**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR**

B.Tech (CSE)– III-II Sem          L   T   P   C

                  3   0   0   3

### (20A05601T) COMPILER DESIGN

**Course Objectives:**
- Teach the concepts related to assemblers, loaders, linkers and editors
- Introduce the basic principles of the compiler construction
- Explain the Concept of Context Free Grammars, Parsing and various Parsing Techniques.
- Expose the process of intermediate code generation.
- Instruct the process of Code Generation and various Code optimization techniques

**Course Outcomes:**

After completion of the course, students will be able to
- Differentiate the various phases of a compiler
- Design code generator
- Apply code optimization techniques
- Identify the tokens and verify the code

**UNIT I**            **Introduction**                     Lecture 8Hrs

**Introduction:** The structure of a compiler, the science of building a compiler, programming language basics

**Lexical Analysis:** The Role of the Lexical Analyzer, Input Buffering, Recognition of Tokens, The Lexical-Analyzer Generator Lex, Finite Automata, From Regular Expressions to Automata, Design of a Lexical-Analyzer Generator, Optimization of DFA-Based Pattern Matchers.

**UNIT II**            **Syntax Analysis**                  Lecture 9Hrs

Introduction, Context-Free Grammars, Writing a Grammar, Top-Down Parsing, Bottom-Up Parsing, Introduction to LR Parsing: Simple LR, More Powerful LR Parsers, Using Ambiguous Grammars and Parser Generators.

**UNIT III**          **Syntax-Directed Translation**            Lecture 9Hrs

**Syntax-Directed Translation:** Syntax-Directed Definitions, Evaluation Orders for SDD's, Applications of Syntax-Directed Translation, Syntax-Directed Translation Schemes, Implementing L-Attributed SDD's.

**Intermediate-Code Generation:** Variants of Syntax Trees, Three-Address Code, Types and Declarations, Type Checking, Control Flow, Switch-Statements, Intermediate Code for Procedures.

**UNIT IV**            **Code Generation**                 Lecture 8Hrs

**Run-Time Environments:** Stack Allocation of Space, Access to Nonlocal Data on the Stack, Heap Management, Introduction to Garbage Collection, Introduction to Trace-Based Collection.

**Code Generation:** 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, Peephole Optimization, Register Allocation and Assignment, Dynamic Programming Code-Generation.

**UNIT V**            **Machine-Independent Optimization**       Lecture 8Hrs

**Machine-Independent Optimization:** The Principal Sources of Optimization, Introduction to Data-Flow Analysis, Foundations of Data-Flow Analysis, Constant Propagation, Partial-Redundancy Elimination, Loops in Flow Graphs

**Textbooks:**
1. Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman, "Compilers Principles, Techniques and Tools", 2nd Edition, Pearson.

**Reference Books:**
1. Yunlin Su, Song Y. Yan, "Principles of Compilers", Springer, 2012.
2. Andrew W. Appel, "Modern Compiler Implementation in JAVA", 2$^{nd}$ edition, Cambridge University Press, 2004.
3. Lex &Yacc – John R. Levine, Tony Mason, Doug Brown, O'reilly
4. Compiler Construction, Louden, Thomson.

**Online Learning Resources:**
1. htttps://nptel.ac.in/courses/106108052/
2. http://openclassroom.stanford.edu/MainFolder/CoursePage.php?course=Compilers