CSE2002	THEORY OF COMPUTATION AND COMPILER DESIGN	L	T	P	J	C
		4	0	0	0	4
Pre-requisite	**NIL**	**Syllabus version**				
		v1.1				

Course Objectives:

1. Provides required theoretical foundation for a computational model and compiler design
2. Discuss Turing machines as a abstract computational model
3. Compiler algorithms focus more on low level system aspects.

Expected Course Outcome:

On successful completion of the course, the student should be able to:

1. Design computational models for formal languages
2. Design scanners and parsers using top-down as well as bottom-up paradigms
3. Design symbol tables and use them for type checking and other semantic checks
4. Implement a language translator
5. Use tools such as lex, YACC to automate parts of implementation process

Student Learning Outcomes (SLO): 1,9,18

Module:1	**Introduction To Languages and Grammers**	**3 hours**

Overview of a computational model - Languages and grammars – alphabets – Strings - Operations on languages, Introduction to Compilers - Analysis of the Source Program - Phases of a Compiler

Module:2	**Regular Expressions and Finite Automata**	**9 hours**

Finite automata – DFA – NFA – Equivalence of NFA and DFA (With Proof) - Regular expressions – Conversion between RE and FA (With Proof) Lexical Analysis - Recognition of Tokens - Designing a Lexical Analyzer using finite automata

Module:3	**Myhill-Nerode Theorem**	**4 hours**

Myhill-Nerode Theorem - Minimization of FA – Decision properties of regular languages – Pumping lemma for Regular languages (With Proof)

Module:4	**CFG, PDAs and Turing Machines**	**15 hours**

CFG – Chomsky Normal Forms - NPDA – DPDA - Membership algorithm for CFG. Syntax Analysis - Top-Down Parsing - Bottom-Up Parsing - Operator-Precedence Parsing - LR Parsers

Module:5	**Turing Machines**	**5 hours**

Turing Machines – Recursive and recursively enumerable languages – Linear bounded automata - Chomsky's hierarchy – Halting problem

Module:6	**Intermediate Code Generation**	**10 hours**

Intermediate Code Generation - Intermediate Languages – Declarations - Assignment Statements - Boolean Expressions - Case Statements – Backpatching - Procedure Calls.

Module:7	**Code Optimization**	**7 hours**

Code Optimization - Basic Blocks and Flow Graphs – The DAG Representation of Basic Blocks - The Principal Sources of Optimization - Optimization of Basic Blocks - Loops in Flow Graphs - Peephole Optimization - Introduction to Global Data-Flow Analysis

Module:8	**Code Generation**	**7 hour**
Code Generation – Issues in the Design of a Code Generator - The Target Machine - Run-Time Storage Management - Next-Use Information - Register Allocation and Assignment - A Simple Code Generator - Generating Code from DAG Recent Trends – Just-in-time compilation with adaptive optimization for dynamic languages - Parallelizing Compilers Total Lecture Hours		
	Total Lecture hours:	**60 hours**

Text Book(s)	
1.	Introduction to Automata Theory, Languages, and Computation (3rd Edition), John E Hopcroft, Rajeev Motwani, Jeffery D. Ullman, Pearson education, 2013.
2.	Principles of Compiler Design, Alferd V. Aho and Jeffery D. Ullman, Addison Wesley, 2006
Reference Books	
1.	Introduction to Languages and the Theory of Computation, John Martin, McGraw-Hill Higher Education,2010
2.	Modern Compiler Implementation in Java, 2nd ed., Andrew W. Appel Cambrdige University Press, 2012.

Mode of Evaluation: CAT / Assignment / Quiz / FAT / Project / Seminar			
Recommended by Board of Studies			
Approved by Academic Council	No. 47	Date	05.10.2017