Core Courses

(CT-22001) Compiler Construction

Teaching Scheme:
Lectures: 3 Hrs/week

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

Course Outcomes:

Students will be able to:

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

Introduction: The role of language translation in the programming process, Translator Issues, Types of translators, Design of an assembler Introduction to phases of compilation, Front end and Back end model of compiler, Cross compiler, Incremental compiler.

[6 Hrs]

Lexical Analysis: Concept of Lexical Analysis, Regular Expressions, Tokens, Lexemes, and Patterns, Block Diagram of Lexical analyser, Revision of finite automata, Introduction to LEX Tool and LEX file specification, Error detection and recovery in LEX.

[6 Hrs]

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

[8 Hrs]

Semantic Analysis & Intermediate Representation: Need of semantic analysis, Abstract Parse trees, Syntax directed definitions, Syntax directed translation schemes, Symbol Tables, Symbol Table management, Data type as set of values and with set of operations Type Checking, Intermediate code generation: Intermediate languages, Design issues, Intermediate code representations: three address code, abstract & concrete syntax trees,.

[8 Hrs]

Code Optimization: Introduction, Principal sources of optimization, Machine Independent

Optimization, Machine dependent Optimization, Various Optimizations: Function preserving transformation, Common Sub-expressions, Copy propagation, Dead-code elimination, Loop Optimizations, Code Motion, Induction variables and strength reduction, Peephole Optimization, Redundant –instruction elimination.

[6 Hrs]

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

[6 Hrs]

Text Books:

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

Reference Books:

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

(CT-22005) Compiler Construction Laboratory

Teaching Scheme:
Laboratory: 2 Hrs/ Week

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

Course Outcomes:

Students will be able to:

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

Suggested List of Assignments:

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

(CT-22003) Cryptography and Network Security

Teaching Scheme	**Examination Scheme**
Lectures: 3 Hrs/ Week	Assignment/Quizzes : 40 marks
	End Semester Exam: 60 marks

Course Outcomes

Students will be able to:

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

Course Contents

Introduction: Cryptography and modern cryptography, Need of security, Security services, Basic network security terminology, Security attacks, Classical cryptosystems and their cryptanalysis, Operational model of network security

[4 Hrs]

Mathematical Foundations: Prime Number, relatively prime numbers, Modular Arithmetic, Fermat's and Euler's Theorem, The Euclidean and Extended Euclidean Algorithms, The Chinese Remainder Theorem, Discrete logarithms

[6 Hrs]

Symmetric Key Ciphers: Symmetric Key Ciphers, Feistel Networks, Modern Block Ciphers, Modes of Operation, Cryptanalysis of Symmetric Key Ciphers: Linear Cryptanalysis, Differential Cryptanalysis