College of Engineering, Pune

(An Autonomous Institute of Govt. of Maharashtra, Permanently Affiliated to S.P. Pune University)

Department of Computer Engineering and Information Technology

Curriculum Structure & Detailed Syllabus (UG Program)

Final Year B. Tech. Computer Engineering

(Revision: 2015-19, Effective from: A.Y. 2018-19)

INDEX

Program Education Objectives (PEOs)

I. To create graduates with sufficient capabilities in computer engineering who can become researchers, entrepreneurs and software professionals to satisfy the needs of the core industry, research, academia and society at large.

II. To provide opportunity to learn the latest trends in computer engineering and prepare for lifelong learning process.

III. To make the students aware of professional ethics of the software Industry and prepare them with basic soft skills essential for working in community and professional teams.

.

Program Outcomes (POs)

At the end of the program, the graduates will

1. Computer engineering knowledge: Apply the knowledge of mathematics, science, computer engineering fundamentals, and emerging fields of computer engineering to the solution of complex real life problems.

2. Problem analysis: Identify, formulate, review research literature, and analyze complex computer engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and computer engineering sciences.

3. Design/development of solutions: Design solutions for complex computer engineering problems and design system components or processes that meet the specified needs considering public health and safety, and the cultural, societal, and environmental considerations.

4. Conduct investigations of complex problems: Use knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern computer engineering and IT tools including FOSS tools.

6. Social responsibility: Apply reasoning informed by the contextual knowledge to assess social, health, safety, legal and cultural issues and the consequent responsibilities.

7. Environment and sustainability: Understand the impact of the professional computer engineering solutions in socio-environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Demonstrate knowledge and practice of engineering ethics.

9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary, multi-cultural settings.

10. Communication: Communicate effectively with engineering community and with society at large, demonstrating ability to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of the computer engineering, finance and management principles.

12. Life-long learning: Recognize the need for, and ability to engage in independent and life-long learning.

Note: The cells filled in with ✓ indicate the fulfillment/correlation of the concerned PEO with the PO.

List of Abbreviations

Abbreviation	Title
S.P. P.U.	Savitribai Phule Pune University
A.Y.	Academic Year
BSC	Basic Science Course
EFC	Engineering Foundation Course
MLC	Mandatory Learning Course
ILOE	Institute Level Open Elective Course
SLC	Self Learning Course
HSMC	Humanities/Social Sciences/Management Course
LLC	Liberal Learning Course
SBC	Skill Based Course
PCC	Program Core Course
DEC	Department Elective Course
LC	Laboratory Course

B. Tech. Computer Engineering
Sem VII: Effective AY 2018-19

			Teaching Scheme			
Sr. No.	**Course Type**	**Course Name**	**L**	**T**	**P**	**Credits**
	ILOE	Institute level Open Elective [Science/Technology/Engg.] [To be offered to other Departments] Introduction to Algorithms	3	0	0	3
1	LLC	Liberal Learning Course	1	0	0	1
2	DEC	Department Elective-II	3	0	0	3
3	DEC	Department Elective-III	3	0	0	3
4	SBC	Project Stage-I	0	0	4	2
5	PCC	Parallel Computer Architecture and Programming	3	0	0	3
6	PCC	Cryptography and Network Security	2	1	0	3
7	LC	Department Elective-II Laboratory	0	0	2	1
8	LC	Department Elective-III Laboratory	0	0	2	1
9	LC	Parallel Computer Architecture and Programming Laboratory	0	0	2	1
10	LC	Cryptography and Network Security Laboratory	0	0	2	1
		Total Academic Engagement and Credits	**15**	**1**	**12**	**22**

B. Tech. Computer Engineering
Sem VIII: Effective AY 2018-19

			Teaching Scheme			
Sr. No.	**Course Type**	**Course Name**	**L**	**T**	**P**	**Credits**
	MLC	Intellectual Property Rights	1	0	0	0
1	LLC	Liberal Learning Course	1	0	0	1
2	PCC	Compiler Construction	3	0	0	3
3	DEC	Department Elective-IV	2	1	0	3
4	DEC	Department Elective-V	3	0	0	3
5	LC	Project Stage-II	0	0	12	6
6	LC	Compiler Construction Laboratory	0	0	2	1
7	LC	Department Elective-V Laboratory	0	0	2	1
		Total Academic Engagement and Credits	**9**	**2**	**14**	**18**

Sem VII Minor Course: Database Management Systems
Sem VIII Minor Course: Internet Technologies

Sem VII Honors Course: Advanced Computer Networks
Sem VIII Honors Course: Multicore Technology

List of Department Elective Courses

Sem VII	**Department Elective-II** **CBD: Cloud and Big Data Platforms** **FP: Functional Programming** **IR: Information Retrieval** **SDP: Software Design Patterns** **IOT: Internet of Things** **Subjects in Association with Domain Experts**
Sem VII	**Department Elective-III** **AUP: Advanced UNIX Programming** **ES: Embedded Systems** **OOMD: Object Oriented Modeling and Design** **NLP: Natural Language Processing** **LP: Language Processors (only for IT)** **Subjects in Association with Domain Experts**
Sem VIII	**Department Elective-IV** **SA: System Administration** **S&V: Storage and Virtualization** **MAN: Mobile and Ad-hoc Networks** **CFDR: Computer Forensics and Data Recovery** **CB: Computational Biology** **Subjects in Association with Domain Experts**
Sem VIII	**Department Elective-V** **MT: Multicore Technology** **ADBMS: Advanced Database Management Systems** **EC: E-commerce** **CS: Cyber Security** **DS: Distributed Systems (only for CE)** **Subjects in Association with Domain Experts**

Semester-VII

(CT) Parallel Computer Architecture and Programming

Teaching Scheme:
Lectures : 3 Hrs/week

Examination Scheme:
Assignment/Quizzes – 40 marks
End Sem Exam - 60 marks

Course Outcomes:

Students will be able to:

1. Justify the need of high performance provided by parallel computer architecture.
2. Demonstrate quantitative design principles of parallel computing systems.
3. Measure and analyze performance through different benchmarks.
4. Comprehend and differentiate various computer architectures.
5. Understand the building blocks of cluster setup.

Unit I :Fundamentals of Computer Design: Defining Computer Architecture, Trends in Technology, Trends in Power in Integrated Circuits, Trends in Cost, Dependability, Measuring, Reporting and Summarizing Performance, Quantitative Principles of Computer Design, Classification of Computer Architectures, Recent Computing Trends Top 500 Ratings, Fundamentals of Computer Design, Basic and Intermediate concepts of pipelining , Pipeline Hazards, Pipelining Implementation issues. **[6 Hrs]**

Unit II :Instruction-Level Parallelism and Its Exploitation: Instruction-Level Parallelism: Concepts and Challenges, Basic Compiler Techniques for Exposing ILP, Reducing Branch Costs with Prediction, Scheduling, Overcoming Data Hazards with Dynamic Scheduling, Dynamic Scheduling: Algorithm and Examples, Hardware-Based Speculation, Studies of the Limitations of ILP, Limitations on ILP for Realizable Processors, Hardware versus Software Speculation, ILP Support to Exploit Thread-Level Parallelism. **[7 Hrs]**

Unit III :Data-Level Parallelism in Vector, SIMD, and GPU Architectures: Vector Architecture, SIMD Instruction Set Extensions for Multimedia, Graphics Processing Units, CPU/GPU architecture comparison, Detecting and Enhancing Loop-Level Parallelism, Data parallelism and SPMD programming model, Nvidia GPU Case study and Programming Model, Example of ARM Heterogeneous Architecture. **[7 Hrs]**

Unit IV :Thread-Level Parallelism: Introduction to Shared Memory Architectures, Loosely and Tightly coupled multiprocessors, Centralized Shared-Memory Architectures, Snoopy Bus Cache Coherence, Performance of Shared-Memory Multiprocessors, Distributed Shared Memory and Directory Cache Coherence, Basics of Synchronization, Models of Memory Consistency. **[7 Hrs]**

Unit V :Memory Hierarchy Design: Cache Performance, Basic Cache Optimizations, Virtual Memory, Protection and Examples of Virtual Memory, Advanced Optimizations of Cache

Performance, Memory Technology and Optimizations, Protection: Virtual Memory and Virtual Machines, The Design of Memory Hierarchies, Study of Memory Hierarchies in different Processors. **[6 Hrs]**

Unit VI :Parallel Programming Paradigms for different Architectures: Cluster: Cluster and Network of Workstations (COW and NOW), Different ways of building a cluster, Parallel Programming Models: Message Passing, Data Parallel, MPI/PVM, Shared Memory Programming OpenMP, CUDA Programming, Parallel Algorithm examples: Matrix Multiplication, Sorting, Introduction to Parallel Programming Languages. Warehouse-Scale Computers: Architecture, Programming Model and Workloads **[7 Hrs]**

Text Books:

- John L Hennessy, David A Patterson, “Computer Architecture: A Quantitative Approach”, Sixth Edition, Morgan Kaufmann, 2018, ISBN: 978-0-12-811905-1.
- Kai Hwang, Naresh Jotwani, “Advanced Computer Architecture”, Third Edition, Tata McGraw-Hill Edition, 2016, ISBN: 978-0-07-070210-3.

Reference Books:

- D. E. Culler, J. P. Singh, and A. Gupta, “Parallel Computer Architecture”, Second Edition, Morgan Kaufmann, 2017, ISBN:978-1-4987-7271-6.
- McCool, Michael D., Arch D. Robison and James Reinders, “Structured Parallel Programming: Patterns for Efficient Computation”, Morgan Kaufmann, 2012, ISBN: 978-0-12-415993-8.

(CT-) Parallel Computer Architecture and Programming Laboratory

Teaching Scheme:
Laboratory : 2 Hours/week

Examination Scheme:
Continuous evaluation: 30 Marks
Mini Project/ Assignments: 20 marks
End Semester Exam: 50 Marks

Course Outcomes:

Students will be able to:

1. Evaluate performance using different benchmarks used in high performance computing systems.
2. Implement shared memory programs using pthreads and OpenMP.
3. Implement message passing programs in distributed environment.
4. Program GPU architecture using CUDA.
5. Demonstrate the different steps involved in building of a simple cluster.

List of Assignments:

1. Study of different benchmarks used to evaluate performance of different systems.
    a. Study of different performance parameters like execution time, throughput etc.
    b. Application benchmarks suits like PARSEC, SPLASH, SPEC.
2. Program to understand shared memory paradigm using pthreads and related issues.
    a. Matrix multiplication using pthreads.
    b. Synchronization and thread scheduling.
3. Program to execute dense matrix multiplication using pthreads.
    a. Explicit code writing to handle dense matrix multiplication and execution time.
    b. Different algorithms for dense matrix multiplication.
4. Program to execute dense matrix multiplication using OpenMP clauses.
    a. Usage of #pragma directives along with OpenMP clauses.
    b. Comparison with pthread implementation.
5. Study of Cluster building steps - MPI Cluster setup and overview of different routines.
    a. Different steps to build a MPI cluster over LAN.
    b. Master-Slave concept and different MPI routines.
6. Program to implement point to point communication using MPI routines.
    a. Parallelizing Trapezoidal Rule using MPI_Send and MPI_Reveive.
7. Program to implement collective communication using MPI routines.
    a. Gather, Scatter and Broadcast operations.
    b. Matrix-Vector multiplication execution.
8. Program to implement Merge sort/ Graph Computation algorithm using MPI routines.
    a. Steps in parallelizing Merge Sort/ Matrix Partitioning.

    b. Execution time comparison of Merge sort/ Graph Computation with normal implementation.
9. Program to execute matrix multiplication using CUDA.
    a. Basic CUDA host, device and memory constructs.
    b. Thread- warp, block, grid usage.
10. Program to study working of Click parallel programming language.
    a. Simple program to execute Fibonacci example and performance evaluation.
11. Program to implement Map-Reduce parallelism for Warehouse -Scale Computer.
    a. Parallelism using Map-Reduce programming model.
    b. Example of word count process with key-value pair.

Reference Books:

- Peter S. Pacheco, “An Introduction to Parallel Programming”, Morgan Kaufmann, Elsevier Series, 2011, ISBN:978-0-12-374260-5.
- Jason Sanders, Edward Kandrot, “CUDA by Example: An Introduction to General Purpose GPU Programming”, 2011, ISBN:978-0-13-138768-3.

(CT)Cryptography and Network Security

Teaching Scheme:
Lectures : 2 Hrs/week
Tutorial : 1 Hrs/week

Examination Scheme:
Assignment/Quizzes – 40 marks
End Sem Exam - 60 marks

Course Outcomes:

Students will be able to:

1. Explain the concepts related to applied cryptography, including plaintext, ciphertext, symmetric cryptography, asymmetric cryptography, and digital signatures
2. Apply concepts of finite mathematics and number theory.
3. Demonstrate the understanding of common network vulnerabilities and attacks, defence mechanisms against network attacks, and cryptographic protection mechanisms.
4. Detect possible threats to different defence mechanisms and different ways to protect against these threats.

Unit I :Introduction: Cryptography and modern cryptography, Need of security, Security services, Basic network security terminology, Security attacks, Classical cryptosystems and their cryptanalysis, Operational model of network security **[4 Hrs]**

Unit II: **Mathematical Foundations**: Prime Number, relatively prime numbers, Modular Arithmetic, Fermat's and Euler's Theorem, The Euclidean and Extended Euclidean Algorithms, The Chinese Remainder Theorem, Discrete logarithms **[4 Hrs]**

Unit III :Symmetric Key Ciphers: Symmetric Key Ciphers, Feistel Networks, Modern Block Ciphers, Modes of Operation, Cryptanalysis of Symmetric Key Ciphers: Linear Cryptanalysis, Differential Cryptanalysis **[6 Hrs]**

Unit IV :Asymmetric Cryptography: RSA, Key Distribution and Management, Diffie-Hellman Key Exchange, Elliptic Curve Cryptography, hash functions: The Merkle Damgard Construction, Message Digest algorithms: MD5, Secure Hash algorithm (SHA), Message Authentication Codes **[6 Hrs]**

Unit V: Authentication and Web Security: Digital Signatures, Authentication Protocols, Kerberos, X.509 Digital Certificate Standard, Pretty Good Privacy, Secure Socket Layer, Secure Electronic Transaction. Zero knowledge proof **[4 Hrs]**

Unit VI :Network Security: Intruders, Intrusion Detection, Password Management, Worms, viruses, Trojans, Virus Countermeasures, Vulnerabilities in TCP/IP model, Firewalls, Firewall Design Principles **[4 Hrs]**

Text Books:

- V. K. Pachghare, “Cryptography and Information Security”, 2nd edition, PHI Learning, ISBN: 978-81-203-5082-3.
- Charlie Kaufman, Radia Perlman, and Mike Speciner, Network Security: Private Communication in a Public World, Prentice Hall, ISBN 0-13-046019-2.

Reference Books:

- William Stallings, “Cryptography and Network Security, Principles and Practices”, Pearson Education, Fifth Edition, ISBN: 0-13-60970-9.
- Christopher M. King, “Security architecture, design deployment and operations”, Curtis Patton and RSA Press, ISBN: 0072133856.
- Stephen Northcatt, LenyZeltser, “INSIDE Network Perimeter Security”, Pearson Education Asia, Second Edition, ISBN: 978-0735712324.
- Robert Bragge, Mark Rhodes, HeithStraggberg, “Network Security the Complete Reference”, Tata McGraw Hill Publication, ISBN: 9780072226973.

(CT) Cryptography and Network Security Lab

Teaching Scheme:
Laboratory: 2 Hrs/week

Examination Scheme:
Continuous evaluation: 50 Marks
Oral Exam: 50 Marks

Course Outcomes:

Students will be able to:

1. Analyse the optimal features and time required for an encryption technique.
2. Implement cryptographic algorithms in any programming language.
3. Demonstrate the ability to detect attacks on a system and tackle it.
4. Write a security application to protect a system from some attacks.

List of Assignments:

1. Study papers on a network security topic and write a study report
    1. Wireless Network Security,
    2. Key Exchange Protocols,
    3. Block chain.
2. Implement any one classical encryption technique in any programming language.
3. Design and implement a symmetric encryption algorithm based on Feistel structure.
4. Demonstrate how Diffie-Hellman key exchange works with Man-In-The-Middle attack.
5. Study different approaches for Anti-virus software and write one document.
    a) Examine files to look for viruses by means of a virus dictionary
    b) Identifying the suspicious behavior from any computer program which might indicate infection
6. Study and demonstrate system hacking and write a report.
    a) How to crack a password?
    b) How to use Ophcrack / Crowbar / John the Ripper / Aircrack-ng to Crack Passwords
7. Develop a mini project on
    a) a hack tool to break the security of a system. OR
    b) a tool to protect the system from the hack tool.

ILOE (CT) Introduction to Algorithms

Teaching Scheme:
Lectures : 3 Hrs/week

Examination Scheme:
Assignment/Quizzes – 40 marks
End Sem Exam - 60 marks

Course Outcomes:

Students will be able to:

1. Determine different time complexities of a given algorithm
2. Develop algorithms using various design techniques for a given problem.
3. Determine a design technique and design an algorithm using that technique for a given problem.

Unit 1: : Basic Data Structures
Abstract data types, data types; Stack, Queue, Lists, Graphs; various implementations of the data structures **[8 Hrs]**

Unit 2: Asymptotic notation
Objectives of time and space analysis of algorithms; Order notations (O, Ω, θ notations); (Best average and worst case) time complexity of algorithms such as bubble sort, selection sort, insertion sort, heap sort etc.; Time complexity of recursive programs using recurrence relations **[6 hrs]**

Unit 3: Divide and Conquer:
Binary Search, Quicksort, Mergesort, Strassen's matrix multiplication **[6 Hrs]**

Unit 4: Greedy Algorithms:
Knapsack problem, Job sequencing with deadlines, Minimum cost spanning trees (Prim's and Kruskal's algorithms), Optimal merge patterns, Single source shortest paths [**8 Hrs]**

Unit 5: Dynamic Programming:
All pairs shortest paths, 0-1 Knapsack, Travelling salesperson problem, Chained matrix multiplication, Longest common subsequence etc **[8 Hrs]**

Unit 6: Backtracking:
8-Queens problem, Sum of subsets, Graph coloring, 0-1 Knapsack problem. **[4 Hrs]**

Text Books
1. Thomas Cormen, Charles Leiserson, Ronald Rivest and Cliford Stein, "Introduction to Algorithms", The MIT Press - 3rd edition, ISBN-13: 978-0262033848
2. Ellis Horowitz, Sartaj Sahni and Sanguthevar Rajasekaran, "Fundamentals of Computer Algorithms", Universities Press, second edition (2008), ISBN-13: 978-8173716126

Reference Books

1. Gilles Brassard and Paul Bratley, “Fundamentals of Algorithmics”, PHI, ISBN-13: 978-8120311312
2. Jon Kleinberg and Éva Tardos, “Algorithm Design”, Pearson Education India, ISBN-13: 978-9332518643

List of Assignment topics:

1. Implementation of stack, queue and list data structures and one application each.
2. Verification of (worst and average) complexity of Quicksort
3. Implementation of Heap sort.
4. Implementation of greedy algorithm for knapsack problem
5. Minimum Spanning tree as an example of Greedy approach [Prim’s or Kruskal’s algorithm]
6. Implementation of Single Source and All Pairs Shortest algorithms
7. Finding longest common subsequence of given two sequences using dynamic programming
8. Implementation of sum of subsets problems using backtracking

   This is an indicative list. The instructor is encouraged to update the list.

Semester-VIII

(CT) Compiler Construction

Teaching Scheme:
Lectures : 3 Hrs/week

Examination Scheme:
Assignment/Quizzes – 40 marks
End Sem Exam - 60 marks

Course Outcomes:

Students will be able to:

1. Demonstrate the understanding of different phases of compilation.
2. Demonstrate the ability to generate and code lexical and syntax analyzer.
3. Analyze and differentiate different parsing techniques and syntax directed translation schemes and choose the optimal parsing technique.
4. Apply different intermediate code generation and code optimization techniques for various statements.

Unit I : Introduction: Translator Issues, Introduction to phases of compilation, Front end and Back end model of compiler, Cross compiler, Incremental compiler, Boot strapping. **[6 Hrs]**

Unit II : Lexical Analysis: Concept of Lexical Analysis, Regular Expressions, Tokens, Lexemes, and Patterns, Block Diagram of Lexical analyser, Hand coding of Lexical analyser, Introduction to LEX Tool and LEX file specification, Error detection and recovery in LEX. **[6 Hrs]**

Unit III : Syntax Analysis: Context Free Grammars(CFG),Concept of parsing, Parsing Techniques, Top-Down Parsers : Introduction, Predictive Parsing - Removal of left recursion, Removal of left factoring, Recursive Descent Parsing, Predictive LL(k) Parsing Using Tables, Bottom Up parsing: Introduction, Shift-Reduce Parsing Using the ACTION/GOTO Tables, Table Construction, SLR(1), LR(1), and LALR(1) Grammars, Introduction to YACC Tool & YACC file specification, Error detection and recovery in YACC. **[8 Hrs]**

Unit IV : Semantic Analysis & Intermediate Representation: Need of semantic analysis, Abstract Parse trees, Syntax directed definitions, Syntax directed translation schemes, Symbol Tables, Symbol Table management, Type Checking, Intermediate code generation: Intermediate languages, Design issues, Intermediate code representations: three address code, abstract & concrete syntax trees,. **[8 Hrs]**

Unit V : Code Optimization: Introduction, Principal sources of optimization, Machine Independent Optimization, Machine dependent Optimization, Various Optimizations: Function preserving transformation, Common Sub-expressions, Copy propagation, Dead-code elimination, Loop Optimizations, Code Motion, Induction variables and strength reduction, Peephole Optimization, Redundant –instruction elimination, Prescient store optimizations; Inter-procedural, whole-program or link-time optimization. **[6 Hrs]**

Unit VI : Run-Time Memory Management & Code generation: Model of a program in execution, Stack and static allocation, Activation records , Issues in the design of code generation, Target machine description, Basic blocks & flow graphs, Expression Trees, Unified algorithms for instruction selection and code generation. **[6 Hrs]**

Text Books:

- Alfred V. Aho, Monica S. Lam, A. V. R. Sethi and J.D. Ullman, “Compiler Principle, Techniques and Tools” Addison Wesley, Second Edition, ISBN: 978-0321486813.

Reference Books:

- Barrent W. A., J. D. Couch, “Compiler Construction: Theory and Practice”, Computer Science series, Asian student edition, ISBN: 9780574217653.
- Dhamdhere D.M., “Compiler Construction Principle and Practice”, Macmillan India, New Delhi, 2003, Second Edition, ISBN: 0333904060.
- Ravendra Singh, Vivek Sharma, Manish Varshney, “Design and Implementation of Compiler”, New Age Publications, 2008, ISBN: 978-81-224-2398-3.
- Holub, A.J., “Compiler design in C”, Prentice Hall, 1994, ISBN: 978-0133049572.
- John Levine, Tony Mason & Doug Brown, “Lex and Yacc”, O‟Reilly, 1995, Second Edition, ISBN: 978-1565920002.

(CT) Compiler Construction Laboratory

Teaching Scheme:
Laboratory : 2 hours per week

Examination Scheme:
Continuous evaluation: 50 Marks
End Sem Exam: 50 Marks

Course Outcomes:

Students will be able to:

1. Code lexical analyzers for different languages using hand coding technique and Lex tool
2. Write a parser and semantic analyzer for different Context-Free Grammars using Yacc tool.
3. Implement different representations of Intermediate code
4. Demonstrate ability to optimize intermediate code using different techniques

List of Assignments:

1. Design a lexical analyzer for a subset of C language using Lex tool.
2. Design a hand-coded lexical analyzer for a subset of C language, draw the transition diagrams and then implement the lexical analyzer in C language.
3. Design a scientific calculator using Lex & Yacc or PLY or ANTLR tools.
4. Write a code for finding FIRST & FOLLOW of a grammar.
5. Design a SQL parser / html parser.
6. Implement a SLR parser for a given grammar.
7. Implement a static semantics analyzer.
8. Implement an intermediate code generator in three-address code form represented in quadruples.
9. Implement different optimization techniques on intermediate code.

Minor Course Sem VII: (CT) Database Management Systems

Teaching Scheme:
Lectures : 3 Hrs/week

Examination Scheme:
Assignment/Quizzes – 40 marks
End Sem Exam - 60 marks

Course Outcomes:

Students will be able to:

1. Construct Entity-Relationship Model for given applications and Relational Model for the same.
2. Design and write SQL queries for given problems
3. Normalization to database design
4. Describe, compare, and solve analytical problems based on storage mechanisms, and transactions

Unit I : Introduction: Basic Concepts: Database System Application, Purpose of Database Systems, View of data, Data base languages, Database Architecture: Components of DBMS and overall structure of DBMS **[4 Hrs]**

Unit II : E-R and Relational Model: Database Design and the E-R Model: Modeling, entity, attributes, relationships, constraints, Components of E-R Model. Relational Model: Basic concepts. Attributes and domains, concept of integrity and referential constraints, schema diagram.. **[6 Hrs]**

Unit III : Relational Algebra and SQL: Fundamental relational algebra operations, Additional relational algebra operations, Extended relational algebra operations ,null values, Modification to database, SQL: Basic structure and operations, Aggregate functions ,Nested subqueries , Complex queries, Views. **[8 Hrs]**

Unit IV : Relational Database Design: Basic concept of normalization, Decomposition using Functional dependencies. **[6 Hrs]**

Unit V : Indexing and Hashing : Basic of query processing, Basic concepts, Indices, B+ trees and B tree index file, static and Dynamic hashing **[6 Hrs]**

Unit VI : Transactions and Concurrency control: Transaction: Basic concepts, States, concurrent execution, serializability, Recoverability, Isolation. Concurrency control: timestamps and locking protocols, validation Based protocols, Multiple granularity protocols, Deadlock handling, Recovery **[8 Hrs]**

Text Books:

- Abraham Silberschatz, Henry F. Korth, S. Sudarshan, “Database system concepts”, 5th Edition, McGraw Hill International Edition.
- Raghu Ramkrishnan, Johannes Gehrke, “Database Management Systems”, Second Edition, McGraw Hill International Editions

Reference Books:

- Rob Coronel, “Database systems : Design implementation and management”, 4th Edition, Thomson Learning Press.
- Ramez Elmasri and Shamkant B. Navathe, “Fundamental Database Systems”, 3rd Edition, Pearson Education, 2003

Minor Course Sem VIII: (CT) Internet Technologies

Teaching Scheme:
Lectures : 3 Hrs/week

Examination Scheme:
Assignment/Quizzes – 40 marks
End Sem Exam - 60 marks

Course Outcomes:

Students will be able to:

1. Describe, draw diagrams, solve analytical problems based on, structure of computer networks.
2. Describe, solve analytical problems based on, networking protocols
3. Create WWW pages to serve as front-end for Internet applications
4. Demonstrate the ability to write server side and client side programs.

Unit I: Introduction: Introduction to Internet, Evolution of Internet, Internet applications, Protocols and Standard, Important Internet Services: WWW, File Transfer, Email, DNS, Remote Access: Telnet, SSH, Search Engines, Browser Basics **[6 hrs]**

Unit II: Computer Network – Overview: Introduction to computer network, OSI Model: Layered Architecture, Functions of the layers, Peer-to-Peer Processes, Encapsulation, TCP/IP protocol suite: Addressing, Packet forwarding, Introduction to UDP. **[8 hrs]**

Unit III: Web Essentials: Clients, Servers, Communication, HTTP Request Message, HTTP Response Message, Web Clients, Generations of web applications. **[6 hrs]**

Unit IV : Markup languages : An Introduction to HTML, Fundamental HTML Elements head, body etc. Document publishing, Introduction to XML. Introduction to Cascading Style Sheets, CSS features, CSS syntax, Style properties of text, box, layout, list, table, cursor etc **[6 hrs]**

Unit V : Client-Side Programming : Introduction to JavaScript, Basic Syntax, Variables and Data Types, Statements, Operators, literals, functions. Javascript Objects – properties, references, methods, constructors. Arrays, other built-in objects. Debugging javascript. Browsers. **[6 hrs**]

Unit VI : Server-Side Programming : PHP - Client Request – form data, request headers. Server Response - HTTP Status Codes, HTTP Response Headers. Sessions, Cookies, URL Rewriting. Introduction to Web services. **[8 hrs]**

Text Books:

- Jeffrey C.Jackson, "Web Technologies : A Computer Science Perspective", Pearson Education, 2nd edition, 2007
- B. A. Forouzan and Firouz Mosharraf, Computer Networks, A Top-Down Approach, Tata McGraw-Hill, 2012

Reference Books:

- A S Tanenbaum, “Computer Networks”, 4th Edition, Pearson Education, ISBN 9788177581652
- Marty Hall, Larry Brown, "Core Web Programming", Pearson Education, 2nd Edition, 2001.
- Robert. W. Sebesta, "Programming the World Wide Web", Pearson Education, 4th Edition, 2007.
- H.M. Deitel, P.J. Deitel and A.B. Goldberg, "Internet & World Wide Web: How To Program", Pearson Education, 3rd Edition, 2006.

Honors Course Sem VII: Advanced Computer Networks

Teaching Scheme:
Lectures : 3 Hrs/week

Examination Scheme:
Assignment/Quizzes – 40 marks
End Sem Exam - 60 marks

Course Outcomes:

Students will be able to

6. Understand issues in the design of network processors and design network systems
7. Analyze different possible solutions for communications at each network layer
8. Simulate working of wired and wireless networks to understand networking concepts
9. Develop solutions by applying knowledge of mathematics, probability, and statistics to network design problems
10. Understand and Compare various storage and networking technologies

Unit 1: Internetworking: Routing Algorithms, Congestion Control, Quality of Service, Queue Management, High Speed Networks, Performance Modeling and Estimation

Unit 2: IPv6: IPv4 deficiencies, patching work done with IPv4, IPv6 addressing, multicast, Anycast, ICMPv6, Neighbour discovery, Routing

Unit 3: Software Defined Networking and OpenFlow: Centralized and Distributed Control and Data Planes, SDN Controllers, Data Center Concepts, Network Function Virtualization, Mininet, Programming SDNs, Openflow Switch, Wire Protocol, Openstack Neutron plug-in

Unit 4: Ad Hoc Wireless Networks : MAC Protocols for Ad Hoc Wireless Networks, Routing Protocols for Ad Hoc Wireless Networks, Multicast routing in Ad Hoc Wireless Networks, Transport Layer and Security Protocols for Ad Hoc Wireless Networks, Quality of Service in Ad Hoc Wireless Networks.

Unit 5: Network management Protocols: SNMPv1 Network Management: Organization and Information Models, SNMPv2: major changes, SNMPv3, RMON, Network Management Tools, Systems, and Engineering, Network Management Applications

Unit 6: Storage and Networking: Storage and Networking Concepts, Fiber Channel Internals, Fiber Channel SAN Topologies, Fiber Channel Products, IP SAN Technology, IP SAN Products, Management of SANs, SAN Issues

Text Books:

- Thomas D NAdeau and Ken Grey, Software Defined Networking, O'Reilly, 2013
- Pete Loshin IPv6, Theory, Protocols and Practiice, Morgan Kaufmann, 2nd Edition, 2004
- Mani Subramanian, Timothy A. Gonsalves,N. Usha Rani; Network Management: Principles and Practice; Pearson Education India, 2010

References

- William Stallings, High-Speed Networks and Internets, Pearson Education, 2nd Edition, 2002.
- C. Siva Ram Murthy, B.S. Manoj, Ad Hoc Wireless Networks: Architectures and Protocols, Prentice Hall, 2004
- Muthukumaran B, Introduction to High Performance Networks, Tata Mc Graw Hill, 2008
- Tom Clark, Designing Storage Area Networks, A Practical Reference for ImplementingFibre Channel and IP SANs, Addison-Wesley Professional, 2nd Edition, 2003.

Honors Course Sem VIII: (CT) Multicore Technology

Teaching Scheme:
Lectures : 3 Hrs/week

Examination Scheme:
Assignment/Quizzes – 40 marks
End Sem Exam - 60 marks

Course Outcomes:

Students will be able to:

1. Understand the working principles of multicore architectures.
2. Optimize performance of multicore systems.
3. Specify the necessity of GPU.
4. Comprehend and differentiate between CPU and GPU.
5. Identify and demonstrate the need of domain specific architectures.

Unit I :Introduction to Multicore Systems: Fundamentals, The Era of Multicore Machines, Unicore vs Multicore, Understanding Performance, Shared Memory Multicore Systems, Distributed Memory Multicore Systems, Hybrid Systems, Symmetric and Asymmetric Multicore Systems – Overview of Multithreading – Multithreading in different forms – Homogeneous and Heterogeneous Multicore systems – Examples of different Multicore Systems. **[5 Hrs]**

Unit II :Cache Memory: Large Cache Design: Shared vs. Private Caches, Centralized vs. Distributed, Shared Caches – Coherence: Snooping, based cache coherence protocol, directory, based cache coherence protocol, Uniform Cache Access, Non, Uniform Cache Access, S, NUCA, D, NUCA, Inclusion, Exclusion – Examples of different Cache Organization, Consistency Models – Case Study. **[7 Hrs]**

Unit III :Performance and Optimizations for Multicore Systems: Select the right "core", Improve serial performance, Achieve proper load balancing, Improve data locality, Reduce or eliminate false sharing, Use of affinity scheduling, Lock granularity and frequency, Remove synchronization barriers, Minimize communication latencies, Use of thread pools, Managing thread count, Use of parallel libraries. **[4 Hrs]**

Unit IV :Programming Multicore Systems: Programming models for Multicore Systems – Shared Memory Programming using pthreads, Shared Memory Programming using OpenMP – Use of OpenMP compiler directives – #pragma with different clauses – Understanding parallelized loops – Synchronization Constructs towards dependencies – Function parallel program, OpenMP Library Functions, OpenMP Environment Variables, Compilation, Debugging, Performance. **[8 Hrs]**

Unit V :Special Case Graphics Processing Unit: CPU architecture, GPU hardware – CPU and GPU: Design Goals – Compute levels – Case Study: Nvidia GPU – GPGPU, Compute Unified Device Architecture (CUDA) Programming model – Applications of CUDA, Threads, Blocks, Grids – Memory management – Examples – Alternatives to CUDA. **[7 Hrs]**

Unit VI :Domain Specific, Architectures: Guidelines for domain specific architectures – Deep Learning Architecture, Google's Tensor Processing Unit (TPU) for Deep Neural Networks (DNNs), Pixel Visual Core, a Personal Mobile Device Image Processing Unit. **[5 Hrs]**

List of Assignments:

1. Program for matrix vector multiplication using pthreads.
2. Program to implement data sharing and thread scheduling.
3. Program for matrix vector multiplication using OpenMP clauses.
4. Program to implement sorting using OpenMP clauses.
5. Program to find n^{th} term of Fibonacci sequence to implement function/ task parallel aspect using OpenMP clauses.
6. Setting up CUDA environment on Linux and first CUDA program.
7. Program to implement Matrix-Matrix Multiplication using CUDA.
8. Program to implement sorting using CUDA.
9. Program to Histogram calculation using CUDA.
10. Program to create threads using default stream in CUDA.
11. CUDA for Deep Learning - A Case Study.

Text Books:

- Gerassimos Barlas, "Multicore and GPU Programming: An Integrated Approach", Morgan Kaufmann, 2015, ISBN: 978-0-12-417137-4.
- Rob Oshana, "Multicore Application Development Techniques: Applications, Tips and Tricks", Elsevier, 2016, ISBN: 978-0-12-800958-1.
- John L Hennessy, David A Patterson, "Computer Architecture: A Quantitative Approach", Sixth Edition, Morgan Kaufmann, 2018, ISBN: 978-0-12-811905-1.

Reference Books:

- Rajeev Balasubramonian, Norman P. Jouppi, and Naveen Muralimanohar, "Multi-Core Cache Hierarchies", Morgan & Claypool Publishers, 2011,ISBN: 9781598297546.
- Daniel J. Sorin, Mark D. Hill, David A. Wood "A Primer on Memory Consistency and Cache Coherence", Morgan & Claypool Publishers, 2011, ISBN: 9781608455652.
- Shane Cook, "CUDA Programming: A Developer's Guide to Parallel Computing with GPUs", Morgan Kaufmann, 2013, ISBN: 978-0-12-415933-4.
- Barbara Chapman, Gabriele Jost, Ruud van der Pas, "Using OpenMP Portable Shared Memory Parallel Programming", The MIT Press, 2008, ISBN-13: 978-0-262-53302-7.
- David B. Kirk and Wen-mei W. Hwu, "Programming Massively Parallel Processors", Second Edition, Morgan Kaufmann, 2013, ISBN: 978-0-12-415992-1.