Course Code	PCS21E09J	Course Name	COMPILER DESIGN	Course Category	D	Discipline Elective Course	L	T	P	C
							3	0	2	4

Pre-requisite Courses	Nil	**Co-requisite Courses**	Nil	**Progressive Courses**	Nil
Course Offering Department	Computer Science	**Data Book / Codes/Standards**		Nil	

Course Learning Rationale (CLR):	The purpose of learning this course is to:	Learning			Program Learning Outcomes (PLO)														
		1	2	3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CLR-1 :	*Utilize the mathematics and engineering principles for the Design of Compilers*																		
CLR-2 :	*Acquire knowledge of Lexical Analyzer from a specification of a language's lexical rules*																		
CLR-3 :	*Acquire knowledge of Syntax Analyzer for parsing the sentences in a compiler grammar*																		
CLR-4 :	*Gain knowledge to translate a system into various intermediate codes*																		
CLR-5 :	*Analyze the methods of implementing a Code Generator for compilers*																		
CLR-6 :	*Analyze and Design the methods of developing a Code Optimizer*																		
Course Learning Outcomes (CLO):	At the end of this course, learners will be able to:	Level of Thinking (Bloom)	Expected Proficiency (%)	Expected Attainment (%)	Engineering Knowledge	Problem Analysis	Design & Development	Analysis, Design, Research	Modern Tool Usage	Society & Culture	Environment & Sustainability	Ethics	Individual & Team Work	Communication	Project Mgt. & Finance	Life Long Learning	PSO - 1	PSO - 2	PSO – 3
CLR-1 :	*Utilize the mathematics and engineering principles for the Design of Compilers*	3	80	70	H	H	H	H	M										
CLR-2 :	*Acquire knowledge of Lexical Analyzer from a specification of a language's lexical rules*	3	85	75	H	H	H	H	M										
CLR-3 :	*Acquire knowledge of Syntax Analyzer for parsing the sentences in a compiler grammar*	3	75	70	H	H	H	H	M										
CLR-4 :	*Gain knowledge to translate a system into various intermediate codes*	3	85	80	H	H	H	H	M										
CLR-5 :	*Analyze the methods of implementing a Code Generator for compilers*	3	85	75	H	H	H	H	M										
CLR-6 :	*Analyze and Design the methods of developing a Code Optimizer*	3	80	70	H	H	H	H	M										

Duration (hour)		**15**	**15**	**15**	**15**	**15**
S-1	SLO-1	*Compilers – Analysis of the source program*	*Syntax Analysis Definition - Role of parser*	*Bottom Up Parsing*	*Intermediate Code Generation*	*Code optimization*
	SLO-2	*Phases of a compiler – Cousins of the Compiler*	*Lexical versus Syntactic Analysis*	*Reductions*	*Intermediate Languages - prefix - postfix*	*Introduction– Principal Sources of Optimization*
S-2	SLO-1	*Grouping of Phases – Compiler construction tools*	*Representative Grammars*	*Handle Pruning*	*Quadruple - triple - indirect triples Representation*	*Function Preserving Transformation*
	SLO-2	*Lexical Analysis – Role of Lexical Analyzer*	*Syntax Error Handling*	*Shift Reduce Parsing*	*Syntax tree- Evaluation of expression - three-address code*	*Loop Optimization*
S-3	SLO-1	*Input Buffering*	*Elimination of Ambiguity, Left Recursion*	*Problems related to Shift Reduce Parsing*	*Synthesized attributes – Inherited attributes*	*Optimization of basic Blocks*
	SLO-2	*Specification of Tokens*	*Left Factoring*	*Conflicts During Shift Reduce Parsing*	*Intermediate languages – Declarations*	*Building Expression of DAG*
S 4-5	SLO-1 SLO-2	*Laboratory 1: Implementation of Lexical Analyzer*	*Laboratory 4:Elimation of Ambiguity, Left Recursion and Left Factoring*	*Laboratory 7 : Shift Reduce Parsing*	*Laboratory 10:Intermediate code generation – Postfix, Prefix*	*Laboratory 13:Implementation of DAG*
S-6	SLO-1	*Finite automation - deterministic*	*Top down parsing*	*LR Parsers- Why LR Parsers*	*Assignment Statements*	*Peephole Optimization*
	SLO-2	*Finite automation - non deterministic*	*Recursive Descent Parsing, back tracking*	*Items and LR(0) Automaton, Closure of Item Sets,*	*Boolean Expressions, Case Statements*	*Basic Blocks, Flow Graphs*
S-7	SLO-1	*Transition Tables*	*Computation of FIRST*	*LR Parsing Algorithm*	*Back patching – Procedure calls*	*Next -Use Information*

Duration (hour)		15	15	15	15	15
	SLO-2	*Acceptance of Input Strings by Automata*	*Problems related to FIRST*	*Operator Precedence Parser Computation of LEADING*	*Code Generation*	*Introduction to Global Data Flow Analysis*
S-8	SLO-1	*State Diagrams and Regular Expressions*	*Computation of FOLLOW*	*Computation of TRAILING*	*Issues in the design of code generator*	*Computation of gen and kill*
	SLO-2	*Conversion of regular expression to NFA – Thompson's*	*Problems related to FOLLOW*	*Problems related to LEADING AND TRAILING*	*The target machine – Runtime Storage management*	*Computation of in and out*
S 9-10	SLO-1	*Laboratory 2: conversion from Regular Expression to NFA*	*Laboratory 5:FIRST AND FOLLOW computation*	*LaboratoryLab 8: Computation of LEADING AND TRAILING*	*Laboratory 11: Intermediate code generation – Quadruple, Triple, Indirect triple*	*Laboratory 14 : Implementation of Global Data Flow Analysis*
	SLO-2					
S-11	SLO-1	*Conversion of NFA to DFA*	*Construction of a predictive parsing table*	*SLR Grammars*	*A simple Code generator*	*Parameter Passing.*
	SLO-2	*Simulation of an NFA*	*Predictive Parsers LL(1) Grammars*	*SLR Parsing Tables*	*Code Generation Algorithm*	*Runtime Environments*
S-12	SLO-1	*Converting Regular expression directly to DFA*	*Transition Diagrams for Predictive Parsers*	*Problems related to SLR*	*Register and Address Descriptors*	*Source Language issues*
	SLO-2	*Minimization of DFA*	*Error Recovery in Predictive Parsing*	*Construction of Canonical LR(1) and LALR*	*Generating Code of Assignment Statements*	*Storage Organization*
S-13	SLO-1	*Minimization of NFA*	*Predictive Parsing Algorithm*	*Construction of LALR*	*Cross Compiler – T diagrams*	*Activation Records*
	SLO-2	*Design of lexical analysis (LEX)*	*Non Recursive Predictive Parser*	*Problems related to Canonical LR(1) and LALR Parsing Table*	*Issues in Cross compilers*	*Storage Allocation strategies*
S 14-15	SLO-1	*Laboratory 3: Conversion from NFA to DFA*	*Laboratory 6 :Predictive Parsing Table*	*Laboratory9 : Computation of LR(0) items*	*Laboratory12 : A simple code Generator*	*Laboratory 15: Implement any one storage allocation strategies(heap, stack, static)*
	SLO-2					

Learning Resources	1. AlfredVAho,JefferyDUllman,RaviSethi,"Compilers,Principlestechniquesandtools",Pearson Education2011 2. S.GodfreyWinster,S.ArunaDevi,R.Sujatha,"CompilerDesign",YesdeePublishingPvt.Ltd,2016 3. WilliamM.WaiteandGerhardGoos.CompilerConstruction.Springer-Verlag,New York,2013.	4. K.Muneeswaran,,"CompilerDesign",OxfordHigherEducation,Fourthedition2015 5. DavidGalles,"ModernCompilerDesign",PearsonEducation,Reprint2012. 6. RaghavanV.,"PrinciplesofCompilerDesign",TataMcGrawHillEducationPvt.Ltd.,2010

Learning Assessment

Bloom's Level of Thinking		Continous Learning Assessment(50% Weightage)								Final Examination (50% weightage)	
		CLA – 1 (10%)		CLA – 2 (10%)		CLA – 3 (20%)		CLA – 4# (10%)			
		Theory	Practice	Theory	Practice	Theory	Practice	Theory	Practice	Theory	Practice
Level 1	Remember	20%	20%	15%	15%	15%	15%	15%	15%	15%	15%
	Understand										
Level 2	Apply	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%
	Analyze										
Level 3	Evaluate	10%	10%	15%	15%	15%	15%	15%	15%	15%	15%
	Create										
	Total	100 %		100 %		100 %		100 %		100%	

# CLA – 4 can be from any combination of these: Assignments, Seminars, Tech Talks, Mini-Projects, Case-Studies, Self-Study, MOOCs, Certifications, Conf. Paper etc.,

Course Designers		
Experts from Industry	**Experts from Higher Technical Institutions**	**Internal Experts**
Mr. S. Karthik, Assistant Consultant, Tata Consultancy Services	Dr. S. Sasikala, Associate Professor and Head, Dept. of Computer Science, University of Madras	Dr.S.P.Angelin Claret
		Mrs.P.Yogalakshmi