COMPILER DESIGN

Course Code: 19CS1106

	L	T	P	C
	3	0	0	3

Course Outcomes: At the end of the Course the student will be able to:

CO1: Build the lexical analyzer phase of compiler.
CO2: Construct parsing tables for a given grammar.
CO3: Develop Syntax Directed Translation Schemes.
CO4: Model SDD's using Intermediate Representations.
CO5: Develop algorithms to generate code for a target machine.

UNIT- I **(10 Lectures)**

INTRODUCTION TO COMPILING: Overview of Compilers, Phases of a Compiler, Bootstrapping.
LEXICAL ANALYSIS: The Role of Lexical Analyzer, Input Buffering, Specification of Tokens: Regular Expressions, Regular Definitions- Recognition of Tokens, A language for specifying Lexical Analyzers(LEX).

Learning Outcomes: At the end of the unit the student will be able to

1. understand the phases of a compiler. (L2)
2. identify the tokens and lexemes in a given input. (L3)
3. apply Regular Expressions for specifying tokens. (L3)

UNIT-II **(10 Lectures)**

SYNTAX ANALYSIS: The role of the Parser, Context-free Grammars, Elimination of Left Recursion, Left factoring a grammar.
TOP-DOWN PARSING: Recursive descent Parsing, First and Follow, Predictive Parsing, LL(1) Grammars.
BOTTOM-UP PARSING: Shift-Reduce Parser, LR Parsers-SLR, Canonical LR, LALR, Operator Precedence Parser,Parser Generator(YACC).

Learning Outcomes: At the end of the unit the student will be able to

1. apply rules to make the grammar ready for parsing.(L3)
2. construct Predictive Parsing table for the given grammar. (L3)
3. build various LR Parsing tables for a given grammar. (L3)

UNIT-III **(10 Lectures)**

SYNTAX-DIRECTED TRANSLATION: Syntax-Directed Definition, S-Attributed SDD, L-Attributed SDD, Translation Schemes.

TYPE CHECKING: Type Systems, Specification of a Simple type checker, Equivalence of Type Expressions, Type Conversions, Overloading of functions and operators.

Learning Outcomes: At the end of the unit the student will be able to

1.compare S-Attributed SDD and L-Attributed SDD. (L2)
2.build a type system for a simple language. (L3)
3.explain the two kinds of type conversions .(L2)

UNIT-IV **(10 Lectures)**

RUN-TIME ENVIRONMENTS: Source Language Issues, Storage Organization, Storage Allocation Strategies, Blocks, Access Links, Procedure Parameters, Displays, Parameter Passing, Symbol Tables.

INTERMEDIATE CODE GENERATION: Intermediate Languages- Graphical Representations, Three address code, Implementations.

Learning Outcomes: At the end of the unit the student will be able to

1.summarize various storage allocation strategies. (L2)
2 explain various parameter passing methods.(L2)
3.develop various representations for three address code.(L3)

UNIT- V **(10 Lectures)**

CODE OPTIMIZATION: Introduction, Principle sources of optimization.

CODE GENERATION: Issues in the Design of a Code Generator, The Target Language, Basic Blocks and Flow Graphs, A Simple Code Generator, Peephole optimization.

Learning Outcomes: At the end of the unit the student will be able to

1.apply optimization techniques on a given code.(L3)
2.build a flow graph from the identified basic blocks.(L3)
3.apply rules to design a simple code generator.(L3)

TEXT BOOKS:

1. Alfred V Aho, Monica S Lam, Ravi Sethi, Jeffrey D. Ullman, *Compilers- Principles Techniques, and Tool,*2nd Edition, Pearson Education India,2013.

REFERENCE BOOKS:

1. V.raghavan, *Principles of Compiler Design*,1st Edition, McGraw Hill Education,2017.
2. Alfred V Aho, Ravi Sethi, Jeffrey D. Ullman, *Compilers- Principles Techniques, and Tool,* 2nd Edition, Pearson Education,2008.
3. Kenneth C. Louden, *Compiler Construction Design*,2nd Edition, Cengage, 2010.

WEB REFERENCES:

1. https://swayam.gov.in/nd1_noc20_cs13/preview