# FACULTY OF ENGINEERING & TECHNOLOGY

## Effective from Academic Batch: 2020-21

**Programme:**      **Bachelor of Engineering (Computer Engineering)**

**Semester:**      **VII**

**Course Code:**      **102046704**

**Course Title:**      **Compiler Design**

**Course Group:**      **Professional Core Course**

**Course Objectives:** The principles and techniques of compiler design are so pervasive that the concepts covered in this subject will be useful during the career of a computer scientist. Compiler construction requires the understanding of programming languages, machine architecture, language theory and algorithms. The course focuses on compiler writing techniques that can be used to construct translators for a wide variety of languages.

**Teaching & Examination Scheme:**

| Contact hours per week | | | Course Credits | Examination Marks (Maximum / Passing) | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | Theory | | J/V/P* | | Total |
| Lecture | Tutorial | Practical | | Internal | External | Internal | External | |
| 3 | 0 | 2 | 4 | 40 / 14 | 60 / 21 | 20 / 7 | 30 / 10 | 150 / 52 |

\* **J**: Jury; **V**: Viva; **P**: Practical

**Detailed Syllabus:**

| Sr. | Contents | Hours |
|---|---|---|
| 1 | **Basics of the Compiler:** <br> Analysis-Synthesis model of Compilation, Phases of a compiler, Cousins of Compiler, Front-end and back-end of compiler, Pass structure Compiler construction tools. | 03 |
| 2 | **Regular Languages and Finite Automata:** <br> Strings and Languages, Operations on Languages, Regular Expressions, Regular Definitions, Nonregular Sets, Finite Automata, From a regular expression to an NFA, NFA to DFA Conversion, NFA with e-transitions | 05 |
| 3 | **Lexical Analysis:** <br> The Role of the Lexical Analyzer, Specification of Tokens, Recognition of Tokens, Input Buffering, elementary scanner design and its implementation (Lex), Applying concepts of Finite Automata for recognition of tokens, Design of a lexical analyser generator, Optimization of DFA based pattern matchers | 05 |

| 4 | **Context Free Grammars:**<br>Definition, derivations, Parse tree , ambiguity in CFG, Normal forms  for CFG, Chomsky classification for Grammar, Regular expressions vs CFGs, Eliminating ambiguity, eliminating left recursion, left factoring, Non CFL constructs | 07 |
|---|---|---|
| 5 | **Syntax Analysis:**<br>Role of parser, syntax error handling<br>Top Down Parsers:<br>Predictive parsers, Non recursive predictive parsers, Construction of predictive parsing tables, Error recovery in predictive parsing<br>Bottom Up parsers:<br>Handles, Handle pruning, Shift Reduce parsing, Viable prefixes, Conflicts during SR parsing<br>Operator-Precedence Parsing, Error recovery in operator precedence parsing, LR Parsers, Using Ambiguous Grammars, Parser Generators, Automatic Generation of Parsers.<br>Syntax-Directed Definitions, Construction of Syntax Trees, Bottom-Up Evaluation of S-Attributed Definitions, L-Attributed Definitions, syntax directed definitions and translation schemes | 10 |
| 6 | **Intermediate-Code Generation:**<br>Variants of Syntax Trees, Three-Address Code, Types and Declarations, Translation of Expressions, Type Checking, Syntax Directed Translation Mechanisms, Attributed Mechanisms And Attributed Definition, Type Checking and Run time environments | 07 |
| 7 | **Code Generation and Optimization:**<br>Issues in the Design of a Code Generator, The Target Language, Addresses in the Target Code, Basic Blocks and Flow Graphs, Optimization of Basic Blocks, A Simple Code Generator, Machine dependent optimization, Machine independent optimization Error detection of recovery | 05 |
| | Total | 42 |

List of Practicals / Tutorials:

| 1 | a) Write a C program to remove all the comments from the program.<br>b) Write a C program to recognize identifiers and numbers. |
|---|---|
| 2 | Write a C program to generate tokens for a C program. |
| 3 | a) To Study about Lexical Analyzer Generator (LEX).<br>b) Create a Lex program to take input from text file and count no of characters, no. of lines & no. of words. |
| 4 | a) WAP to implement yytext method in a LEX program.<br>b) WAP to implement ECHO, REJECT functions provided in Lex.<br>c) WAP to implement BEGIN directive in a LEX program. |
| 5 | a) Write a Lex program to count number of vowels and consonants in a given input string.<br>b) Write a Lex program to print out all numbers from the given file.<br>c) Write a Lex program to count the number of comment lines in a given C program. |

| 6 | a) WAP to implement unput and input. |
|---|---|
| | b) WAP to implement yyterminate, yy_flush_bufferin LEX program. |
| | c) WAP to implement yywrap in LEX program. |
| | d) WAP to implement yymore and yyless in LEX program. |
| 7 | WAP to Find the "First" set |
| | Input: The string consists of grammar symbols. |
| | Output: The First set for a given string. |
| | Explanation: |
| | The student has to assume a typical grammar. The program when run will ask for the string to be entered. The program will find the First set of the given string. |
| 8 | WAP to Find the "Follow" set. |
| | Input: The string consists of grammar symbols. |
| | Output: The Follow set for a given string. |
| | Explanation: The student has to assume a typical grammar. The program when run will ask for the string to be entered. The program will find the Follow set of the given string. |
| 9 | Construct a recursive descent parser for a given grammar. |
| 10 | Write a C program for constructing of LL (1) parsing. |
| 11 | Implement a C program to implement operator precedence parsing. |
| 12 | Given a parsing table, Parse the given input using Shift Reduce Parser for any unambiguous grammar. |
| 13 | Introduction to YACC and generate calculator program. |
| 14 | Generate 3-tuple intermediate code for given infix expression. |
| 15 | Extract predecessor and successor from given control flow graph. |

**Reference Books:**

| 1 | Compilers: Principles, Techniques and Tools - A.V.Aho, Ravi Sethi, J.D.Ullman, Addison Wesley |
|---|---|
| 2 | The Theory and Practice of Compiler Writing - Tremblay J.P. And Sorenson P.G. |
| 3 | Compiler Design in C, Allen Holub, Prentice Hall |
| 4 | Compiler Construction - Waite W.N. And Goos G., Springer Verlag |
| 5 | Compiler Construction-Principles and Practices - D.M.Dhamdhere, Mcmillian |
| 6 | Principles of Compiler Design, V. Raghavan, McGrawHill |
| 7 | Compilers: Principles and practice, Dave and Dave, Pearson |
| 8 | Lex and Yacc: John R. Levine, Orielly |

| **Supplementary learning material:** |
|---|
| 1 | NPTEL - Swayam Courses |

**Pedagogy:**
- Direct classroom teaching
- Audio Visual presentations/demonstrations
- Assignments/Quiz
- Continuous assessment
- Seminar/Poster Presentation

**Suggested Specification table with Marks (Theory) (Revised Bloom's Taxonomy):**

| Distribution of Theory Marks in % | | | | | | **R**: Remembering; **U**: Understanding; |
|------|------|------|------|------|------|---|
| **R** | **U** | **A** | **N** | **E** | **C** | **A**: Applying; |
| 15% | 25% | 25% | 15% | 20% | --- | **N**: Analyzing; **E**: Evaluating; **C**: Creating |

Note: This specification table shall be treated as a general guideline for students and teachers. The actual distribution of marks in the question paper may vary slightly from above table.

**Course Outcomes (CO):**

| Sr. | Course Outcome Statements | %weightage |
|-----|--------------------------|------------|
| **CO-1** | Understand the basic concepts of formal languages and automata theory. | **10** |
| **CO-2** | Apply the knowledge of regular expressions and finite automata for lexical analyzer generator. | **25** |
| **CO-3** | Understand and use context free grammars for top-down and bottom-up parsing. | **30** |
| **CO-4** | Understand different representations of intermediate code, code optimization techniques and error recovery mechanisms. | **20** |
| **CO-5** | Apply the knowledge of lex tool & yacc tools to develop a scanner & parser. | **15** |

| Curriculum Revision: | |
|----------------------|---|
| Version: | 1.0 |
| Drafted on (Month-Year): | June-2020 |
| Last Reviewed on (Month-Year): | - |
| Next Review on (Month-Year): | June-2025 |