COMPILER DESIGN

B.Tech. III Year II Semester								
Course Code	**Category**	**Hours / Week**			**Credits**	**Maximum Marks**		
CS3202PC	**Core**	**L**	**T**	**P**	**C**	**CIA**	**SEE**	**Total**
		3	**1**	**0**	**4**	**25**	**75**	**100**
Contact classes: 45	**Tutorial Classes : 15**	**Practical classes : NIL**				**Total Classes :60**		

Prerequisites

1. **A course on "Formal Languages and Automata Theory"**
2. **A course on "Computer Organization and architecture"**
3. **A course on "Computer Programming and Data Structures"**

Course Objectives:

- Introduce the major concepts of language translation and compiler design and impart the knowledge of practical skills necessary for constructing a compiler.
- Topics include phases of compiler, parsing, syntax directd translation, type checking use of symbol tables, code optimization techniques, intermediate code generation, code generation and data flow analysis.

Course Outcomes:

- Demonstrate the ability to design a compiler given a set of language features.
- Demonstrate the knowledge of patterns, tokens & regular expressions for lexical analysis.
- Acquire skills in using lex tool & yacc tool for devleoping a scanner and parser.
- Design and implement LL and LR parsers
- Design algorithms to do code optimization in order to improve the performance of a program in terms of space and time complexity.
- Design algorithms to generate machine code.

COURSE SYLLABUS

UNIT- I

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: 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: 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

Run-Time Environments: Stack Allocation of Space, Access to Non local 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, Peep hole Optimization, Register Allocation and Assignment, Dynamic Programming Code-Generation.

UNIT- V

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

TEXT BOOK:

1. Compilers : Principles, Techniques and Tools, Second Edition, Alfred V.Aho, Monica S.Lam, RaviSethi, Jeffry D.Ullman.

REFERENCE BOOKS:

1. Lex & Yacc– John R.Levine, Tony Mason, Doug Brown, O'reilly
2. Compiler Construction, Louden, Thomson.