# 19CS3101 - COMPILER DESIGN

|  |  |  |  |
| --- | --- | --- | --- |
| **Course**  **Category:** | Program Core | **Credits:** | 4 |
| **Course**  **Type:** | Theory | **Lecture-Tutorial-Practical:** | 3-1-0 |
| **Prerequisite:** | Basics of Programming Languages and Theory of Computation. | **Sessional Evaluation:**  **Univ. Exam Evaluation:**  **Total Marks:** | 40  60  100 |
| **Objectives** | * To make the student to understand the process involved in compilation. * Creating awareness among students on various types of bottom up parsers. * Understand the syntax analysis, intermediate code generation, type checking, and the role of symbol table etc. | | |

|  |  |  |
| --- | --- | --- |
| **Course Outcomes** | Upon successful completion of the course, the students will be able to: | |
| CO1 | Understand the basics of Compiler Design and the role of Lexical Analyzer |
| CO2 | Study various Syntax analyzers, grammar rules, SLR parsing techniques |
| CO3 | Get exposure on syntax translation mechanisms to be motivated to develop interpreters or compiles. |
| CO4 | Identify various storage allocation strategies, intermediate code generation and their applicability |
| CO5 | Acquire knowledge on code generation and Run-time storage Management |
| CO6 | Explore the techniques for code optimization. |
| **Course Content** | UNIT-I  **Introduction to Compiling:** Language Processors, Phases of a compiler, Cousins of the Compiler. Grouping of phases, Compiler construction tools.  **Lexical Analysis:** Role of the Lexical analyzer, Input buffering, Specification of tokens, Recognition of tokens, The Lexical analyzer Generator Lex.  UNIT-II  **Syntax Analysis:** Role of the parser, Context-free grammars, Writing a grammar, Top-down parsing, Bottom-up parsing.  UNIT-III  **LR Parsers:** Introduction to LR Parsing: Simple LR, More Powerful LR Parsers, Using ambiguous grammars, Parser generators.  **Syntax-Directed Translation:** Syntax-Directed Definitions, Evaluation Orders for SDD's, Applications of Syntax-Directed Translation, Syntax-Directed Translation Schemes..  UNIT-IV  **Run-Time Environments:** Storage organization, Stack Allocation of Space, Access to Nonlocal Data on the Stack.  Intermediate Code generation: Variants of Syntax Trees, Three-Address Code, Types and Declarations, Type Checking.  UNIT-V  **Code Generation:** Issues in the Design of a code generator, The target Language, Run-time storage management, Basic blocks and flow graphs, A simple code generator, Register allocation and assignment.    UNIT-VI  **Code Optimization:** The principal source of optimization, Peephole Optimization, Introduction to Data flow Analysis, Loops in flow graphs- Dominators, Depth-First Ordering, Edges in a Depth- First Spanning Tree, Depth of a Flow Graph, Natural Loops. | |
| **Text Books &**  **References**  **Books** | |  | | --- | | **TEXT BOOKS** | | 1. Compilers, Principles Techniques and Tools.Alfred V Aho, Monical S. Lam, Ravi Sethi Jeffery D. Ullman,2nd edition, pearson,2007 | |  | | **REFERENCE BOOKS** |  1. Alfred V.Aho, Jeffrey D.Ullman, Principles of Compiler Design,Publications. 2. Compiler Design K.Muneeswaran, OXFORD 3. Principles of compiler design,2nd edition,Nandhini Prasad,Elsevier 4. Compiler Construction-Principles and Practice, Kenneth C Louden, Cengage Learning | |
| **E-Resources** | 1. <https://nptel.ac.in/courses> 2. https://freevideolectures.com/university//iitm | |