COMPUTER ORGANIZATION

(Common to CSE, IT and AI&DS)

II Year – II Semester

Lecture	: 3	Internal Marks	: 30
Credits	: 3	External Marks	: 70

Course Objectives

- To familiarize with organizational aspects of memory, processor and I/O.

Course Outcomes

Upon successful completion of the course, the students will be able to

- identify different types of instructions.
- differentiate micro-programmed and hard-wired control units.
- analyze the performance of hierarchical organization of memory.
- summarize different data transfer techniques.
- demonstrate arithmetic operations on fixed- and floating-point numbers and illustrate concepts of parallel processing.

Course Content

UNIT – I: Register Transfer Language and Micro Operations

Introduction- Functional units, computer registers, register transfer language, register transfer, bus and memory transfers, arithmetic, logic and shift micro operations, arithmetic logic shift unit.

Basic Computer Organization and Design: Instruction codes, instruction cycle. Register reference instructions, Memory – reference instructions, input – output and interrupt.

UNIT – II: CPU and Micro Programmed Control

Central Processing unit: Introduction, instruction formats, addressing modes. Control memory, address sequencing, design of control unit - hard wired control, micro programmed control.

UNIT – III: Memory Organization

Memory hierarchy, main memory, auxiliary memory, associative memory, cache memory, cache coherence

UNIT – IV: Input-Output Organization

Peripheral Devices, input-output interface, asynchronous data transfer, modes of transfer- programmed I/O, priority interrupt, direct memory access, Input –Output Processor (IOP).

UNIT – V: Computer Arithmetic and Parallel Processing

Data representation- fixed point, floating point, addition and subtraction, multiplication and division algorithms.

Parallel Processing-Parallel Processing, Pipelining, Arithmetic Pipeline, Instruction Pipeline.

Text Books

1. M. Moris Mano, “Computer Systems Architecture”, 3rd edition, Pearson/ PHI.

Reference Books

1. Carl Hamacher, ZvonksVranesic, SafeaZaky, “Computer Organization”, 5thedition, McGraw Hill.
2. William Stallings, “Computer Organization and Architecture”, 6thedition, Pearson/PHI.
3. John L. Hennessy and David A. Patterson, “Computer Architecture a Quantitative Approach”, 4thedition, Elsevier.

* * *

DISCRETE MATHEMATICAL STRUCTURES
(Common to CSE, IT and AI&DS)
II Year – II Semester

Lecture	: 2	Tutorial	: 1	Internal Marks	: 30
Credits	: 3			External Marks	: 70

Course Objectives

- To impart the knowledge on mathematical reasoning, relations, graphs and recurrence relations.

Course Outcomes

Upon successful completion of the course, the students will be able to

- use mathematical logic for analyzing propositions and proving theorems.
- describe the properties of relations, functions and lattice theories.
- categorize different types of algebraic structures and describe their properties.
- apply the concepts of graph theory in modeling and solving non-trivial problems incomputer networks.
- apply pigeon hole principle in computer applications and solve recurrence relations.

Course Content

UNIT – I: Mathematical Logic

Propositional Calculus: Statements and Notations, Connectives, Truth Tables, Tautologies, Equivalence of Formulas, Tautological Implications, Theory of Inference for Statement Calculus.

UNIT – II: Relations, Functions and Lattice Theory

Relations: Properties of Binary Relations, Equivalence, Compatibility and Partial order relations, Hasse Diagram.

Functions: Inverse, Composite functions. Lattice – Definition. Principle of duality, types of lattices – distributive & modular lattices.

UNIT – III: Algebraic Structures

Introduction to algebraic systems, Quasi- group, Semi-group,Monoid, Group and abelian group. Subgroupsand Cyclic Groups.

UNIT – IV: Graph Theory

Introduction to Graphs. Representation of Graphs: Adjacency Matrices, Incidence Matrices. Isomorphism in Graphs. Eulerian Graphs, Planar Graphs, Hamiltonian Graphs and chromatic number of a graph.

UNIT – V: Pigeon hole principle and Recurrence Relations

Pigeonhole principle and its applications.Recurrence relations - Homogeneous and Non-Homogeneous recurrence relations using method of characteristic roots and generating functions.

Text Books

1. J.P.Trembley, R Manohar, "Discrete Mathematical Structures with Applications to Computer Science", Tata McGraw Hill, New Delhi.
2. Mott, Kandel, Baker, "Discrete Mathematics for Computer Scientists & Mathematicians", 2nd edition, PHI.

Reference Books

1. J K Sharma, "Discrete Mathematics", 2nd edition, Macmillan Publications.
2. Schaum's Outlines," Discrete Mathematics", 2nd edition, Tata McGraw Hill, New Delhi.
3. Rosen, "Discrete Mathematics and its Application with combinatorics and Graph Theory", 7thediton, Tata McGraw Hill, New Delhi.

* * *

ARTIFICIAL INTELLIGENCE

II Year – II Semester

Lecture	: 3	Internal Marks	: 30
Credits	: 3	External Marks	: 70

Course Objectives

- To familiarize with the basic principles of AI towards problem solving, inference, and learning.
- To investigate applications of AI in intelligent agents, expert systems and other machine learning models.

Course Outcomes

Upon successful completion of the course, the students will be able to

- analyze and formalize the problems as a state space, graph, or tree.
- use search algorithms to discover solution to a given problem.
- solve problems with uncertain information using probabilistic reasoning.
- formalize sequential decision making using Markov decision process.
- apply reinforcement learning to take suitable action to maximize reward in particular situation.

Course Content

UNIT – I: Introduction

Concept of AI, history, current status, scope, agents, environments, Problem Formulations, Review of tree and graph structures, State space representation, Search graph and Search tree.

UNIT – II: Search Algorithms

Random search, Search with closed and open list, Depth first and Breadth first search, Heuristic search, Best first search, A* algorithm, Game Search.

UNIT – III: Probabilistic Reasoning

Probability, conditional probability, Bayes Rule, Bayesian Networks- representation, construction and inference, temporal model, hidden Markov model.

UNIT – IV: Markov Decision Process

MDP formulation, utility theory, utility functions, value iteration, policy iteration and partially observable MDPs.

UNIT – V: Reinforcement Learning

Passive reinforcement learning, direct utility estimation, adaptive dynamic programming, temporal difference learning, active reinforcement learning- Q learning.

Text Books

1. Stuart Russel, Peter Norvig, Artificial intelligence, A modern Approach, 2nd edition, PEA.
2. Rich, Kevin Knight, Shiv Shankar B Nair, Artificial Intelligence- 3rd edition, TMH.

Reference Books

1. Saroj Kaushik, Artificial Intelligence- CENGAGELearning.
2. Patterson, Introduction to Artificial Intelligence, PHI.
3. George F Lugar, Artificial intelligence, structures and Strategies for Complex problem solving, 5th edition, PEA.

* * *

COMPILER DESIGN
(Common to CSE, IT and AI&DS)

II Year – II Semester

Lecture	: 2	Tutorial	: 1	Internal Marks	: 30
Credits	: 3			External Marks	: 70

Course Objectives

- To familiarize with lexical analyzer and different parsers.
- To introduce various storage allocation strategies, code generation and code optimization techniques.

Course Outcomes

Upon successful completion of the course, the students will be able to

- list compilation process steps of a language and represent tokens using regular expressions.
- design a parser to verify the syntax of a programming language.
- design syntax directed translation schemes for a given context free grammar.
- construct symbol table to access identifier information and perform various operations on it.
- apply code optimization techniques to enhance the efficiency of the intermediate code and generate code using generic code generation or DAG.

Course Content

UNIT – I: Lexical Analysis

Overview of language processing, preprocessors, compiler, assembler, interpreters, linkers and loaders, phases of a compiler. Lexical Analysis- role of lexical analysis, token, patterns and lexemes, transition diagram for recognition of tokens, reserved words and identifiers.

UNIT – II: Parsing

Syntax analysis, role of a parser, classification of parsing techniques.

Top-down parsing: Recursive descent parsing, first and follow, LL(1) grammars, non-recursive predictive parsing.

Bottom-up Parsing:Shift-Reduce parsing, operator precedence parsing, LR Parsers: construction of SLR, CLR (1), LALR parsers.

UNIT – III: Semantic Analysis

SDT, evaluation of semantic rules, Symbol tables- use of symbol tables, contents of symbol-table, operations on symbol tables, symbol table organization for block and non-block structured languages.

UNIT – IV: Intermediate Code Generation

Intermediate code- Three address code-quadruples, triples, abstract syntax trees. Machine-independent code optimization- common sub expression elimination, constant folding, copy propagation, dead code elimination, liveness analysis,loop optimization-strength reduction, code motion.

UNIT – V: Code Generation

Code Generation- issues in code generation, generic code generation, code generation from DAG. Machine dependent code optimization: Peephole optimization.

Text Books

1. Alfred V Aho, Monica S Lam, Ravi Sethi, Jeffrey D. Ullman, "Compilers, Principles Techniques and Tools", 2ndedition, Pearson.
2. V. Raghavan, "Principles of compiler design", 2ndedition, TMH.

Reference Books

1. Kenneth C Louden, "Compiler construction, Principles and Practice", 1st edition, Cengage.
2. Jean-Paul Trembly, Paul G. Sorenson, "The theory and practice of Compiler writing", 1stedition, McGraw-Hill.
3. Nandini Prasad, "Principles of Compiler Design", 2ndedition, Elsevier.

* * *

SOFTWARE ENGINEERING
(Common to CSE and AI&DS)
II Year – II Semester

Lecture	: 3	Practical	: 2	Internal Marks	: 30
Credits	: 4			External Marks	: 70

Course Objectives

- To familiarize with the software engineering principles to be followed in software development.
- To impart knowledge on software project management.

Course Outcomes

Upon successful completion of the course, the students will be able to

- apply suitable process model for software development based on stake holder requirements.
- estimate cost and schedule required to develop a software.
- analyze customer requirements and prepare SRS document.
- use software design principles in the design of a software.
- test a software using different testing techniques.

Course Content

UNIT – I: Introduction and Software Life Cycle Models

Evolving - From an art form to an Engineering Discipline. Life Cycle Models- Classical waterfall model, Iterative waterfall model, Prototyping model, Agile development models.

Lab Experiment

1. Assume that a software development company is already experienced in developing payroll software and has developed similar software for several customers (organizations). Assume that the software development company has received a request from certain customer (organization), which was still using manually processing of its pay rolls. For developing payroll software for this organization, which life cycle model should be used? Justify your answer.
2. Identify the criteria based on which a suitable life cycle model can be chosen for a given project development. Illustrate your answer using suitable examples.

UNIT – II: Software Project Management

Project Planning, Metrics for Project size Estimation- Lines of code- Function Point, COCOMO- Basic- Intermediate- Complete, Scheduling- work breakdown structure- Critical path method- PERT charts.

Lab Experiment

1. Compute the FP value for the grade calculation of students. Assume that it is an average complexity size project. The information domain values are as follows:

a. Number of user inputs = 13
b. Number of user outputs = 4
c. Number of user inquiries = 2
d. Number of files = 5
e. Number of external interfaces = 2
f. The total value of complexity adjustment attribute is 13

2. Assume that a system for simple student registration in a course is planned to be developed and its estimated size is approximately 10,000 LOC. The organization proposed to pay 25,000 per month to software engineers. Compute the development effort, development time, and the total cost for product development.
3. Suppose a library management system (LMS) is to be designed for an academic institution. From the project proposal, the following five major components are identified:online data entry 1.0 KLOC,data update 2.0 KLOC, file input and output 1.5 KLOC, library reports 2.0 KLOC, query and search 0.5 KLOC.The data base size and application experience are very important.Use of software tool and main storage is highly considerable.Virtual machine and its volatility can be kept low.All other cost drivers have nominal requirements.Use the COCOMO model to estimate the development effort and the development time.

UNIT – III: Requirements Analysis and Specification

Requirements gathering and analysis, software requirement specification- users of SRS document- characteristics of a good SRS document, Attributes of Bad SRS documents, important categories of customer Requirements, Functional requirements, Traceability, organization of the SRS document.

Lab Experiment

1. Suppose you have been appointed as the analyst for a large software development project. Discuss the aspects of the software product you would document in the software requirement specification (SRS) document? What would be the organization of your SRS document? How would you validate your SRS document? (Take any real time problem and prepare SRS document)

UNIT – IV: Software Design and Modelling

Software Design: Approaches to software design- function oriented design- object oriented design.

Object Modelling Using UML: Basic object orientation Concepts, Use case Model, Class diagram, Interaction diagrams, Activity Diagram, state chart Diagram, Component and Deployment diagrams.

Lab Experiment

1. To create a UML diagrams of ATM APPLICATION.
2. To create a UML diagram of BANKING SYSTEM.
3. To create a UML diagram of LIBRARY MANAGEMENT SYSTEM.

UNIT – V: Testing

Testing, Black-Box Testing- Equivalence class partitioning- Boundary value analysis, White-Box Testing- Basic concepts- statement coverage- branch coverage- multiple condition coverage- path coverage- McCabe's cyclomatic complexity metric, Integration testing, System Testing.

Lab Experiment

Design of Test cases based on requirements and design.

Text Books

1. Rajib Mall, "Fundamentals of Software Engineering",4th Edition, PHI
2. Roger S Pressman,"Software engineering A practitioner's Approach", Sixth edition McGraw Hill International Edition.

Reference Books

1. K.K Aggarwal and Yogesh Singh, "Software Engineering", 3rd edition, NewAge Publications.
2. Ian Sommerville, "Software Engineering" ,7thedition, Pearson education.
3. Pankaj Jalote,, "Software Engineering, A Precise Approach", Wiley India, 2010.
4. Booch, James Rumbaugh, Ivar Jacobson, "The Unified Modeling Language User Guide", 2ndedition, Pearson Education.

* * *