Learning Outcomes based Curriculum

Department of Computer Science and Engineering

Tezpur University

Bachelor of Technology in Computer Science and Engineering

Preamble

The Bachelor in Technology in Computer Science and Engineering (BTech(CSE)) programme is a 4 year, 8 semester, 161 credit programme conforming to the model prescribed by the AICTE. Students having completed 10+2 level with Mathematics, Physics, Chemistry and English, are admitted to the programme mainly through the Joint Entrance Examination (JEE).

1. Introduction

The BTech (CSE) programme of Tezpur University was started in 2006 under the School of Engineering in view of the requirement of quality engineering graduates in the field of computer science and engineering. The large-scale adoption of ICT in different spheres calls for human resource skilled in the domains of computing, and design and development of computer based systems and applications. There is also need for graduates who would proceed to acquire higher education and contribute in academia and research. CSE graduates should also take up leadership positions in organisations and the society. The curriculum of the BTech(CSE) programme has been developed keeping these in mind and according to the model prescribed by the AICTE. Courses in the programme are from the domain of CSE, as well as from science, humanities, management and other engineering disciplines. Interdisciplinary courses are there both at the foundation level and advanced levels. Project works in the curriculum in the final semesters provide an opportunity to the students to apply the knowledge acquired in creating solutions for real problems. Similarly, an industrial internship after three years prepares the students to acquire a real world perspective of the professional environment they would face upon graduation.

2. Qualification descriptors for the graduates

Knowledge & Understanding:

1. Graduates have sound knowledge of use of computing in information processing to solve different problems.
2. Graduates have understanding of the state-of-the-art and the challenges in their field.
3. Graduates have understanding of other related disciplines, such as humanities and management, which are essential for professional practice.

Skills & Techniques:

1. Graduates have skills of software development for different purposes, and overall computing system development.
2. Graduates are familiar with tools and techniques available for different computing purposes and are capable to learn and develop new techniques as and when required.
3. Graduates have the right communication skills required for success in their profession.

Competence:

1. Graduates are aware of roles and responsibilities they would assume in their professional career.
2. Graduates are ready to work individually as well as in teams, in industry, academia, research, and entrepreneurship.
3. Graduates are ready for lifelong learning to stay relevant as the field of CSE advances.

3. Graduates Attributes

1. Graduates have a solid foundation in engineering, basic sciences and mathematics for successful professional careers in industry, academia, and public service.

2. Graduates have skills for effective technical communication/collaboration in multidisciplinary teams providing technical leadership to comprehend, analyze, design, and create innovative computing solutions for challenging real life problems.
3. Graduates will pursue lifelong learning through such activities as higher studies, research & development, distance education, professional training and membership in professional societies to be able to adapt to challenges of changing environment.

4. Program Outcomes

1. Graduate has gained thorough understanding of the CSE as reflected in the curriculum.
2. Graduate has gained skills to apply knowledge in the development of useful ICT products/processes/ applications/models for the society at large aligned to a developing economy and the environment.
3. Graduate has gained the ability to critically examine a real world problem, identify the role of computing in a solution for the problem, solve it, and be aware of the scope, challenges and impact of the solution.
4. Graduate has acquired some subject domain specific competence through elective and open-elective courses and project works.

1. Programme structure

Total Credits: 161

Structure of the curriculum:

Course Category	**Credits**
Humanities, Social Science and Management Courses (HSMC)	12
Basic Science Courses (BSC)	27
Engineering Science Courses (ESC)	26
Professional Core Courses (PCC)	53
Professional Elective Courses (PEC)	17
Open Elective Courses (OEC)	12
Projects	14
Total-	**161**

Basic Science Courses:

Course Code	Course Name	Credit				CH
		L	T	P	Total	
MS104	Mathematics-I	3	1	0	4	4
MS105	Mathematics-II	3	1	0	4	4
MS205	Mathematics-III	3	0	0	3	3
CO105	Discrete Maths	3	1	0	4	4
PH103	Physics-I	2	0	1	3	4
PH104	Physics-II	2	0	0	2	2
CH103	Chemistry	3	0	1	4	5
BT201	Biology	3	0	0	3	3
	Total-	22	3	2	27	29

HSS and Management Courses:

Course Code	Course Name	Credit				CH
		L	T	P	Total	
EF103	English	2	0	1	3	4
BA201	Economics	3	0	0	3	3
IC361	Accounting and Financial Management	3	0	0	3	3
	HMS Elective-I	3	0	0	3	3
	Total-	11	0	2	12	13

HSS and Management Elective Courses:

Course Code	Course Name	Credit				CH
		L	T	P	Total	
EG4XX	Introduction to Academic Reporting in English	2	0	1	3	4
BA4XX	Fundamentals of Management	3	0	0	3	3
BA4XX	Social Responsibility and Professional Ethics in Engineering	3	0	0	3	3

Engineering Science Courses:

Course	Course Name	Credit	CH

Code						
CE103	Engineering Graphics and Design	1	0	2	3	5
EE103	Basic Electrical Engineering	3	0	0	3	3
EE104	Basic Electrical Engineering Lab	0	0	1	1	2
EC102	Basic Electronics	2	1	1	4	5
ME103	Workshop Practice	0	0	2	2	4
CO103	Introductory Computing	2	1	0	3	3
CO104	Computing Laboratory	0	0	2	2	4
EC205	Signals and Systems	2	1	0	3	3
CO218	Data Communication	3	0	0	3	3
CO209	Computing Workshop (SciLab/MatLab/R & Simulator Lab)	0	0	2	2	4
	Total-				26	36

Professional Core Courses:

Course Code	Course Name	Credit				CH
		L	T	P	Total	
CO202	Digital Logic Design	3	0	1	4	5
CO210	Data Structures	3	1	0	4	4
CO216	Formal Languages and Automata	3	0	0	3	3
CO214	Computer Architecture and Organization	3	1	0	4	4
CO215	Computer Architecture and Organization lab	0	0	1	1	2
CO206	Design and Analysis of Algorithm	3	0	1	4	5
CO211	Data Structures using Object Oriented Programming Lab	0	1	2	3	5
CO309	Operating Systems	3	0	0	3	3
CO312	Database Systems	3	0	0	3	3
CO315	Computer Networks	3	0	0	3	3
CO310	Operating Systems Lab	0	0	1	1	2
CO316	Computer Network Lab	0	0	1	1	2
CO313	Database Systems Lab	0	1	1	2	3

CO314	System software and Compiler Design	3	0	1	4	5
CO311	Software Engineering	3	0	0	3	3
CO401	Artificial Intelligence	3	0	0	3	3
CO217	Graph Theory	3	0	0	3	3
CO303	Computer Graphics	3	0	1	4	5
	Total-			10	53	63

2. SEMESTER-WISE SCHEDULE

Semester-I

S. No.	Course Code	Course Name	L	T	P	CR	CH
1	CH103	Chemistry	3	0	1	4	5
2	PH103	Physics-I	2	0	1	3	4
3	MS104	Mathematics-I	3	1	0	4	4
4	EE103	Basic Electrical Engineering	3	0	0	3	3
5	EE104	Basic Electrical Engineering Lab	0	0	1	1	2
6	EF103	Communicative English	2	0	1	3	4
7	SE100	Induction Program	-	-	-	-	8
		Total	13	1	4	18	22

Semester-II

S. No.	Course Code	Course Name	L	T	P	CR	CH
1	PH104	Physics-II	2	0	0	2	2
2	MS105	Mathematics-II	3	1	0	4	4
3	CO105	Discrete Mathematics	3	1	0	4	4
4	EC102	Basic Electronics	2	1	1	4	5
5	ME103	Workshop Practice	0	0	2	2	4
6	CO103	Introductory Computing	2	1	0	3	3

7	CO104	Computing Lab	0	0	2	2	4
8	CE103	Engineering Graphics	1	0	2	3	5
		Total	13	4	7	24	31

Semester-III

S. No.	Course Code	Course Name	L	T	P	CR	CH
1	MS205	Mathematics – III	3	0	0	3	3
2	CO202	Digital Logic Design	3	0	1	4	5
3	CO209	Computing Workshop	0	0	2	2	4
4	BA201	Economics	3	0	0	3	3
5	CO210	Data Structures	3	1	0	4	4
6	CO211	Data structures using Object Oriented Programming Lab	0	1	2	3	5
7	EC205	Signals and Systems	2	1	0	3	3
8	ES201	Environmental Science	2	0	1	0	3
		Total	14	3	5	22	30

Semester-IV

S. No.	Course Code	Course Name	L	T	P	CR	CH
1	BT201	Biology	3	0	0	3	3
2	CO218	Data Communication	3	0	0	3	3
3	CO214	Computer Architecture and Organization	3	1	0	4	4
4	CO215	Computer Organization Lab	0	0	1	1	2
5	CO216	Formal Language and Automata	3	0	0	3	3
6	CO206	Design and Analysis of Algorithms	3	0	1	4	5
7	CO217	Graph Theory	3	0	0	3	3
		Total	18	1	2	21	23

Semester-V

S. No.	Course Code	Course Name	L	T	P	CR	CH
1	CO309	Operating Systems	3	0	0	3	3
2	CO311	Software Engineering	3	0	0	3	3
3	CO310	Operating Systems Lab	0	0	1	1	2
4	CO312	Database Systems	3	0	0	3	3
5	CO313	Data base Systems Lab	0	1	1	2	3
6	CO303	Computer Graphics	3	0	1	4	5
7		Elective-I (PEC01)	3	0	0	3	3
8		Open Elective-I (OEC01)	3	0	0	3	3
9	LW301	Indian Constitution (MC- Non Credit)	1	0	0	0	1
		Total	18	1	3	22	25

Semester-VI

S.No	Course Code	Course Name	L	T	P	CR	CH
1	CO314	System Software and Compiler Design	3	0	1	4	5
2		Elective-II (PEC02)	3	0	1	4	5
3		Open Elective-II (OEC02)	3	0	0	3	3
5	CO315	Computer Networks	3	0	0	3	3
6	CO316	Computer Networks Lab	0	0	1	1	2
7	IC361	Accounting and Financial Management	3	0	0	3	3
8	CO317	Project-I (using SE perspective)	0	0	2	2	4
		Total	15	0	5	20	25

Semester-VII

S. No.	Course Code	Course Name	L	T	P	CR	CH
1	XXxxx	* HSS/Management Elective	3	0	0	3	3
2	CO401	Artificial Intelligence	3	0	0	3	3
3		Elective-III (PEC03)	3	0	1	4	5

4		Elective-IV (PEC04)	3	0	0	3	3
5		Open Elective-III (OEC03)	3	0	0	3	3
6	CO402	Project-II	0	0	4	4	8
7	CT430	Essence of Indian Traditional Knowledge (MC- Non Credit)	1	0	0	0	1
		Total-	15	0	6	20	26

Semester-VIII

S.No.	Course Code	Course Name	L	T	P	CR	CH
1		Elective-V (PEC05)	3	0	0	3	3
2		Open Elective-IV (OEC04)	3	0	0	3	3
3	CO403	Project-III	0	0	8	8	16
		Total	6	0	10	14	22

Mandatory Non-credit Courses:

S. No.	Course Code	Course Name	Schedule
1	SE100	Induction Programme	Semester I
2	CO404	Summer Internship /Industrial Training	After Semester VI
3	CO405	Comprehensive Written Exam	Semester VII/Semester VIII
4	ES201	Environmental Science	Semester III
5	LW301	Indian Constitution	Semester V
6	CT465	Indian Traditional Knowledge	Semester VI

Semester-wise Break-up:

Semester	Category-wise Credit Distribution							Total Credits
	HSMC	BSC	ESC	PCC	PEC	OEC	PCE	
I	3	11	4	0	0	0	0	18
II	0	10	14	0	0	0	0	24
III	3	3	5	11	0	0	0	22

IV	0	3	3	15	0	0	0	21
V	0	0	0	16	3	3	0	22
VI	3	0	0	8	4	3	2	20
VII	3	0	0	3	7	3	4	20
VIII	0	0	0	0	3	3	8	14
Total	**12**	**27**	**26**	**53**	**17**	**12**	**14**	**161**

Professional Elective Courses (PEC):

PEC01, PEC02, PEC03, PEC04, PEC05 will be offered from the following list of Elective courses from the Department. Some of the elective courses are already running in the B.Tech. programme and some of them are added newly.

Course Code	Course Name	L-T-P	Credit
CO304	Principles of Programming Languages	3-0-0	3
CO318	Cryptography	3-0-0	3
CO432	Information Theory and Coding	3-0-0	3
CO319	Statistical Modelling and Applications	3-0-0	3
CO423	Web Technology	3-0-1	4
CO306	Embedded Systems	3-0-1	4
CO426	Advanced Computer Architecture	3-0-1	4
CO422	Theory of Computation	3-1-0	4
CO406	Distributed Systems	3-0-1	4
CO509	Computer Vision & Image Processing	3-0-1	4
CO512	Parallel Programming	3-0-1	4
CO513	Fundamentals of Speech Processing	3-0-1	4
CO514	Machine Learning	3-0-1	4
CO515	Knowledge Representation and Reasoning	4-0-0	4
CO516	Advanced Algorithms	4-0-0	4
CO517	Virtual and Augmented Reality	3-0-1	4
CO518	Cloud Computing	3-0-0	3

CO504	Natural Language Processing	3-0-0	3
CO519	Internet of Things	3-0-0	3
CO520	Software Defined Networking and Network Function Virtualization	3-0-0	3
CO521	Computational Geometry	3-0-0	3
CO522	Bioinformatics	3-0-0	3
CO523	Quantum Computing	3-0-0	3
CO524	Linear Optimization	3-0-0	3
CO505	Advanced Database Management System	3-0-0	3
CO503	Fuzzy Logic and Neural Networks	3-0-0	3
CO435	Mobile Computing	3-0-0	3
CO525	Data Mining	3-0-0	3
CO526	Operation Research	3-0-0	3

3. Mapping of course with program outcomes (POs)

Course code	Course title	PO1	PO2	PO3	PO4
CH103	Chemistry	40%	30%	30%	-
PH103	Physics-I	40%	30%	30%	-
MS104	Mathematics-I	40%	30%	30%	-
EE103	Basic Electrical Engineering	40%	30%	30%	-
EE104	Basic Electrical Engineering Lab	40%	30%	30%	-
EF103	Communicative English	40%	30%	30%	-
SE100	Induction Program	40%	30%	30%	-
PH104	Physics-II	40%	30%	30%	-
MS105	Mathematics-II	40%	30%	30%	-

CO105	Discrete Mathematics	50%	-	25%	25%
EC102	Basic Electronics	40%	30%	30%	
ME103	Workshop Practice	40%	30%	30%	
CO103	Introductory Computing	20%	40%	40%	-
CO104	Computing Lab	20%	40%	40%	-
CE103	Engineering Graphics	40%	30%	30%	
MS205	Mathematics – III	40%	30%	30%	
CO202	Digital Logic Design	40%	20%	40%	-
CO209	Computing Workshop	30%	30%	40%	-
BA201	Economics	40%	30%	30%	
CO210	Data Structures	40%	20%	40%	-
CO211	Data structures using Object Oriented Programming Lab	40%	30%	30%	
EC205	Signals and Systems	40%	30%	30%	
ES201	Environmental Science	40%	30%	30%	
BT201	Biology	40%	30%	30%	
CO218	Data Communication	40%	30%	30%	-
CO214	Computer Architecture and Organization	60%	20%	20%	-
CO215	Computer Organization Lab	60%	20%	20%	-
CO216	Formal Language and Automata	60%	20%	20%	-
CO206	Design and Analysis of Algorithms	60%	20%	20%	-
CO217	Graph Theory	60%	20%	20%	-
CO309	Operating Systems	50%	30%	20%	-
CO311	Software Engineering	20%	40%	40%	-
CO310	Operating Systems Lab	50%	30%	20%	-

CO312	Database Systems	30%	40%	30%	-
CO313	Data base Systems Lab	30%	40%	30%	-
CO303	Computer Graphics	40%	40%	20%	-
	Elective-I (PEC01)	40%	20%	20%	20%
	Open Elective-I (OEC01)				
LW301	Indian Constitution (MC- Non Credit)				
CO314	System Software and Compiler Design	60%	20%	20%	-
	Elective-II (PEC02)	40%	20%	20%	20%
	Open Elective-II (OEC02)				
CO315	Computer Networks	40%	30%	30%	-
CO316	Computer Networks Lab	40%	30%	30%	-
IC361	Accounting and Financial Management				
CO317	Project-I (using SE perspective)	20%	40%	40%	-
CO314	System Software and Compiler Design				
XXxxx	* HSS/Management Elective				
CO401	Artificial Intelligence	40%	30%	30%	-
	Elective-III (PEC03)	40%	20%	20%	20%
	Elective-IV (PEC04)	30%	25%	25%	20%
	Open Elective-III (OEC03)				
CO402	Project-II	25%	25%	25%	25%
CT430	Essence of Indian Traditional Knowledge (MC-Non Credit)				
	Elective-V (PEC05)	40%	25%	25%	10%
	Open Elective-IV (OEC04)				
CO403	Project-III	25%	25%	25%	25%

4. **Evaluation plan:**

There shall be minimum two Sessional Tests and two Examinations for each Theory Course, and two Examinations for a Practical Course having L-T-P structure. Details as follows:

Evaluation plan for Theory Courses:

Sessional Test/ Examination		Course Credit ≤ 2		Course Credit ≥ 3		Semester period
Nomenclature	Type	Marks	Duration	Marks	Duration	
Sessional Test-I	Written	20	30 min	25	45 min	Within 5th week
Mid-Semester Examination	Written	30	90 min	40	2 hours	Within 10th week
Sessional Test-II	Written/Assignment/Seminar etc.	20	XX	25	XX	Within 14th week
End-Semester Examination	Written	50	2 hours	60	3 hours	Within 18th week

Evaluation plan for Practical Courses:

Examination		L-T-P Structure-wise Marks		Semester period
		L-T-P: 0-0-z	L-T-P: x-y-z	
Nomenclature	Type	Marks	Marks	
Mid-Semester (Practical) Examination	Viva, Report	30	-	Before Mid Semester
End-Semester (Practical) Examination	Practical Examination, Viva, Report	70	50	Before End Semester

5. DETAILED SYLLABUS

Introductory Computing **CO103**

2-1-0: 3 Credits: 3 Hours *Prerequisites:* None

Course Outcomes:

1. Understand computing environment, how computers work and the strengths and limitations of computers.
2. Identify and understand the representation of numbers, alphabets and other characters in computer system.
3. Understand, analyse, and implement software development tools like algorithm, pseudo codes and programming structure.
4. Write small programs related to simple/moderate mathematical and logical problems in 'C'.

Course Contents:

Computer Fundamentals: - History, Generations, Classification of Computers; - Organization of a Computer; - Concept of Programming and Programming Languages. Introduction to Programming: - Concept of Algorithm, Flow Chart, Pseudocode, Illustrative Problem Solving Examples. - Features of a Programming Language: Character Set, Identifiers, Keywords, Data Types, Variables, Declarations, Operators & Expressions; Statements: Assignment, Input/Output; Flow Control- Conditionals and Branching; Iteration; Functions, Function Types, Scope Rule; Recursion; Arrays, Pointers, Structures. (A programming language like C/C++ shall be used as a basis language. The same language is to be used for the laboratory).

Text Books:

1. Programming in C, Balaguruswamy.
2. Let us C, Kanetkar Y.
3. Programming in C, Gotfreid, McGrawHill
4. Fundamentals of Computers, Rajaram, V.

Reference Books:

5. The Elements of Programming Style, Kerningham, B. W.
6. Techniques of Program Structures and Design, Yourdon, E.
7. Theory and Problems of Computers and Programming, Schied, F. S.
8. The C Programming Language, Kerningham & Ritchie.

Computing Laboratory **CO104**

0-0-2: 2 Credits: 4 Hours *Prerequisites:* None

Course Outcomes:

Towards the end of the course the student would:

1. develop knowledge on computations such as algorithm, flowcharts etc.
2. become aware of the existence of various other primitive programming languages that are used for computation.
3. do beginners and average level of programming problems such as factorial, recursion, problems involving use of structures etc.
4. implement some basic data structures using the C language. Some data structures that they can implement using C are Array, Stack, Queue, Linked List etc.

Course Contents:

Laboratory exercises shall involve the following:

1. Familiarization of a computer and the environment and execution of sample programs
2. Expression evaluation
3. Conditionals and branching
4. Iteration
5. Functions
6. Recursion
7. Arrays
8. Structures
9. Linked lists
10. Data structures

It is suggested that some problems related to continuous domain problems in engineering and their numerical solutions are given as laboratory assignments. It may be noted that some of basic numerical methods are taught in the Mathematics course.

Text Books:

1. The Elements of Programming Style, Kerningham, B. W.
2. The C Programming Language, Kerningham & Ritchie.

Reference Books:

3. Programming in C, Balaguruswamy.
4. Let us C, Kanetkar Y.
5. Programming in C, Gotfreid, McGrawHill

Computing Workshop **CO209**

0-0-2: 2 Credits: 4 Hours *Prerequisites:* None

Abstract:

This course aims to familiarize students with the basic concepts of two programming environments, MATLAB and Python, and their design tradeoffs. The course covers basic programming aspects, and discusses the implementation of MATLAB and Python for different numerical methods. It also summarizes how both these environments provide a wide platform for problem solving and encourages students to explore this course for their upcoming project assignments. A group project has to be taken up by the students during this course.

Course Outcome:

Towards the end of the course the student would:

1. be familiar with basic programming concepts of MATLAB and Python.
2. have a clear understanding of the syntax which will allow him/her to correctly model a problem.
3. be able to design and simulate any real time environment using MATLAB or Python.

Course Contents:

Introduction to MATLAB: MATLAB interface; variables; keywords; commands; operators: arithmetic, relational, logical, bitwise.

Vectors and Matrices in MATLAB: Vectors and matrices: creation, deletion, access, manipulation; Special matrices; complex matrix; matrix commands; matrix operations: determinant, minor, inverse, rank, eigen value and vectors.

MATLAB Scripts: M-files; Function files: primary function, sub-function; ways of creating script files; input/output functions.

Conditional statements in MATLAB: Statements: IF, IF-ELSE, nested IF-ELSE, SWITCH case; IS-function.

Iteration and Loops: Loops: FOR loop, WHILE loop, Nested loops; control statements: break, continue; Vectorizing.

Cell arrays: Creation, deletion, access, manipulation and operations in cell arrays.

Numerical methods using MATLAB: Set operations; Solving of linear equations; Non-linear equations; differentiation and integration.

Plotting in MATLAB: Visualizing results using plot; subplot; histogram; bar graph; pie chart etc.

Introduction to Python: Python overview; Interactive mode and Script mode; variables; keywords; datatypes: numeric, dictionary, Boolean, set, list, tuple, string; creation, deletion, access in different datatypes; operators: arithmetic, relational, assignment, logical, bitwise, membership, identity; input/output functions.

Conditional statements in Python: Statements: IF, IF-ELSE, nested IF-ELSE.

Iteration and Loops: Loops: FOR loop, WHILE loop, Nested loops; control statements: break, continue, pass.

Functions in Python: arguments: required, keyword, default, variable-length; creating function; return statements.

Matrix in Python: Numpy module for matrix in Python; creation, deletion, access, manipulation; types of matrices; matrix operations: determinant, minor, inverse, rank, eigen value and vectors; solving linear equations.

Plotting in Python: Matplotlib module for plotting in Python: plot, bar graph, pie chart, histogram, scatter plot, contour plot etc.

Textbooks:

1. Steven I. Gordon and Brian Guilfoos, Introduction to Modeling and Simulation with MATLAB and Python, Chapman & Hall, CRC Press, Computational Science Series, 2017.
2. Stormy Attaway, MATLAB: A Practical Introduction to Programming and Problem Solving, College of Engineering, Bosten University, Elsevier, 2009.

References:

1. Mathworks, MATLAB: The Language of Technical Programming.
2. M. C. Brown, Python: The Complete Reference, Mc Graw Hills, 4th Edition, 2018.
3. A. Gilat, MATLAB: An Introduction with Applications, Wiley, 4th Edition, 2012

Discrete Mathematics **CO105**

3-1-0: 4 Credits: 4 Hours *Prerequisites:* None

Course Outcomes:

Towards the end of the course the student would:

1. gain a basic understanding of the main topics of discrete mathematics,
2. develop insight into the mathematics of these structures,
3. be exposed to the applications of these structures in engineering,
4. be able to solve mathematical problems in these topics.

Course Contents:

History and overview: Reasons for studying discrete structures, Some people who influenced or contributed to the area of discrete structures.

Sets, relations, and functions: Basic operations on sets, cartesian products, disjoint union (sum), and power sets. Different types of relations, their compositions and inverses. Different types of functions, their compositions and inverses.

Propositional Logic: Syntax and semantics, proof systems, satisfiability, validity, soundness, completeness, deduction theorem, etc. Decision problems of propositional logic. Introduction to first order logic and first order theory.

Partially ordered sets: Complete partial ordering, chain, lattice. Complete, distributive, modular, and complemented lattices. Boolean and pseudo Boolean lattices.

Algebraic Structures: Algebraic structures with one binary operation – semigroup, monoid and group. Cosets, Lagrange's theorem, normal subgroup, homomorphic subgroup. Congruence relation and quotient structures. Error correcting code. Algebraic structures with two binary operations- ring, integral domain, and field. Boolean algebra and Boolean ring. (Definitions and simple examples only).

Introduction to Counting: Basic counting techniques – inclusion and exclusion, pigeon-hole principle, permutation, combination, summations. Introduction to recurrence relation and generating function.

Introduction to Graph: Graphs and their basic properties – degree, path, cycle, subgraph, isomorphism, Eulerian and Hamiltonian walk, trees.

Text Books:

1. Discrete Mathematical Structures, Trembly and Manohar, McGrawHill.
2. Introduction to Discrete Mathematics, C. L. Liu.

Digital Logic Design CO202

3 - 0 - 1: 4 Credits: 5 Hours *Prerequsites:* None

Abstract:
This course is designed to provide knowledge and understanding to sophomores in electrical and computer engineering of Boolean algebra and digital concepts, with concentration on the analysis and design of combinational and sequential logic networks. Furthermore, it provides a foundation for subsequent study in computer organization, architecture, and VLSI design.

Course outcome: Towards the end of the course the student would

1. Have a thorough understanding of the fundamental concepts and techniques used in digital electronics
2. Be able to design, analyze and synthesize logic circuits (combinational & sequential)
3. Be able to design complex digital systems
4. Be able to identify and prevent various hazards and timing problems in a digital system

Course Contents:

History & overview : Reasons for studying digital logic, people who influenced/contributed to the area of digital logic, applications of Digital Logic and introduction to a digital system.

Switching theory: Number systems and codes, Binary arithmetic, Complements, Boolean and switching algebra, Representation and manipulation of switching functions, Minimization of switching functions using algebraic method, K-map(2-,3-,4-,5-variable), Quine McCluskey method.

Combinational logic circuits: Basic logic gates (AND,OR,NOT,NAND,NOR,XOR), Realization of switching functions with networks of logic gates, 2-level networks: AND-OR,OR-AND,NAND-NAND, NOR-NOR, Multi-level networks, Physical properties of logic gates (technology, fan-in, fan-out, propagation delay), Elimination of timing hazards/glitches.

Modular design of combinational circuits: Design of medium scale combinational logic modules - Multiplexers, demultiplexers, decoders, encoders, comparators, Arithmetic functions (adders, subtracter, carry look ahead), Multipliers, dividers, Arithmetic and logic units (ALUs), Hierarchical design of combinational circuits using logic modules.

Memory elements: Unclocked and clocked memory devices (latches, flip flops), Level vs. edge-sensitive, and master-slave devices, Basic flip flops (SR, D, JK, T), Asynchronous flip flop inputs (preset, clear), Timing constraints (setup time, hold time) and propagation delays, Data registers (selection, clocking, timing), Random-access memory (RAM).

Sequential logic circuits : Finite state machines (FSMs), clocked and unclocked, Mealy vs. Moore models of FSMs, Modelling FSM behaviour: State diagrams and state tables, timing diagrams, algorithmic state machine charts, Analysis of synchronous and asynchronous circuits, Design of synchronous sequential circuits: State minimization, state assignment, next state and output equation realization, Sequential functional units: Data registers, shift registers, counters, sequence detectors, synchronizers, debouncers, controllers.

Fault detection and Location: Fault models for combinational and sequential circuits, Fault detection in combinational circuits; Homing experiments, Distinguishing experiments, machine identification and fault detection experiments in sequential circuits.

Laboratory component:

Study of TTL gate characteristics, Open collector and Tri-state gates, Clock generator and timer circuit.
Synthesis of combinational circuits using NAND, NOR and Multiplexers, Decoder and driver circuits for 7- segment LED displays, D/A converter and 4-bit ALU realization.
Synthesis of sequential circuits – study of various types of flip-flops, realization of counters, shift registers and sequence generators.
ASM chart based synthesis such as, Traffic light controller, Blackjack dealer and dice game ASM synthesis, etc.

Text Books:

1. M.M.Mano : Digital Logic and Computer Design, PHI (EEE)
2. Floyd and Jain : Digital Fundamentals, Pearson Education
3. Switching and Finite Automata Theory, Z. Kohavi, TMH.

Reference Books:

1. R.P.Jain : Modern Digital Electronics, Tata McGraw-Hill Education
2. J.F.Wakerly : Digital Design – Principles and Practices, Pearson Education.
3. Digital Circuits and Logic Design, S. Lee, PHI

Data Structures	**CO210**
3-1-0: 4 Credits: 4 Hours	**Prerequisite: CO103, CO104**

Course Outcomes:

Towards the end of the course the student would be able to:

1. use and manipulate several core data structures including: Arrays, Linked Lists, Trees, Stacks, Queues for varieties of problems.
2. use critical thinking and problem solving skills in analyzing computational problems from a variety of sources.
3. identify a problem and analyze it in terms of its characteristics and the information needed to solve it.
4. look at different perspectives of the problem, e.g, storage, time complexity

Course Contents:

Time and Space analysis of Algorithms – Order Notations.

Linear Data Structures : Sequential representations – Arrays and Lists, Stacks, Queues, Strings; Link Representations – Linear linked lists, Circular linked lists, Doubly linked lists; Applications.

Recursion – Design of Recursive Algorithms, Tail Recursion.

Nonlinear Data Structures : Trees – Binary Trees, Traversals and Threads, Binary Search Trees, Insertion and Deletion algorithms, Height Balanced Trees and Weight Balanced Trees, B-trees, B+ trees, Application of trees; Graphs – Representations, Breadth-first and Depth-first Search.

Hashing – Hashing Functions, Collision Resolution Techniques.

Sorting and Searching Algorithms : Bubble sort, Selection sort, Insertion sort, Quick sort, Merge sort, Heap sort, Radix sort.

File Structures: Sequential and Direct Access, Relative files, Indexed files, B+ tree aas index, Multi-index files, Hashed files.

Books:

1. Data Structures and Algorithms, A. V. Aho, J. E. Hoppcroft, J. E. Ullman, Addision Wesley.
2. Fundamentals of Data Structures, E. Horowitzz, S. Sahni, Galgotia Publ.
3. Data Structures using C, A. S. Tanenbaum
4. Algorithms, Data Structures, and Problem Solving, Addision Wesley.
5. Data Management and File Structures, Loomis, Marry, PHI

Formal Languages & Automata **CO216**

3 - 0 - 0 : 3 Credits : 3 Hours ***Prerequsites:*** **CO101**

Course Outcomes:

1. understand the notion of effective computability.
2. emphasize the engineering applications of the theory developed.
3. appreciate the central issues by semi-formal intuitive reasoning.
4. develop the ability to apply the ideas and proof techniques in varied environments

Course Contents:

Alphabet, languages and grammars.

Production rules and derivation of languages.

Chomsky hierarchy of languages.

Regular grammars, regular expressions and finite automata (deterministic and nondeterministic). Closure and decision properties of regular sets. Pumping lemma of regular sets. Minimization of finite automata.

Left and right linear grammars. Context free grammars and pushdown automata.

Chomsky and Griebach normal forms. Parse trees, Cook, Younger, Kasami, and Early's parsing algorithms.

Ambiguity and properties of context free languages. Pumping lemma, Ogden's lemma, Parikh's theorem.

Deterministic pushdown automata, closure properties of deterministic context free languages.

Turing machines and variation of Turing machine model, Turing computability ,

Type 0 languages. Linear bounded automata and context sensitive languages.

Primitive recursive functions. Cantor and Godel numbering. Ackermann's function, mu-recursive functions, recursiveness of Ackermann and Turing computable functions.

Church Turing hypothesis. Recursive and recursively enumerable sets. Universal Turing machine and undecidable problems. Undecidability of post correspondence problem. Valid and invalid computations of Turing machines and some undecidable properties of context free language problems.

Books:

1. J. E. Hopcroft and J. D Ullman: Introduction to Automata Theory, Languages and Computation, Addison Wesley Publ., New York.
2. McNaughton R, Elementary Computability, Formal Languages and Automata, Prentice-Hall.
3. Martin J C, Introduction to Languages and the Theory of Computation, McGraw-Hill International Edition.

References:

1. Buchi A, Finite Automata, Their Algebras and Grammars: Towards a Theory of Formal Expressions, Springer-Verlag.
2. H. R. Lewis and C. H. Papadimitriou: Elements of the Theory of Computation, Prentice Hall, Englewood Cliffs.
3. F. Hennie: Introduction to Computability, Addison Wesley Publ., New York.

Computer Architecture & Organization — CO214

3 - 0 - 1 : 4 Credits : 5 Hours — ***Prerequsites:*** **CO 103, CO104**

Course Outcomes:

1. be well conversant in the concepts of computer architecture and organization for several engineering applications.
2. be able to analyze and design different memory organizations.
3. be well equipped with the knowledge of Cache Mapping techniques and Virtual memory.
4. be able to demonstrate programming proficiency using the various addressing modes and data transfer instructions of the target computer.

Course Contents:

Basic organization of the computer and block level description of the functional units from program execution point of view; Fetch, decode and execute cycle;
Assembly language programming: Instruction set, instruction cycles, registers and storage, addressing modes; discussions about RISC versus CISC architectures;

Inside a CPU: information representation, computer arithmetic and their implementation; control and data path, data path components, design of ALU and data path, control unit design;
Memory and I/O access: Memory maps, Read Write operations, Programmed I/O, Concept of handshaking, Polled and Interrupt driven I/O, DMA data transfer; I/O subsystems: Input-Output devices such as Disk, CD- ROM, Printer etc.; Interfacing with IO devices, keyboard and display interfaces;
Inside the Memory: memory organization, static and dynamic memory; Cache memory and Memory Hierarchy – Cache memory access techniques; Virtual memory;
Introduction to Parallel Architectures: Instruction Level Parallel Processors- Pipelined, VLIW, Superscalar; Multiprocessors & Multicomputer Architectures, Vector Processing.

Laboratory experiments:

The assignments should cover the following:

1. Assignments on assembly language programming;
2. Experiments on synthesis / design of simple data paths and control unit;
3. Assignments on interfacing devices and systems like data acquisition systems ;

Development kits as well as PCs/Workstations may be used for the laboratory, along with design / simulation tools as and when necessary.

Books:

1. Computer Architecture and Organization, Hayes J. P., McGrawHill
2. Computer Organization, Hamacher, Zaky, Vranesic, McGrawHill
3. Computer System Architecture, Mano M. M.

Computer Organization Lab **CO215**

0 - 0 - 1 : 1 Credits : 2 Hours ***Prerequsites:*** **CO103**

Abstract:

This course is comprised of a laboratory component that is to be covered in parallel to Computer Architecture and Organization (CO212). The course is aimed at providing a practical understanding on building of the functional units and their integration and for appreciation of the issues on programming at the machine level.

Course Outcome:

1. Practical understanding of the architectural aspects of a computer at the machine level
2. Getting equipped with practical aspects of the working of a computer that will be useful for courses such as Operating Systems, Embedded Systems etc.
3. Ability to work on development of digital systems.

Course Content:

Circuit Design and Simulation: Register, Counter, Adder, Multiplier, Data Paths, Control Unit, ALU etc.
Introduction to 8086 family microprocessors, Architecture of 8086 - Register set, Concept of segments, use of the register set, use of stack, Instruction set, PSW Flags.

8086 assembly language programming: Software interrupts, Data Input/Output, Arithmetic Operations, String Handling, Branching, Looping, showing conditional and unconditional branches, and LOOP instruction, Creating Subroutines.

List of Laboratory Assignments:

Set 1: Designing CPU components using a Simulator:

- Register, Counter
- Adder, Multiplier
- Data Paths, Control Unit
- ALU, Memory unit

Set 2: 8086 Assembly Language Programming:

- Taking keyboard input for characters and numbers, Displaying and working with multi-digit number, Arithmetic operations
- Comparison operators, conditional and unconditional jumps, loops
- Working with array of numbers and strings of characters, finding average/mean of numbers, searching
- Writing Procedures, passing and returning values, use of stack

Text Books:

1. Computer System Architecture, Mano M. M, Pearson.
2. Guide to Assembly Language Programming in Linux, Sivarama Dandamudi, Springer

Reference Books:

1. IBM PC Assembly Language and Programming, Peter Abel, 3e, PHI.
2. Computer Organization and Design: The Hardware/ Software Interface, Patterson and Hennessy, Elsevier.
3. Computer Organization and Architecture: Designing for Performance 9E, William Stallings, Pearson Education.
4. Computer Organization, Hamacher, Zaky, Vranesic, McGrawHill.

Design and Analysis of Algorithms — CO206

3 - 0 - 1 : 4 Credits : 5 Hours — *Prerequsites:* CO203

Course Outcomes:

At the end of the course, students will be able to

1. compare, contrast, and apply the key algorithmic design paradigms: brute force, divide and conquer, decrease and conquer, transform and conquer, greedy, dynamic.
2. compare, contrast, and apply key data structures: trees, lists, stacks, queues, hash tables, and graph representations.
3. define, compare, analyse, and solve general algorithmic problem types: sorting, searching, string processing, graphs, and geometric algorithms.
4. implement, empirically compare, and apply fundamental algorithms and data structures to real-world problems.

5. compare, contrast, and apply algorithmic trade offs, get an introduction to the world of complexity theory.

Course Contents:

Algorithms and Complexity – asymptotic notations, orders, worst-case and average-case, amortized complexity.

Basic Techniques – divide & conquer, dynamic programming, greedy method, backtracking, branch and bound, randomization.

Data Structures – heaps, search trees, union-find problems.

Applications – sorting & searching, combinatorial problems, optimization problems, computational geometric problems, string matching.

Graph Algorithms – BFS and DFS, connected components, spanning trees, shortest paths, max-flow. NP-completeness.

Approximation algorithms.

Laboratory: The laboratory component will emphasize two areas:

Implementation of algorithms covered in class: This will involve running the algorithms under varying input sets and measuring running times, use of different data structures for the same algorithm (wherever applicable) to see its effect on time and space, comparison of different algorithms for the same problem etc.

Books:

1. Introduction to Algorithms, Cormen et al., McGrawHill
2. Aho A, Hopcroft J., Ullman J., The Design and Analysis of Algorithms, Addison-Wesley.

Operating Systems **CO309**

3-0-0: 3 Credits: 3 Hours *Prerequisites:* CO210 (Data Structure), CO214(CAO)

Abstract: The course CO309 Operating Systems intends to build the basic concepts of the operating system software and its implementation details for computer system. It also covers topics with case studies giving the students opportunity to explore practical operating systems like - MS Windows, Unix, Linux. The course will be instrumental to build the confidence in the students to develop the introductory knowledge of design and develop an operating system.

Course Outcomes:

Towards the end of the course the student would be conversant with

1. Operating systems and OS design issues
2. OS based programming fundamentals
3. be familiar with OS system software design concepts
4. be able to design and develop OS features
5. OS based application development

Course Contents:

Overview: Evolution of Operating Systems, current status and future trends. Structural overview, system calls, functions of OS, Hardware requirements: protection, context switching, privileged mode

Concept of a process: states, operations with examples from UNIX/Linux (fork, exec) and/or Windows. Process scheduling, interprocess communication (shared memory and message passing), UNIX/Linux signals, cooperating and concurrent processes, tools, and constructs for concurrency,

Threads: thread management, multithreaded model, scheduler activations, examples of threaded programs and applications.

Scheduling: multi-programming and time sharing, scheduling algorithms, multiprocessor scheduling, thread scheduling (examples using POSIX threads).
Process synchronization: mutual exclusion, shared data, critical sections, classical two process and n-process solutions, hardware primitives for synchronization, lock, semaphores, monitors, block and wakeup, classical problems in synchronization (producer-consumer, readers-writer, dining philosophers, etc.).

Deadlocks: modeling, characterization, prevention and avoidance, detection, and recovery.

Memory management: with and without swapping, MMU, Contiguous and non-contiguous allocation, paging and segmentation, demand paging, virtual memory, page replacement algorithms, working set model, thrashing, and implementations from operating systems such as UNIX, Windows. Current Hardware support for paging: e.g., Pentium/ MIPS processor etc.

Secondary storage and Input/Output: device controllers and device drivers, disks, scheduling algorithms, file systems, directory structure, device controllers and device drivers, disks, disk space management, disk scheduling, NFS, RAID, other devices and operations on them, UNIX FS, UFS protection and security.

Virtualization: Virtual Machine (VM), concept of hypervisor and virtual machine manager (VMM), types of hypervisor: kernel-based and hosted hypervisor, open source virtual machine design in Linux: Kernel-based Virtual Machine (KVM).

Protection and security: Illustrations of security model of UNIX and other OSs. Examples of attacks. Pointers to advanced topics (distributed OS, multimedia OS, embedded OS, real-time OS, OS for multiprocessor machines, mobile OS, cluster OS).

Case study: Design of UNIX, Linux, Windows, Android

Textbooks:

1. Operating System Concepts, Abraham Silberschatz, Peter B. Galvin, Greg Gagne, 9th Ed., John Wiley, 2018.
2. Operating Systems: Internals and Design Principles, William Stallings, Pearson, 9th Ed., 2019.

Reference Books:

1. Operating systems: Concepts and Design, M. Milenkovic (McGraw Hill, 2001)
2. Modern Operating Systems, A. S. Tanenbaum (Pearson, 2009)

3. Design of the Unix Operating System, M. J. Bach (Prentice Hall of India, 1986)
4. Operating Systems: Three Easy Pieces, Remzi and Andrea Arpaci-Dusseau
5. http://pages.cs.wisc.edu/~remzi/OSTEP/
6. Understanding Linux Kernel 2.6, Bovet and Chesti (Orelly), 2005)
7. Professional Linux Kernel Architecture, Wolfgang Mauerer, (Wiley)
8. Linux Kernel Development, R. Love (Addison Wesely, 2010)
9. Professional Android Application Development, R. Meier (John Wiley & Sons)

Operating Systems Lab	**CO310**
0-0-1: 1 Credit: 2 Hours	*Prerequisites:* CO309, CO103 (IC), CO104 (CL), CO210 (Data Structure)

Abstract: The course CO310 Operating System Lab intends to build hand on understanding about some of the important topics covered in CO309 course. It will familiarize with UNIX system calls for process management and inter-process communication, process synchronization, memory management and file system management; experiments on process scheduling and other operating system tasks through programming, simulation / implementation under a simulated environment.

Course Outcomes:

After completion of course, students would be able

1. to grasp knowledge for implementation of operating system software features,
2. to understand the working of an operating system kernel,
3. to master on using system level programming especially the kernel coding using C and java /C++ languages,
4. to master on develop algorithm for operating system design
5. to grasp knowledge for implementing file system concepts
6. to master in using simulation tools, system utilities

Course Contents:

Shell scripting primer using Bourne shell, Bash scripting for beginner, use of awk etc.

Create process (use of fork(), exec() etc. system calls), implement a process ownselves

Use system calls signal(), kill(), creating POSIX threads, using thread library Pthread library using system calls pthread_create() and pthread_exit()

Implementation of file locks using fcntl for basic file access synchronization

Use of basic IPC mechanism with pipe(), mknod(), using message queue, shared memory.

Learn to use synchronization of processes with semaphore and other tools,

Dynamic memory allocation, LKM programming, Device driver for char and block devices

Open source Linux kernel source code browsing and understanding.

Android based application development.

Open source hypervisor development

Reference Books:

1. Unix Network Programming, Stevens, Vol-1, Addison-Wesley, 3rd Ed, 2003 & Vol-2, Prentice Hall, 2nd ed, 1998
2. Design of Unix Operating System, M. J. Bach (Prentice Hall of India, 1986)
3. Linux Device Drivers, J. Corbet and A. Rubini, Orelly, 2005
4. Professional Android Application Development, R. Meier (John Wiley & Sons, 2014)
5. Writing a simple Operating System from Scratch, N. Blundell, University of Birmingham, UK, 2010
6. http://www.ee.surrey.ac.uk/Teaching/Unix/
7. http://developer.android.com/guide/index.html
8. Operating Systems: Three Easy Pieces, Remzi and Andrea Arpaci-Dusseau http://pages.cs.wisc.edu/~remzi/OSTEP/
9. Web links and tutorial materials will be provided time to time

Data Structures using Object Oriented Programming Lab CO211

0 - 1 - 2 : 3 Credits : 5 Hours ***Prerequsites:*** **CO101, CO102**

Course Content:

Review of elementary programming

Recursion: The concept of recursion; recursive specification of mathematical functions (such as factorial and Fibonacci); simple recursive procedures (Towers of Hanoi, permutations, fractal patterns); divide- and-conquer strategies; recursive backtracking; implementation of recursion

Introduction to computational complexity: Asymptotic analysis of upper and average complexity bounds; big-O notation; standard complexity classes; empirical measurements of performance

Fundamental computing algorithms: O(N log N) sorting algorithms (Quicksort, heapsort, mergesort); hashing, including collision-avoidance strategies; binary search trees

Fundamental data structures: Linked structures; implementation strategies for stacks, queues, hash tables, graphs, and trees; strategies for choosing data structures

Object-oriented programming: Object-oriented design; encapsulation and information-hiding; separation of behaviour and implementation; classes, subclasses, and inheritance; polymorphism; class hierarchies; collection classes and iteration protocols; fundamental design patterns

Books:

1. Data Structures and Algorithms, A. V. Aho, J. E. Hoppcroft, J. E. Ullman, Addision Wesley.
2. Fundamentals of Data Structures, E. Horowitz, S. Sahni, Galgotia Publ.

3. Data Structures using C, A. S. Tanenbaum, PHI
4. Herbert Schild: The Complete Reference to C++, Osborne McGrawHill.Bjarne Stroustrup: The C++ Programming Language, Addison Wesley

Data Communication **CO218**

3 - 0 - 0 : 3 Credits : 3 Hours ***Prerequsites:*** **CO103**

Course Outcomes:

1. gain overall idea on the network architecture and issues associated with the different layers.
2. calculate medium capacity for given bandwidth and signal-noise-ratio.
3. understand different analog/digital signals, conversion of signal from one from to another and different modulation techniques.
4. know about different medium access control mechanisms used in telephone/computer network along with relative advantages and disadvantages of one technique over the other.
5. apply error detection/correction techniquesused in data communications.
6. know about different devices associated with datacommunications such as Repeaters, Switches, Routers etc. and their working principles.

Course Contents:

Overview: Objectives and Applications of Computer Communication.

Computer Communication and Network Architecture: ISO-OSI reference model, design philosophy, layer, protocol, interface, and service concepts. Layer wise functionality.

Physical Layer: Concepts of data transmission: signal, communication channel, channel capacity, distortion & noise, line coding; modulation(analog and digital), modem; multiplexing- FDM, TDM, WDM, CDM etc; OFDM & spread spectrum techniques; switching, communication media—guides & unguided; standard protocols, RS- 232C, RS-449, X.21, xDSL, SONET, Frame relay, ATM etc.

Medium Assess Control in broadcast networks: ALOHA, CSMA, CSMA/CD, CSMA/CA, token ring, token bus etc.

Standard LAN Protocols: (IEEE 802.X), FDDI, satellite networks, LAN switching, VLAN, WLAN, PAN and WiMax.

Data link layer: Framing, Error control techniques, Data link protocols and their performance, HDLC and PPP protocol.

Network layer: Introductory concepts and issue: Routing, Congestion and deadlock control Algorithms, Internetworking issues and devices, gateways, bridges and routers, IP & X.25 protocols.

Communication Laboratory:

Laboratory: Generation, testing, of AM, FM, and PM, Transmitter and receiver, PCM codec; Flow control, Error Control and MAC protocols on LAN trainer kit.

Books:

1. Stalling, Data and Computer Communication, 8e, PHI (EEE)
2. Tanenbaum A.S., Computer Network, 5e, PHI (EEE)

References:

1. Forouzan B. A, Data Communication and Networking, 5e, Tata McGrawHill
2. Leon-Garcia, Widijaja, I., Communication, 5e, PHI (EEE)
3. B. P. Lathi, "Modern Analog and Digital Communication Systems", 3/e, Oxford University Press, 1998.

Software Engineering	**CO311**
3-0 -0: 3 Credits: 3 Hours	**Prerequisites: CO209(CW)**

Abstract: This course is to make students familiar with overall concepts involved in the software development process including the current scenario prevailing in the software industry. The course covers topics on software development process such as feasibility study, requirement specification, different facets of software design, coding, testing etc. The course also covers project planning and management.

Course Outcomes:

After the course, a student will be:

1. Familiar with overall concepts involved in the software development project.
2. Able to understand fundamental concepts of requirements engineering and Analysis Modelling.
3. Able understand the various software design methodologies
4. Able to understand various testing and maintenance measures.
5. Able to understand the process of project planning and management techniques.

Course Contents:

Module I: Introduction to software engineering

Evolution and Impacts of Software Engineering, Software life cycle models and their comparative study

Module II: Requirements analysis and specification

Software requirements, Software requirements engineering, Requirements specifications techniques. Formal requirements specification and verification - axiomatic and algebraic specifications

Module III: Software design

Basic Issues in software design- modularity, cohesion, coupling and layering. Design approaches- top-down & bottom-up, Function-oriented software design, data flow diagram and structured charts, Object-oriented design, Object modelling using UML, Use case model development, Design specification and notations

Module IV: Software implementation

Structured coding techniques, coding styles, and standards; Guidelines for coding and documentation, automatic code generation.

Module V: Software verification, validation, and maintenance

Theoretical foundation; black box and white box approaches; Integration and system testing, Static and Dynamic Analysis tools, Software Maintenance – Types, maintenance models, reverse and forward Engineering, Maintenance Cost models, Computer Aided Software Engineering (CASE)

Module VI: Software reliability

Definition and concept of reliability; software faults, errors, repair, and availability; reliability and availability models.

Module VII: Software Project Management

Project Management, Project planning and control, Cost estimation and evaluation techniques, cost estimation based on COCOMO model and Raleigh model; Project Scheduling using PERT and GANTT charts, Organizational structure planning, project formats and team structures; Risk analysis and planning, Software configuration management

Text Books:

1. Rajib Mall, Fundamentals of Software Engineering, Prentice Hall India Learning Private Limited (Fifth Edition, 2018)
2. Ian Sommerville, Software Engineering, Pearson Education, 2017

References:

1. R.S. Pressman, Software Engineering: A Practitioner's Approach, McGraw Hill, 2014
2. Robert C Martin, Clean Code, Pearson Education, 2012
3. Steve McConnell, Code Complete, Dreamtech Press, 2011
4. Gregor Hohpe, Enterprise Integration Patterns, Pearson, 2011
5. Martin Fowler, Patterns of Enterprise Application Architecture, Pearson, 2012

Project I (using Software Engineering perspective) **CO317**

0-0-2 :2 Credits: 4 Hours **Prerequisites:CO311**

Abstract: Each student will carry out a project work individually under the guidance of a faculty member. The project shall consist of design/ development/ implementation work with a view to understand and apply Software Engineering principles to construct software that is reliable, reasonably easy to maintain. Students will gain hands-on experience on the design and development of a software system through application of software engineering knowledge and skills.

Course Outcomes:

After the course, a student will be able to:

1. Earn practical knowledge of developing a software system.
2. Apply the software engineering concepts in a real-life software project.
3. Earn an exposure to technology framework and tools for software development.
4. Develop test suites based on software testing principles, test and debug software modules in Java and C++ etc.
5. Use of appropriate CASE tools and other tools such as configuration management tools, program analysis tools in the software life cycle
6. Prepare relevant project documents.

Course Contents:

Design, development and deployment of software system over multiple iterations; Application of software engineering concepts and knowledge such as requirements engineering, software architecture, design, development, testing and validation; Usage of technology frameworks and tools for software development; Exposure to real world software development environment under constraints of uncertain requirements and deadlines.

References:

1. Rajib Mall, Fundamentals of Software Engineering, Prentice Hall India Learning Private Limited (Fifth Edition, 2018)
2. R.S. Pressman, Software Engineering: A Practitioner's Approach, McGraw Hill, 2014
3. M. Shooman, Software Engineering, McGraw Hill, 1983
4. Roy Osherove, The Art of Unit Testing, Second Edition, Manning Publications; 2nd edition (December 7, 2013)
5. Steve McConnell, Code Complete: A Practical Handbook of Software Construction, Microsoft Press; 2nd edition (June 19, 2004)
6. Emma Jane Hogbin West, Git for Teams: A User-Centered Approach to Creating Efficient Workflows in Git, O'Reilly Media; 1 edition (September 12, 2015)

Database Systems **CO312**

3-0-0: 3 Credits: 3 Hours *Prerequisites:* CO210 (DS)

Abstract: The course Database System is an introductory course on database systems. The course covers the basic concepts of database, data models, database architecture, relational database languages, SQL, functional dependency and normalization, database transactions

Course Outcomes:

After completion of this course:

1. The students should clearly understand the concepts of database systems, their advantages, and applications.
2. The students should be able to design a database system for a given database problem.
3. The students should be familiar with the emerging database application areas.

Course Contents:

Introduction & Overview: Concept of database, Characteristics of database, Advantages, data independence, redundancy Control; Database architecture - ANSI model.

Modelling of real-world situation (data models): ER model, EER model

Relational data model: relational model concepts, relational algebra and calculus, SQL, ER/EER to relational model mapping,

Functional dependencies and normalization: functional dependencies, normal forms, decomposition, multi-valued functional dependency, and higher normal forms

Database Indexing and hashing: B-Tree, B+ Tree, static and dynamic hashing

Database Transaction concepts, query evaluation overview, security, and recovery

Distributed Database

Brief introduction to emerging database applications (like Hadoop, NoSQL etc.)

Text Books:

1. Fundamentals of Database Systems, Sixth(2011)/Seventh(2017)Edition, ELMASRI and NAVATHE, Pearson
2. Database Systems Concepts, Sixth (2010)/Seventh(2019) Edition, A. SILBERSCHATZ, H. F. KORTH, S SUDARSHAN, McGraw Hill,

References:

1. Database Management Systems, 3rd Edition, - Raghu Ramakrishnan, Johannes Gehrke, McGraw Hill, 2014
2. An Introduction to Database Systems, 8th Edition, C.J Date, Pearson, 2003
3. Fundamentals of Database Systems, by Leon & Leon, Tata McGraw Hil, 2008
4. SQL & NoSQL Databases, Meier, Andreas and Kaufmann, Michael, eBook, Springer 2019
5. Getting Started with NoSQL, Gaurav Vaish, Packt Publishing, March 2013
6. Hadoop: The Definitive Guide, 4th Edition, Tom White, O'Reilly Media, Inc., 2015

Database Systems Lab **CO313**

0 - 1 - 1: 2 Credits: 3 Hours *Prerequisites:* CO210

Course Outcomes:

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

1. Create and manage a database using RDBMS like Oracle.
2. Perform queries on the database.
3. Build user-defined functions and procedures using PL/SQL.
4. Develop database applications.

Course Contents:

Introduction: Introduction to a RDBMS like Oracle

SQL: data types, DDL, DML

SQL functions, PL/SQL and user-defined function, triggers

DBA commands

Role-based authorization

Development of database applications using tools/languages like Forms & Reports, PHP, Java, JavaScript etc.

Laboratory works / experiments:

- Basic commands of SQL and SQL Plus
- Creating and modifying database tables, inserting, deleting and updating data in database tables using SQL
- SQL commands for retrieving data from database tables
- Creating and updating database triggers
- Writing and executing SQL scripts
- SQL scripts for building simple reports
- Basics of PL/SQL
- User-defined functions and procedures
- DBA commands and database authorization
- Developing GUIs using Oracle Forms & reports/PHP/Java etc.
- Developing reports using Oracle Forms & reports/PHP/Java etc.

Text Books:

1. SQL The Complete Reference, 3rd Edition, James Groff, Paul Weinberg, McGraw Hill Education, 2017
2. Oracle 12c: The Complete Reference, Kevin Loney, George Koch McGraw Hill/Osborne, 2017
3. Learning PHP, MySQL & JavaScript 5e (Learning PHP, MYSQL, Javascript, CSS & HTML5), Robin Nixon, O'Reilly, 2018

Reference Books:

1. Beginning PHP and Oracle, W. J. Gilmore and B. Bryla, Apress, Berkley, CA, USA, 2007
2. The Underground PHP and Oracle Manual, Release 2 Christopher Jones and Alison Holloway, Oracle, 2012
3. Mastering Oracle SQL, 2nd Edition Sanjay Mishra and Alan Beaulieu, O'Reilly Media, 2004
4. Oracle PL/SQL Programming, 5/6th Edition, By Steven Feuerstein, Bill Pribyl, O'Reilly Media, 2016
5. Oracle Forms developer's handbook, Albert Lulushi, Pearson Education, 2012
6. Oracle 9i development by example, Dan Hotka, Pearson Education, 2001

Computer Graphics **CO303**

3-0-1: 4 Credits: 5 Hours *Prerequisites*: *CO210*

Abstract: This course aims to familiarize students with drawing images while learning the underlying algorithms for two and three dimensional graphics and working with animations

on the computer. This course is programming intensive starting from basic drawing on the computer to special effects in movies.

Course Outcomes:
This course will give the students hands-on experience at developing interactive, real-time rendering applications using C/OpenGL. At the end of the semester the course outcomes would be:

1. Designing and implementing an application which illustrates the use of various rasterization and transformation techniques.
2. Studying, comparing, and implementing various methods for computer representation of objects.
3. Animating the smooth motions of objects in a scene and predicting collisions between the implemented objects.

Course Contents:

Display Devices: Line and point plotting systems; raster, vector, pixel and plotters, Continual refresh and storage displays, Digital frame buffer, Plasma panel displays, Very high resolution devices, High-speed drawing, Display processors, Character generators, Color-display techniques (Shadow-mask and penetration CRT, analog false colors, hard-copy color printers).

Display description: Screen co-ordinates, user co-ordinates; Graphical data structures (compressed incremental list, vector list, use of homogeneous co-ordinates); Display code generation; Graphical functions.

Output Primitives: Line drawing algorithms, Circle and Ellipse generating algorithms, Other curves & Conic sections, Polynomials and spline curves.

Filled area primitives: Scan-line polygon fill algorithm, Inside-outside tests, Boundary fill algorithm, Flood fill algorithm, Character generation.

Attributes of output primitives: Line attributes, Curve attributes, Color and grayscale levels, color tables, Area fill attributes: fill styles, Character attributes, Antialiasing.

2D geometric transformations: Basic transformation: translation, rotation, scaling, Composite transformations, Reflection and shearing, Transformations between coordinate systems, Affine transformations.

2D viewing: Viewing pipeline, window-to-viewport coordinate transformation, Clipping operations, Point clipping, Line clipping algorithms, Polygon clipping algorithms, Curve clipping, Text clipping

Interactive Graphics: Pointing and positioning devices (cursor, light pen, digitizing tablet, the mouse, track balls). Interactive graphical techniques; Positioning, Elastic Lines, Inking, Zooming, Panning, Clipping, Windowing, Scissoring. Basic positioning methods.

3D Concepts: 3D display methods: Parallel & perspective projection, Depth cueing, Visible line and surface, identification, Exploded and cutaway views, 3D and stereoscopic views, Polygon surfaces, tables, equations, meshes, Curved lines and surfaces. Quadric surfaces, sphere, ellipsoid, torus, superellipse, superellipsoid, Spine representations, Bezier, cubic Bezier curves and surfaces, Sweep representations, Octrees & BSP trees, Fractals.

3D transformations and Viewing: 3D transformations & composite transformations, Viewing pipeline, viewing coordinates, Wire-frame perspective display, Perspective depth, Projective transformations

Visible surface detection methods: Back-face detection, A-buffer method, Scan-line method, Depth-sorting method, BSP-tree method, Octree methods, Ray casting and wireframe methods

Illumination models and surface rendering: Basic illumination models, specular reflection and Phong model, Hidden line and surface elimination, Transparent solids, Shading, halftone patterns and dithering, Ray tracing, Texture mapping

Animation: Animation sequence designing, key framing, morphing, simulated accelerations, motion specifications.

Computer Graphics using OpenGL: An introduction to OpenGL basic graphics primitives, Transformations using OpenGL, Drawing 3D scenes with OpenGL, Introduction to Rendering methods (with various shaders - vertex shader, fragment shader).

Text Books:

1. John F. Hughes, Andries Van Dam, Morgan Mc Guire ,David F. Sklar , James D. Foley, Steven K. Feiner and Kurt Akeley, "Computer Graphics: Principles and Practice", , 3rd Edition, Addison- Wesley Professional,2013.
2. Edward Angel, Interactive Computer Graphics: A Top-Down Approach with OpenGL, 4th edition, Addison-Wesley, 2005.

Reference Books:

1. Donald Hearn and M. Pauline Baker, Warren Carithers,"Computer Graphics With Open GL", 4th Edition, Pearson Education, 2010.
2. Peter Shirley, Michael Ashikhmin, Michael Gleicher, Stephen R Marschner, Erik Reinhard, KelvinSung, and AK Peters, Fundamental of Computer Graphics, CRC Press, 2010.

System Software and Compiler Design **CO314**

3-0-1: 4 Credits: 5 Hours *Prerequisites:* CO214, CO216

Abstract: The course CO314 System Software and Compiler Design gives an idea of system software in general, and programming tools. It provides an overview of text editors, translators, linkers, loaders, debugger, and some common text processing tools. The course includes design and implementation of assemblers and compilers and macro processors. The task of compilation is an elaborate one with multiple phases and many theoretical aspects. This course includes suitable modelling and mechanisms for this task and some useful tools. The theoretical concepts of translators are complemented by adequate practical exercises. This course also covers the concept of program linking, relocation and loading, and the functions and features of text editors, debuggers, and some common text processing tools.

Course Outcomes:

Towards the end of the course the student would understand-

1. The task of translation of assembly language and high-level language programs into machine language programs,
2. Description of languages using formal grammars and use of tools for recognition of input strings.
3. Automate the task of parser construction and produce translations of input.

4. Implement symbols tables.
5. The run-time memory and execution environment of programs.
6. The task of code optimization.
7. The scope of a compiler in leveraging the advances of computer architecture.
8. The various types of linking of program modules.
9. The tools available for program debugging, version control, building large executable programs.
10. The common tools for text search and simple editing.

Course Contents:

Overview: Definition and classification of system software, System software for program creation- editors, programming language translators, linkers and loaders, debuggers.

Introduction to machine language, assembly language and high-level language.

Assemblers: Outline of an assembler Lexical analysis: specification of tokens, regular expressions, token recognition Assembler data structures, single pass and multi-pass assemblers, Assembler macros and macro-processors.

Compilers: Characteristics of High Level Languages (HLL), Outline of a compiler; Syntactic specification of HLL using CFG, ambiguity.

Parsing: Parse trees and derivations, top-down parsing, recursive descent parser; bottom-up parsing, Operator precedence parser, LR parsers- SLR, CLR and LALR, handling ambiguity Syntax directed translation, Intermediate Code generation, Semantic analysis, Attribute grammars, Type checking, type conversion and overloading; Symbol tables for compilers.

Runtime environments: activation records, heap management, garbage collection Error detection and recovery.

Code optimization: basic blocks, directed acyclic graph representation, local and global optimization, data flow analysis, register allocation, Loop optimization, Instruction Scheduling and Software Pipelining, Automatic Parallelization.

Text editors: Basic features of a text editors, data structures for text editors.

Linkers and loaders: Basic concepts, Relocation, Static and Dynamic linking, shared libraries, loaders, overlays.

Debugger: features, case study: gdb (laboratory practice)

Unix Utilities: make, RCS, grep, sed (laboratory practice).

Practical Topics:

1. Familiarization with text files and binary file operations.
2. Implementation of lexical analyser. Use of tool *flex*.
3. Implementation of a two-pass assembler
4. Implementation of a simple desk calculator
5. Implementation of operator precedence parsing
6. Representation of context free grammar in data structure and algorithm for creation of an operator precedence table.
7. Implementation of recursive descent parsing
8. Algorithm for creation of LL(1) parsing table.
9. Use of tool *bison* for creating an LALR parser.
10. Use *bison* for creating three address code for simple program segment and a symbol table.

11. Use of *gdb* debugger.
12. Use of tools make, RCS, grep, sed
13. Familiarization of object code format and linker in Linux.

Text Books:

1. D M Dhandhere. System programming and operating systems, 2e. Tata McGraw Hill, 1999.
2. Alfred Aho, Jeffrey Ullman, Monica S. Lam, and Ravi Sethi. Compilers: Principles, Techniques, and Tools, 2e. Pearson, 2007.

Reference Books:

1. Andrew W Appel. Modern compiler implementation in Java, 2e. Cambridge, 2009.
2. Sumitabha Das, Unix System V.4 Concepts and Applications, TMH.
3. Linux Manuals.
4. Windows Manuals.

Computer Networks **CO315**

3-0-0: 3 Credits: 3 Hours *Prerequisites:* CO218 (Data Communication)

Abstract: CO315 is an introductory course in the field of computer network at Tezpur University. This course assumes that the students have the prerequisite knowledge of data communications, where students have learned basic networking concepts up to the medium access control sublayer. This course covers the basics of TCP/IP protocol stacks starting from Network Layer and going up to the different applications of computer networks. It is designed to teach the students the very basics of computer networks as an infrastructure and to make them understand the needs of different kinds of applications which are designed to run on this infrastructure.

Course Outcomes:

At the end of the course, a student will be able to

1. Understand the need of a layered Architecture for Computer Network.
2. Understand the key functions performed by different widely known protocols at Network, Transport and Application Layers.
3. Configure a network with Router, ARP, DHCP, DNS, and Gateway etc.
4. Apply mathematical foundations to solve computational problems in computer networking
5. Understand the basic working behavior of TCP and UDP.
6. Know the use of different networking devices.
7. Understand the needs of network security and usages of different security mechanisms.
8. Understand the needs and protocols used in different network applications
9. Understand the basic mechanisms used in data compression

Course Contents:

Review of computer network architecture and the subnet layers.

Network layer: Design issues, Addressing, Routing, internetworking issues, Internet Protocol, IPv4, IPv6, ARP, DHCP, ICMP, Routing algorithms: Distance vector, Link state, Metrics, Inter-domain routing. Subnetting, Classless addressing, CIDR based routing and forwarding, Network Address Translation, Mobile IP, MPLS, VPN.

Introductory queuing theory.

Transport layer: Design issues, UDP, TCP. Connection establishment and termination, sliding window, flow and congestion control, timers, retransmission, TCP extensions (Tahoe, Reno, New-Reno, SRP etc.), RPC, RTTP

End-to-end Data : Presentation formatting issues and methods: XDR , ASN.1, NDR; data compression, lossless compression algorithm, run length encoding, DPCM, Huffman coding, dictionary-based methods: LZ77, LZ78 and their variations, image compression- JPEG, video compression- MPEG, newer compression standards;

Network Security: Concepts of symmetric and asymmetric key cryptography. Sharing of symmetric keys - Diffie Hellman. Public Key Infrastructure. Public Key Authentication Protocols. Symmetric Key Authentication Protocols. Pretty Good Privacy (PGP), IPSec, Firewalls.

Application: Client-Server and Peer-to-Peer applications, Application layer examples: DNS, SMTP, IMAP, HTTP, P2P systems, BitTorrent, network file system, network management.

New trend in Networking: SDN, IoT, Cloud etc.

Text Books:

1. AS Tanenbaum and DJ Wetherall, Computer Networks, 5th Ed., Pearson, 2016.
2. LL Peterson and BS Davie, Computer Networks: A System Approach, 5th Ed.,Morgan Kaufmann Publishers Inc., 2011

References Books:

1. JF Kurose and KW Ross, Computer Networking: A Top-Down Approach featuring the Internet, 3rd Edition, Pearson Education, 2017.
2. K Sayood, Introduction to data Compression, 5th Ed., Morgan Kaufmann, 2020
3. WR Stevens, UNIX Network Programming, 2nd Ed., 2015, Pearson
4. DE Comer and DL Stevens, TCP/IP Programming Vol. I, II , III, 2nd Ed, 2015, Pearson.

Computer Networks Lab **CO316**

0-0-1: 1 Credit: 2 Hours *Prerequisites:* CO218 (DC)

Course Outcomes:

At the end of this course the student should be able to:

1. Apply packet sniffing tools to capture packets and analyse
2. Write client-server applications in C/C++/Java
3. Use different OS tools to configure network and network protocols
4. Set up LAN using networking devices
5. Use network simulators to study behaviour of different protocols
6. Analyse performance of various communication protocols using packet sniffers

Course Contents:

Experimental study of application protocols such as HTTP, FTP, SMTP, using network packet sniffers and analyzers such as Ethereal, tcpdump, Wireshark etc.

Client-server based application development using socket programming in C/C++/Java (both TCP and UDP sockets)

Experiments with packet sniffers to study the behaviour of TCP protocol.

Using OS tools (netstat etc.) to understand TCP protocol (connection establishment, connection termination, retransmission timer behaviour, congestion control behaviour)

Setting up a small IP network - configuring interfaces, IP addresses and routing protocols to set up a small IP network

Introduction to network simulator tools (NS-2/3, GNS3, Mininet, QualNet etc.) to study behaviour of MAC (IEEE 802.3, IEEE 802.11) and other protocols.

Books:

1. WR Stevens, UNIX Network Programming, 2nd Ed., 2015, Pearson
2. Kirch and Dawson, Linux Network Administrator's Guide, O'relly

References:

1. Online and weblink reference manuals for network simulators

Graph Theory	**CO217**
3-0-0: 3 Credits: 3 Hours	*Prerequisites:* CO210

Course Outcomes:

1. understand clearly the basic theories of Graph Theory.
2. map a given real life problem to a graph theory problem.
3. apply graph theory concepts in solving real life problems.
4. design and/or implement algorithms for solving some common graph theory problem.

Course Contents:

Graph : Incidence and degree; Handshaking Lemma; Isomorphism; Subgraphs and Union of graphs; Connectedness; Walks, Paths and Circuits; Components and Connectedness; Walks, Walks, Paths and Circuits; Components and Connectedness algorithms; Shortest Path Algorithms, Eulerian graph, Fleury's algorithm and Chinese postman problem; Hamiltonian graph - necessary and sufficient conditions; Traveling salesman; Bipartite graph.

Tree: Properties of trees; Pedant vertices in a tree; Center of a tree; Rooted binary trees; Spanning trees, Spanning tree algorithms; Fundamental circuits; Spanning trees of a weighted graph; cut-sets and cut- vertices; Fundamental cut-sets; Connectivity and separativity; network flow; max-flow min-cut theorem.

*Planner graph***:** Combinatorial and geometric dual; Kuratowski's graph; detection of planarity; Thickness and crossings.

Matrix representations of graph: Incidence; Adjacency; matrices and their properties.

Colourings: Chromatic number : Chromatic polynomial; The six and five colour theorems; The four colour problem.

Directed graphs: Binary relations; Directed graphs and connectedness; directed trees; Aborecence; Polish method; Tournaments.

Counting of labeled trees: Cayley's theorem; Counting methods; Polya theory.

Books:

1. Deo, N.: Graph Theory with Applications to Engineering and Computer Science.
2. Harary : Graph Theory, PHI (EEE)

Principles of Programming Languages **CO304**

3-0 -0:3 Credit : 3 Hours *Prerequisites:* CO103, CO104

Abstract: This course intends to allow the students to understand the basic concepts underlying different programming languages and their design tradeoffs. The course covers the pragmatic aspects of programming languages that requires a basic knowledge of programming language translation and runtime features such as storage allocation. These topics strengthen students' grasp of the power of computation, help students choose the most appropriate programming model and language for a given problem, and improve their design skills.

Course Outcomes:

1. Compare programming languages
2. Describe and analyze the main principles of imperative, functional, object oriented and logic oriented programming languages
3. Recite the high points of programming language history
4. Read the central formalisms used in the description of programming languages
5. Assess programming languages critically and in a scientific manner

Course Contents:

Overview of programming languages: History of programming languages, Brief survey of programming paradigms such as Procedural languages, Object-oriented languages, Functional languages, Declarative, non-algorithmic languages, and Scripting languages, Effects of scale on programming methodology, Issues as space efficiency, time efficiency, safety, and power of expression.

Virtual machines: The concept of a virtual machine, hierarchy of virtual machines, Intermediate languages.

Language translation: Comparison of interpreters and compilers, Language translation phases, Machine-dependent and machine-independent aspects of translation.

Declarations and types: The conception of types as a set of values together with a set of operations, Declaration models, Overview of type-checking, Garbage collection

Abstraction mechanisms: Procedures, functions, and iterators as abstraction mechanisms, Parameterization mechanisms (reference vs. value), activation records and storage management, type parameters and parameterized types, modules in programming languages.

Functional programming: Overview and motivation of functional languages, recursion over lists, natural numbers, trees, and other recursively-defined data, pragmatics (debugging by divide and conquer; persistency of data structures), closures and uses of functions as data (infinite sets, streams)

Logic programming: Prolog

Type systems: Data type as set of values with set of operations, Data types such as elementary types, product and co-product types, algebraic types , recursive types, arrow (function) types, and parameterized types, type-checking models, semantic models of user-defined types such as type abbreviations, abstract data types, and type equality, typed-Lamda Calculus, type inference using ML, OCAML or Haskell.

Programming language syntax and semantics: General problem of describing syntax and semantics, formal methods of describing syntax - BNF, EBNF for common programming languages features, parse trees, ambiguous grammars, attribute grammars, overview of formal semantics, informal semantics, denotational semantics, axiomatic semantics, and operational semantics.

Programming language design: General principles of language design, design goals, typing regimes, data structure models, control structure models, and abstraction mechanisms.

Exception Handling: Exception handling in various languages, programming events.

Text Books:

1. Programming Languages-Design and Implementation, TW Pratt, MV Zelkowski, TV Gopal, 4th Edition, Pearson Education India, 2006
2. Programming Languages-Principles and Practice, K. C. Louden, K.A.Lambert, 3rd Edition, Cengage Publications, 2012
3. Programming Languages- Concepts and Constructs, , R. Sethi, 2nd Edition, Pearson Education, 2006

Reference Books:

1. Fundamentals of Programming Languages, Ellios Horowitz, 2nd Edition, Galgotia Publications
2. Concepts of Programming Languages, R.W. Sebesta, 10th Edition, Pearson Education India, 2013

Cryptography		**CO318**
3-0-0: 3 Credits : 3 Hours	*Prerequisites*: CO105(DM), CO206 (DAA)	

Abstract: In today's world of the internet, the most basic objective is to ensure secure communication across an insecure channel. The art and science of establishing confidentiality, integrity and authenticity of communication is recognized as cryptography. This course is an introduction to modern cryptography. This course introduces basic concepts in cryptography from both theoretical and practical perspectives.
With the emergence of ideas such as power, battles, politics and supremacy, a wide scope of financial, legal, and social cryptographic applications has emerged, from using a credit card

on-line or sending an encrypted email, to more ambitious goals of electronic commerce, electronic voting, contract-signing, database privacy, and so on. The most important characteristic of modern cryptography is its *rigorous, scientific approach*, based on firm complexity-theoretical foundations. This course will cover how cryptography works, how security is analysed theoretically, such as symmetric-key encryption, stream ciphers, block ciphers, message authentication codes, asymmetric encryption (RSA- and discrete-log-based), and digital signatures.

Course Outcomes:

Towards the end of the course the student will be able to-

1. identify, conceptualize, and rigorously formalize the concept of secure communication
2. design and analyse security protocols including asymptotic efficiency and provable security
3. recognize and explain aspects of number theory which are relevant to cryptography
4. identify, explain, and apply cryptographic techniques like key management, digital signatures, digital certificates, and a Public-Key Infrastructure (PKI) to various disciplines in information science.

Course Contents:

Overview of cryptography. Cryptanalysis, Brief history. Classical and modern cryptography

Number Theory: gcd, divisibility, euclidean algorithm, extended euclidean algorithm, congruences, modular arithmetic, Chinese Remainder Theorem residue classes, reduced residue systems, Groups, quadratic residues, and finite fields, congruences, modular arithmetic, Computing with large numbers, Algorithms for finding gcd, primality testing and factoring.

Basic symmetric-key encryption, perfect secrecy, Shannon's definition of

perfect secrecy, Shannon's Theorem, One-time pad, stream ciphers - LFSR based stream ciphers, RC4, block ciphers – DES, AES, Different modes of operation of Block Cipher – CBC, CFB, Counter Mode, OFB

Pseudorandom Generators (PRG); Pseudo Random Functions (PRF); Pseudo Random Permutations (PRP); security against Ciphertext Only Attacks (COA); Known Plaintext Attacks (KPA), chosen plaintext attacks (CPA), chosen ciphertext attacks (CCA)

Message integrity: definition and applications, MAC, Collision resistant hashing, SHA, HMAC, Authenticated encryption: security against active attacks, session setup using a key distribution center (KDC)

Public key cryptography, Cryptography using arithmetic modulo primes, Diffie-Hellman key exchange protocol, CDH and discrete-log assumptions, Public key encryption, RSA, ElGamal encryption, limitations of RSA and ElGamal PKCs and various attacks; CCA security

Digital signatures: Digital signatures: definitions and applications, signature using RSA, Hash based signatures. certificates, certificate transparency, certificate revocation.

Security Protocols: Identification protocols, Password protocols, salts; one-time passwords (S/Key and SecurID); challenge response authentication, Authenticated key exchange and SSL/TLS session setup, HTTPs, SSH, Zero knowledge protocols,

Secure Multi-party computation, Cryptography in the age of quantum computers

Text Books:

1. Jonathan Katz and Yehuda Lindell, Introduction to Modern Cryptography, Chapman and Hall/CRC Press, 2nd edition, 2018
2. Rafael Pass and Abhi Shelat, A Course in Cryptography, 3rd edition, 2010

Reference Books:

1. O. Goldreich, Foundations of Cryptography, CRC Press (Low Priced Edition Available), Part 1 and Part 2, 1st edition, 2009
2. D. R. Stinson and M. Paterson, Cryptography: Theory and Practice, CRC Press, 4th Edition, 2018
3. Mike Rosulek. The Joy of Cryptography, 2020
4. Dan Boneh and Victor Shoup. A Graduate Course in Applied Cryptography, 2020
5. Nigel Smart. Cryptography: An Introduction, McGraw-Hill College, 2004

Information Theory and Coding **CO432**

3-0-0: 3 Credits: 3 Hours *Prerequisites:* MS105, MS205

Abstract:

This course is about how to measure, represent, and communicate information efficiently. Information Theory has been the driving force behind the revolution in digital communication and has led to various practical data compression and error correcting codes that meet the fundamental theoretical limits of performance. "Information theory", the breakthrough work of Claude Shannon and Warren Weaver, provided the basis to build the internet, digital computers and telecommunications systems. This course will study how information is measured in terms of probability and entropy, and the relationships among conditional and joint entropies; how these are used to calculate the capacity of a communication channel, with and without noise; coding schemes, including error correcting codes; how discrete channels and measures of information generalise to their continuous forms; the Fourier perspective; and extensions to wavelets, complexity, compression, and efficient coding of audio-visual information

Course Outcomes:

Towards the end of the course the student would understand how to -

1. calculate the information content of a random variable from its probability distribution
2. define channel capacities and properties using Shannon's Theorems
3. construct efficient codes for data on imperfect communication channels
4. generalise the discrete concepts to continuous signals on continuous channels
5. understand Fourier Transforms
6. describe the information resolution and compression properties of wavelets

Course Contents:

A brief review of probability, uncertainty, information, Concepts of randomness, redundancy, compressibility, noise, bandwidth, and uncertainty, Ensembles, random variables, marginal and conditional probabilities

Entropies as measures of information. Marginal entropy, joint entropy, conditional entropy, and the Chain Rule for entropy. Mutual information between ensembles of random variables.

Source coding theorem; prefix, variable-, and fixed-length codes, Noiseless coding, Huffman coding and its optimality, Kraft and McMillan's inequality, Shannon-Fano code, Elias code, Arithmetic coding and universal coding, Error correcting codes

Algebraic codes-Linear Block codes, Cyclic codes-BCH codes, perfect code, galley codes, Finite geometry codes, Hadamard codes, Maximal distance separable codes, sphere packing and singleton bounds.

Continuous information; density; noisy channel coding theorem, Fourier theorems; transform pairs. Sampling; aliasing, Gabor-Heisenberg-Weyl uncertainty relation.

Lossless compression and cryptographic codes, algebraic error correction, Kolmogorov complexity.

Text Books:

1. Raymond W. Yeung, A First Course in Information Theory, Springer, 2002
2. Thomas M. Cover and Joy A. Thomas, Elements of information theory, Wiley, 2nd Ed., 2006.
3. Simeon Ball, A Course in Algebraic Error-Correcting Codes, Springer Nature Switzerland AG, 2020

Reference Books:

1. Claude Shannon, A mathematical theory of communication, Bell systems Tech Journal. Vol. 27, pp. 379–423, 623–656, July, October 1948.
2. David MacKay, Information theory, inference, and learning algorithms, Cambridge University Press 2003.
3. V. Guruswami, A. Rudra, and M. Sudan, Essential Coding Theory, University at Buffalo 2014

Statistical Modelling and Applications **CO319**

3-0 -0: 3 Credits : 3 Hours ***Prerequisites:*** **MS205, CO105**

Abstract

In today's world, Data is constantly growing in volume, variety, uncertainty. Uncertainty in data arises due to the presence of noise that adds errors to the original values. Uncertain data is found in abundance today on the web, in sensor networks, finance, urban informatics, and business informatics, meteorology, genomics, complex physics simulations, biology. This course relates to multiple fields, and techniques from various disciplines — probability theory, computer science, optimization, statistics, and many more. Statistical model enables the compact representation and manipulation of exponentially large probability distributions. That allows them to efficiently manage the uncertainty and partial observability that commonly occur in real-world problems.

This course shows the way to construct an effective model, by learning from data that is only an approximation of our past experience. **Representation, inference, and learning** are the pillars of modeling any real-life situation. We discuss random variables, probability distribution for acquiring and **representation** of real-life data. **Inference** is about designing algorithms to be applied on the constructed model or representation to reach a desired

conclusion. With newer experiences we **learn** and improve our model so that we don't have to change our **inference** algorithms always. On the whole, we will learn to put probability distributions on real life experiences, learn them from data and do reasoning with them.

Course Outcomes:
At the end of the course, the students will be able to -
1. Understand basic notions of discrete and continuous probability.
2. Understand the philosophy behind basic methods of statistical inference, and the role that sampling distributions play in those methods.
3. perform correct and meaningful statistical analyses of simple to moderate complexity; and
4. Have a first level understanding of Monte Carlo simulation.

Course Contents:
Univariate and Multivariate Distributions
Probability mass, density, and cumulative distribution functions
Weak and Strong law of large numbers, Central Limit Theorem
Parametric families of distributions
Expected value, variance, conditional expectation
Applications of the univariate and multivariate Central Limit Theorem
Probabilistic inequalities
Markov chains

Sampling, Estimation and Model Building
- Random samples, sampling distributions of estimators
- Statistical inference: Methods of Moments and Maximum Likelihood, confidence interval and testing of hypotheses, tests of significance, chi-square tests, P-values
- Introduction to multivariate statistical models: regression and classification problems, Cluster analysis, principal components analysis
- The problem of overfitting; model assessment

Computer science and engineering applications (Case study on one or more of the following**)**
- Data mining
- Network protocols, analysis of Web traffic
- Computer security
- Software engineering
- Computer architecture, operating systems, distributed systems
- Bioinformatics
- Machine learning

Laboratory experiments:
Programming using the open-source, platform-independent programming language R or the MATLAB package.

Text Books:
1. K. Trivedi. *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. Wiley, New York, 2001.
2. M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis, Cambridge,* 2005.

3. N. Matloff. *A Course in Probabilistic and Statistical Modeling in Computer Science*. http://heather.cs.ucdavis.edu/~matloff/132/PLN
4. N. Matloff, *The Art of R Programming: A Tour of Statistical Software Design , 2011.*
5. Alan Agresti, *An Introduction to Categorical Data Analysis,* Wiley, New York, 2013
6. **Reference Books:**
7. *D. Koller & N. Friedman,* Probabilistic Graphical Models: Principles and Techniques, *MIT Press**, 2009***
8. M. Baron, *Probability and Statistics for Computer Scientists,* Chapman and Hall/CRC, 2007.

Computational Geometry	**CO521**
3-0-0: 3 Credits : 3 Hours	***Prerequisites:*** **MS205, CO105**

Abstract

Computational geometry is the study of the representation and storage of geometric data and relationships, and the design, implementation and analysis of computational algorithms that operate on geometric data to answer questions of practical interest. This course introduces students to the essentials of geometric algorithms and presents an in-depth study of the fundamental geometric structures and techniques used in this field.

Course Outcomes:

At the end of the course, the students will be able to -

1. improve problem-solving skills and algorithm design techniques using gometry.
2. Study the efficiency of geometric algorithms
3. Design and analyze algorithms for the efficient solution of numerous geometric problems that arise in other application areas such as astronomy, geographic information systems, CAD/CAM, data mining, graph drawing, graphics, medical imaging, metrology, molecular modeling, robotics, signal processing, textile layout, typography, video games, vision, VLSI
4. use and manipulate several geometric data structures
5. provide an insight into how one can recognize the inherent geometric properties of a particular application domain.

Course Contents:

Convex hull

Line segment intersection

Triangulation

Linear programming

Range search

Point location

Voronoi diagram

Arrangement and duality

Visibility graph

Well separated pair decomposition

VC-dimension, ε-approximation, and ε -nets.

Surfaces: reconstruction, surface simplification, Mesh generation

Text Books:

1. M. de Berg, M. Van Kreveld, M. Overmars, and O. Schwarzkopf, Computational Geometry: Algorithms and Applications (3rd Edition), Springer, 2008.
2. F. Preparata and M. Shamos, Computational Geometry, Springer-Verlag, 1985.
3. M. Mitzenmacher and E. Upfal. Probability and Computing: Randomized Algorithms and Probabilistic Analysis, Cambridge, 2005.

Reference Books:

1. S.L. Devadoss and J. O'Rourke, Discrete and Computational Geometry, Princeton University Press, 2011
2. J. O'Rourke, Computational Geometry in C, 2nd ed., Cambridge Univ. Press, 1998.
3. Boissonnat, J.-D.; Yvinec, M, Algorithmic geometry - Cambridge University Press , 1997.
4. K. Mulmuley, Computational Geometry: An Introduction Through Randomized Algorithms, Prentice Hall, 1994.

Web Technology **CO423**

3- 0-1: 4 Credits : 5 Hours Prerequisites: CO103, CO104

Abstract: This course provides a basic overview and understanding of many key Web technologies. It starts with understanding the basics of the Internet, thereafter, followed by delving into the underlying technologies such as HTTP, FTP, and middleware such as CORBA, DCOM etc. that supports the smooth functioning of any Internet applications. Alongside, fundamentals on web programming such as HTML, XHTML, Cascading Style Sheets (CSS), Extensible Markup Language (XML), Extensible Stylesheet Language (XSL), XPATH, XSLT are also covered, and describes how scripting, such as JavaScript, jQuery, and AJAX, works in dynamic websites. The course also discusses browser and server architectures, Web and application servers, Hypertext Preprocessor (PHP), Python, Common Gateway Interface (CGI) and Web security.

Course Outcomes:

Towards the end of the course the student would -

1. Understand key Internet technologies supporting the Internet applications.
2. Implement interactive web page(s) using HTML, CSS and JavaScript.
3. Implement dynamic web page(s) using AJAX, jQuery, JavaScript, PHP etc. and
4. database connectivity.
5. Describe and differentiate different Web Extensions and Web Services.
6. Implement at least one web security mechanism using PHP, JavaScript, or Python.

Course Contents:

Basics Of Internet: Computer network, Network connectors, Network software, LAN,CAN, MAN, WAN, Introduction to Internet, World Wide Web (WWW), Internet Protocols, Web browser, Web server, Internet services, Web services, DNS, Virtual hosting.

Internet Details: Internet topology, Virtual High Speed Backbone Network Service (vBNS), Network Access Providers (NAS), Internet Service Provider (ISP) and architecture, Internet communication protocols such as HTTP and its different versions, FTP, SMTP, POP, MIME, Email Privacy such as Pretty Good Privacy (PGP), and Privacy Enhanced Email (PEM).

Client/Server Computing: What is C/S Computing, Fat client VS Fat Servers, N-tiered Software Architecture, Middleware, Distributed Object Models such Common Object Requests Broker Architecture (CORBA), Distributed Component Object Model (DCOM), Java Remote Method Invocation (JRMI) and Enterprise Java Bean (EJB).

Markup Languages And Their Grammars: SGML, DTD Resource; HTML, XHTML, CSS, XML, XSL, Query Languages for XML, XML validator script.

Client-Side programming: HTML,CSS, AJAX, JavaScript, jQuery

Server-Side programming: PHP, Python, CGI.

Overview of Java, JAVA Applet, JAVA Servlets, JSP.

Web Service: JAX-RPC-Concepts-Writing a Java Web Service-Writing a Java Web Service Client-Describing Web Services: WSDL- Representing Data Types: XML Schema-Communicating Object Data: SOAP Related Technologies-Software Installation-Storing Java Objects as Files-Databases and Java Servlets.

Web Browser: Browser Architecture, Document Object Model (DOM), DOM Tree, Render Tree, Rendering Engine and its use on various browsers, case study on browser architectures of Mozilla Firefox, Google Chrome, and IE.

Web Server Apache Architecture: Web Server Architecture, Server Features, Configuration of Apache and IIS.

ASP and JSP Search Engines; Web Database Connectivity.

Interface to database: CGI, JDBC

Web Security: Web security threats, Firewalls, Proxy Servers, Cryptography, Digital Signature, Digital Certificates, Secure Socket Layer (SSL), S-HTTP, Secure Electronic Transaction (SET), 3D Secure Protocol.

Practical Topics:

1. Implement a basic static web page(s) using HTML and CSS, thereafter, converting it into an interactive web page(s) using JavaScript.
2. Implement dynamic web page(s) using AJAX, jQuery, JavaScript, PHP etc. and
3. database connectivity.
4. Implement client and server-side scripting for creating dynamic web pages(s).
5. Implement at least one web security mechanism using PHP, JavaScript, or Python.

Text Books:

1. Web Technologies: A Computer Science Perspective by Jaffrey C. Jackson, Prentice Hall, 2007
2. Web Technologies: TCP/IP to Internet Application Architecture by Achyut S. Godbole and Atul Kahate, Tata McGraw-Hill Education, 2003, 5th Reprint 2006

Reference Books:

1. Dynamic Web Publishing Unleashed, Shelly Powers et al., 2nd Edition, Sams, 1997
2. Java 1.2 Unleashed, Jamie Jaworski, 4th Edition, Sams, 1998
3. CGI by Example, Jeffry Dwight et.al., Que, 1996.
4. Using Active Server Pages, Scot Johnson et.al., Que, 1997

Embedded Systems **CO306**

3-0-1: 4 Credits : 5 Hours Prerequisite: CO214(CAO), CO309(OS)

Abstract:
This course is an introduction to embedded system technology and provides hands on experience in embedded system design and implementation using the ARM processor/8051 microcontroller. It provides basic knowledge on embedded system design issues and real time systems. The course encourages students to explore the scope of embedded systems in everyday life and come up with innovative ideas. It includes lectures, laboratory work and a group project.

Course Outcomes:

Towards the end of the course the student would …

1. be familiar with the basic technology behind embedded computing systems
2. be well conversant with Embedded Systems and ES design issues
3. be familiar with Real Time Systems and design issues
4. be able to design, simulate and develop intelligent products and systems

Course Contents:

Introduction to embedded systems*:* Why ES? ES classification and cores, Difference between GPC and ES, ES applications, Components and characteristics of ES, Examples of ES, ES Design metrics, Typical ES architecture, ES Technology, Future & scope of ES.

ES processor design*:* Categories of ES processor, Processor technologies, Processor architecture, SPP and GPP design, Advantages and disadvantages of using SPPs and GPPs in an ES.

ES Peripherals*:* Standard SPP or peripherals, Peripheral categorization and classification, User peripherals- GPIO, Timers & Counters, Watchdog Timers, RTC, UART, PWM, LCD controllers, Keypad controllers, ADC, Stepper Motor controllers.

Memory*:* Classification and technologies-RAM (SRAM, DRAM, PSRAM, NVRAM) & ROM (mask-PROM, OTP-ROM, EPROM, EEPROM, OTP-EPROM, UV-EPROM), Flash, SD-MMC, Advanced RAM, Memory management-cache memory, Cache mapping, Software overlays, Virtual memory.

Interfacing*:* Basic Terminology, ISA Bus Protocol, Basic protocol concepts, Microprocessor Interfacing: I/O Addressing (port based, bus based), Interrupts, DMA and DMA controller, Programmable priority controller, Bus Arbitration, Advanced communication principles, Communication protocols: Serial, parallel, wireless.

Design and implementation of an ES: Embedded system design, SDLC, Introduction to a simple Digital Camera as an example ES, Basic functions, Designer's perspective, Various Design implementations, Comparison of such implementations.

Sensors and their use in ES: Role of sensors in ES, Sensor interfacing and signal conditioning, Sensor classification, Criteria for choosing a sensor, Types of sensors (temperature sensor, Ultrasonic sensor, LDRs, IR sensor, etc.) and their working principles.

Programmable System Peripherals*:* AHB and VPB, Programming peripherals in ES, System peripherals: On chip Flash, SRAM memory, External bus interface, PLL, VLSI peripheral bus divider, Power Control, Interrupt systems.

Low Power Computing: What is LPC? Motivation for LPC especially in ES, Power dissipation sources, Static and dynamic power dissipation, Power modelling and estimation, Voltage/frequency scaling, Power reduction in FSM, Power reduction in datapath, Low power memory optimization.

Embedded Software Architectures: Simple and complex architectures, Round Robin, Round Robin with interrupt, Function Queue scheduling architecture, RTOS architecture, Selecting an architecture.

Real Time Systems and RT Operating System*:* What is a RT system? RT system applications, Basic model, Characteristics, Classification of RT tasks, Tasks and task states, What is RTOS?, RTOS vs OS, RTOS features, RTOS scheduler, Shared data problem, Semaphores and Shared Data, Re-entrant functions, Priority Inversion problem, Scheduling algorithms in RTOS.

8051 Microcontroller/ARM Processor: Introduction to the Architecture, Addressing modes, brief overview of the instruction set, I/O port programming (I/O ports and their functions, bit addressability), Timer programming (timer modes, timer as counter), Serial Port Programming, Interrupts, 8255 PPI, Interfacing to common peripherals.

Practical topics (Demonstration of C/Assembly language programs using microC/Keil software for the 8051μC/ ARM-7 processor**):**
(i) LED on/off control via I/O ports
(ii) Blinking of LEDs in ascending and descending order
(iii) LED control via push buttons
(iv) Display of numbers on 7-segment LED
(v) Displaying data from push buttons on 7-segment LED
(vi) Display of 2-digit number using two 7-segment LEDs
(vii) Interfacing a keypad to AT89S52 microcontroller
(viii) Display message on LCD
(ix) Interfacing ADC to AT89S52 microcontroller
(x) Display of ADC data on PC hyperterminal
(xi) Display temperature from temperature sensor on LCD
(xii) Control of DC motor via microcontroller

Mini Project (Samples):
Display of real time on LCD, Traffic Light System, Automatic turn-on/off of fan and light, automatic control of street lights, object detection, temperature measurement and display, automatic watering of plants, water tank overflow control system, etc.

Text Books:

1. Embedded System Design: a unified hardware/software introduction, Frank Vahid and Tony Givargis, Wiley, 3rd edition, 2006.
2. An Embedded Software Primer, David E. Simon, Addison-Wesley Professional, 13th printing, 2004.
3. The 8051 Microcontroller and Embedded Systems, Mazidi, Mazidi, McKinlay, Pearson Education Ltd., 2014.

Reference Books:

1. Computers and Components, Wayne Wolf, Elsevier, 3rd edition, 2012.
2. The 8051 Microcontroller based Embedded Systems, Manish K Patel, Mc Graw Hill, 1st edition, 2014.
3. The Insider's Guide to the Philips ARM7-Based Microcontrollers, T. Martin, Hitex (UK) Ltd., 1st edition, 2005.
4. Embedded and Real Time Operating Systems, K C Wang, Springer, 1st edition, 2017.
5. Embedded Software Development with C, Kai Qian, David den Haring and Li Cao, Springer, 1st edition, 2009.

Advanced Computer Architecture — CO426

3-0-1: 4 Credits : 5 Hours — Prerequisite: CAO

Abstract: CO426 is an advanced level course for undergraduate students in the field of computer architecture at Tezpur University. This course is designed to teach the students about the state-of-the-art in computer architectures and its current trends. Students will gain the knowledge of different architectural developments to improve the efficiency of a computer system and develop the skill to measure the performance of a computer system. They will understand the performance bottlenecks in typical Von-Neumann based architecture and will be familiar with how to exploit different kinds of parallelism possible in the underlying execution model. Students will learn what are the architectural features used in Multiprocessor and Multicore based systems and associated synchronization and Cache Coherence Issues. This will make the students understand how memory hierarchy design can influence the performance of a computer system, what are the different design trade-offs in Cache Memory and Virtual Memory design. Furthermore, they will be familiar with architectures used in Mobile devices, Cloud, and high-performance computing devices. Students will be trained to develop parallel programming skills with OpenMP and CPU-GPU combined environments.

Course Outcomes:

At the end of the course, a student will be able to –

1. Understand the state-of-the-art in computer architectures and processing
2. Evaluate the performance of computer Systems
3. Determine the key architectural elements to improve the performance of a computer system
4. Understand the Instruction Level Parallelism and the techniques to handle different types of data and control hazards
5. Design Memory Hierarchy for improving performance of computer system
6. Understand the Vector Processor and GPU Architecture for exploiting parallelism

7. Understand the architectures used in mobile, cloud and high-performance computing devices
8. Write programs for Parallel Processors and CPU-GPU environment

Course Contents:

Introduction: Defining Computer Architecture, Flynn's Classification of Computers, Metrics for Performance Measurement, Parallel programming, and performance issues.

Instruction Level Parallelism Instruction-level Parallelism: Concepts and Challenges, loop unrolling, VLIW and superscalar processors, Basic Compiler Techniques for Exposing ILP, Reducing Branch Costs with Branch Prediction, Dynamic Scheduling, Advanced Techniques for Instruction Delivery and Speculation, Limitations of ILP, Multithreading: Exploiting Thread-Level Parallelism to Improve Uniprocessor Throughput, Case Study: Dynamic Scheduling in Intel Core i7 and ARM Cortex-A8.

Memory Hierarchy Introduction, Design of Memory Hierarchy, Optimizations of Cache Performance, Memory Technology and Optimizations, Virtual Memory and Virtual Machines, Case Study: Memory Hierarchies in Intel Core i7 and ARM Cortex-A8.

Thread Level Parallelism - Introduction, Shared-Memory Multicore Systems, Performance Metrics for Shared-Memory Multicore Systems, Cache Coherence Protocols, cache coherence on snoopy buses, Scalable multiprocessors, Directory-based cache coherence, Synchronization, Memory Consistency, Multithreaded Programming using OpenMP, Case Study: Intel Skylake and IBM Power8.

Data Level Parallelism Introduction, Vector Architecture, SIMD Instruction Set Extensions for Multimedia, Graphics Processing Units, GPU Memory Hierarchy, Detecting and Enhancing Loop- Level Parallelism, CUDA Programming, Case Study: Nvidia Maxwell.

Laboratory:

Designing parallel version of basic algorithms,

Multithreaded Programming using OpenMP,

CPU-GPU programming, CUDA Programming

Text Books:

1. J.L. Hennessy and D.A. Patterson. Computer Architecture: A Quantitative Approach. 5th Edition, Morgan Kauffmann Publishers, 2012.

References:

1. J.P. Shen and M.H. Lipasti. Modern Processor Design: Fundamentals of Superscalar Processors. McGraw-Hill Publishers, 2005.
2. K. Huang, Naresh Jotwani, Advanced Computer Architecture: Parallelism, Scalability, Programmability, 3e, 2015,
3. V. Kumar et el. An Introduction to Parallel Computing: Design and Analysis of Algorithms, 2e, Pearson, 2004.
4. D.B. Kirk and W.W. Hwu. Programming Massively Parallel Processors. 2e, Morgan Kauffmann Publishers, 2012.
5. OpenMP programming (https://www.openmp.org)
6. CUDA programming (https://developer.nvidia.com/cuda-zone)

Theory of Computation **CO422**

3-1- 0: 4 Credits : 4 Hours *Prerequisites:* CO105, CO216

Abstract:
When we engage in solving any problem, the first two questions that comes to our mind:
Is it possible using a given machine? (*computability*)

- How much time and space are required to solve it? (*complexity*)
- The course CO422 explores these questions about the limits of abstract models of computing machines and reasoning about what they can and cannot compute efficiently. This course provides an exciting introduction to some of the fundamental ideas of theoretical computer science. It attempts to present a vision of "computer science beyond computers": that is, CS as a set of mathematical tools for understanding complex systems such as universes and minds.

Course Outcomes:

Towards the end of the course the student will be able to:

1. Model, compare and analyse different computational models.
2. discuss key notions of computation, such as algorithm, computability, decidability, reducibility, and complexity, through problem solving.
3. Construct algorithms for different problems and argue formally about correctness on different restricted machine models of computation.
4. explain the models of computation, including formal languages, grammars and automata, and their connections.
5. Identify limitations of some computational models and possible methods of proving them.
6. to introspect how the theoretical study in this course is applicable to various engineering applications including compiler design. artificial intelligence

Course Contents:

Review of formal languages, grammar, and automata theory.

Turing machines, Church-Turing thesis, Properties of Recursive & Recursively Enumerable Languages, Turing computable functions, Random Access Machines

Halting problem, Decidability, Reducibility, Recursion theorem, Godel's incompleteness theorem, Universal TM, Rice's Theorem, Post's Correspondence Problem;

Time Complexity: Complexity classes P, NP, and NP-completeness, the Cook-Levin Theorem, P versus NP problem and why it's hard

Space Complexity: Savitch's Theorem, PSPACE, NL, and EXP.

Intractability, Time and Space Hierarchy theorems.

Introduction to emerging models of computation: Quantum computing; Probabilistic and randomized computation etc.

Text Books:

1. Michael Sipser, Introduction to the Theory of Computation Cengage, 2014
2. Lewis & Papadimitriou, Elements of the Theory of Computation, Pearson Education.

Reference Books:

1. Moore, Cristopher, and Stephan Mertens. The Nature of Computation. Oxford University Press, 2011
2. Arora, Sanjeev, and Boaz Barak. Computational Complexity: A Modern Approach. Cambridge University Press, 2009
3. Apostolos Doxiadis, Christos Papadimitriou and Alecos Papadatos, Logicomix: An epic search for truth, Bloomsbury, 2009

Artificial Intelligence	**CO401**
3-0-0: 3 Credits : 3 Hours	*Prerequisites:CO210*

Abstract:

This course is aimed at providing an overview of the various aspects and behaviours of intelligent systems. It covers the various techniques that have been devised to emulate intelligent behaviour in artefacts. The course discusses ideas related to perception, reasoning, learning, acting and communicating in a complex environment.

Course Outcomes:

1. Getting students to appreciate the foundations of Artificial Intelligence and its future trends
2. Understanding of the basic elements constituting problems in an intelligent system and various approaches of solving them
3. Understanding the notions of uncertainty and decision making in real world problems
4. Introduction to learning mechanisms by which an intelligent system can improve its behaviour

Course contents:

Introduction to AI, background, related field

Uninformed search strategies: Breadth First Search, Depth First Search, Depth Limited Search, Iterative Deepening Depth First Search, Bidirectional Search and comparisons, Hill Climbing, Simulated Annealing

Informed Search Strategies: Heuristic Functions Significance of heuristic Functions, Desirable Properties, Design of Heuristic Functions Greedy Best First Search, A* Search,

Constraint Satisfaction Problems (CSP): Backtracking Search

Adversarial Search for Game Playing: Minimax Algorithm, Alpha-Beta Pruning, Making Real-Time Decisions

Knowledge Representation and Reasoning: Propositional Logic and Inference, Resolution principle, First Order Logic and Inference

Planning: Planning with State Space Search, Planning Graphs, STRIPS, Planning in the Real world, POP

Reasoning Under Uncertainties: Bayesian Networks, Exact and Approximate Inference, Expressing vagueness and ignorance

Markov and Sequential Decision Problems: Definition, Optimality in SDP, Algorithms for optimal Policies (Value Iteration, Policy Iteration)

Overview of Machine Learning: Goals and forms of Learning, Performance Assessment, Decision Theory, Inductive Learning, Concept Learning, Statistical Learning Methods, Introduction to Neural Networks, Genetic Algorithms, Reinforcement Learning.

Selected applications

Text Books

1. Rusell, S., and Norvig, P., *Artificial Intelligence: A Modern Approach*, 4th Ed, Pearson, 2020.

Reference Books

1. Koller, D., and Freidman N., *Probabilistic Graphical Models*, MIT Press, 2009
2. Mitchell, T., *Machine Learning*, Tata-McGraw-Hill, 1997.
3. Bishop, C. M., *Pattern Recognition and Machine Learning*, 2nd edition, Springer, 2006.

Industrial Summer Training	**CO404**
0-0-1: 1 Credit: 2 Hours	*Prerequisites:* - CO311

Course objectives:

- To expose students to the 'real' working environment outside the academic life of the university and get acquainted with the organization structure, business operations and administrative functions.
- To have hands-on experience in software development/research so that they can relate and reinforce what has been taught at the university.
- To promote cooperation and to develop synergetic collaboration between industry and the university in promoting a knowledgeable society.
- To set the stage for future recruitment by potential employers.

Training will be of 12 weeks duration carried out during the summer break after the 6thsemester. The students will submit their reports in the 7th semester.

Course Outcomes:

After completion of the training, the student will:

1. gain valuable practical experience,
2. able to define their own career interests.
3. develop the students' job-related skills.
4. enhance the students' computer science engineering knowledge acquired in class through field experience.
5. be able to deal with the societal problems outside the university.

Project II **CO402**

0-0-4: 4 Credits: 08 Hours *Prerequisites:* CO311

The students will carry out project works in groups of 2 or 3 students each under the guidance of a faculty member. The project shall consist of research/ design/ development/ implementation work.

Course Outcomes:
At the end of the project the student will be able to:
1. Conduct a background study of the topic of interest
2. Practically apply the basic computer science and engineering knowledge acquired
3. Communicate and present their work orally and in written.
4. Conceive, design, and implement projects using the inherent tools of engineering.
5. Organize, plan, and distribute the project-related works amongst the members of a group in a coherent manner.

Project III **CO403**

0-0-8: 08 Credits: 16 Hours *Prerequisites:* CO402

The students will carry out project works in groups of 2 or 3 students each under the guidance of a faculty member. The project shall consist of research/ design/ development/ implementation work. It may also be a continuation of the Project II work.

Course Outcomes:
At the end of the project the student will be able to:
1. Conduct a background study of the topic of interest
2. Practically apply the basic computer science and engineering knowledge acquired
3. Communicate and present their work orally and in written.
4. Conceive, design, and implement projects using the inherent tools of engineering.
5. Organize, plan, and distribute the project-related works amongst the members of a group in a coherent manner.

Fundamentals of Speech Processing **CO513**

3 - 0 - 1: 4 Credits: 5 Hours *Prerequisites:* Signals and Systems (EL204)

Abstract:
This course is an introduction to Speech Processing technology and presents a comprehensive overview of digital speech processing that ranges from the basic nature of a speech signal through a variety of methods of representing speech in digital form, to applications in voice communication and automatic synthesis and recognition of speech. It provides a useful introduction to the wide range of important concepts such as speech signal representation, feature extraction, speech models, etc., that comprise the field of digital speech processing. The course also presents a brief overview of various speech applications areas such as speech

recognition, speaker identification, speech synthesis, speech enhancements and recent topics of research in speech technology.

Course Outcomes:

Towards the end of the course the student would …

1. be able to understand the mechanism of human speech production system and clearly describe how speech sounds are produced.
2. be well conversant with the basic techniques used in the processing of a speech signal.
3. be well conversant with a range of commonly used feature extraction techniques as well as speech modeling techniques.
4. be able to use Matlab tools to analyse and model speech data so as to design speech systems.
5. be able to design and develop simple speech recognition, speaker identification, speech synthesis systems.
6. be motivated to explore research areas in speech processing technology.

Course Contents:

Introduction to Speech Processing: Speech production and perception, linguistic aspect of speech, acoustic and articulatory phonetics, IPA, nature of speech, models for speech analysis and perception.

Analysis of speech: Sampling, quantization, aliasing effects, filtering, frequency and time domain based methods, FFT, computation of pitch, spectrograms, formant analysis, LP analysis, cepstrum analysis.

Modeling techniques for developing speech systems: Vector Quantization (VQ), Hidden Markov Models (HMM), Gaussian Mixture Models (GMM), Support Vector Machines (SVM), Artificial Neural Networks (ANN) and Deep Neural Networks (DNN).

Speech Systems: *Speech Recognition:* Issues in speech recognition, isolated/connected word recognition, continuous speech recognition, large vocabulary continuous speech recognition. *Speech Synthesis:* Issues in speech synthesis, types of speech synthesis: Unit-selection, Statistical-parametric and Deep-learning based TTS systems. *Speaker Recognition:* Issues in speaker recognition, speaker verification vs identification, text-dependent vs text-independent speaker recognition, development of speaker recognition systems.

Practical Topics:

1. Introduction to speech analysis tools
2. Speech signal to symbol transformation (transcription)
3. Study of speech production mechanism
4. Extraction of pitch and formant frequencies from the speech signal
5. Study of quantization and aliasing effects
6. Formant analysis of vowels
7. Synthesis of vowels such as /a/, /i/ and /e/
8. Study issues of short term spectral analysis of speech signals
9. Linear prediction analysis of speech
10. Speech classification problem.

Text Books:

1. L.R. Rabiner and R.W. Schafer, “Digital Processing of Speech Signals”, Pearson Education, Delhi, India, 2004.

2. L. R. Rabiner, B. H. Jhuang and B. Yegnanarayana, “Fundamentals of speech recognition”, Pearson Education, 2009.
3. D. O’Shaughnessy, “Speech Communication: Human and Machine”, 2nd edition, IEEE Press, NY, USA, 1999.

Reference Books:

1. T. F. Quatieri, “Discrete time processing of speech signals”, Pearson Education, 2005.
2. J. Benesty, M. M. Sondhi and Y. Huang, “Springer Handbook on Speech Processing”, Springer publishers, 2008.
3. X. Huang, A. Acero and H. W. Hon, R foreword by Reddy , “Spoken Language Processing: A Guide to Theory, Algorithm and System Development”, Prentice-Hall PTR, 2001.
4. S. Furui and M. Sondhi, “Advances in speech signal processing”, CRC Press, 1991.
5. S. Sen, A. Dutta and N. Dey, “Audio Processing and Speech Recognition: Concepts, Techniques and Research Overviews”, Springer, 2019.
6. L.Mary, “Extraction of Prosody for Automatic Speaker, Language, Emotion and Speech Recognition”, Springer, 2019.
7. Ian Mcloughlin, “Applied Speech and Audio Processing, With MATLAB Examples”, Cambridge University Press, 2009.

Virtual and Augmented Reality — CO517

3-0-1: 4 Credits : 5 Hours — *Prerequisites:* CO303(CG)

Abstract:

This course is aimed at providing an overview of the various aspects and applications of Virtual and Augmented Reality systems. It covers 3D Computer Graphics and basics concepts of Computer Vision as essential tools in the development of Virtual Reality Systems. The course discusses related technologies necessary in the Virtual and Augmented Reality systems and their evaluation.

Course Outcomes:

After completion of course, students would appreciate and be conversant with:

1. Concepts of the topics in VR and AR systems design involving the foundations of VR and AR systems and the current trends
2. Understanding of the basic components and peripheral systems in VR and AR System
3. Understanding of the various stages in the VR and AR System pipeline
4. Knowledge of practical and developmental aspects of VR applications

Course Contents:

Unit 1: Introduction to Virtual and Augmented Reality
Review of Computer Graphics and Vision in VR

Unit 2: AR/VR Peripheral and Programming foundation
Introduction to basic architecture and functioning of head-mounted device and other peripherals
Art of AR/VR programming using OpenGL/C++/Unity

Unit 3: 3D Computer Graphics[LAV]
Geometry and Transformations
Perception and Rendering
Motion in Real and Virtual worlds
Tracking technology.

Unit 4: Virtual Reality Systems [LAV]
Locomotion
Manipulation
Interaction
Cinematic VR, Spatial Sound

Unit 5: Augmented Reality Systems [SCH]
Registration
Coherence
Visualization
Unit 6: Evaluation of VR/AR Systems

Laboratory Work:

Laboratory work will involve

- Assignments on 3D Graphics
- Mini projects on AR/VR applications for desktop, hand-held, and head-worn displays
- Applications can be deployed on handheld Android and iOS devices with cameras (smartphones and tablets) or head-worn displays.

Text Books and References:

1. LaValle "Virtual Reality", Cambridge University Press, 2016**[LAV]**
2. D. Schmalstieg and T. Höllerer. Augmented Reality: Principles and Practice. Addison-Wesley, Boston, 2016, ISBN-13 978-0-32-188357-5**[SCH]**
3. Reference Books
4. J. LaViola Jr., E. Kruijff, R. McMahan, D. Bowman, and I. Poupyrev. 3D User Interfaces: Theory and Practice, 2nd Edition. Addison-Wesley, Boston, 2017, ISBN-13 978-0-13-403432-4 **[3DUI]**
5. Marschner, Shirley "Fundamentals of Computer Graphics", 4th Edition, CRC Press 2016 **[MAR]**

***