# UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

## Lecture-wise Plan

Subject Name: **Economics for Engineers** Subject Code: **HU-501**
Year: **3RD Year** Semester: **Fifth**

| Module Number | Topics | Number of Lectures |
|---|---|---|
| 1 | 1. Economic Decisions Making – Overview, Problems, Role, Decision making process.<br>2. Engineering Costs & Estimation– Fixed, Variable, Marginal & Average Costs, Sunk Costs ,Opportunity Costs, Recurring And Nonrecurring 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. | **10L** |
| 2 | 3. Cash Flow, Interest and Equivalence: Cash Flow–Diagrams, Categories & Computation, Time Value of Money, Debtre payment, Nominal& Effective Interest.<br>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 Breakeven Analysis. Economic Analysis In The Public Sector – Quantifying And Valuing Benefits & drawbacks. | **6L** |

| | | |
|---|---|---|
| 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.<br>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, and Multiple Alternatives.<br>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. | **6L** |
| 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.<br>9. Replacement Analysis- Replacement Analysis Decision Map, Minimum Cost Life of a New Asset, Marginal Cost, Minimum Cost Life Problems.<br>10. Accounting–Function, Balance Sheet, Income Statement, Financial Ratios Capital Transactions, Cost Accounting, Direct and Indirect Costs, Indirect Cost Allocation. | **8L** |
| | **TOTAL NO. OF HOURS= 30L** | |

# UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

## Lecture-wise Plan

Subject Name: **Electrical Machines-II** Subject Code: **EE501**

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

| Module Number | Topics | Number of Lectures |
|---|---|---|
| 1 | **Single phase induction motor** | **10L** |
| | 1. Construction, Double revolving field theory | 3L |
| | 2. Starting methods | 1L |
| | 3. Phasor diagram, Speed — Torque characteristics, Condition of maximum torque | 2L |
| | 4. Determination of equivalent circuit parameters, Applications | 2L |
| | 5. Single Phase AC series motor, Compensated & uncompensated motors | 2L |
| 2 | **Synchronous machines** | **18L** |
| | 1. Construction | 1L |
| | 2. Armature Winding, winding factors | 2L |
| | 3. Excitation systems | 1L |
| | 4. Armature reaction | 1L |
| | 5. Theory for salient pole machine, Two reaction theory | 2L |
| | 6. Voltage regulation (EMF,MMF,ZPF) | 3L |
| | 7. Parallel operation of Alternators | 3L |
| | 8. Synchronous machine connected to infinite bus, effect of change of excitation and speed of prime mover | 2L |
| | 9. Construction and Starting of Synchronous motor | 2L |
| | 10. V- Curve, Damper winding. Hunting | 1L |
| 3 | **Special Electromechanical Devices** | **7L** |
| | 1. Principle and construction of Reluctance motor | 1L |
| | 2. Permanent magnet machines, Brushless D.C machines | 1L |

| | | |
|---|---|---|
| | 3. Stepper motor | 2L |
| | 4. AC servo motors | 1L |
| | 5. Principle, Construction and operational characteristics of Induction Generators | 2L |
| **Total Number Of Hours = 35L** | | |

Faculty In-Charge HOD, CSE Dept.

# UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

## Lecture-wise Plan

**Subject Name:-**Power System-I **Subject Code**:-EE502
**Year:** Third Year **Semester:** -Fifth

| Module No. | Topics | Planned Lectures(H) |
|---|---|---|
| 1. | **Overhead transmission line:-** | **8 H** |
| | 1. Choice of frequency and Choice of voltage | 1 H |
| | 2. Type of conductors | 1 H |
| | 3. Inductance and capacitance of single phase. | 1 H |
| | 4. Three phase symmetrical configuration. | 1 H |
| | 5. Three phase unsymmetrical configuration. | 1 H |
| | 6. Bundle conductors. | 1 H |
| | 7. Transposition concept of GMD and GMR | 1 H |
| | 8. Influence of earth on conductor capacitance | 1 H |
| 2. | **Overhead Line construction:-** | **4 H** |
| | 1. Line support | 2 H |
| | 2. Tower, poles, Sag, Tension and Clearance. | 1 H |
| | 3. Effect of wind and Ice on Sag, Dampers. | 1 H |
| 2. | **Insulators:-** | **4H** |
| | 1. Types of Insulator, Voltage distribution across a suspension insulator string | 1H |
| | 2. String efficiency, Arching shield and rings. | 1H |
| | 3. Methods of improving voltage distribution across the Insulator string, | 1H |
| | 4. Electrical test on line Insulators. | 1H |
| 3. | **Corona:** | **5H** |
| | 1. Principle of corona formation, critical disruptive voltage. | 2H |
| | 2. Visual critical corona discharge potential, corona loss. | 2H |
| | 3. Advantage and disadvantages of corona, Methods of reduction of corona. | 1H |
| 4. | **Cables:** | **3H** |
| | **1.** Types of cables, cable components. | 1H |
| | 2. Capacitance of single core and 3 core cables. | 1H |
| | 3. Dielectric stress, optimum cable thickness , grading, dielectric loss and loss angle. | 1 H |
| 5. | **Performance of lines:-** | **5H** |
| | 1. Short, medium (nominal π, T) and long lines and their representation, A,B,C,D constants, | 2H |
| | 2. Voltage regulation, Ferranti effect. | 2H |
| | 3. Power equations and line compensation, Power circle diagrams. | 1 H |
| | **Generation of Electric Power:-** | **4H** |
| | 1. General layout of a typical coal fired power station, hydroelectric power station, Nuclear power station, their components and working | 2 H |

| | | |
|---|---|---|
| 6. | principles.<br>2. Comparison of different methods of power generation, Introduction to solar and wind energy system. | <br>2 H |
| 7. | **Tariff and Indian Electricity Rule-1956:**<br>3. Guiding principle of tariff, different types of tariff. General Introduction | **1H**<br>1H |
| **TOTAL HOUR REQUIRED=34** | | |

Faculty In-Charge HOD, EE Dept.

# UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

## Lecture-wise Plan

**Subject Name:**-Control system-1 **Subject Code**:-EE503

**Year:**Third Year **Semester:** -Fifth

| Module No. | Topics | Planned Lectures(H) |
|---|---|---|
| 1. | **INTRODUCTION TO CONTROL SYSTEM:** | **5 H** |
| | a) . Concept of feedback and Automatic control, Effects of feedback, Definition of linear and nonlinear systems, Elementary concepts ofsensitivity and robustness. Types of control systems | 01 |
| | b) Objectives of control system, Servomechanisms and regulators, examples of feedback control systems | 02 |
| | c) Transfer function concept. Pole and Zeroes of a transfer function. Properties of Transfer function | 02 |
| 2. | **MATHEMATICAL MODELING OF DYNAMIC SYSTEMS:** | **5H** |
| | a) Translational systems, Rotational systems, Mechanical coupling, Liquid level systems | 02 |
| | b) Electrical analogy of Spring–Mass-Dashpot system | 01 |
| | c) Block diagram representation of control systems. Block diagram algebra. Signal flow graph. Mason's gain formula | 02 |
| 3. | **CONTROL SYSTEM COMPONENTS:** | **4H** |
| | a) Potentiometer, Synchros, Resolvers, Position encoders. DC and AC tacho-generators. Actuators. Block diagram level description of feedback control systems for position control | 02 |
| | b) speed control of DC motors, temperature control, liquid level control, voltage control of an Alternator. | 02 |
| 4. | **TIME DOMAIN ANALYSIS:** | **6H** |
| | a) Time domain analysis of a standard second order closed loop system. Concept of undamped natural frequency | 02 |
| | b) damping, overshoot, rise time and settling time.Dependence of time domain performance parameters on natural frequency and damping ratio. | 01 |
| | c) Step and Impulse response of first and second order systems. Effects of Pole and Zeros on transient response. | 02 |

| | | |
|---|---|---|
| | Stability by pole location.<br><br>d) Routh-Hurwitz criteria and applications | 01 |
| 5. | **ERROR ANALYSIS:** | **3H** |
| | a) Steady state errors in control systems due to step, ramp and parabolic inputs.<br>Concepts of system types and error constants | 03 |
| 6. | **STABILITY ANALYSIS:** | **3H** |
| | a) Root locus techniques, construction of Root Loci for simple systems. Effects of gain on the movement of Pole and Zeros | 03 |
| **Module No.** | **TOPICS** | **Planed Lectures** |
| 7. | **FREQUENCY DOMAIN ANALYSIS OF LINEAR SYSTEM:** | **7h** |
| | a) Bode plots<br><br>b) Polar plots, Nichols chart, Concept of resonance frequency of peak magnification<br><br>c) Nyquist criteria, measure of relative stability, phase and gain margin<br><br>d) Determination of margins in Bode plot. Nichols chart. M-circle and M-Contours in Nichols chart. | 03<br><br>02<br><br>02<br><br>01 |
| 8. | **CONTROL SYSTEM PERFORMANCE MEASURE:** | **3H** |
| | a) Improvement of system performance through compensation. Lead, Lag and Lead- lag compensation, PI, PD and PID control | 03 |
| **TOTAL HOUR REQUIRED=36 h** | | |

ASSIGNMENTS:

MODULE1:

1. What is the effect of adding feedback to a control system?
2. Explain sensitivity and robustness.
3. What are the differences between open and closed loop control system?
4. What is stochastic and adaptive control system?

# UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

## Lecture-wise Plan

**Subject Name:**-Control system-1 **Subject Code**:-EE503

**Year:**Third Year **Semester:** -Fifth

MODULE2 & 3:

1. What is an electrical analogous of spring, mass and damper system? Explain.
2. What are the prime differences between block diagram reduction and signal flow graph?
3. Write a short note on potentiometer.
4. Write a short note on resolver.
5. Write a short note on synchro.
6. What are the major methods for the speed control of DC motor? Explain the methods.

MODULE 4:

1. What is natural frequency of oscillation?
2. Explain damping ratio.
3. Deduce the expression for step response in a second order system.
4. Deduce the expression for peak time, peak overshoot and rise time.
5. Analyse the system from stability point of view with the help of Routh – Hurwitz criteria:

$$S^6+2s^5+5s^4+3s^3+s^2+s+4 = 0$$

MODULE 5:

1. Explain the terms: Position error constant, velocity error constant, acceleration error constant.
2. What is the significance of steady state error?
3. What is the effect on adding a pole or a zero in a transfer function?

MODULE 6:

1. What is root locus?
2. What is break away and break in point?
3. How the gain of the system varies with the variation of pole and zero?
4. What is an asymptote? Explain the significance.

MODULE 7:

1. What is gain and phase margin? How they effect stability?
2. What is the significance of gain crossover frequency and phase crossover frequency?
3. What is resonant frequency and bandwidth?
4. Explain Nyquist's criterion.
5. Compare relative stability with absolute stability.

MODULE 8:

1. Write short notes on lead and lag compensator.
2. Explain the significance of P, PI and PID controllers.

Faculty In-Charge HOD, EE Dept.

# UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

## Lecture-wise Plan

Subject Name: Advanced OOPs using C++ Subject Code-EE504A

Year:3RD Year Semester: **Fifth**

| Module Number | Topics | Number of Lectures |
|---|---|---|
| 1 | **Introduction** | **5L** |
| | 1. Basics of OOP, Features; Structure of C++ program; Class and object; Concept of Constructor& destructor; Abstraction | 2L |
| | 2. Encapsulation; Inheritance; | 1L |
| | 3. Static and dynamic binding;Polymorphism. | 2L |
| 2 | **Exception Handling** | **5L** |
| | 1. Exception handling mechanism; throwing, catching, rethrowing mechanism; | 2L |
| | 2. Multiple catch statement; Nested try-catch block; | 2L |
| | 3. Exception in constructor & destructor; exceptions in operator overloaded functions. | 1L |
| 3 | **Template** | **6L** |
| | 1. Class template; Member function inclusion; Class template with different parameter; | 3L |
| | 2. Function template; Function template with multiple parameters; | 2L |
| | 3. Overloading of template function; member function template. | 1L |
| 4 | **Console I/O operations** | **6L** |
| | 1. C++ streams; C++ stream classes; | 1L |
| | 2. Unformatted I/O operations; | 2L |
| | 3. Formatted I/O operations; | 2L |
| | 4. Managing output with Manipulators. | 1L |
| 5 | **Working with Files** | **10L** |
| | 1. Data File Handling: Need for a data file, Types of data files – Text file and Binary file; | 2L |
| | 2. Text File: Basic file operations on text file: Creating/Writing text into file, reading and manipulation of text from an already existing text File (accessing sequentially). | 4L |
| | 3. Binary File: Creation of file, Writing data into file, Searching for required data from file, Appending data to a file, Insertion of data in sorted file, Deletion of data from file, Modification of data in a file; opening and closing files; classes | 4L |

| | | |
|---|---|---|
| | for file stream operations; Error handling during file operations; command line arguments. | |
| 6 | **Standard Template Library** | **4L** |
| | 1. Components of STL; Containers, Iterator;<br>2. Applications of container classes. | 2L<br>2L |
| | **Standard Functions Library** | **4L** |
| | 1. C-based I/O functions (fflush, fgetc, ferror, fscanf, fprintf etc.); Time, Date, Localization functions (asctime, clock, ctime, difftime, localtimemktime,strftime etc.);<br>2. Dynamic memory allocation functions (calloc, malloc, realloc, free). | 2L<br>2L |
| 7 | **String Manipulation** | **4L** |
| | 1. The String class; Creating String object; Manipulating strings;<br>2. Relational operations on strings; String comparison characteristics, swapping; Accessing characters in strings. | 1L<br>3L |
| **Total Number Of Lectures = 44** | | |

Faculty In-Charge HOD, CSE Dept.

# UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

## Lecture-wise Plan

Subject Name: Advanced OOPs using C++ Subject Code-EE504A
Year:3RD Year Semester: **Fifth**
**Assignment:**

**Module-1: Introduction**

1. Write a program in which a class has three data members: name, roll no, marks of 5 subjects and a member function Assign() to assign the streams on the basis of table given below:

| Avg. Marks | Stream |
|---|---|
| 90% or more | Computers |
| 80% - 89% | Electronics |
| 75% - 79% | Mechanical |
| 70% - 74% | Electrical |

2. What is virtual base class? Write a small code to explain it.Why pure virtual functions are needed?Define a function area() to compute the area of objects of different classes – triangle, rectangle, square. Invoke these in the main program. Comment on the binding (static or dynamic) that takes place in your program.

**Module-2: Exception Handling**

1. Explain exceptions with different types. How exception is handled in C++?Write down the steps by which we can handle the exceptions with a proper example.
2. How we can restrict a function to throw only certain specified exception in C++. Explain with proper example.When we use catch(…) handler? Explain with a proper example.Write programs that demonstrate how certain exception types are not allowed to be thrown.

**Module-3: Template**

1. What do you mean by generic programming?Explain class template and function template with proper example.A template can be considered as a kind of macro. Then what is the difference between them?
2. What is the difference between class template and template class?Write a function template to perform linear search/ binary search/ bubble sort/ merge sort/ selection sort in an array.Write a C++ program where template function is overloaded.Explain inline function template.

**Module-4: Console I/O operations**

1. Differentiate get(), getline() and write() function with proper example.What do you mean by stream? Write down its different types of stream with proper example.What is streambuf? Explain stream.Why it is necessary to include the file iostream in all C++ program?
2. A. What the following statement do?
   i. cout.write(s1,m).write(s2,n)
   ii. cout.write(line, size)
   iii. cout.precision(a)

   B. How do the following two statements differ in operation?
   a. Cin>>a;
   b. Cin.get(a);

**Module-5: Working with Files**

1. Write a C++ program to write number 1 to 100 in a data file NOTES.TXT.Explain how while(f1) statement detects the end of file that is connected to f1 stream.
2. Explain the difference between normal text file and binary file. What are the advantages of saving data in binary form?How many file objects are required to do the following operation?
   iv. To process four files sequentially.
   v. To merge two files into third file.

**Module-6: STL**

1. What are the different types of algorithm in STL? Explain.What are the applications of container class?How list works in STL? Explain with proper example.What do you mean by function objects? How you use function objects in algorithm?
2. A table gives a list of car models and the number of units sold in each type in a specified period. Write a program to store this table in a suitable container, and to display interactively the total value of a particular model sold, given the unit-cost of that model.

**Module-7: String Manipulation**

1. What is the importance of using string class? What do you mean by C-Style string?Write a C++ program to create string objects in C++.
2. Write a program that reads the name "RAMAN KUMAR BANERJEE" from keyboard into three separate string objects and then concatenates them into a new string object using
    i. + operator
    ii. append() function

# UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

## Lecture-wise Plan

Subject Name: **Computer Organization** Subject Code: **EE504B**

Year: **3RD Year** Semester: **Fifth**

| Module Number | Topics | Number of Lectures |
|---|---|---|
| 1 | **Introduction to digital computer organization** | **7L** |
| | 1. Concept of basic components of a digital computer, High level view of computer.<br>2. Commonly used number systems<br>3. Fixed and floating point representation of a number<br>4. Booth's Algorithm – restoring and non restoring. | 6L |
| | 5. What are the H/W resources that we will need in computer<br>6. Conversion of high level code to m/c level language. | 1L |
| 2 | **CPU design** | **3L** |
| | 1. Basic organization of the stored program computer and operation sequence for execution of a program.<br>2. Description of ALU, Design of circuit of a small scale CPU | 2L |
| | 3. Fetch, decode and execute cycle, Concept of operator, operand, registers and storage, Instruction format. Instruction sets and addressing modes. | 1L |
| 3 | **CPU design - Timing and control** | **2L** |
| | 1. Timing diagram and control design | 2L |
| 4 | **Micro programmed control** | **5L** |
| | 1. Concepts of Micro operations<br>2. Horizontal and vertical micro-program<br>3. Optimization of hardware resources for designing of micro-programmed control unit | 5L |
| 5 | **Pipeline concept** | **3L** |
| | 1. Instruction and Arithmetic Pipelining,<br>2. Synchronous and Asynchronous pipeline<br>3. Solving problems on pipeline speed-up, efficiency, throughput | 2L |
| 6 | **Memory Organization** | **5L** |
| | 1. Static and dynamic memory, Memory hierarchy, Associative memory, Bare machine | 3L |
| | 2. Memory unit design with special emphasis on implementation of CPU-memory interfacing. | 2L |

# UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

## Lecture-wise Plan

Subject Name: **Computer Organization** Subject Code: **CS304**
Year: **2nd Year** Semester: **Third**

| | | |
|---|---|---|
| 7 | **Cache memory Architecture** | **4L** |
| | 1. Cache memory organizations, Set associative cache | 1L |
| | 2. Techniques for reducing cache misses | 1L |
| | 3. Discussion on Buffer cache | 2L |
| 8 | **RAM architecture** | **2L** |
| | 1. Basic concepts of architecture of RAM | 2L |
| 9 | **Discussion on DRAM & SRAM** | **3L** |
| | 1. Architecture of Static Ram and Dynamic Ram | 2L |
| | 2. Difference between SRAM and DRAM | 1L |
| 10 | **I-O subsystem organization** | **3L** |
| | 1. Concept of handshaking, Polled I/O | 1L |
| | 2. Interrupt and DMA | 2L |
| **Total Number Of Hours = 37L** | | |

Faculty In-Charge HOD, CSE Dept.

# UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

## Lecture-wise Plan

Subject Name: **Computer Organization** Subject Code: **EE504B**
Year: **3RD Year** Semester: **Fifth**

**Assignments:-**

**Unit 1:-**

1. What is the role of operating system?
2. Discuss the basic organization of digital computer.
3. Convert (1B3F)16 → (?)8
4. Why do we need 2's compliment method?
5. Briefly explain IEEE 754 standard format for floating point representation in single precision.
6. Using Booth's algorithm multiply ( – 12 ) and ( + 6 ).
7. Explain various assembler directives used in assembly language program.
8. Write +710 in IEEE 754 floating point representation in double precision.

**Unit 2 + Unit 3 + Unit 4:-**

1. What are the advantages of micro programming control over hardwired control?
2. Write down the control sequence for Move (R1), R2.
3. Define hardwired control.
4. Discuss the principle of operation of a micro programmed control.
5. Write the control sequence for execution of the instruction Add(R3), R1.
6. Give the organization of typical hardwired control unit and explain the functions performed by the various blocks.
7. Explain the multiple bus organization in detail.
8. What is microprogrammed sequencer?
9. Design a basic ALU which can perform 8 different arithmetic and logical operations.
10. Design the basic block diagram of Intel microprocessor 8085 and discuss the working principle of the different parts of it.

**Unit 5 :-**

1. What is pipelining?
2. What are the major characteristics of a pipeline?
3. What is a pipeline hazard?
4. What is the use of pipelining?
5. What are the remedies commonly adopted to overcome/minimize these hazards.
6. What is the ideal speedup expected in a pipelined architecture with n stages. Justify your answer.
7. Draw the structure of two stage instruction pipeline.

**Unit 6 + Unit 7 + Unit 8 + Unit 9 :-**

1. Define Memory Access time for a computer system with two levels of caches.
2. How to construct an 8M * 32 memory using 512 K * 8 memory chips.
3. Write two advantages of MOS device.

4. List the factors that determine the storage device performance.
5. What will be the width of address and data buses for a 512K * 8 memory chip?
6. Define memory cycle time.
7. What is RAM?
8. What is cache memory?
9. Explain virtual memory.
10. List the various semiconductors RAMs?
11. Define DRAM's.
12. What is ROM?
13. Give the format for main memory address using direct mapping function for 4096 blocks in main memory and 128 blocks in cache with 16 blocks per cache.
14. Give the format for main memory address using associative mapping function for 4096 blocks in main memory and 128 blocks in cache with 16 blocks per cache.
15. Give the format for main memory address using set associative mapping function for 4096 blocks in main memory and 128 blocks in cache with 16 blocks per cache.
16. Define Hit and Miss rate?
17. What are the enhancements used in the memory management?
18. Define latency time.
19. A computer system has a main memory consisting of 16 M words. It also has a 32Kword cache organized in the block-set-associative manner, with 4 blocks per set and 128 words per block.
20. How will the main memory address look like for a fully associative mapped cache?
21. A digital computer has a memory unit of 64K*16 and a cache memory of 1K words. The cache uses direct mapping with a block size of four words. How many bits are there in the tag, index, block and word fields of the address format? How many blocks can the caches accommodate?
22. Define the terms "spatial locality" and "temporal locality", and explain how caches are used to exploit them for a performance benefit. Be specific in the different ways that caches exploit these two phenomena.
23. Suppose physical addresses are 32 bits wide. Suppose there is a cache containing 256K words of data (not including tag bits), and each cache block contains 4 words. For each of the following cache configurations,
    a. direct mapped
    b. 2-way set associative
    c. 4-way set associative
    d. fully associative
    specify how the 32-bit address would be partitioned. For example, for a direct mapped cache, you would need to specify which bits are used to select the cache entry and which bits are used to compare against the tag stored in the cache entry.
24. Cache misses can be characterized as one of the following: compulsory misses, capacity misses, and conflict misses. Describe how each of these kinds of misses can be addressed in the hardware.
25. Why virtual memory is called virtual ? What are the different address spaces ? Explain with example how logical address is converted into physical address and also explain how page

replacements take place. Explain the instruction cycle with a neat diagram. Explain the disadvantages of stored program computer.

26. A three-level memory system having cache access time of 5 nsec and disk access time of 40 nsec, has a cache hit ratio of 0.96 and main memory hit ratio of 0.9. What should be the main memory access time to achieve an overall access time of 16 nsec ?
27. According to the following information, determine size of the subfields ( in bits ) in the address for Direct Mapping and Set Associative Mapping cache schemes :
    We have 256 MB main memory and 1 MB cache memory
    The address space of the processor is 256 MB
    The block size is 128 bytes
    There are 8 blocks in a cache set.

**Unit 10 :-**

1. What are the functions of I/O interface?
2. How does the processor handle an interrupt request?
3. What are the necessary operations needed to start an I/O operation using DMA?
4. What is the advantage of using interrupt initiated data transfer?
5. Why do you need DMA?
6. What is the difference between subroutine and interrupt service routine?
7. Why I/O devices cannot be directly be connected to the system bus?
8. What is polling?
9. State the differences between memory mapped I/O and I/O mapped I/O.
10. Explain the functions to be performed by a typical I/O interface with a typical input output interface.

# UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

## Lecture-wise Plan

Subject Name: Microprocessors & Microcontrollers Subject Code- EE504C
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 |
| | **8085 INTERRUPTS:** | **3L** |
| | 1. Introduction of various type of Interrupts | 1L |

| | | |
|---|---|---|
| 6 | 2. Concept of EI, DI, SIM, RIM instructions and examples | 1L |
| | 3. Hardware Interrupts including INTR (Handshake Interrupt) and INA | 1L |
| 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?

# UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

## Lecture-wise Plan

Subject Name: Microprocessors & Microcontrollers Subject Code- EE504C
Year: 3rd Year Semester: Fifth

**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

## Lab Manual

**Title of Course: Electric Machine-II Lab**
**Course Code: EE591**
**L-T-P scheme: 0-0-3** **Course Credit: 2**

**Objectives:**
**1.** To introduce the student fundamentals of Electromechanical energy conversion
**2.** Providing an in-depth understanding of Electrical Machine.
**3.** To learn the role of Electrical Machines in real life applications.

**Learning Outcomes:** The students will have a detailed knowledge of the concepts Electrical Machines and their operating principles. Upon the completion of Operating Systems practical course, the student will be able to:

- **Understand** constructional details, principle of operation, Performance of Induction Motors.
- **Understand** general features of alternators.
- **Learn** the concepts of voltage regulation.
- **Learn** the Concepts of Synchronous motor operation

**Course Contents:**
**Exercises that must be done in this course are listed below:**
Exercise No.1:Study the star delta starter of three phase induction motor
Exercise No. 2: Study the characteristics of three phase alternator by OCC & SCC test
Exercise No. 3: Perform slip test and determine $X_d$&$X_q$of an alternator.
Exercise No. 4: Perform no load and block rotor test of single phase induction motor.
Exercise No. 5: Study V curve of Synchronous Motor.

**Text Book:**
Ashfaq Husain, Electric Machines, DhanpatRai& Co.

**Recommended Systems/Apparatus Requirements:**
Laboratory Kits, Multimeters, Connecting wires, Watt Meters.

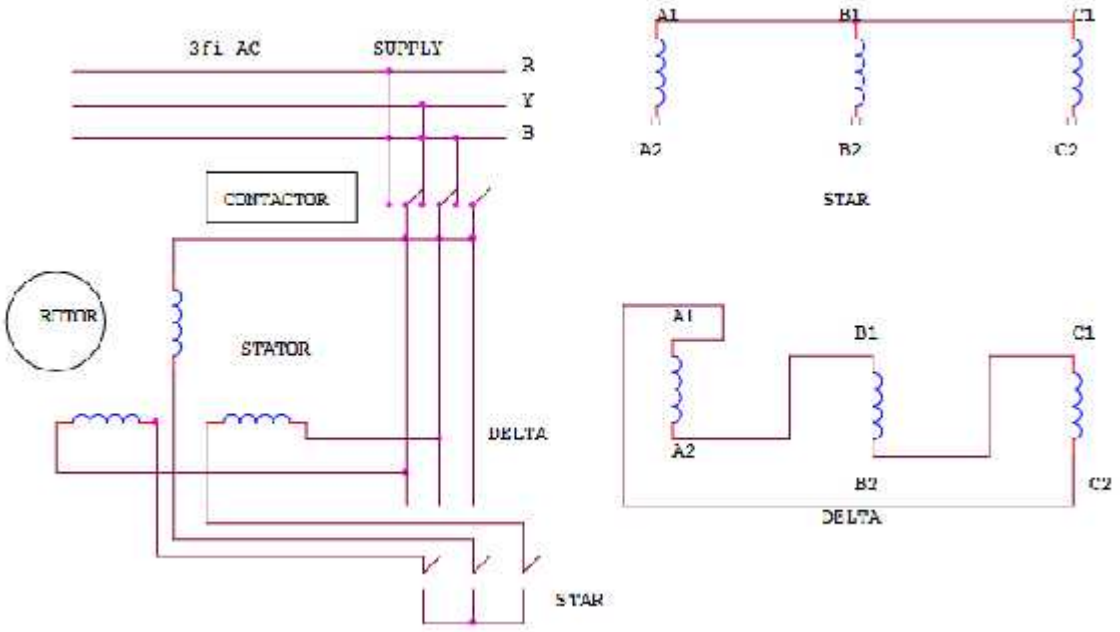
**Experiment No: 1**
Study the star delta starter of three phase induction motor

**Aim:**

**To study the star delta starter of three phase induction motor**

**Theory:**

At the standstill the motor behaves as the short circuit secondary transformer and it draws heavy current from mains, which can cause the damages at the starting. It can cause the heavy drops in power line. So direct online starting of motor is not desirable. The motor has to be started at reduced voltage. For heavy duty motors some starting methods are used or resistance has to be included in the circuit at starting




**Procedure:**

1. All the six terminals of stator winding are brought out and are connected as shown in Fig.
2. In the starting the stator winding is connected in start and full voltage is applied across these terminals.
3. The voltage of each phase is 1/3 of normal value. As the motor picks up the speed, the change over switch disconnects the winding of motor.
4. Now it connects the winding in delta across supply terminals.
5. This method reduces the current taken by the motor to one third the current it would have drawn if it was directly connected in delta

**Observation Table:**
Compare the starting current and running current .

**Conclusion:**

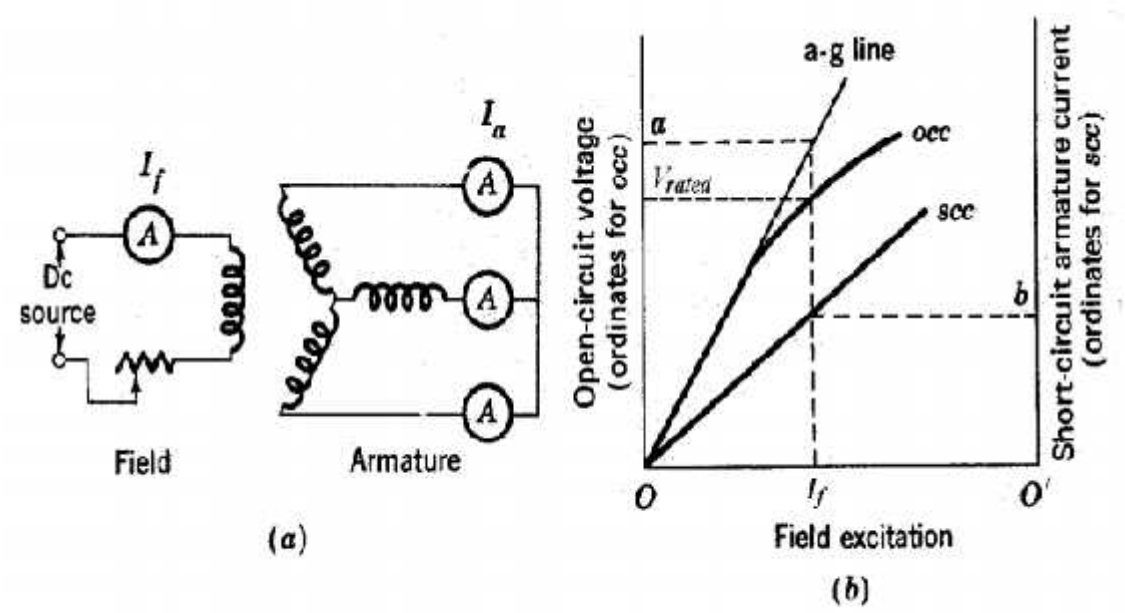
# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
## Lab Manual

**Experiment No: 2**
Study the characteristics of three phase alternator by OCC & SCC test

**Aim:**
**To study the characteristics of three phase alternator by OCC & SCC test**

**Theory:**
**Open Circuit Test** Drive the synchronous machine at the synchronous speed using a prime mover when the stator windings are open circuited. Vary the rotor winding current, and measure stator winding terminal voltage. The relationship between the stator winding terminal voltage and the rotor field current obtained by the open circuit test is known as the open circuit characteristic of the synchronous machine.

**Short Circuit Test** Reduce the field current to a minimum, using the field rheostat, and then open the field supply circuit breaker. Short the stator terminals of the machine together through three ammeters; Close the field circuit breaker; and raise the field current to the value noted in the open circuit test at which the open circuit terminal voltage equals the rated voltage, while maintain the synchronous speed. Record the three stator currents. (This test should be carried out quickly since the stator currents may be greater than the rated value).




(a) Connections for short-circuit test; (b) open- and short-circuit characteristics.

**Procedure:**

1. For open circuit test the machine is driven at its rated speed without load
2. Readings of line-to-line voltage are taken for various values of field current.
3. The OCC is plotted with these reading
4. For short circuit test The three terminals of the armature are short-circuited each through a current measuring circuit
5. The machine is driven at approximately synchronous (rated) speed and measurements of armature short-circuit current are made for various values of field current.
6. The SCC is plotted with the readings

**Observation Table:**

| S.NO | $V_L$ V | $I_f$ A |
|---|---|---|
| 1 | | |
| 2 | | |
| 3 | | |
| 4 | | |
| 5 | | |
| 6 | | |
| 7 | | |

Open circuit test

| S.NO | $I_a$ A | $I_f$ A |
|---|---|---|
| 1 | | |
| 2 | | |
| 3 | | |
| 4 | | |
| 5 | | |
| 6 | | |
| 7 | | |

Short circuit test

**Conclusion:**

**Experiment No: 3**

Perform slip test and determine $X_d$&$X_q$of an alternator

**Aim:**

**To perform slip test and determine $X_d$&$X_q$of an alternator**

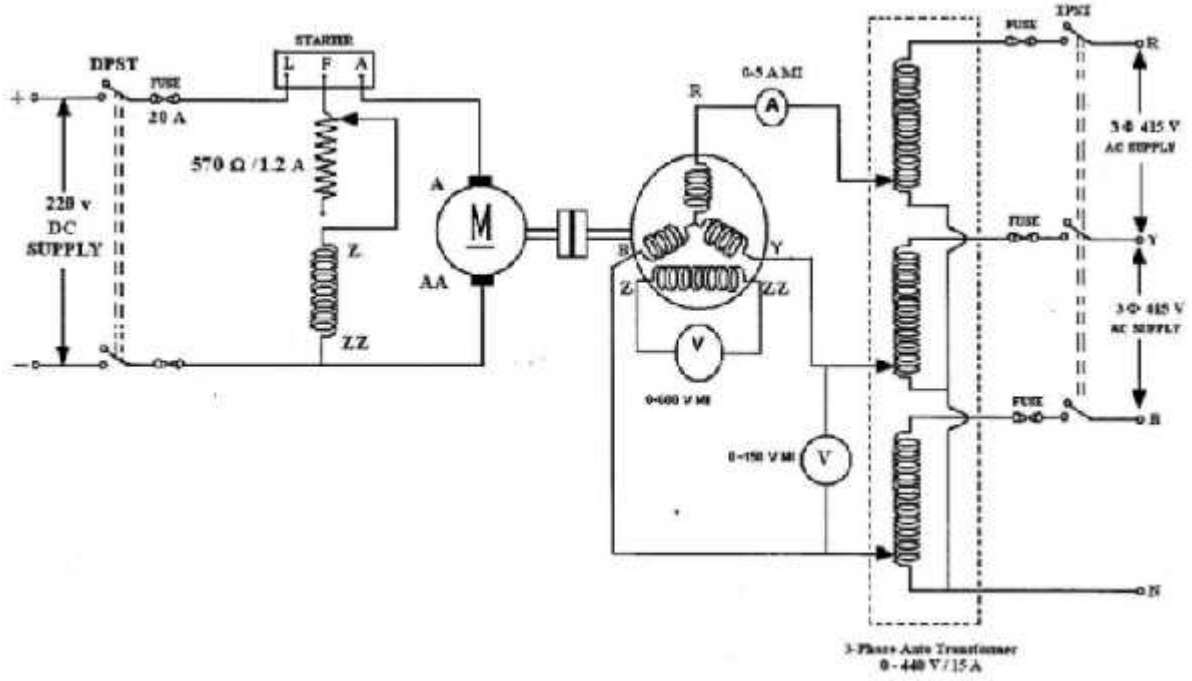
**Theory:**

In a salient pole alternator, the reactance of magnetic circuit along is along its quad stator axis. The alternator is driven by auxiliary prime mover at a speed slightly less than the synchronous speed under these conditions. The armature current is when the armature current mmf is in line with the field poles. The reactance by the magnetic field current is minimum. The ratio of maximum voltage to minimum current gives the direct axis impedance and the ratio of minimum voltage to maximum current gives the armature axis impedance.The values of Xd&Xq are determined by conducting the slip-test. The syn. machine is

driven by a separate prime mover at a speed slightly different from synchronous speed. The field winding is left open and positive sequence balanced voltages of reduced magnitude(around 25%of the rated value)and of rated frequency and impressed across the armature terminals. Here, the relative velocity b/w the field poles and the rotating armature mmf wave is equal to the difference b/w syn. speed and the rotor speed i.e, the slip speed. When the rotor is along the d-axis, then it has a position of min reluctance, min flux linkage and max flux produced links with the winding. Then Xd=(max. armature terminal voltage/ph) /(min. armature current/ph)As the current is small then Vt will be high as drop will be small. When the rotor is alongq-axis, then it is max, then the flux linkage would be max. Then The min flux produced links with winding. So max emf. Xq=(min. armature terminal voltage/ph) /(max. armature current/ph)

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

## Lab Manual




**Procedure:**

1. Note down the name plate details of motor and alternator.
2. Connections are made as per the circuit diagram.
3. Give the supply by closing the DPST switch.
4. Using the three point starter, start the motor to run at the synchronous speed by varying the motor field rheostat at the same time check whether the alternator field has been opened or not.
5. Apply20%to 30% of the rated voltage to the armature of the alternator by adjusting the autotransformer.
6. To obtain the slip and the maximum oscillation of pointers the speed is reduced slightly lesser than the synchronous speed.
7. Maximum current, minimum current, maximum voltage and minimum voltage are noted.
8. Find out the direct and quadrature axis impedances

**Observation Table:**

| S.NO | Vmax | Vmin | Imax | Imin |
|---|---|---|---|---|
| 1 | | | | |

**Conclusion:**

**Experiment No: 4**

Perform no load and block rotor test of single phase induction motor.

**Aim:**

**To perform no load and block rotor test of single phase induction motor**

**Theory:**

No load test:

The test is conducted by rotating the motor without load. The input current, voltage and power are measured by connecting the ammeter, voltmeter and wattmeter in the circuit. These readings are denoted as $V_o$, $I_o$ and $W_o$.

Now

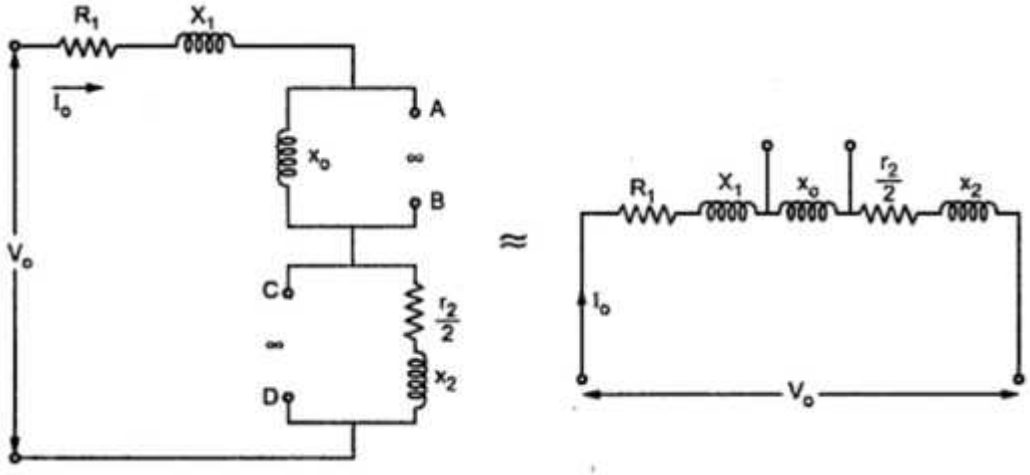
$W_o = V_o I_o \cos\Phi$

$$\therefore \quad \boxed{\cos \phi_0 = \frac{W_o}{V_o I_o} = \text{no load power factor}}$$

The motor speed on no load is almost equal to its synchronous speed hence for practical purposes, the slip can be assumed zero. Hence $r_2/s$ becomes $\infty$ and acts as open circuit in the equivalent circuit. Hence for forward rotor circuit, the branch $r_2/s + j\, x_2$ gets eliminated.

While for a backward rotor circuit, the term $r_2/(2 - s)$ tends to $r_2/2$. Thus $x_o$ is much higher then the impedance $r_2/2 + j\, x_2$. Hence it can be assumed that no current can flow through and that branch can be eliminated.

So circuit reduces to as shown in the Fig.1.




Now the voltage across $x_o$ is $V_{AB}$

$$\therefore \quad \bar{V}_{AB} = \bar{V}_o - \bar{I}_o \times \left[\left(R_1 + \frac{r_2}{2}\right) + j\,(x_1 + x_2)\right]$$

But $V_{AB} = I_o x_o$

$\therefore$ $x_o = V_{AB}/I_o$

But $x_o = X_o/2$

$$\therefore \quad \boxed{X_o = 2x_o = \frac{2\,V_{AB}}{I_o}}$$

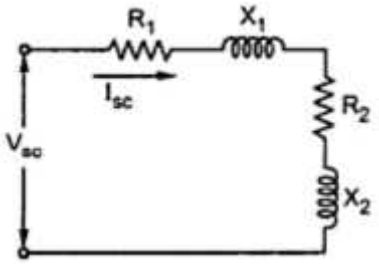
Thus magnetising reactance $X_o$ can be determined.

The no load power $W_o$ is nothing but the rotational losses.

Block Rotor Test:

In balanced rotor test, the rotor is held fixed so that it will not rotate. A reduced voltage is applied to limit the short circuit current. This voltage is adjusted with the help of autotransformer so that the rated current flows through main winding. The input voltage, current and power are measured by connecting voltmeter, ammeter and wattmeter respectively. These readings are denoted as $V_{sc}$, $I_{sc}$ and $W_{sc}$.

Now as rotor is blocked, the slip s = 1 hence the magnetising reactance $x_o$ is much higher than the rotor impedance and hence it can be neglected as connected in parallel with the rotor. Thus the equivalent circuit for blocked rotor test is as shown in the Fig.2.




$W_{sc} = V_{sc} I_{sc} \cos\Phi sc$

$\cos\Phi sc = W_{sc} / V_{sc} I_{sc}$

= blocked rotor power factor

Now $Z_{eq} = V_{sc} / Isc$

$R_{eq} = W_{sc} / (Isc)^2$

But $R_{eq} = R_1 + R_2$

$\therefore$ $R_2 = R_{eq} - R_1$

= rotor resistance referred to stator

$X_{eq} = \sqrt{(Z_{eq}^2 - R_{eq}^2)}$

$X_1 = X_2$ we get,

$$X_2 = \frac{X_{eq}}{2} = \text{rotor reactance referred to stator}$$

The stator resistance is measured by voltmeter-ammeter method, by disconnecting the auxiliary winding and capacitors present if any. Due to skin effect, the a.c. resistance is 1.2 to 1.5 times more than the d.c. resistance.

Key point : Thus with two tests, all the parameters of single phase induction motor can be obtained.

**Procedure:**

1. Note down the name plate details of the motor
2. Connections are made as per the circuit diagram
3. Note down the readings of $V_o$ , $I_o$and $W_o$while performing no load test
4. Note down the readings of $V_{sc}$ ,$I_{sc}$and $W_{sc}$while performing blocked rotor test
5. Do necessary calculations to find out the parameters of equivalent circuit of single phase induction motor

**Observation Table:**

| $V_o$ | $I_o$ | $W_o$ | $V_{sc}$ | $I_{sc}$ | $W_{sc}$ |
|---|---|---|---|---|---|
| | | | | | |

**Conclusion:**

**Experiment No: 5**
Study V curve of Synchronous Motor.

**Aim:**
**To Study V curve of Synchronous Motor**

**Theory:**

The variation of field current affects the power factor at which the synchronous motor operates. For

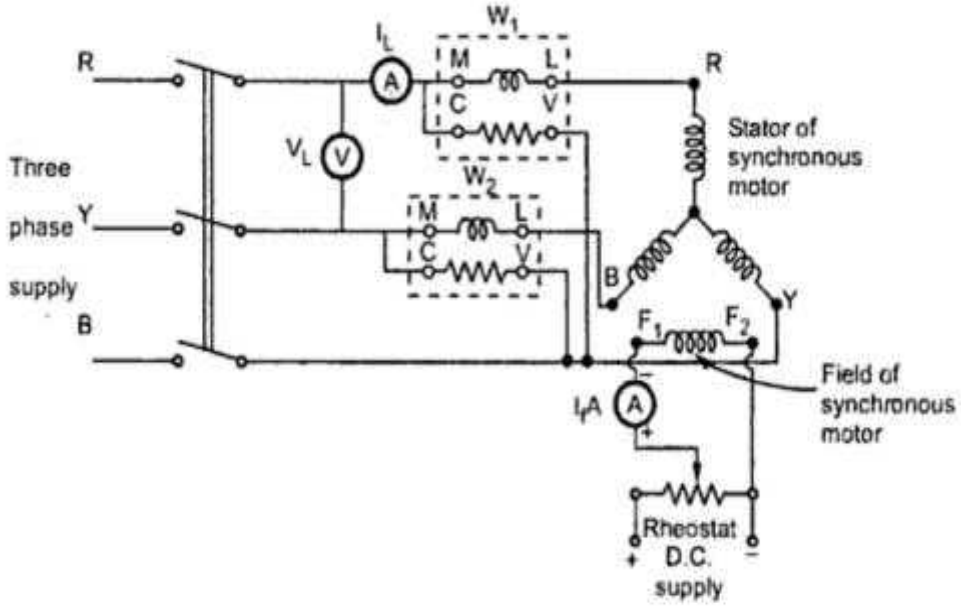
excitation. When the syn. Motor is operated at constant load with variable field excitation, it is observed that:

a) When the excitation is low, the armature current is lag in nature &the magnitude is comparatively high.

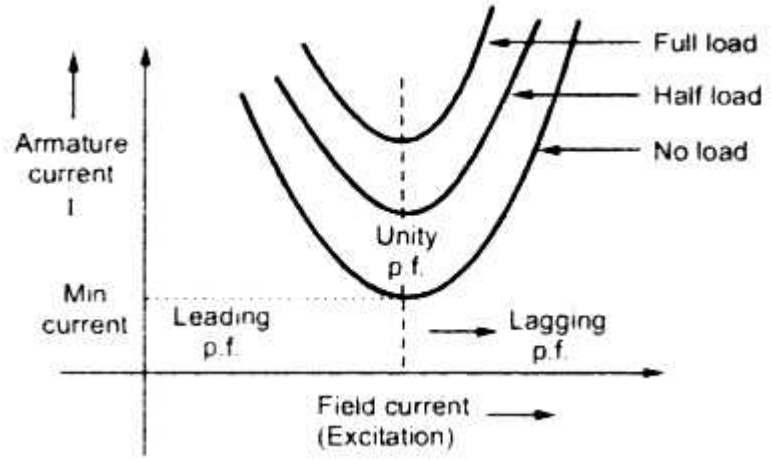
b) If the excitation is gradually increased, the magnitude of Ia is gradually decreasing and the angle of lag is gradually reduced.

c) At one particular excitation, the magnitude of Ia corresponding to that load in minimum and vector will be in phase with V vector.

d) If the excitation is further increased, the magnitude of Ia again gradually increased and Ia, vector goes to leading state and the angle of load is also gradually increased




The plot of v curve :




**Procedure:**

6. Note down the name plate details of the motor
7. Connections are made as per the circuit diagram.
8. The motor starts as an induction motor.
9. Give the excitation to the field for making it to run as the synchronous motor
10. By varying the field rheostat note down the excitation current, armature current and the power factor for various values of excitation.
11. The same process has to be repeated for loaded condition.
12. Later the motor is switched off and the graph is drawn

**Observation Table:**

| S.NO | $I_f$ Amps | $I_a$ Amps |
|---|---|---|
| 1 | | |
| 2 | | |
| 3 | | |
| 4 | | |
| 5 | | |
| 6 | | |
| 7 | | |

**Conclusion:**

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

## Lab Manual

**Title of Course: Power System-I Lab**
**Course Code: EE592**
**L-T-P scheme: 0-0-3** **Course Credit: 2**

**Objectives:**

1. To learn how to handle various electrical equipment to perform experiments of electrical background.
2. To provide an understanding of safety measures necessary to take while nurture electrical equipment of different voltage and current level.
3. To provide a window to investigate and verify various laws, theories, and concepts regarding power system analysis.

**Learning Outcomes:** The students will have a detailed knowledge of electrical equipment handling and will get to be comfortable with various safety measures and caution which is of outmost importance to be taken while implementing electrical equipments of different voltage and current level practically. The students will also get the opportunity & better understanding of various concepts, laws, & theories applicable regarding power system by investigating and verifying them practically. Upon the completion of Operating Systems practical course, the student will be able to:

- **Understand** and will be able to handle various electrical equipment to perform experiments, and as well as to design practically if required.
- **Use** of different safety precautions for experiment or practical purposes.
- **Analyze** designed circuit to see weather various laws, theories, and concepts regarding power system holds or not.

**Course Contents:**

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

Exercise No.1: Study of Ferranti Effect experimentally.
Exercise No. 2: Evaluation of ABCD parameters by Short and open circuit test of long transmission line experimentally.
Exercise No. 3: Evaluation of ABCD parameters by Short and open circuit test of medium transmission line experimentally.
Exercise No. 4: Evaluation of ABCD parameters by Short and open circuit test of short transmission line experimentally.
Exercise No. 5: Transformer Oil Testing

**Text Book:**

1. Electrical Power System, Subir Roy, Prentice Hall

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

## Lab Manual

**Aim: Study and verification of Ferranti Effect on a practical circuit.**

**Description:**

**Theory**

A long transmission line has a large capacitance. If such a line is open circuited or very lightly loaded at the receiving end, the magnitude of the voltage at the receiving end becomes higher than the voltage at the sending end. This phenomenon is called Ferranti effect. Ferranti effect is due to charging current of the line.

The voltage magnification in a long transmission line considering the a nominal π-model due to Ferranti effect can be expressed as:

$$V_s - V_r = -\left(\frac{4\pi^2}{18}\times 10^{-6}\right) f^2 S^2 V_r$$

Here, Vs = sending end voltage

Vr = receiving end voltage

f = frequency of the line

S = electrical length of the line

In this equation we can see that (Vs-Vr) is negative. That is Vr>Vs and Ferranti effect depends on frequency and the electrical length of the line.

**CIRCUIT DIAGRAM:**

Draw the circuit diagram of long transmission line considering it as a nominal π-model as instructed in the lab.

**CALCULATIONS:**

Calculate the theoretical data's got from experiment of the given circuit.

**OBSERVATION TABLE:**

| SL. No. | Variation of either S or f | (Vs – Vr) |
|---|---|---|
| 1. | | |
| 2. | | |
| 3. | | |
| 4. | | |

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

**RESULT:**

The percentage error is found to be__%.

**DISCUSSION:**

**Experiment No: 2. Evaluation of ABCD parameters by Short and open circuit test of long transmission line experimentally.**

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

## Lab Manual

**Aim: Evaluation of ABCD parameters by Short and open circuit test of long transmission line.**

**Description:**

**Theory**

The predominance of one or more of the parameters of a line is governed by its length and conductor configuration. The term long line refers to a line having its length more than 240km. The line parameters like resistance (R), inductance (L), Capacitance (C) and leakage conductance (G) are distributed uniformly along the whole length of the line. It may be assumed that a line consists of a large number of short sections connected together.
The steady state values of voltage and current at any intermitted point distance s from the receiving end for long transmission line can be written as:

$$V = \frac{1}{2}(V + Z_0 I_r)e^{\gamma} + \frac{1}{2}(V - Z_0 I_r)e^{-\gamma}$$

$$I = \frac{1}{2}\left(I_r + \frac{V_r}{Z_0}\right)e^{\gamma} + \frac{1}{2}(I_r - \frac{V_r}{Z_0})e^{-\gamma}$$

Here, Vs = sending end voltage
Vr = receiving end voltage
Is = sending end current
Ir = receiving end current
f = frequency of the line
S = electrical length of the line

$Z_0 = \sqrt{\frac{Z}{Y}}$

Using complex exponential or power series and with the basic knowledge of ABCD parameters we can find the ABCD parameters of long transmission line. Which looks like:

$A = D = \cosh \gamma s = 1 + ZY/2$

$B = Z_0 \sinh \gamma s = Z(1 + ZY/6)$

$C = \frac{1}{Z_0} \sinh \gamma s = Y(1 + ZY/6)$

**CIRCUIT DIAGRAM:**

Draw the circuit diagram of long transmission line as instructed in the lab.

**CALCULATIONS:**

Calculate the theoretical data's got from experiment of the given circuit and find out values of parameters ABCD.

**OBSERVATION TABLE:**

| Sl. No. | When Output is open circuited (i.e. $I_r$ =0) | | | When Output is short circuited (i.e. $V_r$ =0) | | |
|---|---|---|---|---|---|---|
| | $V_s$ | $I_s$ | $V_r$ | $V_s$ | $I_s$ | $I_r$ |
| 1. | | | | | | |
| 2. | | | | | | |
| 3. | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 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: 3. Evaluation of ABCD parameters by Short and open circuit test of medium transmission line experimentally.**

**Aim: Evaluation of ABCD parameters by Short and open circuit test of medium transmission line.**

**Description:**

**Theory**

The predominance of one or more of the parameters of a line is governed by its length and conductor configuration. The term medium line refers to a line having its length in the range of 80-240km.For such lines the capacitance (C) of the line cannot be neglected and t is considered to be lumped at one or more points of the line. The effect of capacitance is more at higher frequency. The leakage conductance (G) is neglected.

A number of localized capacitance models have been used to make approximate line performance calculations. There are two common models:

Nominal T model

The steady state values of sending end voltage and current can be written as:

$$V = \left(1 + \frac{Z}{2}\right)V + Z\left(1 + \frac{Z}{4}\right)I$$

$$I = Y + \left(1 + \frac{Z}{2}\right)I$$

Here, Vs = sending end voltage
Vr = receiving end voltage
Is = sending end current
Ir = receiving end current
Z = series impedance of the line
Y = series admittance of the line

With the basic knowledge of ABCD parameters we can write the ABCD parameters of this model of medium transmission line as following.

A = D = $\left(1 + \frac{Z}{2}\right)$

B = $Z\left(1 + \frac{Z}{4}\right)$

C = Y

Nominal π model

The steady state values of sending end voltage and current can be written as:

$$V = \left(1 + \frac{Z}{2}\right)V + Z$$

$$I = Y\left(1 + \frac{Z}{4}\right)V + \left(1 + \frac{Z}{2}\right)I$$

With the basic knowledge of ABCD parameters we can write the ABCD parameters of this model of medium transmission line as following.

A = D = $\left(1 + \frac{Z}{2}\right)$

B = Z

C = Y $\left(1 + \frac{Z}{4}\right)$

**CIRCUIT DIAGRAM:**

Draw the circuit diagram of medium transmission line as instructed in the lab.

Calculate the theoretical data's got from experiment of the given circuit and find out values of parameters ABCD.

**OBSERVATION TABLE:**

| Sl. No. | When Output is open circuited (i.e. $I_r$ =0) | | | When Output is short circuited (i.e. $V_r$ =0) | | |
|---|---|---|---|---|---|---|
| | $V_s$ | $I_s$ | $V_r$ | $V_s$ | $I_s$ | $I_r$ |
| 1. | | | | | | |
| 2. | | | | | | |
| 3. | | | | | | |
| 4. | | | | | | |
| 5. | | | | | | |

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

**RESULT:**

The percentage error is found to be__%.

**DISCUSSION:**

**Experiment No: 4. Evaluation of ABCD parameters by Short and open circuit test of short transmission line experimentally.**

**Aim: Evaluation of ABCD parameters by Short and open circuit test of short transmission**

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
## Lab Manual

**Description:**

**Theory**

The predominance of one or more of the parameters of a line is governed by its length and conductor configuration. The term short line refers to a line having its length up to 80km. For such lines the capacitance (C) is negligibly small but for cable line where the distance between the conductors is small, the effect of capacitance cannot be ignored. In a short line the shunt capacitance C and shunt conductance G are neglected. The series resistance R and series inductance L for the total length of the time is considered.

The steady state values of sending end voltage and current can be written as:

$$Vs = Vr + Z\,Ir$$
$$Is = Ir$$

Here, Vs = sending end voltage
Vr = receiving end voltage
Is = sending end current
Ir = receiving end current
Z = series impedance of the line

With the basic knowledge of ABCD parameters we can write the ABCD parameters of this model of medium transmission line as following.
A = 1
B = Z
C = 0
D = 1

**CIRCUIT DIAGRAM:**

Draw the circuit diagram of short transmission line as instructed in the lab.

**CALCULATIONS:**

Calculate the theoretical data's got from experiment of the given circuit and find out values of parameters ABCD.

**OBSERVATION TABLE:**

| Sl. No. | When Output is open circuited (i.e. $I_r$ =0) | | | When Output is short circuited (i.e. $V_r$ =0) | | |
|---|---|---|---|---|---|---|
| | $V_s$ | $I_s$ | $V_r$ | $V_s$ | $I_s$ | $I_r$ |
| 1. | | | | | | |
| 2. | | | | | | |
| 3. | | | | | | |
| 4. | | | | | | |
| 5. | | | | | | |

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

The percentage error is found to be__%.

**DISCUSSION:**

**Experiment No: 5. Transformer Oil Testing**
**Aim: Study Transformer Oil Testing**

**Description:**

**Theory**

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

## Lab Manual

Oil, a type of insulating and cooling oil used in transformers and other electrical equipment, needs to be tested periodically to ensure that it is still fit for purpose. This is because it tends to deteriorate over time. Testing sequences and procedures are defined by various international standards, many of them set by ASTM. Testing consists of measuring breakdown voltage and other physical and chemical properties of samples of the oil, either in a laboratory or using portable test equipment on site.

The transformer oil (insulation oil) of voltage- and current-transformers fulfills the purpose of insulating as well as cooling. Thus, the dielectric quality of transformer oil is essential to secure operation of a transformer.

As transformer oil deteriorates through aging and moisture ingress, transformer oil should, depending on economics, transformer duty and other factors, be tested periodically. Power utility companies have a vested interest in periodic oil testing since transformers represent a large proportion of their total assets. Through such testing, transformers' life can be substantially increased, thus delaying new investment of replacement transformer assets.

Recently time-consuming testing procedures in test labs have been replaced by on-site oil testing procedures. There are various manufacturers of portable oil testers. With low weight devices in the range of 20 to 40 kg, tests up to 100 kV rms can be performed and reported on-site automatically. Some of them are even battery-powered and come with accessories.

To assess the insulating property of dielectric transformer oil, a sample of the transformer oil is taken and its breakdown voltage is measured. The lower the resulting breakdown voltage, the poorer the quality of the transformer oil.

**CIRCUIT DIAGRAM:**

Draw the equivalent line as instructed in the lab.

**PROCEDURE**

**OBSERVATIONS:**

Two standard-compliant test electrodes with a typical clearance of 2.5 mm are surrounded by the dielectric oil. A test voltage is applied to the electrodes and is continuously increased up to the breakdown voltage with a constant, standard-compliant slew rate of e.g. 2 kV/s.

Hence the breakdown voltage of the oil as measured is _____KV

**DISCUSSION:**

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

## Lab Manual

**Title of Course: Control System-I Lab**
**Course Code: EE593**
**L-T-P scheme: 0-0-3** **Course Credit: 2**

**Objectives:**

**1.** To provide the students with a hands-on experience on the theoreticalconcepts through simple experiments.
**2.** To develop the ability to designand validate their knowledge through open ended experiments.

**Learning Outcomes:**

**On successful completion of this lab course, the students would beable to**

1. Demonstrate and analyze the response of Transfer function for various input.
2. Analyze the response of various signal like Impulse Ramp etc.
3. Carry out the root locus of given signal.
4. Analyse different plot and state model.
5. Conduct an open ended experiment in a group of2 to 3.

**Course Contents:**

**List of Experiments:**

1. To obtain a transfer function from given poles and zeroes using MATLAB
2. To obtain zeros and poles from a given transfer function using MATLAB
3. To obtain the step response of a transfer function of the given system using MATLAB
4. To obtain the impulse response of a transfer function of the given system using MATLAB
5. To obtain the ramp response of a transfer function of the given system using MATLAB.
6. To plot the root locus for a given transfer function of the system using MATLAB.
7. To obtain bode plot for a given transfer function of the system using MATLAB.
8. To obtain the transfer function from the state model.
9. To obtain the state model from the given transfer function.
10. To design a lag compensator for a closed loop system.

**Text Book:**

**1) Katsuhiko Ogata, (2002), Modern Control Engineering, Prentice Hall of India Private Ltd., New Delhi.**
**2) Nagrath I.J. and Gopal M., (2006), Control Systems Engineering, New Age International Publisher, New Delhi.**

**Recommended Systems/Software Requirements:**

SCILAB, MATLAB

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
# Lab Manual

## 1 .TRANSFER FUNCTION FROM ZEROS AND POLES

**AIM:** To obtain a transfer function from given poles and zeroes using MATLAB

**APPARATUS:**
Software: MATLAB

THEORY: A transfer function is also known as the network function is a mathematical representation, in terms of spatial or temporal frequency, of the relation between the input and output of a (linear time invariant) system. The transfer function is the ratio of the output Laplace Transform to the input Laplace Transform assuming zero initial conditions. Many important characteristics of dynamic or control systems can be determined from the transfer function. The transfer function is commonly used in the analysis of single-input single-output electronic system, for instance. It is mainly used in signal processing, communication theory, and control theory. The term is often used exclusively to refer to linear time-invariant systems (LTI). In its simplest form for continuous time input signal x(t) and output y(t), the transfer function is the linear mapping of the Laplace transform of the input, X(s), to the output Y(s). Zeros are the value(s) for z where the numerator of the transfer function equals zero. The complex frequencies that make the overall gain of the filter transfer function zero. Poles are the value(s) for z where the denominator of the transfer function equals zero. The complex frequencies that make the overall gain of the filter transfer function infinite. The general procedure to find the transfer function of a linear differential equation from input to output is to take the Laplace Transforms of both sides assuming zero conditions, and to solve for the ratio of the output Laplace over the input Laplace.

**MATLAB PROGRAM:**

```
z=input('enter zeroes')
p=input('enter poles')
k=input('enter gain')
[num,den]=zp2tf(z,p,k)
tf(num,den)
```

**PROCEDURE:**

1. Write MATLAB program in the MATLAB editor document.
2. Then save and run the program
3. Give the required input.
4. The syntax "zp2tf(z,p,k)" and "tf(num,den)" solves the given input poles and zeros and gives the transfer function.
5. zp2tf forms transfer function polynomials from the zeros, poles, and gains of a system in factored form

**EXAMPLE:**

Given poles are -3.2+j7.8,-3.2-j7.8,-4.1+j5.9,-4.1-j5.9,-8 and the zeroes are -0.8+j0.43,-0.8- j0.43,-0.6 with a gain of 0.5

**THEORITICAL CALCULATIONS:**

Enter zeros
Z=

Enter poles
P =

Enter gain
K=

num =

den =

Transfer function=

**RESULT:**

## 2. ZEROS AND POLES FROM TRANSFER FUNCTION

**AIM:**
To obtain zeros and poles from a given transfer function using MATLAB.

**APPARATUS:**
Software: MATLAB

**THEORY**:
The transfer function provides a basis for determining important system response characteristics without solving the complete differential equation. As defined, the transfer function is a rational function in the complex variable that is It is often convenient to factor the polynomials in the numerator and the denominator, and to write the transfer function in terms of those factors: where, the numerator and denominator polynomials, N(s) and D(s), have real coefficients defined by the system's differential equation.

**MATLAB PROGRAM:**

```
num = input('enter the numerator of the transfer function')
den = input('enter the denominator of the transfer function')
[z,p,k] = tf2zp(num,den)
```

**PROCEDURE:**
1.Type the program in the MATLAB editor that is in M-file.
2.Save and run the program.
3.Give the required inputs in the command window of MATLAB in matrix format.
3.tf2zp converts the transfer function filter parameters to pole-zero-gain form.

4.[z,p,k] = tf2zp(b,a) finds the matrix of zeros z, the vector of poles p, and the associated vector of gains k from the transfer function parameters b and a:
5. The numerator polynomials are represented as columns of the matrix b.
6.The denominator polynomial is represented in the vector a.
7.Note down the output of the program that is zeros, poles and gain obtained in MATLAB.
8.The zeros, poles and gain are also obtained theoretically.

**THEORITICAL CALCULATIONS:**
Enter the numerator of the transfer function
num =
Enter the denominator of the transfer function
den =

z =
p =

**RESULT:**

## 3. STEP RESPONSE OF A TRANSFER FUNCTION

**AIM:** To obtain the step response of a transfer function of the given system using MATLAB

**APPARATUS**: Software: MATLAB

**THEORY**: A step signal is a signal whose value changes from one level to another level in zero time.

**MATLAB PROGRAM:**
num = input('enter the numerator of the transfer function')
den = input('enter the denominator of the transfer function')
step (num,den)

**PROCEDURE**:
- Type the program in MATLAB editor that is in M-file.
- Save and run the program.
- Give the required inputs in the command window of MATLAB in matrix format.
- 'step' function calculates the unit step response of a linear system.
- Zero initial state is assumed in state-space case.
- When invoked with no output arguments, this function plots the step response on the screen.
- step (sys) plots the response of an arbitrary LTI system.
- This model can be continuous or discrete, and SISO or MIMO.
- The step response of multi-input systems is the collection of step responses for each

input channel.
- The duration of simulation is determined automatically based on the system poles and zeroes.
- Note down the response of the transfer function obtained in MATLAB.
- The response of the transfer function is also obtained theoretically.
- Both the responses are compared.

**THEORETICAL CALCULATIONS:** Calculation will be in the form of graph.

**RESULT:**

## 4. IMPULSE RESPONSE OF A TRANSFER FUNCTION

**AIM**:To obtain the impulse response of a transfer function of the given system using MATLAB.

**APPARATUS:**
Software: MATLAB

**THEORY:**
An impulse signal is a signal whose value changes from zero to infinity in zero time.

**MATLAB PROGRAM:**

```
num = input('enter the numerator of the transfer function')
den = input('enter the denominator of the transfer function')
impulse(num,den)
```

**PROCEDURE**:
- Type the program in the MATLAB editor that is in M-file.
- Save and run the program.
- Give the required inputs in the command window of MATLAB in matrix format.
- 'impulse' calculates the impulse response of a linear system.
- The impulse response is the response to the Dirac input, δ (t) for continuous time systems and to a unit pulse at for discrete time systems.
- Zero initial state is assumed in the state space case.
- When invoked without left hand arguments, this function plots the impulse response on the screen.
- 'impulse(sys)' plots the impulse response of an arbitrary LTI model sys.
- This model can be continuous or discrete, SISO or MIMO.
- The impulse response of multi-input systems is the collection of impulse responses for each input channel.
- The duration of simulation is determined automatically to display the transient behavior of the response.
- Note down the response of the given transfer function obtained in MATLAB.
- The response of the transfer function is also obtained theoretically.
- Both the responses are compared.

**GRAPH-**

**RESULT:**

## 5. RAMP RESPONSE OF A TRANSFER FUNCTION

**AIM:**
To obtain the ramp response of a transfer function of the given system using MATLAB.

**APPARATUS:**
Software: MATLAB

**THEORY:**
A ramp signal is a signal which changes with time gradually in a linear fashion

**MATLAB PROGRAM:**
num = input('enter the numerator of the transfer function')
den = input('enter the denominator of the transfer function')
lsim(num,den,u,t)

**PROCEDURE:**
- Type the program in the MATLAB editor that is in M-file.
- Save and run the program.
- Give the required inputs in the command window of MATLAB in matrix format.
- lsim simulates the (time) response of continuous or discrete linear systems to arbitrary inputs.
- When invoked without left-hand arguments, lsim plots the response on the screen.
- lsim(sys,u,t) produces a plot of the time response of the LTI model sys to the input time historyt,u.
- The vector t specifies the time samples for the simulation and consists of regularly spaced time samples.
- t = 0:dt:Tfinal
- The matrix u must have as many rows as time samples (length(t)) and as many columns as system inputs.
- Each row u(i,:) specifies the input value(s) at the time sample t(i).
- Note down the response of the transfer function obtained in MATLAB.
- The response of the transfer function is also obtained theoretically.
- Both the responses are compared.

**GRAPH:**

**RESULT:**

## 6.ROOT LOCUS FROM A TRANSFER FUNCTION

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
## Lab Manual

**AIM:**To plot the root locus for a given transfer function of the system using MATLAB.
APPARATUS:

Software: MATLAB

**THEORY:**
rlocus computes the Evans root locus of a SISO open-loop model. The root locus gives the closed-loop pole trajectories as a function of the feedback gain k (assuming negative feedback).
Root loci are used to study the effects of varying feedback gains on closed-loop pole locations.
In turn, these locations provide indirect information on the time and frequency responses.
rlocus(sys) calculates and plots the rootlocus of the open-loop SISO model sys. This function can be applied to any of the following feedback loops by setting sys appropriately.

**MATLAB PROGRAM:**
num=input('enter the numerator of the transfer function')
den=input('enter the denominator of the transfer function')
h=tf(num,den)
rlocus(h)

**PROCEDURE:**
- Write MATLAB program in the MATLAB specified documents.
- Then save the program to run it.

27
- The input is to be mentioned.
- The syntax "h=tf(num,den)" gives the transfer function and is represented as h.
- The syntax "rlocus(h)" plots the rootlocus of the transfer function h.
- Generally the syntax is of the form

rlocus(sys)
rlocus(sys,k)
rlocus(sys1, sys2, ….)
[r,k] = rlocus(sys)
r = rlocus(sys,k)

- rlocus(sys) calculates and plots the root locus of the open loop SISO model sys.
- Now we have to solve it theoretically.
- Now we have to compare the practical and theoretical ouputs to verify each other correctly.

THEORETICAL CALCULATIONS:
enter the numerator of the transfer function
num=
enter the denominator of the transfer function
den=

Transfer function :

**RESULT:**

## 7.BODE PLOT FROM A TRANSFER FUNCTION

**AIM**: To obtain bode plot for a givan transfer function of the system using MATLAB.

**APPARATUS:**
Software: MATLAB

**THEORY:**
Bode computes the magnitude and phase of the frequency response of LTI models. When invoked without left-side arguments, bode produces a Bode plot on the screen. The magnitude is plotted in decibels (dB), and the phase in degrees. The decibel calculation for mag is computed as 20log10(|H(jw)|), where H(jw) is the system's frequency response. Bode plots are used to analyze system properties such as the gain margin, phase margin, DC gain, bandwidth, disturbance rejection, and stability

bode(sys) plots the Bode response of an arbitrary LTI model sys. This model can be continuous or discrete, and SISO or MIMO. In the MIMO case, bode produces an array of Bode plots, each plot showing the Bode response of one particular I/O channel. The frequency range is determined automatically based on the system poles and zeros.
bode(sys,w) explicitly specifies the frequency range or frequency points to be used for the plot.
To focus on a particular frequency interval [wmin,wmax], set w = {wmin,wmax}. To use particular frequency points, set w to the vector of desired frequencies. Use logspace to generatelogarithmically spaced frequency vectors. All frequencies should be specified in radians/sec.
bode(sys1,sys2,...,sysN) or bode(sys1,sys2,...,sysN,w) plots the Bode responses of several LTI models on a single figure. All systems must have the same number of inputs and outputs, but may otherwise be a mix of continuous and discrete systems. This syntax is useful to compare the Bode responses of multiple systems.
bode(sys1,'PlotStyle1',...,sysN,'PlotStyleN') specifies which color, linestyle, and/or marker should be used to plot each system. For example,
bode(sys1,'r--',sys2,'gx') uses red dashed lines for the first system sys1 and green 'x' markers for the second system sys2.
When invoked with left-side arguments
[mag,phase,w] = bode(sys)
[mag,phase] = bode(sys,w)
return the magnitude and phase (in degrees) of the frequency response at the frequencies w (in rad/sec). The outputs mag and phase are 3-D arrays with the frequency as the last dimension (see "Arguments" below for details). You can convert the magnitude to decibels bymagdb = 20*log10(mag)

**MATLAB PROGRAM:**

```
num=input('enter the numerator of the transfer function')
den=input('enter the denominator of the transfer function')
h=tf(num,den)
[gm pm wcpwcg]=margin(h)
```

bode(h)

**PROCEDURE:**

- Write the MATLAB program in the MATLAB editor.
- Then save and run the program.
- Give the required inputs.
- The syntax "bode(h)" solves the given input transfer function and gives the bode plot,
- wherenum,den are the numerator and denominator of the transfer function.
- Now plot the bode plot theoretically for the given transfer function and compare it with theplot obtained practically.

**THEORETICAL CALCULATIONS:**

enter the numerator of the transfer function
num =
enter the denominator of the transfer function
den =
Transfer function:
gm =
pm =
wcp =
wcg =

**RESULT:**

## 8.TRANSFER FUNCTION FROM STATE MODEL

**AIM:** To obtain the transfer function from the state model.

**APPARATUS:**

Software: MATLAB

**THEORY:**

The transfer function is defined as the ratio of Laplace transform of output to Laplace transform of input. A state space representation is a mathematical model of a physical system as a set of input, output and state variables related by first-order differential equations. The state space representation (also known as the "time-domain approach") provides a convenient and compact way to model and analyze systems with multiple inputs and outputs.

Unlike the frequency domain approach, the use of the state space representation is not limited to systems with linear components and zero initial conditions.

"State space" refers to the space whose axes are the state variables. The state of the system can be represented as a vector within that space.

**MATLAB PROGRAM:**

A =input('enter the matrix A')
B= input('enter the matrix B')
C = input('enter the matrix C')
D= input('enter the matrix D')
Sys =ss2tf(A,B,C,D)

**EXAMPLE:**
Obtain the transfer function from the State Model given below:
A=
B=
C=
D=

**PROCEDURE:**
- Type the program in the MATLAB editor that is in M-file.
- Save and run the program.
- Give the required inputs in the command window of MATLAB in matrix format.
- The command ss2tf(A,B,C,D)) converts the given transfer function into a state model.
- Note down the output obtained in MATLAB.
- The Transfer Function is also obtained theoretically.
- Both the state models are compared.

**RESULT:**

## 9.STATE MODEL FROM TRANSFER FUNCTION

**AIM:**
To obtain the state model from the given transfer function.

**APPARATUS:**
Software: MATLAB

**THEORY:**
There are three methods for obtaining state model from transfer function:
1. Phase variable method
2. Physical variable method
3. Canonical variable method

Out of three methods given above canonical form is probably the most straightforward method
for converting from the transfer function of a system to a state space model is to generate a model in "controllable canonical form." This term comes from Control Theory but its exact meaning is not important to us. To see how this method of generating a state space model works, consider the third order differential transfer function

**MATLAB PROGRAM:**
num=input('enter the numerator of the transfer function')
den=input('enter the denominator of the transfer function')
ss(tf(num,den))

**PROCEDURE:**
- Type the program in the MATLAB editor that is in M-file.
- Save and run the program.
- Give the required inputs in the command window of MATLAB in matrix format.
- The command ss(tf(num,den)) converts the given transfer function into a state model.
- Note down the output obtained in MATLAB.
- The state model is also obtained theoretically.

- Both the state models are compared.

**RESULT:**

## 10.STATE MODEL FROM ZEROS AND POLES

**AIM**: To obtain a state model from given poles and zeros using MATLAB.

**APPARATUS:**
Software: MATLAB

**THEORY:**
Let's say we have a transfer function defined as a ratio of two polynomials:
H(s)=
Where N(s) and D(s) are simple polynomials.
Zeroes are the roots of N(s) (the numerator of the transfer function)obtained by setting N(s)=0
and solving for s. Poles are the roots of D(s) (the denominator of the transfer function),obtained
by setting D(s)=0 and solving for s.
The state space model represents a physical system as n first order coupled differential equations.
This form is better suited for computer simulation than an n
th
order input-output differential
equation.
The general vector-matrix form of state space model is:
Where,
X = state vector
U = input vector
A = n x n matrix
B = n x 1 matrix
The output equation for the above system is,
42

MATLAB PROGRAM:

```
z=input('enter zeros')
p=input('enter poles')
k=input('enter gain')
[A,B,C,D]=zp2ss(z,p,k)
```

PROCEDURE:

- Open the MATLAB window and open a new MATLAB editor.
- Write the MATLAB program in the MATLAB editor.
- Save and run the MATLAB program.
- Enter the given poles, zeros and gain as input in matrix format.
- The syntax "[A,B,C,D]=zp2ss(z,p,k)" solves zeroes, poles and gain given in the matrix format as input and gives the output in the form of a state model.
- This syntax transforms the given zeros, poles and gain into a state model.

- Note down the output state model obtained practically by using the syntax "[A,B,C,D]=zp2ss(z,p,k)" .
- Now find the state model theoretically for the given poles, zeros and gain.
- Compare the theoretically obtained state model from the given poles, zeros and gain with the one obtained practically. Write the result based on the comparison betweenthoretical and practical result.

EXAMPLE:
zeros are:
poles are:
gain=

RESULT:

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

## Lab Manual

**Title of Course: Advanced OOPs using C++ Lab**
**Course Code: EE594A**
**L-T-P scheme: 0-0-3** **Course Credit: 2**

**Objectives:**
The course presents C++ programming including: advanced C++ environment, exception handling, conception of different file handling, template, STL that aims to:

- Be able to code using more advanced C++ features such as class, objects, operator overloads, dynamic memory allocation, inheritance and polymorphism, exception handling, etc.
- Be able to build class template, function template and also they will able to know how practically STL are works.
- Be able to understand practically different string operations and different file operations, like text file, binary file.

**Learning Outcomes:**

- Be able to develop different types of computer programs using C++.
- Understand exception handling mechanism and different file (text, binary) operations.
- Understand the usage of template: class template & function template and STL.
- Be able to do different operations on string in C++ programming.

**Course Contents:**
**Exercises that must be done in this course are listed below:**
**Exercise No.1:** Introduction, Basics of C++, Inline function, friend function, function and overloading, inheritance
**Exercise No. 2:** Exception Handing: throwing, catching, rethrowing mechanism; Multiple catch statement
**Exercise No. 3:** Template: Class template, Function template
**Exercise No. 4:** Console I/O operations: C++ streams; C++ stream classes; Unformatted I/O operations; Formatted I/O operations; Managing output with Manipulators.
**Exercise No. 5:** Working with Files: Text File: Basic file operations on text file: Creating/Writing text into file; Binary File: Creation of file, writing data into file, searching.
**Exercise No. 6:**Standard Template Library: Components of STL; Containers, Iterator; Applications of container classes.
**Exercise No. 7:**String Manipulation: The String class; Creating String object; Manipulating strings; Relational operations on strings; String comparison characteristics.

**Text Books:**

1. Schildt, H., The Complete Reference C++, Tata McGraw Hill Education Pvt. Ltd.
2. E.Balagurusamy; Object Oriented programming with C++; Tata McGraw Hill Education Pvt. Ltd.


**Reference Books:**

3. Debasish Jana, C++ object oriented programming paradigm, PHI.
4. D. Ravichandran, Programming with C++, Tata McGraw Hill Education Pvt. Ltd.
5. Y.I. Shah and M.H. Thaker, Programming In C++, ISTE/EXCEL BOOKS.


**Recommended Systems/Software Requirements:**
**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++compiler in Windows XP/7 or Linux Operating System.

# UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR
## Lab Manual

**Experiment No 1:Inline Function**

**Aim: Write a C++ program to find the largest of three numbers using inline function.**

**Description:**
Inline function is one of the important feature of C++. When the program executes the function call instruction the CPU stores the memory address of the instruction following the function call, copies the arguments of the function on the stack and finally transfers control to the specified function. The CPU then executes the function code, stores the function return value in a predefined memory location/register and returns control to the calling function.C++ provides an inline functions to reduce the function call overhead. Inline function is a function that is expanded in line when it is called. When the inline function is called whole code of the inline function gets inserted or substituted at the point of inline function call. This substitution is performed by the C++ compiler at compile time.

**Algorithm:**
Step 1: Start the program.
Step 2: Declare and define the function largest() as inline.
Step 3: Compare with other variables.
Step 4: Return largest number.
Step 5: Stop the program.

**/* Program */**

```
#include<iostream.h>
inline int largest(int&a,int&b,int&c)
{
int big=0;
 if(a>b)
big=a;
 else
big=b;
 if(c>big)
 big=c;
 return big;
}
int main()
{
inta,b,c;
cout<<"Enter Three Numbers To Find The Largest "<<endl;
cout<<"a = ";
cin>>a;
cout<<"\nb = ";
cin>>b;
cout<<"\nc = ";
cin>>c;
int large=largest(a,b,c);
cout<<"\n Largest of "<<a<<","<<b<<" and "<<c<<" is "<<large;
getch();
 return(0);
}
```

**INPUT 1:**

```
Enter Three Numbers To Find The Largest
a = 24
b = 45
c = 23
```

**OUTPUT 1:**
Largest of 24,45 and 23 is 45

**INPUT 2:**
Enter Three Numbers To Find The Largest
a = 22
b = 34
c = 56

**OUTPUT2:**
Largest of 22,34 and 56 is 56

**Experiment No. 2:Concept of Class and Object**
**Aim: Create a class called 'EMPLOYEE' that has**
**- EMPCODE and EMPNAME as data members**
**- member function getdata( ) to input data**
**- member function display( ) to output data**
**Write a main function to create EMP, an array of EMPLOYEE objects. Accept anddisplay the details of at least 6 employees.**

**Description:**
A class is used to specify the form of an object and it combines data representation and methods for manipulating that data into one neat package. The data and functions within a class are called members of the class.A class definition starts with the keyword class followed by the class name; and the class body, enclosed by a pair of curly braces. A class definition must be followed either by a semicolon or a list of declarations.The keyword public determines the access attributes of the members of the class that follow it. A public member can be accessed from outside the class anywhere within the scope of the class object. You can also specify the members of a class as private or protected which we will discuss in a sub-section.A class provides the blueprints for objects, so basically an object is created from a class. We declare objects of a class with exactly the same sort of declaration that we declare variables of basic types.
Here "EMPLOYEE" is the class, EMPCODE and EMPNAME are the data member. Here getdata() function is used to get input and display() is to show output.

**Algorithm:**

STEP 1: Start the program.
STEP 2: Declare the class Employee.
STEP 3: empcode and empname are the data members
STEP 4: Declare and define getdata() function to take input from user.
STEP 5: Display() function shows the output.
STEP 6: Array of objects of the class Employee is declared in main() function.
STEP 7: By using Emp[i], access the class members.
STEP 10: Stop the program.

**/* Program */**

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
class Employee
{
 private: intempcode;
 char empname[10];
 public: void getdata();
 void display();
```

```
};
void Employee::getdata()
{
cout<<"\nNAME :";
cin>>empname;
cout<<"\nCODE :";
cin>>empcode;
}
void Employee::display()
{
cout<<endl<<setw(20)<<empname<<setw(10)<<empcode;
}
int main()
{
 Employee Emp[6];
clrscr();
cout<< "Enter employee details:\n ";
 for(inti=0;i<6;i++)
 {
cout<<"\nemployee "<<i+1<<endl;
Emp[i].getdata();
 }
cout<<"\nEmployee details are as follows :";
 cout<<"\n\n"<<setw(20)<<"NAME"<<setw(10)<<setiosflags(ios::right)<<"CODE";
cout<<"\n------------------------------";
 for(i=0;i<6;i++)
Emp[i].display();
getch();
 return(0);
}
```

**INPUT 1:**

```
Enter employee details:
employee 1
NAME :ashok
CODE :111
employee 2
NAME :annapurna
CODE :112
employee 3
NAME :anupama
CODE :113
employee 4
NAME :anuradha
CODE :114
employee 5
NAME :ashraya
CODE :115
employee 6
NAME :akash
CODE :116
```

**OUTPUT 1:**

Employee details are as follows :

| NAME | CODE |
|---|---|
| Ashok | 111 |
| Annapurna | 112 |
| anupama | 113 |
| anuradha | 114 |
| ashraya | 115 |
| akash | 116 |

**INPUT 2:**

Enter employee details:
employee 1
NAME :ram
CODE :111
employee 2
NAME :shyam
CODE :112
employee 3
NAME :jodu
CODE :113
employee 4
NAME :madhu
CODE :114
employee 5
NAME :sri
CODE :115
employee 6
NAME :teja
CODE :116

**OUTPUT 2:**

Employee details are as follows:

| NAME | CODE |
|---|---|
| ram | 111 |
| shyam | 112 |
| jodu | 113 |
| madhu | 114 |
| sri | 115 |
| teja | 116 |

**Experiment No 3:Friend Function**

**Aim: Create a class 'COMPLEX' to hold a complex number. Write a friend function toadd two complex numbers. Write a main function to add two COMPLEX objects.**

**Description:**

A friend function of a class is defined outside that class' scope but it has the right to access all private and protected members of the class. Even though the prototypes for friend functions appear in the class definition, friends are not member functions.A friend can be a function, function template, or member function, or a class or class template, in which case the entire class and all of its members are friends.Here by using friend function two complex objects are added.

**Algorithm:**

STEP 1: Start the program.
STEP 2: Declare the class complex.
STEP 3: Declare get_complex(), show_complex(). Also declare add_complex() as friend function.
STEP 4: Define get_complex() to enter the input.
STEP 5: Define show_complex() to show the output.
STEP 6: add_complex() is defined to add two complex number.
STEP 7: c1, c2, c3 are the objects of class complex.
STEP 8: By using the above object member function of the class complex is called.
STEP 10: Stop the program.

**/* Program */**

```
#include<iostream>
Using namespace std;
class complex
{
 float real,imag;
 public: void get_complex();
 void show_complex();
 friend complex add_complex(complex c1,complex c2);
};
void complex::get_complex()
{
cout<<"Enter real number :";
cin>> real;
cout<<"Enter Imaginary number :";
cin>>imag;
}
void complex::show_complex()
{
cout<<real<<"+i"<<imag;
}
complex add_complex(complex c1,complex c2)
{
 complex c;
c.real=c1.real+c2.real;
c.imag=c1.imag+c2.imag;
 return c;
}
int main()
{
clrscr();
 complex c1,c2,c3;
 c1.get_complex();
 c2.get_complex();
 c3=add_complex(c1,c2);
cout<<"\nComplex Number 1 = ";
 c1.show_complex();
cout<<"\nComplex Number 2 = ";
 c2.show_complex();
cout<<"\nSum of Complex Number 1 and 2 = ";
 c3.show_complex();
getch();
 return 0;
}
```

**INPUT 1:**
Enter real number:12
Enter Imaginary number:10
Enter real number:3
Enter Imaginary number:5

**OUTPUT 1:**
Complex Number 1 = 12+i10
Complex Number 2 = 3+i5
Sum of Complex Number 1 and 2 = 15+i15

**INPUT 2:**
Enter real number:112
Enter Imaginary number:110
Enter real number:13
Enter Imaginary number:15

**OUTPUT2:**
Complex Number 1 = 112+i110
Complex Number 12 = 13+i15
Sum of Complex Number 1 and 2 = 125+i125

**Experiment No. 4: OperatorOverloading**
**Aim: Create a 'MATRIX' class of size m X n. Overload the '+' operator to add two MATRIX objects. Write a main function to implement it.**

**Description:**
Operator overloading is an important concept in C++. It is a type of polymorphism in which an operator is overloaded to give user defined meaning to it. Overloaded operator is used to perform operation on user-defined data type. For example '+' operator can be overloaded to perform addition on various data types, like for Integer, String(concatenation) etc. Almost any operator can be overloaded in C++. However there are few operator which can not be overloaded. Operator that are not overloaded are follows
scope operator - ::
sizeof
member selector - .
member pointer selector - *
ternary operator - ?:
Here addition of two matrix of size m X n are added using '+' operator. So, here '+' operator is overloaded.

**Algorithm:**
STEP 1: Start the program.
STEP 2: Declare the class mat.
STEP 3: Define the operator '+'.
STEP 4: The function readmat() is defined to insert the elements of the matrix.
STEP 5: The function display() is defined for displaying the outputs.
STEP 6: Objects of mat class is declared in the main() function.
STEP 7: By using that the desired operations are done.
STEP 10: Stop the program.

```
/* Program */
#include<iostream.h>
#include<conio.h>
class mat
{
intm,n,a[20][20];
 public:
 mat(intx,int y);
 void readmat();
 mat operator +(mat);
 void display();
};
 mat :: mat(intx,int y)
{
 m=x;n=y;
 for(inti=0;i<m;i++)
 {
 for(int j=0;j<n;j++)
 a[i][j]=0;
 }
}
 void mat :: readmat()
 {
cout<<"\nenter matrix elements\n";
 for(inti=0;i<m;i++)
 for(int j=0;j<n;j++)
cin>>a[i][j];
 }
 mat mat:: operator +(mat obj)
 {
 mat temp(m,n);
 for(inti=0;i<m;i++)
 for(int j=0;j<n;j++)
 {
temp.a[i][j]=a[i][j]+obj.a[i][j];
 }
 return temp;
 }
 void mat:: display()
 {
inti,j;
 for(i=0;i<m;i++)
 {
cout<<"\n\n";
 for(j=0;j<n;j++)
cout<<"\t"<<a[i][j];
 }
 }
int main()
 {
int m1,n1;
clrscr();
cout<<"\nEnter the size(m,n) of matrix: ";
cin>>m1>>n1;
 mat a(m1,n1),b(m1,n1),c(m1,n1);
cout<<"\nEntermartix 1: ";
```

```
a.readmat();
cout<<"\nEnter matrix 2: ";
b.readmat();
 c=a.operator +(b);
cout<<"\nFirst Matrix :\n";
a.display();
cout<<"\nSecond Matrix :\n";
b.display();
cout<<"\nmatrix 1+matrix 2: ";
c.display();
getch();
 return 0;
 }
```

**INPUT 1:**

```
Enter the size(m,n) of matrix: 2 2
Enter martix 1: enter matrix elements
3 3
3 3
Enter matrix 2: enter matrix elements
4 4
4 4
```

**OUTPUT 1:**

```
First Matrix :
3 3
3 3
Second Matrix :
4 4
4 4
matrix 1 + matrix 2:
7 7
7 7
```

**INPUT 2:**

```
Enter the size(m,n) of matrix: 2 2
Enter martix 1: enter matrix elements
55
5 5
Enter matrix 2: enter matrix elements
4 4
4 4
```

**OUTPUT2:**

```
First Matrix :
55
5 5
Second Matrix :
4 4
4 4
matrix 1 + matrix 2:
99
9 9
```

# UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

## Lab Manual

**Experiment No. 5: Function Overloading**

**Aim: Simple Program for Function Overloading Using C++ Programming to calculate the area of circle, rectangle and triangle using function overloading.**

**Description:**

If any class have multiple functions with same names but different parameters then they are said to be overloaded. Function overloading allows you to use the same name for different functions, to perform, either same or different functions in the same class. Function overloading is usually used to enhance the readability of the program. If you have to perform one single operation but with different number or types of arguments, then you can simply overload the function. There are two ways to overload a function 1. By changing number of Arguments 2. By having different types of argument.

**Algorithm:**

STEP 1: Start the program.
STEP 2: Declare the class name as fn with data members and member functions.
STEP 3: Read the choice from the user.
STEP 4: Choice=1 then go to the step 5.
STEP 5: The function area() to find area of circle with one integer argument.
STEP 6: Choice=2 then go to the step 7.
STEP 7: The function area() to find area of rectangle with two integer argument.
STEP 8: Choice=3 then go to the step 9.
STEP 9: The function area() to find area of triangle with three arguments, two as Integer and one as float.
STEP 10: Choice=4 then stop the program.

**/*program*/**

```
#include<iostream.h>
#include<stdlib.h>
#include<conio.h>
#define pi 3.14
class fn
{
    public:
      void area(int);  //circle
      void area(int,int);  //rectangle
      void area(float ,int,int);  //triangle
};


void fn::area(int a)
{


cout<<"Area of Circle:"<<pi*a*a;
}
void fn::area(inta,int b)
{
cout<<"Area of rectangle:"<<a*b;
}
void fn::area(float t,inta,int b)
{
cout<<"Area of triangle:"<<t*a*b;
}
```

```
void main()
{
intch;
inta,b,r;
clrscr();
fnobj;
cout<<"\n\t\tFunction Overloading";
cout<<"\n1.Area of Circle\n2.Area of Rectangle\n3.Area of Triangle\n4.Exit\n:”;
cout<<”Enter your Choice:";
cin>>ch;

    switch(ch)
    {
          case 1:
cout<<"Enter Radious of the Circle:";
cin>>r;
obj.area(r);
            break;
          case 2:
cout<<"Enter Sides of the Rectangle:";
cin>>a>>b;
obj.area(a,b);
            break;
          case 3:
cout<<"Enter Sides of the Triangle:";
cin>>a>>b;
obj.area(0.5,a,b);
            break;
          case 4:
            exit(0);
    }
getch();
}
```

**OUTPUT:**

Function Overloading
1. Area of Circle
2. Area of Rectangle
3. Area of Triangle
4. Exit
Enter Your Choice: 2

Enter the Sides of the Rectangle: 5 5

Area of Rectangle is: 25

1. Area of Circle
2. Area of Rectangle
3. Area of Triangle
4. Exit
Enter Your Choice: 4

# UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

## Lab Manual

**Experiment No. 6: Inheritance**
**Aim: To find out the student details using multiple inheritance.**

**Description:**
Multiple inheritance is a feature of some object-oriented computer programming languages in which an object or class can inherit characteristics and features from more than one parent object or parent class. It is distinct from single inheritance, where an object or class may only inherit from one particular object or class.

**Algorithm:**
Step 1: Start the program.
Step 2: Declare the base class student.
Step 3: Declare and define the function get() to get the student details.
Step 4: Declare the other class sports.
Step 5: Declare and define the function getsm() to read the sports mark.
Step 6: Create the class statement derived from student and sports.
Step 7: Declare and define the function display() to find out the total and average.
Step 8: Declare the derived class object,call the functions get(),getsm() and display().
Step 9: Stop the program.

**/*program*/**

```
#include<iostream.h>
using namespace std;
class student
{
   protected:
int rno,m1,m2;
   public:
             void get()
            {
cout<<"Enter the Roll no :";
cin>>rno;
cout<<"Enter the two marks   :";
cin>>m1>>m2;
            }
};
class sports
{
   protected:
intsm;                  // sm = Sports mark
   public:
             void getsm()
            {
cout<<"\nEnter the sports mark :";
cin>>sm;

            }
};
class statement:publicstudent,public sports
{
inttot,avg;
   public:
   void display()
            {
              tot=(m1+m2+sm);
avg=tot/3;
```

```
cout<<"\n\n\tRoll No    : "<<rno<<"\n\tTotal      : "<<tot;
cout<<"\n\tAverage    : "<<avg;
          }
};
int main()
{
clrscr();
  statement obj;
obj.get();
obj.getsm();
obj.display();
}
```

**INPUT:**
Enter the Roll no: 100

Enter two marks
90
80

Enter the Sports Mark: 90

**OUTPUT:**

Roll No: 100
Total    : 260
Average: 86.66

**Experiment No. 7:Exception Handling**
**Aim: Write a C++ program illustrating Exception Handling.**

**Description:**
An exception is a problem that arises during the execution of a program. A C++ exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero.Exceptions provide a way to transfer control from one part of a program to another. C++ exception handling is built upon three keywords: try, catch, and throw.

**Algorithm:**
Step1: Start
Step2: Divide a number by 0 within try block
Step3: Throw the exception
Step4: In catch block display a massage
Step5: stop

**/* Program */**

```
#include<iostream>
Using namespace std;
int main()
{
inta,b;
cout<<"enter values of a and b \n";
cin>>a;
cin>>b;
int x=a-b;
try
```

```
{
If(x!=0)
{
Cout<<"result is "<<a/x<<"\n";
}
Else
{
Throw(x);
}
}
Catch(inti)
{
Cout<<"Exception caught: x ="<<x<<"\n";
}
```

**INPUT 1:**
enter values of a and b
20 15

**OUTPUT 1:**
result is 4

**INPUT 2:**
enter values of a and b
10 10

**OUTPUT2:**
Exception caught: x=0

**Experiment No. 8: Exception Handling**
**Aim:**To perform exception handling with multiple catch.

**Description:**
Exceptions can be thrown anywhere within a code block using throw statements. The operand of the throw statements determines a type for the exception and can be any expression and the type of the result of the expression determines the type of exception thrown.The catch block following the try block catches any exception. You can specify what type of exception you want to catch and this is determined by the exception declaration that appears in parentheses following the keyword catch.

**Algorithm:**
Step 1: Start the program.
Step 2: Declare and define the function test().
Step 3: Within the try block check whether the value is greater than zero or not.
a. if the value greater than zero throw the value and catch the corresponding exception.
b. Otherwise throw the character and catch the corresponding exception.
Step 4: Read the integer and character values for the function test().
Step 5: Stop the program.

**/* Program */**

```
#include<iostream.h>
#include<conio.h>
void test(int x)
{
  try
  {
```

```
        if(x>0)
          throw x;
    else
          throw 'x';
  }

  catch(int x)
  {
cout<<"Catch a integer and that integer is:"<<x;
  }

  catch(char x)
  {
cout<<"Catch a character and that character is:"<<x;
  }
}

void main()
{
clrscr();
cout<<"Testing multiple catches\n:";
  test(10);
  test(0);
getch();
}
```

**OUTPUT:**

Testing multiple catches
Catch a integer and that integer is: 10
Catch a character and that character is: x

**Experiment No. 9:Function Template**
**Aim: Write a C++ program to swap the numbers using the concept of function template.**

**Description:**
Templates are the foundation of generic programming, which involves writing code in a way that is independent of any particular type.A template is a blueprint or formula for creating a generic class or a function. The library containers like iterators and algorithms are examples of generic programming and have been developed using template concept.

Syntax of function template:

```
template <class type> ret-type func-name(parameter list)
 {
   // body of function
}
```

**Algorithm:**
STEP 1: Start the program.
STEP 2: Declare the template class.
STEP 3: Declare and define the functions to swap the values.
STEP 4: Declare and define the functions to get the values.
STEP 5: Read the values and call the corresponding functions.
STEP6: Display the results.
STEP 7: Stop the program.

```
/* Program */
#include<iostream.h>
#include<conio.h>

template<class t>

void swap(t &x,t&y)
{
  t temp=x;
  x=y;
  y=temp;
}

void fun(inta,intb,floatc,float d)
{
cout<<"\na and b before swaping :"<<a<<"\t"<<b;
  swap(a,b);
cout<<"\na and b after swaping  :"<<a<<"\t"<<b;
cout<<"\n\nc and d before swaping :"<<c<<"\t"<<d;
  swap(c,d);
cout<<"\nc and d after swaping  :"<<c<<"\t"<<d;
}

void main()
{
inta,b;
   float c,d;
clrscr();
cout<<"Enter A,B values(integer):";
cin>>a>>b;
cout<<"Enter C,D values(float):";
cin>>c>>d;
   fun(a,b,c,d);
getch();
}
```

**INPUT 1:**
Enter A, B values (integer): 10  20
Enter C, D values (float):  2.50  10.80

**OUTPUT 1:**
A and B before swapping: 10 20
A and B after swapping:  20 10

C and D before swapping: 2.50  10.80
C and D after swapping: 10.80  2.50

**Experiment No. 10: Function template.**
**Aim: Program to display largest among two numbers using function templates.**

**Description**:
A function template Large() is defined that accepts two arguments n1 and n2 of data type T. T signifies that argument can be of any data type.Large() function returns the largest among the two arguments using a simple conditional operation.Inside the main() function, variables of three different data types: int, float and char are declared. The variables are then passed to the Large()

function template as normal functions.During run-time, when an integer is passed to the template function, compiler knows it has to generate a Large() function to accept the int arguments and does so.Similarly, when floating-point data and char data are passed, it knows the argument data types and generates the Large() function accordingly.This way, using only a single function template replaced three identical normal functions and made your code maintainable.

**Algorithm:**

STEP 1: Start the program.
STEP 2: Declare the template function.
STEP 3: Declare and define the function Large() tofind the largest value.
STEP 4: After taking input from user, the largest value is identified
STEP 5: Display the results.
STEP 7: Stop the program.

**/*Program*/**

```
#include <iostream>
using namespace std;
// template function
template <class T>
T Large(T n1, T n2)
{
        return (n1 > n2) ? n1 : n2;
}
int main()
{
        int i1, i2;
        float f1, f2;
        char c1, c2;

        cout<< "Enter two integers:\n";
        cin>> i1 >> i2;
        cout<< Large(i1, i2) <<" is larger." <<endl;

        cout<< "\nEnter two floating-point numbers:\n";
        cin>> f1 >> f2;
        cout<< Large(f1, f2) <<" is larger." <<endl;
        cout<< "\nEnter two characters:\n";
        cin>> c1 >> c2;
        cout<< Large(c1, c2) << " has larger ASCII value.";
        return 0;
}
```

**INPUT 1:**

Enter two integers:
5
10

**OUTPUT 1:**

10 is larger.

**INPUT 2:**

Enter two floating-point numbers:
12.4
10.2

**OUTPUT 2:**

12.4 is larger.

**INPUT 3:**
Enter two characters:
z
Z
**OUTPUT 3:**
z has larger ASCII value.

**Experiment No. 11: Class template**
**Aim: Simple calculator using Class template. Program to add, subtract, multiply and divide two numbers using class template.**

**Description**:
The class contains two private members of type T: num1 & num2, and a constructor to initalize the members.It also contains public member functions to calculate the addition, subtraction, multiplication and division of the numbers which return the value of data type defined by the user. Likewise, a function displayResult() to display the final output to the screen.In the main() function, two different Calculator objects intCalc and floatCalc are created for data types: int and float respectively. The values are initialized using the constructor.
We use <int> and <float> while creating the objects. These tell the compiler the data type used for the class creation.This creates a class definition each for int and float, which are then used accordingly.Then, displayResult() of both objects is called which performs the Calculator operations and displays the output.

**Algorithm:**
STEP 1: Start the program.
STEP 2: Declare the template class.
STEP 3: Declare and define the function displayResult() tofind the result of the mathematical operations.
STEP 4: intCalc(2, 1) doing the operations on integer values.
STEP 5: floatCalc(2.4, 1.2) doing the operations on float values.
STEP 6: Display the results.
STEP 7: Stop the program.

**/*Program*/**
```
#include <iostream>
using namespace std;
template <class T>
class Calculator
{
private:
        T num1, num2;

public:
        Calculator(T n1, T n2)
        {
                num1 = n1;
                num2 = n2;
        }
                void displayResult()
        {
                cout<< "Numbers are: " << num1 << " and " << num2 << "." <<endl;
                cout<< "Addition is: " << add() <<endl;
                cout<< "Subtraction is: " << subtract() <<endl;
                cout<< "Product is: " << multiply() <<endl;
                cout<< "Division is: " << divide() <<endl;
        }
```

```
        T add() { return num1 + num2; }

        T subtract() { return num1 - num2; }

        T multiply() { return num1 * num2; }

        T divide() { return num1 / num2; }
};

int main()
{
        Calculator<int>intCalc(2, 1);
        Calculator<float>floatCalc(2.4, 1.2);

        cout<< "Int results:" <<endl;
        intCalc.displayResult();

        cout<<endl<< "Float results:" <<endl;
        floatCalc.displayResult();

        return 0;
}
```

**OUTPUT :**

```
Int results:
Numbers are: 2 and 1.
Addition is: 3
Subtraction is: 1
Product is: 2
Division is: 2

Float results:
Numbers are: 2.4 and 1.2.
Addition is: 3.6
Subtraction is: 1.2
Product is: 2.88
Division is: 2
```

**Experiment No. 12:CONSOLE INPUT OUTPUT**

**Aim: Write a C++ program where user will input the text and how many characters are entered would be shown in the output.**

**Description**:

The classes istream and ostream define two member function put() and get() respectively to handle the single character input/output operations. When we type a line of input, the text is sent to the program as soon as we press the RETURN key. The program then reads one character at a time using the statement cin.get( c); and displays it using the statement cout.put( c); The process is terminated when the newline character is encountered.

**Algorithm:**

STEP 1: Start the program.
STEP 2: Taking inputs from user.
STEP 3: Input will be taken character wise.
STEP 4: After taking input from user, it will count the number of characters.
STEP 5: Display the number of characters.

STEP 6: Stop the program.

```
/*Program*/
#include<iostream>
 using namespace std;
int main()
 {
        int count=0;
        char c;
        cout<<"input text:";
        cin.get(c);
        while(c!='\n')
        {
                cout.put(c);
                count++;
                cin.get(c);
        }
        cout<<"\n number of characters= "<<count;
        return 0;
 }
```

**INPUT 1:**
input text: object oriented programming

**OUTPUT 1:**
object oriented programming
number of characters=27

**INPUT 1:**
input text: happy new year
**OUTPUT2:**
happy new year
number of characters=14

**Experiment No. 13: CONSOLE INPUT OUTPUT**
**Aim: Write a C++ program to implement the working principle ofgetline() function.**

**Description**:
We can read and display a line of text more efficiently using the line oriented input functions getline(). The getline() function reads a whole line of a text that ends with a newline character. The reading is terminated as soon as either the newline character '\n' is encountered or size-1 characters are read (whichever occurs first). The newline character is read byt not saved. Instead it is replaced by NULL character.

**Algorithm:**
STEP 1: Start the program.
STEP 2: Taking inputs from user.
STEP 3: Input will be taken as a whole line.
STEP 4: After taking input from user, it will print the desired output.
STEP 5: Display the result.
STEP 6: Stop the program.

```
/*Program*/
#include<iostream>
using namespace std;
int main()
{
        int size=20;
        char city[20];
        cout<<"enter the city name: \n";
        cin>>city;
        cout<<"city name: "<<city<<"\n\n";
        cout<<"enter the city name again: \n";
        cin.getline(city,size);
        cout<<"city name now: "<<city<<"\n\n";
        cout<<"enter the another city name: \n";
        cin.getline(city,size);
        cout<<"New city name is: "<<city<<"\n\n";
}
```

**INPUT 1:**
Enter city name:
Delhi

**OUTPUT 1:**
City name:
Delhi
City name: Delhi
enter the city name again:
city name now:
enter the another city name:
chennai
New city name is: Chennai

**INPUT 1:**
Enter city name:
New Delhi

**OUTPUT 1:**
City name: New
enter the city name again:
city name now: Delhi
enter the another city name:
Greater Kolkata
New city name is: Greater Kolkata

**Experiment No. 14:CONSOLE INPUT OUTPUT**
**Aim: Write a C++ program to create the following format.**

| ITEMS | COST | TOTAL VALUE |
|---|---|---|
| 10 | 75 | 750 |
| 6 | 100 | 600 |
| 12 | 60 | 720 |
| 15 | 99 | 1485 |

**GRAND TOTAL= 3555**

**Description**:
For formatting the output width() function is used. To set the required field width we use it. The output will be displayed in given width.

**Algorithm:**

STEP 1: Start the program.
STEP 2: Define arrays of items and cost.
STEP 3: Set the output with width() function
STEP 4: Display the desired output.
STEP 5: Stop the program.

**/*Program*/**

```
#include<iostream>
using namespace std;
int main()
{
        int items[4]={10,6,12,15};
        int cost[4]={75,100,60,99};
        cout.width(5);
        cout<<"ITEMS";
        cout.width(8);
        cout<<"COST";
        cout.width(15);
        cout<<"TOTAL VALUE"<<"\n";
        int sum=0;
        for(inti=0;i<4;i++)
        {
                cout.width(5);
                cout<<items[i];
                cout.width(8);
                cout<<cost[i];
                int value=items[i] * cost[i];
                cout.width(15);
                cout<<value<<"\n";
                sum=sum+value;
        }
        cout<<"\n GRAND TOTAL= ";
        cout.width(2);
        cout<<sum<<"\n";
        return 0;
}
```

**OUTPUT 1:**

| ITEMS | COST | TOTAL VALUE |
|---|---|---|
| 10 | 75 | 750 |
| 6 | 100 | 600 |
| 12 | 60 | 720 |
| 15 | 99 | 1485 |

GRAND TOTAL= 3555

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

## Lab Manual

**Experiment No. 15:CONSOLE INPUT OUTPUT**

**Aim: Write a C++ program where you find the square root of any five values. Format the desired output with precision() function.**

**Description**:

By using precision () function we just format the output. This function is used to set number of decimal points to a float value.

**Algorithm:**

STEP 1: Start the program.
STEP 2: Set precision.
STEP 3: Set the width of output and find the square root of particular values.
STEP 4: Display the desired output.
STEP 5: Stop the program.

**/*Program*/**

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{       cout<<"precision set to 3 digits \n\n";
        cout.precision(3);
        cout.width(10);
        cout<<"VALUE";
        cout.width(15);
        cout<<"SORT_OF_VALUE"<<"\n";
        for(int n=1;n<=5;n++)
        {
                cout.width(8);
                cout<<n;
                cout.width(13);
                cout<<sqrt(n)<<"\n";
        }
        cout<<"\n PRECISION SET TO 5 DIGITS \n\n";
        cout.precision(5);
        cout<<"sqrt(10)= "<<sqrt(10)<<"\n\n";
        cout.precision(0);
        cout<<"sqrt(10)= "<<sqrt(10)<<"\n default settings \n";
        return 0;

}
```

**OUTPUT:**

precision set to 3 digits

| VALUE | SORT_OF_VALUE |
|---|---|
| 1 | 1 |
| 2 | 1.41 |
| 3 | 1.73 |
| 4 | 2 |
| 5 | 2.24 |

PRECISION SET TO 5 DIGITS

sqrt(10)= 3.1623
sqrt(10)= 3
default settings

# UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

## Lab Manual

**Experiment No. 16:CONSOLE INPUT OUTPUT**
**Aim: Fill the blank spaces of the above program with '.'**

**Description**:
By using fill () function we just format the output. The blank spaces of the above output is replaced by '.'

**Algorithm:**
STEP 1: Start the program.
STEP 2: Set precision.
STEP 3: Set the width of output and find the square root of particular values.
STEP 4: Fill up the blank spaces with '.' Using fill() function.
STEP 5: Display the desired output.
STEP 6: Stop the program.

**/*Program*/**

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
        cout<<"precision set to 3 digits \n\n";
        cout.precision(3);
        cout.fill('.');
        cout.width(10);
        cout<<"VALUE";
        cout.width(15);
        cout<<"SORT_OF_VALUE"<<"\n";
        for(int n=1;n<=5;n++)
        {
                cout.width(8);
                cout<<n;
                cout.width(13);
                cout<<sqrt(n)<<"\n";
        }
        cout<<"\n PRECISION SET TO 5 DIGITS \n\n";
        cout.precision(5);
        cout<<"sqrt(10)= "<<sqrt(10)<<"\n\n";
        cout.precision(0);
        cout<<"sqrt(10)"<<sqrt(10)<<"default setting \n";
        return 0;

}
```

**OUTPUT:**

```
precision set to 3 digits

    …….VALUE…… SORT_OF_VALUE
……..1……1
……...……...2……...1.41
……..3……...……1.73
……..4……2
……..5……2.24
PRECISION SET TO 5 DIGITS

sqrt(10)= 3.1623
```

sqrt(10)= 3
default settings

**Experiment No. 17:FILE HANDLING**
**Aim: Write a Program for Read File Operation Using C++ Programming**

**Description**:
File handling concept in C++ language is used for store a data permanently in computer. Using file handling we can store our data in Secondary memory (Hard disk). fstream is used to both read and write data from/to files

**Algorithm:**
STEP 1: Start the program.
STEP 2: Declare the variables.
STEP 3: Get the file name to read.
STEP 4: Using ifstreamin(filename) check whether the file exist.
STEP 5: If the file exist then check for the end of file condition.
STEP 6: Read the contents of the file.
STEP 7: Print the contents of the file.
STEP 8: Stop the program.

**/*Program*/**

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
        char c,fname[10];
cout<<"Enter file name:";
cin>>fname;
ifstream in(fname);
        if(!in)
        {
cout<<"File Does not Exist";
                return;
        }
cout<<"\n\n";
        while(in.eof()==0)
        {
in.get(c);
cout<<c;
        }

}
```

**OUTPUT:**
Enter File name: one.txt
INDIA

# UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

## Lab Manual

**Experiment No. 18:FILE HANDLING**

**Aim:Write a Program for Read & Write File Operation (Convert lowercase to uppercase) Using C++ Programming**

**Description**:

A file must be opened before you can read from it or write to it. Either the ofstream or fstream object may be used to open a file for writing or ifstream object is used to open a file for reading purpose only.C++ provides a special function, eof( ), that returns nonzero (meaning TRUE) when there are no more data to be read from an input file stream, and zero (meaning FALSE) otherwise.

**Algorithm:**

STEP 1: Start the program.
STEP 2: Declare the variables.
STEP 3: Read the file name.
STEP 4: open the file to write the contents.
STEP 5: writing the file contents up to reach a particular condition.
STEP6: write the file contents as uppercase.
STEP7: open the file to read the contents.
STEP 8: Stop the program.

**/*Program*/**

```
#include<fstream.h>
#include<stdio.h>
#include<ctype.h>
#include<string.h>
#include<iostream.h>
#include<conio.h>
void main()
{           char c,u;
          char fname[10];
clrscr();
ofstream out;
cout<<"Enter File Name:";
cin>>fname;
out.open(fname);
cout<<"Enter the text(Enter # at end)\n";   //write contents to file
          while((c=getchar())!='#')
          {
                    u=c-32;
                    out<<u;
          }
out.close();
ifstream in(fname);       //read the contents of file
cout<<"\n\n\t\tThe File contains\n\n";
          while(in.eof()==0)
          {         in.get(c);
cout<<c;
          }
getch();
 }
```

**OUTPUT:**

```
Enter File Name: two.txt
Enter contents to store in file (enter # at end)
oops programming
The File Contains
OOPS PROGRAMMING
```

**Experiment No. 19:FILE HANDLING**
**Aim:** Program to count number of words in a file.

**Description**:
File handling concept in C++ language is used for store a data permanently in computer. Using file handling we can store our data in Secondary memory. For read and write from a file we need another standard C++ library called fstream. Always test for the end-of-file condition before processing data read from an input file stream. Use a while loop for getting data from an input file stream. A for loop is desirable only when you know the exact number of data items in the file, which we do not know.

**Algorithm:**
STEP 1: Start the program.
STEP 2: Declare the variables.
STEP 3: Read the file name.
STEP 5: Check the eof condition
STEP6: Count the number of words
STEP 7: Stop the program.

**/*Program*/**

```
#include<fstream>
#include<iostream>
using namespace std;
int main()
{
ifstream fin;
fin.open("out.txt");

int count = 0;
   char word[30];

   while(!fin.eof())
   {
      fin >> word;
      count++;
   }

cout<< "Number of words in file are " << count;

fin.close();
   return 0;
}
```

**OUTPUT:**
Number of words in file are 20.

# UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

## Lab Manual

**Experiment No. 20:FILE HANDLING**

**Aim:**Program to count number of lines in a text file.

**Description**:

File handling concept in C++ language is used for store a data permanently in computer. Using file handling we can store our data in Secondary memory. For read and write from a file we need another standard C++ library called fstream. Always test for the end-of-file condition before processing data read from an input file stream. Use a while loop for getting data from an input file stream. A for loop is desirable only when you know the exact number of data items in the file, which we do not know.

**Algorithm:**

STEP 1: Start the program.
STEP 2: Declare the variables.
STEP 3: Read the file name.
STEP 5: Check the eof condition
STEP 6: Count the number of lines
STEP 7: Stop the program.

**/*Program*/**

```
#include<fstream>
#include<iostream>
using namespace std;
int main()
{
ifstream fin;
fin.open("out.txt");

int count = 0;
   char str[80];

   while(!fin.eof())
   {
fin.getline(str,80);
      count++;
   }
cout<< "Number of lines in file are " << count;
fin.close();
   return 0;
}
```

**OUTPUT:**

Number of lines in file are 5.

**Experiment No. 21:FILE HANDLING**

**Aim:**Program to implement searching operation on binary file in C++

**Description**:

When data is stored in a file in the binary format, reading and writingdata is faster because no time is lost in converting the data from one format to another format. Such files are called binary files. This following program explains how to create binary files and also how to read, write, search, delete and modify data from binary files.

**Algorithm:**

STEP 1: Start the program.
STEP 2: Declare the variables and functions with definitions.
STEP 3: Read the file name.
STEP 5: Check the eof condition
STEP 6: searching is done.
STEP 7: Stop the program.

**/*Program*/**

```
#include<fstream.h>
#include<conio.h>
#include<stdlib.h>
class student
{
        introllno;
        char name[20];
        char branch[3];
        float marks;
        char grade;
        public:
                void getdata()
                {
                        cout<<"Rollno: ";
                        cin>>rollno;
                        cout<<"Name: ";
                        cin>>name;
                        cout<<"Branch: ";
                        cin>>branch;
                        cout<<"Marks: ";
                        cin>>marks;

                        if(marks>=75)
                        {
                                grade = 'A';
                        }
                        else if(marks>=60)
                        {
                                grade = 'B';
                        }
                        else if(marks>=50)
                        {
                                grade = 'C';
                        }
                        else if(marks>=40)
                        {
                                grade = 'D';
                        }
                        else
```

```
                {
                        grade = 'F';
                }
        }
        void putdata()
        {
                cout<<"Rollno: "<<rollno<<"\tName: "<<name<<"\n";
                cout<<"Marks: "<<marks<<"\tGrade: "<<grade<<"\n";
        }
        intgetrno()
        {
                return rollno;
        }
}stud1;
void main()
{
        clrscr();

        fstreamfio("marks.dat", ios::in | ios::out);
        char ans='y';
        while(ans=='y' || ans=='Y')
        {
                stud1.getdata();
                fio.write((char *)&stud1, sizeof(stud1));
                cout<<"Record added to the file\n";
                cout<<"\nWant to enter more ? (y/n)..";
                cin>>ans;
        }
        clrscr();
        intrno;
        long pos;
        char found='f';
        cout<<"Enter rollno of student to be search for: ";
        cin>>rno;
        fio.seekg(0);
        while(!fio.eof())
        {
                pos=fio.tellg();
                fio.read((char *)&stud1, sizeof(stud1));
                if(stud1.getrno() == rno)
                {
                        stud1.putdata();
                        fio.seekg(pos);
                        found='t';
                        break;
                }
        }
        if(found=='f')
        {
                cout<<"\nRecord not found in the file..!!\n";
                cout<<"Press any key to exit...\n";
                getch();
                exit(2);
        }
fio.close();
getch();
}
```

**OUTPUT:**
Rollno: 1
Name: Aman
Branch: CSE
Marks: 96
Recorded added to the file

Want to enter more? (y/n)..y
Rollno: 2
Name: Savvy
Branch: IT
Marks: 91
Recorded added to the file

Want to enter more? (y/n)..n

**Searching**
Enterrollno of student to be search for: 2
Rollno: 2 Name: Savvy
Marks: 91 Grade: A

**Experiment No. 22:STL**
**Aim:**C++ Program to Implement Queue in STL.

**Description**:
The C++ STL (Standard Template Library) is a powerful set of C++ template classes to provide general-purpose templatized classes and functions that implement many popular and commonly used algorithms and data structures like vectors, lists, queues, and stacks.This C++ Program demonstrates implementation of Queue in STL. Here is source code of the C++ Program to demonstrate Queue in STL.

**Algorithm:**
STEP 1: Start the program.
STEP 2: Declare the variables and functions with definitions.
STEP 3: Define STL of queue
STEP 5: Doing queue operations
STEP 6: Stop the program.

**/*Program*/**
```
#include <iostream>
#include <queue>
#include <string>
#include <cstdlib>
using namespace std;
int main()
{
   queue<int> q;
int choice, item;
   while (1)
   {
cout<<"\n---------------------"<<endl;
cout<<"Queue Implementation in Stl"<<endl;
cout<<"\n---------------------"<<endl;
cout<<"1.Insert Element into the Queue"<<endl;
cout<<"2.Delete Element from the Queue"<<endl;
```

```
            cout<<"3.Size of the Queue"<<endl;
cout<<"4.Front Element of the Queue"<<endl;
cout<<"5.Last Element of the Queue"<<endl;
cout<<"6.Exit"<<endl;
cout<<"Enter your Choice: ";
cin>>choice;
        switch(choice)
        {
        case 1:
cout<<"Enter value to be inserted: ";
cin>>item;
q.push(item);
            break;
        case 2:
            item = q.front();
q.pop();
cout<<"Element "<<item<<" Deleted"<<endl;
            break;
        case 3:
             cout<<"Size of the Queue: ";
             cout<<q.size()<<endl;
            break;
        case 4:
cout<<"Front Element of the Queue: ";
             cout<<q.front()<<endl;
            break;
        case 5:
cout<<"Back Element of the Queue: ";
cout<<q.back()<<endl;
            break;
        case 6:
            exit(1);
                break;
        default:
cout<<"Wrong Choice"<<endl;
        }
    }
    return 0;
}
```

**OUTPUT:**

```
1.Insert Element into the Queue
2.Delete Element from the Queue
3.Size of the Queue
4.Front Element of the Queue
5.Last Element of the Queue
6.Exit
Enter your Choice: 1
Enter value to be inserted: 9

---------------------
Queue Implementation in Stl

---------------------
1.Insert Element into the Queue
2.Delete Element from the Queue
3.Size of the Queue
```

```
4.Front Element of the Queue
5.Last Element of the Queue
6.Exit
Enter your Choice: 1
Enter value to be inserted: 8
---------------------
Queue Implementation in Stl

---------------------
1.Insert Element into the Queue
2.Delete Element from the Queue
3.Size of the Queue
4.Front Element of the Queue
5.Last Element of the Queue
6.Exit
Enter your Choice: 3
Size of the Queue: 2
```

**Experiment No. 23:STL**
**Aim:**C++ Program to Implement Vector in STL

**Description**:
Vector is a template class that is a perfect replacement for the good old C-style arrays. It allows the same natural syntax that is used with plain arrays but offers a series of services that free the C++ programmer from taking care of the allocated memory and help operating consistently on the contained objects.This C++ Program demonstrates implementation of Vector in STL.Here is source code of the C++ Program to demonstrate Vector in STL.

**Algorithm:**
STEP 1: Start the program.
STEP 2: Declare the variables and functions with definitions.
STEP 3: Define STL of vector
STEP 5: Doing vector operations
STEP 6: Stop the program.

**/*Program*/**

```
#include <iostream>
#include <vector>
#include <string>
#include <cstdlib>
using namespace std;
int main()
{
   vector<int>ss;
   vector<int>::iterator it;
int choice, item;
   while (1)
   {
cout<<"\n---------------------"<<endl;
cout<<"Vector Implementation in Stl"<<endl;
cout<<"\n---------------------"<<endl;
cout<<"1.Insert Element into the Vector"<<endl;
cout<<"2.Delete Last Element of the Vector"<<endl;
```

```
cout<<"3.Size of the Vector"<<endl;
cout<<"4.Display by Index"<<endl;
cout<<"5.Dislplay by Iterator"<<endl;
cout<<"6.Clear the Vector"<<endl;
cout<<"7.Exit"<<endl;
cout<<"Enter your Choice: ";
cin>>choice;
      switch(choice)
      {
      case 1:
cout<<"Enter value to be inserted: ";
cin>>item;
ss.push_back(item);
         break;
      case 2:
cout<<"Delete Last Element Inserted:"<<endl;
ss.pop_back();
         break;
      case 3:
cout<<"Size of Vector: ";
cout<<ss.size()<<endl;
         break;
      case 4:
cout<<"Displaying Vector by Index: ";
         for (inti = 0; i<ss.size(); i++)
         {
cout<<ss[i]<<" ";
         }
cout<<endl;
         break;
      case 5:
cout<<"Displaying Vector by Iterator: ";
         for (it = ss.begin(); it != ss.end(); it++)
         {
cout<<*it<<" ";
         }
cout<<endl;
         break;
      case 6:
ss.clear();
cout<<"Vector Cleared"<<endl;
         break;
      case 7:
         exit(1);
         break;
      default:
cout<<"Wrong Choice"<<endl;
      }
   }
   return 0;
}
```

**OUTPUT:**

```
----------
Vector Implementation in Stl

----------
1.Insert Element into the Vector
2.Delete Last Element of the Vector
3.Size of the Vector
4.Display by Index
5.Dislplay by Iterator
6.Clear the Vector
7.Exit
Enter your Choice: 1
Enter value to be inserted: 4

----------
Vector Implementation in Stl

----------
1.Insert Element into the Vector
2.Delete Last Element of the Vector
3.Size of the Vector
4.Display by Index
5.Dislplay by Iterator
6.Clear the Vector
7.Exit
Enter your Choice: 1
Enter value to be inserted: 6

----------
Vector Implementation in Stl

----------
1.Insert Element into the Vector
2.Delete Last Element of the Vector
3.Size of the Vector
4.Display by Index
5.Dislplay by Iterator
6.Clear the Vector
7.Exit
Enter your Choice: 1
Enter value to be inserted: 3

----------
Vector Implementation in Stl

----------
1.Insert Element into the Vector
2.Delete Last Element of the Vector
3.Size of the Vector
4.Display by Index
5.Dislplay by Iterator
6.Clear the Vector
7.Exit
Enter your Choice: 1
Enter value to be inserted: 8
```

---
Vector Implementation in Stl

---
1.Insert Element into the Vector
2.Delete Last Element of the Vector
3.Size of the Vector
4.Display by Index
5.Dislplay by Iterator
6.Clear the Vector
7.Exit
Enter your Choice: 1
Enter value to be inserted: 9

---
Vector Implementation in Stl

---
1.Insert Element into the Vector
2.Delete Last Element of the Vector
3.Size of the Vector
4.Display by Index
5.Dislplay by Iterator
6.Clear the Vector
7.Exit
Enter your Choice: 1
Enter value to be inserted: 2

---
Vector Implementation in Stl

---
1.Insert Element into the Vector
2.Delete Last Element of the Vector
3.Size of the Vector
4.Display by Index
5.Dislplay by Iterator
6.Clear the Vector
7.Exit
Enter your Choice: 3
Size of Vector: 6

---
Vector Implementation in Stl

---
1.Insert Element into the Vector
2.Delete Last Element of the Vector
3.Size of the Vector
4.Display by Index
5.Dislplay by Iterator
6.Clear the Vector
7.Exit
Enter your Choice: 4
Displaying Vector by Index: 4 6 3 8 9 2

---

Vector Implementation in Stl

---
1.Insert Element into the Vector
2.Delete Last Element of the Vector
3.Size of the Vector
4.Display by Index
5.Dislplay by Iterator
6.Clear the Vector
7.Exit
Enter your Choice: 2
Delete Last Element Inserted:

---
Vector Implementation in Stl

---
1.Insert Element into the Vector
2.Delete Last Element of the Vector
3.Size of the Vector
4.Display by Index
5.Dislplay by Iterator
6.Clear the Vector
7.Exit
Enter your Choice: 3
Size of Vector: 5

---
Vector Implementation in Stl

---
1.Insert Element into the Vector
2.Delete Last Element of the Vector
3.Size of the Vector
4.Display by Index
5.Dislplay by Iterator
6.Clear the Vector
7.Exit
Enter your Choice: 5
Displaying Vector by Iterator: 4 6 3 8 9

# UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

## Lab Manual

**Experiment No. 24:STL**
**Aim:**C++ Program to Implement Set in STL

**Description**:
This C++ Program demonstrates implementation of Set in STL.Here is source code of the C++ Program to demonstrate Set in STL.

**Algorithm:**
STEP 1: Start the program.
STEP 2: Declare the variables and functions with definitions.
STEP 3: Define STL of set
STEP 5: Doing set operations
STEP 6: Stop the program.

**/*Program*/**

```
#include <iostream>
#include <set>
#include <string>
#include <cstdlib>
using namespace std;
int main()
{
   set<int>st;
   set<int>::iterator it;
int choice, item;
   while (1)
   {
cout<<"\n----------------------"<<endl;
cout<<"Set Implementation in Stl"<<endl;
cout<<"\n----------------------"<<endl;
cout<<"1.Insert Element into the Set"<<endl;
cout<<"2.Delete Element of the Set"<<endl;
cout<<"3.Size of the Set"<<endl;
cout<<"4.Find Element in a Set"<<endl;
cout<<"5.Dislplay by Iterator"<<endl;
cout<<"6.Exit"<<endl;
cout<<"Enter your Choice: ";
cin>>choice;
       switch(choice)
       {
       case 1:
cout<<"Enter value to be inserted: ";
cin>>item;
st.insert(item);
          break;
       case 2:
cout<<"Enter the element to be deleted: ";
cin>>item;
st.erase(item);
          break;
       case 3:
cout<<"Size of the Set: ";
cout<<st.size()<<endl;
          break;
       case 4:
           cout<<"Enter the element to be found: ";
           cin>>item;
```

```
        it = st.find(item);
            if (it != st.end())
cout<<"Element "<<*it<<" found in the set" <<endl;
        else
cout<<"No Element Found"<<endl;
        break;
      case 5:
cout<<"Displaying Map by Iterator: ";
        for (it = st.begin(); it != st.end(); it++)
        {
cout<< (*it)<<" ";
        }
cout<<endl;
        break;
      case 6:
        exit(1);
            break;
      default:
cout<<"Wrong Choice"<<endl;
      }
   }
   return 0;
}
```

**OUTPUT:**

```
----------------------
Set Implementation in Stl
 ----------------------
1.Insert Element into the Set
2.Delete Element of the Set
3.Size of the Set
4.Find Element in a Set
5.Dislplay by Iterator
6.Exit
Enter your Choice: 1
Enter value to be inserted: 1

----------------------
Set Implementation in Stl
 ----------------------
1.Insert Element into the Set
2.Delete Element of the Set
3.Size of the Set
4.Find Element in a Set
5.Dislplay by Iterator
6.Exit
Enter your Choice: 1
Enter value to be inserted: 2

----------------------
Set Implementation in Stl
 ----------------------
1.Insert Element into the Set
2.Delete Element of the Set
3.Size of the Set
4.Find Element in a Set
5.Dislplay by Iterator
```

```
6.Exit
Enter your Choice: 1
Enter value to be inserted: 3

---------------------
Set Implementation in Stl
 ---------------------
1.Insert Element into the Set
2.Delete Element of the Set
3.Size of the Set
4.Find Element in a Set
5.Dislplay by Iterator
6.Exit
Enter your Choice: 1
Enter value to be inserted: 4

---------------------
Set Implementation in Stl
 ---------------------
1.Insert Element into the Set
2.Delete Element of the Set
3.Size of the Set
4.Find Element in a Set
5.Dislplay by Iterator
6.Exit
Enter your Choice: 1
Enter value to be inserted: 5

---------------------
Set Implementation in Stl
 ---------------------
1.Insert Element into the Set
2.Delete Element of the Set
3.Size of the Set
4.Find Element in a Set
5.Dislplay by Iterator
6.Exit
Enter your Choice: 1
Enter value to be inserted: 4

---------------------
Set Implementation in Stl
 ---------------------
1.Insert Element into the Set
2.Delete Element of the Set
3.Size of the Set
4.Find Element in a Set
5.Dislplay by Iterator
6.Exit
Enter your Choice: 1
Enter value to be inserted: 3

---------------------
Set Implementation in Stl
 ---------------------
1.Insert Element into the Set
2.Delete Element of the Set
```

```
3.Size of the Set
4.Find Element in a Set
5.Dislplay by Iterator
6.Exit
Enter your Choice: 1
Enter value to be inserted: 2

---------------------
Set Implementation in Stl
 ---------------------
1.Insert Element into the Set
2.Delete Element of the Set
3.Size of the Set
4.Find Element in a Set
5.Dislplay by Iterator
6.Exit
Enter your Choice: 1
Enter value to be inserted: 1

---------------------
Set Implementation in Stl
 ---------------------
1.Insert Element into the Set
2.Delete Element of the Set
3.Size of the Set
4.Find Element in a Set
5.Dislplay by Iterator
6.Exit
Enter your Choice: 3
Size of the Set: 5

---------------------
Set Implementation in Stl
 ---------------------
1.Insert Element into the Set
2.Delete Element of the Set
3.Size of the Set
4.Find Element in a Set
5.Dislplay by Iterator
6.Exit
Enter your Choice: 5
Displaying Map by Iterator: 1 2 3 4 5

---------------------
Set Implementation in Stl
 ---------------------
1.Insert Element into the Set
2.Delete Element of the Set
3.Size of the Set
4.Find Element in a Set
5.Dislplay by Iterator
6.Exit
Enter your Choice: 4
Enter the element to be found: 3
Element 3 found in the set
```

---
Set Implementation in Stl
---
1.Insert Element into the Set
2.Delete Element of the Set
3.Size of the Set
4.Find Element in a Set
5.Dislplay by Iterator
6.Exit
Enter your Choice: 2
Enter the element to be deleted: 5

**Experiment No. 25:String Manipulation**
**Aim:**Write a C++ program to copy a string from another string. Also concatenate them and find the total length after concatenation.

**Description**:
C++ provides following two types of string representations:
-The C-style character string.
-The string class type introduced with Standard C++.
The C-style character string originated within the C language and continues to be supported within C++. This string is actually a one-dimensional array of characters which is terminated by a null character '\0'. Thus a null-terminated string contains the characters that comprise the string followed by a null.
C++ supports a wide range of functions that manipulate null-terminated strings:
**strcpy(s1, s2);**
Copies string s2 into string s1.
**strcat(s1, s2);**
Concatenates string s2 onto the end of string s1.
**strlen(s1);**
Returns the length of string s1.
**strcmp(s1, s2);**
Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2.
**strchr(s1, ch);**
Returns a pointer to the first occurrence of character ch in string s1.
**strstr(s1, s2);**
Returns a pointer to the first occurrence of string s2 in string s1.

**Algorithm:**
STEP 1: Start the program.
STEP 2: Declare the string variables.
STEP 3: Copy the content of str1 to str3
STEP 5: Concatenate str1 and str2
STEP 6: Find the length of str1
STEP 7: Stop the program.

**/*Program*/**
```
#include <iostream>
#include <cstring>
using namespace std;
int main ()
{
  char str1[10] = "Hello";
  char str2[10] = "World";
  char str3[10];
intlen ;
```

```
  // copy str1 into str3
strcpy( str3, str1);
cout<< "strcpy( str3, str1) : " << str3 <<endl;

  // concatenates str1 and str2
strcat( str1, str2);
cout<< "strcat( str1, str2): " << str1 <<endl;

  // total lenghth of str1 after concatenation
len = strlen(str1);
cout<< "strlen(str1) : " <<len<<endl;

  return 0;
}
```

**OUTPUT:**

```
strcpy( str3, str1) : Hello
strcat( str1, str2): HelloWorld
strlen(str1) : 10
```

**Experiment No. 26:String Manipulation**

**Aim:**Write a C++ program to implement he String Class in C++.

**Description**:

The standard C++ library provides a string class type that supports all the operations mentioned above, additionally much more functionality.

**Algorithm:**

STEP 1: Start the program.
STEP 2: Declare the string variables.
STEP 3: Copy str1 into str3
STEP 5: Concatenate str1 and str2 and stores into str3
STEP 6: Find the length of str3
STEP 7: Stop the program.

**/*Program*/**

```
#include <iostream>
#include <string>
using namespace std;
int main ()
{
  string str1 = "Hello";
  string str2 = "World";
  string str3;
intlen ;

  // copy str1 into str3
  str3 = str1;
cout<< "str3 : " << str3 <<endl;

  // concatenates str1 and str2
  str3 = str1 + str2;
cout<< "str1 + str2 : " << str3 <<endl;

  // total length of str3 after concatenation
```

```
len = str3.size();
cout<< "str3.size() :  " <<len<<endl;

  return 0;
}
```

**OUTPUT:**

```
str3 : Hello
str1 + str2 : HelloWorld
str3.size() :  10
```

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

## Lab Manual

**Title of Course: Computer Organization Lab**
**Course Code: EE594B**
**L-T-P Scheme: 0-0-3** **Course Credits: 2**

**Objective:**
1. Understand the architecture of a modern computer with its various processing units.
2. To learn and understand IC of basic gates.
3. To provide an efficient understanding of the Hardware, design complete circuit.

**Learning Outcomes:** The students will have a detailed knowledge of the concept of IC
**1.** Students can understand the architecture of modern computer.
2. They can analyze the Performance of a computer using performance equation
3. Students can calculate the effective address of an operand by addressing modes
4. They can understand how computer stores positive and negative numbers.
5. Understanding of how a computer performs arithmetic operation of positive and negative numbers.
6. Understanding of how computer stores floating point numbers in IEEE 754 standard.
7. Students can understand how cache mapping occurs in computer and can solve various problems related to this.
8. Secondary storage organization and problem solving

**Course Contents:**

**Unit –I: Basic gates**

Study about logic gates and verify their truth tables. XOR ( IC 7486), OR (IC 7432),NOT (IC 7404), AND (IC 7408),NAND (IC 7400), etc. Also implementation basic gates using universal gate (NAND).

**Unit –II: Half adder, Full Adder**

Implement Half and Full Adder using basic gates and check with the following truth table. Half Adder and Full Adder circuits is explained with their truth tables in this article. Design of Full Adder using Half Adder circuit is also shown. Single-bit Full Adder circuit and Multi-bit addition using Full Adder

**Unit –III: Half Substractor, Full Substractor.**

Implement Half and Full Adder using basic gates and check with the following truth table. Half Subtractor is used for subtracting one single bit binary digit from another single bit binary digit. Full Subtractor, A logic Circuit Which is used for Subtracting Three Single bit Binary digit is known as Full Subtractor

**Unit –IV: 4-bit parallel Binary adder and substractor.**

The arithmetic addition of two binary digits, together with an input carry from a previous stage. The serial addition method uses only one full-adder circuit and a storage device to hold the generated output carry and sum.

**Unit –V: BCD adder**

The arithmetic addition of two decimal digits in BCD, together with an input carry from a previous stage. Since each input digit does not exceed 9, the output sum cannot be greater than 19, the 1 in the sum being an input carry.

**Unit –VI: 8 to 1 Multiplexer unit (MUX)**
It transfer a large number of information units over a smaller number of channels, (usually one channel) under the control of selection signals. Multiplexer means many to one. A multiplexer is a circuit with many inputs but only one output.

**Unit –VII: DEMULTIPLEXER**
**It** perform the opposite function of multiplexers.

**Unit –VIII: BCD to 7 segment decoder**
Using digital kit implement Digital number (0,1,2,3,4,5,6,7,8,9)

**Unit –IX: BCD TO EXCESS 3CODE CONVERTOR**

The excess-3 code digit is obtained by adding three to the corresponding BCD digit.

**Unit –X: FLIP FLOP**
S-R Flip Flop, J-K Flip Flop, T Flip Flop, T Flip Flop

**Unit –X: Design a composite ALU.**

Implement Airthmatic Logic Unit Arithmetic operations are like addition ,substraction, multiplication, and division. Logical operations are like and, or nand, nor ,not operations on bits

**Text Book:**
**1.** David A. Patterson, John L. Hennessy, “Computer Organization and Design”, Elsevier.

**References:**
**1.** S.Salivahanan & S.Arivazhagan, **“**Digital Circuits and Design”, VIKAS publishing house PVT LTD

**Recommended Systems/ Software Requrements:**
1. Trainer kit
2. IC (Integrated Circuit)
3. Wire/ Probes

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

## Lab Manual

# LIST OF EXPERIMENTS

1. Realization of the basic gates (AND, OR, NOT) and universal gates (NAND, NOR).
2. Design and implementation of basic gates using universal gate (NAND).
3. Design and implementation of half adder.
4. Design and implementation of full adder.
5. Design and implementation of half substractor.
6. Design and implementation of full substractor.
7. Design of a 4-bit parallel Binary adder circuit using the IC-Chip 7483.
8. Design of an Adder/Subtractor composite unit circuit using the IC-Chip 7483.
9. Design a BCD adder using two 7483 IC chip.
10. Design an 8 to 1 Multiplexer unit (MUX) using basic gates and using IC 74151
11. Design an 8 to 1 Multiplexer unit (MUX) using basic gates and using IC 74153.
12. Design and implementation of DEMULTIPLEXER .
13. Design of a BCD to 7 segment decoder.
14. Design and implementation of BCD TO EXCESS 3CODE CONVERTOR
15. Design and implementation of SR LATCH ,SR FLIP FLOP AND JK FLIP FLOP.
16. Use a multiplexer unit to design a composite ALU.

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
## Lab Manual

### EXPERIMENT NO.-1

**AIM:-** To study about logic gates and verify their truth tables.

**APPARATUS REQUIRED:** IC 7486, IC 7432, IC 7408, IC 7400, etc.

**THEORY:**

Circuit that takes the logical decision and the process are called logic gates. Each gate has one or more input and only one output. OR, AND and NOT are basic gates. NAND, NOR and X-OR are known as universal gates. Basic gates form these gates.

**AND GATE:**

The AND gate performs a logical multiplication commonly known as AND function. IC 7408 The output is high when both the inputs are high. The output is low level when any one of the inputs is low.

**OR GATE:**

The OR gate performs a logical addition commonly known as OR function. IC 7432. The output is high when any one of the inputs is high. The output is low level when both the inputs are low.

**NOT GATE:**

The NOT gate is called an inverter. IC 7404 The output is high when the input is low. The output is low when the input is high.

**NAND GATE:**

The NAND gate is a contraction of AND-NOT. IC 7400. The output is high when both inputs are low and any one of the input is low .The output is low level when both inputs are high.

**NOR GATE:**

The NOR gate is a contraction of OR-NOT. IC7402. The output is high when both inputs are low. The output is low when one or both inputs are high.

**X-OR GATE:**

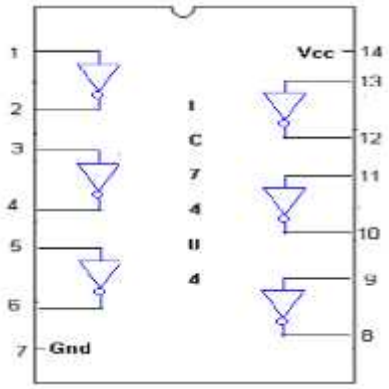
The output is high when any one of the inputs is high. IC 7486 The output is low when both the inputs are low and both the inputs are high.

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

## Lab Manual

**LOGIC GATES SYMBOL TRUTH TABLE AND PIN DIAGRAM:**

**NOT GATE**

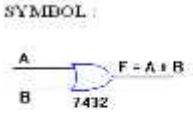



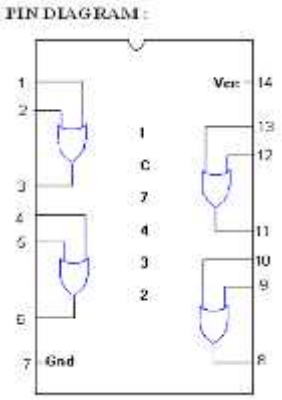

TRUTH TABLE :

| A | $\overline{A}$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

**OR GATE:**

**SYMBOL:** **PIN DIAGRAM:**



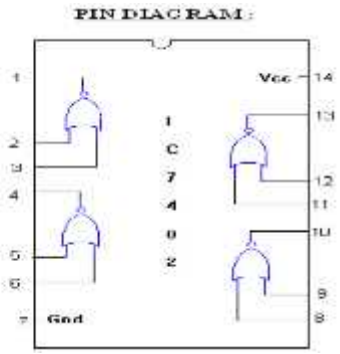




TRUTH TABLE

| A | B | A+B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

---

NOR GATE:

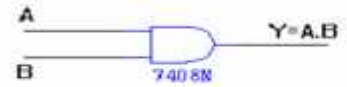

TRUTH TABLE

| A | B | $\overline{A+B}$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

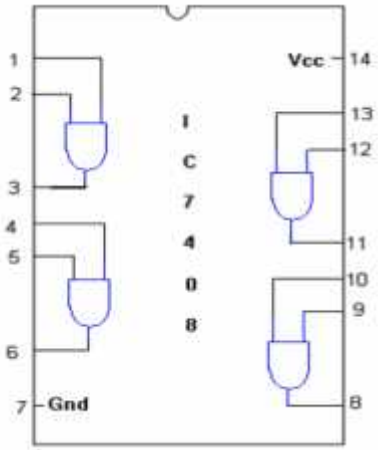
## Lab Manual

**AND GATE:**




TRUTH TABLE

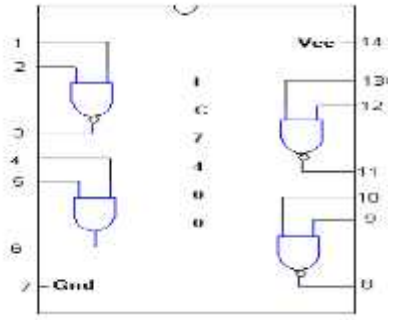
| A | B | A.B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |




**NAND GATE:**


TRUTH TABLE

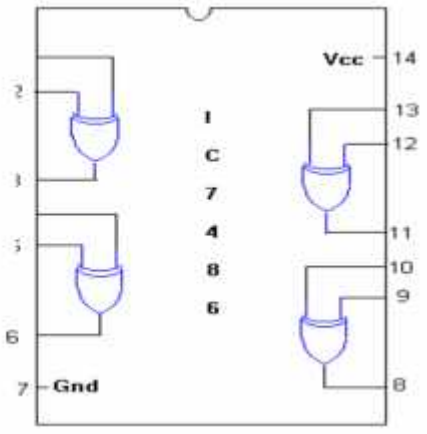
| A | B | $\overline{A.B}$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |




**EX-OR GATE:**


TRUTH TABLE :

| A | B | $\overline{A}B + A\overline{B}$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |



**PROCEDURE:**

1. Check the components for their working.
2. Insert the appropriate IC into the IC base.
3. Make connections as shown in the circuit diagram.
4. Provide the input data via the input switches and observe the output on output LEDs.

**CONCLUSION:**

A logic gate is an elementary building block of a digital circuit. Most logic gates have two inputs and one output. At any given moment, every terminal is in one of the two binary conditions low (0) or high (1), represented by different voltage levels. The logic state of a terminal can, and generally does, change often, as the circuit processes data. In most logic gates, the low state is approximately zero volts (0 V), while the high state is approximately five volts positive (+5 V).

**PRECAUTIONS:**

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
## Lab Manual

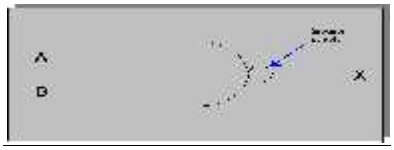
### EXPERIMENT NO:2

**AIM:** Design and implementation of basic gates using universal gate (NAND).

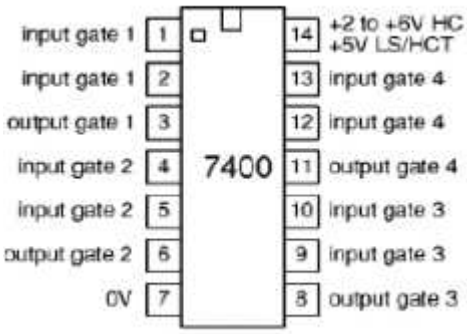
**APPRATUS REQUIRED: IC 7400**

**THEORY:** The NAND Gate:

The NAND, which is composed of two or more inputs and a single output, is a very popular logic element because it may be used as a universal function. That is, it may be employed to construct an inverter, an AND gate, an OR gate, or any combination of theses functions. The term NAND is formed by the concatenation NOT-AND and implies an AND function with an inverted output. The standard symbol for the NAND gate is shown in Figure 1-7 and its truth table listed in Table 1-4. The logical operation of the NAND gate is such that the output is LOW (0) only when all the inputs are HIGH (1).




PIN DIAGRAM:




| INPUT | | OUTPUT |
|---|---|---|
| A | B | Y |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

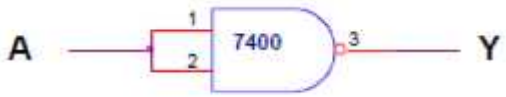
Circuit Diagrams

Truth Table

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

## Lab Manual

Not gate

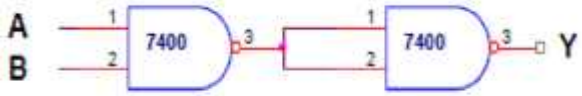



| INPUT A | OUTPUT Y |
|---|---|
| 0 | 1 |
| 1 | 0 |
| | |

Truth Table

---

AND gate




| INPUT | | OUTPUT Y |
|---|---|---|
| A | B | |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

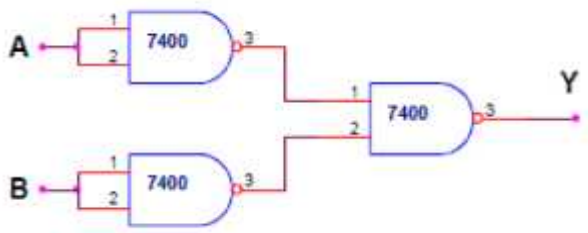
Truth Table

---

OR gate

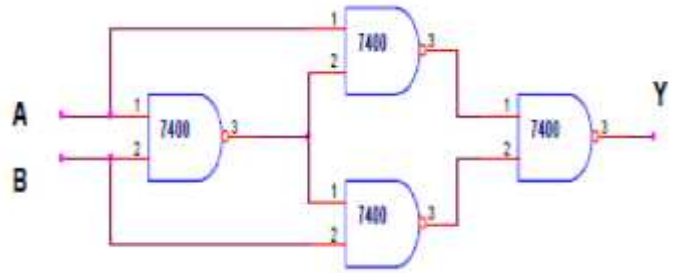
| INPUT | | OUTPUT Y |
|---|---|---|
| A | B | |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |

| | | |
|---|---|---|
| **1** | **1** | **1** |

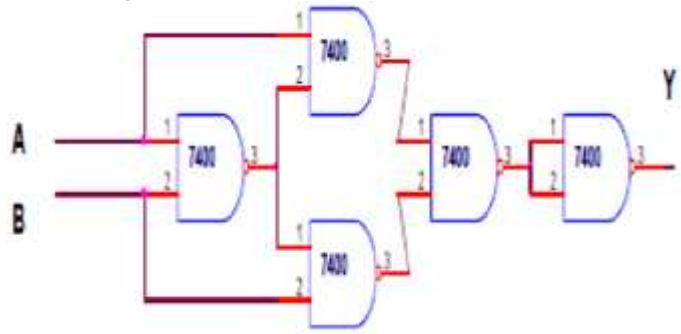



| INPUT | | OUTPUT |
|---|---|---|
| **A** | **B** | **Y** |
| **0** | **0** | **0** |
| **0** | **1** | **1** |
| **1** | **0** | **1** |
| **1** | **1** | **0** |

Ex-OR gate




Ex-NOR gate




| INPUT | | OUTPUT |
|---|---|---|
| **A** | **B** | **Y** |
| **0** | **0** | **1** |
| **0** | **1** | **0** |
| **1** | **0** | **0** |
| **1** | **1** | **1** |

**PROCEDURE:**

1. Connect the logic gates as shown in the diagrams using IC 7400 NAND gate.

2. Feed the logic signals 0 or 1 from the logic input switches in different combinations at the inputs A & B.
3. Monitor the output using logic output LED indicators.
4. Repeat steps 1 to 3 for NOT, AND, OR, EX – OR & EX-NOR operations and compare the outputs with the truth tables.

## CONCLUSION:

A logic gate is an elementary building block of a digital circuit. Most logic gates have two inputs and one output. At any given moment, every terminal is in one of the two binary conditions low (0) or high (1), represented by different voltage levels. The logic state of a terminal can, and generally does, change often, as the circuit processes data. In most logic gates, the low state is approximately zero volts (0 V), while the high state is approximately five volts positive (+5 V).

## PRECAUTIONS:

1. All the connections should be made properly.
2. IC should not be reversed.

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
## Lab Manual

### EXPERIMENT NO:3

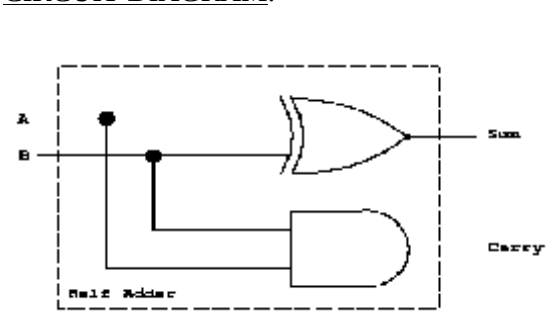
**AIM:** Design and implementation of HALF ADDER.

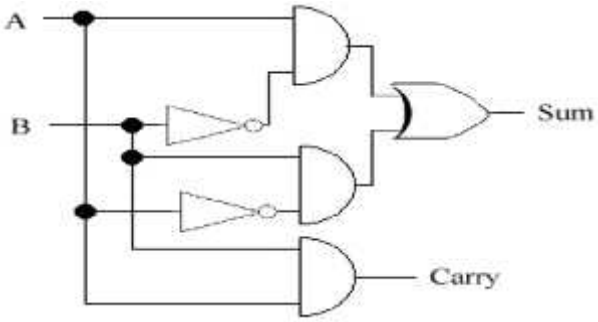
**APPRATUS REQUIRED: IC 7486, IC 7432, IC 7408, IC 7404, IC 7400, etc.**

**THEORY:**

**Half-Adder:** A combinational logic circuit that performs the addition of two data bits, A and B, is called a half-adder. Addition will result in two output bits; one of which is the sum bits, and the other is the carry bit, C.

CIRCUIT DIAGRAM:




TRUTH TABLE:

| Truth Table of a Half-Adder | | | |
|---|---|---|---|
| Inputs | | Outputs | |
| | | | |
| A | B | Sum S | Carry C |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| | | | |

**Logical Expression:-**

$S_1 = \bar{A}B + A\bar{B} = A \oplus B$

carry $C_o = A$

**PROCEDURE:**

1. Check the components for their working.
2. Insert the appropriate IC into the IC base.

3. Make connections as shown in the circuit diagram.
4. Provide the input data via the input switches and observe the output on output LEDs.

## CONCLUSION:

Thus, for example, a binary input of 101 results in an output of 1 + 0 + 1 = 10 (decimal number 2). The carry-out represents bit one of the result, while the sum represents bit zero. Likewise, a half adder can be used as a 2:2 lossy compressor, compressing four possible inputs into three possible outputs.

## PRECAUTIONS:

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
## Lab Manual

### EXPERIMENT NO:4

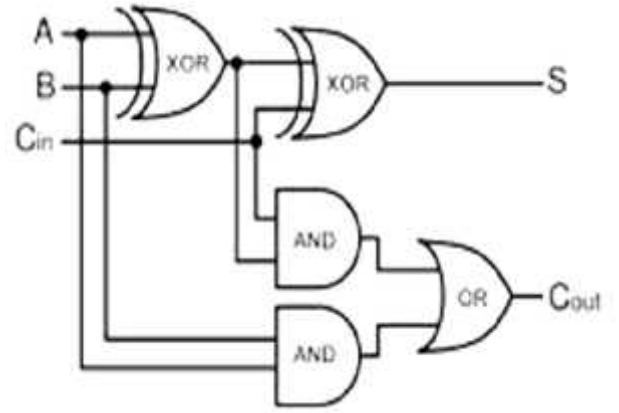
**AIM:** Design and implementation of FULL ADDER.

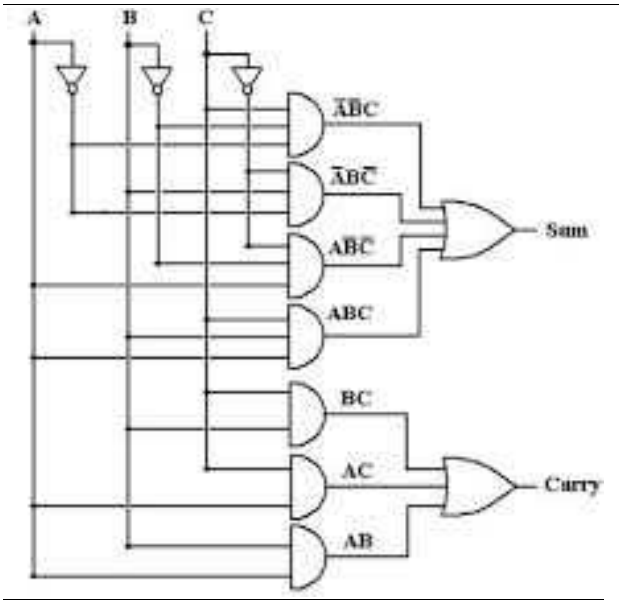
**APPRATUS REQUIRED:** IC 7486, IC 7432, IC 7408, IC 7404, IC 7400, etc.

**THEORY:**

**Full-Adder***:* The half-adder does not take the carry bit from its previous stage into account. This carry bit from its previous stage is called carry-in bit. A combinational logic circuit that adds two data bits, A and B, and a carry-in bit, Cin , is called a full-adder.

CIRCUIT DIAGRAM:



**TRUTH TABLE**:

| Truth Table of a Full-Adder | | | | |
|---|---|---|---|---|
| Inputs | | | Outputs | |
| Augend Bit A | Addend Bit B | Carry input $C_{in}$ | Sum S | Carry C |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

**BOOLEAN EXPRESSIONS:**
**S= A ⊕ B ⊕ C**
**C=A B + B Cin + A Cin**

**PROCEDURE:**
1. Check the components for their working.
2. Insert the appropriate IC into the IC base.
3. Make connections as shown in the circuit diagram.
4. Provide the input data via the input switches and observe the output on output LEDs.

**CONCLUSION:**
Thus, for example, a binary input of 101 results in an output of 1 + 0 + 1 = 10 (decimal number 2). The carry-out represents bit one of the result, while the sum represents bit zero.

**PRECAUTIONS:**

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged.

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
## Lab Manual

### EXPERIMENT NO:5

**AIM:** Design and implementation of HALF SUBSTRACTOR.

**APPRATUS REQUIRED:** IC 7486, IC 7432, IC 7408, IC 7404, IC 7400, etc.

**THEORY:**

**Half Subtractor***:* Subtracting a single-bit binary value B from another A (i.e. A -B ) produces a difference bit D and a borrow out bit B-out. This operation is called half subtraction and the circuit to realize it is called a half subtractor.

CIRCUIT DIAGRAM:

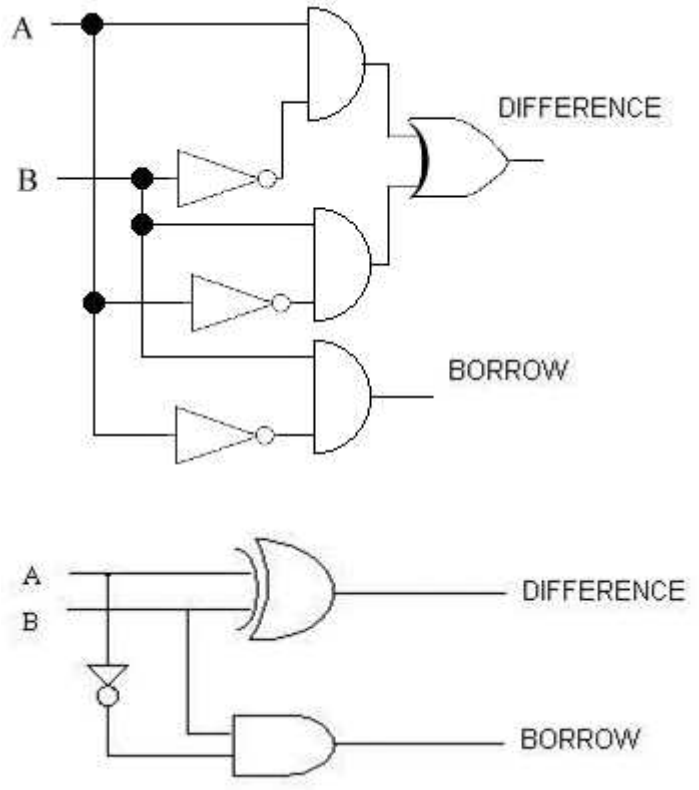



Fig: half substactor.

TRUTH TABLE:

| Truth Table of a Half-Subtractor | | | |
|---|---|---|---|
| **Inputs** | | **Outputs** | |
| **Minuend A** | **Subtrahend B** | **Difference D** | **Borrow Bout** |
| **0** | **0** | **0** | **0** |
| **0** | **1** | **1** | **1** |
| **1** | **0** | **1** | **0** |
| **1** | **1** | **0** | **0** |

BOOLEAN EXPRESSION:

$D \quad ., D = \bar{A}B + A\bar{B} = A \oplus B$

$B_o \quad = \bar{A}B$

**PROCEDURE:**

1. Check the components for their working.
2. Insert the appropriate IC into the IC base.
3. Make connections as shown in the circuit diagram.
4. Provide the input data via the input switches and observe the output on output LEDs.

**CONCLUSION:**

The Binary Subtractor is another type of combinational arithmetic circuit that is the opposite of the Binary Adder we looked at in a previous tutorial. As their name implies, a Binary Subtractor is a decision making circuit that subtracts two binary numbers from each other, for example, X – Y to find the resulting difference between the two numbers.

**PRECAUTIONS:**

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged.

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
## Lab Manual

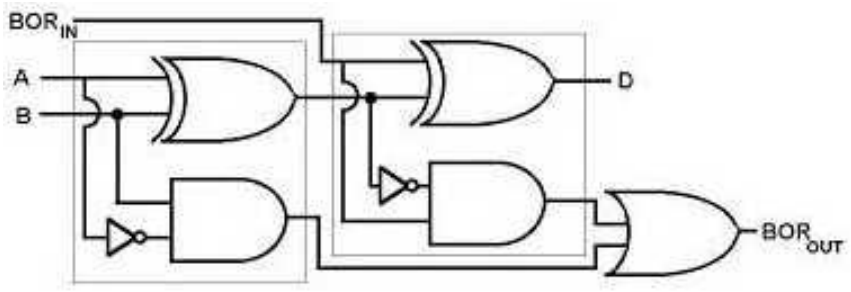
### EXPERIMENT NO:6

**AIM:** Design and implementation of FULL SUBSTRACTOR.

**APPRATUS REQUIRED:** IC 7486, IC 7432, IC 7408, IC 7404, IC 7400, etc.

**THEORY:** Full Subtractor: Subtracting two single-bit binary values, B, Cin from a single-bit value A produces a difference bit D and a borrow out Br bit. This is called full subtraction.

CIRCUIT DIAGRAM:




TRUTH TABLE:

| Truth Table of a Full-Subtractor | | | | |
|---|---|---|---|---|
| Inputs | | | Outputs | |
| Minuend Bit A | Subtrahend Bit B | Borrow input $B_{in}$ | Difference D | Borrow Out $B_{out}$ |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

BOOLEAN EXPRESSION: $D = \overline{A}\overline{B}B_{in} + \overline{A}B\overline{B_{in}} + A\overline{B}\overline{B_{in}} + ABB_{in} = A \oplus B \oplus B_{in}$

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

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

**PROCEDURE:**

1. Check the components for their working.
2. Insert the appropriate IC into the IC base.
3. Make connections as shown in the circuit diagram.
4. Provide the input data via the input switches and observe the output on output LEDs.

## CONCLUSION:

The Binary Subtractor is another type of combinational arithmetic circuit that is the opposite of the Binary Adder we looked at in a previous tutorial. As their name implies, a Binary Subtractor is a decision making circuit that subtracts two binary numbers from each other, for example, X – Y to find the resulting difference between the two numbers.

## PRECAUTIONS:

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged.

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
## Lab Manual

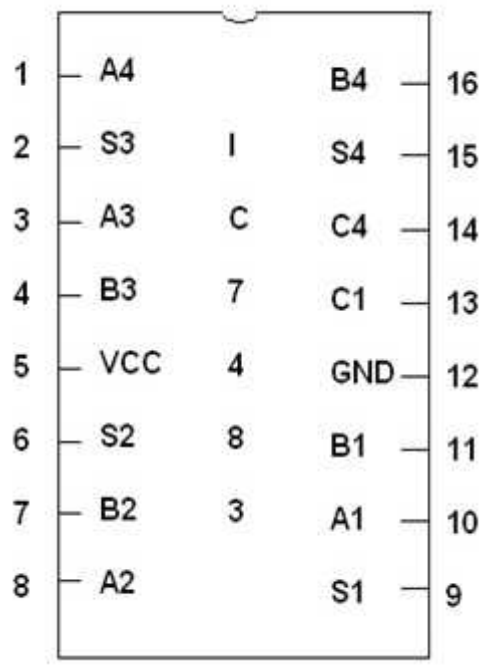
### EXPERIMENT NO-07

**AIM:** Design of a 4-bit parallel Binary adder circuit using the IC-Chip 7483.

**APPRATUS REQUIRED:** IC 7483 etc.

**THEORY:** 4-bit parallel Binary adder circuit using the IC-Chip 7483.
Consider the arithmetic addition of two binary digits, together with an input carry from a previous stage. The serial addition method uses only one full-adder circuit and a storage device to hold the generated output carry and sum. The parallel method uses n full-adder circuit. A binary parallel adder is a digital function that produces the arithmetic sum of two binary numbers in parallel.

**PIN DIAGRAM FOR IC 7483:**




Here $A_1$, $A_2$, $A_3$, $A_4$ and $B_1$, $B_2$, $B_3$, $B_4$ are the 4+4 =8 Input. $S_1$, $S_2$, $S_3$, $S_4$ are the 4 out put where the sum value is stored. So put any two 4 digit binary number and get the sum of them.

**PROCEDURE:**

- Check all the components for their working.
- Insert the appropriate IC into the IC base.
- Make connections as shown in the circuit diagram.
- Verify the Truth Table and observe the outputs.

## CONCLUSION:

Parallel adder is a combinatorial circuit (not clocked, does not have any memory and feedback) adding every bit position of the operands in the same time.

**PRECAUTIONS:**

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged.

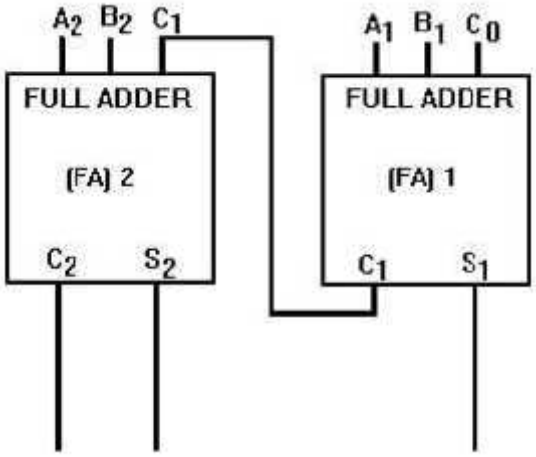
# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
## Lab Manual

### EXPERIMENT NO-08

**AIM:** 4 Bit binay adder/substractor composite unit.

**APPRATUS REQUIRED:**

**THEORY** : The Full adder can add single-digit binary numbers and carries. The largest sum that can be obtained using a full adder is 112. Parallel adders can add multiple-digit numbers. If full adders are placed in parallel, we can add two- or four-digit numbers or any other size desired. Figure below uses STANDARD SYMBOLS to show a parallel adder capable of adding two, two-digit binary numbers The addend would be on A inputs, and the augend on the B inputs. For this explanation we will assume there is no input to C0 (carry from a previous circuit)




To add 102 (addend) and 012 (augend), the addend inputs will be 1 on A2 and 0 on A1. The augend inputs will be 0 on B2 and 1 on B1. Working from right to left, as we do in normal addition, let's calculate the outputs of each full adder. With A1 at 0 and B1 at 1, the output of adder1 will be a sum (S1) of 1 with no carry (C1). Since A2 is 1 and B2 is 0, we have a sum (S2) of 1 with no carry (C2) from adder1. To determine the sum, read the outputs (C2, S2, and S1) from left to right. In this case, C2 = 0, S2 = 1, and S1 = 1. The sum, then, of 102 and 012is 0112. To add four bits we require four full adders arranged in parallel. IC 7483 is a 4- bit parallel adder whose pin diagram is shown.

CIRCUIT DIAGRAM OF ADDER AND SUBSTRACTOR AND 7483 IC PIN DIAGRAM :

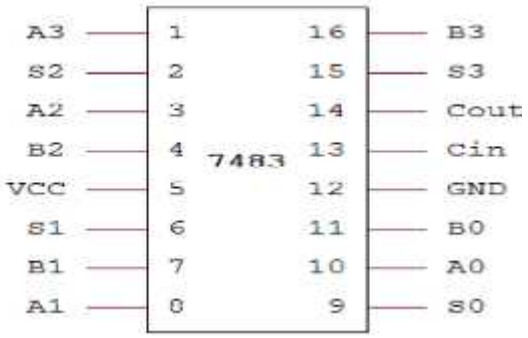


Fig-7483 ic pin diagram.

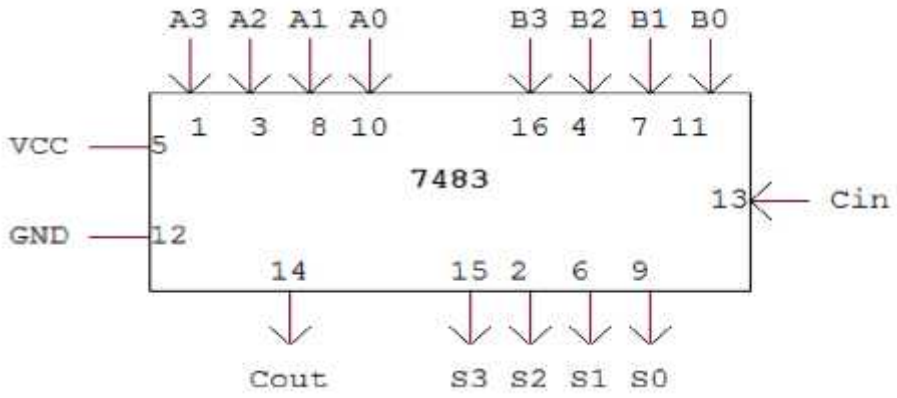



Fig-adder circuit.

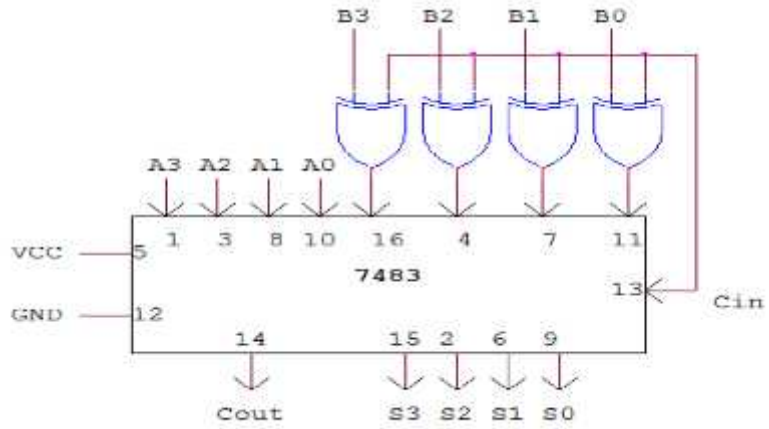
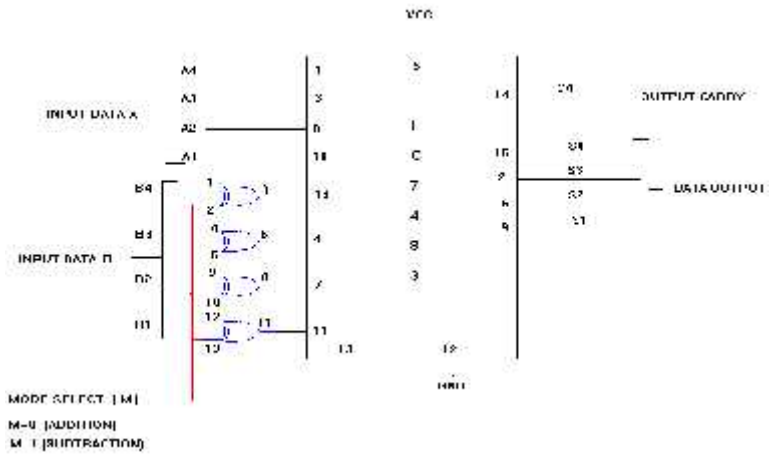



Fig-substractor circuit.

**4-BIT BINARY ADDER/SUBTRACTOR**

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

## Lab Manual



Here $A_1$, $A_2$, $A_3$, $A_4$ and $B_1$, $B_2$, $B_3$, $B_4$ are the 8 Input. $A_1$, $A_2$, $A_3$, $A_4$ data set directly connected to the IC 7483 and $B_1$, $B_2$, $B_3$, $B_4$ fast connected to the XOR gates then the out put of XOR gates connected to 7483. Now XOR gates other I/P line are connected to the pin no 13 of IC 7483, If the value of pin no 13 is '0' means ADDITION and '1' means SUBTRACTION $S_1$, $S_2$, $S_3$, $S_4$ are the 4 out put where the sum/ borrow value is stored. So put any two 4 digit binary number and get the sum of them.

**PROCEDURE:**

1.for adder-

- Check all the components for their working.
- Insert the appropriate IC into the IC base.
- Make connections as shown in the circuit diagram.
- Apply augend and addend bits on A and B and cin=0.
- Verify the results and observe the outputs.

2.for substractor-

- Check all the components for their working.
- Insert the appropriate IC into the IC base.
- Make connections as shown in the circuit diagram.
- Apply Minuend and subtrahend bits on A and B and cin=1.
- Verify the results and observe the outputs.

**CONCLUSION:**

Binary adder is one of the basic combinational logic circuits. The outputs of a combinational logic circuit depend on the present input only. In other words, outputs of combinational logic circuit do not depend upon any previously applied inputs. It does not require any memory like component. Binary adder is one of the basic combinational logic circuits as present state of input variables.

**PRECAUTION:**

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged.

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

## Lab Manual

### EXPERIMENT NO-09

**AIM:**Design and implementation of 4 BCD adder using 7483 ic.

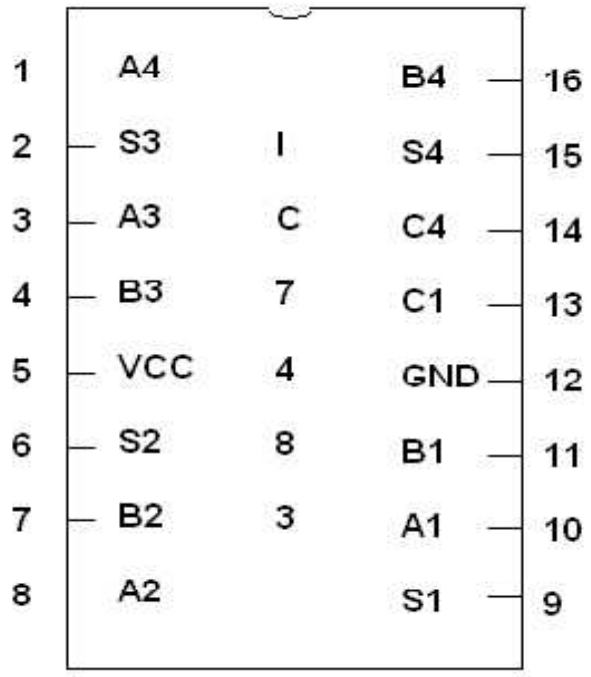
**APPRATUS REQUIRED:** IC 7483, IC 7432, IC 7408, IC 7400, etc.

**THEORY: 4 BIT BCD ADDER:**

Consider the arithmetic addition of two decimal digits in BCD, together with an input carry from a previous stage. Since each input digit does not exceed 9, the output sum cannot be greater than 19, the 1 in the sum being an input carry. The output of two decimal digits must be represented in BCD and should appear in the form listed in the columns. A BCD adder that adds 2 BCD digits and produce a sum digit in BCD. The 2 decimal digits, together with the input carry, are first added in the top 4 bit adder to produce the binary sum.

**CIRCUIT DIAGRAM:**

**PIN DIAGRAM FOR IC 7483:**

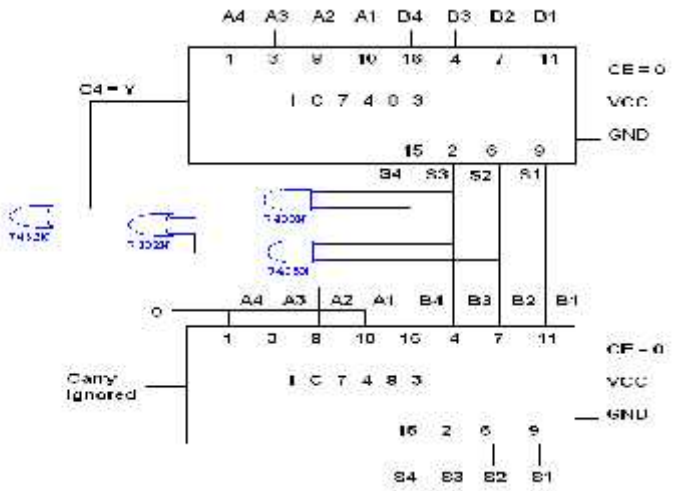


# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

## Lab Manual

**LOGIC DIAGRAM:**

**BCD ADDER:**




TRUTH TABLE:

| BCD SUM | | | | CARRY |
|---|---|---|---|---|
| **S4** | **S3** | **S2** | **S1** | **C** |
| **0** | **0** | **0** | **0** | **0** |
| **0** | **0** | **0** | **1** | **0** |
| **0** | **0** | **1** | **0** | **0** |
| **0** | **0** | **1** | **1** | **0** |
| **0** | **1** | **0** | **0** | **0** |
| **0** | **1** | **0** | **1** | **0** |
| **0** | **1** | **1** | **0** | **0** |
| **0** | **1** | **1** | **1** | **0** |
| **1** | **0** | **0** | **0** | **0** |
| **1** | **0** | **0** | **1** | **0** |
| **1** | **0** | **1** | **0** | **1** |
| **1** | **0** | **1** | **1** | **1** |
| **1** | **1** | **0** | **0** | **1** |
| **1** | **1** | **0** | **1** | **1** |
| **1** | **1** | **1** | **0** | **1** |
| **1** | **1** | **1** | **1** | **1** |

**PROCEDURE:**

- Check all the components for their working.
- Insert the appropriate IC into the IC base.
- Make connections as shown in the circuit diagram.
- Verify the Truth Table and observe the outputs.

**PRECAUTIONS:**

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged.

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

## Lab Manual

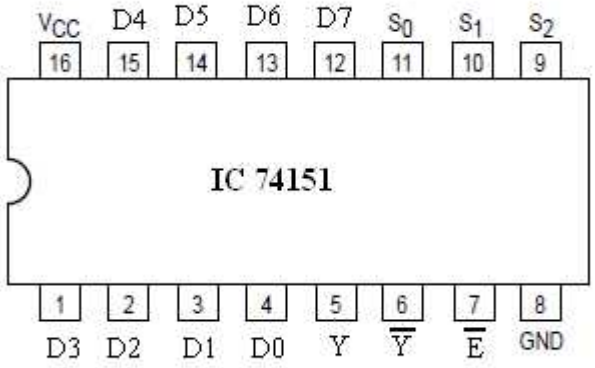
### EXPERIMENT NO:10

**AIM:** Design and implementation of 8:1 MULTIPLEXER .

**APPRATUS REQUIRED: IC 74151**

**THEORY:** Multiplexers are very useful components in digital systems. They transfer a large number of information units over a smaller number of channels, (usually one channel) under the control of selection signals. Multiplexer means many to one. A multiplexer is a circuit with many inputs but only one output. By using control signals (select lines) we can select any input to the output. Multiplexer is also called as data selector because the output bit depends on the input data bit that is selected. The general multiplexer circuit has 2n input signals, n control/select signals and 1 output signal.

CIRCUIT DIAGRAM:




TRUTH TABLE:

| Select Data Inputs | | | Output |
|---|---|---|---|
| $S_2$ | $S_1$ | $S_0$ | Y |
| 0 | 0 | 0 | $D_0$ |
| 0 | 0 | 1 | $D_1$ |
| 0 | 1 | 0 | $D_2$ |
| 0 | 1 | 1 | $D_3$ |
| 1 | 0 | 0 | $D_4$ |
| 1 | 0 | 1 | $D_5$ |
| 1 | 1 | 0 | $D_6$ |
| 1 | 1 | 1 | $D_7$ |

**PROCEDURE:**

- Check all the components for their working.
- Insert the appropriate IC into the IC base.
- Make connections as shown in the circuit diagram.
- Verify the Truth Table and observe the outputs.

**CONCLUSION:**

The truth table for an 8-to1 multiplexer is given below with eight combinations of inputs so as to generate each output corresponds to input.
For example, if S2= 0, S1=1 and S0=0 then the data output Y is equal to D2. Similarly the data outputs D0 to D7 will be selected through the combinations of S2, S1

**PRECAUTIONS:**

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged.

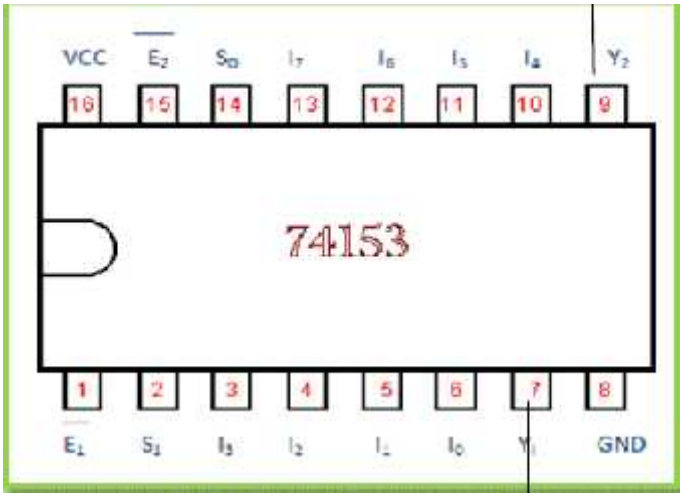
# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
## Lab Manual

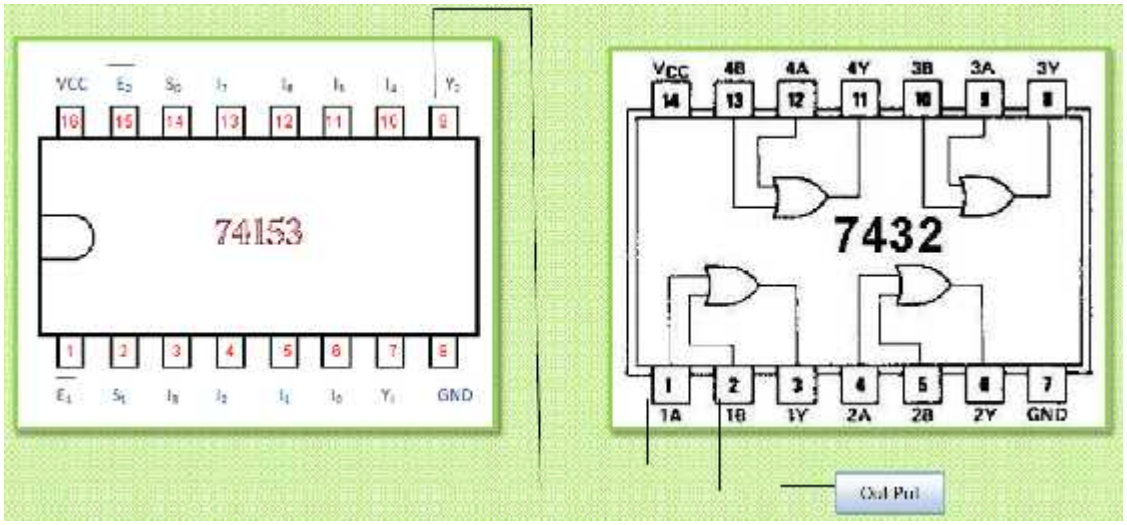
**AIM:** Design and implementation of 8:1 MULTIPLEXER .

**APPRATUS REQUIRED: IC 74153, IC 7432**

**THEORY:** Multiplexers are very useful components in digital systems. They transfer a large number of information units over a smaller number of channels, (usually one channel) under the control of selection signals. Multiplexer means many to one. A multiplexer is a circuit with many inputs but only one output. By using control signals (select lines) we can select any input to the output. Multiplexer is also called as data selector because the output bit depends on the input data bit that is selected. The general multiplexer circuit has 2n input signals, n control/select signals and 1 output signal.

CIRCUIT DIAGRAM:



TRUTH TABLE:

| $E_1$ | $E_2$ | $S_1$ | $S_0$ | $I_0$ | $I_1$ | $I_2$ | $I_3$ | $I_4$ | $I_5$ | $I_6$ | $I_7$ | Y (O/P) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | X | X | X | X | X | X | X | ON |
| 1 | 0 | 0 | 0 | 0 | X | X | X | X | X | X | X | OFF |
| 1 | 0 | 0 | 1 | X | 1 | X | X | X | X | X | X | ON |
| 1 | 0 | 0 | 1 | X | 0 | X | X | X | X | X | X | OFF |
| 1 | 0 | 1 | 0 | X | X | 1 | X | X | X | X | X | ON |
| 1 | 0 | 1 | 0 | X | X | 0 | X | X | X | X | X | OFF |
| 1 | 0 | 1 | 1 | X | X | X | 1 | X | X | X | X | ON |
| 1 | 0 | 1 | 1 | X | X | X | 0 | X | X | X | X | OFF |
| 0 | 1 | 0 | 0 | X | X | X | X | 1 | X | X | X | ON |
| 0 | 1 | 0 | 0 | X | X | X | X | 0 | X | X | X | OFF |
| 0 | 1 | 0 | 1 | X | X | X | X | X | 1 | X | X | ON |
| 0 | 1 | 0 | 1 | X | X | X | X | X | 0 | X | X | OFF |
| 0 | 1 | 1 | 0 | X | X | X | X | X | X | 1 | X | ON |
| 0 | 1 | 1 | 0 | X | X | X | X | X | X | 0 | X | ON |
| 0 | 1 | 1 | 1 | X | X | X | X | X | X | X | 1 | ON |
| 0 | 1 | 1 | 1 | X | X | X | X | X | X | X | 0 | OFF |

**PROCEDURE:**

- Check all the components for their working.
- Insert the appropriate IC into the IC base.
- Make connections as shown in the circuit diagram.
- Verify the Truth Table and observe the outputs.

**CONCLUSION:**

The truth table for an 8-to1 multiplexer is given below with eight combinations of inputs so as to generate each output corresponds to input.

**PRECAUTIONS:**

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged.

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
## Lab Manual

### EXPERIMENT NO:12

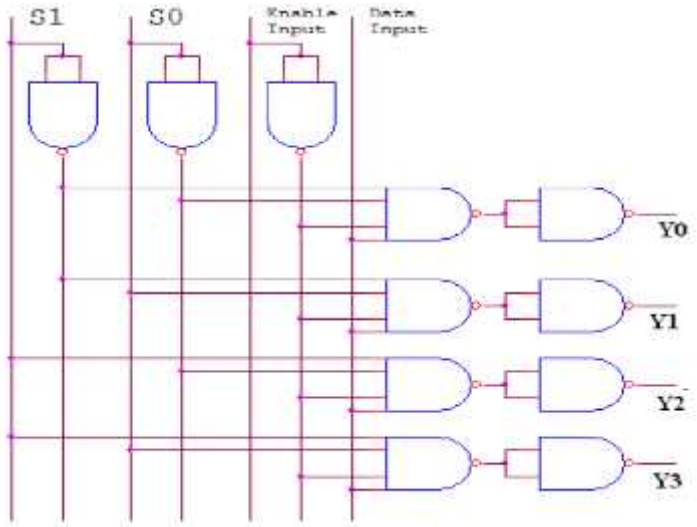
**AIM:**Design and implementation of DEMULTIPLEXER .

**APPRATUS REQUIRED:**

**THEORY** : De-multiplexers perform the opposite function of multiplexers. They transfer a small number of information units (usually one unit) over a larger number of channels under the control of selection signals. The general de-multiplexer circuit has 1 input signal, n control/select signals and 2n output signals. De-multiplexer circuit can also be realized using a decoder circuit with enable.

CIRCUIT DIAGRAM:

DE-MUX USING NAND GATES:



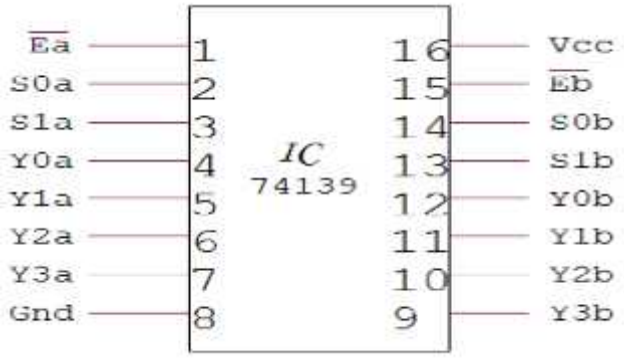

TRUTH TABLE:

| Enable input | Data input | Select inputs | | Outputs | | | |
|---|---|---|---|---|---|---|---|
| E | D | $S_1$ | $S_0$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
| 1 | 0 | X | X | X | X | X | X |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| | | | | | | | |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

IC 74139 (DEMUX) AND TRUTH TABLE:

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

## Lab Manual




| Inputs | | | Outputs | | | |
|---|---|---|---|---|---|---|
| Ea | S1 | S0 | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
| 1 | X | X | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 |

**PROCEDURE:**

1. Check the components for their working.
2. Insert the appropriate IC into the IC base.
3. Make connections as shown in the circuit diagram.
4. Provide the input data via the input switches and observe the output on output LEDs.

**CONCLUSION:**

A **demultiplexer** (or demux) is a device taking a single input signal and selecting one of many data-output-lines, which is connected to the single input. A multiplexer is often used with a complementary **demultiplexer** on the receiving end.

**PRECAUTIONS:**

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged.

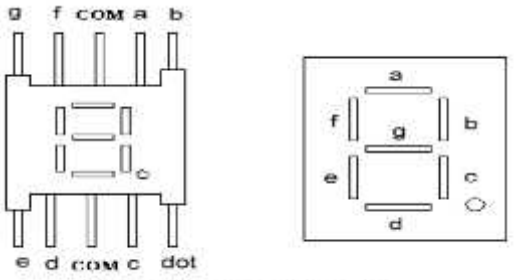
# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
## Lab Manual

### EXPERIMENT NO-13

**AIM:** Design and implementation of BCD TO SEVEN SEGMENT DECODER.

**APPRATUS REQUIRED:** IC7447, 7-Segment display (common anode), Patch chords, resistor (1K_) & IC Trainer Kit .

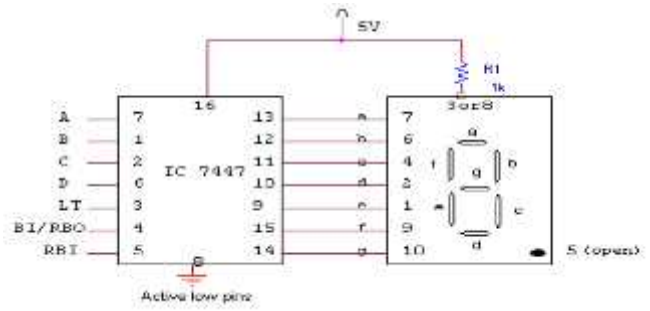
**THEORY:** The Light Emitting Diode (LED) finds its place in many applications in these modern electronic fields. One of them is the Seven Segment Display. Seven-segment displays contains the arrangement of the LEDs in "Eight" (8) passion, and a Dot (.) with a common electrode, lead (Anode or Cathode). The purpose of arranging it in that passion is that we can make any number out of that by switching ON and OFF the particular LED's. Here is the block diagram of the Seven Segment LED arrangement. The Light Emitting Diode (LED), finds its place in many applications in this modern electronic fields. One of them is the Seven Segment Display. Seven-segment displays contains the arrangement of the LEDs in "Eight" (8) passion, and a Dot (.) with a common electrode, lead (Anode or Cathode). The purpose of arranging it in that passion is that we can make any number out of that by switching ON and OFF the particular LED's. Here is the block diagram of the Seven Segment LED arrangement.



Seven-Segment Display

LED's are basically of two types-
Common Cathode (CC) -All the 8 anode legs uses only one cathode, which is common. Common Anode (CA)-The common leg for all the cathode is of Anode type. A decoder is a combinational circuit that connects the binary information from 'n' input lines to a maximum of 2n unique output lines. The IC7447 is a BCD to 7-segment pattern converter. The IC7447 takes the Binary Coded Decimal (BCD) as the input and outputs the relevant 7 segment code.

CIRCUIT DIAGRAM:



TUTH TABLE:

| Decimal Digit | Input lines | | | | Output lines | | | | | | | Display pattern |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | a | b | c | d | e | f | g | |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |  |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |  |
| 2 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |  |
| 3 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |  |
| 4 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |  |
| 5 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |  |
| 6 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |  |
| 7 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |  |
| 8 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |
| 9 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |  |

**PROCEDURE:**

- Check all the components for their working.
- Insert the appropriate IC into the IC base.
- Make connections as shown in the circuit diagram.
- Verify the Truth Table and observe the outputs.

**CONCLUSION:**

A decoder is a combinational circuit which is used to convert a binary or **BCD** (Binary Coded Decimal) number to the corresponding decimal number . It can be a simple binary to decimal decoder or a **BCD to 7 segment** decoder. Another relevant section is the combinational logic circuitry

**PRECAUTIONS:**

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged.

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
## Lab Manual

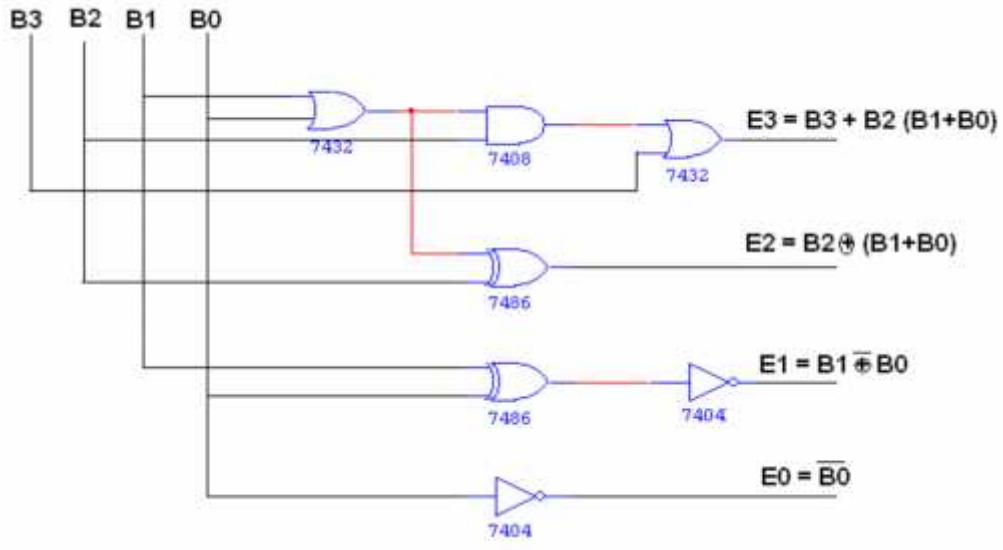
### EXPERIMENT NO:14

**AIM:** Design and implementation of BCD TO EXCESS 3CODE CONVERTOR..

**APPRATUS REQUIRED:** IC 7486, IC 7432, IC 7408, IC 7400, etc.

**THEORY:** Code converter is a combinational circuit that translates the input code word into a new corresponding word. The excess-3 code digit is obtained by adding three to the corresponding BCD digit. To Construct a BCD-to-excess-3-code converter with a 4-bit adder feed BCDcode to the 4-bit adder as the first operand and then feed constant 3 as the second operand.
The output is the corresponding excess-3 code.
To make it work as a excess-3 to BCD converter, we feed excess-3 code as the first operand and then feed 2's complement of 3 as the second operand. The output is the BCD code.

CIRCUIT DIAGRAM:



# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
## Lab Manual

**TRUTH TABLE:**

| BCD | | | | EXCESS 3 | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |

**PROCEDURE:**

1. Check the components for their working.
2. Insert the appropriate IC into the IC base.
3. Make connections as shown in the circuit diagram.
4. Provide the input data via the input switches and observe the output on output LEDs.

**CONCLUSION:**

The Excess-3 BCD system is formed by adding 0011 to each BCD value THE *BCD TO EXCESS 3* CODE CONVERTER

**PRECAUTIONS:**

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged.

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
## Lab Manual

### EXPERIMENT NO:15

**AIM:**To design and implement SR latch, SR flip flop and JK flip flop.

**APPRATUS REQUIRED:** IC 7486, IC 7432, IC 7408, IC 7400, etc.

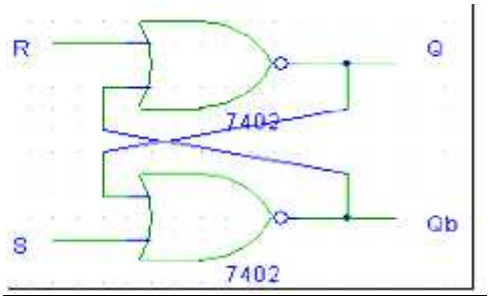
**THEORY:** Logic circuits that incorporate memory cells are called *sequential logic circuits*; their output depends not only upon the present value of the input but also upon the previous values. Sequential logic circuits often require a timing generator (a clock) for their operation. The latch (flip-flop) is a basic bi-stable memory element widely used in sequential logic circuits. Usually there are two outputs, Q and its complementary value. Some of the most widely used latches are listed below.

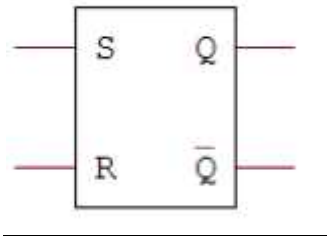
*SR LATCH:*

An S-R latch consists of two cross-coupled NOR gates. An S-R flip-flop can also be design using cross-coupled NAND gates as shown. The truth tables of the circuits are shown below. A clocked S-R flip-flop has an additional clock input so that the S and R inputs are active only when the clock is high. When the clock goes low, the state of flip-flop is latched and cannot change until the clock goes high again. Therefore, the clocked S-R flip-flop is also called "enabled" S-R flip-flop.

CIRCUIT DIAGRAM:

R-S LATCH:




TRUTH TABLE:

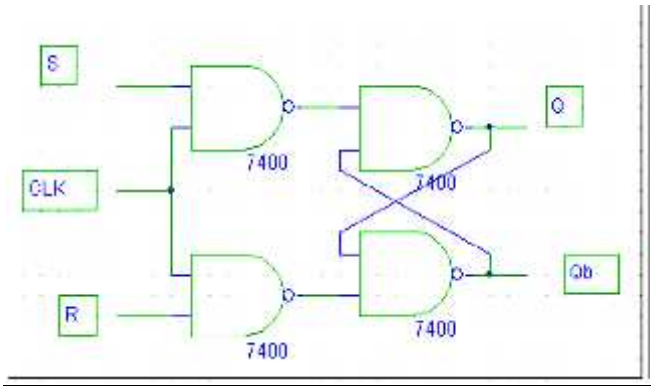
| S | R | Q | Q' |
|---|---|---|---|
| 0 | 0 | NC | NC |

| | | | |
|---|---|---|---|
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | FB | FB |

TRUTH TABLE:

| S | R | Q | Q’ |
|---|---|---|---|
| 0 | 0 | FB | FB |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | NC | NC |

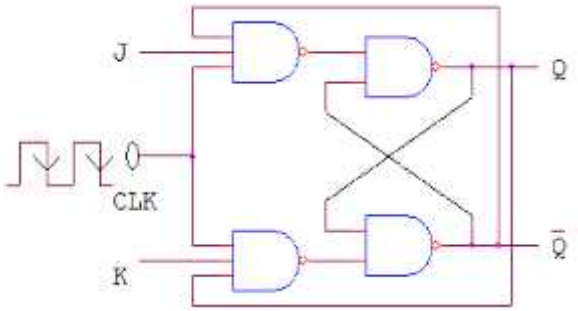
S-R FLIP FLOP:




TRUTH TABLE:

| CLK | S | R | Q | Q’ |
|---|---|---|---|---|
| 0 | 0 | 0 | | |
| 0 | 0 | 1 | | |
| 0 | 1 | 0 | | |
| 0 | 1 | 1 | | |
| 1 | 0 | 0 | | |
| 1 | 0 | 1 | | |
| 1 | 1 | 0 | | |
| 1 | 1 | 1 | | |

J-K FLIP FLOP :
CIRCUIT DIAGRAM AND TRUTH TABLE:




**PROCEDURE:**

- Check all the components for their working.
- Insert the appropriate IC into the IC base.
- Make connections as shown in the circuit diagram.
- Verify the Truth Table and observe the outputs.

**CONCLUSION:**

n **electronics**, a **flip-flop** or latch is a circuit that has two stable states and can be used to store state information. A **flip-flop** is a bistable multivibrator. The circuit can be made to change state by signals applied to one or more control inputs and will have one or two outputs.

**PRECAUTIONS:**

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged.

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
## Lab Manual

### EXPERIMENT NO:16

**AIM:**To design and implement ALU.

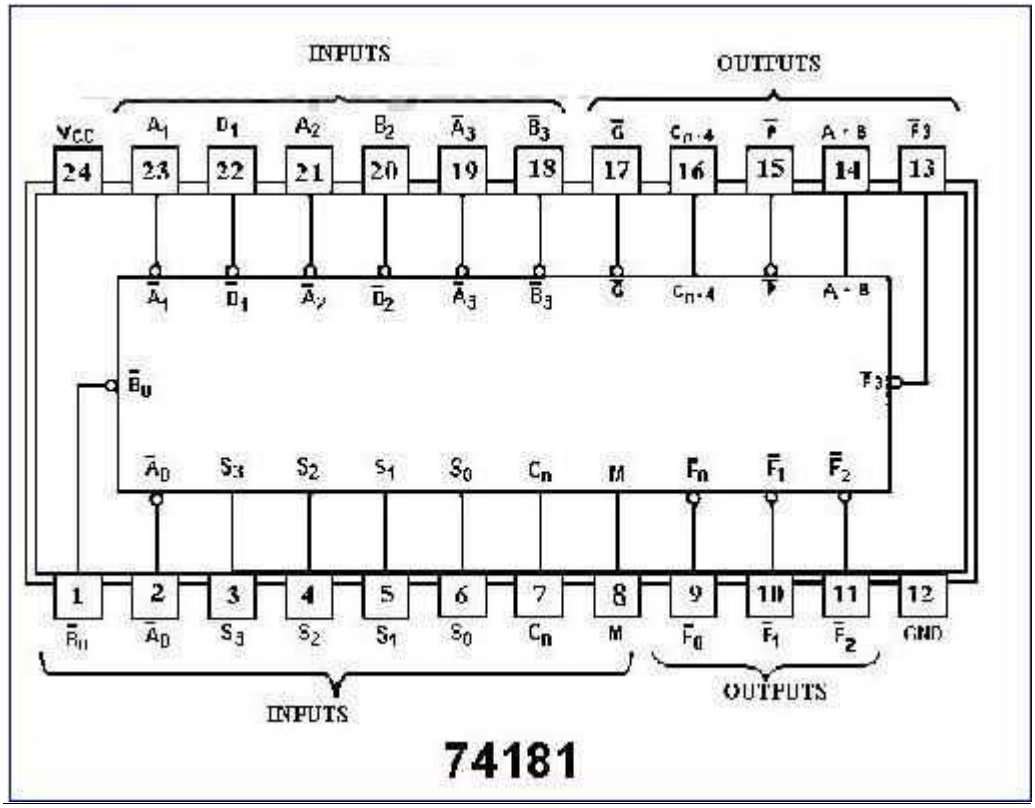
**APPRATUS REQUIRED:** IC 74181

**THEORY:** ALU (AIRTHMATIC LOGIC UNIT) is a circuit which performs arithmetic and logical operations. Arithmetic operations are like addition ,substraction, multiplication, and division. Logical operations are like and, or nand, nor ,not operations on bits. Here we will design the ALU for addition, substration and all logical operations. For this we need to design circuits for all the arithmetic and logical operations we want to perform. All these circuits then will be multiplexed through multiplexers. keeping in mind the complexity of circuit we will multiplexer only some number of circuits. for a particular operation we have to select that particular circuit through multiplexer by selecting appropriate select lines of multiplexer.

CIRCUIT DIAGRAM:

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

## Lab Manual




**PROCEDURE:**

- Check all the components for their working.
- Insert the appropriate IC into the IC base.
- Make connections as shown in the circuit diagram.
- Verify the Truth Table and observe the outputs.

**CONCLUSION:**

The 74181 is a bit slice arithmetic logic unit (ALU), implemented as a 7400 series TTL integrated circuit. The first complete ALU on a single chip,[1] it was used as the arithmetic/logic core in the CPUs of many historically significant minicomputers and other devices.

The 74181 represents an evolutionary step between the CPUs of the 1960s, which were constructed using discrete logic gates, and today's single-chip CPUs or microprocessors.

**PRECAUTIONS:**

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged.

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

## Lab Manual

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

## Lab Manual

**Title of Course: Microprocessors & Microcontrollers lab**
**Course Code: EE594C**
**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

**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 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
Monitor data area 4100 – 5FFF
RAM Expansion 6000 – BFFF

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"

i) Hex key entry "y"
ii) Signal step program (instruction by instruction)

ii) Much a block of memory from a linear block
iii) Register key "SH"

i) Hex key D
ii) Compare 2 memory block

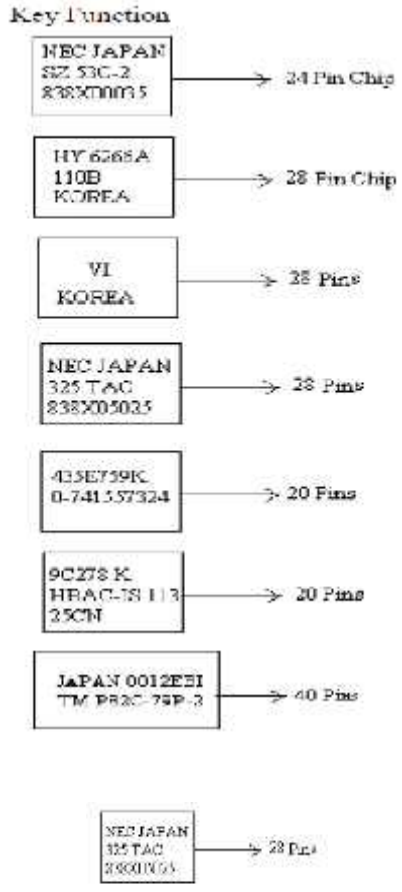
i) Hex key entry 'B'
ii) Check a block from flame
iii) Register key "SPL"

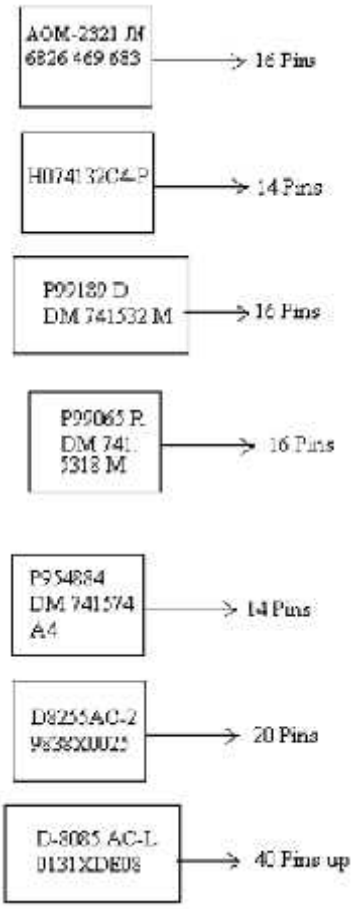
i) Hex key 'E'
ii) Insert by test into memory (RAM)

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”

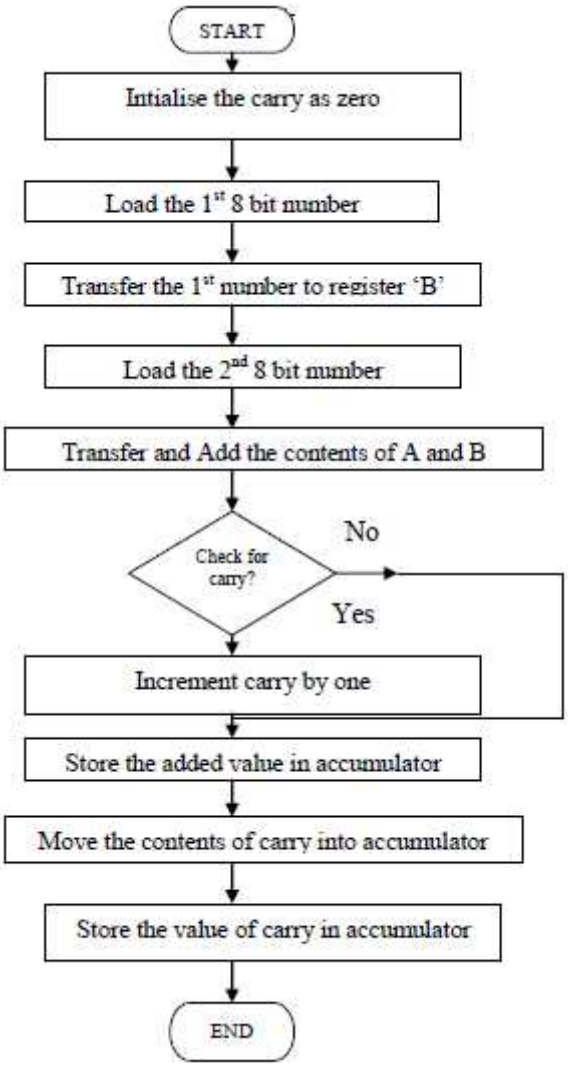
How to executive program
1. Press “RESET”
2. Press “GO”
3. Enter the address location in which line program was executed
4. Press “Execute” key

Result:
Thus 8085 microprocessor was studied successfully.

# UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

## Lab Manual

**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.



| 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.

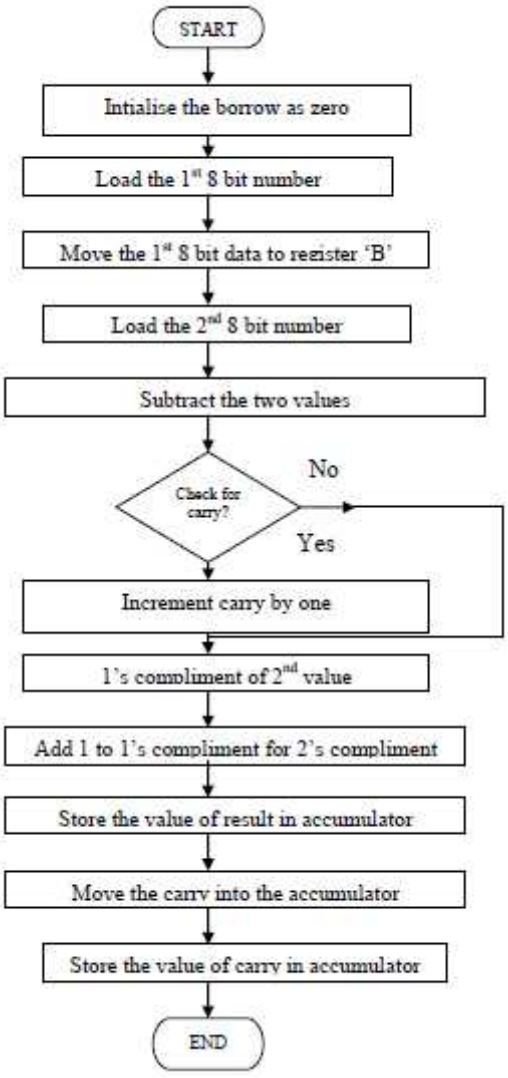
# UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

## Lab Manual

**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



| 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.

# UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

## Lab Manual

**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

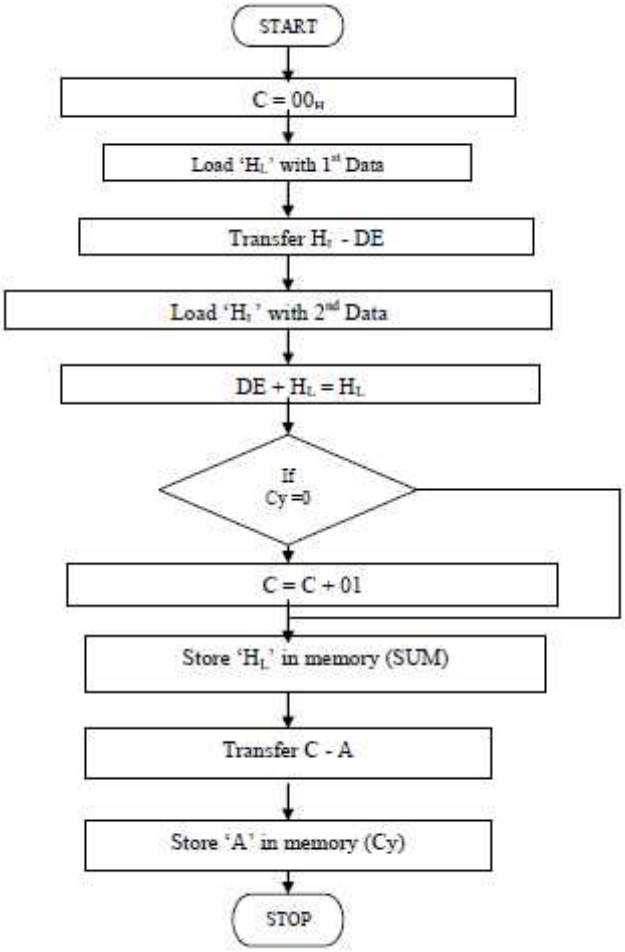
Step 8 : Get the higher byte of 1st number into accumulator

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.

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

## Lab Manual

**Experiment 3(b) :-**

**Aim:**

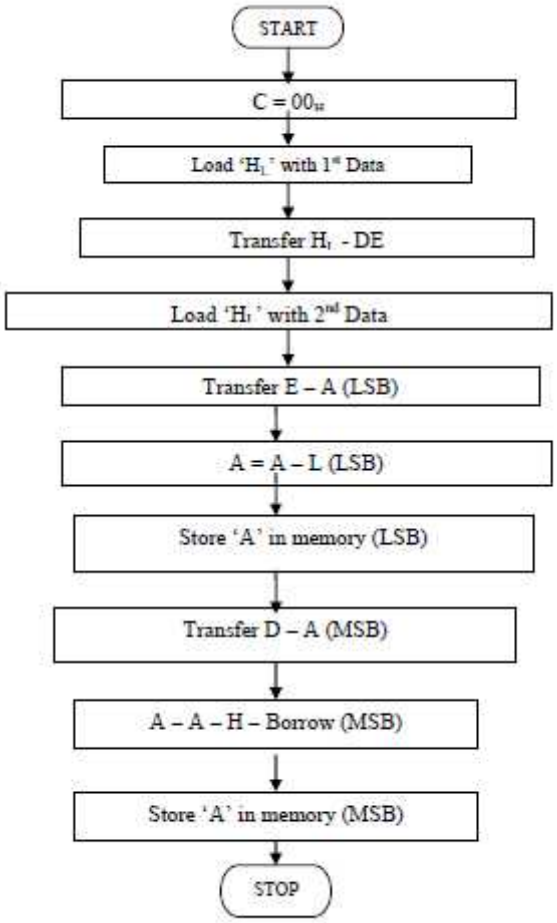
To write an assembly language program for subtracting two 16 bit numbers using 8085 microprocessor kit.

**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 | | MOV | A,E | 7B | LSB of '1' to 'A' |
| 450A | | SUB | L | 95 | A – A – L |
| 450B | | STA | 4804 | 32 | A – memory |
| 450C | | | | 04 | |
| 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 |
| 4805 | 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.

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

## Lab Manual

**Experiment 4(a) :-**

**Aim:**

To write an assembly language for multiplying two 8 bit numbers by using 8085 micro processor kit.

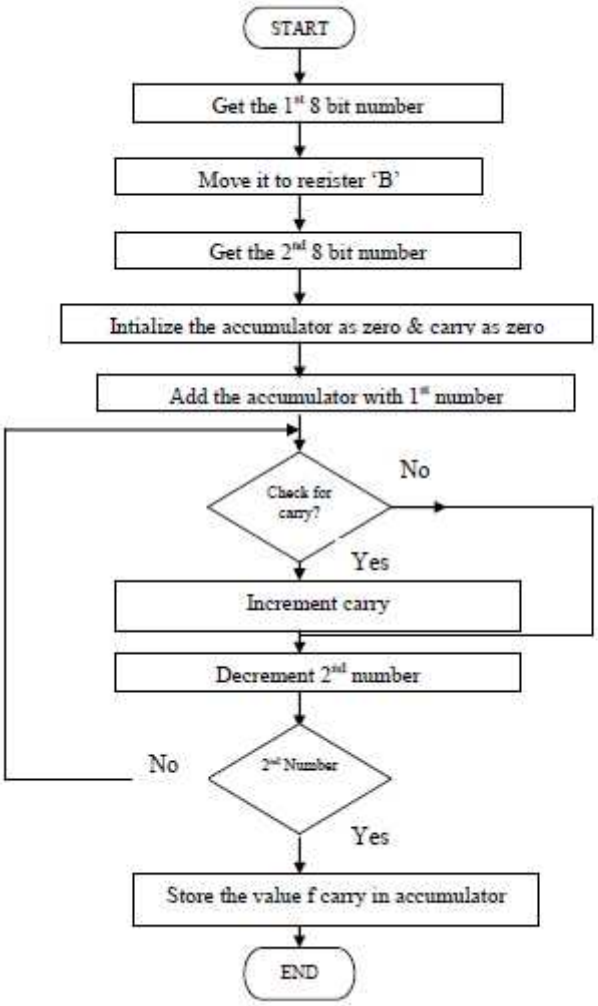
**Apparatus required:**

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.

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

## Lab Manual

**Experiment 4(b) :-**

**Aim:**

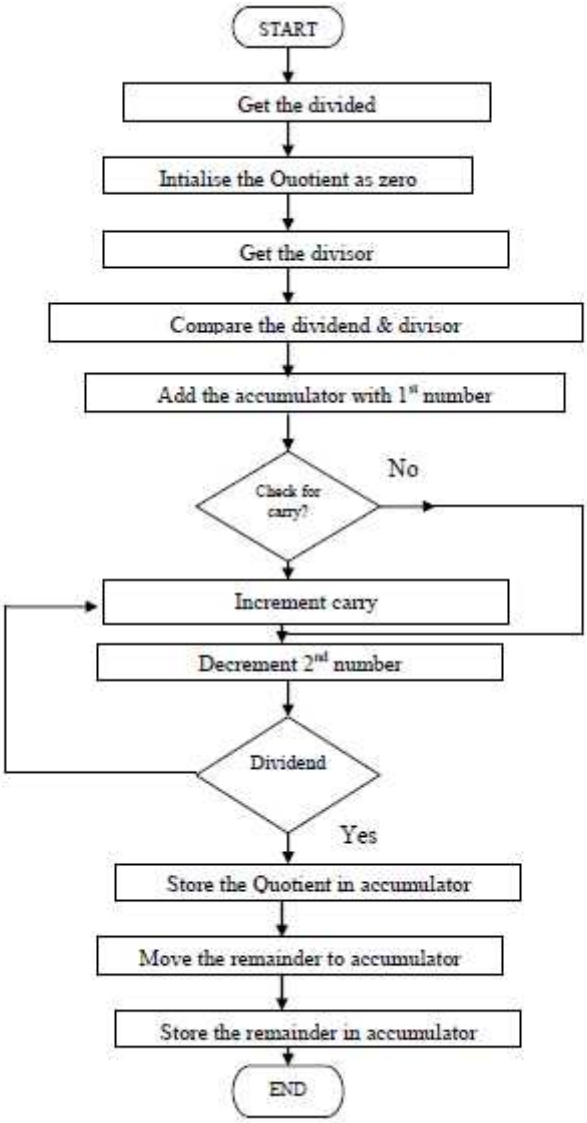
To write an assembly language program for dividing two 8 bit numbers using microprocessor kit.

**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



| 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

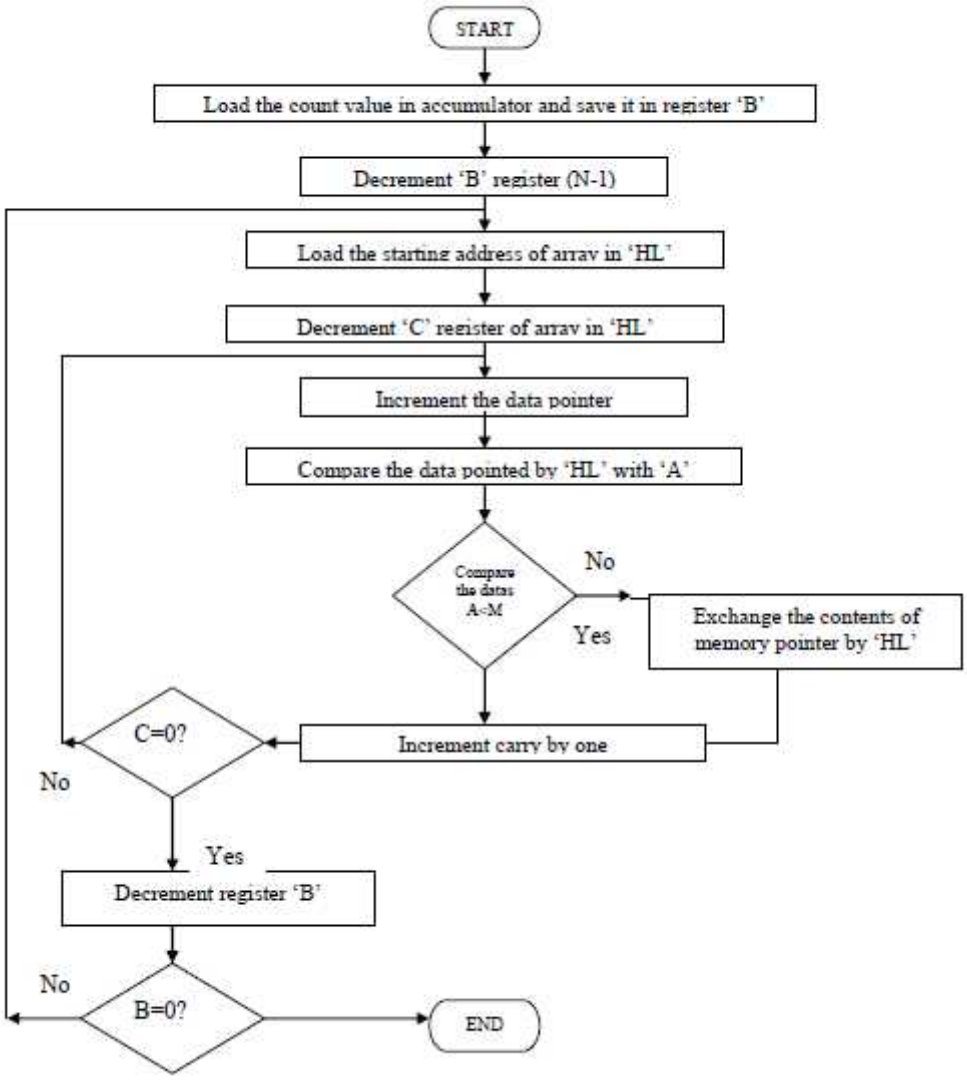
**Result:**
The assembly language program for division of two 8 bit numbers was executed using 8085 micro processing kit.

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

## Lab Manual

**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



| 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.

# UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

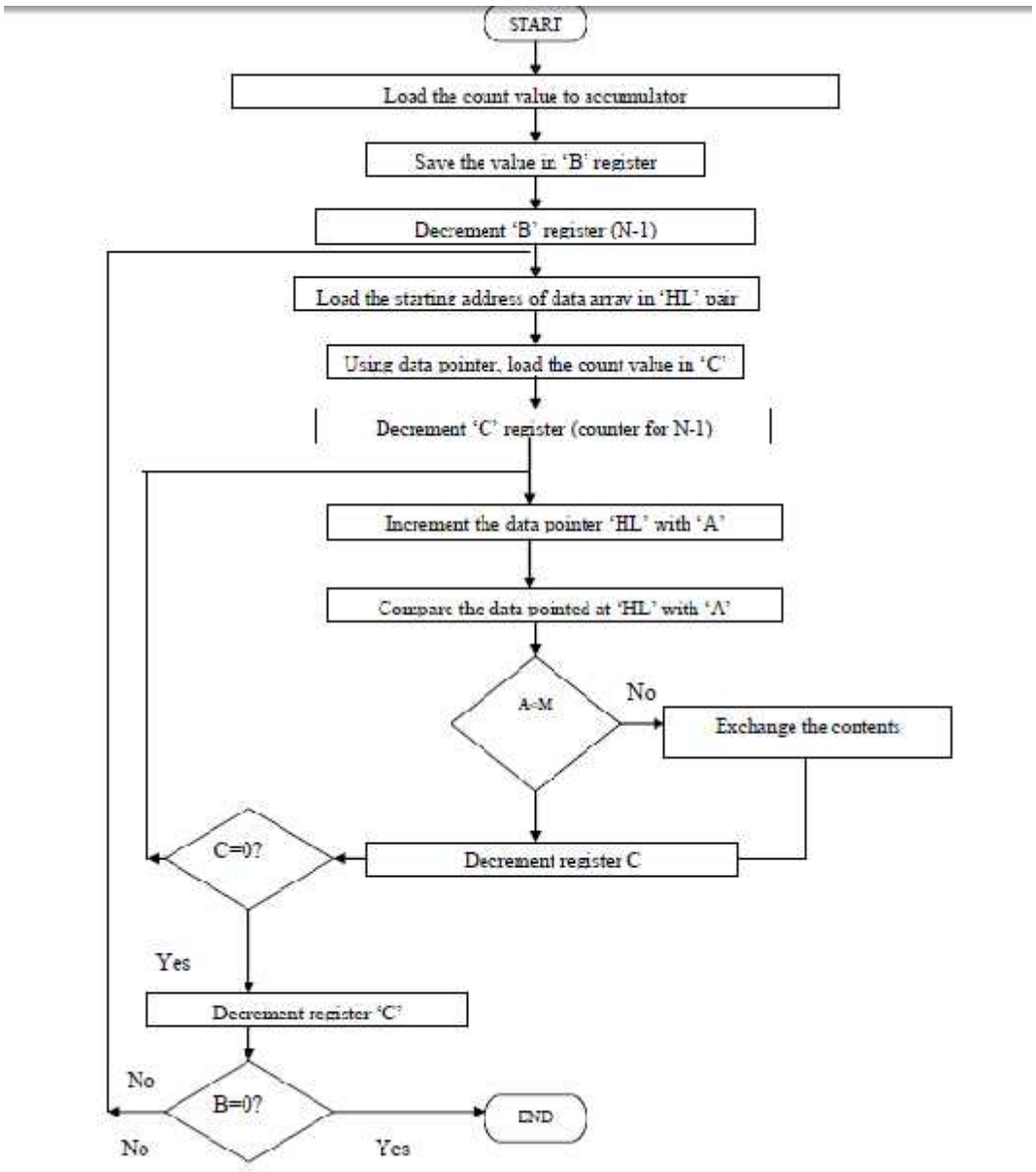
## Lab Manual

**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



| 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 |

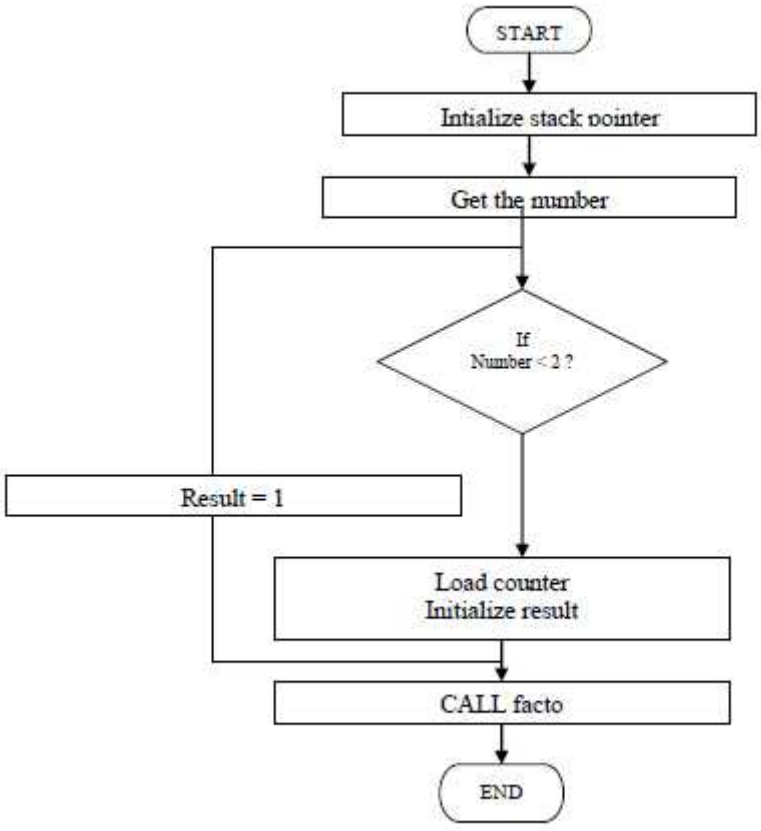
**Result:**

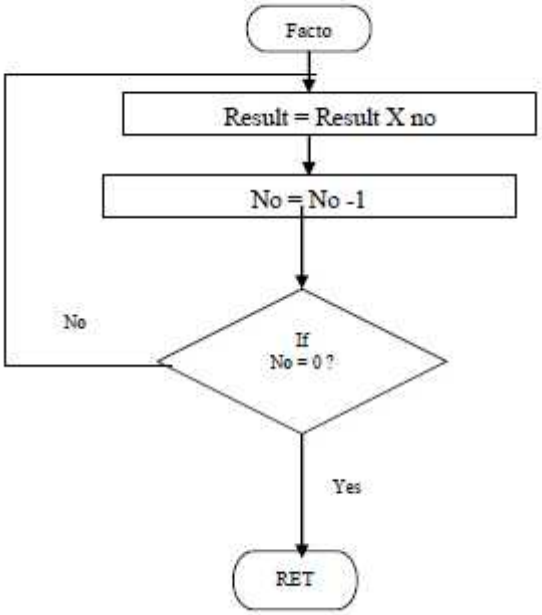
The assembly language program for sorting '4' numbers in descending order was executed successfully using microprocessor kit.

**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<br>4201<br>4202 | 3A<br>50<br>42 | | LDA | 4250 | Get the number in accumulator |
| 4203<br>4204 | FE<br>02 | | CPI | 02H | Compare data with 2 and check it is greater than 1 |
| 4205<br>4206<br>4207 | DA<br>17<br>42 | | JC | Loop 1 | If cy =1 jump to loop 1<br>If cy = 0 proceed |
| 4208 | 5F | | MOV | E,A | Move content of A to E |
| 4209<br>420A | 16<br>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<br>420E<br>420F | CD<br>00<br>46 | | CALL | Facto | Call sub routine programe Facto |
| 4210 | EB | | XCHG | | Exchange (DE) – (HL) |
| 4211<br>4212<br>4213 | 22<br>51<br>42 | | SHLD | 4251 | Store content of HL in specified memory location |
| 4214<br>4215<br>4216 | C3<br>1D<br>42 | | JMP | Loop 3 | Jump to Loop 3 |
| 4217<br>4218<br>4219 | 21<br>00<br>01 | Loop 1 | LXI | $H,0001_H$ | HL is loaded with data 01 |
| 421A<br>421B<br>421C | 22<br>51<br>42 | | SHLD | 4251 | Store the result in memory |
| 421D | 76 | Loop 3 | HLT | | Terminate the program |

| Sub Routine | | | | | |
|---|---|---|---|---|---|
| 4600<br>4601<br>4602 | 21<br>00<br>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<br>4607<br>4608 | C2<br>04<br>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<br>460C<br>460D | C4<br>00<br>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

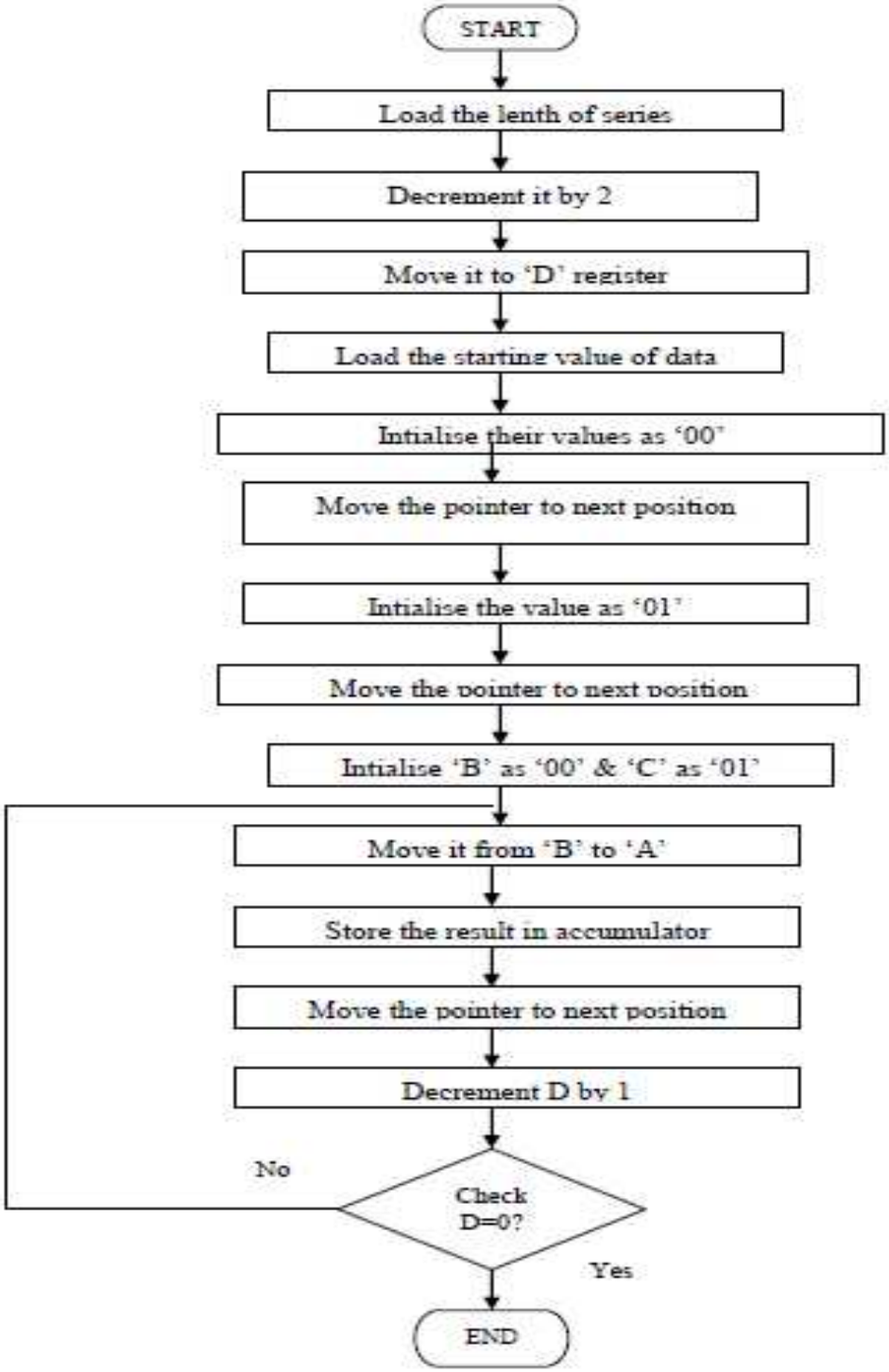
# UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

## Lab Manual

**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

## Course Description

**Title of Course: Group Discussion**
**Course Code: HU581**
**L-T –P Scheme: 0-0-3** **Course Credits: 2**

A group discussion aims at a structured but informal exchange of knowledge, ideas, and perceptions among the participants on any issue, topic or sub-topic. Contributions are pooled together and examined in terms of their relevance and validity to the discussion objectives. If planned and organized in a structured way and certain essential conditions are met, it can provide a highly enriching and stimulating experience to the participants. Lets us see, the objectives, different steps involved in it and its limitations.

**Objectives of a Group Discussion**

- Produce a range of options or solutions, addressing a particular problem or an issue.
- Generate a pile of ideas by examining issues in greater depth, looking at different dimensions of these issues.
- Broaden the outlook of the participants through cross-fertilization and exposure to new and different experiences and ideas and enrich their understanding of the issues under discussion.
- Develop their skills in interpersonal communication and in expressing their views in a clear and succinct manner.
- Effective means of changing attitudes through the influence of peers in the group
- Valuable means of obtaining feedback for the training team on verbal skills, motivation level and personal traits of the participants and characteristics of the group

**Steps in organizing a Group Discussion**

- Setting up the Groups
- Planning a Group Discussion
- Preparation of Group Reports
- Presentation and Consolidation of Group Reports

**Limitations**

- If the group is large, not all the members may get the opportunity to participate and contribute to the discussion.
- If the task is not clearly defined, the discussion may lack focus and, as a result, it may be unproductive.
- Difficulties can arise if the leader is unskilled in guiding the discussion and/or not familiar with the topic or the issues.
- Some members may dominate and, in a way, hijack the discussion.

## Course Description

- As this is a group task, some members may take it easy and not feel constrained to participate.

Learning outcomes

After studying this course, you should be able to:

- understand the key skills and behaviours required to facilitate a group discussion
- prepare effectively before facilitating a meeting
- consider some of the difficult behaviours that can occur in meetings
- think of some possible strategies for dealing with these.