| Compiler Design | L | P | C |
|---|---|---|---|
| | 3 | | 3 |

| Discipline(s) / EAE / OAE | Semester | Group | Sub-group | Paper Code |
|---|---|---|---|---|
| CSE/IT/CST/ITE | 5 | PC | PC | CIC-303 |

**Marking Scheme:**

1. Teachers Continuous Evaluation: 25 marks
2. Term end Theory Examinations: 75 marks

**Instructions for paper setter:**

1. There should be 9 questions in the term end examinations question paper.
2. The first (1st) question should be compulsory and cover the entire syllabus. This question should be objective, single line answers or short answer type question of total 15 marks.
3. Apart from question 1 which is compulsory, rest of the paper shall consist of 4 units as per the syllabus. Every unit shall have two questions covering the corresponding unit of the syllabus. However, the student shall be asked to attempt only one of the two questions in the unit. Individual questions may contain upto 5 sub-parts / sub-questions. Each Unit shall have a marks weightage of 15.
4. The questions are to be framed keeping in view the learning outcomes of the course / paper. The standard / level of the questions to be asked should be at the level of the prescribed textbook.
5. The requirement of (scientific) calculators / log-tables / data – tables may be specified if required.

**Course Objectives :**

| | |
|---|---|
| 1. | introduce the major concept areas of language translation and compiler design. |
| 2. | To enrich the knowledge in various phases of compiler ant its use, code optimization techniques, machine code generation, and use of symbol table. |
| 3. | To extend the knowledge of parser by parsing LL parser and LR parser. |
| 4. | To provide practical programming skills necessary for constructing a compiler. |

**Course Outcomes (CO)**

| | |
|---|---|
| **CO 1** | Able to apply the knowledge of LEX tool & YACC tool to develop a scanner & parser. |
| **CO 2** | Able to design & implement a software system for backend of the compiler. |
| **CO 3** | Able to design syntax tree and intermediate code generator. |
| **CO 4** | To understand the concept of symbol table and to use various code optimization techniques |

**Course Outcomes (CO) to Programme Outcomes (PO) mapping (scale 1: low, 2: Medium, 3: High)**

| | PO01 | PO02 | PO03 | PO04 | PO05 | PO06 | PO07 | PO08 | PO09 | PO10 | PO11 | PO12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **CO 1** | 3 | 2 | - | 2 | 3 | 2 | - | - | - | - | - | 3 |
| **CO 2** | 3 | 2 | - | 2 | 3 | 2 | - | - | - | - | - | 3 |
| **CO 3** | 3 | 2 | - | 2 | 3 | 2 | - | - | - | - | - | 3 |
| **CO 4** | 3 | 2 | - | 2 | 3 | 2 | - | - | - | - | - | 3 |

**UNIT-I**

Compilers and translators, need of translators, structure of compiler: its different phases, compiler construction tools, Lexical analysis: Role of lexical analyzer, Input Buffering, A simple approach to the design of Lexical Analyzers, Specification and recognition of tokens, Finite automata, From regular expressions to automata, and vice versa, minimizing number of states of DFA, A language for specifying Lexical Analyzers, Design and implementation of lexical analyzer.

**UNIT-II**

The role of the parser, Context free grammars, Writing a grammar: Lexical versus Syntactic analysis, Eliminating ambiguity, Elimination of left recursion, Left factoring, Top Down Parsing: Recursive- Decent parsing, Non-recursive Predictive parsing, LL(1) grammars, Bottom Up Parsing: Shift Reduce Parsing, Operator precedence parsing, LR Parsing: SLR, LALR and Canonical LR parser, Parser Generators.

**UNIT-III**

Syntax Directed Translation: Syntax directed definitions, Evaluation orders for SDD's, construction of syntax trees, syntax directed translation schemes, implementation of syntax directed translation,
Intermediate Code Generation: Kinds of intermediate code: Postfix notation, Parse trees and syntax trees, Three-address code, quadruples and triples, Semantic Analysis: Types and Declarations, Translation of Expressions, Type checking.

**UNIT - IV**

Symbol Table: Symbol tables, its contents, Data Structure for Symbol Table: lists, trees, linked lists, hash tables, Error Detection and Recovery: Errors, lexical phase errors, syntactic phase errors, semantic errors, Error seen by each phase.
Code Optimization: The principal sources of optimizations, Loop optimization, Basic blocks and Flow Graphs, DAG representation of basic blocks, Code Generation: Issues in the design of code generation, A simple target machine mode, A Simple Code Generator, Peep-hole optimization, Register allocation and assignment.

**Textbook(s):**

1. Alfred V. Aho, Monica S. Lam, Ravi Sethi and Jeffrey D. Ullman, "Compilers Principle, Techniques, and Tool", Pearson.
2. Alfred V. Aho, Ravi Sethi and Jeffrey D. Ullman, "Compilers Principle, Techniques, and Tool", Addison Wesley.

**References:**

1. Trembley and Sorenson, "Theory and Practice of Compiler Writing", McGraw Hill.
2. Jhon R. Levine, Tony Mason and Doug Brown, —Lex &Yacc, O'Reilly.
3. M. Joseph, "Elements compiler Design", University Science Press.