COMPILER DESIGN	
CSE 321	**Credits : 4**
Instruction : 4 Periods & 1 Tut/week	Sessional Marks : 40
End Exam : 3 Periods	End Exam Marks : 60

Prerequisites:

Basic fundamentals of Discrete Mathematics
Principles of Automata Theory.

Course Objectives:

- Introduce the major concept areas of language translation and compiler design.
- Learn the design of lexical analyzer, syntax analyzer.
- Enrich the knowledge in various phases of compiler and its use, intermediate code generation, optimization techniques, machine code generation, and use of symbol table.
- Provide practical programming skills necessary for constructing a compiler.

Course Outcomes:

By the end of the course, the student will be able to:	
1.	Identify the challenges of theory of computations and Explain different phases of a compiler and design of lexical analyzer, and differentiate between various parsers.
2.	Explain how Top down parsing is done
3.	Identify the differences in the functioning of various bottom up parsers
4.	Differentiate different intermediate code generation techniques and
5.	Compare different code generation techniques, and how symbol table and run time storage are managed.

Mapping of course outcomes with program outcomes:

Mapping		PO												PSO	
		1	2	3	4	5	6	7	8	9	10	11	12	1	2
CO	1	1	2		1								1		
	2	2	1	2	2								1		
	3	2	3		2								2		
	4	2	3		2								1		
	5	3	3		3								2		

SYLLABUS

UNIT-I :

The Theory of Automata: **12 Periods**

Overview of Finite Automata and Formal Languages.

Overall view of Compilers:

Types of Translators, Brief discussion on various phases of Compilers, Design of lexical analyzer, LEX tool.

UNIT-II :

Design of Parsers: **10 Periods**

Top down Parsers, Problems with Top down Parsers, Backtracking, Left recursion, Left factorial, Predictive Parser.

UNIT-III : **18 Periods**

Bottom up parser: Shift Reduce parser, Operator Precedence Parser, LR parser: LR(0), SLR,CLR parsers. LALR parser, parsing of string, YACC TOOL.

UNIT-IV :

Syntax Directed Translation: **18 Periods**

Syntax directed translation and implementation, Intermediate code, Postfix notation, DAG, t Periodsee address Code, Quadruples, and Triples, indirect triples.

Machine independent Code Optimization: The principle sources of optimization, local Optimization, Loop Optimization, DAG, Global data flow analysis.

UNIT-V :

Code Generation: **18 Periods**

Problems, Machine model, A simple code generator, Machine dependent code Optimization**,** Register allocation and assignment, Code generation from DAG, Peephole optimization.

Brief discussion on symbol tables, Run-time storage administration.

Text Book:

1. Aho, D. Ullman "*Principles of Compiler Design* ",Second Edition,Pearson Education

Reference Books:

1. Santanu Chattopadhyay, "*Compiler Design*", Sixth Edition,PHI Learning Pvt. Ltd.
2. A.A.Puntambekar , *"Compiler design".* First Edition, Technical Publications .
3. Alfred V. Aho, ,Monica S. Lam, ,Ravi Sethi, Jeffrey D. Ullman, "*Compilers: Principles, Techniques, and Tools*",2nd Edition, Pearson Education

Web resources:

1. http://nptel.ac.in/courses/106104123/.
2. http://www.nptelvideos.in/2012/11/compiler-design.html.