| Regulations 2022 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Department of Information Technology | | | | | | | | |
| Course Code | Course Title | Hours / Weeks | | | Credits | Maximum Marks | | |
| | | L | T | P | C | CA | EA | Total |
| **422ITT06** | **COMPILER ENGINEERING** | **3** | **0** | **0** | **3** | **40** | **60** | **100** |

**OBJECTIVE(S):**

- To learn about automata theory and regular expressions.
- To learn to design and implement a lexical analyzer.
- To learn the role of a parser and to study the different ways of parsing tokens.
- To study the process of Intermediate Code generation and its representations.
- To study the concepts of machine code generation.
- To study the concepts of Code Optimization

**PREREQUISITES: Nil**

**UNIT-I INTRODUCTION TO AUTOMATA THEORY AND REGULAR EXPRESSIONS 9**

Finite Automata – Deterministic Finite Automata – Non-deterministic Finite Automata – NFA to DFA – Finite Automata with Epsilon Transitions – Epsilon-NFA to DFA – Kleene's Theorem – Minimization of Automata – Regular Expressions – Equivalence between Regular Expression and Automata – Properties of Regular Expressions.

**UNIT-II LEXICAL ANALYSIS 7**

The Structure of a Compiler – Evolution of Programming Languages – Application of Compiler Technology – Lexical Analysis – Role of Lexical Analyzer **– Input Buffering** - Specification and Recognition of Tokens – Lexical Analyzer Generators.

**UNIT-III SYNTAX ANALYSIS 11**

Introduction – Context Free Grammar – **Writing a grammar** - Top Down Parsing – Recursive Descent Parsing – Predictive Parsing – NonRecursive Predictive Parsing – Error Recovery – Bottom Up Parsing – LR Parsers – Construction of SLR (1) Parsing Table, Canonical LR (1) Parsing Table and LALR (1) Parsing Table – Parser Generators.

**UNIT-IV INTERMEDIATE CODE GENERATION 9**

Symbol Table – Construction – Syntax Directed Definitions – Evaluation Orders for Syntax Directed Definitions – Applications of Syntax Directed Translation – Intermediate Code Generation – Three Address Code – Types and Declarations – Expression Translation – Type Checking **– Control Flow**- Back Patching.

**UNIT-V CODE GENERATION AND OPTIMIZATION 9**

Issues – Design of Code Generator – Addresses in the Target Code – Basic Blocks in Flow Graph – Simple Code Generator – Peephole Optimization – Machine Independent Optimization – Principal Sources of Optimizations – Bootstrapping a Compiler – Compiling Compilers – Full Bootstrap.

Total Hours: 45

**OUTCOMES:**

Upon completion of the course, the students will be able to:
CO1: Understand the construction of deterministic and nondeterministic automata.
CO2: Understand the concept of lexical analysis and various phases of a compiler
CO3: Apply different parsing algorithms to develop the parsers for a given grammar.
CO4: Represent the intermediate code for the source languages
CO5: Design and analyze code generation schemes and various optimization techniques.

Mapping of CO's with PO and PSO

| CO/PO | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 | PS01 | PS02 | PSO3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C01 | 3 | 2 | 2 | 3 | 3 | - | - | - | - | - | - | 3 | 1 | - | - |
| C02 | 3 | 3 | 2 | 3 | - | - | - | - | - | - | - | 3 | 1 | 3 | - |
| C03 | 3 | 3 | 2 | 3 | 3 | - | - | - | - | - | -- | 3 | - | 2 | 3 |
| C04 | 3 | 2 | 3 | 3 | - | - | - | - | - | - | - | 3 | - | 2 | 3 |
| C05 | 2 | 2 | 3 | 3 | 3 | - | - | - | - | - | - | 3 | - | 2 | 3 |
| AVG | 3 | 2 | 2 | 3 | 3 | - | - | - | - | - | - | 3 | 1 | 2 | 3 |

**TEXT BOOKS:**

1. Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman, "Compilers: Principles, Techniques, and Tools", Second Edition, Pearson Education, 2023.
2. John Hopcroft, Rajeev Motwani, Jeffrey Ullman, "Introduction To Automata Theory Languages, and Computation", Third Edition, Pearson Education, 2021.

**REFERENCES:**

1.Dhamdhere D M, "Compiler Construction Principles and Practice" Second edition, Macmillan India Ltd., New Delhi, 2005.
2. Torbengidius Mogensen, "Basics of Compiler Design", Springer, 2011.
3. Charles N, Ron K Cytron, Richard J LeBlanc Jr., "Crafting a Complier", Pearson Education, 2010.
4. K. D. Cooper, L. Torczon, "Engineering a Compiler", Morgan-Kaufmann, Second Edition, 2011.
5. Micheal Sipser, "Introduction to the Theory of Computation", Third Edition,2014.