. This distinguishes whether the address is for memory or input.
2. When this pins go high, the address is for an I/O device.

S0 – S1
S0 and S1 are status signal which provides different status and functions.
RD
1. This is an active low signal
2. This signal is used to control READ operation of the microprocessor.

WR
1. WR is also an active low signal
2. Controls the write operation of the microprocessor.

HOLD
1. This indicates if any other device is requesting the use of address and data bus.

HLDA
1. HLDA is the acknowledgement signal for HOLD
2. It indicates whether the hold signal is received or not.

INTR
1. INTE is an interrupt request signal
2. IT can be enabled or disabled by using software

INTA
1. Whenever the microprocessor receives interrupt signal
2. It has to be acknowledged.

RST 5.5, 6.5, 7.5
1. These are nothing but the restart interrupts
2. They insert an internal restart junction automatically.

TRAP
1. Trap is the only non-maskable interrupt
2. It cannot be enabled (or) disabled using program.

RESET IN
1. This pin resets the program counter to 0 to 1 and results interrupt enable and HLDA flip flops.

X1, X2
These are the terminals which are connected to external oscillator to produce the necessary and suitable clock operation.
SID
This pin provides serial input data
SOD
This pin provides serial output data
VCC and VSS
1. VCC is +5V supply pin
2. VSS is ground pin

Specifications
1. Processors
Intel 8085 at E144 MHz clock
2. Memory
Monitor RAM: 0000 – IFFF
EPROM Expansion: 2000 – 3FFF’s
0000 – FFF
System RAM: 4000 – 5FFF

RAM Expansion 6000 – BFFF

3. Input / Output

Parallel: A8 TTL input timer with 2 number of 32-55 only input timer available in □-85 EBI.

Serial: Only one number RS 232-C, Compatible, crucial interface using 8281A

Timer: 3 channel -16 bit programmable units, using 8253 channel ‘0’ used for no band late. Clock generator. Channel ‘1’ is used for single stopping used program.

Display: 6 digit – 7 segment LED display with filter 4 digit for adder display and 2 digit for data display.

Key board: 21 keys, soft keyboard including common keys and hexa decimal keys.

RES: Reset keys allow to terminate any present activity and retain to □ - 85 its on initialize state.

INT: Maskable interrupt connect to CPU’s RST 7.5 interrupt

DEC: Decrement the adder by 1

EXEC: Execute line particular value after selecting address through go command.

NEXT: Increment the address by 1 and then display its content.

Key Functions:

i. Hex entry key ‘0’

ii. Substituting memory content where “next” key is paused immediately after 1, take used to st cutting address.

iii. Register key ‘E’

i) Hex code entry (1)

ii) Register key ‘D’

i) Hex code entry ‘2’

ii) Retricre data from data ‘memory’ to data top

iii) Register key ‘C’

i) Hex code entry ‘3’

ii) Retricre data from memory to top

iii) Register key ‘B’

i) Hex key entry ‘C’

ii) Block search from byte

iii) Register key ‘F’

i) Hex key entry ‘5’

ii) Fill block of RAM memory with desired data

iii) Register key ‘A’

i) Hex key entry ‘6’

ii) TN/Tl used for sending (or) receiving

iii) Register key ‘H’

i) Hex key entry ‘7’

ii) Register key ‘H’

i) Register key ‘S’

ii) Register key ‘I’

i) Hex key entry ‘A’

ii) Function key F3

iii) Register key “ph”

ii) Signal step program (instruction by instruction)

i) Hex key entry “c”
ii) Much a block of memory from a linear block
iii) Register key “SH”

i) Hex key D
ii) Compare 2 memory block

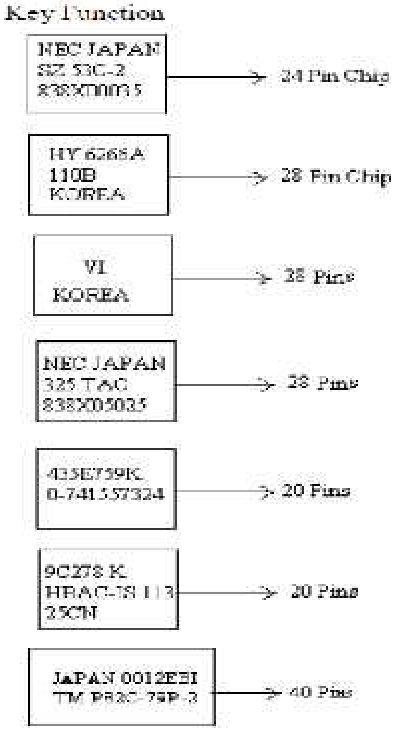
i) Hex key entry ‘B’
ii) Check a block from flame
iii) Register key “SPL”

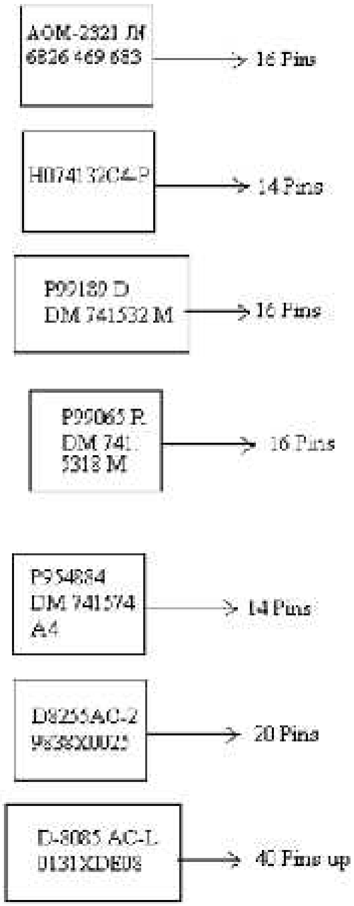
i) Hex key ‘E’
ii) Insert by test into memory (RAM)

i) Hex key ‘F’
ii) Delete byte from memory RAM

System Power Consumption
Micro BSEB2 MICRO SSEB
+5V @ 1Amp +5V@ 800 mA
+12V @ 200 mA
- 12V @ 100 mA

Power Supply Specification
MICRO SSEM
230V, AC @ 80 Hz
+5V @ 600 mA





IC's Used
8085 - 8 bit □p
8253 - programmable internal timer
8255 - programmable peripheral interface
8279 - programmable key boards / display interface
8251 - programmable communication interface
2764 - 8 KV VV EPROM
6264 - 8K STATIC PROM
7414 - Hex inverter
7432 - Quad 21/p OR GATE
7409 - Quad 21/p AND GATE
7400 - NAND Gate
7404 - Dual D-FF
74373 - Octal 'D' Latch
74139 - Dual 2 to 4 line decoder
74138 - 3 to 8 line decoder

In Enter Program into Trainer Kit
1. Press 'RESET' key
2. Sub (key processor represent address field)
3. Enter the address (16 bit) and digit in hex
4. Press 'NEXT' key
5. Enter the data
6. Again press "NEXT"

8. Press “NEXT”

How to executive program
1. Press “RESET”
2. Press “GO”
3. Enter the address location in which line program was executed
4. Press “Execute” key

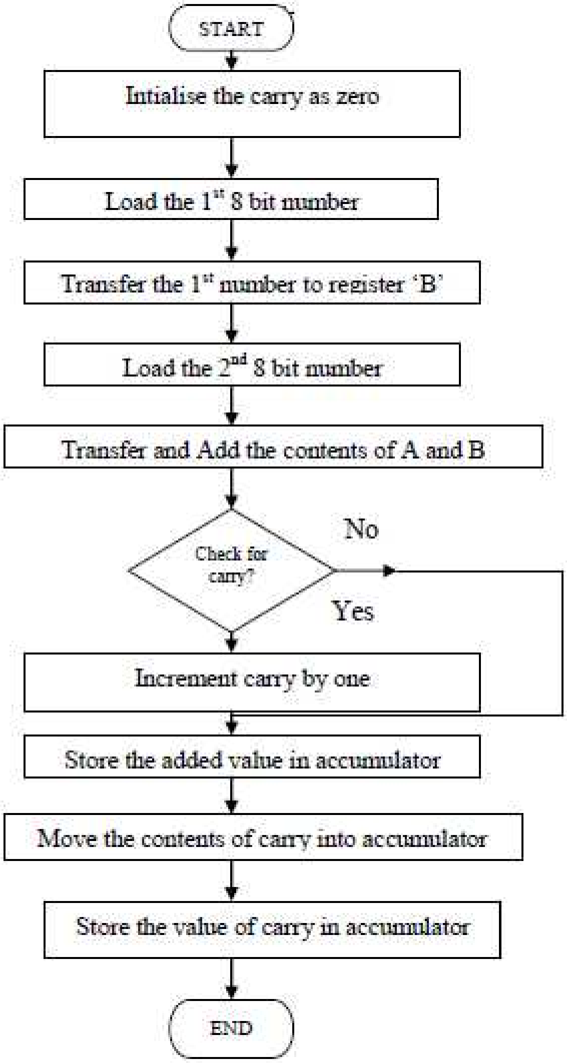
Result:
Thus 8085 microprocessor was studied successfully.

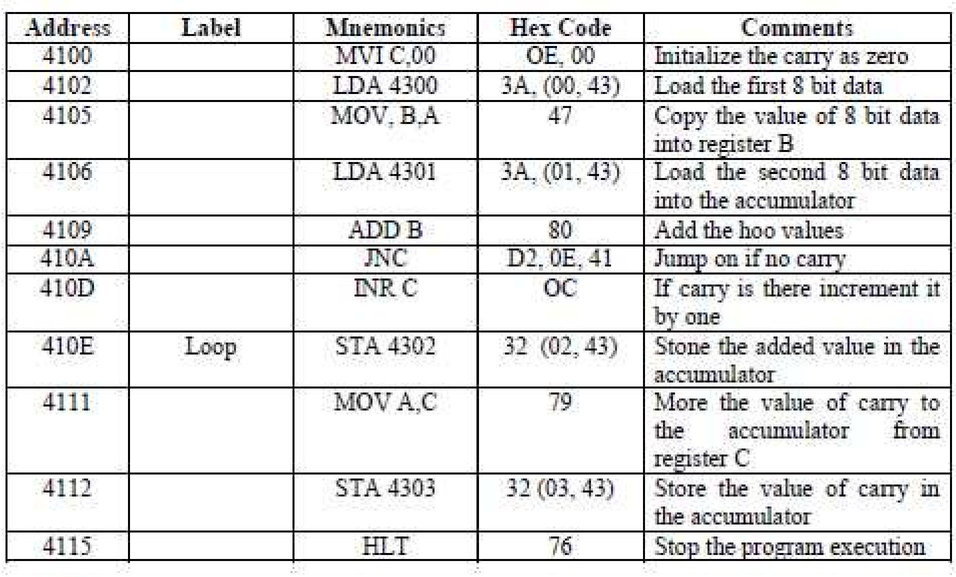
Experiment 2(a) :-

Aim:
To write an assembly language for adding two 8 bit numbers by using micro processor kit.

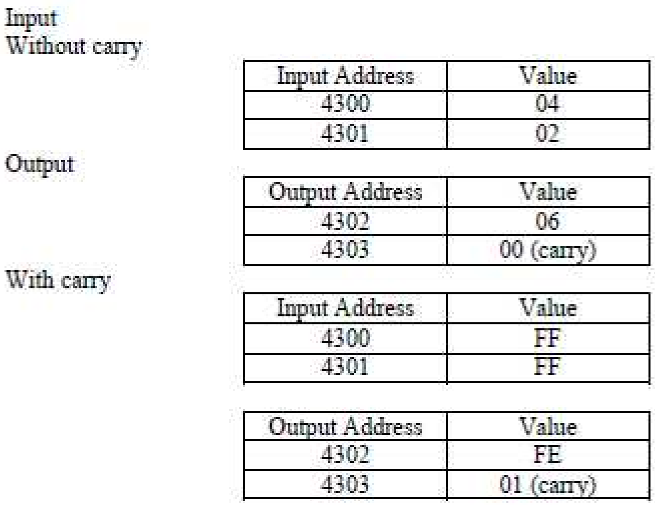
Apparatus required:
8085 micro processor kit
(0-5V) DC battery

Algorithm:
Step 1 : Start the microprocessor
Step 2 : Intialize the carry as 'Zero'
Step 3 : Load the first 8 bit data into the accumulator
Step 4 : Copy the contents of accumulator into the register 'B'
Step 5 : Load the second 8 bit data into the accumulator.
Step 6 : Add the 2 - 8 bit datas and check for carry.
Step 7 : Jump on if no carry
Step 8 : Increment carry if there is
Step 9 : Store the added request in accumulator
Step 10 : More the carry value to accumulator
Step 11 : Store the carry value in accumulator
Step 12 : Stop the program execution.


START
Intialise the carry as zero
Load the 1st 8 bit number
Transfer the 1st number to register 'B'
Load the 2nd 8 bit number
Transfer and Add the contents of A and B
Check for carry?
No
Yes
Increment carry by one
Store the added value in accumulator
Move the contents of carry into accumulator
Store the value of carry in accumulator
END



Address	Label	Mnemonics	Hex Code	Comments
4100		MVI C,00	OE, 00	Initialize the carry as zero
4102		LDA 4300	3A, (00, 43)	Load the first 8 bit data
4105		MOV, B,A	47	Copy the value of 8 bit data into register B
4106		LDA 4301	3A, (01, 43)	Load the second 8 bit data into the accumulator
4109		ADD B	80	Add the hoo values
410A		JNC	D2, 0E, 41	Jump on if no carry
410D		INR C	OC	If carry is there increment it by one
410E	Loop	STA 4302	32 (02, 43)	Stone the added value in the accumulator
4111		MOV A,C	79	More the value of carry to the accumulator from register C
4112		STA 4303	32 (03, 43)	Store the value of carry in the accumulator
4115		HLT	76	Stop the program execution



Input
Without carry

Input Address	Value
4300	04
4301	02

Output

Output Address	Value
4302	06
4303	00 (carry)

With carry

Input Address	Value
4300	FF
4301	FF

Output Address	Value
4302	FE
4303	01 (carry)

Calculation 1111 1111
1111 1111
----
(1) 1111 1110
=========
F E

Result:

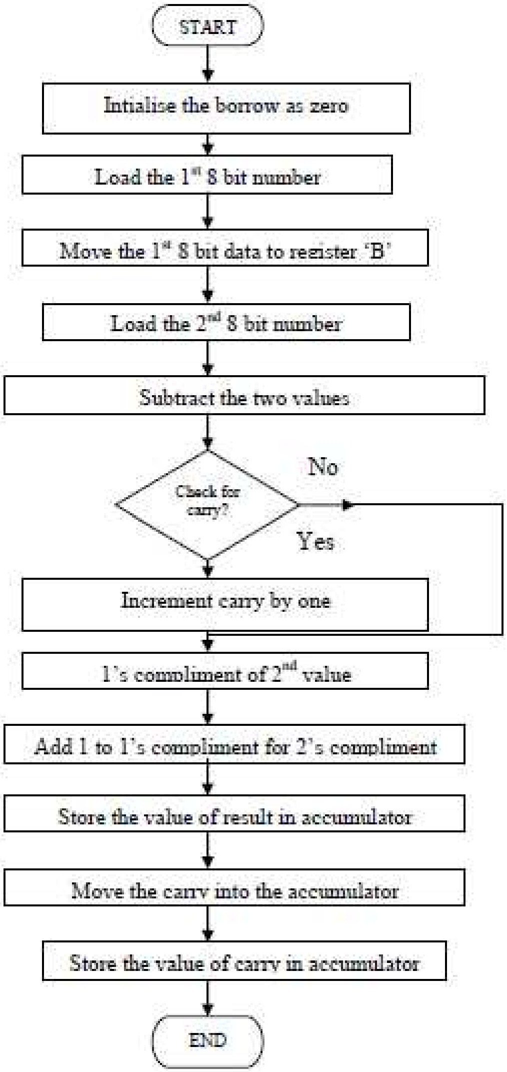
The assembly language program for 8 bit addition of two numbers was executed successfully by using 8085 micro processing kit.

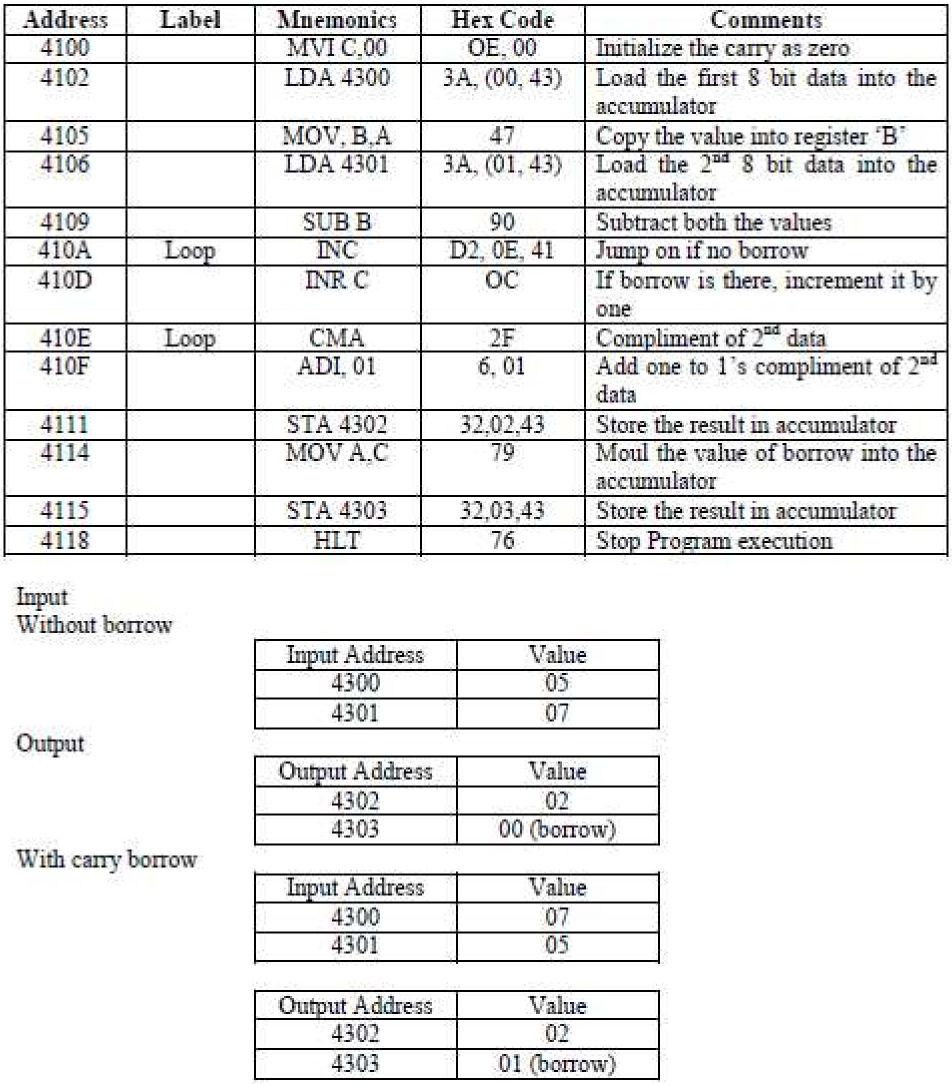
Experiment 2 (b): -

Aim:
To write a assembly language program for subtracting 2 bit (8) numbers by using- 8085 micro processor kit.

Apparatus required:
8085 micro processor kit
(0-5V) DC battery

Algorithm:
Step 1 : Start the microprocessor
Step 2 : Intialize the carry as ‘Zero’
Step 3 : Load the first 8 bit data into the accumulator
Step 4 : Copy the contents of contents into the register ‘B’
Step 5 : Load the second 8 bit data into the accumulator.
Step 6 : Subtract the 2 8 bit datas and check for borrow.
Step 7 : Jump on if no borrow
Step 8 : Increment borrow if there is
Step 9 : 2’s compliment of accumulator is found out
Step 10 : Store the result in the accumulator
Step 11 : More the borrow value from ‘c’ to accumulator
Step 12 : Store the borrow value in the accumulator
Step 13 : Stop program execution


START
Intialise the borrow as zero
Load the 1st 8 bit number
Move the 1st 8 bit data to register ‘B’
Load the 2nd 8 bit number
Subtract the two values
Check for carry?
No
Yes
Increment carry by one
1’s compliment of 2nd value
Add 1 to 1’s compliment for 2’s compliment
Store the value of result in accumulator
Move the carry into the accumulator
Store the value of carry in accumulator
END



Address	Label	Mnemonics	Hex Code	Comments
4100		MVI C,00	OE, 00	Initialize the carry as zero
4102		LDA 4300	3A, (00, 43)	Load the first 8 bit data into the accumulator
4105		MOV, B,A	47	Copy the value into register 'B'
4106		LDA 4301	3A, (01, 43)	Load the 2nd 8 bit data into the accumulator
4109		SUB B	90	Subtract both the values
410A	Loop	INC	D2, 0E, 41	Jump on if no borrow
410D		INR C	OC	If borrow is there, increment it by one
410E	Loop	CMA	2F	Compliment of 2nd data
410F		ADI, 01	6, 01	Add one to 1's compliment of 2nd data
4111		STA 4302	32,02,43	Store the result in accumulator
4114		MOV A,C	79	Moul the value of borrow into the accumulator
4115		STA 4303	32,03,43	Store the result in accumulator
4118		HLT	76	Stop Program execution

Input
Without borrow

Input Address	Value
4300	05
4301	07

Output

Output Address	Value
4302	02
4303	00 (borrow)

With carry borrow

Input Address	Value
4300	07
4301	05

Output Address	Value
4302	02
4303	01 (borrow)

Calculation 05 – 07
07 – 0111
CMA 1000
ADJ 0.1 0001

1001
05 - 0101

1110 (-2)

Result:
The assembly language program subtraction of two 8 bit numbers was executed successfully by using 8085 micro processing kit.

Experiment 3(a) :-

Aim:

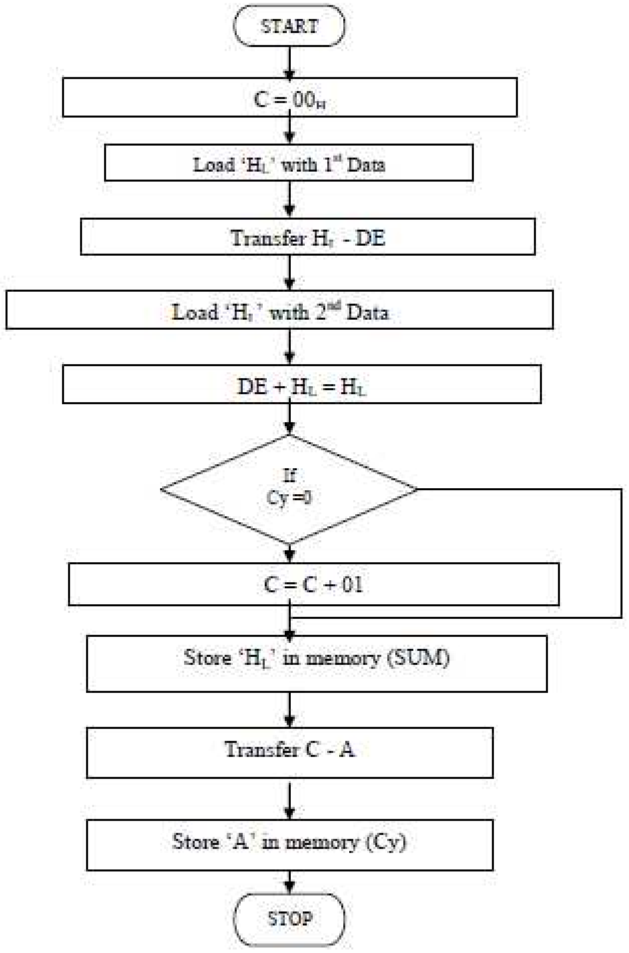
To write an assembly language program for adding two 16 bit numbers using 8085 micro processor kit.

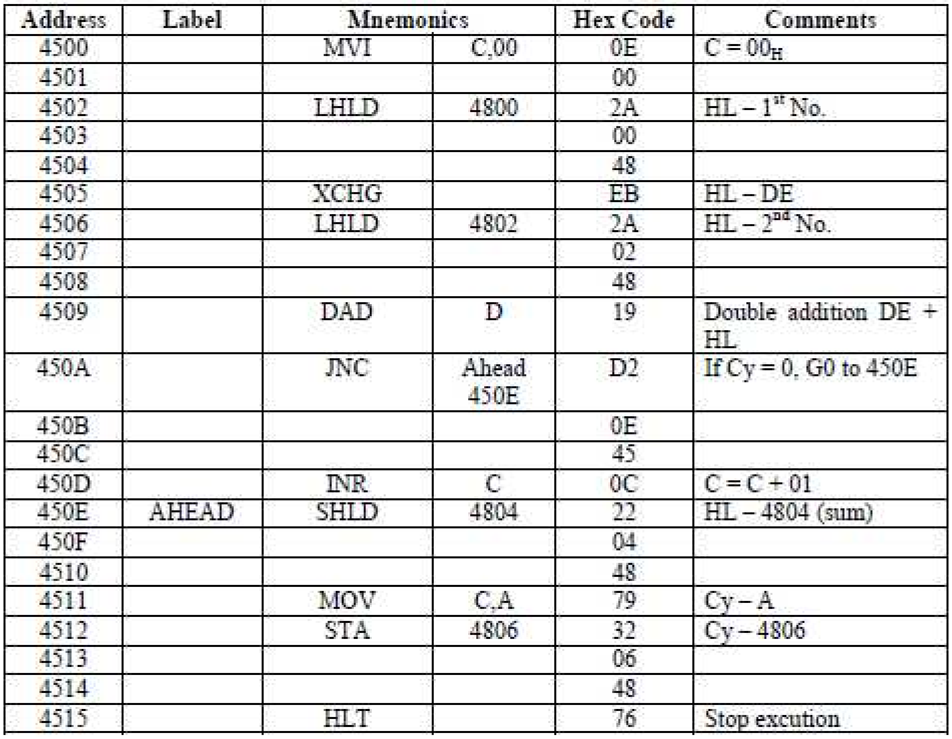
Apparatus required:

8085 micro processor kit
(0-5V) DC battery

Algorithm:

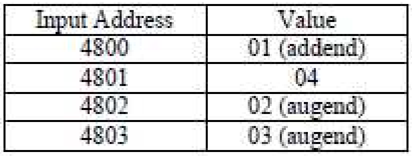
Step 1 : Start the microprocessor
Step 2 : Get the 1st 8 bit in ‘C’ register (LSB) and 2nd 8 bit in ‘H’ register (MSB) of 16 bit number.
Step 3 : Save the 1st 16 bit in ‘DE’ register pair
Step 4 : Similarly get the 2nd 16 bit number and store it in ‘HL’ register pair.
Step 5 : Get the lower byte of 1st number into ‘L’ register
Step 6 : Add it with lower byte of 2nd number
Step 7 : tore the result in ‘L’ register
Step 8 : Get the higher byte of 1st number into accumulator
Step 9 : Add it with higher byte of 2nd number and carry of the lower bit addition.
Step 10 : Store the result in ‘H’ register
Step 11 : Store 16 bit addition value in ‘HL’ register pair
Step 12 : Stop program execution





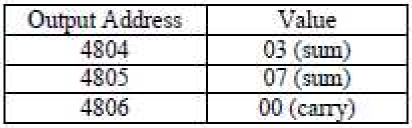
Address	Label	Mnemonics		Hex Code	Comments
4500		MVI	C,00	0E	C = 00_H
4501				00	
4502		LHLD	4800	2A	HL – 1st No.
4503				00	
4504				48	
4505		XCHG		EB	HL – DE
4506		LHLD	4802	2A	HL – 2nd No.
4507				02	
4508				48	
4509		DAD	D	19	Double addition DE + HL
450A		JNC	Ahead 450E	D2	If Cy = 0, G0 to 450E
450B				0E	
450C				45	
450D		INR	C	0C	C = C + 01
450E	AHEAD	SHLD	4804	22	HL – 4804 (sum)
450F				04	
4510				48	
4511		MOV	C,A	79	Cy – A
4512		STA	4806	32	Cy – 4806
4513				06	
4514				48	
4515		HLT		76	Stop excution

Input
Without



Input Address	Value
4800	01 (addend)
4801	04
4802	02 (augend)
4803	03 (augend)

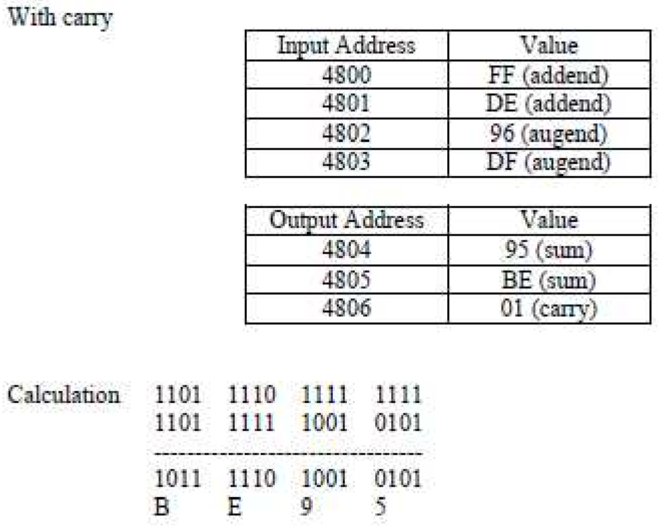
Output



Output Address	Value
4804	03 (sum)
4805	07 (sum)
4806	00 (carry)

Calculation 0000 0100 0000 0001
0000 0011 0000 0010

0000 0111 0000 0011
0 7 0 3



With carry

Input Address	Value
4800	FF (addend)
4801	DE (addend)
4802	96 (augend)
4803	DF (augend)

Output Address	Value
4804	95 (sum)
4805	BE (sum)
4806	01 (carry)

```
Calculation   1101   1110   1111   1111
              1101   1111   1001   0101
              ----------------------------------
              1011   1110   1001   0101
              B      E      9      5
```

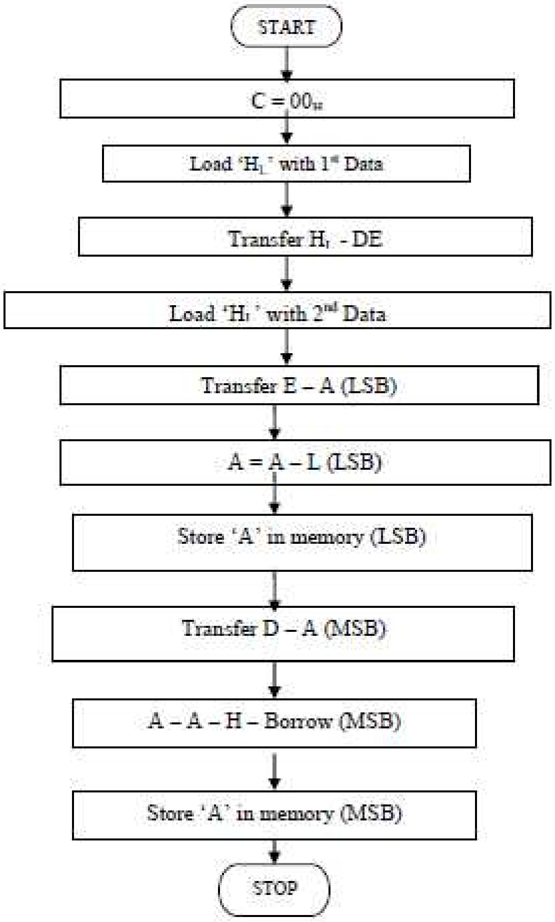
Result:
The assembly language program for addition of two 16 bit numbers was executed using 8085 micro processing kit.

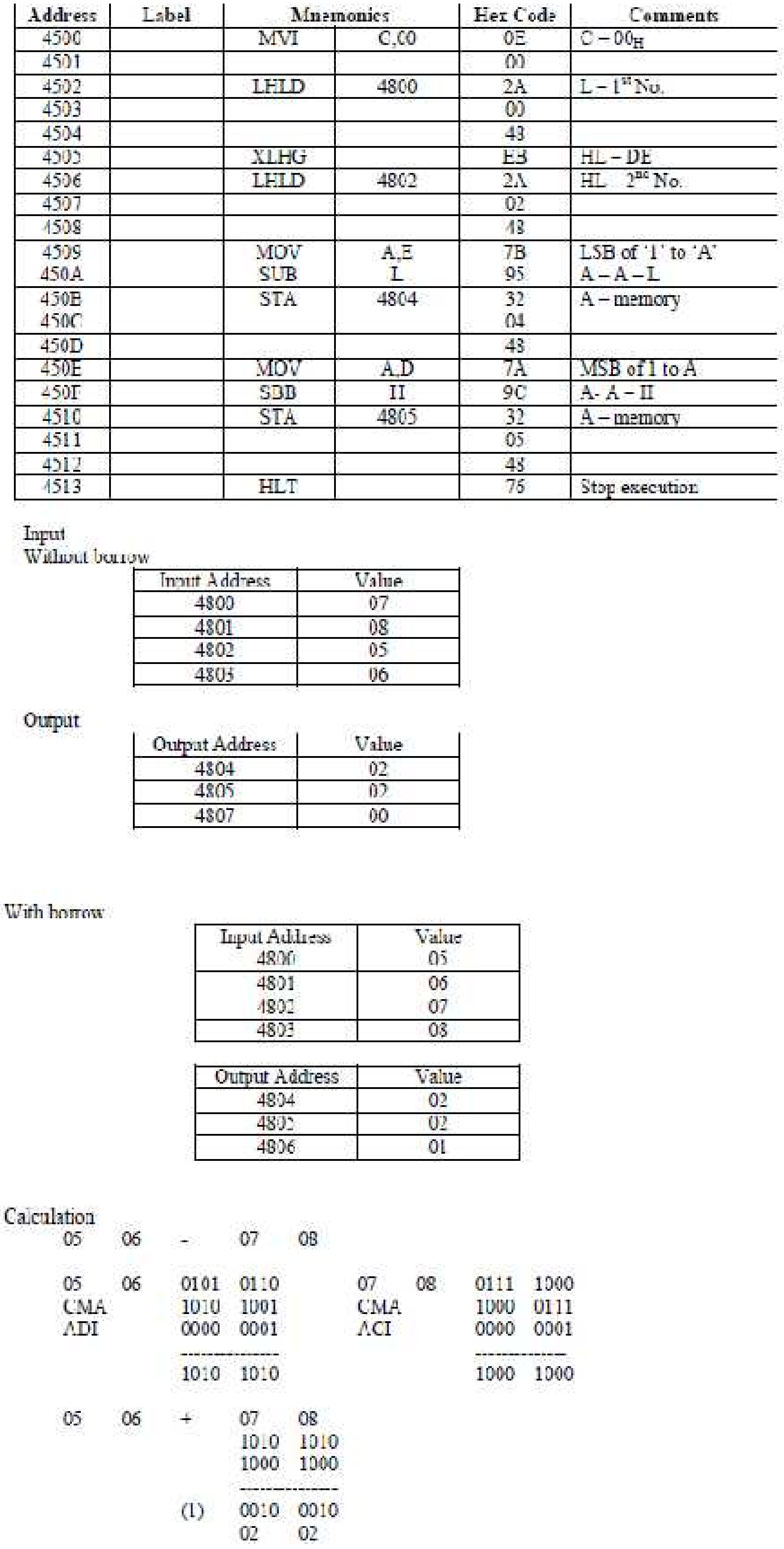
Experiment 3(b) :-

Aim:
To write an assembly language program for subtracting two 16 bit numbers using 8085 microprocessor kit.

Apparatus required:
8085 microprocessor kit
(0-5V) DC battery

Algorithm:
Step 1 : Start the microprocessor
Step 2 : Get the 1st 16 bit in ‘HL’ register pair
Step 3 : Save the 1st 16 bit in ‘DE’ register pair
Step 4 : Get the 2nd 16 bit number in ‘HL’ register pair
Step 5 : Get the lower byte of 1st number
Step 6 : Get the subtracted value of 2nd number of lower byte by subtracting it with lower byte of 1st number
Step 7 : Store the result in ‘L’ register
Step 8 : Get the higher byte of 2nd number
Step 9 : Subtract the higher byte of 1st number from 2nd number with borrow
Step 10 : Store the result in ‘HL’ register
Step 11 : Stop the program execution





Address	Label	Mnemonics		Hex Code	Comments
4500		MVI	C,00	0E	C – 00_H
4501				00	
4502		LHLD	4800	2A	L – 1st No.
4503				00	
4504				48	
4505		XLHG		EB	HL – DE
4506		LHLD	4802	2A	HL 2nd No.
4507				02	
4508				48	
4509 450A		MOV SUB	A,E L	7B 95	LSB of '1' to 'A' A – A – L
450B 450C		STA	4804	32 04	A – memory
450D				48	
450E		MOV	A,D	7A	MSB of 1 to A
450F		SBB	H	9C	A- A – H
4510		STA	4805	32	A – memory
4511				05	
4512				48	
4513		HLT		76	Stop execution

Input
Without borrow

Input Address	Value
4800	07
4801	08
4802	05
4803	06

Output

Output Address	Value
4804	02
4805	02
4807	00

With borrow

Input Address	Value
4800	05
4801	06
4802	07
4803	08

Output Address	Value
4804	02
4805	02
4806	01

Calculation

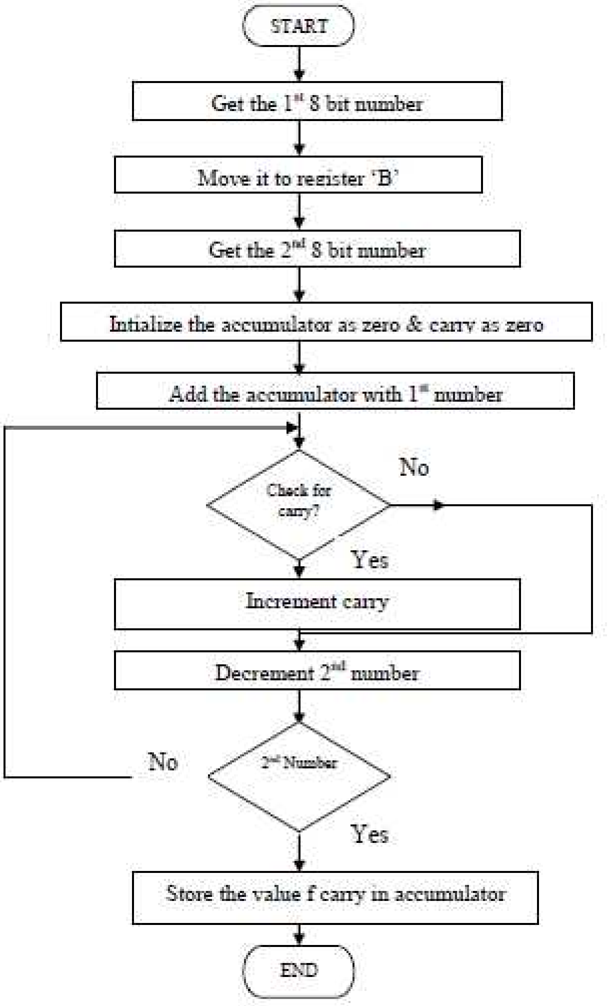
```
    05     06     -     07     08

    05     06     0101  0110        07     08     0111  1000
    CMA           1010  1001        CMA           1000  0111
    ADI           0000  0001        ACI           0000  0001
                  -----------                     -----------
                  1010  1010                      1000  1000

    05     06     +     07     08
                        1010   1010
                        1000   1000
                        -----------
                  (1)   0010   0010
                        02     02
```

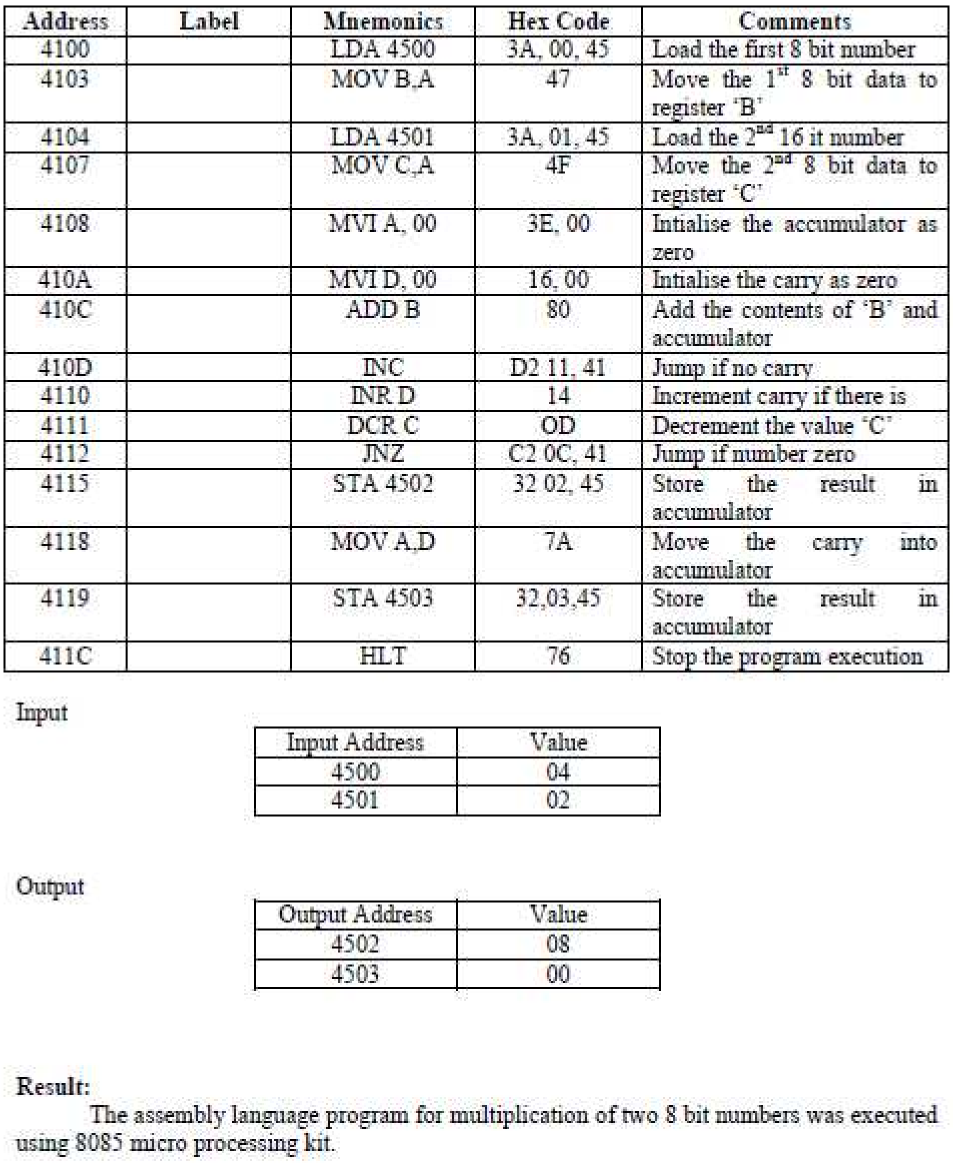
Result:

The assembly language program for subtraction of two 16 bit numbers was executed by using 8085 micro processing kit.

Experiment 4(a) :-

Aim:
To write an assembly language for multiplying two 8 bit numbers by using 8085 micro processor kit.
Apparatus required:
8085 microprocessor kit
(0-5V) DC battery
Algorithm:
Step 1 : Start the microprocessor
Step 2 : Get the 1st 8 bit numbers
Step 3 : Move the 1st 8it number to register 'B'
Step 4 : Get the 2nd 8 bit number
Step 5 : Move the 2nd 8 bit number to register 'C'
Step 6 : Intialise the accumulator as zero
Step 7 : Intialise the carry as zero
Step 8 : Add both register 'B' value as accumulator
Step 9 : Jump on if no carry
Step 10 : Increment carry by 1 if there is
Step 11 : Decrement the 2nd value and repeat from step 8, till the 2nd value becomes zero.
Step 12 : Store the multiplied value in accumulator
Step 13 : Move the carry value to accumulator
Step 14 : Store the carry value in accumulator





Address	Label	Mnemonics	Hex Code	Comments
4100		LDA 4500	3A, 00, 45	Load the first 8 bit number
4103		MOV B,A	47	Move the 1st 8 bit data to register 'B'
4104		LDA 4501	3A, 01, 45	Load the 2nd 16 it number
4107		MOV C,A	4F	Move the 2nd 8 bit data to register 'C'
4108		MVI A, 00	3E, 00	Intialise the accumulator as zero
410A		MVI D, 00	16, 00	Intialise the carry as zero
410C		ADD B	80	Add the contents of 'B' and accumulator
410D		INC	D2 11, 41	Jump if no carry
4110		INR D	14	Increment carry if there is
4111		DCR C	OD	Decrement the value 'C'
4112		JNZ	C2 0C, 41	Jump if number zero
4115		STA 4502	32 02, 45	Store the result in accumulator
4118		MOV A,D	7A	Move the carry into accumulator
4119		STA 4503	32,03,45	Store the result in accumulator
411C		HLT	76	Stop the program execution

Input

Input Address	Value
4500	04
4501	02

Output

Output Address	Value
4502	08
4503	00

Result:

The assembly language program for multiplication of two 8 bit numbers was executed using 8085 micro processing kit.

Experiment 4(b) :-

Aim:

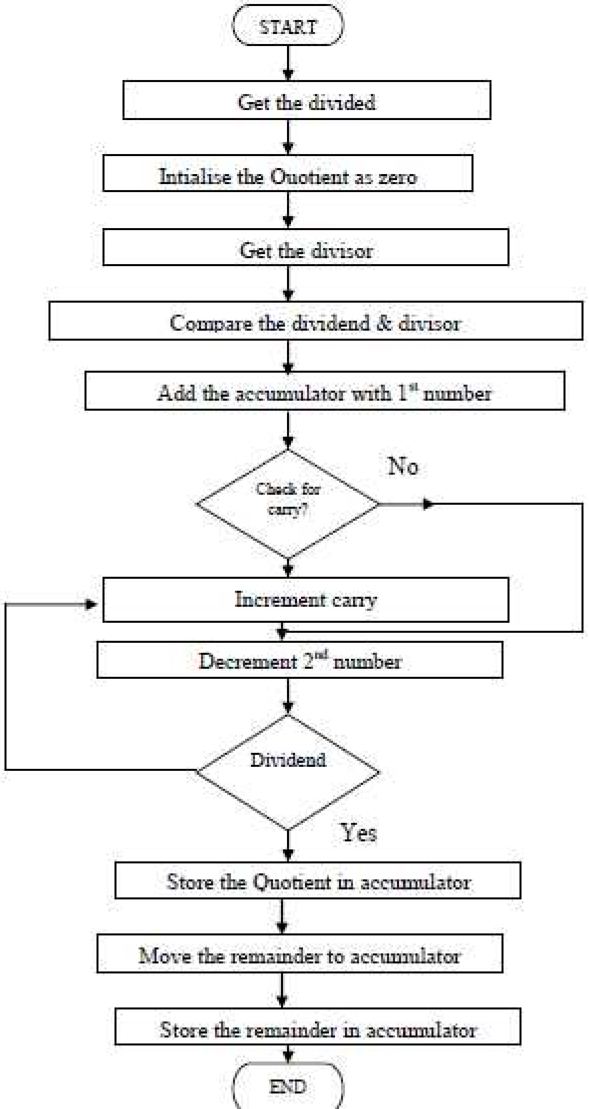
To write an assembly language program for dividing two 8 bit numbers using microprocessor kit.

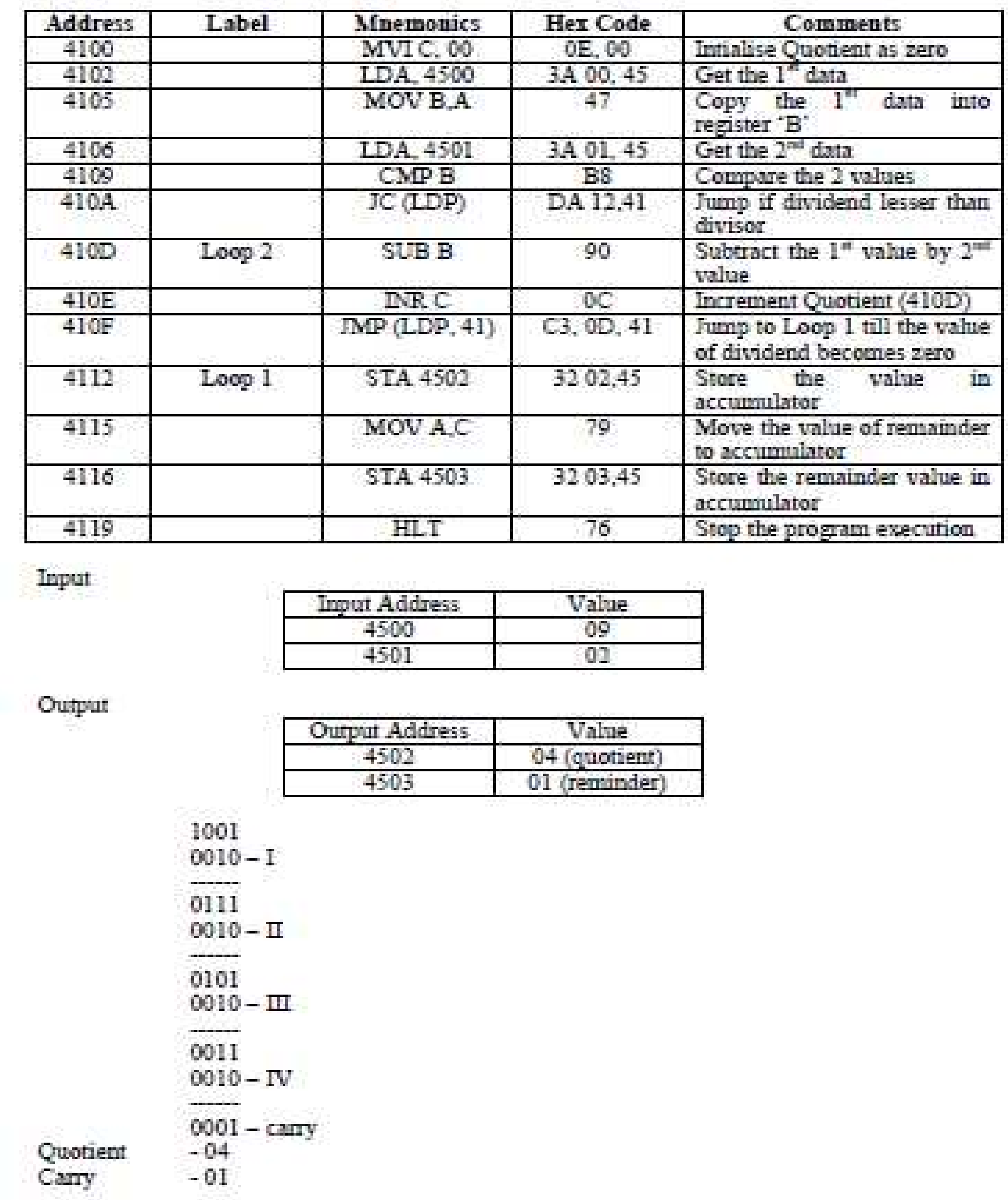
Apparatus required:

8085 microprocessor kit
(0-5V) DC battery

Algorithm:

Step 1 : Start the microprocessor
Step 2 : Intialise the Quotient as zero
Step 3 : Load the 1st 8 bit data
Step 4 : Copy the contents of accumulator into register ‘B’
Step 5 : Load the 2nd 8 bit data
Step 6 : Compare both the values
Step 7 : Jump if divisor is greater than dividend
Step 8 : Subtract the dividend value by divisor value
Step 9 : Increment Quotient
Step 10 : Jump to step 7, till the dividend becomes zero
Step 11 : Store the result (Quotient) value in accumulator
Step 12 : Move the remainder value to accumulator
Step 13 : Store the result in accumulator
Step 14 : Stop the program execution


START
Get the divided
Intialise the Quotient as zero
Get the divisor
Compare the dividend & divisor
Add the accumulator with 1st number
Check for carry?
No
Increment carry
Decrement 2nd number
Dividend
Yes
Store the Quotient in accumulator
Move the remainder to accumulator
Store the remainder in accumulator
END



Address	Label	Mnemonics	Hex Code	Comments
4100		MVI C, 00	0E, 00	Initialise Quotient as zero
4102		LDA, 4500	3A 00, 45	Get the 1st data
4105		MOV B,A	47	Copy the 1st data into register 'B'
4106		LDA, 4501	3A 01, 45	Get the 2nd data
4109		CMP B	B8	Compare the 2 values
410A		JC (LDP)	DA 12,41	Jump if dividend lesser than divisor
410D	Loop 2	SUB B	90	Subtract the 1st value by 2nd value
410E		INR C	0C	Increment Quotient (410D)
410F		JMP (LDP, 41)	C3, 0D, 41	Jump to Loop 1 till the value of dividend becomes zero
4112	Loop 1	STA 4502	32 02,45	Store the value in accumulator
4115		MOV A,C	79	Move the value of remainder to accumulator
4116		STA 4503	32 03,45	Store the remainder value in accumulator
4119		HLT	76	Stop the program execution

Input

Input Address	Value
4500	09
4501	02

Output

Output Address	Value
4502	04 (quotient)
4503	01 (reminder)

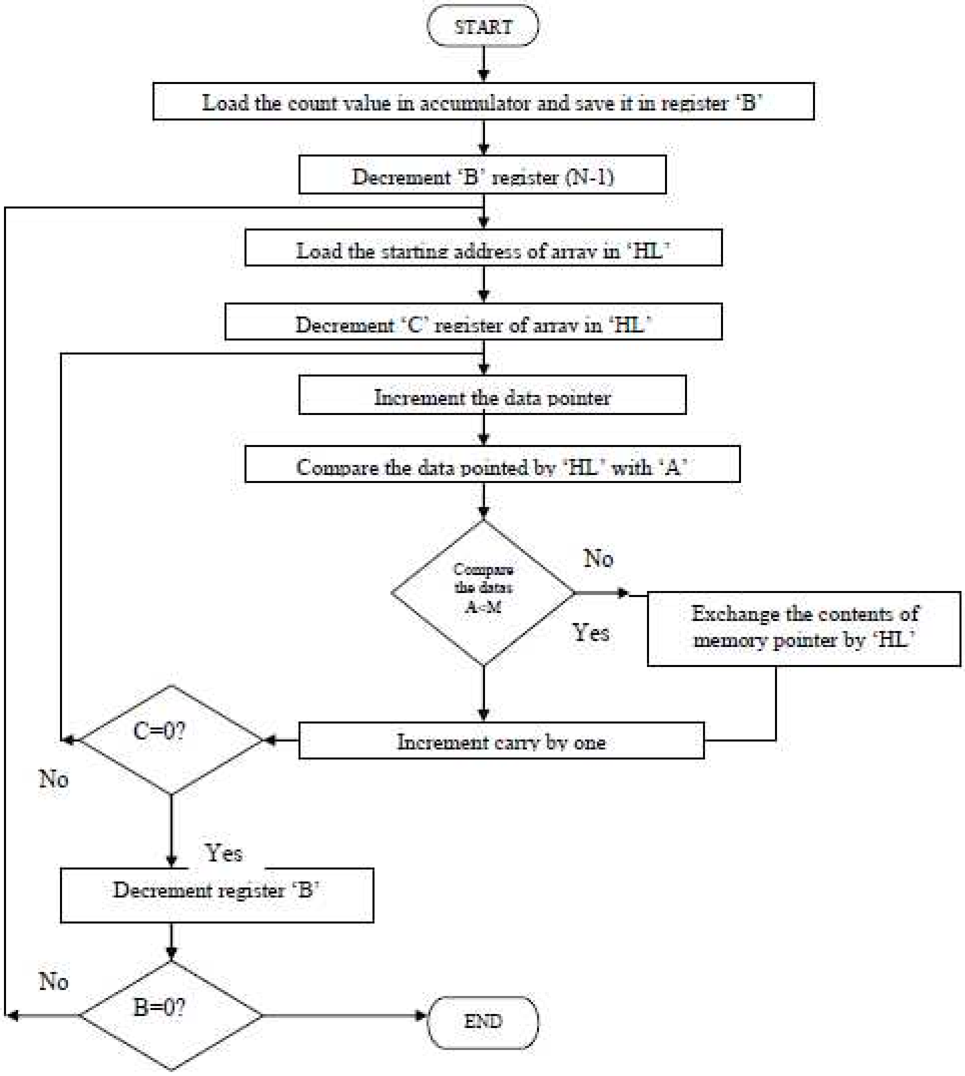
```
          1001
          0010 – I
          ------
          0111
          0010 – II
          ------
          0101
          0010 – III
          ------
          0011
          0010 – IV
          ------
          0001 – carry
Quotient  - 04
Carry     - 01
```

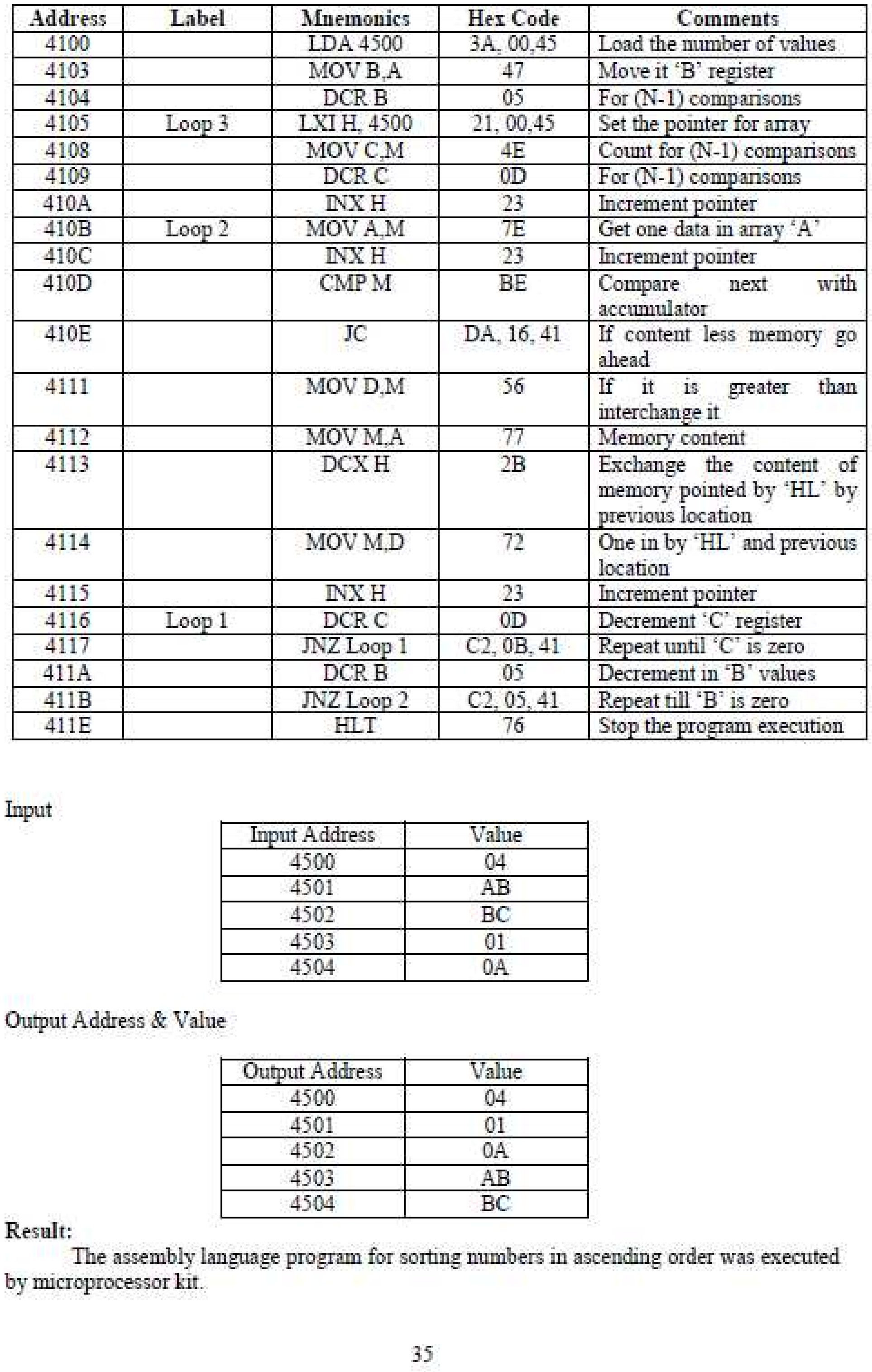
Result:

The assembly language program for division of two 8 bit numbers was executed using 8085 micro processing kit.

Experiment 5(a) :-

Aim:
To write a program to sort given ‘n’ numbers in ascending order
Apparatus required:
8085 microprocessor kit
(0-5V) DC battery
Algorithm:
Step 1 : Start the microprocessor
Step 2 : Accumulator is loaded with number of values to sorted and it is saved
Step 3 : Decrement 8 register (N-1) Repetitions)
Step 4 : Set ‘HL’ register pair as data array
Step 5 : Set ‘C’ register as counter for (N-1) repetitions
Step 6 : Load a data of the array in accumulator
Step 7 : Compare the data pointed in ‘HL’ pair
Step 8 : If the value of accumulator is smaller than memory, then jump to step 10.
Step 9 : Otherwise exchange the contents of ‘HL’ pair and accumulator
Step 10 : Decrement ‘C’ register, if the of ‘C’ is not zero go to step 6
Step 11 : Decrement ‘B’ register, if value of ‘B’ is not zero, go step 3
Step 12 : Stop the program execution


START
Load the count value in accumulator and save it in register ‘B’
Decrement ‘B’ register (N-1)
Load the starting address of array in ‘HL’
Decrement ‘C’ register of array in ‘HL’
Increment the data pointer
Compare the data pointed by ‘HL’ with ‘A’
Compare the datas A<M
No
Yes
Exchange the contents of memory pointer by ‘HL’
Increment carry by one
C=0?
No
Yes
Decrement register ‘B’
No
B=0?
END



Address	Label	Mnemonics	Hex Code	Comments
4100		LDA 4500	3A, 00,45	Load the number of values
4103		MOV B,A	47	Move it 'B' register
4104		DCR B	05	For (N-1) comparisons
4105	Loop 3	LXI H, 4500	21, 00,45	Set the pointer for array
4108		MOV C,M	4E	Count for (N-1) comparisons
4109		DCR C	0D	For (N-1) comparisons
410A		INX H	23	Increment pointer
410B	Loop 2	MOV A,M	7E	Get one data in array 'A'
410C		INX H	23	Increment pointer
410D		CMP M	BE	Compare next with accumulator
410E		JC	DA, 16, 41	If content less memory go ahead
4111		MOV D,M	56	If it is greater than interchange it
4112		MOV M,A	77	Memory content
4113		DCX H	2B	Exchange the content of memory pointed by 'HL' by previous location
4114		MOV M,D	72	One in by 'HL' and previous location
4115		INX H	23	Increment pointer
4116	Loop 1	DCR C	0D	Decrement 'C' register
4117		JNZ Loop 1	C2, 0B, 41	Repeat until 'C' is zero
411A		DCR B	05	Decrement in 'B' values
411B		JNZ Loop 2	C2, 05, 41	Repeat till 'B' is zero
411E		HLT	76	Stop the program execution

Input

Input Address	Value
4500	04
4501	AB
4502	BC
4503	01
4504	0A

Output Address & Value

Output Address	Value
4500	04
4501	01
4502	0A
4503	AB
4504	BC

Result:

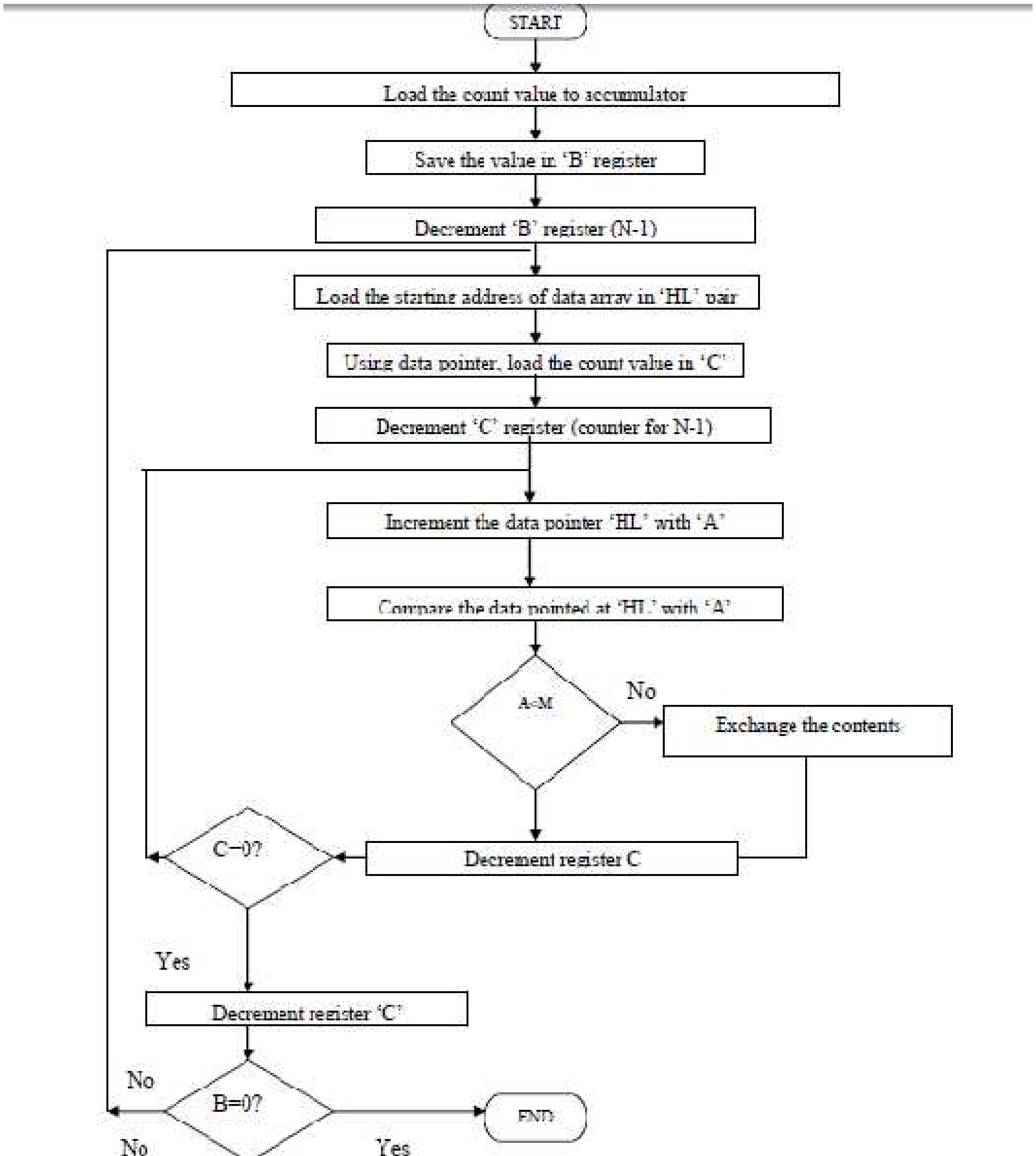
The assembly language program for sorting numbers in ascending order was executed by microprocessor kit.

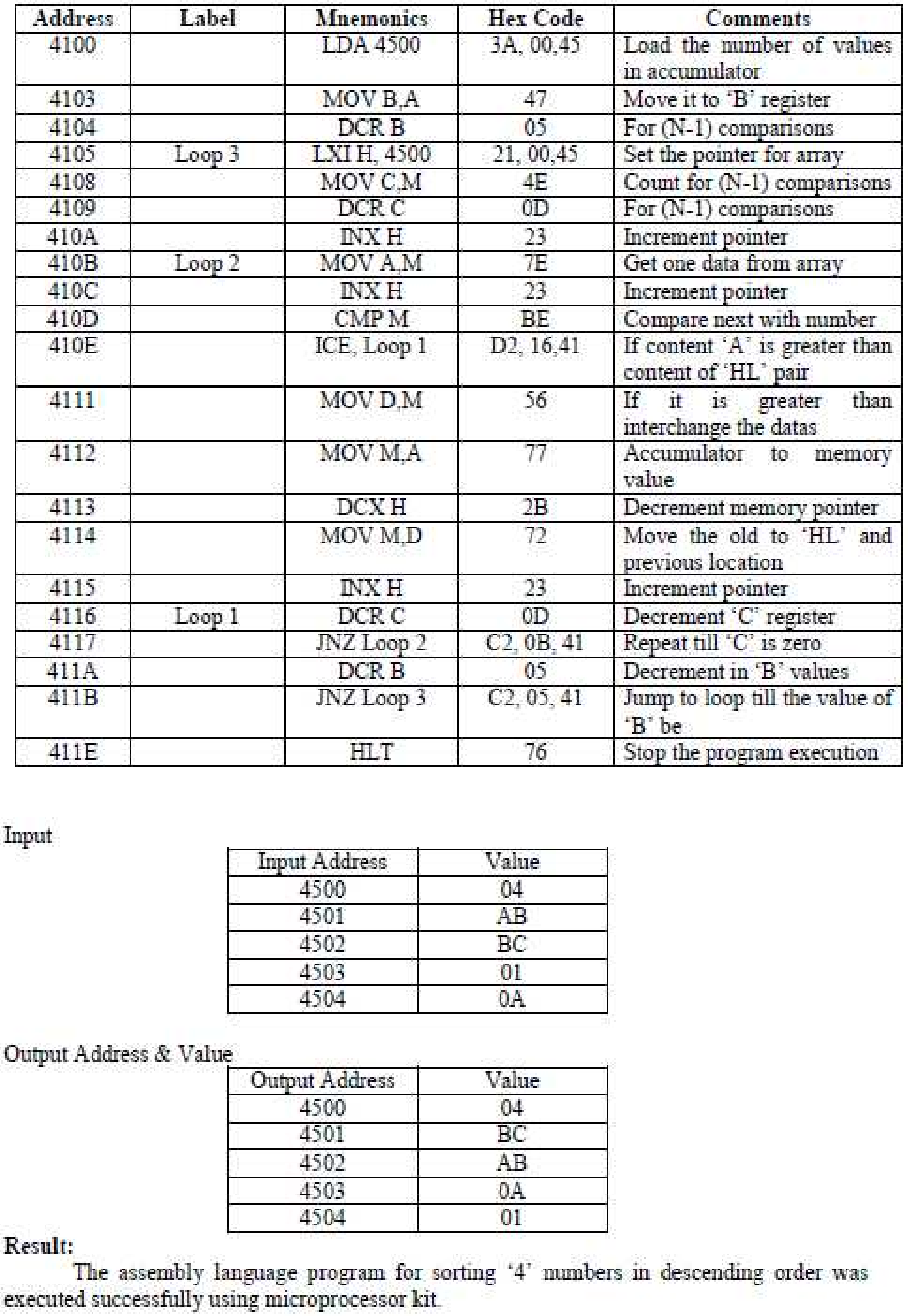
Experiment 5(b):-

Aim:
To write a program to sort given ‘n’ numbers in descending order

Apparatus required:
8085 microprocessor kit
(0-5V) DC battery

Algorithm:
Step 1 : Start the microprocessor
Step 2 : Load the number of values into accumulator and save the number of values in register ‘B’
Step 3 : Decrement register ‘B’ for (N-1) Repetitions
Step 4 : Set ‘HL’ register pair as data array address pointer and load the data of array in accumulator
Step 5 : Set ‘C’ register as counter for (N-1) repetitions
Step 6 : Increment ‘HL’ pair (data address pointer)
Step 7 : Compare the data pointed by ‘HL’ with accumulator
Step 8 : If the value of accumulator is larger than memory, then jump to step 10, otherwise next step.
Step 9 : Exchange the contents of memory pointed by ‘HL’ and accumulator
Step 10 : Decrement ‘C’ register, if the of ‘C’ is not zero go to step 6, otherwise next step.
Step 11 : Decrement ‘B’ register, if ‘B’ is not zero, go step 3, otherwise next step.
Step 12 : Stop the program execution


START
Load the count value to accumulator
Save the value in ‘B’ register
Decrement ‘B’ register (N-1)
Load the starting address of data array in ‘HL’ pair
Using data pointer, load the count value in ‘C’
Decrement ‘C’ register (counter for N-1)
Increment the data pointer ‘HL’ with ‘A’
Compare the data pointed at ‘HL’ with ‘A’
A<M
No
Exchange the contents
C=0?
Decrement register C
Yes
Decrement register ‘C’
No
B=0?
END
No
Yes



Address	Label	Mnemonics	Hex Code	Comments
4100		LDA 4500	3A, 00,45	Load the number of values in accumulator
4103		MOV B,A	47	Move it to 'B' register
4104		DCR B	05	For (N-1) comparisons
4105	Loop 3	LXI H, 4500	21, 00,45	Set the pointer for array
4108		MOV C,M	4E	Count for (N-1) comparisons
4109		DCR C	0D	For (N-1) comparisons
410A		INX H	23	Increment pointer
410B	Loop 2	MOV A,M	7E	Get one data from array
410C		INX H	23	Increment pointer
410D		CMP M	BE	Compare next with number
410E		ICE, Loop 1	D2, 16,41	If content 'A' is greater than content of 'HL' pair
4111		MOV D,M	56	If it is greater than interchange the datas
4112		MOV M,A	77	Accumulator to memory value
4113		DCX H	2B	Decrement memory pointer
4114		MOV M,D	72	Move the old to 'HL' and previous location
4115		INX H	23	Increment pointer
4116	Loop 1	DCR C	0D	Decrement 'C' register
4117		JNZ Loop 2	C2, 0B, 41	Repeat till 'C' is zero
411A		DCR B	05	Decrement in 'B' values
411B		JNZ Loop 3	C2, 05, 41	Jump to loop till the value of 'B' be
411E		HLT	76	Stop the program execution

Input

Input Address	Value
4500	04
4501	AB
4502	BC
4503	01
4504	0A

Output Address & Value

Output Address	Value
4500	04
4501	BC
4502	AB
4503	0A
4504	01

Result:

The assembly language program for sorting '4' numbers in descending order was executed successfully using microprocessor kit.

Experiment 6:-

Aim:

To write an program to calculate the factorial of a number (between 0 to 8)

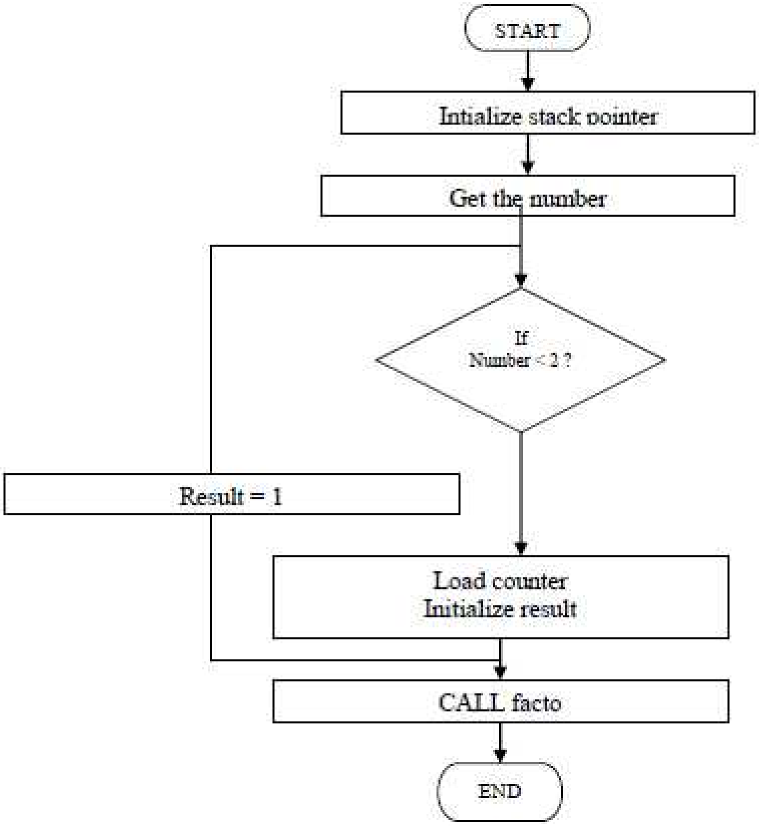
Apparatus required:

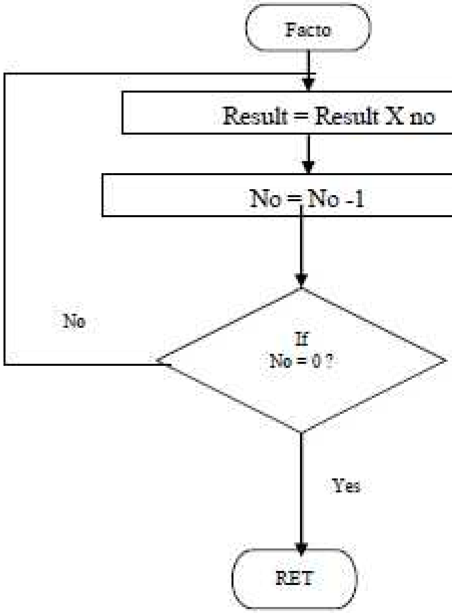
8085 microprocessor kit
(0-5V) power supply

Algorithm:

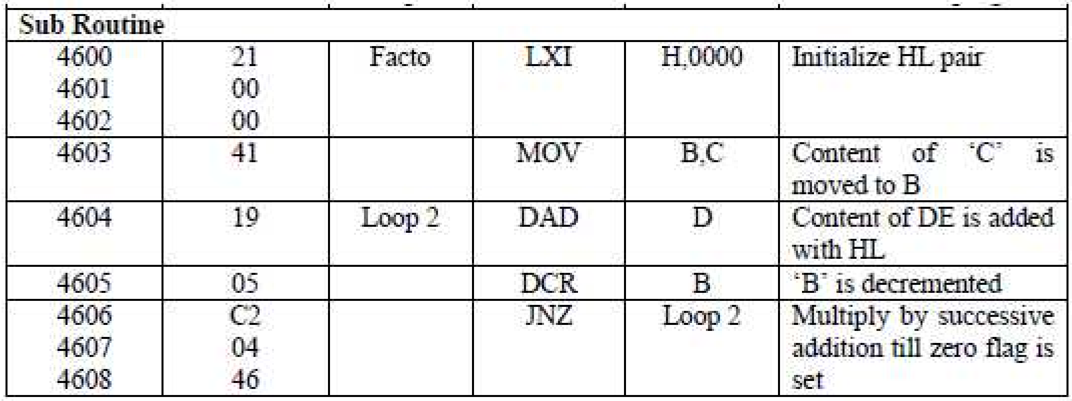
Step 1 : Intialize the stack pointer
Step 2 : Get the number in accumulator
Step 3 : Check for if the number is greater than 1. If no store the result otherwise go to next step.
Step 4 : Load the counter and initialize result
Step 5 : Now factorial program in sub-routine is called.
Step 6 : In factorial, initialize HL RP with 0. Move the count value to B Add HL content with Rp. Decrement count (for multiplication)
Step 7 : Exchange content of Rp (DE) with HL.
Step 8 : Decrement counter (for factorial) till zero flag is set.
Step 9 : Store the result
Step 10 : Hault

Memory address	Content
4250	05
4251	(12010)

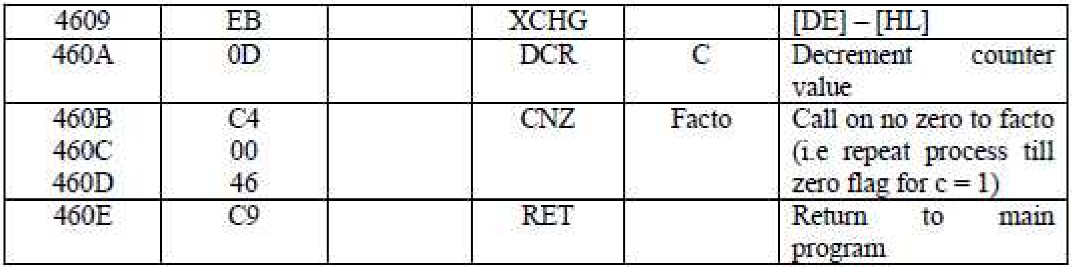




Memory Location	Hex Code	Label	Mnemonics		Comments
			Op code	Operand	
4200 4201 4202	3A 50 42		LDA	4250	Get the number in accumulator
4203 4204	FE 02		CPI	02H	Compare data with 2 and check it is greater than 1
4205 4206 4207	DA 17 42		JC	Loop 1	If cy =1 jump to loop 1 If cy = 0 proceed
4208	5F		MOV	E,A	Move content of A to E
4209 420A	16 00		MVI	D,00	Load this term as a result
420B	3D		DCR	A	Decrement accumulator by 1
420C	4F		MOV	C,A	Move 'A' content to 'C' (counter 1 less than A)
420D 420E 420F	CD 00 46		CALL	Facto	Call sub routine programe Facto
4210	EB		XCHG		Exchange (DE) – (HL)
4211 4212 4213	22 51 42		SHLD	4251	Store content of HL in specified memory location
4214 4215 4216	C3 1D 42		JMP	Loop 3	Jump to Loop 3
4217 4218 4219	21 00 01	Loop 1	LXI	H,0001$_H$	HL is loaded with data 01
421A 421B 421C	22 51 42		SHLD	4251	Store the result in memory
421D	76	Loop 3	HLT		Terminate the program



Sub Routine					
4600 4601 4602	21 00 00	Facto	LXI	H,0000	Initialize HL pair
4603	41		MOV	B,C	Content of 'C' is moved to B
4604	19	Loop 2	DAD	D	Content of DE is added with HL
4605	05		DCR	B	'B' is decremented
4606 4607 4608	C2 04 46		JNZ	Loop 2	Multiply by successive addition till zero flag is set



4609	EB		XCHG		[DE] – [HL]
460A	0D		DCR	C	Decrement counter value
460B 460C 460D	C4 00 46		CNZ	Facto	Call on no zero to facto (i.e repeat process till zero flag for c = 1)
460E	C9		RET		Return to main program

Memory address	Content
4250	04
4251	18

1 x 2 x 3 x 4 = 24

Hexadecimal

```
16 | 24
   | 1-8
```

Result:

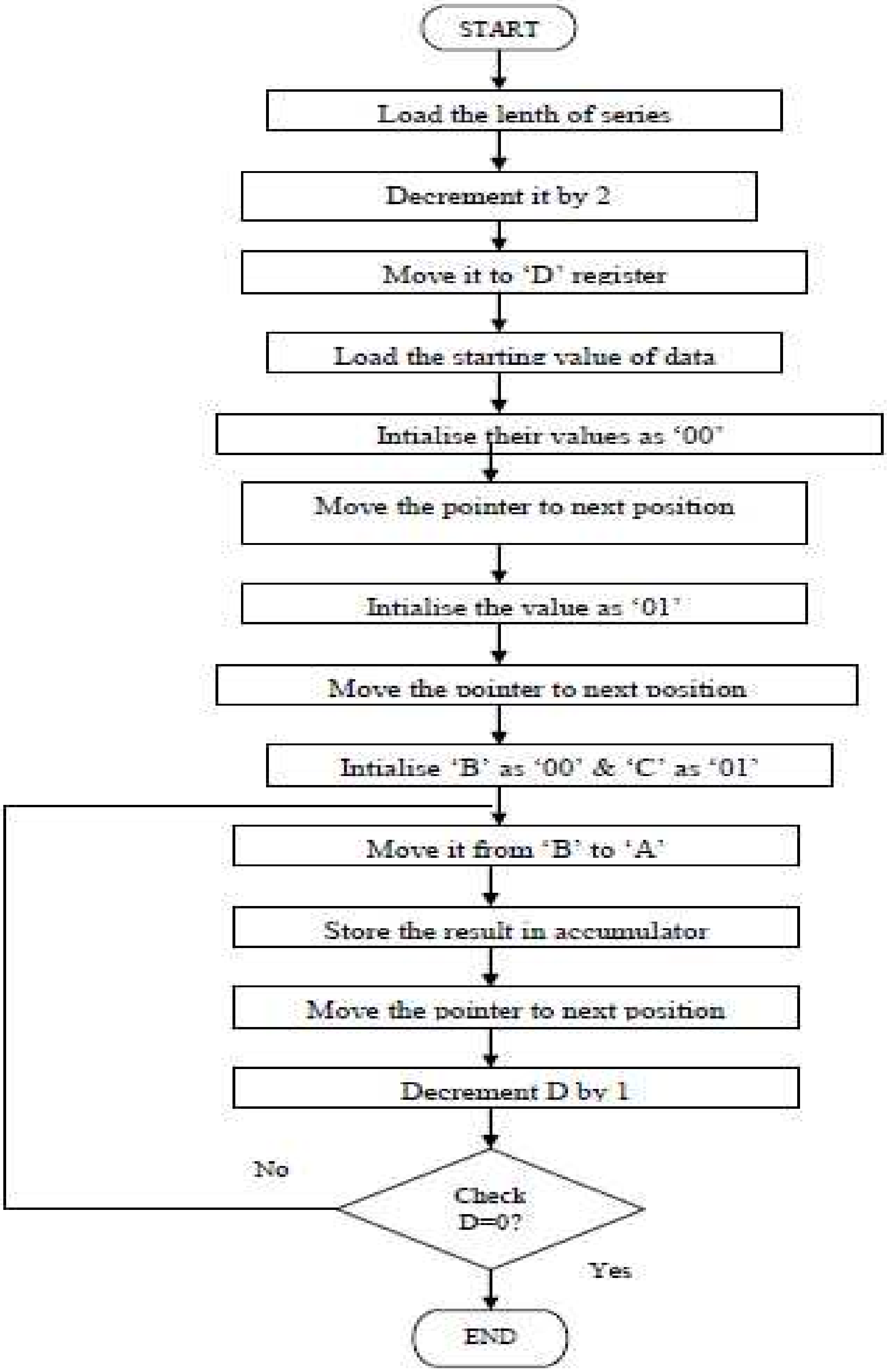
Thus, factorial program was done successfully

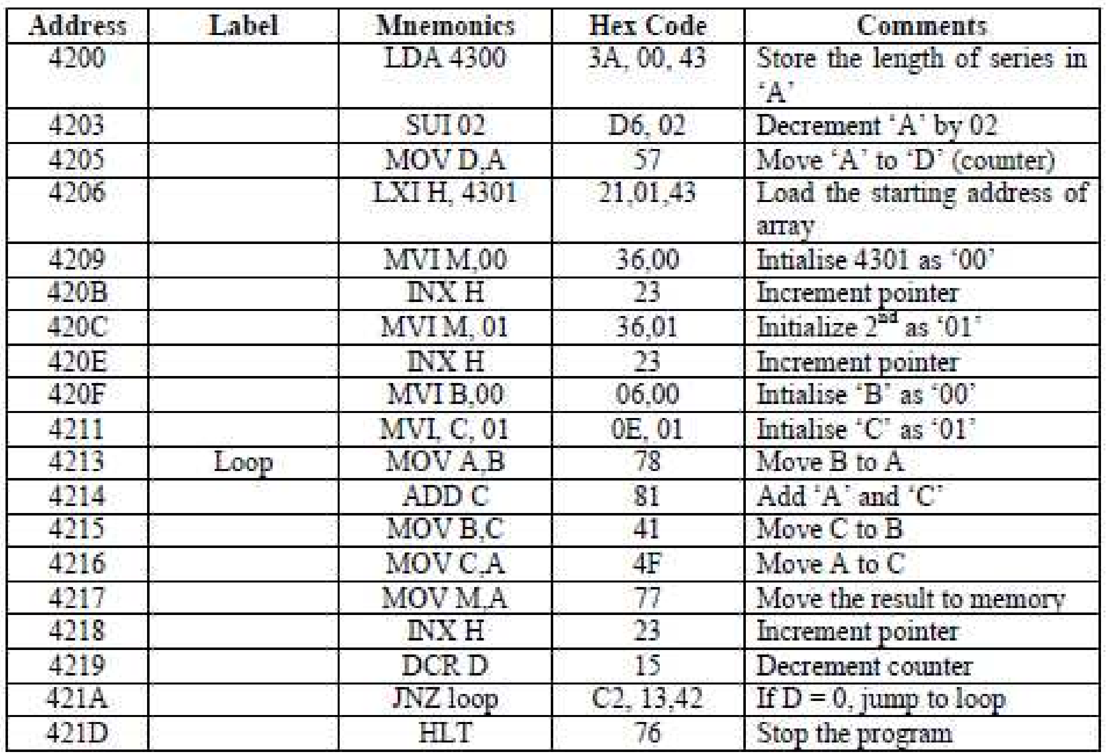
Experiment 7:-

Aim:
To write an assembly language program to displace Fibanocci Series.

Apparatus required:
8085 microprocessor kit
(0-5V) DC battery

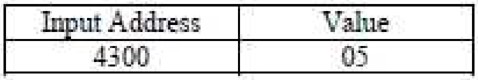
Algorithm:
Step 1 : Start the microprocessor
Step 2 : Load the length of series in the accumulator and decrement it by 2
Step 3 : Move the value to register ‘D’
Step 4 : Load the starting value of data value address
Step 5 : Intialise the 1st number as 00
Step 6 : Move the pointer to 2nd data and intialise them as ‘01’
Step 7 : Move the pointer to next position for next data
Step 8 : Intialise B as ‘00’ and C as ‘01’ for calculations
Step 9 : Copy the contents of ‘B’ to accumulator
Step 10 : Add the content of ‘C’ register to accumulator
Step 11 : Move the content ‘C’ to ‘B’ and ‘A’ to C
Step 12 : Now store the result to memory pointed by ‘HL’ pair
Step 13 : Move the pointer to next pointer
Step 14 : Decrement 0 by 1 for counter
Step 15 : If ‘D’ is not zero, go to step 9
Step 16 : if ‘D’ is zero, end the program


START
Load the lenth of series
Decrement it by 2
Move it to 'D' register
Load the starting value of data
Intialise their values as '00'
Move the pointer to next position
Intialise the value as '01'
Move the pointer to next position
Intialise 'B' as '00' & 'C' as '01'
Move it from 'B' to 'A'
Store the result in accumulator
Move the pointer to next position
Decrement D by 1
No
Check
D=0?
Yes
END



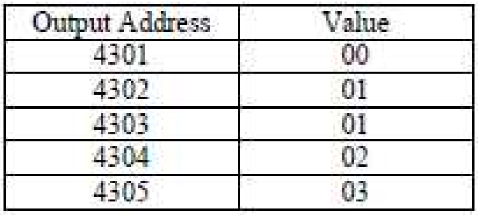
Address	Label	Mnemonics	Hex Code	Comments
4200		LDA 4300	3A, 00, 43	Store the length of series in 'A'
4203		SUI 02	D6, 02	Decrement 'A' by 02
4205		MOV D,A	57	Move 'A' to 'D' (counter)
4206		LXI H, 4301	21,01,43	Load the starting address of array
4209		MVI M,00	36,00	Intialise 4301 as '00'
420B		INX H	23	Increment pointer
420C		MVI M, 01	36,01	Initialize 2nd as '01'
420E		INX H	23	Increment pointer
420F		MVI B,00	06,00	Intialise 'B' as '00'
4211		MVI, C, 01	0E, 01	Intialise 'C' as '01'
4213	Loop	MOV A,B	78	Move B to A
4214		ADD C	81	Add 'A' and 'C'
4215		MOV B,C	41	Move C to B
4216		MOV C,A	4F	Move A to C
4217		MOV M,A	77	Move the result to memory
4218		INX H	23	Increment pointer
4219		DCR D	15	Decrement counter
421A		JNZ loop	C2, 13,42	If D = 0, jump to loop
421D		HLT	76	Stop the program

Input



Input Address	Value
4300	05

Output



Output Address	Value
4301	00
4302	01
4303	01
4304	02
4305	03

00 + 01 = 01
01+ 01 = 02
02 + 01 = 03

Result:

The assembly language for Fibonaci series was executed successfully using 8085 microprocessor kit.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: PHP, .NET Lab
Course Code: CS593
L-T-P scheme: 1-0-2 **Course Credit: 2**

Objectives:

- Understand the importance of the web as a medium of communication.
- Understand the principles of creating an effective web page, including an in-depth consideration of information architecture.
- Become familiar with graphic design principles that relate to web design and learn how to implement these theories into practice.
- Develop skills in analyzing the usability of a web site.
- Learn the language of the web:Sql, .net, PHP, HTML and CSS.

Learning Outcomes:

- Apply critical thinking and problem solving skills required to successfully design and implement a web site.
- Demonstrate the ability to analyse, identify and define the technology required to build and implement a web site.
- Demonstrate knowledge of artistic and design components that are used in the creation of a web site.
- Utilize and apply the technical, ethical and interpersonal skills needed to function in a cooperative environment.
- Apply critical thinking and problem solving skills required to successfully design and implement a web site.
- Demonstrate the ability to analyze, identify and define the technology required to build and implement a web site.

Course Contents:
Exercises that must be done in this course are listed below:
Exercise No.1: Create a form in HTML for entering value for some specific fields. (Registration Page)
Exercise No.2: Create table in SQL for storing data of registration page. (Using sql query)
Exercise No.3: Create a webpage to show the data which is entered in sql tables through registration page.
Exercise No. 4: Create a web page to file upload option, so user can upload document on website.
Exercise No. 5: Create a webpage to show the uploaded document.
Exercise No. 6: assemble all the web page to create a website for a specific organization and set authentication for user (Minor Project).
Exercise No. 7: Create master page for previous developed pages.
Exercise No. 8: Apply validators for all fields which are used in previous developed pages.
Exercise No. 9: Major project.
Text Book:
1. Maurice J. Bach, Design of the UNIX Operating System, PHI.

Recommended Systems/Software Requirements:

1. Desktop PC with minimum of 166 MHZ or faster processor with at least 1 GB RAM and 160 GB disk space.
2. Visual studio 2012, Microsoft sql server 2008 R2

Experiment No 1: Registration Page

Aim: Create a form in HTML for entering value for some specific fields.

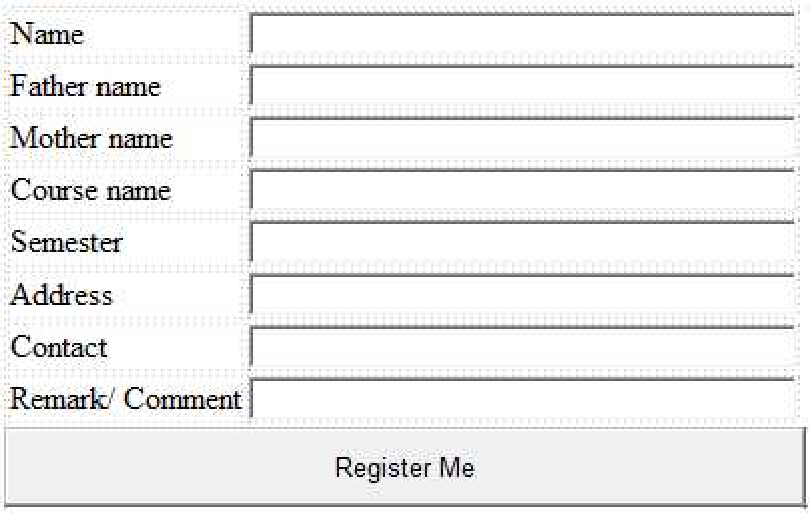
Description:

Here a registration page will be developed which have the following fields.
Name, Father Name, Mother name, Course name, Semester, Address, Contact number, Comment, and submit button

INPUT 1:

Name: Subrat
Father name: Gautam
Mother name: Gautam
Course: B.tech
Semester: 5th
Address: Jaipur
Contact: 0123456789

OUTPUT 1:



Name	
Father name	
Mother name	
Course name	
Semester	
Address	
Contact	
Remark/ Comment	

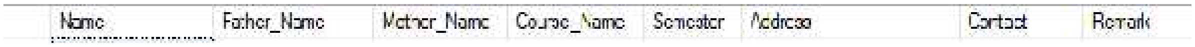
Experiment No 2: Using sql query

Aim: Create table in SQL for storing data of registration page.

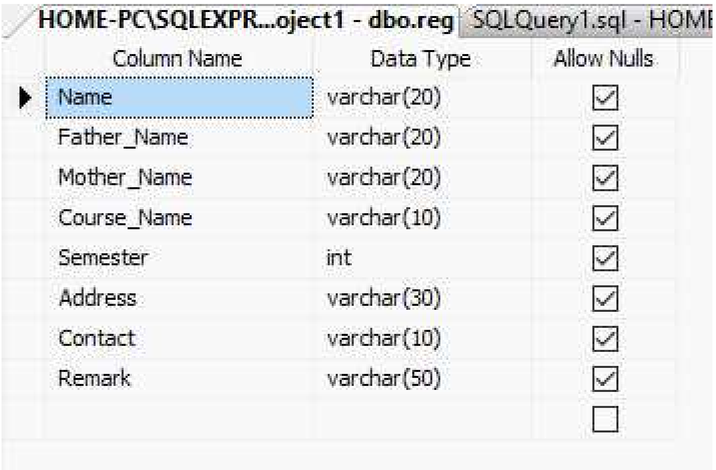
Description:

Here student have to develop a Sql table in which the following should be cover:
Name, Father Name, Mother name, Course, Semester, Address, Contact, Comment

OUTPUT 1:



Name	Father_Name	Mother_Name	Course_Name	Semester	Address	Contact	Remark



HOME-PC\SQLEXPR...oject1 - dbo.reg | SQLQuery1.sql - HOMI

Column Name	Data Type	Allow Nulls
Name	varchar(20)	☑
Father_Name	varchar(20)	☑
Mother_Name	varchar(20)	☑
Course_Name	varchar(10)	☑
Semester	int	☑
Address	varchar(30)	☑
Contact	varchar(10)	☑
Remark	varchar(50)	☑
		□

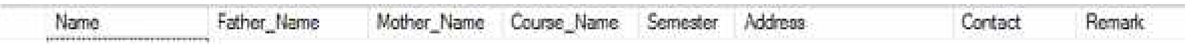
Experiment No: 3 Grid View

Aim: Create a webpage to show the data which is entered in sql tables through registration page.

Description:

Here student have to show the data which is stored in sql table with the help of grid view.

Output 1:



Name	Father_Name	Mother_Name	Course_Name	Semester	Address	Contact	Remark

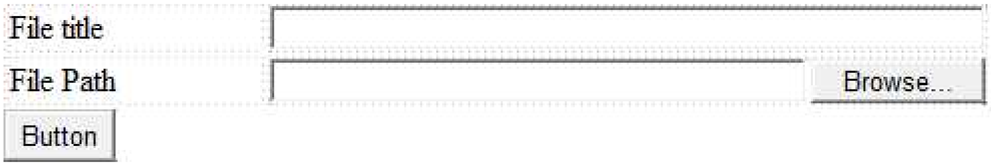
Experiment No:4 File upload

Aim: Create a web page to file upload option, so user can upload document on website.

Description:

Here student have to develop a web page from where user can upload any file.

OUTPUT 1:



File title

File Path

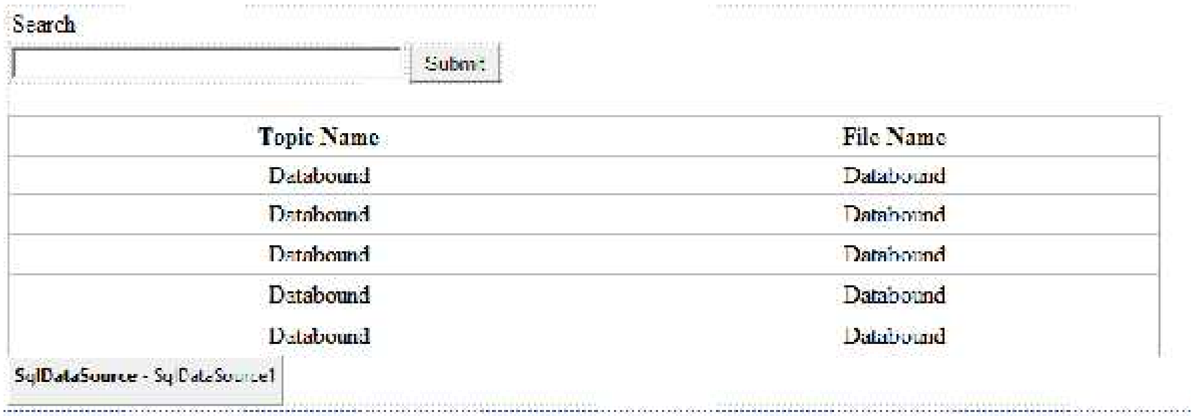
Experiment No: 5Grid view with hyperlink option.

AIM: Create a webpage to show the uploaded document.

Description:

Here student have to develop a grid view so user can see the list of file which is uploaded on website.

OUTPUT 1:

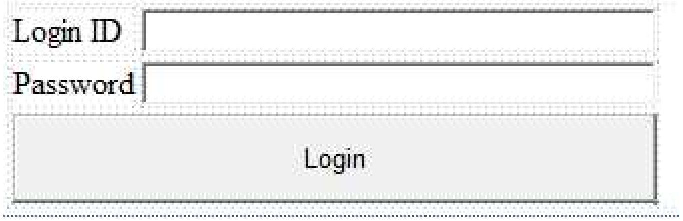


Topic Name	File Name
Databound	Databound
Databound	Databound
Databound	Databound
Databound	Databound
Databound	Databound

Experiment No: 6 Authentications (Login)
AIM: Develop login page.

Description:
Here student have to develop a webpage where user can enter registered id and password to access system.

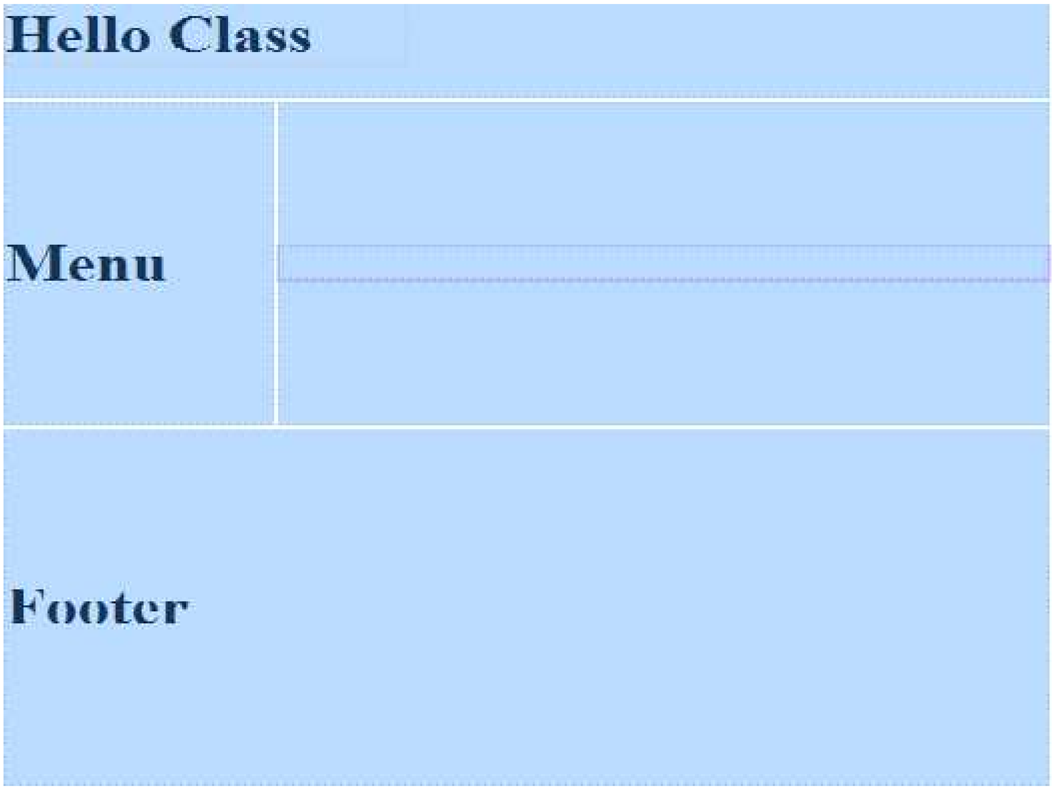
OUTPUT 1:



Experiment No 7: Master pages
AIM: Create master page for previous developed pages.

Description:
Here student have to develop master page to provide good look and strong functionality to website.

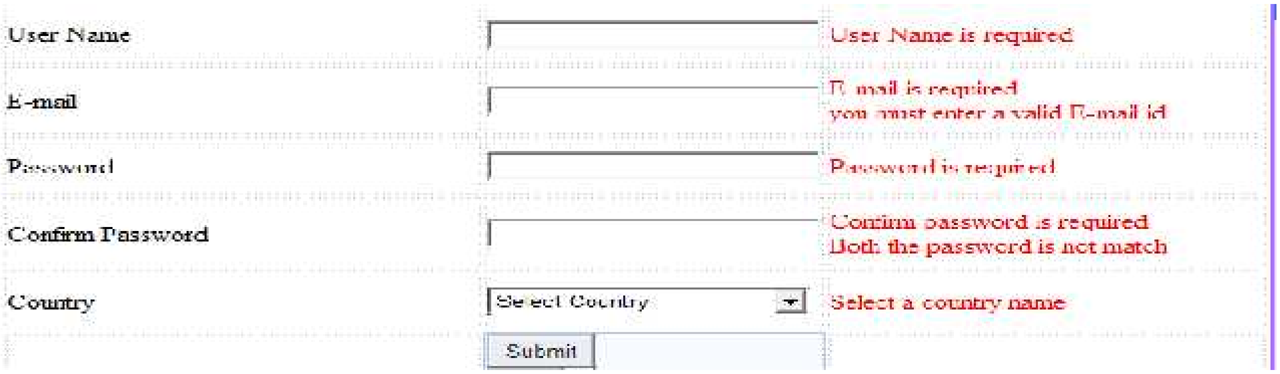
OUTPUT 1:



Experiment No: 8 Validators.
AIM: Apply validators for all fields which are used in previous developed pages

Description:
Here student have to set validators for fields in registration page.

OUTPUT 1:



User Name		User Name is required
E-mail		E-mail is required you must enter a valid E-mail id
Password		Password is required
Confirm Password		Confirm password is required Both the password is not match
Country	Select Country	Select a country name
	Submit	

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Circuit Theory & Network Lab
Course Code: CS504A
L-T-P scheme: 0-0-3 **Course Credit: 2**

Objectives:

1. To learn and understand to design electrical circuit practically or through any simulation software.
2. To provide an understanding of the circuit designing aspects in bread board.
3. To provide a window to investigate and verify various laws, theories, and concepts regarding electrical circuits practically or virtually by simulation software.

Learning Outcomes: The students will have a detailed knowledge of electrical circuit design using different electrical elements and sources through bread board or by any simulation software. The students will also get the opportunity & better understanding of various concepts, laws, & theories applicable in any electrical circuit by investigating and verifying them in the practically designed circuit.Upon the completion of Operating Systems practical course, the student will be able to:

- **Understand** and implement electrical circuit design knowledge to realize any electrical circuit practically
- **Use** modern simulation software to recreate any practical circuit virtually**.**
- **Understand** the benefits of circuit design in bread board.
- **Analyze** designed circuit to see weather various laws, theories, and concepts regarding electrical circuits holds or not.
- **Simulate**electrical circuits through any simulation software to check weather various laws, theories, and concepts regarding electrical circuits they studied holds or not.

Course Contents:
Exercises that must be done in this course are listed below:

Exercise No.1: Verification of Thevenin’s Theorem: Hardware/Simulation
Exercise No. 2: Verification of Norton’s Theorem: Hardware/Simulation
Exercise No. 3: Verification of TheoremSuperposition Theorem: Hardware/Simulation
Exercise No. 4: Verification of Maximum Power Transfer: Hardware/Simulation
Exercise No. 5: Study of Z-parameters of any practical circuit treated as Two-port network: Hardware/Simulation
Exercise No. 6: Study of Y-parameters of any practical circuit treated as Two-port network: Hardware/Simulation
Exercise No. 7: Study Resonance of a series RLC circuit: Hardware

Text Book:

1. S.P.Ghosh&A.Chakraborty , “Circuit Theory & Networks”, TMH
2. Muhammad H. Rashid, “Introduction to PSpice Using Orcad for circuits and Electronics”, Pearson Education.

Recommended Systems/Software Requirements:

1. MATLAB
2. SPICE.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Experiment No: 1.Verification of Thevenin's Theorem experimentally
Aim: To Verify the Thevenin's Theorem in breadboard or through MATLAB/SPICE.

Description:

APPARATUS REQUIRED:-If Practically by circuit design

(i) Bread Board
(ii) Connecting Wire
(iii) Different values of resistances
(iv) A Dc power Source

If by any simulation software

(i) MATLAB/SPICE

THEORY:

Sometimes, we wish to determine the response in a single load resistance in a network. Thevenin Theorem enables us to replace the remainder of the network by a simple equivalent circuit. Determining response in the load resistance, then becomes easier. The use of Thevenin Theorem is specially very helpful and time saving when we wish to find the response for different values of load resistance. Thevenin Theorem states that current through a load resistance connected across any two points of an active network can be obtained by the formula:

$$I_L=V_{th}/(R_{th}+R_L)$$

Where V_{th} is the open circuit voltage at the terminals of R_L when R_L is disconnected and R_{th} is the equivalent resistance viewed from the load terminals when all the sources replaced by their internal resistance only(Deactivating all the sources).

CIRCUIT DIAGRAM:

Draw the circuit diagram as per the resistance and circuit are given in the lab.

CALCULATIONS:

Calculate the theoretical/simulation data's of the given circuit

Values	V_{th}	R_{th}	I_L
Theoretical Value			
Practical Value			

Percentage Error= [(Observed-Calculated)/Calculated]*100

RESULT:

The percentage error is found to be__%.

DISCUSSION:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Experiment No: 2.Verification of Norton's Theorem experimentally
Aim: To Verify the Norton's Theorem in breadboard or through MATLAB/SPICE.

Description:

APPARATUS REQUIRED:-If Practically by circuit design

(v) Bread Board
(vi) Connecting Wire
(vii) Different values of resistances
(viii) A Dc power Source

If by any simulation software

(ii) MATLAB/SPICE

THEORY:

Sometimes, we wish to determine the response in a single load resistance in a network. Norton's Theorem enables us to replace the remainder of the network by a simple equivalent circuit. Determining response in the load resistance, then becomes easier. The use of Norton Theorem is specially very helpful and time saving when we wish to find the response for different values of load resistance. Thevenin Theorem states that current through a load resistance connected across any two points of an active network can be obtained by the formula:

$$I_L=(I_N*R_N)/(R_N+R_L)$$

Where I_N is the Short circuit current at the terminals of R_L when R_LShort circuited and wefind out the short circuit current through the short circuit terminal and R_N is the norton'sequivalent resistance viewed from the load terminals when all the sources are replaced by their internal resistance only(Deactivatingall the sources).

CIRCUIT DIAGRAM:

Draw the circuit diagram as per the resistance and circuit are given in the lab.

CALCULATIONS:

Calculate the theoretical/simulation data's of the given circuit

OBSERVATION TABLE:

Values	I_N	R_N	I_L
Theoretical Value			
Practical Value			

Percentage Error= [(Observed-Calculated)/Calculated]*100

RESULT:

The percentage error is found to be__%.

DISCUSSION:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Experiment No: 3.Verification of Superposition Theorem experimentally
Aim: To Verify the Superposition Theorem in breadboard or through MATLAB/SPICE.

Description:

APPARATUS REQUIRED:-If Practically by circuit design

(i) Bread Board
(ii)Connecting Wire
(iii) Different values of resistances
(iv) A Dc power Source

If by any simulation software

(i) MATLAB/SPICE

THEORY:

Superposition theorem states that in a linear network containing several independent sources, the overall response at any point in the network equals the sum of responses due to each independent source considered separately with all other independently sources made inoperative(short circuited). To make a source inoperative, it is short circuited leaving behind its internal resistance if it is a voltage source, and it is open circuited leaving behind its internal resistance if it is a current source.

In most electrical circuit analysis problems, a circuit is energized by a single independent energy source. In such cases, it is quite easy to find the response (i.e., current, voltage, power) in a particular branch of the circuit using simple network reduction techniques (i.e., series parallel combination, star delta transformation, etc.).

However, in the presence of more than one independent source in the circuit, the response cannot be determined by direct application of network reduction techniques. In such a situation, the principle of superposition may be applied to a linear network, to find the resultant response due to all the sources acting simultaneously.

The superposition theorem is based on the principle of superposition. The principle of Superposition states that the response (a desired current or the voltage) at any point in the linear network having more than one independent source can be obtained as the sum of responses caused by the separate independent sources acting alone. The validity of principle of superposition means that the presence of one excitation sources does not affect the response due to other excitations.

CIRCUIT DIAGRAM:

Draw the circuit diagram as per the resistance and circuit are given in the lab.

Calculate the theoretical/simulation data's of the given circuit

OBSERVATION TABLE:

Values	I_3	I_3'	I_3''
Theoretical Value			
Practical Value			

I_{3} = Current through the load terminal when we deactivate second source and consider for first source

I_3'= Current through the load terminal when we deactivate first source and consider for second source

I_3''= $I_{3+}I_3$'

Percentage Error= [(Observed-Calculated)/Calculated]*100

RESULT:

The percentage error is found to be__%.

DISCUSSION:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Experiment No: 4.Verification of Maximum Power Transfer Theorem experimentally
Aim: To Verify the Maximum Power Transfer Theorem in breadboard or through MATLAB/SPICE.

Description:

APPARATUS REQUIRED:-If Practically by circuit design

(i) Bread Board
(ii) Connecting Wire
(iii) Different values of resistances
(iv) A Dc power Source

If by any simulation software

(i) MATLAB/SPICE

THEORY:

This theorem is applicable for analysing communication networks. According to this theorem" a resistive load will draw the maximum power from a network when the load resistance is equal to the resistance of the network as viewed from its output terminals, with all energy sources removed leaving behind their internal resistances." If R_L is the load resistance connected across terminals a and b which consist of variable DC supply and internal resistance is R_S, then according to this theorem, the load resistance will draw maximum power when it is equal to R_S i.e. $R_L = R_{S.}$

And the maximum power drawn= $\mathbf{V^2_{oc}/4\ R}_L$

Where, Voc is the open circuit voltage at the terminals from which R_L is disconnected.

The variable resistor taken should be larger than fixed resistor. Then only power can be calculated.

CIRCUIT DIAGRAM:

Draw the circuit diagram as per the resistance and circuit are given in the lab.

CALCULATIONS:

Calculate the theoretical/simulation data's of the given circuit

OBSERVATION TABLE:

S.No	Load Resistance(R_L)	$I_{L(}$Load Current)	Power(P=I_L^2*R_L)

RESULT:

Plot a graph between load resistance and power and observe that the power will be maximum when (Load resistance= Internal Resistance)

DISCUSSION:

Experiment No: 5.Study of Z-parameters of a Two-port network experimentally
Aim: To Study Z-parameters of any practical circuit treated as Two-port network in breadboard or through MATLAB/SPICE.

Description:

APPARATUS REQUIRED:-If Practically by circuit design

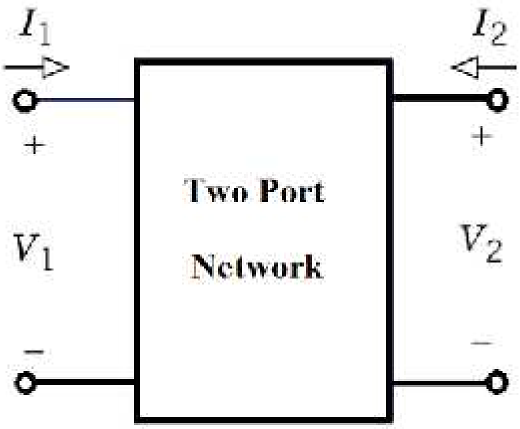
(i) Bread Board
(ii) Connecting Wire
(iii) Different values of resistances
(iv) A Dc power Source

If by any simulation software

(i) MATLAB/SPICE

THEORY

A two-Port network basically consists in isolating either a complete circuit or part of it and finding its characteristics parameters. Once this is done, the isolated part of the circuit becomes a “black box” with a set of distinctive properties, enabling us to abstract away its specific build up, thus simplifying analysis.



Here,

V_1 = Input voltage

V_2 = Output voltage

I_1 = Input current

I_2 = output current

Z-model: In the Z-model or impedance model the two currents I_1 & I_2 are assumed to be known and the voltage V_1& V_2 can be found by:

$$\begin{pmatrix} V1 \\ V2 \end{pmatrix} = \begin{pmatrix} Z11 & Z12 \\ Z21 & Z22 \end{pmatrix} \begin{pmatrix} I1 \\ I2 \end{pmatrix}$$

Where,

$Z11 = V1/_{I1}$ taking I2=0 $\qquad$ $Z12 = V1/_{I2}$ taking I1=0

Z21 = $V2/_{I1}$ taking I2=0 Z22 = $V2/_{I2}$ taking I1=0

CIRCUIT DIAGRAM:

Draw the circuit diagram as per the resistance and circuit are given in the lab.

CALCULATIONS:

Calculate the theoretical/simulation data's of the given circuit

OBSERVATION TABLE:

Sl. No.	When Output is open circuited (i.e. I2 =0)			When Input is open circuited (i.e. I1 =0)		
	V1	V2	I1	V1	V2	I2
1.						
2.						
3.						
4.						
5.						

Percentage Error= [(Observed-Calculated)/Calculated]*100

RESULT:

The percentage error is found to be__%.

DISCUSSION:

Experiment No: 6. Study of Y-parameters of a Two-port network experimentally
Aim: To Study Y-parameters of any practical circuit treated as Two-port network in breadboard or through MATLAB/SPICE.

Description:

APPARATUS REQUIRED:-If Practically by circuit design

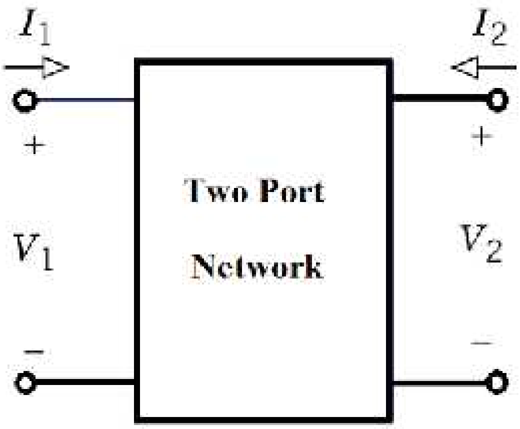
(i) Bread Board
(ii) Connecting Wire
(iii) Different values of resistances
(iv) A Dc power Source

If by any simulation software

(i) MATLAB/SPICE

THEORY

A two-Port network basically consists in isolating either a complete circuit or part of it and finding its characteristics parameters. Once this is done, the isolated part of the circuit becomes a "black box" with a set of distinctive properties, enabling us to abstract away its specific build up, thus simplifying analysis.



Here,

V_1 = Input voltage

V_2 = Output voltage

I_1 = Input current

I_2 = output current

Y-model: In the Y-model or admittance model, the two voltages V_1& V_2 are assumed to be known and the currents I_1 & I_2can be found by:

$$\begin{pmatrix} I1 \\ I2 \end{pmatrix} = \begin{pmatrix} Y11 & Y12 \\ Y21 & Y22 \end{pmatrix} \begin{pmatrix} V1 \\ V2 \end{pmatrix}$$

Where,

Y11 = $I1/_{V1}$ taking V2=0 Y12 = $I1/_{V2}$ taking V1=0

Y21 = $I2/_{V1}$taking V2=0 Y22 = $I2/_{V2}$taking V1=0

CIRCUIT DIAGRAM:

Draw the circuit diagram as per the resistance and circuit are given in the lab.

CALCULATIONS:

Calculate the theoretical/simulation data's of the given circuit

OBSERVATION TABLE:

Sl. No.	When Output is short circuited (i.e. V2 =0)			When Input is short circuited (i.e. V1 =0)		
	V1	I1	I2	V2	I1	I2
1.						
2.						
3.						
4.						
5.						

Percentage Error= [(Observed-Calculated)/Calculated]*100

RESULT:

The percentage error is found to be__%.

DISCUSSION:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

Experiment No: 7. Study of Resonance experimentally
Aim: To Study Resonance of a series RLC circuit in breadboard

Description:

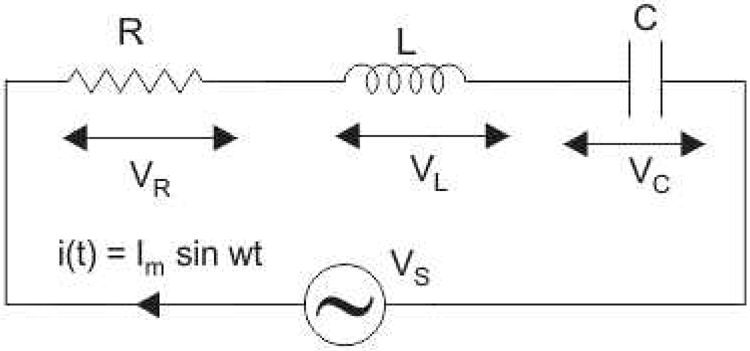
APPARATUS REQUIRED:-

(i) Bread Board
(ii) Connecting Wire
(iii) Different values of resistances
(iv) A Dc power Source

THEORY

Resonance occurs in an electrical circuit excited by AC source when the net inductive reactance (X_L) and net capacitive reactance (X_C) become equal either because for a fixed frequency as circuit's inductance (L) is equal to capacitance (C) or due to a particular frequency where $X_L = X_C$.

A simple series RLC circuit



When stated above condition arises in a circuit as given here the net impedance (Z_{net}) becomes minimum or equal to only resistance (R) of the circuit and the circuit starts operating in resonance with unity power factor. At resonance the value of net current is maximum as the net impedance is minimum.

CIRCUIT DIAGRAM:

Draw the circuit diagram as per the resistance, inductance & capacitancevalues as given in the lab.

CALCULATIONS:

Calculate the theoretical data's of the given circuit to find out the value of net impedance and current at resonance.

OBSERVATION TABLE:

SL. No.	Different frequencies	Load/source current
1.	f_1	
2.	f_2	
3.	f_0	
4.	f_3	
5.	f_4	

Here, f_0 is resonance frequency and $f_1<f_2<f_0<f_3<f_4$

Percentage Error= [(Observed-Calculated)/Calculated]*100

RESULT:

The percentage error is found to be__%.

DISCUSSION:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Digital Communication Lab
Course Code: CS594B
L-T-P scheme: 0-0-3 **Course Credit: 2**

Objectives:
Digital communication uses electrical signaling methods to transmit information over a physical channel separating a transmitter and receiver with the channel properties often time varying. This course presents the theory and practice of digital communication including signal design, modulation methods, demodulation methods, wireless channel basics and the application of this to the design of modern OFDM systems.

Learning Outcomes:
Upon successful completion of this course, the students will be able to;
1. Understand the basic concepts of advanced digital communication systems
2. Apply different modulation schemes to baseband signals
3. Analyze the BER characteristics of Baseband Modulated signals

Course Contents:
Exercises that must be done in this course are listed below:

1. To study different types of signal sampling and its reconstruction.
2. To study Pulse Position Modulation
3. To generate the pulse width modulated and demodulated signals**.**
4. Study of Time Division Multiplexing System.
5. To study delta modulation & demodulation and observe the effect of slope overload
6. To study the operation of Amplitude Shift Keying modulation and demodulation with the help of circuit connections.
7. To study the operation of Frequency Shift Keying modulation and demodulation with the help of kit.
8. To generate Pulse shift key (PSK) With Wave forms.
9. To study Quadri Phase Shift Keying (QPSK).
10. To study Differential Phase Shift Keying (DPSK).

Text Book:
1.J. D. Proakis and M. Salehi (2008), Digital Communication,
2.McGraw HilDavid Silage (2009), Digital Communication

Recommended Kits and Equipment Requirements:
1. Digital communication,Advance Digital communication kits
2. DSO, FG, Probes

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

EXPERIMENT-1

Aim: To study different types of signal sampling and its reconstruction.

ApparatusRequired:

1. Sampling and its reconstruction Kit-DCL01
2. Digital Storage Oscilloscope(DSO)
3. Power supply
4. Patch cords

Theory: Regardless of the sampling method used, by definition it captures only pieces of the message. So, how can the sampled signal be used to recover the whole message? This question can be answered by considering the mathematical model that defines the sampled signal:

Sampled message = the sampling signal × the message

As you can see, sampling is actually the multiplication of the message with the sampling signal. And, as the sampling signal is a digital signal which is actually made up of a DC voltage and many sinewaves (the fundamental and its harmonics) the equation can be rewritten as:

Sampled message = (DC + fundamental + harmonics) × message

Block Diagram:

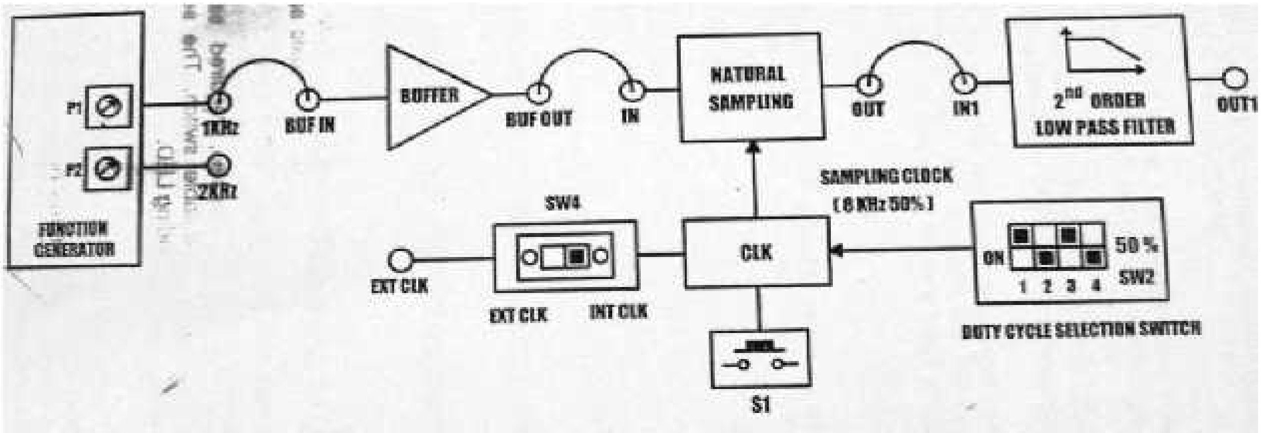


Fig. 1.1BlockDiagram for Natural Sampling

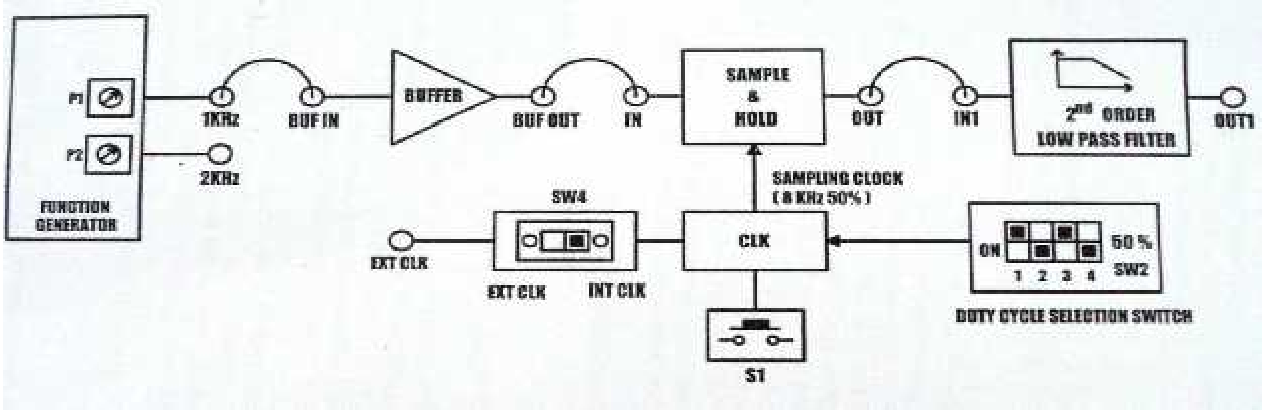


Fig. 1.2BlockDiagram for Sample and Hold

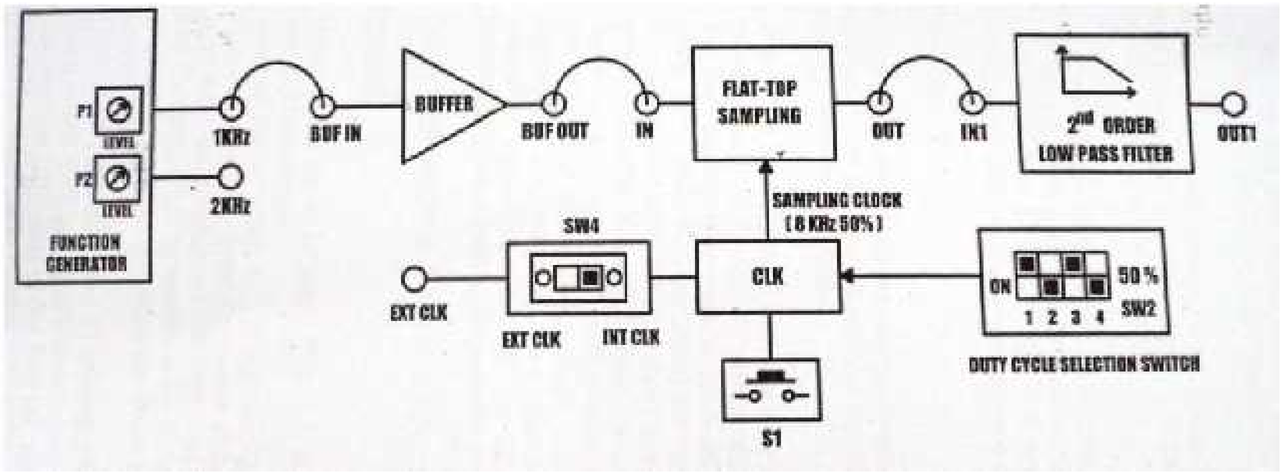
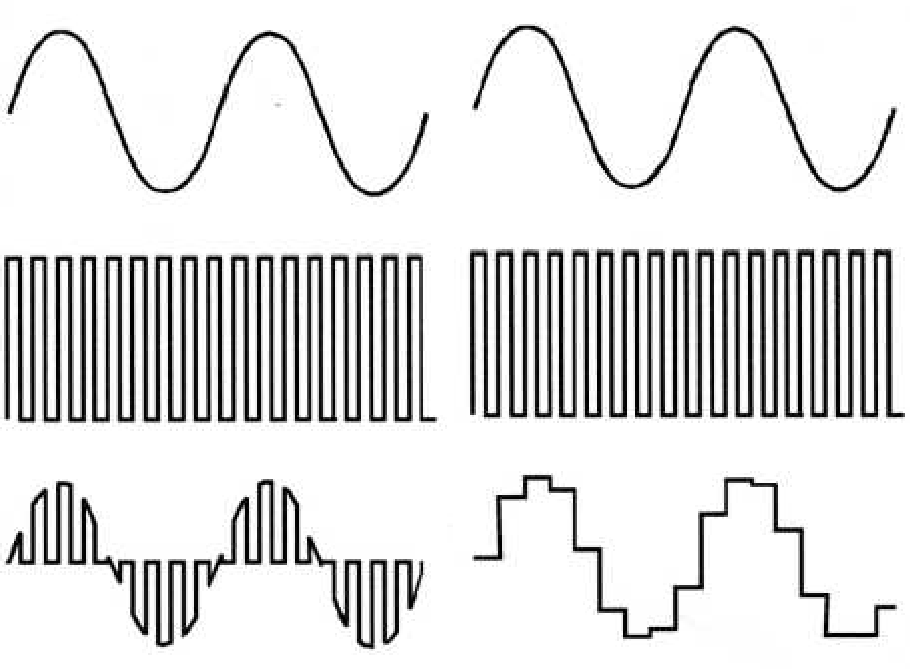


Fig. 1.3BlockDiagram for Flat Top Sampling

Waveforms



a. Sampling **b. Reconstruction**

Fig. 1. Showing sampling and reconstruction waves

Conclusion:

EXPERIMENT-2

Aim: To study Pulse Position Modulation.

ApparatusRequired: Dual trace CRO, PPM kit, connecting leads.

THEORY:

In Pulse Position Modulation, both the pulse amplitude and pulse duration are held constant but the position of the pulse is varied in proportional to the sampled values of the message signal. Pulse time modulation is a class of signalling techniques that encodes the sample values of analog signal onto the time axis of a digital signal and it is analogous to angle modulation techniques. The two main types of PTM are PWM and PPM. In PPM the analogsample value determines the position of a narrow pulse relative to the clocking time. In PPM rise time of pulse decides the channel bandwidth. It has low noise interference.

CIRCUITDIAGRAMandWAVEFORM:

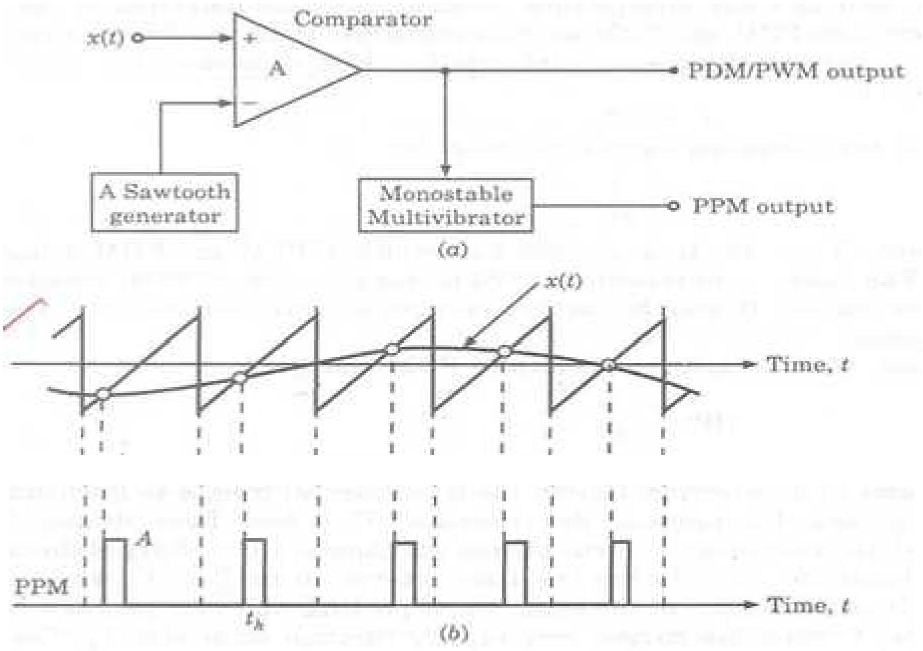


Fig. 1. PPM generation circuit and PPM waveform

Conclusion:

EXPERIMENT-3

AIM: To generate the pulse width modulated and demodulated signals**.**

Apparatus Required: Capacitors-0.01µF, 1µF, Resistors- 1.2kΩ, 1.5kΩ, 8.2kΩ, IC555, Function generator, DSO**.**

THEORY:

Pulse Time Modulation is also known as Pulse Width Modulation or Pulse Length Modulation. In PWM,the samples of the message signal are used to vary the duration of the individual pulses. Width may bevaried by varying the time of occurrence of leading edge, the trailing edge or both edges of the pulse inaccordance with modulating wave. It is also called Pulse Duration Modulation.

CIRCUIT DIAGRAM:

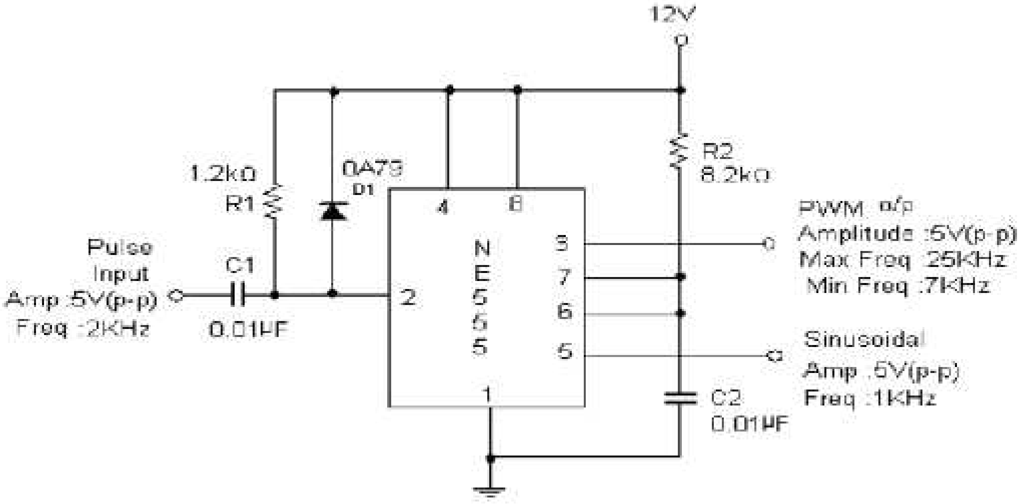


Fig. 1. Pulse width modulation generator circuit

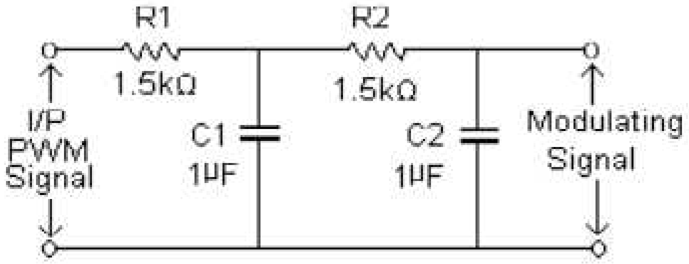


Fig. 2. Pulse width modulation demodulation circuit

Waveforms:

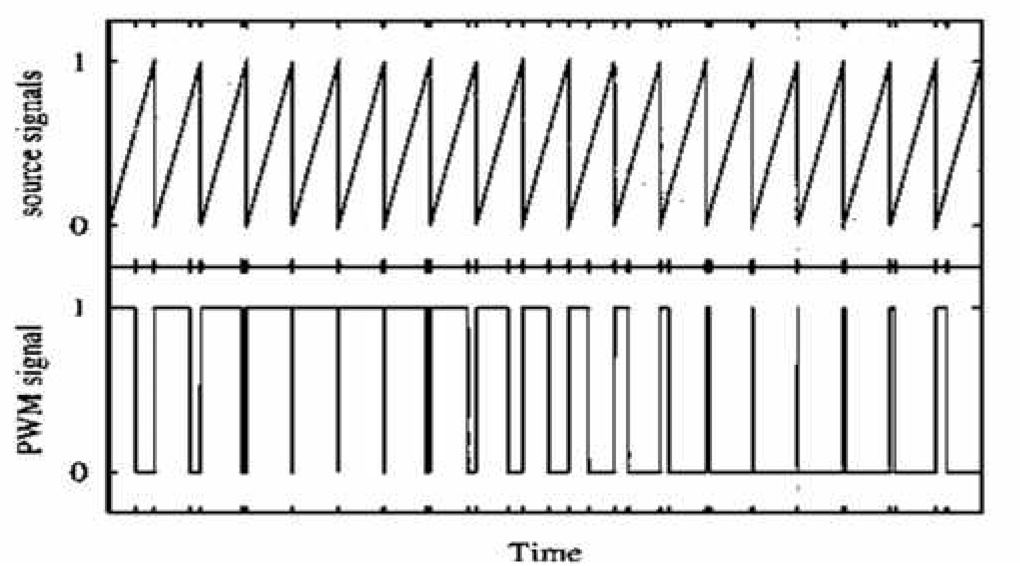


Fig. 2. Showing PWM signal

Observation Table:

l. No.	ontrol Voltage (V_{p-p})	)/P Pulse Width (m sec)

Conclusion:

EXPERIMENT-4

Aim: Study of Time Division Multiplexing System

Apparatusrequired: TDM Transmitter Trainer, Connecting wires, DSO.

Theory: Time Division is a technique of transmitting more than one information on the same channel. The samples consist of short pulses followed by another pulse after a long-time interval, this is as shown in fig.1.

This no-activity time intervals can be used to include samples from the other channels as well. This means that several information can be transmitted over a single channel by sending samples from different information sources at different moments in time. This technique is known as Time Division Multiplexing or TDM. TDM is widely used in digital communication systems to increase the efficiency of the transmitting.

Block diagram:

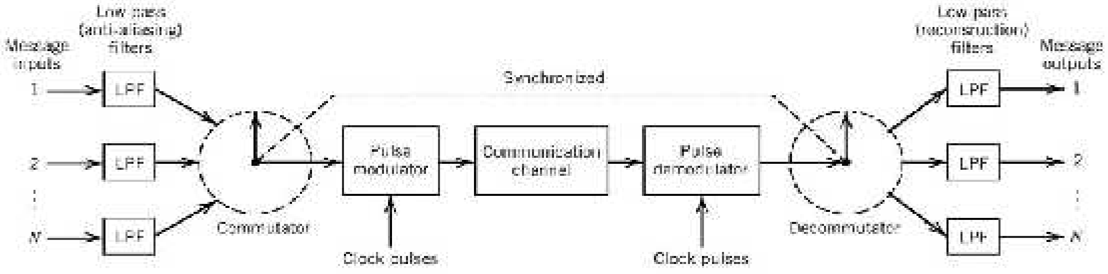


Fig. 1. Block diagram of TDM system.

Waveform:

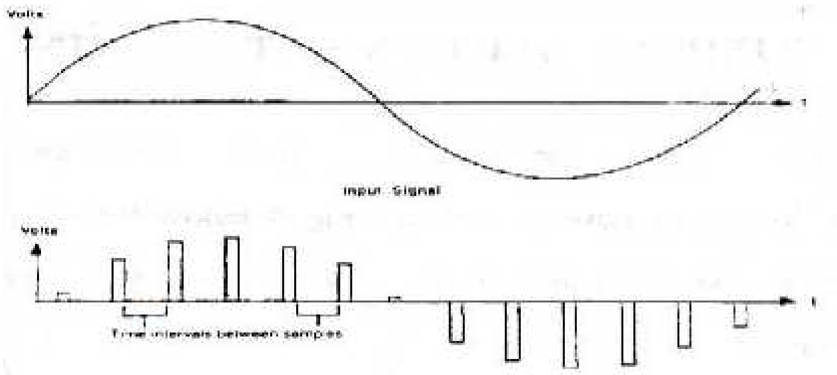


Fig. 2. Showing TDM waveform with long time intervals between the samples.

Conclusion:

EXPERIMENT-4

Aim: To study delta modulation & demodulation and observe the effect of slope overload.

Apparatusrequired: Delta modulation / demodulation trainer, Connecting wires, DSO.

Theory: Delta Modulation is a system of Digital Modulation scheme in which the difference between the sample value at sampling time K and sample value at the previous sampling time (k – 1) is encoded into just a single bit.

The baseband signal m(t) and its quantized approximation m'(t) are applied as inputs to a comparator. A comparator simply makes a comparison between inputs. If signal amplitude has increased, then modulators output is at logic level 1. If the signal amplitude has decreased, the modulator output is at logic level 0. Thus the output from the modulator is a series of 0's and 1's to indicate rise and fall of the waveform since the previous value. The comparator output is then latched into a D flip-flop which is clocked by the transmitter clock. Thus the output of the flip-flop is a latched 1 or 0 synchronous with the transmitter clock edge. The binary sequence is transmitted to receive and is also fed to the unipolar to bipolar converter. This block converts logic 0 to voltage level of +4V and 1 to voltage level of – 4V. The bipolar output ia applied to the integrator whose output is : a) Rising linear ramp signal when – 4V is applied to it b) Falling linear ramp signal when + 4V is applied to it. The integrator output is then connected to the – ve terminal of voltage comparator.

Block Diagram:

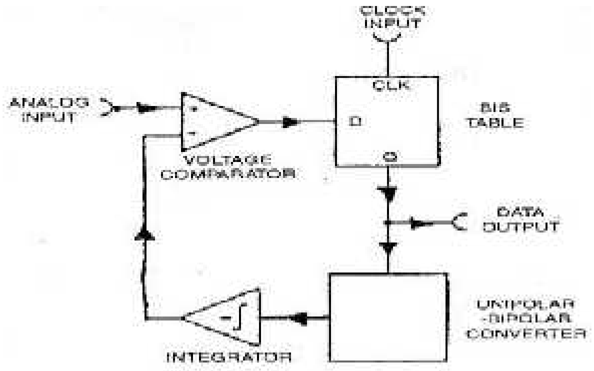


Fig.1.: Delta Modulator

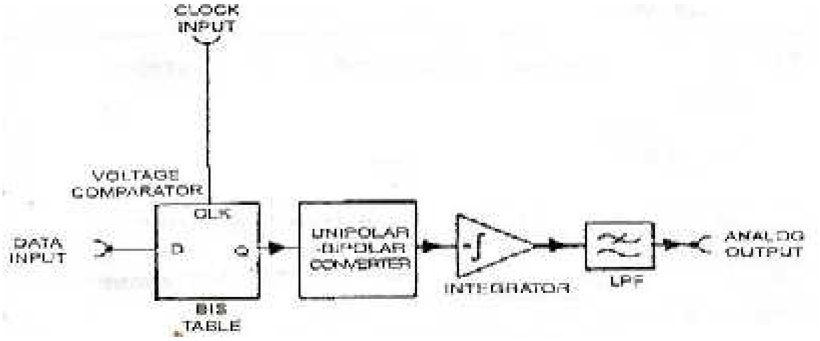


Fig.2.: Delta demodulator

Observation:

Conclusion:

EXPERIMENT-6

Aim: To study the operation of Amplitude Shift Keying modulation and demodulation with the help of circuit connections.

Apparatusrequired:

1. Resistors	1.2KΩ	3
2. Transistor	BC 107	2
3. DSO		1
4. Functiongenerator	0-1MHz	1
5. Regulated PowerSupply	0-30V,1A	1
6. Probes	---	1

Theory:

The binary ASK system was one of the earliest form of digital modulation used in wireless telegraphy. In an binary ASK system binary symbol 1 is represented by transmitting a sinusoidal carrier wave of fixed amplitude A_c and fixed frequency f_c for the bit duration T_b whereas binary symbol 0 is represented by switching of the carrier for T_b seconds. This signal can be generated simply by turning the carrier of a sinusoidal oscillator ON and OFF for the prescribed periods indicated by the modulating pulse train. For this reason, the scheme is also known as on-off shift testing. Let the sinusoidal carrier can be represented by $E_c(t) = A_c \cos(2\Pi f_c t)$ then the binary ASK signal can be represented by a wave s(t) given by $S(t) = A_c \cos(2\Pi f_c t)$, symbol 1 ASK signal can be generated by applying the incoming binary data and the sinusoidal carrier to the two inputs of a product modulator. The resulting output is the ASK wave. The ASK signal which is basically product of the binary sequence and carrier signal has a same as that of base band signal but shifted in the frequency domain by $\pm f_c$. The band width of ASK signal is infinite but practically it is $3/T_b$.

Circuit diagram:

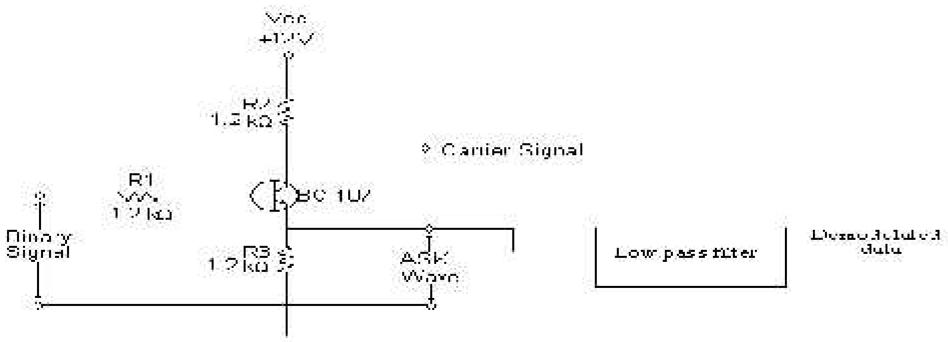


Fig. 1. Amplitude Shift Keying and demodulation Circuit

Waveforms:

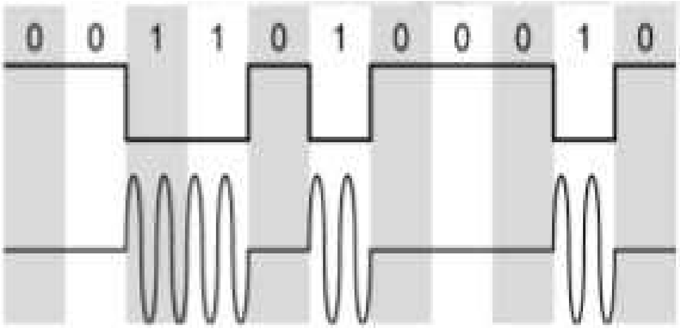


Fig. 2. Showing Amplitude Shift Keying signal

Conclusion:

EXPERIMENT-7

Aim: To study the operation of Frequency Shift Keying modulation and demodulation with the help of kit.

Apparatus Required: FSK kit, DSO and connecting probes

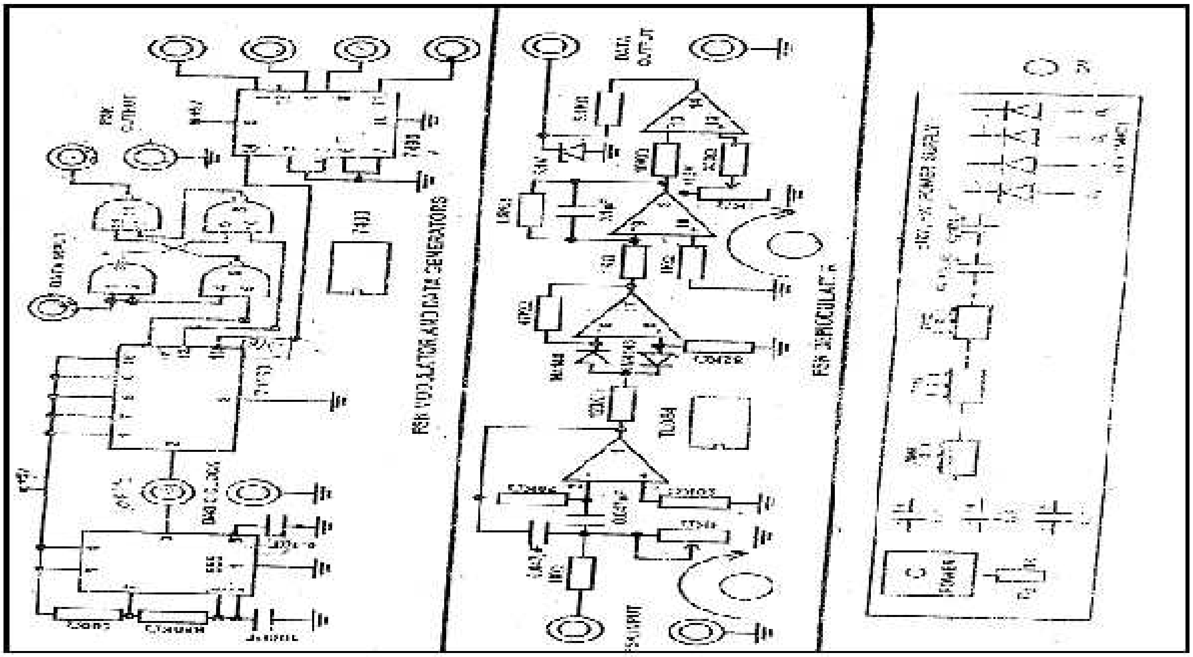
Theory:

Frequency Shift Keying is the process generating a modulated signal from a digital data input. If the incoming bit is 1, a signal with frequency f1 is sent for the duration of the bit. If the bit is 0, a signal with frequency f2 is sent for the duration of this bit. This is the basic principle behind FSK modulation.

Basically a 555 timer is used as an Astablemultivibrator, which generates a clock pulse of frequency determined by the values of R and C in this circuit. This is divided by 2, 4, 8 and 16 using 74163 IC, and two of these outputs are used in a NAND logic gates circuit, to generate a FSK modulated wave. To this NAND gates' circuit a binary data sequence is also supplied. The circuit operation causes a frequency f1 for bit 1, and f2 for bit 0.

In the demodulator circuit, the FSK modulated signal is applied to a high Q tuned filter. This filter is tuned to the frequency of either 0 or 1. This filter passes the selected frequency and rejects the other. The output is then passed through a FWR (Full Wave Rectifier) circuit and the output is now above zero volts only. It is then passed through a comparator; if the input to the comparator is greater than threshold value, the output is 1, else it is 0. This digital output of the comparator is the demodulated FSK output.

Circuit diagram:



Waveforms

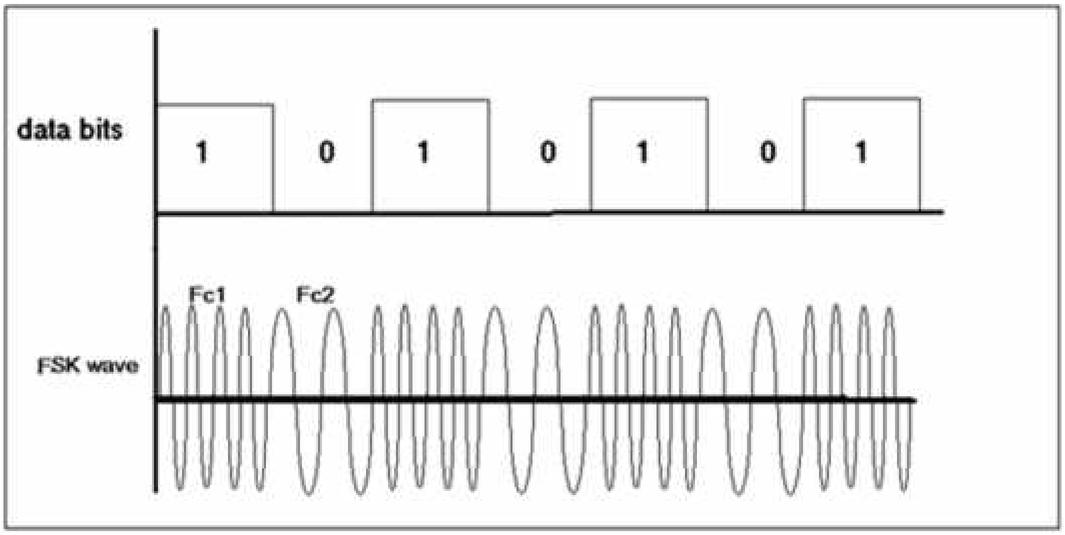


Fig. 1. Showing waveform of Frequency shift keying.

Conclusion:

EXPERIMENT No. :8

Aim: To study Phase Shift Keying (PSK).

Apparatus Used: Trainer kit, Connecting wires, DSO.

Theory: Phase shift keying (PSK) involves the phase shift change of the carrier sine wave between 0º and 180º in accordance with the data stream to be transmitted. PSK is also known as phase reversal keying (PSK).
Functionally, the PSK modulator is very similar to the ASK modulator . both uses balanced modulator to multiply the carrier with the modulating signal. Bit in contrast to ASK technique, the digital signal applied to the modulation input for PSK generation is bipolar i.e. have equal positive and negative voltage levels. When the modulating input is positive the output of modulator is a sine wave in phase with the carrier input. Whereas for the negative voltage levels, the output of modulator is a sine wave which is shifted out of phase 180º from the carrier input.

Block Diagram:

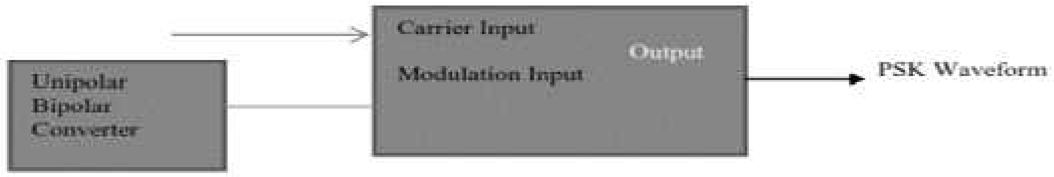


Fig. 1: Block diagram of PSK modulator

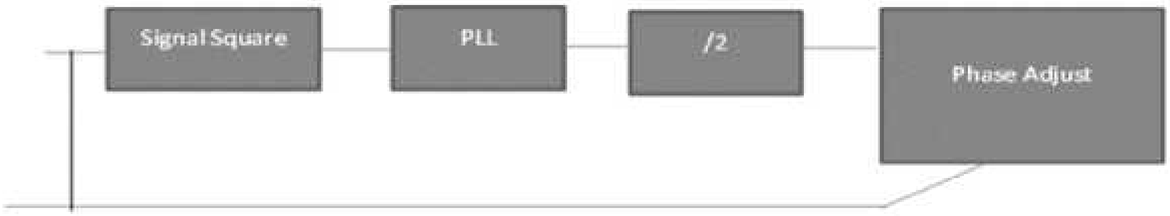


Fig. 2. Block diagram of PSK demodulator

Waveforms:

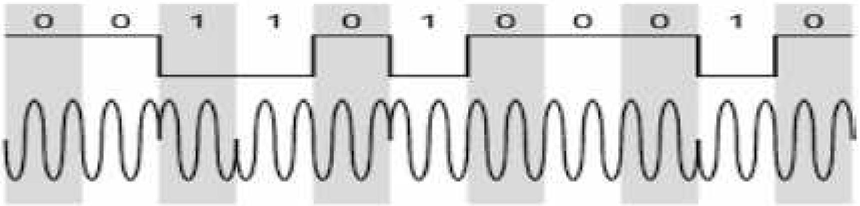


Fig. 3. PSK waveform

Conclusion:

Lab Manual

EXPERIMENT No. :9

Aim: To study QuadriPhase Shift Keying (QPSK).

Apparatus Used: Trainer kit, Connecting wires, DSO.

Theory: With quadrature phase shift keying modulation (also called quaternary PSK,Quadriphase PSK or 4-PSK), a sinusoidal waveform is varied in phase whilekeeping the amplitude and frequency constant. The term quadrature indicatesthat there are four possible phases.
Equation (1) shows the general expression for a QPSK waveform.

$$S_i(t) = A \quad [\omega_c t + \varphi_i(t)] \qquad (1)$$

Block Diagram:

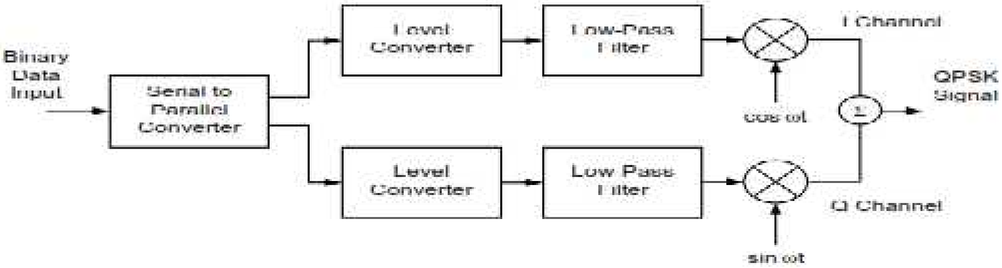
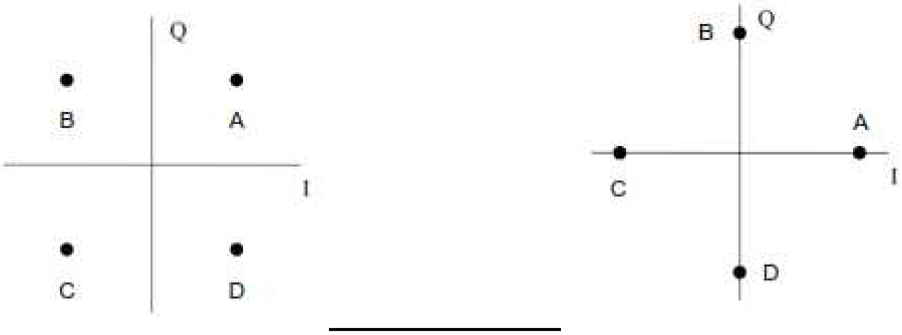


Fig. 1. Block diagram of QPSK Modulator.

Waveforms and Constellation diagram:



(a) (b)

Fig. 2. Constellation diagram (a) $\varphi = \frac{\pi}{4},\frac{3\pi}{4},\frac{5\pi}{4},\frac{7\pi}{4}$ (b) $\varphi = 0,\frac{\pi}{2},\pi,\frac{3\pi}{2}$

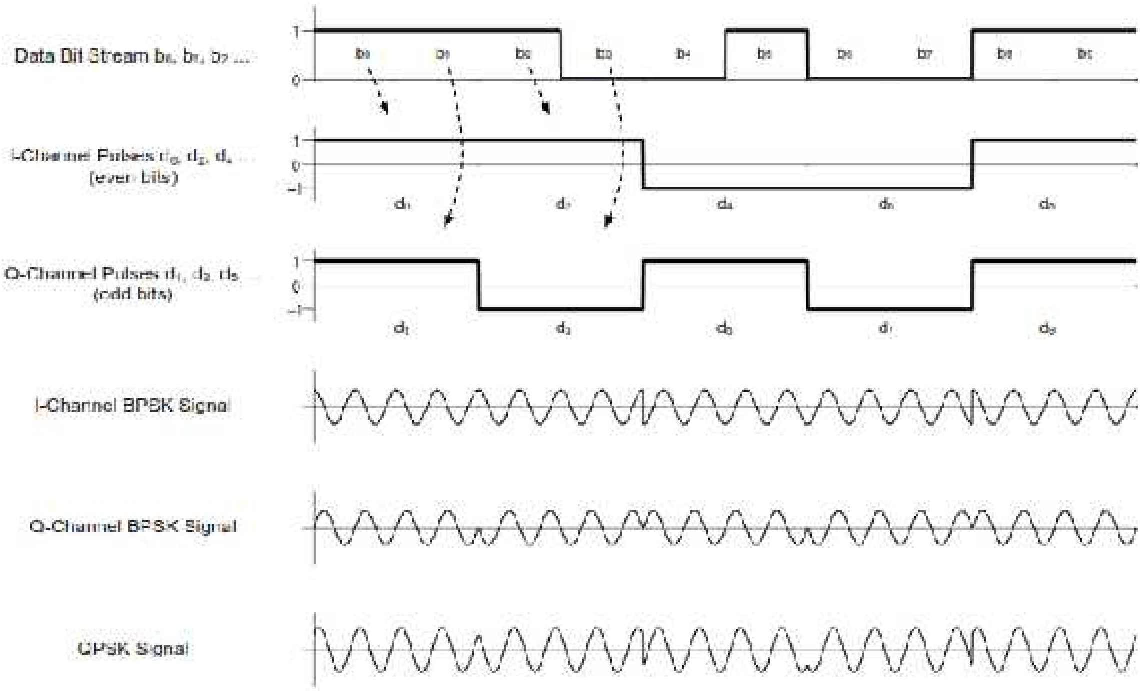


Fig. 3. Waveform of QPSK signal

Conclusion:

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR
Lab Manual

EXPERIMENT No. :10

Aim: To study Differential Phase Shift Keying (DPSK).

Apparatus Used: Trainer kit, Connecting wires, DSO.

Theory:DPSK is noncoherent form of phase shift keying which avoids the need for a coherent reference signal at the receiver.Input binary sequence is first differentially encoded and then modulation using a BPSK modulator. Differentially encoded sequence {d_k} is generated from the input binary sequence {m_k} by complementing the modulo-sum of m_k and d_{k-1}. DPSK generation sequence is illustrated in Table 1.

Table 1. Illustrating DPSQ sequence

m_k		1	0	0	1	0	0	1	1
d_{k-1}		1	1	0	1	1	0	1	1
d_k	1	1	0	1	1	0	1	1	1

Block diagram:

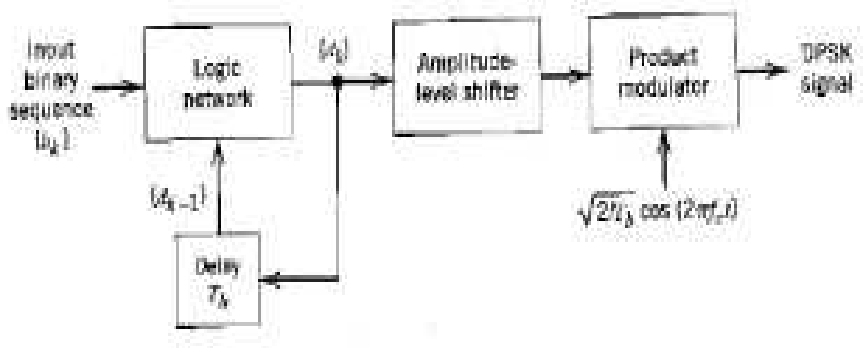


Fig. 1: Block diagram of generating DPSK signal

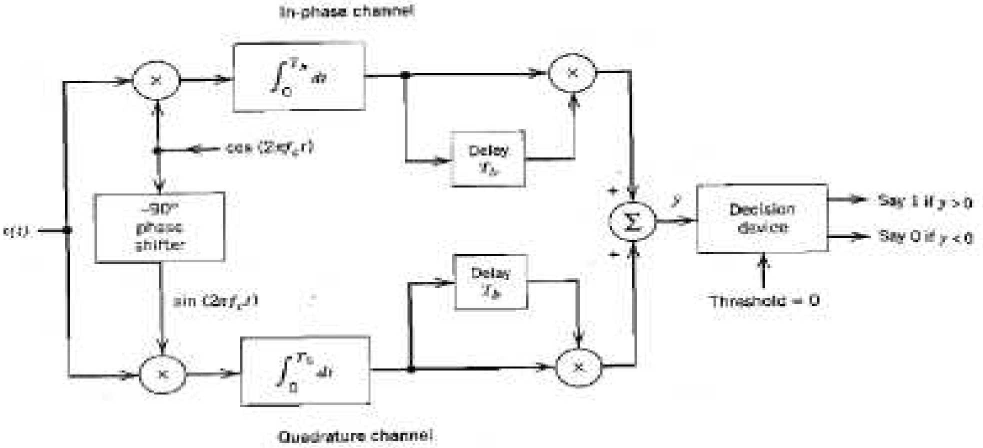


Fig. 2: Block diagram of demodulating DPSK signal

Waveform:

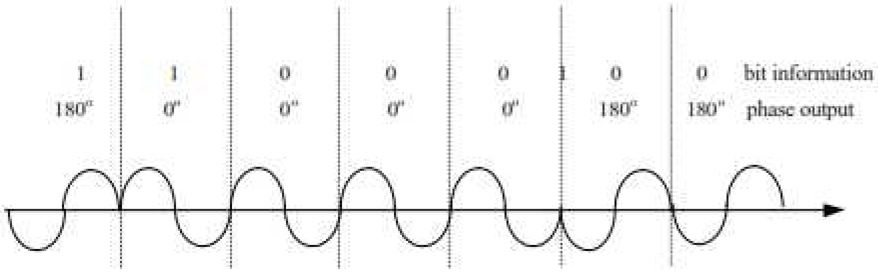


Fig. 3: Waveform showing DPSK signal.

Conclusion:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Digital Signal Processing lab
Course Code: CS594C
L-T-P scheme: 0-0-3 **Course Credit: 2**

Objectives:

The main objective of this course is to introduce the architecture of DSP processor for developing real-time applications. In this course students, will learn about the computational building blocks and the basic architectural features of DSP. They will learn about programmable digital signal processors and implementation details of DSP algorithms like digital filters, including basic adaptive filters and FfTs. They will also be introduced to CODEC programming and interfacing codec and DSP as
well as several real-world applications of DSP processors.

Learning Outcomes:

1. Understand the architecture and building blocks of digital signal processor.
2. Analyze and process signals using DSP Processor.
3. Implementing FIR, IIR and basic adaptive filters to suit specific requirements for specific applications.
4. Learn codec programming and interfacing it with DSP.
5. Understand the applications of DSP processors
6. Designing and implementing a small application using DSP processor

Course Contents:

Exercises that must be done in this course are listed below:

Experiment 1: - Generate continuous and Discrete signal
Experiment 2: - Graphical representation of unit step signal
Experiment 3: - Graphical representation of unit sample signal
Experiment 4: - Graphical representation of unit ramp signal
Experiment 5: - Graphical representation of exponential signal
Experiment 6: - Graphical representation of exponential increasing- decreasing signal
Experiment 7: - Graphical representation of even signal
Experiment 8: - Graphical representation of odd signal
Experiment 9:- Determine whether given signal is periodic or not
Experiment 10: - Convolution of given sequences
Experiment 11: - Cross correlation of given sequences
Experiment 12: - Plot Magnitude and Phase Response
Experiment 13: - Impulse Response of a given System
Experiment 14: -Z Transform of the Sequence a given sequence
Experiment 15: - Inverse Z Transform of the Sequence a given sequence
Experiment 16: - DFT and IDFT of a Sequence
Experiment 17: - 8- point DFT of the Sequence
Experiment 18: - Circular convolution of following sequences

Text Book:

1. J. G. Proakis and D. G. Manolakis, “Digital Signal Processing”, Prentice Hall India, 3rd edition, 1997, ISBN: 81-203-1129-9

Recommended Systems/Software Requirements:

1. SCILAB

Experiment 1: -Generate continuous and Discrete signal

AIM: - To write a scilab code to sketch the continuous time signal x (t) =2∗exp(-2 t) and also its discrete time equivalent signal with a sampling period T = 0 . 2 sec

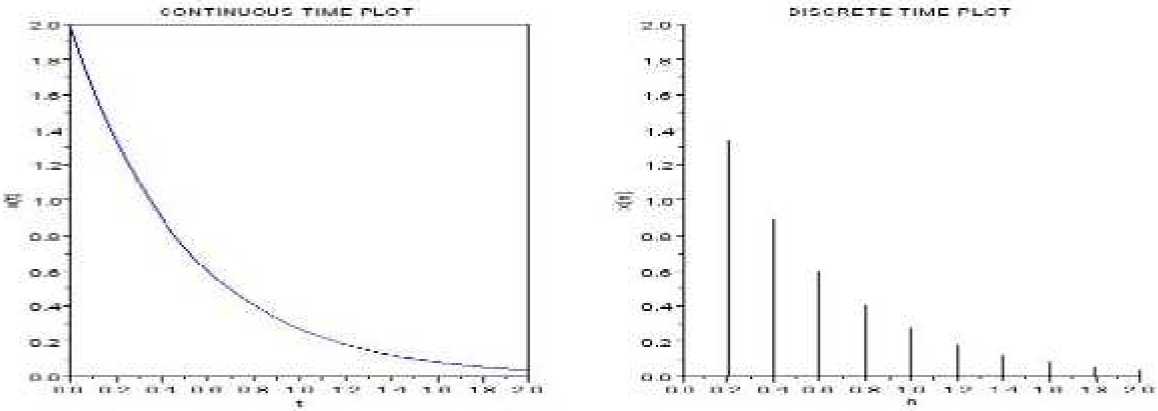
Algorithm: -

Step 1: - Start the program
Step 2: - Get the input for signal generation
Step 3: - Use the appropriate library function
Step 4: - Display the output and output wave form

Source Code: -

```
clear;
clc ;
 close ;
 t =0:0.01:2;
 x1 =2* exp ( -2*t);
 subplot (1 ,2 ,1);
 plot (t,x1);
xlabel ( ’ t ’ );
ylabel ( ’ x ( t ) ’ );
 title ( ’CONTINUOUS TIME PLOT ’ );
 n =0:0.2:2;
 x2 =2* exp ( -2*n);
subplot (1 ,2 ,2);
plot2d3 (n, x2);
xlabel ( ’n ’ );
ylabel ( ’ x (n) ’ );
title ( ’DISCRETE TIME PLOT ’ );
```

Output: -



Experiment 2: - Graphical representation of unit step signal

AIM: - To write the Scilab code to find the unit step signal and sketch the output wave form.

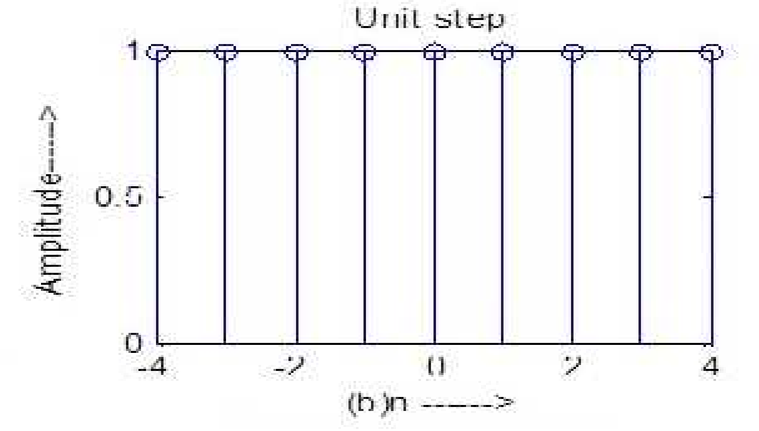
Algorithm: -

Step 1: - Start the program
Step 2: - Get the input for signal generation
Step 3: - Use the appropriate library function
Step 4: - Display the output and output wave form

Source Code: -

```
clear;
clc;
close;
L = 4; // Upper limit
n = -L: L;
x = [ zeros (1, L), ones (1, L +1)];
a= gca ();
a. thickness = 2;
a. y_location = "middle ";
plot2d3 (‘gnn‘, n,x)
xtitle(‘Graphical Representation of Unit Step Signal', ' n ', ' x [ n] ' );
```

Output: -



Experiment 3: - Graphical representation of unit sample signal

AIM: - To write the Scilab code to find the unit sample signal and sketch the output wave form.

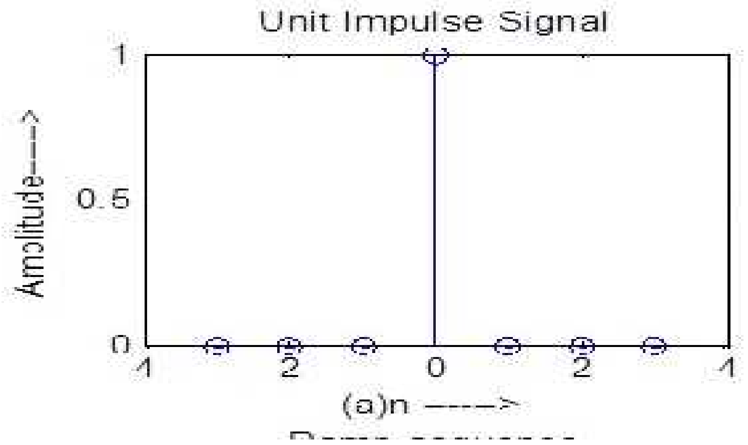
Algorithm: -

Step 1: - Start the program
Step 2: - Get the input for signal generation
Step 3: - Use the appropriate library function
Step 4: - Display the output and output wave form

Source Code: -

```
clear;
clc;
close;
L = 4; // Upper limit
n = -L: L;
x = [ zeros (1,L) ,1, zeros (1,L)];
a= gca ();
a.thickness = 2;
a.y_location = " middle ";
plot2d3 ( ’ gnn ’ ,n,x)
xtitle ( ’ Graphical Representation of Unit Sample Sequence ’ , ’ n ’ , ’ x [ n ] ’ );
```

Output: -



Experiment 4: - Graphical representation of unit ramp signal

AIM: - To write the Scilab code to find the unit ramp signal and sketch the output wave form.

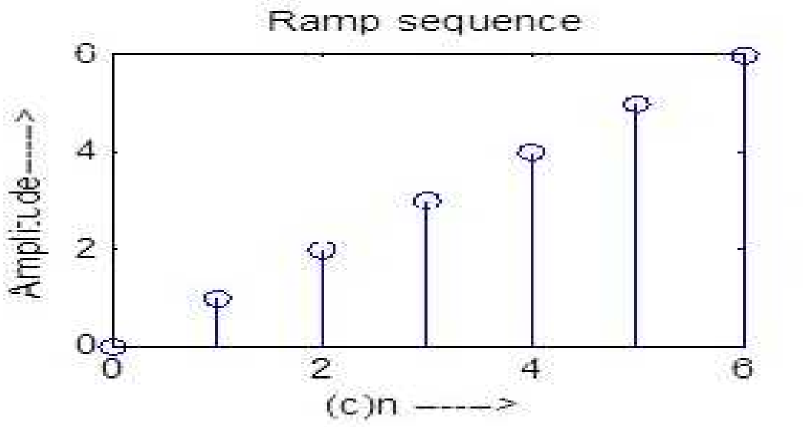
Algorithm: -

Step 1: - Start the program
Step 2: - Get the input for signal generation
Step 3: - Use the appropriate library function
Step 4: - Display the output and output wave form

Source Code: -

```
clear;
clc;
close;
L = 4; // Upper limit
n = -L: L;
x = [ zeros (1, L) ,0: L];
a= gca ();
a.thickness = 2;
a.y_location = "middle ";
plot2d3 (‘gnn‘, n,x)
xtitle('Graphical representation of unit ramp signal, ' n ' , ' x [ n ] ' );
```

Output: -



Experiment 5: - Graphical representation of exponential signal

AIM: - To write the Scilab code to find the exponential signal and sketch the output wave form.

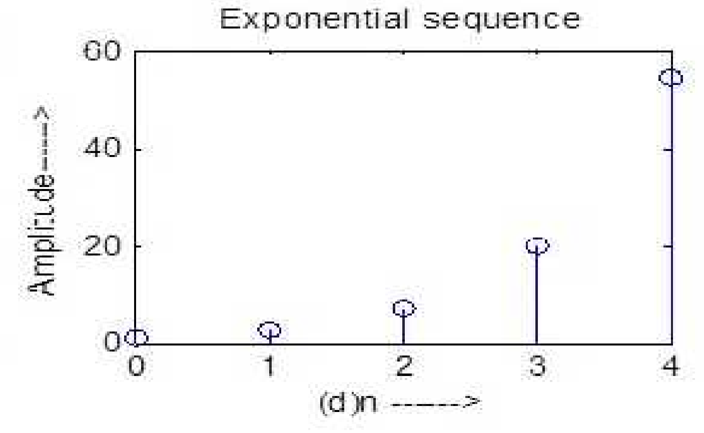
Algorithm: -

Step 1: - Start the program
Step 2: - Get the input for signal generation
Step 3: - Use the appropriate library function
Step 4: - Display the output and output wave form

Source Code: -

```
clear;
clc ;
close ;
a =1.5;
n =1:10;
x = (a)^n;
a= gca ();
a.thickness = 2;
plot2d3 ( ' gnn ' ,n,x)
xtitle ( ' Graphical representation of exponential signal', ' n ' , ' x [ n ] ' );
```

Output: -



Experiment 6: - Graphical representation of exponential increasing- decreasing signal

AIM: - To write the Scilab code to find the exponential increasing- decreasing signal and sketch the output wave form.

Algorithm: -

Step 1: - Start the program
Step 2: - Get the input for signal generation
Step 3: - Use the appropriate library function
Step 4: - Display the output and output wave form

Source Code: -

```
clear;
clc;
close;
a = -1.5;
n = 0:10;
x = (a)^n;
a= gca ();
a. thickness = 2;
a. x_location = " o r i g i n ";
a. y_location = " o r i g i n ";
plot2d3 ( ' gnn ' ,n,x)
xtitle ( ' Graphical representation of exponential increasing- decreasing signal', ' n ' , ' x [ n ] ' );
```

Experiment 7: - Graphical representation of even signal

AIM: - To write the Scilab code to find the even signal and sketch the output wave form.

Algorithm: -

Step 1: - Start the program
Step 2: - Get the input for signal generation
Step 3: - Use the appropriate library function
Step 4: - Display the output and output wave form

Source Code: -

```
clear;
clc;
close;
n = -7:7;
x1 = [0 0 0 1 2 3 4];
x = [x1 ,5, x1(length (x1) : -1:1) ];
```

```
a= gca ();
a. thickness = 2;
a. y_location = " middle ";
plot2d3 ( ’ gnn ’ ,n,x)
xtitle ( ’ Graphical representation of even signal’, ’ n ’, ’ x [ n ] ’ );
```

Experiment 8: - Graphical representation of odd signal

AIM: - To write the Scilab code to find the odd signal and sketch the output wave form.

Algorithm: -

Step 1: - Start the program
Step 2: - Get the input for signal generation
Step 3: - Use the appropriate library function
Step 4: - Display the output and output wave form

Source Code: -

```
clear ;
clc ;
close ;
n = -5:5;
x1 = [0 1 2 3 4 5];
x = [-x1($ : -1:2) ,x1 ];
a= gca ();
a. thickness = 2;
a. y_location = " middle ";
a. x_location = " middle "
plot2d3 ( ’ gnn ’ ,n,x)
xtitle ( ’ Graphical representation of even signal’, ’ n ’, ’ x [ n ] ’ );
```

Experiment 9:-Determine whether given signal is periodic or not

AIM: - To write the Scilab code to find the given signal is periodic or not

Algorithm: -

Step 1: - Start the program
Step 2: - Get the input for signal generation
Step 3: - Use the appropriate library function
Step 4: - Display the output and output wave form

Source Code: -

```
clear ;
clc ;
close ;
t =0:0.01:10;
x1=cos (2* %pi *t /3) ;
subplot (1 ,2 ,1);
plot (t,x1);
xlabel ( ’ t ’ );
ylabel ( ’ x ( t ) ’ );
title ( ’CONTINUOUS TIME PLOT ’ );
n =0:0.2:10;
x2= cos (2* %pi *n /3) ;
subplot (1 ,2 ,2);
plot2d3 (n,x2);
xlabel ( ’ n ’ );
ylabel ( ’ x ( n ) ’ );
title ( ’DISCRETE TIME PLOT ’ );
```

Experiment 10: -Convolution of given sequences

AIM: - To write the Scilab code to find the convolution of a given signal x (n) = [1 2 1 1] , h (n) = [1 -1 1 -1]and sketch the output wave form.

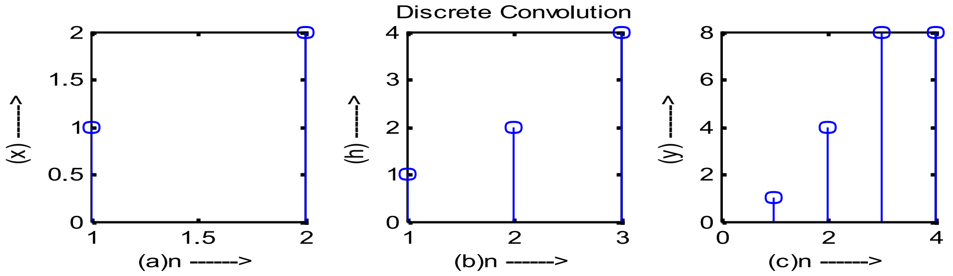
Algorithm: -

Step 1: - Start the program
Step 2: - Get the input for signal generation
Step 3: - Use the appropriate library function
Step 4: - Display the output and output wave form

Source Code: -

```
clear;
clc;
close;
x = [1 2 1 1];
h =[1 -1 1 -1];
y= convol (x,h);
disp ( round (y))
```

Output: -



Experiment 11: - Cross correlation of given sequences

AIM: - To write the Scilab code to find the cross correlation of a given signal x (n) = [1 2 1 1] , h (n) = [1 1 2 1]and sketch the output wave form.

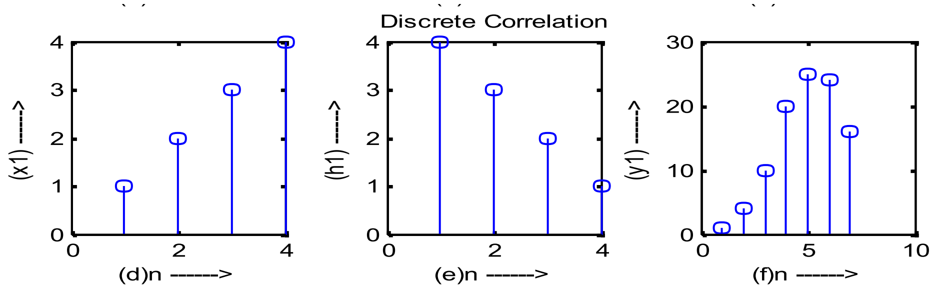
Algorithm: -

Step 1: - Start the program
Step 2: - Get the input for signal generation
Step 3: - Use the appropriate library function
Step 4: - Display the output and output wave form

Source Code: -

```
clear;
clc;
close;
x = [1 2 1 1];
h = [1 1 2 1];
h1 = [1 2 1 1];
y= convol (x, h1);
11 disp (round (y));
```

Output:-



Experiment 12: - Plot Magnitude and Phase Response

AIM: - To write the Scilab code to find the magnitude and phase plot of a given system and sketch the output wave form.

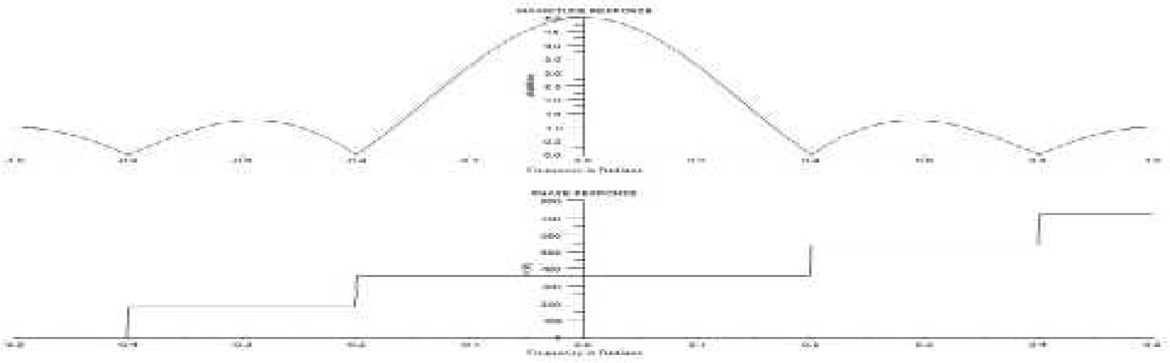
Algorithm: -

Step 1: - Start the program
Step 2: - Get the input for signal generation
Step 3: - Use the appropriate library function
Step 4: - Display the output and output wave form

Source Code: -

```
clear ;
clc ;
close ;
w=- %pi :0.01: %pi ;
H =1+2* cos(w) +2* cos (2* w);
[ phase_H ,m]= phasemag (H);
Hm=abs(H);
a= gca ();
subplot (2 ,1 ,1);
a. y_location =" origin ";
plot2d (w/%pi ,Hm);
xlabel ( ’ Frequency in Radians ’ )
ylabel ( ’ abs (Hm) ’ );
title ( ’MAGNITUDE RESPONSE ’ );
subplot (2 ,1 ,2);
a= gca ();
a. x_location =" origin ";
a. y_location =" origin ";
plot2d (w /(2* %pi ),phase_H );
xlabel ( ’ Frequency in Radians ’ );
ylabel ( ’ <(H) ’ );
title ( ’PHASE RESPONSE ’ );
```

Output: -



Experiment 13: - Impulse Response of a given System

AIM: - To write the Scilab code to find the impulse response of a given system and sketch the output wave form.

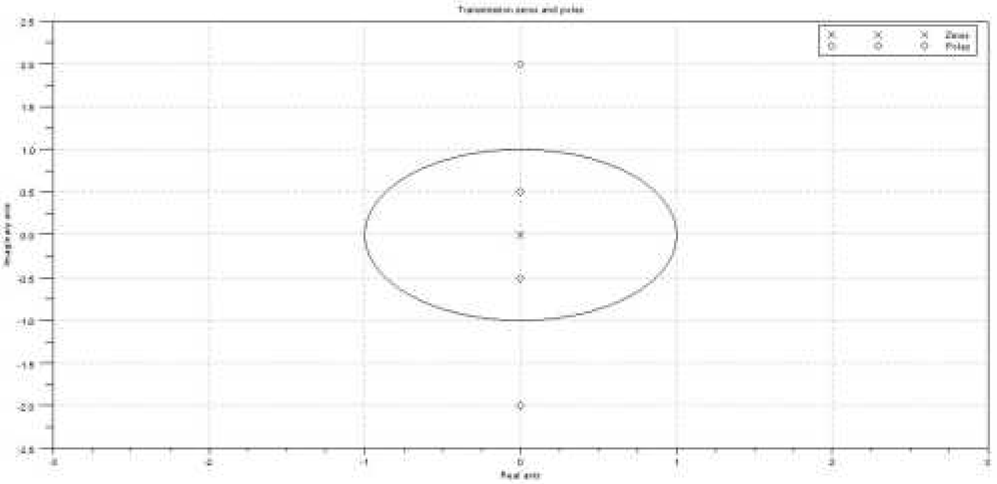
Algorithm: -

Step 1: - Start the program
Step 2: - Get the input for signal generation
Step 3: - Use the appropriate library function
Step 4: - Display the output and output wave form

Source Code: -

```
clear;
clc;
close ;
z=%z;
a=z ^3+2*( z ^(2) ) -4*(z) +1;
b=z ^3;
h = ldiv (a,b ,4) ;
disp (h,"h(n)=");
```

Output:-



Experiment 14: -Z Transform of the Sequence a given sequence

AIM: - To write the Scilab code to calculate the z transform of a given sequence and sketch the output wave form.

Algorithm: -

Step 1: - Start the program
Step 2: - Get the input for signal generation
Step 3: - Use the appropriate library function
Step 4: - Display the output and output wave form

Source Code: -

```
clear ;
clc ;
close ;
function [za ]= ztransfer ( sequence ,n)
z= poly (0, ’ z ’ , ’ r ’ )
za= sequence *(1/ z)^n’
endfunction
x1 =[2 -1 3 2 1 0 2 3 -1];
n = -4:4;
zz= ztransfer (x1 ,n);
disp (zz ,"Z-transform of sequence is : ");
disp ( ’ROC is the entire plane except z = 0 and z =%inf ’ );
```

Experiment 15: - Inverse Z Transform of the Sequence a given sequence

AIM: - To write the Scilab code to calculate the inverse z transform of a given sequence and sketch the output wave form.

Algorithm: -

Step 1: - Start the program
Step 2: - Get the input for signal generation
Step 3: - Use the appropriate library function
Step 4: - Display the output and output wave form

Source Code: -

```
clear;
clc ;
close ;
z=%z;
a =(2+2* z+z ^2) ;
b=z ^2;
h = ldiv (b,a ,6) ;
disp (h," First six values of h ( n )=");
```

Experiment 16: -DFT and IDFT of a Sequence

AIM: - To write the Scilab code to calculate the DFT of a given sequence x [n] = [1, 1 , 0 , 0] and IDFT of a Sequence Y[k] = [1 , 0 , 1 , 0] and sketch the output wave form.

Algorithm: -

Step 1: - Start the program
Step 2: - Get the input for signal generation
Step 3: - Use the appropriate library function
Step 4: - Display the output and output wave form

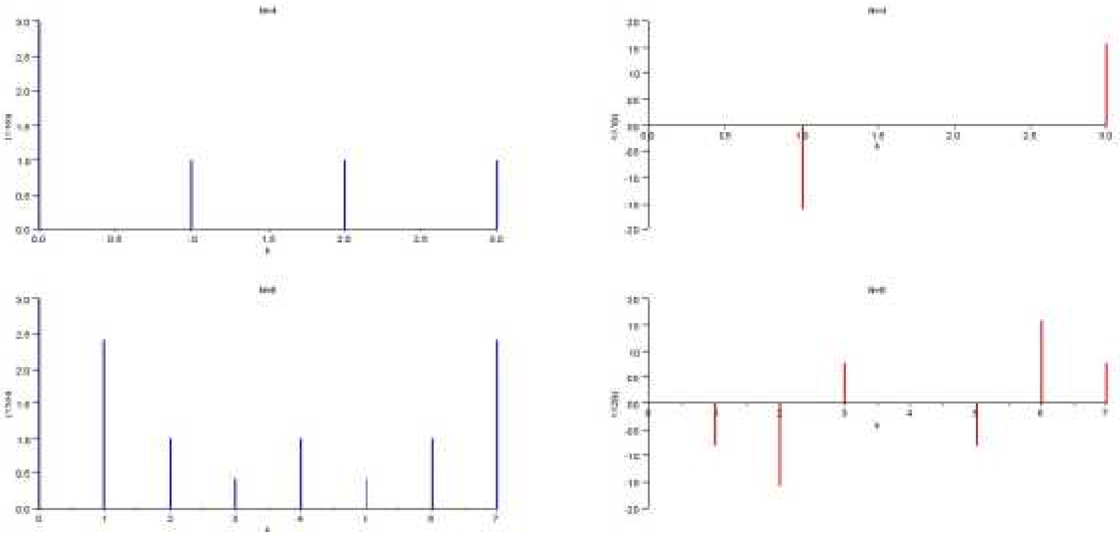
Source Code: -

x [n] = [1, 1 , 0 , 0] and IDFT of a Sequence Y[k] = [1 , 0 , 1 , 0]

```
clear ;
clc ;
close ;
```

```
x = [1 ,1 ,0 ,0];
//DFT Computation
X = fft (x , -1);
Y = [1 ,0 ,1 ,0];
//IDFT Computation
y = fft (Y , 1);
// Display sequence X[ k ] and y [ n ] in command window
disp (X,"X[ k]=");
disp (y,"y [ n]=");
```

Output: -



Experiment 17: - 8- point DFT of the Sequence

AIM: - To write the Scilab code to calculate the 8- point DFT of a given sequence x [n] = [1, 1 , 1 , 1 , 1 , 1 , 0 , 0]and sketch the output wave form.

Algorithm: -

Step 1: - Start the program
Step 2: - Get the input for signal generation
Step 3: - Use the appropriate library function
Step 4: - Display the output and output wave form

Source Code: -

```
clear ;
clc ;
close ;
x = [1 ,1 ,1 ,1 ,1 ,1 ,0 ,0];
//DFT Computation
X = fft (x , -1);
// Displaysequence X[ k ] in command window
disp (X,"X[ k]= ");
```

Experiment 18: -Circular convolution of following sequences

AIM: - To write the Scilab code to calculate the circular convolution of a given sequence x [n] = [1 , 2 , 2 , 1 , 0] and Y[k]=exp (- j *4*pi *k /5) . X [k] and sketch the output wave form.

Algorithm: -

Step 1: - Start the program
Step 2: - Get the input for signal generation
Step 3: - Use the appropriate library function
Step 4: - Display the output and output wave form

Source Code: -

```
clear ;
clc ;
close ;
x=[1 ,2 ,2 ,1 ,0];
X= fft(x , -1);
k =0:1:4;
j= sqrt ( -1);
pi =22/7;
H= exp(-j *4* pi*k /5) ;
Y=H.*X;
//IDFT Computation
y= fft(Y ,1) ;
// Displaysequence y [ n ] in command window
disp ( round (y),"y [ n]= ");
// Plots
n =0:1:4;
a = gca ();
```

```
a. y_location = "origin";
a. x_location = "origin";
plot2d3 (n, round (y) ,5);
poly1 =a. children (1) . children (1) ;
poly1 . thickness =2;
xtitle ( 'Plotofsequencey[n]' , 'n' , 'y[n]' );
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Object Oriented Programming Using Java
Course Code: CS594D
L-T-P scheme: 0-0-3 **Course Credit: 2**

Objectives:

1. To strengthen their problem solving ability by applying the characteristics of an object oriented approach.
2. To introduce object oriented concepts in Java.

Learning Outcomes:

1. Explain what constitutes an object-oriented approach to programming and identify the potential benefits of object-oriented programming over other approaches.
2. Apply an object-oriented approach to developing applications of varying complexities.

Course Contents:
Exercises that must be done in this course are listed below:
Exercise No. 1: Class creation with main method and steps of source code compilation and execution.
Exercise No. 2: Design a stack and a queue.
Exercise No. 3: Design different types of linked lists for different operations.
Exercise No. 4: Methods of String.
Exercise No. 5: Implement different types of polymorphism: overloading and overriding.
Exercise No. 6: Implement different types of inheritance.
Exercise No. 7: Use of package with access specifier.
Exercise No. 8: Write a program using static keyword.
Exercise No. 9: Write a program to use this, this(), super, super().
Exercise No. 10: Exception handling.
Exercise No. 11: Threading.
Exercise No. 12: Applet programming and Action Event.
Exercise No. 13: Swing programming and Layout.

Text Book:

1. E. Balagurusamy – " Programming With Java: A Primer" – 3rd Ed. – , Tata Mc Graw Hill.
2. Herbert Schildt, Java: The Complete Reference (Tata Mcgraw Hill Education Private , 7th Ed).

Recommended Systems/Software Requirements:

Java Development Kit and Java Runtime Environment, preferable latest version.

Experiment No: 1: Class creation with main method and steps of source code compilation and execution.

Description:
Create a file with .java extension. Write a class with main method. Then compile that source code using javac command(java compiler) then execute generated bytecode using java command.

```
/* First java program First.java */
public class First {
        public static void main(String args[]){
                System.out.println(“Welcome”);
        }
}
```

Steps of execution:

```
javac First.java //For compilation
```

OUTPUT:

Welcome

Experiment No: 2A: Design a stack.

Aim: Write a program in java to design stack.

Description:

A stack is a container of objects that are inserted and removed according to the last-in first-out (LIFO) principle. In the pushdown stacks only two operations are allowed: push the item into the stack, and pop the item out of the stack. A stack is a limited access data structure - elements can be added and removed from the stack only at the top. push adds an item to the top of the stack, pop removes the item from the top. A helpful analogy is to think of a stack of books; you can remove only the top book, also you can add a new book on the top.

/* Java program Stack.java */

```
class Stack {

   //stack holds 10 values

   int stack[] = new int[10];
   //top of the stack
   int tos;

   Stack() {
      //initially top of stack -1, denotes empty
      tos = -1;
   }

   //push or add an element top of the stack
   void push(int element) {
      //if stack is full
      if (tos == 9) {
         System.out.println("Stack is full!");
      } else {
         stack[++tos] = element;
      }
   }

   //pop or remove element from top of stack
   int pop() {
      //if no element in stack
      if (tos < 0) {
         System.out.println("Stack is empty");
         //0 indicates operation fails
         return 0;
      } else {
         return stack[tos--];
      }
   }
}
```

INPUT:

If push(1) is invoked, 1 will be added onto the stack. Again if push(2) is invoked then 2 will be added onto the stack and now top of the stack is 2.

OUTPUT:

If return value of pop() is printed then 2 will be printed as top of the element is 2.

Experiment No: 2B: Design a queue.

Aim: Write a program in java to design queue.

Description:

A queue is a container of objects (a linear collection) that are inserted and removed according to the first-in first-out (FIFO) principle. An excellent example of a queue is a line of students in the food court of the UC. New additions to a line made to the back of the queue, while removal (or serving) happens in the front. In the queue only two operations are allowed enqueue and dequeue. Enqueue means to insert an item into the back of the queue, dequeue means removing the front item. The picture demonstrates the FIFO access.

/* Java program Queue.java */

```
public class Queue {

    private static final int capacity = 3;
    int arr[] = new int[capacity];
    int size = 0;
    int top = -1;
    int rear = 0;

    public void push(int pushedElement) {
        if (top < capacity - 1) {
            top++;
            arr[top] = pushedElement;
        } else {
            System.out.println("Overflow !");
        }
    }

    public int pop() {
        int e = arr[rear];
        if (top >= rear) {
            rear++;
        } else {
            System.out.println("Underflow !");
        }
        return e;
    }

    public void display() {
        if (top >= rear) {
            System.out.println("Elements in Queue : ");
            for (int i = rear; i <= top; i++) {
                System.out.println(arr[i]);
            }
        }
    }
}
```

INPUT:

The push() method is for inserting element in a queue, pop() to get element from the rear

OUTPUT:

If display() is invoked, all the elements will be printed from the rear position to top position.

Experiment No: 3A: Design Singly Link List.

Aim: Write a program in java to design singly link list.

Description:

Singly Linked Lists are a type of data structure. It is a type of list. In a singly linked list each node in the list stores the contents of the node and a pointer or reference to the next node in the list. It does not store any pointer or reference to the previous node. It is called a singly linked list because each node only has a single link to another node. To store a single linked list, you only need to store a reference or pointer to the first node in that list. The last node has a pointer to nothingness to indicate that it is the last node.

/* Java program SinglyLinkList.java */

```
public class SinglyLinkedList<T> {

    private Node<T> head;
    private Node<T> tail;

    public void add(T element) {

        Node<T> nd = new Node<T>();
        nd.setValue(element);
        System.out.println("Adding: " + element);
        /**
         * check if the list is empty
         */
        if (head == null) {
            //since there is only one element, both head and
            //tail points to the same object.
            head = nd;
            tail = nd;
        } else {
            //set current tail next link to new node
            tail.setNextRef(nd);
            //set tail as newly created node
            tail = nd;
        }
    }

    public void addAfter(T element, T after) {

        Node<T> tmp = head;
        Node<T> refNode = null;
        System.out.println("Traversing to all nodes..");
        /**
         * Traverse till given element
         */
        while (true) {
            if (tmp == null) {
                break;
            }
            if (tmp.compareTo(after) == 0) {
```

```
            break;
        }
        tmp = tmp.getNextRef();
    }
    if (refNode != null) {
        //add element after the target node
        Node<T> nd = new Node<T>();
        nd.setValue(element);
        nd.setNextRef(tmp.getNextRef());
        if (tmp == tail) {
            tail = nd;
        }
        tmp.setNextRef(nd);

    } else {
        System.out.println("Unable to find the given element...");
    }
}

public void deleteFront() {

    if (head == null) {
        System.out.println("Underflow...");
    }
    Node<T> tmp = head;
    head = tmp.getNextRef();
    if (head == null) {
        tail = null;
    }
    System.out.println("Deleted: " + tmp.getValue());
}

public void deleteAfter(T after) {

    Node<T> tmp = head;
    Node<T> refNode = null;
    System.out.println("Traversing to all nodes..");
    /**
     * Traverse till given element
     */
    while (true) {
        if (tmp == null) {
            break;
        }
        if (tmp.compareTo(after) == 0) {
            //found the target node, add after this node
            refNode = tmp;
            break;
        }
        tmp = tmp.getNextRef();
    }
    if (refNode != null) {
        tmp = refNode.getNextRef();
        refNode.setNextRef(tmp.getNextRef());
        if (refNode.getNextRef() == null) {
            tail = refNode;
            System.out.println("Deleted: " + tmp.getValue());
        } else {
            System.out.println("Unable to find the given element...");
        }
    }

    public void traverse() {

        Node<T> tmp = head;
        while (true) {
            if (tmp == null) {
                break;
            }
            System.out.println(tmp.getValue());
            tmp = tmp.getNextRef();
        }
    }

    public static void main(String a[]) {
        SinglyLinkedList<Integer> sl = new SinglyLinkedList<Integer>();
        sl.add(3);
        sl.add(32);
        sl.add(54);
        sl.add(89);
        sl.addAfter(76, 54);
        sl.deleteFront();
        sl.deleteAfter(76);
        sl.traverse();

    }
}

class Node<T> implements Comparable<T> {

    private T value;
    private Node<T> nextRef;

    public T getValue() {
        return value;
    }

    public void setValue(T value) {
        this.value = value;
    }

    public Node<T> getNextRef() {
        return nextRef;
    }

    public void setNextRef(Node<T> ref) {
        this.nextRef = ref;
    }

    @Override
    public int compareTo(T arg) {
        if (arg == this.value) {
            return 0;
        }
    }
}
```

INPUT:

Input is provided in main method.

OUTPUT:

```
Adding: 3
Adding: 32
Adding: 54
Adding: 89
Traversing to all nodes..
Deleted: 3
Traversing to all nodes..
Deleted: 89 32 54 76
```

Experiment No: 3B: Design Doubly Linked List.

Aim: Write a program in java to design doubly link list.

Description:

A doubly-linked list is a linked data structure that consists of a set of sequentially linked records called nodes. Each node contains two fields, called links that are references to the previous and to the next node in the sequence of nodes. The beginning and ending nodes previous and next links, respectively, point to some kind of terminator, typically a sentinel node or null, to facilitate traversal of the list. If there is only one sentinel node, then the list is circularly linked via the sentinel node. It can be conceptualized as two singly linked lists formed from the same data items, but in opposite sequential orders.

/* Java program DoublyLinkedList.java */

```
import java.util.NoSuchElementException;

public class DoublyLinkedList<E> {

    private Node head;
    private Node tail;
    private int size;

    public DoublyLinkedListImpl() {
        size = 0;
    }

    /**
     * this class keeps track of each element information
     *
     * @author java2novice
     *
     */
    private class Node {

        E element;
        Node next;
        Node prev;
        this.element = element;
        this.next = next;
        this.prev = prev;
    }
}

/**
 * returns the size of the linked list
 *
 * @return
 */
public int size() {
    return size;
}

/**
 * return whether the list is empty or not
 *
 * @return
 */
public boolean isEmpty() {
    return size == 0;
}

/**
 * adds element at the starting of the linked list
 *
 * @param element
 */
public void addFirst(E element) {
    Node tmp = new Node(element, head, null);
    if (head != null) {
        head.prev = tmp;
    }
    head = tmp;
    if (tail == null) {
        tail = tmp;
    }
    size++;
    System.out.println("adding: " + element);
}

/**
 * adds element at the end of the linked list
 *
 * @param element
 */
public void addLast(E element) {

    Node tmp = new Node(element, null, tail);
    if (tail != null) {
        tail.next = tmp;
    }
    tail = tmp;
    if (head == null) {
        head = tmp;
    }
```

```
}

/**
 * this method walks forward through the linked list
 */
public void iterateForward() {

    System.out.println("iterating forward..");
    Node tmp = head;
    while (tmp != null) {
        System.out.println(tmp.element);
        tmp = tmp.next;
    }
}

/**
 * this method walks backward through the linked list
 */
public void iterateBackward() {

    System.out.println("iterating backword..");
    Node tmp = tail;
    while (tmp != null) {
        System.out.println(tmp.element);
        tmp = tmp.prev;
    }
}

/**
 * this method removes element from the start of the linked list
 *
 * @return
 */
public E removeFirst() {
    if (size == 0) {
        throw new NoSuchElementException();
    }
    Node tmp = head;
    head = head.next;
    head.prev = null;
    size--;
    System.out.println("deleted: " + tmp.element);
    return tmp.element;
}

/**
 * this method removes element from the end of the linked list
 *
 * @return
 */
public E removeLast() {
    if (size == 0) {
        throw new NoSuchElementException();
    }
    Node tmp = tail;
    tail = tail.prev;
```

```
        size--;
        System.out.println("deleted: " + tmp.element);
        return tmp.element;
    }

    public static void main(String a[]) {

        DoublyLinkedListImpl<Integer> dll = new DoublyLinkedListImpl<Integer>();
        dll.addFirst(10);
        dll.addFirst(34);
        dll.addLast(56);
        dll.addLast(364);
        dll.iterateForward();
        dll.removeFirst();
        dll.removeLast();
        dll.iterateBackward();
    }
}
```

INPUT:

Input is provided in main method.

OUTPUT:

```
adding: 10
adding: 34
adding: 56
adding: 364
iterating forward.. 34 10 56 364
deleted: 34
deleted: 364
iterating backword.. 56 10
```

Experiment No: 4A: Methods of String: indexOf()

Aim: Write a program in java to implement indexOf().

Description:

The String class is immutable (constant), i.e. Strings in java, once created and initialized, cannot be changed. The String is a final class, no other class can extend it, and you cannot change the state of the string. String values cannot be compare with '==', for string value comparison, use equals() method. String class supports various methods, including comparing strings, extracting substrings, searching characters & substrings, converting into either lower case or upper case, etc.

Below method shows how to get index of a specified character or string from the given string. By using indexOf() method you get get the position of the specified string or char from the given string. You can also get the index strting from a specified position of the string.

/* Java program MyStringIndexOf.java */

```
public class MyStringIndexOf {

    public static void main(String[] a) {

        String str = "Use this string for testing this";
        System.out.println("Basic indexOf() example");
        System.out.println("Char 's' at first occurrence: " + str.indexOf('s'));
        System.out.println("String \"this\" at first occurrence: " + str.indexOf("this"));
        /**
        System.out.println("First occurrence of char 's' from 4th index onwards : "
                + str.indexOf('s', 4));
        System.out.println("First occurrence of String \"this\" from 6th index onwards: "
                + str.indexOf("this", 6));

    }
}
```

INPUT: Input is provided in main method.

OUTPUT:

```
Basic indexOf() example
Char 's' at first occurrence: 1
String "this" at first occurrence: 4
First occurrence of char 's' from 4th index onwards : 7
First occurrence of String "this" from 6th index onwards: 28
```

Experiment No: 4B: Methods of String: lastIndexOf()

Aim: Write a program in java to implement lastIndexOf().

Description:

Below example shows how to get index of a given character or string from a string in the reverse order, means last occurring index. By using lastIndexOf() method you can get last occurrence of the reference string or character.

/* Java program MyStrLastIndexOf.java */

```
public class MyStrLastIndexOf {

    public static void main(String a[]) {

        String str = "Use this string for testing this";
        System.out.println("Basic lastIndexOf() example");
        System.out.println("Char 's' at last occurrence: " + str.lastIndexOf('s'));
        System.out.println("String \"this\" at last occurrence: " + str.lastIndexOf("this"));
        /**
         * Returns the last occurrence from specified start index, searching
         * backward starting at the specified index.
         */
        System.out.println("first occurrence of char 's' from 24th index backwards: "
                + str.lastIndexOf('s', 24));
        System.out.println("First occurrence of String \"this\" from 26th index backwards: "
                + str.lastIndexOf("this", 26));
    }
}
```

INPUT: Input is provided in main method.

OUTPUT:

```
Basic lastIndexOf() example
Char 's' at last occurrence: 31
String "this" at last occurrence: 28
first occurrence of char 's' from 24th index backwards: 22
First occurrence of String "this" from 26th index backwards: 4
```

Aim: Write a program in java to implement startWith().

Description:

Below example shows how to find whether a string value start with another string value. By using startsWith() method, you can get whether the string starts with the given string or not. Also this method tells that the string occurrence at a specific position.

/* Java program MyStrStartsWith.java */

```
public class MyStrStartsWith {

    public static void main(String a[]) {

        String str = "This is an example string.";
        System.out.println("Is this string starts with \"This\"? "
                + str.startsWith("This"));
        System.out.println("Is this string starts with \"is\"? "
                + str.startsWith("is"));
        System.out.println("Is this string starts with \"is\" after index 5? "
                + str.startsWith("is", 5));
    }
}
```

INPUT: Input is provided in main method.

OUTPUT:

```
Is this string starts with "This"? true
Is this string starts with "is"? false
Is this string starts with "is" after index 5? true
```

Experiment No: 4D: Methods of String: endsWith()

Aim: Write a program in java to implement endsWith().

Description:

Below example shows how to find whether a string value ends with another string value. By using endsWith() method, you can get whether the string ends with the given string or not. Also this method tells that the string occurrence at a specific position.

/* Java program MyStringEnd.java */

```
public class MyStringEnd {

    public static void main(String a[]) {

        String str = "This is a java string example";
        if (str.endsWith("example")) {
            System.out.println("This String ends with example");
        } else {
            System.out.println("This String is not ending with example");
        }
        if (str.endsWith("java")) {
            System.out.println("This String ends with java");
        } else {
            System.out.println("This String is not ending with java");
        }
```

INPUT: Input is provided in main method.

OUTPUT:

```
This String ends with example
This String is not ending with java
```

Experiment No: 4E: Methods of String: split()

Aim: Write a program in java to implement split().

Description:

Below example shows how to split or brake a string. The split() method splits the string based on the given regular expression or delimiter, and returns the tokens in the form of array. Below example shows splitting string with space, and second split is based on any kind of spaces, that includes tab, enter, line breaks, etc.

/* Java program MyStrSplit.java */

```
public class MyStrSplit {

    public static void main(String a[]) {

        String str = "This program splits a string based on space";
        String[] tokens = str.split(" ");
        for (String s : tokens) {
            System.out.println(s);
        }
        str = "This     program  splits a string based on space";
        tokens = str.split("\\s+");
    }
}
```

INPUT: Input is provided in main method.

OUTPUT:

```
This
program
splits
a
string
based
on
space
```

Experiment No: 4F: Methods of String: getChars()

Aim: Write a program in java to implement getChars().

Description:

Below example shows how to copy range of characters from the given string to another character array. By suing getChars() method, you can copy range of characters from the given string.

/* Java program MyCharArrayCopy.java */

```
public class MyCharArrayCopy {
        String str = "Copy chars from this string";
        char[] ch = new char[5];
        /**
         * The getChars() method accepts 4 parameters first one is the start
         * index from string second one is the end index from string third one
         * is the destination char array forth one is the start index to append
         * in the char array.
         */
        str.getChars(5, 10, ch, 0);
        System.out.println(ch);
    }
}
```

INPUT: Input is provided in main method.

OUTPUT:

```
chars
```

Experiment No: 4G: Methods of String : replace()

Aim: Write a program in java to implement replace().

Description:

Below example shows how to get replace character or a string into a string with the given string. String provides replace() method to replace a specific character or a string which occurs first. replaceAll() method replaces a specific character or a string at each occurrence.

/* Java program MyStringReplace.java */

```
public class MyStringReplace {

    public static void main(String a[]) {

        String str = "This is an example string";
        System.out.println("Replace char 's' with 'o':" + str.replace('s', 'o'));

        System.out.println("Replace first occurrence of string\"is\" with \"ui\":"
                + str.replaceFirst("is", "ui"));

        System.out.println("Replacing \"is\" everywhere with \"no\":"
                + str.replaceAll("is", "no"));
    }
}
```

INPUT: Input is provided in main method.

OUTPUT:

```
Replace char 's' with 'o': This is an example string
Replace first occurrence of string “is" with "ui": This is an example string Replacing "is"
everywhere with "no": This no an example string
```

Experiment No: 4H: Methods of String : equals()

Aim: Write a program in java to implement equals().

Description:

The below example shows how to compare two string objects in java. You can not use "=="

Also you can ignore case during string compare by calling equalsIgnoreCase() method. '==' operator compares the object reference but not the string value.

/* Java program MyStringEquals.java */

```
public class MyStringEquals {

    public static void main(String a[]) {
        String x = "JUNK";
        String y = "junk";
        /**
         * We cannot use '==' operator to compare two strings. We have to use
         * equals() method.
         */
        if (x.equals(y)) {
            System.out.println("Both strings are equal.");
        } else {
            System.out.println("Both strings are not equal.");
        }
        /**
         * We can ignore case with equalsIgnoreCase() method
         */
        if (x.equalsIgnoreCase(y)) {
            System.out.println("Both strings are equal.");
        } else {
            System.out.println("Both strings are not equal.");
        }
    }
}
```

INPUT: Input is provided in main method.

OUTPUT:

```
Both strings are not equal.
Both strings are equal.
```

Experiment No: 4I: Methods of String: concat()

Aim: Write a program in java to implement concat().

Description:

Below example shows different ways of append or concat two string objects. You can append two strings by just using "+" sign. Also you can concatinate two string objects by calling concat() method.

/* Java program MyStringConcat.java */

```
public class MyStringConcat {

    public static void main(String a[]) {
        String b = "jump ";
        String c = "No jump";
        /**
         * We can do string concatenation by two ways. One is by using '+'
         * operator, shown below.
         */
        System.out.println(d);
        /**
         * Another way is by using concat() method, which appends the specified
         * string at the end.
         */
        d = b.concat(c);
        System.out.println(d);
    }
}
```

INPUT: Input is provided in main method.

OUTPUT:

```
jump No jump
jump No jump
```

Experiment No: 4J: Methods of String: copyValueOf()

Aim: Write a program in java to implement copyValueOf().

Description:

Below example shows how to convert character array to a string object. By using String.copyValueOf() method you can convert char array to string object. Also you can copy range of character array to string.

/* Java program MyArrayCopy.java */

```
public class MyArrayCopy {

    public static void main(String a[]) {
        char ch[] = {'M', 'y', ' ', 'J', 'a', 'v', 'a', ' ', 'e', 'x', 'a', 'm', 'p', 'l', 'e'};
        /**
         * We can copy a char array to a string by using copyValueOf() method.
         */
        String chStr = String.copyValueOf(ch);
        System.out.println(chStr);
        /**
         * We can also copy only range of charactors in a char array by
         * copyValueOf() method.
         */
        String subStr = String.copyValueOf(ch, 3, 4);
        System.out.println(subStr);
    }
}
```

INPUT: Input is provided in main method.

OUTPUT:

```
My Java example
Java
```

Experiment No: 4K: Methods of String: getBytes()

Aim: Write a program in java to implement getBytes().

Description:

Sometimes we have to convert string object into byte array. You can use getBytes() method

/* Java program SinglyLinkList.java */

```
public class MyStringBytes {

    public static void main(String a[]) {

        String str = "core java api";
        byte[] b = str.getBytes();
        System.out.println("String length: " + str.length());
        System.out.println("Byte array length: " + b.length);
    }
}
```

INPUT: Input is provided in main method.

OUTPUT:

```
String length: 13
Byte array length: 13
```

Experiment No: 5A: Implement polymorphism: overloading.

Aim: Write a program in java to implement method overloading.

Description:

Polymorphism is the capability of a method to do different things based on the object that it is acting upon. In other words, polymorphism allows you define one interface and have multiple implementations. I know it sounds confusing. Don’t worry we will discuss this in detail. It is a feature that allows one interface to be used for a general class of actions. An operation may exhibit different behavior in different instances. The behavior depends on the types of data used in the operation. It plays an important role in allowing objects having different internal structures to share the same external interface. Polymorphism is extensively used in implementing inheritance.

Following concepts demonstrate different types of polymorphism in java.

A) Method Overloading
B) Method Overriding

In Java, it is possible to define two or more methods of same name in a class, provided that there argument list or parameters are different. This concept is known as Method Overloading.

/* Java program Overload.java */

```
class OverloadDemo {

    void test() {
        System.out.println("No Parameters");
    }

    //overload test with one integer parameter
    void test(int a) {
        System.out.println("a : " + a);
    }

    //overload test with two integer parameter
    void test(int a, int b) {
        System.out.println("a & b : " + a + " " + b);
    }
    double test(double a) {
        return a * a;
    }
}


class Overload {

    public static void main(String args[]) {
        //create an instance for class OverloadDemo
        OverloadDemo od = new OverloadDemo();

        //call all test methods
        od.test();
        od.test(5);
        od.test(2, 3);
        System.out.println("Result : " + od.test(11.1));
    }
}
```

INPUT: Input is provided in main method

OUTPUT:

```
No Parameters
a : 5
a & B : 2 3
Result : 123.21
```

Experiment No: 5B: Implement different types of polymorphism: overriding

Aim: Write a program in java to implement method overriding.

Description:

Sub class has the same method as of super class. In such cases sub class overrides the super class method without even touching the source code of the super class. This feature is known as method overriding.

/* Java program OverrideTest.java */

```
class Surface {

    int length = 2;
    int width = 3;

    Surface(int l, int w) {
        length = l;
        width = w;
    }

    void showArea() {
        System.out.println("Area of the Surface is : " + (length * width));
    }
}

class Box extends Surface {

    int height = 4;
```

```
        System.out.println("Area of the Surface within Box is : " + (length * width));
    }

    void showVolume() {
        System.out.println("Volume of the Cube is : " + (length * width * height));
    }
}

class OverrideTest {

    public static void main(String args[]) {

        //create an instance of Superclass
        Surface s = new Surface();

        //access member of superclass
        s.showArea();

        //create an instance of subclass
        Box b = new Box();

        //access member of subclass & superclass
        b.showArea();
        b.showVolume();
    }
}
```

INPUT: Input is provided in main method.

OUTPUT:

```
        Area of the Surface is : 6
        Area of the Surface within Box is : 6
        Volume of the Cube is : 24
```

Experiment No: Implement inheritance.

Aim: Write a program in java to implement inheritance.

Description:

Inheritance is one of the features of Object-Oriented Programming (OOPs). Inheritance allows a class to use the properties and methods of another class. In other words, the derived class inherits the states and behaviors from the base class. The derived class is also called subclass and the base class is also known as super-class. The derived class can add its own additional variables and methods. These additional variable and methods differentiates the derived class from the base class.

/* Java program Inheritance.java */

```
// A class to display the attributes of the vehicle

class Vehicle {

    String color;
    int speed;
    int size;

    void attributes() {
        System.out.println("Color : " + color);
        System.out.println("Size : " + size);
    }
}

// A subclass which extends for vehicle
class Car extends Vehicle {

    int CC;
    int gears;

    void attributescar() {
        // The subclass refers to the members of the superclass
        System.out.println("Color of Car : " + color);
        System.out.println("Speed of Car : " + speed);
        System.out.println("Size of Car : " + size);
        System.out.println("CC of Car : " + CC);
        System.out.println("No of gears of Car : " + gears);
    }
}

public class Test {

    public static void main(String args[]) {
        Car b1 = new Car();
        b1.color = "Blue";
        b1.speed = 200;
        b1.size = 22;
        b1.CC = 1000;
        b1.gears = 5;
        b1.attributescar();
    }
}
```

INPUT: Input is provided in main method.

OUTPUT:

```
Color of Car : Blue
Speed of Car : 200
Size of Car : 22
CC of Car : 1000
No of gears of Car : 5
```

Experiment No: 7: Use of package with access specifier.

Aim: Write a program in java of package and access specifire.

Description:

Access Modifiers is the way of specifying the accessibility of a class and its members with respective to other classes and members.

Packages in Java are a mechanism to encapsulate a group of classes, interfaces and sub packages. Many implementations of Java use a hierarchical file system to manage source and class files. It is easy to organize class files into packages. All we need to do is put related class files in the same directory, give the directory a name that relates to the purpose of the classes, and add a line to the top of each class file that declares the package name, which is the same as the directory name where they reside. In java there are already many predefined packages that we use while programming. For example: java.lang, java.io, java.util etc. However one of the most useful feature of java is that we can define our own packages

```
// top-level interface declaration with public modifier
public interface IPub {…}

package pack1;
// top-level class declaration with default modifier
class CDef {….}
// top-level interface declaration with default modifier
interface CDef {…}

package pack1;
// another class in same package Pack1
class A1 {
   // public Class is accessible within the package
   CPub cPubObj;

   // default Class is accessible within the package
   CDef cDefObj;
}

package pack1;
// default Interface is accessible within the package
class B1 implements IDef {…}

package pack1;
// public Interface is accessible within the package
class C1 implements IPub {…}

package Pack2;
// Class in other package Pack1
class A2 {

// public Class is accessible in other package
CPub cPubObj;

// default Class is NOT accessible in other package
CDef cDefObj;
}

package pack2;
// default Interface is NOT accessible outside the package
class B2 implements IDef {…}

package pack2;
// public Interface is accessible outside the package
class C2 implements IPub {…}
```

INPUT: Not Applicable for this program.

OUTPUT:

public Members: If members are declared as public inside a class then such members are accessible to the classes which are inside and outside of the package where this class is visible. This is the least restrictive of all the accessibility modifiers.

protected Members:If members are declared as protected then these are accessible to all classes in the package and to all subclasses of its class in any package where this class is visible.

Default Members: When no accessibility modifier is specified for the member then implicitly it is declared as Default. These are accessible only to the other classes in the class's package.

private Members:This is the most restrictive of all accessibility modifiers. These members are accessible only with in the same class. These are not accessible from any other class within a class's package also.

Experiment No: 8: Write a program using static keyword.

Aim: Write a program in java to implement use of static keyword.

Description:

The static keyword in java is used for memory management mainly. We can apply java static keyword with variables, methods, blocks and nested class. The static keyword belongs to the class than instance of the class.

The static can be:

1. variable (also known as class variable)
2. method (also known as class method)
3. block
4. nested class

/* Java program UseStatic.java */

```
class UseStatic {
        static int a = 3;
        static int b;

        static void dis(int x) {
                System.out.println("x = "+x);
                System.out.println("a = "+a);
                System.out.println("b = "+b);
        }

        static {
                System.out.println("Static block initialized.");
                b = a * 4;
        }

        public static void main(String args[]) {
                dis(9);
        }
}
```

INPUT: Input is provided in main methos.

OUTPUT:

```
Static block initialized.
x = 9
a = 3
b = 12
```

Experiment No: 9A: Write a program to use this.

Aim: Write a program in java to show the use of this.

There can be a lot of usage of java this keyword. In java, this is a reference variable that refers to the current object.

Uses of this keyword:

1. this keyword can be used to refer current class instance variable.
2. this keyword can be used to invoke current class method (implicitly)
3. this can be passed as an argument in the method call.
4. this can be passed as argument in the constructor call.
5. this keyword can also be used to return the current class instance
6.

/* Java program Student.java */

```
//example of this keyword

class Student {

   int id;
   String name;

   Student(int id, String name) {
      this.id = id;
      this.name = name;
   }

   void display() {
      System.out.println(id + " " + name);
   }

   public static void main(String args[]) {
      Student s1 = new Student(111, "Ram");
      Student s2 = new Student(222, "Shan");
      s1.display();
      s2.display();
   }
}
```

INPUT: Input is provided in main method.

OUTPUT:

```
111 Ram
222 Shan
```

Experiment No: 9B: Write a program to use this().

Aim: Write a program in java to show the use of this().

Description:

The this() constructor call can be used to invoke the current class constructor (constructor chaining). This approach is better if you have many constructors in the class and want to reuse that constructor. this() can be used to invoke current class constructor.

/* Java program Student.java */

```
class Student {

   int id;
   String name;
   String city;
```

```
    Student(int id, String name) {
        this.id = id;
        this.name = name;
    }

    Student(int id, String name, String city) {
        this(id, name);//now no need to initialize id and name
        this.city = city;
    }

    void display() {
        System.out.println(id + " " + name + " " + city);
    }

    public static void main(String args[]) {
        Student e1 = new Student(111, "Ram");
        Student e2 = new Student(222, "Shan", "Kolkata");
        e1.display();
        e2.display();
    }
}
```

INPUT: is provided in main method.

OUTPUT:

```
            111 Ram null
            222 Shan Kolkata
```

Experiment No: 9C: Write a program to use super.

Aim: Write a program in java to show the use of super.

Description:

The super keyword in java is a reference variable that is used to refer immediate parent class object. Whenever you create the instance of subclass, an instance of parent class is created implicitly i.e. referred by super reference variable. super is used to refer immediate parent class instance variable and method.

/* Java program Bike.java */

```
//example of super keyword

class Vehicle {

    int speed = 50;
}

class Bike extends Vehicle {

    int speed = 100;

    void display() {
        System.out.println(super.speed);//will print speed of Vehicle now
    }

    public static void main(String args[]) {
        Bike b = new Bike();
        b.display();
```

INPUT: is provided in main method.

OUTPUT: 50

Experiment No: 9D: Write a program to use super().

Aim: Write a program in java to show the use of super().

Description: super() is used to invoke parent class constructor.

/* Java program Bike.java */

```
class Vehicle {

   Vehicle() {
      System.out.println("Vehicle is created");
   }
}

class Bike extends Vehicle {

   Bike() {
      super();//will invoke parent class constructor
      System.out.println("Bike is created");
   }

   public static void main(String args[]) {
      Bike b = new Bike();
   }
}
```

INPUT: Input is provided in main method.

OUTPUT:

```
          Vehicle is created
          Bike is created
```

Experiment No: 10: Exception handling.

Aim: Write a program in java to show the use of exception handling.

Description:

An exception is an event, which occurs during the execution of a program, that interrupts the normal flow of the program. It is an error thrown by a class or method reporting an error in code. The 'Throwable' class is the superclass of all errors and exceptions in the Java language Exceptions are broadly classified as 'checked exceptions' and 'unchecked exceptions'. All RuntimeExceptions and Errors are unchecked exceptions. Rest of the exceptions are called checked exceptions. Checked exceptions should be handled in the code to avoid compile time errors. Exceptions can be handled by using 'try-catch' block. Try block contains the code which is under observation for exceptions. The catch block contains the remedy for the exception. If any exception occurs in the try block then the control jumps to catch block. If a method doesn't handle the exception, then it is mandatory to specify the exception type in the method signature using 'throws' clause. We can explicitly throw an exception using 'throw' clause.

/* Java program MyClass.java */

```
public class MyClass{

    public void show(String[] str) throws ArithmeticException,
ArrayIndexOutOfBoundsException{
        if(str.length<2){
            throw new ArrayIndexOutOfBoundsException("Array length is "
+str.length+" It must be 2");
        } else if(n2==0){
                throw new ArithmeticException("Divided by 0, not possible");
        }else{
            int n1=Integer.parseInt(str[0]);
            int n2=Integer.parseInt(str[1]);
            int result=0;

            result=n1/n2;
            System.out.print("Result : "+result);
        }
    }
    public static void main(String args[]) {
    try {
            new MyClass().show(args);
        } catch(ArithmeticException aex){
            System.out.println(aex);
        } catch(ArrayIndexOutOfBoundsException aibex){
            System.out.println(aibex);
        }
    }
}
```

INPUT-1: If java MyClass 10 2 (command is executed)

OUTPUT-1: Result : 5

INPUT-2: If java MyClass 10 0 (command is executed)

OUTPUT-2: Divided by 0, not possible

INPUT-3: If java MyClass 10 (command is executed)

OUTPUT-4: Array length is 1 It must be 2

Experiment No: 11: Threading

Aim: Write a program in java to implement threading.

Description:

Threading is a facility to allow multiple tasks to run concurrently within a single process. Threads are independent, concurrent execution through a program, and each thread has its own stack.

In Java threads can be implemented in two ways. One is by 'Extending Thread Class' and the other way is by 'Implementing Runnable Interface'

Extending Thread Class is required to 'override run()' method. The run method contains the actual logic to be executed by thread.

Creation of thread object never starts execution, we need to call 'start()' method to run a thread. Examples gives you more details. Other methods supported by Threads are given below.

join(): It makes to wait for this thread to die. You can wait for a thread to finish by calling its join() method.

yield(): It makes current executing thread object to pause temporarily and gives control to other thread to execute.

notify(): This method is inherited from Object class. This method wakes up a single thread that is waiting on this object's monitor to acquire lock.

notifyAll(): This method is inherited from Object class. This method wakes up all threads that are waiting on this object's monitor to acquire lock.

wait(): This method is inherited from Object class. This method makes current thread to wait until another thread invokes the notify() or the notifyAll() for this object.

/* Java program ThreadTest.java */

```
public class ThreadTest {

    public static void main(String args[]) {

        System.out.println("Create ...");

        MyThread th1 = new MyThread("TH : 1");
        MyThread th2 = new MyThread("TH : 22");
        MyThread th3 = new MyThread("TH : 333");

        MyThread2 th21 = new MyThread2("TH2 : 1");
        MyThread2 th22 = new MyThread2("TH2 : 22");
        MyThread2 th23 = new MyThread2("TH2 : 333");

        System.out.println("Start ...");

        th1.start();
        th2.start();
        th3.start();
        new Thread(th21).start();
        new Thread(th22).start();
        new Thread(th23).start();
    }
}

class MyThread extends Thread {

    public MyThread(String str) {
        super(str);
    }

    public synchronized void access() {

        for (int i = 0; i < 5; i++) {
            System.out.println("Loop " + i + ": " + getName());

            try {
                if (getName().equals("TH : 1")) {
                    sleep(4000);
                }

            } catch (InterruptedException e) {
            }
        }
    }
    public void run() {
        access();
        System.out.println("-----------------");
        System.out.println("End: " + getName());
        System.out.println("-----------------");
    }
}

class MyThread2 implements Runnable {

    String str;

    public MyThread2(String str) {
        this.str = str;
    }

    public synchronized void access() {

        for (int i = 0; i < 5; i++) {
            System.out.println("Loop " + i + ": " + str);

            try {
                if (str.equals("TH : 1")) {
                    Thread.sleep(4000);
                }

            } catch (InterruptedException e) {
            }
        }
    }

    public void run() {
        access();
        System.out.println("-----------------");
        System.out.println("End: " + str);
        System.out.println("-----------------");
    }
}
```

INPUT: Input is provided in main method.

OUTPUT: Output sequences may changes time to time.

```
Create ...
Start ...
Loop 0: TH : 22
Loop 0: TH2 : 22
Loop 1: TH : 22
Loop 1: TH2 : 22
Loop 0: TH : 1
Loop 2: TH2 : 22
Loop 3: TH2 : 22
Loop 4: TH2 : 22
-----------------
End: TH2 : 22
-----------------
Loop 2: TH : 22
Loop 3: TH : 22
```

```
End: TH : 22
-----------------
Loop 0: TH2 : 1
Loop 1: TH2 : 1
Loop 2: TH2 : 1
Loop 3: TH2 : 1
Loop 4: TH2 : 1
-----------------
End: TH2 : 1
-----------------
Loop 0: TH : 333
Loop 1: TH : 333
Loop 2: TH : 333
Loop 3: TH : 333
Loop 4: TH : 333
-----------------
End: TH : 333
-----------------
Loop 0: TH2 : 333
Loop 1: TH2 : 333
Loop 2: TH2 : 333
Loop 3: TH2 : 333
Loop 4: TH2 : 333
-----------------
End: TH2 : 333
-----------------
Loop 1: TH : 1
Loop 2: TH : 1
Loop 3: TH : 1
Loop 4: TH : 1
-----------------
End: TH : 1
-----------------
```

Experiment No: 12: Applet programming

Aim: Write a program in java to implement action event with applet.

Description:

A Java applet is a small application which is written in Java or another programming language that compiles to Java bytecode and delivered to users in the form of that bytecode. The user launches the Java applet from a web page, and the applet is then executed within a Java Virtual Machine (JVM) in a process separate from the web browser itself. A Java applet can appear in a frame of the web page, a new application window, Sun's AppletViewer, or a stand-alone tool for testing applets. Java applets were introduced in the first version of the Java language, which was released in 1995.

Changing the state of an object is known as an event. For example, click on button, dragging mouse etc. The java.awt.event package provides many event classes and Listener interfaces for event handling. In computing, an event is an action or occurrence recognized by software that may be handled by the software. Computer events can be generated or triggered by the system, by the user or in other ways. Typically, events are handled synchronously with the program flow, that is, the software may have one or more dedicated places where events are handled, frequently an event loop. A source of events includes the user, who may interact with the software by way of, for example, keystrokes on the keyboard. Another source is a hardware device such as a timer. Software can also trigger its own set of events into the event loop, e.g. to communicate the completion of a task. Software that changes its behavior in response to events is said to be event-driven, often with the with the source. Once event is received by the listener, they processe the event and then return. Events are supported by a number of Java packages, like java.util, java.awt and java.awt.event. Any action that user performs on a GUI component must be listened and necessary action should to be taken. For example, if a user clicks on a Exit button, then we need to write code to exit the program. So for this, we need to know that the user has clicked the button. This process of knowing is called as listening and the action done by the user is called an event. Writing the corresponding code for a user action is called as Event handling. An event listener in Java is an interface that contains methods called handlers in which corresponding action code is to be written. An event class contains the information about an event.

Event source is the GUI component or model on which an event is generated or in other words an action is done.

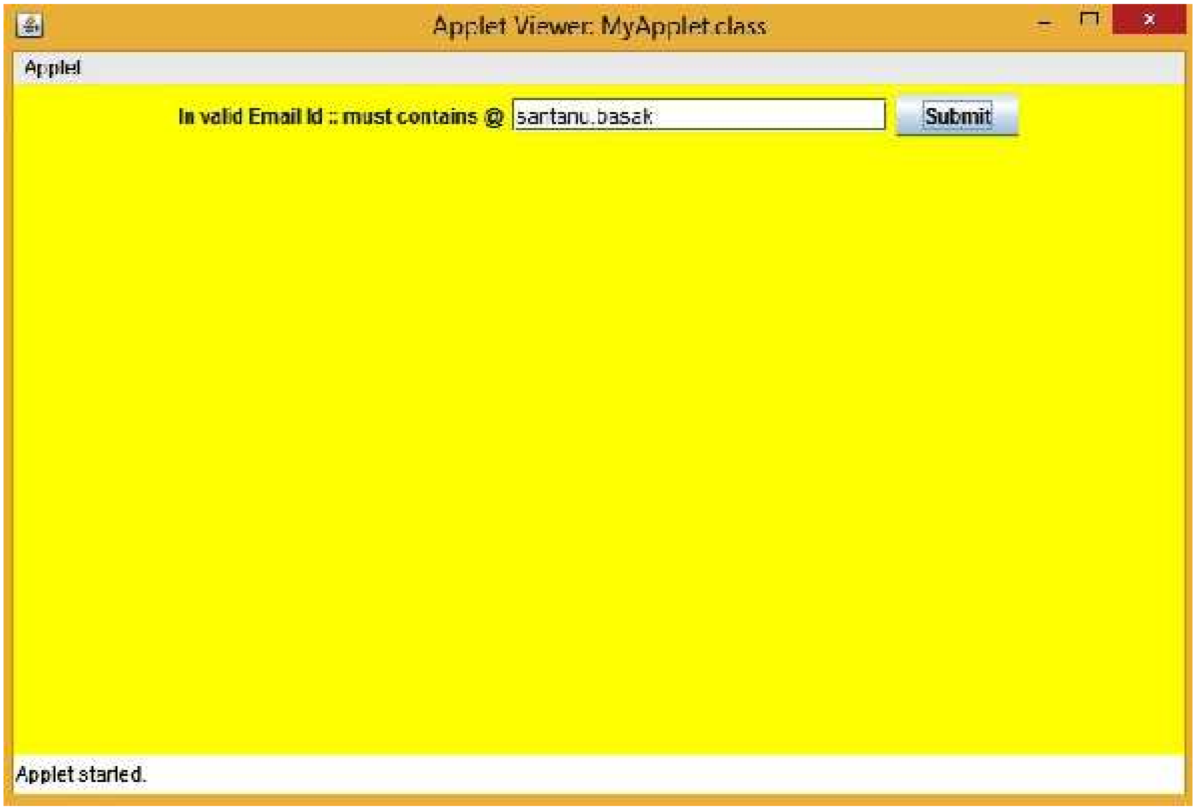
An adapter class is an abstract class implementing a listener interface. This is essential when we don't want to write all the handlers. For example, MouseListener interface contains a lot of methods such as mousePressed(), mouseReleased().. and we want to write only one of them, we use adapter class. This class implements all the methods of an interface giving them an empty body while itself being abstract. For every action user performs, a corresponding event object is generated. This generated event object should be sent to the corresponding listener so that we can handle that event and write the code accordingly. The process of sending of event object to its corresponding listener is called as event dispatching. Events cannot be dispatched if they aren't generated and an event, except MouseEvent cannot be generated on a disabled component.

/* Applet Program & Action Event */

```
import javax.swing.*;
import java.applet.*;
import java.awt.*;
import java.awt.event.*;

public class MyApplet extends Applet implements ActionListener {
        JTextField txtEmail;
        JLabel lblMsg;
        JButton btnSubmit;

        public void init(){
                txtEmail=new JTextField(20);
                btnSubmit = new JButton("Submit");
                lblMsg = new JLabel("****");
                setBackground(Color.YELLOW);

                add(lblMsg);
                add(txtEmail);
                add(btnSubmit);


                btnSubmit.addActionListener(this);
        }
        public void actionPerformed(ActionEvent ae){
                submit();
        }
        private void submit() {
        if(!txtEmail.getText().contains("@")){
                        lblMsg.setText("In valid Email Id :: must contains @");
                }else if(!txtEmail.getText().contains(".com")){
                        lblMsg.setText("In valid Email Id :: must contains .com");
                }else{
                        lblMsg.setText("Email Id valid");
                }
        }
```

INPUT: Typing santanu.basak in the textfield, press Submit button.

OUTPUT:



```
In valid Email Id :: must contains @   santanu.basak
Applet started.
```

Experiment No: 13: Swing programming and Layout

Aim: Write a program in java to implement layout with swing.

Description:

Layout means the arrangement of components within the container. In other way we can say that placing the components at a particular position within the container. The task of layouting the controls is done automatically by the Layout Manager. The LayoutManagers are used to arrange components in a particular manner. LayoutManager is an interface that is implemented by all the classes of layout managers. There are following classes that represent the layout managers:

java.awt.BorderLayout
java.awt.FlowLayout
java.awt.GridLayout
java.awt.CardLayout
java.awt.GridBagLayout
javax.swing.BoxLayout
javax.swing.GroupLayout
javax.swing.ScrollPaneLayout
javax.swing.SpringLayout

The BorderLayout is used to arrange the components in five regions: north, south, east, west and center. Each region (area) may contain one component only. It is the default layout of frame or window. The BorderLayout provides five constants for each region:

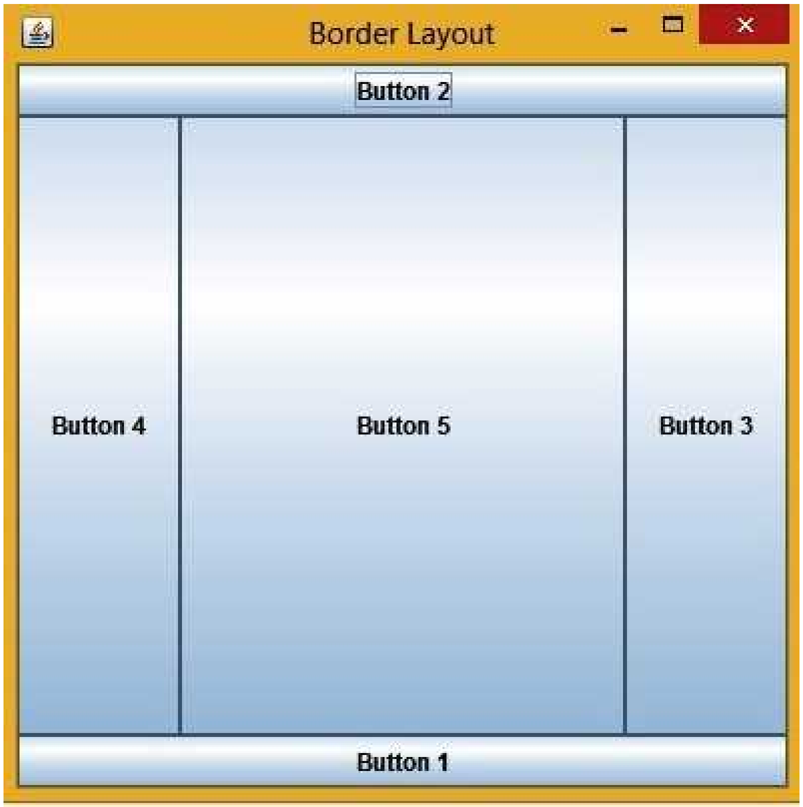
public static final int NORTH
public static final int SOUTH
public static final int EAST

/* Java program BorderLayoutExample.java */

```
import javax.swing.*;

public class BorderLayoutExample {

    JFrame frm;
    JButton btn1;
    JButton btn2;
    JButton btn3;
    JButton btn4;
    JButton btn5;

    public BorderLayoutExample() {
        init();
    }

    private void init() {
        frm = new JFrame("Border Layout");

        btn1 = new JButton("Button 1");
        btn2 = new JButton("Button 2");
        btn3 = new JButton("Button 3");
        btn4 = new JButton("Button 4");
        btn5 = new JButton("Button 5");

        prepare();
    }

    private void prepare() {
        frm.add(btn1, "South");
        frm.add(btn2, "North");
        frm.add(btn3, "East");
        frm.add(btn4, "West");
        frm.add(btn5, "Center");

        frm.setSize(400, 400);

        frm.setVisible(true);
    }

    private void show() {
        frm.setVisible(true);
    }

    public static void main(String args[]) {
        BorderLayoutExample reg = new BorderLayoutExample();
        reg.show();

    }
}
```

INPUT: Input is not required.

OUTPUT:


Border Layout
Button 2
Button 4
Button 5
Button 3
Button 1