DeviAhilyaUniversity,Indore,India InstituteofEngineering&Technology				**IIIYearB.E. (Information Technology)(FullTime)**			
SubjectCode&Name	**InstructionsHoursper Week**			**Credits**			
6ITRG4	**L**	**T**	**P**	**L**	**T**	**P**	**Total**
Compiler Design	**3**	**1**	**1**	**3**	**1**	**1**	**5**
DurationofTheory Paper:3Hours							

Learning Objectives:

- 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.

Course Outcome

CO.No.	**CO**	**PO**
CO1	Acquire a solid understanding of the fundamental concepts in compiler design, including the analysis-synthesis model, the various phases of compilation, and the roles of lexical and syntax analysis.	PO-1, PO-2,PO-6,PO-9
CO2	Develop the ability to identify, analyze, and solve complex problems related to compiler construction using advanced principles of lexical analysis, syntax analysis, and optimization techniques.	PO-2, PO-3,PO-7
CO3	Design and implement various components of a compiler, such as lexical analyzers, parsers, and code generators, with a focus on enhancing efficiency and optimization.	PO-3, PO-5
CO4	Engage in detailed investigations of compiler construction challenges, utilizing research methodologies and experimental design to propose improvements in compiler performance and features.	PO-4, PO-11
CO5	Employ modern tools and technologies, such as lexical analyzer generators and parser generators, to facilitate the efficient development of compilers.	PO-5,PO-10

CO-PO Relationship

CO\PO	**PO-1**	**PO-2**	**PO-3**	**PO-4**	**PO-5**	**PO-6**	**PO-7**	**PO-8**	**PO-9**	**PO-10**	**PO-11**	**PO-12**
CO1	3	3				2			2			
CO2		3	3				3					
CO3			3		3							
CO4				3							3	
CO5					3					3		

* CO (rows) mention nil/very small/insignificant contribution to the PO(column)

1→ relevant and small significance 2 → medium or moderate and 3 →strong

Course Contents

UnitI:Introduction to Compiling

Compilers:TheAnalysis-SynthesisModelofCompilation,TheContext ofaCompiler; 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 GroupingofPhases: Front andBack Ends, Passes, Reducing theNumber of Passes; Compiler Construction Tools.

UnitII:LexicalAnalysis

TheRoleoftheLexicalAnalyzer:LexicalAnalysisVersusParsing, Tokens,PatternsandLexemes, Attributes for Tokens, Lexical Errors; Specification ofTokens: Strings and Languages, Operations on Languages, Regular Expressions, Regular Definitions; Recognition of Tokens: Transition Diagrams.

UnitIII:SyntaxAnalysis

Introduction: TheRoleoftheParser, ClassificationofGrammars, SyntaxError Handling;Context- Free Grammars: Parse Tree and Derivations, Ambiguity, CFG Versus Regular Expressions; WritingaGrammar:LexicalVersusSyntacticAnalysis,EliminatingAmbiguity, Elimination

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

UnitIV:Intermediate-CodeGenerationandRun-TimeEnvironment

VariantsofSyntaxTrees: 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.

UnitV:CodeOptimizationandPlanningaCompiler

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.

LearningOutcomes:

Uponcompletingthecourse,studentswillbeableto:

- Acquireadvanceknowledgeandunderstandingof Compiler.
- UseskillsinCompilerDesign.
- Applyacquiredknowledgetoimproveperformanceofa Compiler.

Inadditionto development intechnologycomputer architecturesoffersa varietyofresourcesof which students will be able to innovate in the Compiler Design.

BooksRecommended:

[1] Compilers:Principles,Techniques,andTools;AlfredV.Aho,RaviSethi,JeffreyD.Ullman; Pearson Education.

[2] Compilers:Principles,Techniques,andTools;AlfredV.Aho,MonicaS.Lam,RaviSethi, Jeffrey D. Ullman; Pearson Education.

[3] SystemProgramming:DMDhamdhere;McGrawHillEducation.

[4] SystemProgrammingandOperatingSystems:DMDhamdhere;McGrawHillEducation.

SuggestedExercises(ForaDefinedLanguage)

- Designasymboltablemechanism.

- Writeaninterpreterforquadruples.
- Writethelexicalanalyser.
- Writethesemanticactions.
- Writetheparser.
- Writetheerror-handlingroutines.

Course Outcomes (CO) for Compiler Design

CO-1: Theoretical Foundation
Acquire a solid understanding of the fundamental concepts in compiler design, including the analysis-synthesis model, the various phases of compilation, and the roles of lexical and syntax analysis.

CO-2: Analytical Skills
Develop the ability to identify, analyze, and solve complex problems related to compiler construction using advanced principles of lexical analysis, syntax analysis, and optimization techniques.

CO-3: Solution Design and Development
Design and implement various components of a compiler, such as lexical analyzers, parsers, and code generators, with a focus on enhancing efficiency and optimization.

CO-4: Research and Investigation
Engage in detailed investigations of compiler construction challenges, utilizing research methodologies and experimental design to propose improvements in compiler performance and features.

CO-5: Technological Proficiency
Employ modern tools and technologies, such as lexical analyzer generators and parser generators, to facilitate the efficient development of compilers.

CO-6: Societal Impact and Responsibility
Understand and evaluate the impact of compiler technologies on society, including considerations of software reliability, security, and efficiency, and apply this knowledge to responsible engineering practice.

CO-7: Environmental Awareness
Assess the environmental implications of software systems and design compilers that promote sustainable computing practices.

CO-8: Ethical Conduct
Commit to ethical principles in the development of compilers, ensuring integrity and professionalism in the software engineering process.

CO-9: Team Collaboration
Function effectively both as an individual contributor and as a leader in diverse, multidisciplinary teams, fostering a collaborative environment in compiler development projects.

CO-10: Effective Communication
Communicate complex technical information clearly and effectively, including writing detailed technical reports, delivering impactful presentations, and providing clear instructions.

CO-11: Project Management Skills
Apply project management principles to plan, execute, and oversee compiler development projects, ensuring they are completed within time constraints and budgetary limits.

CO-12: Commitment to Lifelong Learning
Recognize the importance of continuous learning in the rapidly evolving field of compiler design and maintain the ability to independently pursue knowledge and skill development to stay current with technological advancements.

Program Outcomes (PO) for Compiler Technique

PO-CD1: Foundational Concepts
Understand and apply the fundamental principles of compiler design, including the analysis-synthesis model, compilation phases, and the roles of lexical and syntax analysis.

PO-CD2: Analytical Proficiency
Identify, analyze, and develop solutions for complex compiler-related problems using advanced techniques in lexical analysis, syntax analysis, and code optimization.

PO-CD3: Design and Development
Design and implement key components of a compiler such as lexical analyzers, parsers, and code generators, with a focus on efficiency and performance optimization.

PO-CD4: Research and Inquiry
Conduct thorough investigations into challenges in compiler construction, utilizing research methods and experimental designs to enhance compiler functionality and performance.

PO-CD5: Technological Competence
Utilize state-of-the-art tools and technologies in compiler construction, including lexical analyzer and parser generators, to streamline and enhance the development process.

PO-CD6: Societal and Ethical Awareness
Assess the societal impacts of compiler technology, including software reliability, security, and efficiency, and apply this knowledge responsibly in professional practice.

PO-CD7: Environmental Responsibility
Evaluate the environmental impact of software systems and contribute to sustainable computing practices through thoughtful compiler design.

PO-CD8: Ethical Standards
Commit to and uphold ethical standards in the development and deployment of compilers, ensuring integrity and professionalism in all aspects of the work.

PO-CD9: Team Dynamics and Collaboration
Effectively collaborate within diverse, multidisciplinary teams, contributing to and leading projects that require comprehensive compiler development skills.

PO-CD10: Communication Skills
Communicate technical information effectively, including the ability to write detailed technical reports, make presentations, and provide clear instructions within the engineering community.

PO-CD11: Project Management Expertise
Apply project management principles to effectively plan, execute, and manage compiler development projects, ensuring they are completed on schedule and within budget constraints.

PO-CD12: Lifelong Learning and Adaptability
Recognize and embrace the need for continuous learning in the dynamic field of compiler design, maintaining the ability to independently pursue new knowledge and skills to keep pace with technological advancements.