MAHAMAYA TECHNICAL UNIVERSITY
NOIDA

Syllabus

M.TECH. FIRST YEAR COURSES

Computer Science, Information Technology and Software Engineering

[Effective from the Session : 2012-13]
### Study and Evaluation Scheme

**Computer Science, Information Technology & Software Engineering**  
**Year - I, Semester - I**

<table>
<thead>
<tr>
<th>S N</th>
<th>Course code</th>
<th>Subject</th>
<th>Periods</th>
<th>Evaluation scheme</th>
<th>Subject Total</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>SESSIONAL</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>EXTERNAL</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>L</td>
<td>T</td>
<td>P</td>
<td>CT</td>
<td>AT+TA</td>
</tr>
<tr>
<td>1</td>
<td>CS910/IT910/SW910</td>
<td>Discrete Mathematics and Theory of Computation</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>40</td>
</tr>
<tr>
<td>2</td>
<td>CS911/IT911/SW911</td>
<td>Design Analysis and Algorithm</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>CS912/IT912/SW912</td>
<td>Advanced Computer Architecture</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>CS913/IT913/SW913</td>
<td>Operating Systems and Distributed Systems</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>CS914/IT914/SW914</td>
<td>Programming Practices</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>10</td>
</tr>
</tbody>
</table>

**Total** | **14** | **2** | **7** | **260** | **90** | **510** | **140** | **1000** | **20**

**NOTE:** Wherever the question paper is of 130 marks, 15 short answer questions of 2 marks each shall be asked in the question paper.
Objective: To gain an understanding of the mathematics that underlies the theory of computation. At the end of the course, the student should be able to formalize mathematical models of computations and use these formalisms to explore the inherent limitations of computations.

Unit I 8 hrs

Algebraic structures:
Semigroups, Monoids, groups, Substructures and Morphisms, Rings, Fields; Lattices, Distributive, Modular and Complemented; Boolean Algebra, Normal Forms (Conjunctive and Disjunctive), Simplification of Boolean Expressions Using Laws and K-Map.

Logic and Proofs:
Basic Logic: Propositional Logic: Logical Connectives; Truth Tables; Converse, Inverse, Contra Positive, Negation, and Contradiction, Validity; Predicate Logic; Limitations of Predicate Logic, Universal and Existential Quantifier; Modus Ponens and Modus Tollens.

Unit II 8 hrs

Proof techniques:
Notions of Implication, The Structure of Formal Proofs; Direct Proofs; Proof by Counter Example; Contraposition; Contradiction; Mathematical Induction; Simple Induction, Strong Induction. The Stable Marriage Problem.

Counting:
Counting Arguments; Pigeonhole Principle; Permutations and Combinations, Combinatories and Combinatorial Proofs, Inclusion-Exclusion, Recursive Mathematical Definitions; Well Orderings, Recurrence Relations, Generating Functions.

Unit III 8 hrs

Discrete & Continuous Probability:
Probability Spaces and Events, Conditional Probability and Bayes' rule , Random variables (Discrete and Continuous) and Expectation and Variance of Distributions- Uniform, Binomial, Exponential, Poisson & Normal distributions. Sampling and Law of Large Numbers (Central Limit Theorem ), Estimation , Bayesian Estimation and Inference.

Introduction to Automata Theory:
Alphabets, Languages & Grammars, Classification of Automata, Chomsky Hierarchy of Grammars.

Finite State Automata:
Finite state Automata - Non Deterministic and Deterministic FSA, NFSA with ε- moves, Equivalence of Deterministic and Non-Deterministic Automata

Unit IV 8 hrs

Regular Languages:
**Context Free Languages:**

**Unit V**

**Recursive and recursively enumerable Languages:**
Turing Machines, Grammars, Variations in Turing Machines. Recursive Functions, Church's Thesis. Universal Turing Machine, Closure Properties, Context Sensitive Languages and Linear Bounded Automata

**Undecidability.**
Decidability, Undecidablity/Non-Computability, Reductions. Halting Problem, Post Correspondence and Modified Post Correspondence Problems, Unsolvable Problems about Turing Machines, Unsolvable Problems about Grammars

**Discrete Mathematics:-**

**Text Books:-**

**Reference Books:-**

**Theory of Computation:-**

**Text Books:-**
2. Peter Linz, *An Introduction to Formal Languages and Automata*, Jones and Bartlett Publishers, Inc. , USA ©2006

**Reference Books:-**
Design Analysis and Algorithm  
CS911/IT911/SW911  

Pre-requisites  Programming in C, Data Structures  

Objectives  The objective of this course is to study paradigms and approaches used to design and analyze algorithms and to appreciate the impact of algorithm design in practice.  

Unit I  
8 hrs  
**Introduction to Algorithms:** Algorithms and their Performance Analysis, Space Complexity, Time Complexity, Asymptotic Notations

**Divide and Conquer Algorithms:** General Method, Analyzing Divide and Conquer Algorithms, Applications - Binary Search, Merge Sort, Heap Sort, Priority Queues, Quick Sort, Strassen’s Matrix Multiplication.

Unit II  
8 hrs  
**Advanced Analysis Techniques:** Probabilistic Analysis, Amortized Analysis.

**Review of Data Structures:** Stacks, Queues, Trees, Binary Trees, Binary Search Trees, Graphs – Representation and Traversal, B-Trees, Data Structures for Disjoint Sets

Unit III  
8 hrs  
**Advanced Data Structures:** Splay Trees, Binomial Heaps, Fibonacci Heaps

**Greedy Programming:** Fractional Knapsack Problem, Minimum Spanning Tree, Task Scheduling, Activity Selection Problem, Single Source Shortest Path.

Unit IV  
8 hrs  
**Dynamic Programming:** Longest Common Subsequence, 0-1 Knapsack Problem, Shortest Path Problems (Single-Source and All-Pair), Travelling Salesperson problem.

**Backtracking:** General Method, N-queens Problem, Sum of Subset Problem, Graph Colouring, Longest Common Subsequence.

Unit V  
8 hrs  
**Branch and Bound Technique:** General Method, 0/1 Knapsack problem, Travelling Salesperson problem.

**Max-flow:** Flow Networks, Ford-Fulkerson Method, Bipartite Matching

**Polynomials and FFT:** Representation of Polynomials, DFT and FFT, Efficient FFT implementations

**Text Books:-**


**Reference Books:-**


**CS911P/IT911P/SW911P : Design and Analysis of Algorithms Lab**

Implementation of the following using C, C++ or JAVA

1) Elementary Sorting Algorithms and Searching Algorithms
2) Divide and Conquer: Heap Sort, Priority Queues, Quick Sort
3) Binary Search Trees
4) Graph Traversal
5) Minimum Spanning Tree: Prim’s and Kruskal’s Algorithms
6) Dijkstra’s Shortest Path Algorithm
7) Dynamic Algorithms: LCS, Bellman Ford Shortest Path Algorithm
8) Backtracking Algorithms: N Queens, Sum of Subsets
9) Max Flow: Ford Fulkerson
10) FFT Algorithm
Advanced Computer Architecture
CS912/IT912/SW912

Pre-requisites
This course has a requirement that students have undergone topics of Computer Organization, process and thread in operating system and compilers in their undergraduate.

Objective
Computer architecture course aims towards study of problem specific advanced computer architecture in detail. In particular Advanced Computer Architecture includes parallel architectures. The course includes broadly parallel architecture using ILP, Data Parallel and Thread & Process Level Parallel architecture. Apart from that, the course objective is to learn parallel programming skills using Pragmatic and Non-pragmatic approach.

Unit-I: Introduction & Fundamentals: 8 hrs
The concept of computer Architecture: Interpretation of concept of computer architecture at different level abstraction, Multi level hierarchical framework, description of computer architecture,
Introduction to parallel processing: Basic concept, types of level of parallelism, classification of parallel architecture, Basic parallel techniques, relationship between language and parallel architecture.
Principles of scalable performance: Performance Metrics and Measures, Speedup Performance Law, Scalability Analysis & approaches
Processor and memory hierarchy: Design Space of Processor, ISA, CISC & RISC, Memory Hierarchy Technology, Virtual Memory Technology

Unit-II: Instruction Level Parallel Processor (Parallelism) 8 hrs
Pipelined Processors: Basic concept, ILP: Basics, Exploiting ILP, Limits on ILP, design space of pipelines, performance of pipeline, reservation table, And DLX Case Study.
VLIW architecture, Superscalar Processor: Super Scalar and super-pipeline Design

Unit-III: Data parallel Architecture 8 hrs
SIMD Architecture: Design space, fine grain SIMD architecture, coarse grain SIMD architecture
Associative and Neural Architecture, Systolic Architecture, Vector Architectures: Word length, vectorization, pipelining, and vector instruction format

Unit-IV: Thread and Process Level Parallel Architecture (MIMD Architecture) 8 hrs
Multi-threaded Architecture: Design space, computational model, Data flow architecture, hybrid multi shared architecture
Distributed memory MIMD Architecture: Design space, interconnection networks, topology, fine grain system, medium grain system, coarse grain system, Cache Coherence and Synchronization Mechanism
Shared memory MIMD Architecture.
Unit-V: Parallel Algorithm and Programming  8 hrs

MPI: Basics of MPI

Open MP: OpenMP Implementation in ‘C’, Directives: Conditional Compilation, Internal Control Variables, Parallel Construct, Work Sharing Constructs, Combined Parallel Work-Sharing Constructs, Master and Synchronization Constructs

POSIX thread: IEEE POSIX Threads: Creating and Exiting Threads, Simultaneous Execution of Threads

Text Books:
1. Advanced Computer Architectures, DEZSO SIMA, Pearson Education
2. Advanced Computer Architecture, Kai Hwang, TMH
3. Parallel Programming in C with MPI and Open MP, Quinn, TMH

Reference Books:
1. Beginning Linux Programming, Neil Matthew, Richard Stones, WROX
2. Computer Architecture and Organization, John P. Hayes, TMH

Web References:
1. Open MP Specification and Usage (www.openmp.org)
2. ACM Special Interest Group on Computer Architecture, SIGARCH, (www.sigarch.org)
Operating Systems and Distributed Systems  
CS913/IT913/SW913

Pre-requisites: Computer Organization
Objectives: To study the concepts of Operating Systems and Distributed Systems

Unit I: Process Synchronization 8 hrs
- Critical Section Problem. Requirements of an Ideal Critical Section Solution. Peterson’s Algorithm, Dekker’s Algorithm, Lamport’s Bakery Algorithm and Eisenberg McGuire Algorithm, Semaphore based critical section solutions to Producer Consumer, Reader Writer, Dining Philosopher and Sleeping Barber problems
- Mutual Exclusion and Process Coordination in Distributed Systems, Requirements of ideal mutual exclusion algorithm in Distributed Systems, Token based (Suzuki Kasami Broadcast Algorithm and Raymond’s Tree based Algorithm) and Non Token based (Lamport Algorithm, Ricart Agrawala Algorithm and Maekawa’s Algorithm) and their comparative performance analysis.

Unit II: CPU Scheduling 8 hrs
- Deadlock handling strategies in Distributed Systems, Phantom Deadlocks, Centralized, Distributed and Hierarchical Deadlock Detection Algorithms.

Unit III: Memory Management 8 hrs
- Paging, Segmentation, Segmentation with paging, Virtual Memory, Demand Paging, Page Replacement Algorithms, Thrashing.

Unit IV: Distributed Transaction Management and Concurrency Control 8 hrs

Unit V: Protection and Security 8 hrs
Case study: LINUX concepts

Text Books:-


Reference Books:-

2. Tanenbaum Andrew S. “Modern Operating system”, Pearson
5. Joel M. Cichlow, “Distributed Systems”, PHI
7. Ceri, Pelagati, “Distributed Database”, TMH

CS913P/IT913P/SW913P: Operating Systems and Distributed Systems Lab

General Concepts:

1) Simulation of MUTEX and Counting Semaphores
2) Implementation of Process Synchronization
   a) Producer Consumer Problem  
   b) Reader-Writer Problem  
   c) Dining Philosopher’s Problem  
   d) Sleeping Barber Problem
3) Simulation of CPU Scheduling Algorithms
   a) FCFS  
   b) SJF  
   c) SRTF  
   d) Round Robin  
   e) Priority Preemptive
4) Simulation of Bankers Deadlock Avoidance Algorithm
5) Simulation of Page Replacement Algorithms
   a) FIFO  
   b) LRU  
   c) Optimal

Linux:

1) Commands for general purpose utilities and handling files and its attributes
2) Simple Filters - head, tail, cut, paste, sort, uniq, tr
3) Filters using Regular Expression – grep and sed
4) Shell Programming
5) Basic System Administration – Maintaining Security, User Management, Providing Backup
Objective: To prepare students for rapid prototyping of project work.

Unit-I: 4hours
Scilab: Introduction to Scilab, Matrix operations, Scripts & Functions, if-then and while loops, plotting, Ordinary differential equations,

Unit-II: 4hours
Scilab: Polynomials, Least square fit, read/write data from files, simulation, optimization

Unit-III: 4hours
Scilab: Artificial Neural Network, Image Processing Toolbox, Signal Processing, Wavelets

Unit-IV: 4hours
Python: Python Basics: Code Structure, Variables, Basic data types, Decision making and iterations, if-else control structure, String handling, Functions: Passing arguments and returning a value, scope of variables, predicate and recursive functions.

Unit-V: 4hours
Python: Compound data structures: List comprehensions, tuples and dictionaries, Object Oriented Constructs: Classes and Objects in Python, Constructors and Static methods, controlling attribute access, inheritance and overloading, iterators, Modules: Standard modules, dir() function and Packages.

Books on SCILAB
3. Stephen Campbell, Jean-Philippe Chancelier and Ramine Nikoukhah, Modeling and Simulation in SCILAB / SCICOS

Web resource:

Books on Python:
1. Learning Python by Mark Lutz (O'Reilly & Associates)
2. Core Python Programming by Wesley J. Chun (Pearson Education)

Web resource
www.python.org

CS914P/IT914P/SW914P: Programming Practices Lab

The lab will be that of 6 hours per week out of which 2 hours is for lab on SciLab/Python and 4 hours for the mini-project.

The mini-project will be allocated by the concerned faculty teaching the course. The students have the choice to implement it any language of their choice. The report of the mini-project will consist of an analysis of problem statement, requirement analysis, system design, justification of the choice of programming language/environment and complete code with results. The mini-project will be evaluated by an external examiner.