Course Type	Course Code	Name of Course	L	T	P	Credit
DC9	CSC302	**Compiler Design**	3	0	0	9

Course Objective

The main objective of this course is to make the students understand various phases of a compiler with the associated techniques and algorithms to impart knowledge about designing a new compiler.

Learning Outcomes

Upon successful completion of this course, students will:

- Have a broad understanding of language translator and their need.
- Have a detailed understanding of various phases of a compiler and their design techniques.
- Be able to design a complier for new high level language.

Unit No.	Topics to be Covered	Lecture Hours	Learning Outcome
1	Introduction: Need of compilers; Introduction to phases of compilers, Cousins of compilers; Compiler writing tools, compiler phases.	2	To Introduce with language translators, their need and various phases of a compiler.
2	Lexical analysis: Transition diagrams, Tokens, regular expressions, Finite automata and its use, Implementation of a lexical analyzer.	5	To familiarize with various elements of a lexical analyser and to design it using transition diagram or finite automata.
3	Syntactic Specification of Programming Languages: Parse trees, Ambiguity, Regular expressions vs. Context free grammars (CFGs).	2	To learn about string derivation, parse tree representation and ambiguity in CFGs.
4	Basic Parsing Techniques: Shift reduce parsing, operator precedence parsing, Predictive parsing, Top down parsing.	6	To understand various parsing techniques, basic as well as advanced level and to design them.
5	Advanced Parsing Techniques: LR parsers (SLR, LALR, LR) and their design, Use of ambiguous grammars.	5	To impart knowledge of designing various LR parsers.
6	Syntax Directed Translation (SDT): Scheme, Implementation of SDT, Intermediate code, postfix notation, SDT to postfix code; Parse trees vs. Syntax trees, Three address code, SDT for assignment statement and Boolean expressions.	6	To impart knowledge of intermediate code generator and several syntax directed translation schemes.
7	Error Detection and Recovery: Lexical-phase errors, Syntactic-phase errors. Error detection and recovery from operator precedence, LR and Predictive parsing.	4	To familiarize with various kinds of compiler errors and to learn design of a error handler.
8	Code optimization: Sources, loops in flow graphs, loop optimization, Loop jamming and Loop Unrolling, DAG representation of basic blocks, Use of DAGs for code optimization.	5	To understand importance of code optimization and to learn various code optimization techniques.
9	Code generation: Issues, target machine, runtime storage management, basic block and flow graphs, a simple code generator, peephole optimization, code generation algorithm.	7	To learn about the components of code generation, flow graph, code generation algorithm.

Text Books:

1. Aho, Ullman, Sethi, *Compiler Principles, Techniques and Tools,* Addison-Wesley, 2004.

Reference Books:

1. Alfred Aho and Jeffrey Ullman, *Principles of Compiler Design*, Narosa, 2002.
2. Dave, Parag H., Dave, Himanshu B, Dave, Compilers: Principles and Practice, Pearson Education India, 2012.