Devi Ahilya University, Indore, India Institute of Engineering & Technology				**III Year B.E. (Computer Engineering)(Full Time)**			
Subject Code & Name	**Instructions Hours per Week**			**Credits**			
CER6C3	**L**	**T**	**P**	**L**	**T**	**P**	**Total**
Compiler Techniques	**3**	**1**	**2**	**3**	**1**	**1**	**5**
Duration of Theory Paper:3 Hours							

Learning Objectives:

- Provide a framework for understanding the fundamentals of Compiler.
- To familiarize students with new Compilation Problems.
- Develop skills to understand how to Design a Compiler.
- Develop ability to understand how to enhance performance of a Compiler for newly evolved Programming Languages.

Prerequisites: Knowledge of Computer Programming, Data Structures, Discrete Mathematics. Knowledge of several different Programming Languages will be useful.

Course Contents

Unit I : Introduction to Compiling

Compilers: The Analysis-Synthesis Model of Compilation, The Context of a Compiler; Analysis of the Source Program: Lexical Analysis, Syntax Analysis, Semantic Analysis; The Phases of a Compiler: Symbol-Table Management, Error Detection and Reporting, The Analysis Phases, Intermediate Code Generation, Code Optimization, Code Generation; Cousins of the Compiler: System Software, Interpreters, Kinds of Language Processors, Preprocessors, Assemblers, Linkers and Loaders, Macros; The Grouping of Phases: Front and Back Ends, Passes, Reducing the Number of Passes; Compiler Construction Tools.

Unit II: Lexical Analysis

The Role of the Lexical Analyzer: Lexical Analysis Versus Parsing, Tokens, Patterns and Lexemes, Attributes for Tokens, Lexical Errors; Specification of Tokens: Strings and Languages, Operations on Languages, Regular Expressions, Regular Definitions; Recognition of Tokens: Transition Diagrams.

Unit III: Syntax Analysis

Introduction: The Role of the Parser, Classification of Grammars, Syntax Error Handling; Context-Free Grammars: Parse Tree and Derivations, Ambiguity, CFG Versus Regular Expressions; Writing a Grammar: Lexical Versus Syntactic Analysis, Eliminating Ambiguity, Elimination

of Left Recursion, Left Factoring; Top- Down Parsing: Recursive Descent Parsing, LL(1) Grammars, Nonrecursive Predictive Parsing; Bottom-Up Parsing: Reductions, Shift Reduce Parsing, Conflicts During Shift Reduce Parsing; LR Parsing: Simple LR, Constructing SLR-Parsing Tables, Constructing LALR Parsing Tables; Using Ambiguous Grammars: Precedence and Associativity to Resolve Conflicts, The "Dangling-Else" Ambiguity.

Unit IV: Intermediate-Code Generation and Run-Time Environment

Variants of Syntax Trees: Directed Acyclic Graphs for Expressions, The Value-Number Method for Constructing DAG's; Three Address Code: Addresses and Instructions, Quadruples, Triples; Type Checking: Rules for Type Checking, Type Conversions; Storage Organization: Static Versus Dynamic Storage Allocation; Stack Allocation of Space: Activation Trees, Activation Records; Introduction to Garbage Collection: Design Goals for Garbage Collectors, Reachability.

Unit V: Code Optimization and Planning a Compiler

Basic Blocks and Flow Graphs: Basic Blocks, Flow Graphs, Representation of Flow Graphs; Optimization of Basic Blocks: The DAG Representation of Basic Blocks, Local Common Subexpressions Elimination, Semantic Preserving Transformations, Global Common Subexpressions Eliminations, Dead Code Elimination; Loop Optimization: Loop Invariant Code Motion, Reduction in Strength, Induction Variable Elimination, Loop Unrolling; Planning a Compiler: Source Language Issues, Target Language Issues, Performance Criteria.

Learning Outcomes:

Upon completing the course, students will be able to:

- Acquire advance knowledge and understanding of Compiler.
- Use skills in Compiler Design.
- Apply acquired knowledge to improve performance of a Compiler.

In addition to development in technology computer architectures offers a variety of resources of which students will be able to innovate in the Compiler Design.

Books Recommended:

[1] Compilers: Principles, Techniques, and Tools; Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman; Pearson Education.

[2] Compilers: Principles, Techniques, and Tools; Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman; Pearson Education.

[3] System Programming: D M Dhamdhere; McGraw Hill Education.

[4] System Programming and Operating Systems: D M Dhamdhere; McGraw Hill Education.

Suggested Exercises (For a Defined Language)

- Design a symbol table mechanism.

- Write an interpreter for quadruples.
- Write the lexical analyser.
- Write the semantic actions.
- Write the parser.
- Write the error-handling routines.