R-19 SECOND YEAR SEM I - SYLLABUS

SECOND YEAR SEMESTER – I												
Code	**Course**	**Cate gory**	**L**	**T**	**P**	**E**	**O**	**Tota l**	**Sessio nal Marks**	**Externa l Marks**	**Tota l Mar ks**	**Credits**
IT211	Data Structures	PC	2	1	0	1	4	8	40	60	100	3
IT212	Basics of Electrical Engineering	ES	3	0	0	1	4	8	40	60	100	3
IT213	Discrete Mathematical Structures	BS	3	0	0	1	5	9	40	60	100	3
IT214	Computer Organization	PC	2	1	0	1	3	7	40	60	100	3
IT215	Microprocessors	PC	2	1	0	1	2	6	40	60	100	3
IT216	Data Structures Lab	PC	0	0	3	0	3	6	50	50	100	1.5
IT217	Microprocessor Lab	PC	0	0	3	0	3	6	50	50	100	1.5
IT218	Python Programming lab	PC	0	1	3	0	0	4	50	50	100	1.5
TOTAL			**12**	**4**	**9**	**5**	**24**	54	350	450	**800**	**19.5**

DATA STRUCTURES

IT211 **Credits: 3**
Instruction: 2 Periods & 1 Tut /week Sessional Marks: 40
End- Exam: 3 Hours End-Exam-Marks: 60

Prerequisite: C Programming

Course Objective:

- Assess how the choice of data structures impacts the performance of programs.
- Choose the appropriate data structure and algorithm design method for a specified application.
- Solve problems using data structures such as linear lists, stacks, queues, hash tables, binary trees, heaps, binary search trees, and graphs and writing programs for these solutions.

Course Outcomes:

After completion of this course, a student will be able to :	
1.	Analyze and choose appropriate ADT to solve a given problem
2.	Compare and contrast the benefits of dynamic and static data structures and implement Searching and sorting techniques
3.	Design and implement abstract data types such as linked list, stack, queue and tree in static and dynamic context using C programming language
4.	Develop a C program for solving real world problems using linear and non-linear data structures**.**

Mapping of course outcomes with program outcomes:

		PO												**PSO**	
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**1**	**2**
CO	**1**	2	3	3	1	1			1	1	1		1	3	1
	2	2	2	3		1			1	1	1		1	3	1
	3	2	2	3		1			1	1	1		1	3	1
	4	2	2	3	3	1			1	3	1		1	3	2

SYLLABUS

Unit-I: Introduction **10 Periods**

Introduction to data structures, arrays and structures. Dynamic Memory Management, Abstract Data Type (ADT).
List: Definition and examples- Primitive Operations- Representation using array and Linked List. Types of Linked Lists and implementation: single, double and circular. The array and linked list advantages, disadvantages and applications.
Learning Outcome: At the end of this Unit the student will be able to

- Implementation of linked Lists.
- Solve software requirements that require Linked Lists

Unit-II: Stacks and Queues **12 Periods**

The Stack ADT: Definition, Primitive Operations and representation. Stack ADT implementation using array and linked list. Applications of Stacks: Prefix, infix and Postfix notations, conversion between infix, prefix and postfix, postfix evaluation using stacks.
Queue ADT: Definition, Primitive operations and Representation. Queue ADT implementation using array and linked list. Types of Queue: Circular Queue, Priority Queue, Operations and implementation using array and linked list. The queues advantages, disadvantages, and applications.
Learning Outcome: At the end of this Unit the student will be able to

- Describe specific tasks to which stacks and queues are suited.
- Demonstrate the operations of stacks and queues.
- Apply stacks to a specific application.
- Apply queues to a specific application.

Unit - III: Sorting and Searching **10 Periods**

Sorting: General background, bubble sort, insertion sort, quick sort and merge Sort.
Searching: General background, linear search, binary search
Introduction to Hashing, Hash Function, Hashing techniques, Collision Resolution Methods: Open Addressing, Chaining.
Learning Outcome: At the end of this Unit the student will be able to

- Understand common sort methods.
- Describe quick, and merge sort.
- Compare quick, and merge sorting in terms of their overall memory and runtime efficiency.
- Chart the efficiency of quick, and merge sorting for small, medium, and large data loads.

Unit-IV: Trees **08 Periods**

Trees: Introduction, Terminology, Binary trees: Terminology, Representation. Binary tree implementation using array and linked list. Tree Traversal Techniques,
Types: Binary Search Tree, AVL Tree.
Learning Outcome: At the end of this Unit the student will be able to

- Demonstrate effective trees.
  - State the difference between trees and graphs.
  - Know the difference between binary and nonbinary trees.
  - Apply trees to solve specific application requirements.

Unit-V: Graphs **12 periods**

Graphs: Introduction- terminology, Representation of graphs-linked list and adjacency matrix, Representation in C, Graph Traversals-Breadth-First Search, Depth-First Search. Spanning Trees: Introduction and terminology, Minimum Spanning Tree algorithms: Prims and Krushkals. Applications of Graphs: Dijkstra's & Warshall's Algorithm.
Learning Outcome: At the end of this Unit the student will be able to

- Demonstrate effective graphs.
  - State the components of a graph.
  - Describe the two principal graph traversal paradigms.
  - Demonstrate the use of graphs as a solution to a particular application requirement.
  - Know the difference between directed and undirected graph.
  - Explain means of generating spanning trees.

TEXT BOOKS:

1. Y. Langsam, M. Augenstin and A. Tannenbaum, "Data Structures using C" Pearson Education, 2nd Edition, 1995

REFERENCE BOOKS:

1.Ellis Horowitz, Sartaj Sahni, Fundamentals of Data Structure, computer science Press.

2.Richard F, Gilberg , Forouzan, Cengage ,"Data Structures", 2/e, 2005.

BASICS OF ELECTRICAL ENGINEERING

Code: IT 212 - ES **Credits:3**
Instruction: 3 Periods Sessional Marks: 40
End- Exam: 3 Hours End-Exam-Marks: 60

Prerequisites: Nil

Course Objectives:

- Analysis of circuits by using KCL and KVL
- Finding equivalent circuits by using circuit theorems
- Analysis of magnetic circuits
- Principle of operation and behavior of electrical machines

Course Outcomes:

By the end of the course, the student will be able to:	
1.	Calculate voltage a cross, current through and power supplied / absorbed by an electrical
2.	Analyze magnetic circuit
3.	Obtain the performance characteristics of D.C. Generators.
4.	Obtain the performance characteristics of D.C. Motors.
5.	Obtain the performance characteristics of Transformer and Induction Motor.

SYLLABUS

UNIT –I

Electric Circuits : Circuit Elements, Basic Law's, KVL, KCL, Linearity Principle (Super Position), Mesh and Nodal analysis, Thevenin's and Norton's theorems.

UNIT II

Magnetic Circuits : Definitions of magnetic circuit, Reluctance, Magneto-motive force, magnetic flux, Simple problems on magnetic circuits. Faraday's laws of Electromagnetic Induction, Induced E.M.F., Dynamically induced E.M.F.

UNIT III

D.C. Generators : D.C. Generator principle, construction of D.C. generator, E.M.F equation of D.C. generator, Types of D.C. generators, Efficiency, Applications.

UNIT IV

Motors : Principle, working of D.C. Motors, Significance of back E.M.F., Torque equation of D.C. Motors, Types of D.C. Motors, Special Motors (Stepper Motor and Servo Motor) and Applications.

UNIT V

AC Machines : Transformer working Principle, EMF equation of transformer, Voltage regulation of Transformer. Three-phase Induction Motor working principle, Construction of 3 Phase Induction Motor, Principle of operation, Types of 3 phase induction Motors, Applications.

TEXT BOOKS:

1. **V.K. MEHTA &ROHIT MEHTA** *"Principles of Electrical Engineering"* 2nd edition, S. Chand Publications.
2. **V.K. MEHTA & ROHIT MEHTA** *"Principles of Electrical Machines"* 2nd edition, S. Chand Publications.

REFERENCE BOOK:

1. **J.B. Gupta** *"A Text book of Electrical Engineering"* S.K. Kataria& Sons Publications.

DISCRETE MATHEMATICAL STRUCTURES
[common to CSE & I.T.]

IT213	**Credits: 3**
Instruction: 3 Periods	Sessional Marks: 40
End- Exam: 3 Hours	End-Exam-Marks: 60

1. Prerequisites : Elementary knowledge of Set theory, Matrices and Algebra.

Course Objectives :

The main objectives of the course are to:

- Introduce concepts of mathematical logic for analyzing propositions and proving theorems.
- Use sets for solving applied problems binary relations and introduce concepts of algebraic structures
- Work with an ability to solve problems in Combinatorics
- Solve problems involving recurrence relations and generating functions.
- Introduce basic concepts of graphs, digraphs and trees

3. Course Outcomes: At the end of the course student should be able to:

CO - 1	Understand mathematical logic, mathematical reasoning and to study about the validity of the arguments and also prove mathematical theorems using mathematical induction.
CO - 2	Determine properties of binary relations, identify equivalence and partial order relations, sketch relations and Familiarize with algebraic structures.
CO - 3	Apply counting techniques to solve combinatorial problems and identify, formulate, and solve computational problems in various fields.
CO - 4	Understand Recurrence Relation, Generating functions and solving problems involving recurrence equations.
CO - 5	Familiarize with the applications of graphs , trees and algorithms on minimal spanning tress and apply graph theory in solving computing problems

SYLLABUS

UNIT-I: MATHEMATICAL LOGIC (12Periods)

Fundamentals of logic- Logical inferences-Methods of proof of an implication-First order logic and other proof methods -Rules of inference for quantified propositions – Mathematical induction.

Learning outcome: At the end of this unit, student will be able to

- Find equivalence formulas, implementation of logic for mathematical proofs (L_1)
- Apply inference theory to verify the consistence of data (L_3)
- Construct logical statements from informal language to propositional logic expressions(L_6)
- Apply the pigeonhole principle in the context of a contradiction proof (L_3)
- Prove mathematical theorems using mathematical induction(L_5)

(Sections: 1.5 to 1.10 of Text book [1])

UNIT-II : RELATIONS AND ALGEBRAIC SYSTEMS (12 Periods)

RELATIONS:

Cartesian products of sets –Relations **-** Properties of binary relations in a set – Relation matrix and graph of a relation – Partition and covering of set – Equivalence relations - Composition of Binary relations-Transitive closure of a relation -Partial ordering – Partially ordered set.

(Sections :2-1.9,2-3.1 to 2-3.5,2-3.7, 2-3.8, 2-3.9 of Text book [2])

ALGEBRAIC SYSTEMS:

Definitions and simple examples on Semi groups, Monoids , Group, Ring and Fields.

Learning Outcomes:

Learning outcome: At the end of this unit, student will be able to

- Determine properties of relations, identify equivalence and partial order relations, sketch relations. (L_5)
- Understand concepts of Semi group, Monoid, Group, Ring and Fields. (L_2)

(Sections:3-1.1, 3-2.1,3-2.2, 3-5.1,3-5.11and 3-5.12 of Text book [2])

UNIT-III: ELEMENTARY COMBINATORICS (10Periods)

Basics of counting- Combinations and permutations-Their enumeration with and without repetition-Binomial coefficients-Binomial and multinomial theorems-The principle of inclusion-Exclusion.

Learning outcome: At the end of this unit, student will be able to

- Solve problems on Permutation and Combinations with and without repetition (L_3)
- Solve problems on binomial and Multinomial coefficients(L_3)
- Solve counting problems by using principle of inclusion-exclusion (L_3)

(Sections :2.1to 2.8 of Text book [1])

UNIT-IV: RECURRENCE RELATIONS (10Periods)

Generating functions of sequences-Calculating their coefficients-Recurrence relations-Solving recurrence relations-Method of characteristic roots- Non-homogeneous recurrence relations and their solutions.

Learning outcome: At the end of this unit, student will be able to

- Formulate recurrence relations of the sequences
- Solve problems using generating functions(L_3)
- Solve homogeneous linear recurrence relations(L_3)
- Evaluate complementary function and particular integral for non homogeneous linear recurrence relations (L_5)
- Apply substitution method to solve non-linear recurrence relations (L_3)

(Sections: 3.1 to 3.6 of Text book [1])

UNIT- V: GRAPH THEORY
(16Periods)

Introduction to graphs – Types of graphs – Graphs basic terminology and special types of simple graphs – representation of graphs and graph isomorphism – Euler paths and circuits-Hamilton paths and circuits – Planar graphs – Euler's formula
Introduction to Trees and their properties – Spanning Trees –– Minimum Spanning Trees – Kruskal's Algorithm .

Learning outcome: At the end of this unit, the student will be able to

- Identify different graphs and their properties(L_3)
- prove elementary results about graphs and trees(L_5)
- Construct Euler and Hamiltonian graphs (L_3)
- Construct the graph for the given data (L_3)
- Construct the spanning tree and binary tress from graphs (L_3)
- Build minimal spanning tree by using different algorithms (L_3)

(Sections: 5.1 to 5.4, 5.7,5.8,5.9,5.10 of Text book [1])

TEXT BOOKS:

1). Joe L. Mott, Abraham Kandel & T. P. Baker, "Discrete Mathematics for computer scientists & Mathematicians" Prentice Hall of India Ltd, New Delhi.,2008

2) J.PTremblay,R.Manohar,"DiscreteMathematicalStructures with Applications to Computer Science", Tata McGraw-Hill Publishing Company Limited,1997

REFERENCE BOOKS:

1. Keneth. H. Rosen, Discrete Mathematics and its Applications, 6/e, Tata McGraw-Hill, 2009.
2. Richard Johnsonburg, Discrete mathematics, 7/e, Pearson Education, 2008.

COMPUTER ORGANIZATION

IT214 **Credits: 3**

Instruction hours: 2 Periods & 1 Tut/Week Sessional Marks: 40

End Exam: 3 Hours End Exam Marks: 60

Prerequisite:

Students are expected to be capable of recollecting the concepts of Fundamentals of IT and DLD to relate with the linked topics in CO

Course Objectives

- To discuss in detail the implementation of micro operations and operation of the arithmetic unit
- To understand the basic structure and operation of a digital computer.
- Understand the fundamentals of different instruction set architectures and their relationship to the CPU design.
- To study the different ways of communicating with I/O devices and standard I/O interfaces.
- To study the hierarchial memory system including cache memories

Course outcomes

1. Solve problems using micro operations and computer arithmetic operations.
2. Interpret hardwired and micro programmed way of designing the control unit of a digital computer.
3. Identify and compare different issues related to organization of CPU
4. Judge the working of different hardware components associated with the input-output organization of a computer.
5. Categorize memory organization and explain the function of each element of a memory hierarchy

	PO												PSO	
	1	2	3	4	5	6	7	8	9	10	11	12	1	2
CO-1	3	2	1	1	2				1				2	1
CO-2	2	3	2	2					1	1			2	1
CO-3	3	3		2	2					1		1	2	1
CO-4	2	2		3			1	1	1			1	2	1
CO-5	2	2		3			2	1	1		2	1	2	1

SYLLABUS

UNIT-I **8 Periods**

Register Transfer and Micro operations:
Register Transfer Language, Register Transfer, Bus and Memory Transfers, Arithmetic Micro-operations, Logic Micro-operations, Shift Micro-operations, Arithmetic Logic Shift Unit. **Learning Outcome:** At the end of this Unit the student will be able to

1. Understand concepts of register transfer logic.
2. Summarize the types of micro operations.
3. Design logic circuits for different micro operations.

UNIT-II **12 Periods**

Basic Computer Organization and Design:
Instruction Codes, Computer Registers, Computer Instructions, Timing and Control, Instruction Cycle, Memory-Reference Instructions, Input-Output and Interrupt, Complete Computer Description.
Learning Outcome: At the end of this Unit the student will be able to

1. Understand the role of bus and registers in instruction execution
2. Instruction format how it works explain with the help of diagram
3. Understand the complete computer description of how it works.

UNIT-III **14 Periods**

Microprogrammed Control: **3** Periods
Control Memory, Address Sequencing, Micro program Example.
Pipelining: **4** Periods
Parallel Processing,Pipelining,Arithmetic Pipeline,Instruction Pipeline.
Central Processing Unit: **7** Periods
Introduction, General Register Organization, Stack Organization, Data Transfer and Manipulation, Program Control, Program Interrupt, Types of interrupts, CISC Characteristics, RISC Characteristics.
Learning Outcome: At the end of this Unit the student will be able to

4. Understand concepts of Hardwired control and micro programmed control.
5. Understand the architecture and functionality of central processing unit.
6. Learn the concepts of parallel processing, pipelining.
7. Explain different pipelining processes.
8. Differentiate between RISC and CISC.

UNIT-IV **8 Periods**

Input-Output Organization: Peripheral Devices, Input-Output Interface, Asynchronous Data Transfer, Modes of Transfer, Priority Interrupt, Direct Memory Access.
Learning Outcome: At the end of this Unit the student will be able to

1. Discuss about different types of peripheral devices of computer.
2. Learn the different types of serial communication techniques.

UNIT-V **6 Periods**

Memory Organization:
Memory Hierarchy, Main Memory, Auxiliary Memory, Associative Memory, Cache Memory.
Learning Outcome: At the end of this Unit the student will be able to

1. Learn the concept of memory hierarchy.
2. Discuss the concept of memory organization.
3. Explain the use of cache memory.
4. Understand the concept of memory management hardware.
5. Summarize the types of memory.

Text Book:
Computer System Architecture, M.Morris Mano ,Third Edition, Pearson Education Inc., 2003
Reference Books:
1. John D. Carpinelli , Computer Systems Organization and Architecture, ,Pearson Education Inc., 2003
2. William Stallings, Computer Organization and Architecture,5th Edition,2000

MICROPROCESSORS

IT215	**Credits: 3**
Instruction: 2 Periods & 1 Tut/Week	Sessional Marks: 40
End Exam: 3 Hours	End Exam Marks: 60

Prerequisite:

Basics of electronics, digital logic circuits

Course Objectives:

- To introduce students with the architecture and operation of typical microprocessors
- Edit, compile, execute, and debug an assembly language programs.
- To understand the concepts of interrupts and subroutines.
- To understand the assembly language programming concepts like procedures, macros.

Course Outcomes:

After completion of this course, a student will be able to:	
1.	Describe the architecture of microprocessors 8085/8086. Illustrate machine level instructions with timing diagrams
2.	Demonstrate methods of accessing information in machine memory using direct or indirect addressing schemes, and describe various memory management schemes used in typical microcomputer systems including segmented and virtual memory.
3.	Write debug and analyze assembly language programs for the 8085/8086 microprocessor instruction set.
4.	Illustrate the usage of stack in subroutine and interrupt handling.

		PO												**PSO**	
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**1**	**2**
CO	**1**	1	2						1	1	1		1	1	1
	2	2	2			2			1	1	1		1	1	1
	3	2	3			2			1	1	1		1	1	1
	4	1	3			2			1	1	1		1	1	1

SYLLABUS

UNIT-I: **10 Periods**

The 8085A µP. Architecture :

Introduction to Microprocessors and Microcomputers, Internal Architecture and Functional/Signal Description of typical 8-bit µP.

Learning Outcomes:

1. Describe the architecture of 8085 with the help of diagram.
2. Describe the 8085 pin configuration and its working with the help of diagram.

UNIT-II **8 Periods**

8085 Instruction set and Programming Techniques:

8085 Instruction Set and Timing Diagrams of 8085 µP for Read, Write, Move, Store, Out, In operations, Programming Techniques: Looping, Counting, and Indexing, programming examples.

Learning Outcomes:

1. Illustrate 8085 machine instructions with timing diagrams.
2. Illustrate programing techniques with example problems.

UNIT-III **12 Periods**

Subroutines and Interrupts:

Counter and timing Delays: using one register, using a register pair, using a loop, Stack and Subroutines, Code Conversions: BCD to Binary, Binary to BCD, Binary to ASCII Hex Code, ASCII Hex code to Binary, BCD Arithmetic, 16-bit data Operations, Interrupts and Interrupt Service Routines.

Learning Outcomes:

1. Illustrate timing delays in 8085 machine language programming.
2. Illustrate code conversions using 8085 instructions.

UNIT-IV **10 Periods**

The 8086 µP. Architecture and Instruction Set:

Internal Architecture and Functional/Signal Description of 8086/8088

Segmented Memory, Maximum-Mode and Minimum-Mode Operation, Addressing Modes, Instruction Set.

Learning Outcomes :

1. Describe the internal architecture of 8086 microprocessor.
2. Illustrate the different modes of accessing memory with examples.
3. Describe the memory management schemes in 8086 microprocessor.

UNIT-V **8 Periods**

Programming the 8086 µP.: Programming techniques: Logical Processing, Arithmetic processing, Looping, Procedures, Macros, interrupts.

Learning Outcomes :

1. Write debug and analyze programs using 8086 instructions.
2. Illustrate interrupt handling using stacks.

TEXT BOOKS:

1. Microprocessor Architecture, Programming, and Applications with the 8085 Ramesh S. Gaonkar, 4th
Edition, Penram International, 1999
2. The 80x86 Family, Design, Programming and Interfacing, John E.Uffenbeck, 3rd Edition, Pearson Education Inc., 2002

REFERENCE BOOK:

1. Microprocessors and Interfacing, Douglass V Hall, 2nd Edition,TMH Publishing.

DATA STRUCTURES LAB

IT216 **Credits: 1.5**
Practical: 3 Periods /week Sessional Marks: 50
End Exam: 3 Hours End Exam Marks: 50

Prerequisite:
C Programming, Data Structures.

Course Objective:

- ☐ Assess how the choice of data structures impacts the performance of programs.
- ☐ Choose the appropriate data structure and algorithm design method for a specified application.
- ☐ Solve problems using data structures such as linear lists, stacks, queues, hash tables, binary trees, heaps, binary search trees, and graphs and writing programs for these solutions.

Course Outcomes:

After completion of this course, a student will be able to :	
1.	Implement linear data structures such as stacks, queues, linked lists and apply on real time problem like conversions & evaluations of expressions.
2.	Implement non linear data structures such as Trees and Graphs and apply on real time problem like finding shortest path.
3.	Implement different sorting and searching techniques.

Mapping of course outcomes with program outcomes:

		PO												PSO	
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**1**	**2**
CO	**1**	1	2	3	3	1			1	3	1		1	3	2
	2	1	2	3	3	1			1	3	1		1	3	2
	3	1	2	3	1	1			1	3	1		1	3	

Note: Every lab must be practiced in GDB Compiler/HackerRank platform and the execution part of rubrics (apart from viva, observation and record) must be evaluated based on the GDB/HackerRank performance.

List of Programs:

1) Programs to implement the following using an array. [CO1]
a) Stack b) Queue

2) Programs to implement the following using a singly linked list. [CO1]
a) Stack b) Queue

3) Program to do the following [CO1]
a) Infix to postfix conversion. b) Evaluation of postfix expression.

4) Programs to implement the following data structures. [CO1]
a) Circular Queue b) Priority Queue

5) Program to perform the following operations: [CO2]

a) Insert an element into a binary search tree.
b) Delete an element from a binary search tree.
c) Search for a key element in a binary search tree.

6) Program that use non-recursive functions to traverse the given binary tree in
a) Preorder b) In-order c) Post-order. [CO2]

7) Program to implement BFS and DFS for a given graph. [CO2]

8) Program to implement the following sorting methods: [CO3]
a) Merge sort b) Quick sort c) Insertion Sort d) Bubble Sort

9) Program to implement the following searching methods: [CO3]
a) Linear Search b) Binary search

10) Program to store **k** keys into an array of size n at the location computed using a Hash function, loc = key % n, where k<=n and k takes values from [1 to m], m>n, where m is size of the hash table. [CO3]

Reference Books:

1. Y. Langsam, M. Augenstin and A. Tannenbaum, “Data Structures using C” Pearson Education, 2nd Edition, 1995.
2. Richard F, Gilberg, Forouzan, Cengage, Data Structures, 2/e, 2005.
3. Data Structures using C, 2/2, ISRD Group.

MICROPROCESSOR LAB

IT217	Credits: 1.5
Practical: 3 Periods/week	Sessional Marks:50
End Exam:3Hours	End Exam Marks: 50

Course Objective:

1. Ability to write Assembly Language programs for the Intel 8085/8086 microprocessors using trainer kits and MASM. Ability to load, verify, and save machine language programs.

2. Ability to debug and interpret machine code using the DEBUG software.

3. Ability to decode and encode machine code by hand.

4. Ability to examine and modify the contents of Memory.

Course Outcomes:

1. Design digital logic circuits.
2. Write debug and analyse assembly language programs for the 8085/8086 microprocessor instruction set.
3. Execute assembly language programs using trainer kits/TASM or MASM software and analyze and interpret the results.

		PO												PSO	
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**1**	**2**
	1	1	2	3		2			1	1	1		1	1	1
	2	1	3	1	1	2			1	1	1		1	1	1
CO	**3**	2	3	1	1	2			1	1	1		1	1	1

Digital Logic Design (DLD) Experiments------- CO1

1. Characteristics of TTL
   AND gate
   OR gate
   NOT gate
   NAND gate
   NOR gate
   XOR gate
2. Half adder
3. Full adder
4. Multiplexer
5. Jk flip-flop
6. BCD to 7 segment decoder
7. Universal shift register
8. Decode counter

Microprocessor 8085 experiments ------- CO2

9. ALP to perform addition of two 8-bit numbers
10. ALP to perform subtraction of two 8-bit numbers
11. ALP to perform multiplication of two 8-bit numbers
12. ALP to perform division of two 8-bit numbers
13. ALP to find largest number in array
14. ALP to find smallest number in array
15. ALP to arrange an array in ascending order
16. ALP to arrange an array in Descending order
17. Logical operations
18. ALP to perform 16- bit addition
19. ALP to perform 16-bit subtraction
20. ALP to convert BCD number to binary number
21. ALP to convert a binary number to ASCII
22. ALP to convert a binary number to unpacked BCD number
23. ALP to convert an ASCII to binary number
24. ALP to perform addition of two unsigned BCD numbers
25. ALP to perform different multiplication tables

Microprocessor 8086 experiments (programs are implemented using TASM (turbo assembler)------------CO3

26. ALP to perform addition of two 8-bit numbers
27. ALP to perform subtraction of two 8-bit numbers
28. ALP to perform multiplication of two 8-bit numbers
29. ALP to perform division of two 8-bit numbers
30. ALP to perform addition of two 16-bit numbers
31. ALP to perform subtraction of two 16-bit numbers
32. ALP to perform multiplication of two 16-bit numbers
33. ALP to perform division of two 16-bit numbers
34. ALP to perform swapping of two 16-bit data
35. ALP to perform addition of even numbers and odd numbers separately from an array of data
36. ALP to add elements in an array
37. ALP to find whether the given number is even or odd
38. ALP to perform factorial of a number
39. ALP to copy a string from one location to another location
40. ALP to perform reverse of string operation
41. ALP to find largest number in an array
42. ALP to find smallest number in an array
43. ALP to search a particular character in a string
44. ALP to count no of zeros and ones in a byte of data

PYTHON PROGRAMMING LAB

IT218 **Credits: 2.5**

Tutorial & Practical: 1 & 3 Periods /Week Sessional Marks: 50

End Exam: 3 Hours End Exam Marks: 50

Prerequisite:

Students are expected to be able to open command prompt window or terminal window, edit a text file, download and install software, and understand basic programming concepts.

Course Objectives:

- To develop Python programs with conditional statements and loops.
- To define Python functions and call them.
- To use Python data structures–- dictionaries, tuples and sets,
- To do input/output with files in Python.
- To develop Python programs using OOP concepts.

Course Outcomes:

After completion of this course, a student will be able to:	
1.	Develop python programs using control flow statements.
2.	Express proficiency in handling of strings and Lists.
3.	Develop programs using data structures like dictionaries, tuples and sets.
4.	Design programs using file operations and regular expressions.
5.	Develop applications using Object-Oriented Programming concepts such as encapsulation, inheritance and polymorphism.

Mapping of course outcomes with program outcomes:

		PO												**PSO**	
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**1**	**2**
CO	**1**	3	2	3		1				1			2	2	2
	2	3	2	3		1				1			2	1	2
	3	3	2	2		1				1			3	2	2
	4	3	2	3		1			2	2	2	2	3	3	2
	5	3	2	3		1			2	2	2	2	3	3	3

UNIT-I **12-Hours**

Parts of Python Programming Language: Identifiers, Keywords, Statements and Expressions, Variables, Operators, Precedence and Associativity, Data Types, Indentation, Comments, Reading Input, Print Output, Type Conversions, The type() Function and Is Operator, Dynamic and Strongly Typed Language.

Control Flow Statements, The if, if…else, if…elif…else Decision Control Statements, Nested if Statement, The while, for Loops, The continue and break Statements,

Functions, Built-In Functions, Commonly Used Modules, Function Definition and Calling the Function, The return Statement and void Function, Scope and Lifetime of Variables, Default Parameters, Keyword Arguments, *args and **kwargs, Command Line Arguments.

Learning Outcome:

At the end of this Unit the student will be able to

- Analyze fundamental advantages of python over the other programming languages.
- Solve, test and debug basic problems using python script
- Implement Flow control statements required to real world problems.
- Familiarize the usage of functions and Modules and packages to enhance the problem solving.

UNIT-II **8-Hours**

Strings- Creating and Storing Strings, Basic String Operations, Accessing Characters in String by Index Number, String Slicing and Joining, String Methods, Formatting Strings.
Lists- Creating Lists, Basic List Operations, Indexing and Slicing in Lists, Built-In Functions Used on Lists, List Methods, The del Statement.

Learning Outcome:

At the end of this Unit the student will be able to

- Manipulate python programs by using python data structures like lists and strings.

UNIT-III **8Hour**s

Dictionaries-Creating Dictionary, Accessing and Modifying key:value Pairs in Dictionaries, Built-In Functions Used on Dictionaries, Dictionary Methods, The del Statement.
Tuples and Sets- Creating Tuples, Basic Tuple Operations, Indexing and Slicing in Tuples, Built-In Functions Used on Tuples, Relation between Tuples and Lists, Relation between Tuples and Dictionaries, Tuple Methods, Using zip() Function, Sets, Set Methods, Traversing of Sets.

Learning Outcome:

At the end of this Unit the student will be able to

- Manipulate python programs by using python data structures like dictionaries, tuples and sets.

UNIT-IV **12-Hours**

Files-Types of Files, Creating and Reading Text Data, File Methods to Read and Write Data, Reading and Writing Binary Files.
Regular Expression Operations- Using Special Characters, Regular Expression Methods, Named Groups in Python Regular Expressions.

Learning Outcome:

At the end of this Unit the student will be able to

- Design python programs using File operations.
- Design python programs using Regular Expression Methods.

UNIT-V **12-Hours**

Object-Oriented Programming- Classes and Objects, Creating Classes in Python, Creating Objects in Python, The Constructor Method, Classes with Multiple Objects, Class Attributes versus Data Attributes.
Encapsulation, Inheritance and Polymorphism.

Learning Outcome:

At the end of this Unit the student will be able to

- Design object-oriented programs with Python classes.
- Usage of inheritance, encapsulation, inheritance and polymorphism for reusability

Textbooks: 1. **"Introduction to Python Programming",** by Gowrishankar S, Veena A, 1st Edition, CRC Press/Taylor & Francis, 2018. ISBN-13: 978-0815394372

Reference Textbook: 1. Programming python, by Mark Lutz, 4th Edition, O'REILLY
2. The complete reference python, by Martin C Brown

Sample list of Experiments:

1. A) Running instructions in Interactive interpreter and a Python Script. [CO1]
   B) Write a program add.py that takes 2 numbers as command line arguments and prints its sum. [CO1]

2. A) Write a program to compute distance between two points taking input from the user. [CO1]
   B) Write a Program for checking whether the given number is a even number or not. [CO1]
3. A) Write a program to find greatest among 3 numbers.
   B) Using a for loop, write a program that prints out the decimal equivalents of 1/2, 1/3, 1/4, . . . , 1/10. [CO1]

4. A) Write a program using a while loop that asks the user for a number, and prints a count down from that number to zero. [CO1]
   B) Write a program using for loop to print sum of prime numbers up to the given number. [CO1]

5. A) Write a program using for loop to print sum of even terms in a Fibonacci series up to the given number. [CO1]
   B) Write a program using for loop to find the sum of cosine series up to specified number of terms for a given value of X. [CO1]

6. A) Write a program to count the number of characters in the string and store them in a

dictionary data structure. [CO2]

B) Write a program to use split and join methods in the string and trace a given word in the sub strings. [CO2]

7. A) Write a program to combine lists. Combine these lists into a dictionary as well. [CO2]

   B) Find mean, median, mode for the given set of numbers in a list. [CO2]

8. A) Write a program to print each line of a file in reverse order. [CO4]

   B) Write a program to compute the number of characters, words and lines in a file. [CO4]

9. A) Write a function ball_collide that takes two balls as parameters and computes if they are colliding. Your function should return a Boolean representing whether or not the balls are colliding. [CO1]

   Hint: Represent a ball on a plane as a tuple of (x, y, r), r being the radius

   If (distance between two balls centers) <= (sum of their radius) then (they are colliding).

   B) Write a function Reverse to reverse a list. Without using the reverse function. [CO1]

10. A) Write function to compute gcd, lcm of two numbers. Each function shouldn't exceed one line. [CO1]

    B) Write function Unique to check uniqueness of two lists. Function should return a Boolean true if they are unique otherwise return false. [CO1]

11. A) Write a program to perform addition of two square matrices. [CO2]

    B) Write a program to perform multiplication of two square matrices. [CO2]

12. A) Write a program to perform different operations on sets. [CO3]

    B) Write a program to perform different operations on tuples. [CO3]

13. Write a program to perform compile(), findall(), split(), sub() , subn() functions on expressions using re. [CO4]

14. A) Write a program to illustrate concept of multi level inheritance. [CO5]

    B) Write a program to illustrate concept of multiple inheritance. [CO5]

15. A) Write a program to illustrate concept of method overloading. [CO5]

    B) Write a program to illustrate concept of method overriding. [CO5]