**CS6660 COMPILER DESIGN L T P C**
**3 0 0 3**

**OBJECTIVES:**

**The student should be made to:**

- Learn the design principles of a Compiler.
- Learn the various parsing techniques and different levels of translation
- Learn how to optimize and effectively generate machine codes

**UNIT I INTRODUCTION TO COMPILERS 5**

Translators-Compilation and Interpretation-Language processors -The Phases of Compiler-Errors Encountered in Different Phases-The Grouping of Phases-Compiler Construction Tools - Programming Language basics.

**UNIT II LEXICAL ANALYSIS 9**

Need and Role of Lexical Analyzer-Lexical Errors-Expressing Tokens by Regular Expressions-Converting Regular Expression to DFA- Minimization of DFA-Language for Specifying Lexical Analyzers-LEX-Design of Lexical Analyzer for a sample Language.

**UNIT III SYNTAX ANALYSIS 10**

Need and Role of the Parser-Context Free Grammars -Top Down Parsing -General Strategies-Recursive Descent Parser Predictive Parser-LL(1) Parser-Shift Reduce Parser-LR Parser-LR (0)Item- Construction of SLR Parsing Table -Introduction to LALR Parser - Error Handling and Recovery in Syntax Analyzer-YACC-Design of a syntax Analyzer for a Sample Language .

**UNIT IV SYNTAX DIRECTED TRANSLATION & RUN TIME ENVIRONMENT 12**

Syntax directed Definitions-Construction of Syntax Tree-Bottom-up Evaluation of S-Attribute Definitions- Design of predictive translator - Type Systems-Specification of a simple type checker-Equivalence of Type Expressions-Type Conversions.

RUN-TIME ENVIRONMENT: Source Language Issues-Storage Organization-Storage Allocation-Parameter Passing-Symbol Tables-Dynamic Storage Allocation-Storage Allocation in FORTAN.

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

Principal Sources of Optimization-DAG- Optimization of Basic Blocks-Global Data Flow Analysis-Efficient Data Flow Algorithms-Issues in Design of a Code Generator - A Simple Code Generator Algorithm.

**TOTAL: 45 PERIODS**

**OUTCOMES:**

**At the end of the course, the student should be able to:**

- Design and implement a prototype compiler.
- Apply the various optimization techniques.
- Use the different compiler construction tools.

**TEXTBOOK:**

1. Alfred V Aho, Monica S. Lam, Ravi Sethi and Jeffrey D Ullman, “Compilers – Principles, Techniques and Tools”, 2nd Edition, Pearson Education, 2007.

**REFERENCES:**

1. Randy Allen, Ken Kennedy, “Optimizing Compilers for Modern Architectures: A Dependence-based Approach”, Morgan Kaufmann Publishers, 2002.
2. Steven S. Muchnick, “Advanced Compiler Design and Implementation, “Morgan Kaufmann Publishers - Elsevier Science, India, Indian Reprint 2003.
3. Keith D Cooper and Linda Torczon, “Engineering a Compiler”, Morgan Kaufmann Publishers Elsevier Science, 2004.
4. Charles N. Fischer, Richard. J. LeBlanc, “Crafting a Compiler with C”, Pearson Education, 2008.