UNIVERSITY OFKERALA

B. TECH. DEGREE COURSE

(2018 SCHEME)

SYLLABUS FOR

VI SEMESTER

COMPUTER SCIENCE & ENGINEERING

SCHEME -2018

VI SEMESTER

COMPUTERSCIENCE&ENGINEERING(R)

Course No	Name of subject	Credits	Weekly load, hours			C A Marks	Exam Duration Hrs	U E Max Marks	Total Marks
			L	T	D/ P				
18.601	Compiler Design (FR)	4	3	1	-	50	3	100	150
18.602	Principles of Programming Languages (R)	3	2	1	-	50	3	100	150
18.603	Design and Analysis of Algorithms (FR)	4	2	1	-	50	3	100	150
18.604	Computer Networks (FR)	3	2	1	-	50	3	100	150
18.605	Graph Theory (R)	3	2	1	-	50	3	100	150
18.606	Signals and Systems (R)	3	2	1	-	50	3	100	150
18.607	Microprocessor & Microcontroller Lab (R)	2	-	-	4	50	3	100	150
18.608	System Software Lab (R)	2	-	-	4	50	3	100	150
	Total	**24**	**13**	**6**	**8**	**400**		**800**	**1200**

18.601 COMPILER DESIGN (FR)

TeachingScheme: 3(L)-1(T)-0(P) **Credits**: 4

Course Objective:

- *To introduce the major concept areas of language translation and compiler design*
- *To develop an awareness of the function and complexity of modern compilers.*
- *To provide practical, hands on experience in compiler design.*

Pre-requisites: ***18.306*** *- Data Structures and Algorithms,*

18.504 *- System Programming*

Module – I

Introduction to compilers and interpreters – Overview of compilation, phases of compiler, Compiler writing tools, Bootstrapping.

Lexical Analysis:-Role of lexical analyzer, Specification tokens using regular expressions. Deterministic and non deterministic finite automata.

Module – II

Syntax Analysis:-Context free grammar-Derivation trees and parse trees, ambiguity. Type checking: Type systems, specification of a simple type checker.

Top-Down parsing:-recursive descent parser, predictive parser, simple LL(1) grammar.

Module – III

Bottom-up parsing: - Shift reduce parsing- operator precedence parsing. LR parsing- Constructing SLR, CLR, LALR parsers.

Syntax directed translation:- Syntax directed definitions, Bottom- up evaluation of S-attributed definitions, L- attributed definitions. Top-down translation, Bottom-up evaluation of inherited attributes.

Module – IV

Intermediate code generation:- Intermediate languages, graphical representations, Three address code, quadruples ,triples, assignment statements, Boolean expressions.

Code optimization:- Principal sources of optimization, Optimization of basic blocks.

Code Generation:- Issues in the design of a code generator. The target machine,

A simple code generator.

References:

1. *Aho A. V., M. S. Lam, R. Sethi and J. D. Ullman, Compilers: Principles, Techniques and Tools,* 2nd Edn., Pearson Education.
2. Keith D Cooper and Linda Torczon, *Engineering a Compiler,* 2nd Edn, Elsevier.
3. Andrew W. Appel, *Modern Compiler Implementation in C,* Cambridge University Press.
4. Kenneth C. Louden, *Compiler Construction: Principles and Practice,* Cengage Learning.
5. Kakde O. G., *Algorithms for Compiler Design,* Cengage Charles River Media.
6. Raghavan V., *Principles of Compiler Design,* TMH.

Internal Continuous Assessment *(Maximum Marks-50)*

50% - Tests (minimum 2)

30% - Assignments (minimum 2) such as home work, problem solving, quiz, literature survey, seminar, term-project, software exercises, etc.

20% - Regularity in the class

University Examination Pattern:

Examination duration: 3 hours *Maximum Total Marks: 100*

The question paper shall consist of 2 parts.

Part A (20 marks) - Ten Short answer questions of 2 marks each. All questions are compulsory. There should be at least one question from each module and not more than three questions from any module.

Part B (80 Marks) - Candidates have to answer one full question out of the two from each module. Each question carries 20 marks.

Course Outcome:

After successful completion of this course, the students will be able to:

- *Identify different language translators and explain the concepts and different phases of compilation with compile time error handling.*
- *Represent language tokens using regular expressions, context free grammar and finite automata and design lexical analyzer for a language.*
- *Compare top down with bottom up parsers, and develop appropriate parser to produce parse tree representation of the input.*
- *Explain syntax directed translation schemes for a given context free grammar and generate intermediate code.*
- *Apply optimization techniques to intermediate code and generate machine code for high level language program.*

18.602 PRINCIPLES OF PROGRAMMING LANGUAGES (R)

Teaching Scheme: 2(L) - 1(T) - 0(P) **Credits:** 3

Course Objectives:

- *To introduce the basic constructs that underlie all programming languages*
- *To introduce the basics of programming language design and implementation*
- *To introduce the organizational framework for learning new programming languages.*

Module I

Names, Scopes and Bindings:- Names and Scopes, Binding Time, Scope Rules, Storage Management, Binding of Referencing Environments.

Control Flow: - Expression Evaluation, Structured and Unstructured Flow, Sequencing, Selection, Iteration, Recursion, Non-determinacy.

Data Types:-Type Systems, Type Checking, Records and Variants, Arrays, Strings, Sets, Pointers and Recursive Types, Lists, Files and Input/Output, Equality Testing and Assignment.

Module II

Subroutines and Control Abstraction: - Static and Dynamic Links, Calling Sequences, Parameter Passing, Generic Subroutines and Modules, Exception Handling, Co-routines.

Functional and Logic Languages:- Lambda Calculus, Overview of Scheme, Strictness and Lazy Evaluation, Streams and Monads, Higher-Order Functions,

Logic Programming in Prolog, Limitations of Logic Programming

Module III

Data Abstraction and Object Orientation:-Encapsulation, Inheritance, Constructors and Destructors, Aliasing, Overloading, Polymorphism, Dynamic Method Binding, Multiple Inheritance.

Innovative features of Scripting Languages:-Scoping rules, String and Pattern Manipulation, Data Types, Object Orientation.

Module IV

Concurrency:- Threads, Synchronization.

Run-time program Management:- Virtual Machines, Late Binding of Machine Code, Reflection, Symbolic Debugging, Performance Analysis.

Text book:

1. Scott M L, Programming Language Pragmatics, 3rd Edn., Morgan Kaufmann Publishers, 2009.

References:

1. David A Watt, Programming Language Design Concepts, Wiley Dreamtech, 2004

2. Ghezzi C and M. Jazayeri, Programming Language Concepts, 3rd Edn, Wiley.1997

3. Kenneth C Louden, Programming Languages: Principles and Practice, 3rd Edn., Cengage Learning, 2011.

4. Pratt T W, M V Zelkowitz, and T. V. Gopal, Programming Languages: Design and Implementation, 4th Edn., Pearson Education, 2001

5. R W Sebesta, Concepts of Programming Languages, 11th Edn., Pearson Education, 2015
6. Ravi Sethi, Programming Languages: Concepts & Constructs, 2nd Edn., Pearson Education, 2006

Internal Continuous Assessment (*Maximum Marks-50*)

50% *- Tests (minimum 2)*
30% *- Assignments (minimum 2) such as home work, problem solving, quiz, literature survey, seminar, term-project, software exercises, etc.*
20% *- Regularity in the class*
University Examination Pattern:

Examination duration: ***3 hours*** *Maximum Total Marks:* ***100***
The question paper shall consist of ***2 parts****.*
Part A (20 marks) *- Ten Short answer questions of 2 marks each. All questions are compulsory. There should be at least one question from each module and not more than three questions from any module.*
Part B (80 Marks) *- Candidates have to answer one full question (question may contain subdivisions), out of the two from each module. Each question carries 20 marks.*

Course Outcome:

The Student will be able to:

- *Compare scope and binding of names in different programming languages*
- *Analyze control flow structures in different programming languages*
- *Appraise data types in different programming languages*
- *Analyze different control abstraction mechanisms*
- *Appraise constructs in functional, logic and scripting languages*
- *Analyze object oriented constructs in different programming languages*
- *Compare different concurrency constructs*
- *Interpret the concepts of run- time program management*

18.603 DESIGN AND ANALYSIS OF ALGORITHMS (FR)

TeachingScheme:2(L)-1(T)-0(P) **Credits:** 4

Course Objectives:

- *Analyze the asymptotic performance of algorithms.*
- *Write rigorous correctness proofs for algorithms.*
- *Demonstrate a familiarity with major algorithms and data structures.*
- *Apply important algorithmic design paradigms and methods of analysis.*
- *Synthesize efficient algorithms in common engineering design situations.*

Pre-requisites: ***18.306****- Data Structures and Algorithms*

Module I

Introduction to algorithm analysis – Time and space complexity, Elementary operations and computation of time complexity- best, average and worst case complexities, Solutions of Recurrence Equations – iteration method and master method- Asymptotic notation , Analysis of sorting algorithms – insertion sorting, Description of quick sort, randomized version of quick sort.

Module – II

Height balanced trees – AVL TREES-Rotations, Red-Black trees – Steps involved in insertion and deletion – rotations, Definition of B-trees – basic operations on B-trees, algorithm for insertion and deletion, Algorithm for sets – Union and Find operations on disjoint sets.

Module – III

Graphs – DFS and BFS traversals, Complexity, Spanning trees – Minimum Cost Spanning Trees, Kruskal's and Prim's algorithms, Shortest paths – single source shortest path algorithms, topological sorting, strongly connected components.

Algorithm Design and analysis Techniques – Divide and Conquer techniques: – Merge Sort, Strassen's matrix multiplication algorithm, analysis.

Module – IV

Dynamic programming: -Optimality principle- Matrix multiplication problem, Bellmanford algorithm, analysis, Comparison of Divide and conquer and Dynamic programming strategy. Greedy algorithms –fractional Knapsack problem, Back tracking – N Queens problem,0/1 Knapsack problem, Branch and Bound – Travelling Salesman problem. Definitions and Basic concepts of NP-completeness and NP-Hardness. Study of NP Complete problems

Text book:

1. Thomas H. Cormen, Charles E. Leiserson and Ronald L. Rivest, Introduction to Algorithms, PHI.
2. Horowitz and Sahni,Fundamentals of Computer Algorithms,GalgotiaPublication.

References:

1. Kenneth A. Merman and Jerome L. Paul, Fundamentals of Sequential and Parallel Algorithms, Vikas PublishingCompany.

Internal Continuous Assessment (*Maximum Marks-50*)

50% *- Tests (minimum 2)*
30% *- Assignments (minimum 2) such as home work, problem solving, quiz, literature survey, seminar, term-project, software exercises, etc.*
20% *- Regularity in the class*

University Examination Pattern:

Examination duration: ***3 hours*** *Maximum Total Marks:* ***100***
The question paper shall consist of ***2 parts****.*
Part A (20 marks) *- Ten Short answer questions of 2 marks each. All questions are compulsory. There should be at least one question from each module and not more than three questions from any module.*
Part B (80 Marks) *- Candidates have to answer one full question (question may contain subdivisions), out of the two from each module. Each question carries 20 marks.*
.

Course Outcome:

After successful completion of this course, the student will be able to:

- *Define asymptotic notations to analyze the performance of algorithms. Apply substitution method, iteration method and master method to analyze recursive algorithms.*
- *Analyze and compare performance of sorting algorithms in terms of time and space complexities.*
- *Discuss various operations of Height-balanced trees and analyze performance of the operations.*
- *Illustrate various applications of graphs such as minimum cost spanning tree, shortest path, topological sorting and strongly connected components, and determine their time and space complexities.*
- *Apply different algorithm design paradigms such as divide-and conquer, dynamic programming and the greedy methods to design efficient algorithms forreal world problems.*
- *Use the concepts of NP-Completeness and NP-Hardness to identify whether a given problem is tractable or not.*

18.604 COMPUTER NETWORKS (FR)

Teaching Scheme: 2(L)-1(T)-0(P) **Credits**: 3

Course Objective:

- *Build an understanding of the fundamental concepts of computer networking.*
- *Familiarize the student with the basic taxonomy and terminology of the computer networking area.*
- *Introduce the student to advanced networking concepts, preparing the student for entry Advanced courses in computer networking.*
- *Allow the student to gain expertise in some specific areas of networking such as the design and maintenance of individual networks.*

Pre-requisites: ***18.404*** *- Data Communication*

Module – I

Introduction – Uses – Network Hardware – LAN –MAN – WAN, Internetworks – Network Software – Protocol hierarchies – Design issues for the layers. Reference models – OSI – TCP/IP. Data Link layer Design Issues – Framing –Error Detection and Correction – Elementary Data Link Protocols – Sliding Window Protocols.

Module – II

MAC Sub layer – IEEE 802 FOR LANs & MANs, IEEE 802.3, 802.4, 802.5. Fast Ethernet - Gigabit Ethernet. Wireless LANs - 802.11 a/b/g/n - Bluetooth.
Network layer – Routing – Shortest path routing, Flooding, Distance Vector Routing, Link State Routing, OSPF, Routing for mobile hosts.

Module – III

Congestion control algorithms – QoS - Techniques. Internetworking – Network layer in internet – IP Protocol - IP Addressing – Classless and Classful Addressing. Subnetting, Internet Control Protocols – ICMP, ARP, RARP, BOOTP, DHCP. Internet Multicasting – IGMP, Exterior Routing Protocols – BGP. IPv6.

Module – IV

Transport Layer – UDP – Header – TCP – Segment Header – Connection Establishment & Release. Application layer –DNS, Electronic mail, MIME, SNMP. Introduction to World Wide Web.

References:

1. Andrew S. Tanenbaum, *Computer Networks*, 4/e, PHI.
2. Behrouz A. Forouzan, *Data Communications and Networking*, 4/e, Tata McGraw Hill.

Internal Continuous Assessment *(Maximum Marks-50)*

50% - Tests (minimum 2)

30% - Assignments (minimum 2) such as home work, problem solving, quiz, literature survey, seminar, term-projectetc.

20% - Regularity in the class

University Examination Pattern:

Examination duration: 3 hours *Maximum Total Marks: 100*

The question paper shall consist of 2 parts.

Part A (20 marks) - Ten Short answer questions of 2 marks each. All questions are compulsory. There should be at least one question from each module and not more than three questions from any module.

Part B (80 Marks) - Candidates have to answer one full question (question may contain sub-divisions), out of the two from each module. Each question carries 20 marks.

Course Outcome:

After the successful completion of the course students will be able to:

- *Describe the different aspects of networks, protocols and network design models.*
- *Explain the various Data Link layer design issues and Data Link protocols*
- *Analyze and compare different LAN protocols*
- *Compare and select appropriate routing algorithms for a network.*
- *Describe the important aspects and functions of network layer, transport layer and application layer in internetworking.*

18.605 GRAPH THEORY (R)

Teaching Scheme: 2(L)-1(T)-0(P) **Credits:** 3

Course Objective:

- *To introduce the major concept areas of graph theory.*
- *To develop an awareness regarding the applications of theorems used in graph theory.*
- *To provide practical, hands on experience in real world applications of graph theory.*

Pre-requisites: ***18.303***-Discrete Structures

Module – I

What is graph – Application of graphs – finite and infinite graphs – Incidence and Degree – Isolated vertex, pendent vertex, Null graph. Paths and circuits – Isomorphism, sub graphs, walks, paths and circuits, Connected graphs, disconnect graphs, Euler graphs Hamiltonian paths and circuits – Travelling salesman problem. Trees – properties, pendent vertex, Distance and centres - Rooted and binary tree, counting trees, spanning trees.

Module – II

Combinatorial versus geometric graphs, Planar graphs, Different representation of planar graphs, geometric dual, combinatorial dual, vector spaces of graph, ban2 vectors of a graph, orthogonal vectors and spaces Directed graphs – types of digraphs, Digraphs and binary relation, Euler graphs, trees with directed edges.

Module – III

Graphs theoretic algorithms and computer programming - Algorithm for computer representation of a graph, algorithm for connectedness and components, spanning tree, directed circuits, shortest path, searching the graphs, Isomorphism.

Module – IV

Graphs in switching and cording theory – contact networks, Analysis of contact Networks, synthesis of contact networks, sequential switching networks, unit cube and its graph, graphs in coding theory.

References:

1. Hararay, *Graph theory*, Narosa Publishers, 1969.
2. Narasingh Deo, *Graph theory*, Pearson publications, 2004.
3. Foulds L. R., *Graphs Theory Applications*, Narosa, Springer-Verlag, 1992.
4. John Clark and Derek Allan Hotton, *A First Look at Graph Theory*, Allied.

Internal Continuous Assessment *(Maximum Marks-50)*

50% - Tests (minimum 2)

30% - Assignments (minimum 2) such as class room/home work, problem solving, quiz,

literature survey, seminar, term-project, software exercises, etc.

20% - Regularity in the class

University Examination Pattern:

Examination duration: 3 hours *Maximum Total Marks: 100*

The question paper shall consist of 2 parts.

Part A (20 marks) - Ten Short answer questions of 2 marks each. All questions are compulsory. There should be at least one question from each module and not more than three questions from any module.

Part B (80 Marks) - Candidates have to answer one full question (question may contain subdivisions), out of the two from each module. Each question carries 20 marks.

Course Outcome:

After the successful completion of the course students will be able to:

- *Demonstrate knowledge of fundamental concepts in graph theory, including properties and characterization of bipartite graphs and trees, Euclidian and Hamiltonian graphs.*
- *Understand and apply some of the classical theorems of graph theory.*
- *Represent real life situations with mathematical graphs.*
- *Develop algorithms for connectedness and components, spanning tree, directed circuits, shortest path, searching the graphs, Isomorphism.*
- *Solve real world problems by applying graph theoretic results and algorithms.*

18.606 SIGNALS AND SYSTEMS (R)

Teaching Scheme: 2(L)-1(T)-0(P) **Credits**: 3

Course Objective:

- *Coverage of continuous and discrete-time signals and systems, their properties and representations and methods that are necessary for the analysis of continuous and discrete-time signals and systems.*
- *Knowledge of frequency-domain representation and analysis concepts using Fourier Analysis tools, Z-transform*
- *Knowledge of digital filters both FIR and IIR.*

Module – I

Signals and systems – introduction – basic operations on signals – continuous time and discrete time signals –step, impulse, ramp, exponential and sinusoidal functions. Continuous time and discrete time systems –properties of systems – linearity, causality, time invariance, memory, stability, invertibility. Linear time invariant systems – convolution.

Module – II

Z-transform – region of convergence – properties of Z- transform – inverse Z-transform. Fourier transform (FT) of discrete time signals – properties of FT – relation between Z- transform and FT.

Module – III

Discrete Fourier transform (DFT) - Properties of DFT – inverse DFT - Fast Fourier transform (FFT) – Radix-2 FFT algorithms – butterfly structure.

Module – IV

Digital filter structures – block diagram and signal flow graph representation – structures for IIR – direct form structure – Cascade form structure – parallel form structure – lattice structure.

Structures for FIR – direct form structures – direct form structure of linear phase system – cascade form structure – frequency sampling structure – lattice structure**.**

References:-

1. Bandyopadhyaya M. N., *Introduction to Signals and Systems and Digital Signal Processing,* PHI.
2. Li Tan, *Digital Signal Processing, Fundamentals and Applications,* Elsevier.
3. Oppenheim A. V. and R. W. Schafer, *Digital Signal Processing,* Prentice-Hall Inc.
4. Proakis J.K. and D. G. Manolakis, *Introduction to Digital Signal Processing,* MacMillan.
5. Hayes M. H., *Digital Signal Processing,* Tata McGraw Hill (SCHAUM's Outlines).
6. Apte S. D., *Digital Signal Processing,* Wiley India.

Internal Continuous Assessment *(Maximum Marks-50)*

50% - Tests (minimum 2)

30% - Assignments (minimum 2) such as class room/home work, problem solving, quiz, literature survey, seminar, term-project, software exercises, etc.

20% - Regularity in the class

University Examination Pattern:

Examination duration: 3 hours *Maximum Total Marks: 100*

The question paper shall consist of 2 parts.

Part A (20 marks) - Ten Short answer questions of 2 marks each. All questions are compulsory. There should be at least one question from each module and not more than three questions from any module.

Part B (80 Marks) - Candidates have to answer one full question (question may contain sub-divisions), out of the two from each module. Each question carries 20 marks.

Course Outcome:

After successful completion of this course, students will be able to

- *Apply time and frequency domain analysis techniques to different types of signals and systems*
- *Classify Signals and systems as discrete/continuous, linear/non-linear, causal/non- causal, time variant/invariant etc*

18.607 MICROPROCESSOR & MICROCONTROLLER LAB (R)

TeachingScheme: 0(L)-0(T)-4(P) **Credits**: 2

Course Objective :

- *To design assembly language programs for solving problems*
- *To understand organization of interfacing devices for various peripheral devices and programming them*

Pre-requisites: ***18.505*** *Microprocessors & Microcontrollers*

List of Exercises:

I. Exercises/Experiments using MASM(PC required)

1. Study of Assembler and Debugging commands.
2. Implementation of decimal arithmetic (16 & 32 bit) operations.
3. Implementation of String manipulations.
4. Implementation of searching and sorting of 16 bit numbers.
5. Implementation of matrix operations like addition, transpose, multiplication etc.

II. Exercises/Experiments using 8051 trainer kit

6. Familiarization of the components / Cards inside a computer, standard connectors, cords, different ports, various computer peripherals. NIC and other I/O cards, and their uses.
7. Assembling of PC from Components
8. Familiarization of 8051 trainer kit by executing simple Assembly Language programs,
    - Multi byte addition
    - Multiplication
    - Array operations
    - Matrix operations
    - Code conversion etc.
9. Implementation of stepper motor interfacing , ADC/DAC interfacing & sensor interfacing.

Internal Continuous Assessment *(Maximum Marks-50)*

40% - Test

40% - Class work and Record (Up-to-date lab work, problem solving capability, keeping track of rough record and fair record, term project, etc.)

20% - Regularity in the class

University Examination Pattern:

Examination duration: 3 hours *Maximum Total Marks: 100*

Questions based on the list of exercises prescribed.

Marks should be awarded as follows:

20% - Algorithm/Design

30% - Implementing / Conducting the work assigned

25% - Output/Results and inference

25% - Viva voce

Candidate shall submit the certified fair record for endorsement by the external examiner.

Course Outcome:

After successful completion of this course, students will be able to:

- *Develop assembly language programs for problem solving*
- *Implement assembly language program to interface various I/O devices.*

18.608 SYSTEM SOFTWARES LAB(R)

Teaching Scheme: 0(L)-0(T)-4(P) **Credits**: 2

Course Objective :

- *To design and implement assembler for a hypothetical machine.*
- *To design Macroprocessor.*
- *To get an exposure to design and implement various components of system software*

Pre-requisites: ***18.504*** *Systems Programming,*
18.503 *Operating System*

List of Exercises:

PART-A

1. Simulate the following non-preemptive CPU scheduling algorithms to find turnaround time and waiting time.
   a) FCFS b) SJF c) Round Robin (pre-emptive) d) Priority
2. Simulate the following file allocation strategies.
   a) Sequential b) Indexed c)linked
3. Implement the different paging techniques of memory management.
4. Simulate the following file organization techniques *
   a) Single level directory b) Two level directory c) Hierarchical
5. Implement the banker's algorithm for deadlock avoidance.*
6. Simulate the following disk scheduling algorithms. *
   a) FCFS b)SCAN c) C-SCAN
7. Simulate the following page replacement algorithms
   a) FIFO b)LRU c) LFU
8. Implement the producer-consumer problem using semaphores. *
9. Write a program to simulate the working of the dining philosopher's problem.*

PART-B

10. Implement the symbol table functions: create, insert, modify, search, and display.
11. Implement pass one of a two pass assembler. *
12. Implement pass two of a two pass assembler. *
13. Implement a single pass assembler. *

14. Implement a two pass macro processor *
15. Implement a single pass macro processor.
16. Implement an absolute loader.
17. Implement a relocating loader.
18. Implement pass one of a direct-linking loader.
19. Implement pass two of a direct-linking loader.
20. Implement a simple text editor with features like insertion / deletion of a character, word, and sentence.
21. Implement a symbol table with suitable hashing.*

Internal Continuous Assessment *(Maximum Marks-50)*

40% - Test

40% - Class work and Record (Up-to-date lab work, problem solving capability, keeping track of rough record and fair record, term projects etc.)

20% - Regularity in the class

University Examination Pattern:

Examination duration: 3 hours *Maximum Total Marks: 100*

Marks should be awarded as follows:

20% - Algorithm/Design

30% - Implementing / Conducting the work assigned

25% - Output/Results and inference

25% - Viva voce

Candidate shall submit the certified fair record for endorsement by the external examiner.

Course Outcome:

After successful completion of this course, students will be able to:

- *Understand latest features of translators.*
- *Apply the concept of finite automata to implement components of system software.*
- *Design system software using latest tools.*