Devi Ahilya University, Indore, India Institute of Engineering & Technology				BE III Year Computer Engineering (FullTime)			
Subject Code : 5RCPC1	Instructions Hours per Week			Credits			
Subject Name: Compiler Design	**L**	**T**	**P**	**L**	**T**	**P**	**TOTAL**
Duration of Theory Paper: 3 Hours	2	1	0	2	1	0	3

Learning Objectives:

- Understand the fundamentals of compiler design and the phases of compilation.
- Analyze compiler architecture and its major components.
- Develop skills in implementing and managing different compiler phases.
- Evaluate issues related to code optimization, correctness, and runtime efficiency.

Prerequisites: Computer Programming, Data Structures, Discrete Mathematics

Course Outcomes (CO) and Program Outcomes (PO) Mapping

CO No.	Course Outcome	Program Outcomes (PO)
CO1	Explain the fundamental concepts of compiler design	PO1, PO2, PO10
CO2	Apply lexical analysis using regular expressions and finite automata.	PO1,PO2,PO3, PO4, PO10
CO3	Analyze and construct parsers using CFG, LL(1), SLR and LALR parsing techniques.	PO1, PO2 , PO3, PO5
CO4	Generate intermediate code, perform type checking and understand run-time storage organization.	PO1, PO3, PO4, PO5, PO10
CO5	Apply code optimization, code generation, compiler construction tools and virtual machine concepts	PO1, PO2, PO3, PO6 , PO9, PO10

CO-PO Relationship Matrix:

CO No.	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10
CO1	3	2	-	-	-	-	-	-	-	1
CO2	3	3	2	1	-	-	-	-	-	1
CO3	3	3	3	-	2	-	-	-	-	-
CO4	3	-	3	2	2	-	-	-	-	2
CO5	3	2	3	-	-	2	-	-	2	2

UNIT-I

Introduction to Compiler: Introduction to Compiler, Language Processing Systems: Cousins of the Compiler: System Software, Interpreters, Preprocessors, Assemblers, Linkers and Loaders, Structure of a Compiler, The Analysis-Synthesis Model of Compilation; The Phases of a Compiler: Symbol-Table Management, The Analysis Phases, Intermediate Code Generation, Code Optimization; The Grouping of Phases: Front and Back Ends.

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, Regular Expressions; Recognition of Tokens: Transition Diagrams; Finite Automata: Nondeterministic Finite Automata, Deterministic Finite Automata; From Regular Expression to Automata: Conversion of an NFA to DFA, Construction of an NFA from a Regular Expression.

UNIT-III

Syntax Analysis: Introduction: The Role of the Parser, Classification of Grammars; Context-Free Grammars: Parse Tree and Derivations, Ambiguity, CFG Versus Regular Expressions; Writing a Grammar: Lexical Versus Syntactic Analysis, Eliminating Ambiguity; Top- Down Parsing: Recursive. Descent Parsing, LL(1) Grammars, Nonrecursive Predictive Parsing; Bottom-Up Parsing: Reductions, Shift Reduce Parsing; LR Parsing: Simple LR, Constructing SLR-Parsing Tables, Constructing LALR Parsing Tables.

UNIT-IV

Intermediate-Code Generation and Run-Time Environment: Variants of Syntax Trees: Directed Acyclic Graphs for Expressions; 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; Introduction to Garbage Collection.

UNIT-V

Code Optimization and Code Generation : Basic Blocks and Flow Graphs: Basic Blocks, Flow Graphs; Optimization of Basic Blocks: The DAG Representation of Basic Blocks, Local Common Subexpressions Elimination, Semantic Preserving Transformations; Loop Optimization; Peephole Optimization Virtual Machines and their Interpreters; The LLVM Modular and Reusable Compiler Technologies, Compiler Construction Tools.

Learning Outcomes:

Upon completing the course, students will 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.

Recommended Books:

[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.