CSE11132	Compiler Design	L	T	P	C
Version 1.0	Contact Hours 45	3	0	0	3
Pre-requisite/Exposure	Finite Automata, Data structures, Computer Organization.				
Co-requisite	NIL				

Course Objectives:

- To understand students how the process of language transla tion process.
- To interpret students the theory and practice of compiler implementation.
- To enhance the skills of student to implement lexical analysis, a variety of parsing techniques andsemantic analysis of a programming language, along with error
- detection and recovery.
- To allow the students to identify the various storage allocation, code optimization techniques and codegeneration.
- To understand students the use of object code generation process

Course Outcomes:

On the completion of this course the student will be able to

CO1: Understand the major phases of compilation, particularly lexical analysis.

CO2: Understand the basic concepts of parsing and Design parser for a given language using top down andBottom-up parser.

CO3: Demonstrate the use of formal attributed grammars for specifying the syntax and semantics ofProgramming languages.

CO4: Apply various optimization techniques for the design of a compiler.

CO5: Understand the concepts of symbol tables and implement code generation techniques.

Course Description:

This course will teach students the fundamental concepts and techniques used for building a simple compiler. It will also discuss the major ideas used today in the implementation of programming language compilers, including lexical analysis, parsing, syntax-directed translation, abstract syntax trees, types and type checking, intermediate languages, dataflow analysis, program optimization, code generation, and runtime systems. As a result, you will learn how a program written in a high-level language designed for humans is systematically translated into a program written in low-level assembly more suited to machines. Along the way we will also touch on how programming languages are designed, programming language semantics, and why there are so many different kinds of programming languages.

Course Content:

Units	Lecture Hours
Unit-I: Introduction of Compilers and Lexical Analysis and Lex/Flex: Overview of language processing pre-processors compiler assembler interpreters, pre-processors, linkers & loaders - structure of a compiler phases of a compiler. Lexical Analysis Role of Lexical Analysis Lexical Analysis Vs Parsing Token, patterns and Lexemes Lexical Errors Regular Expressions Regular definitions for the language constructs Strings, Sequences, Comments Transition diagram for recognition of tokens, Reserved words and identifiers, Examples.	**09**
Unit-II: Syntax Analysis and Yacc/Bison: Syntax Analysis discussion on CFG, LMD,RMD, parse trees, Role of a parser classification of parsing techniquesBrute force approach, left recursion, left factoring, Top down parsing First and Follow-LL(1) Grammars, Non- Recursive predictive parsing Error recovery in predictive parsing. Bottom up parsing, Types of Bottom up approaches. Introduction to simple LR Why LR Parsers Model of an LR Parsers Operator Precedence- Shift Reduce Parsing Difference between LR and LL Parsers, Construction of SLR Tables. More powerful LR parses, construction of CLR (1), LALR Parsing tables, Dangling ELSE Ambiguity, Error recovery in LR Parsing, Comparison of all bottoms up approaches with all top down approaches.	**09**
Unit-III: Intermediate Code Generation: Semantic analysis, SDT Schemes, evaluation of semantic rules. Intermediate Code Generation: Intermediate languages, three address code, quadruples, triples, abstract syntax trees. Types and declarations, Assignment statements, Boolean expressions, Case statements, Back Patching, Procedure calls type Checking.	**09**
Unit-IV: Code Optimization: Code Optimization: Introduction, The Principal sources of optimization, Optimization of basic blocks, Loops in flow graphs, Introduction to global data-flow analysis, Iterative solution of data-flow equations, Code improving transformations, Dealing with aliases, Data-flow analysis of structured flow graphs, Efficient data- flow algorithms,A tool for data-flow analysis, Estimation of types, Symbolic debugging of optimized code.	**09**
Unit-V: Run-Time Environment and Code Generation: Symbol tables: use and need of symbol tables. Runtime Environment: storage organization, stack allocation, access to non-local data, heap management, parameter passing mechanisms, introduction to garbage collection, Referencecounting garbage collectors. Code generation: Issues, target language, Basic blocks & flow graphs, Simple code generator, Peephole optimization, Register allocation and assignment.	**09**

Text Books:

- Compilers, Principles Techniques and Tools- Alfred V Aho, Monica S Lam, Ravi Sethi, Jeffrey D. Ullman,2ndEdition, Pearson,2007.

- piler construction, Principles and Practice, Kenneth C Louden, 1st Edition, CENGAGE

Reference Books:

- piler Design, K. Muneeswaran, Oxford. 2012
- Engineering a compiler, Keith D.Cooper& Linda Torczon, Morgan Kaufman, 2nd edition. MK publishers,2012. ciples of compiler design, V. Raghavan, 2nd Edition, Tata Mcgraw Hill, 2011.

Examination Scheme:

Components	Mid Term	Class Assessment	End Term
Weightage (%)	20	30	50

Relationship between Course Outcomes (COs) and Program Outcomes (POs):

1 = Weakly Mapped

2 = Moderately Mapped3 =

Strongly Mapped

Course Code	Course Name	COs	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PS O1	PS O2	PS O3
CSE111	Compi	CO1113	2	3	2	2	3	3	1	-	1	-	-	2	3	1	1
32	ler	2.1															
	Design	CO1113	2	3	2	2	1	1	2	-	1	-	-	3	2	2	2
		2.2															
		CO1113	3	3	3	1	3	1	2	-	3	-	-	3	3	2	1
		2.3															
		CO1113	3	1	1	2	2	1	1	-	2	-	-	2	3	2	2
		2.4															
		CO1113	3	2	3	2	3	2	3	-	2	-	-	3	3	1	3
		2.5															
		CO1113	2.	2.	2.	1.	2.	1.	1.	-	1.	-	-	2.6	2.8	1.6	1.8
		2	6	4	2	8	4	6	8		8						