COMPILER DESIGN

PC 601 CS

Instruction: 3L+1T periods per week *Duration of SEE: 3 hours*
CIE: 30 marks *SEE: 70 marks*
Credits: 3

Objectives:

➢ To understand and list the different stages in the process of compilation.
➢ Identify different methods of lexical analysis
➢ Design top-down and bottom-up parsers
➢ Identify synthesized and inherited attributes
➢ Develop syntax directed translation schemes
➢ Develop algorithms to generate code for a target machine

Outcomes:

Upon completion of the course, the students will be able to:
1. For a given grammar specification, develop the lexical analyzer.
2. For a given parser specification, design top-down and bottom-up parsers.
3. Develop syntax directed translation schemes.
4. Develop algorithms to generate code for target machine.

UNIT-I
Introduction: The Structure of a Compiler, Phases of Compilation, The Translation Process, Major Data Structures in a Compiler, Bootstrapping and Porting. **Lexical Analysis (Scanner)**: The Role of the Lexical Analyzer, Input Buffering, Specification of Tokens, Recognition of Tokens, The Lexical Analyzer Generator Lex.
UNIT-II
Syntax Analysis (Parser): The Role of the Parser, Syntax Error Handling and Recovery, Top-Down Parsing, Bottom-Up Parsing, Simple LR Parsing, More Powerful LR Parsing, Using Ambiguous Grammars, Parser Generator Yaac.
UNIT-III
Syntax-Directed Translation: Syntax-Directed Definitions, Evaluation Orders for SDD's Applications of Syntax-Directed Translation. **Symbol Table:** Structure, Operations, Implementation and Management.
UNIT-IV
Intermediate Code Generation: Variants of Syntax Trees, Three-Address Code, Types and Declarations, Translation of Expressions, Type Checking, Control Flow, Backpatching, Switch-statements, Intermediate Code for Procedures. **Run-time environment**: Storage Organization, Stack Allocation of Space, Access to Nonlocal Data on the Stack, Parameter passing, Heap Management and Garbage Collection.
UNIT-V
Code Generation: Issues in the Design of a Code Generator, The Target Language, Addresses in the Target Code, Basic Blocks and Flow graphs, Optimization of Basic Blocks, Peephole Optimization, Register Allocation and Assignment. **Machine-Independent Optimizations:** The Principal Sources of Optimizations, Introduction to Data-Flow Analysis.

Suggested Readings:

1.	Alfred V. Aho, Monica S. Lam, Ravi Sethi, & Jeffrey D. Ullman , *Compilers :Principles, Techniques and Tools*, 2nd Edition, Pearson Education, 2006.
2.	Kenneth C. Louden**,** *Compiler Construction: Principles and Practice*, *Thomson Learning Inc., 1997.*
3.	P.Trembley and P.S.Sorenson, ***The Theory and Practice of Compiler Writing***, TMH-1985.

COMPILER DESIGN LAB

PC 651 CS

Instruction: 2P periods per week *Duration of SEE: 3 hours*
CIE: 25 marks *SEE: 50 marks*
Credits: 1

Objectives:

1. To learn usage of tools LEX, YAAC
2. To develop a code generator
3. To implement different code optimization schemes

Outcomes:

1. Generate scanner and parser from formal specification.
2. Generate top down and bottom up parsing tables using Predictive parsing, SLR and LR Parsing techniques.
3. Apply the knowledge of YACC to syntax directed translations for generating intermediate code – 3 address code.
4. Build a code generator using different intermediate codes and optimize the target code.

List of Experiments to be performed:

1. Sample programs using LEX.
2. Scanner Generation using LEX.
3. Elimination of Left Recursion in a grammar.
4. Left Factoring a grammar.
5. Top down parsers.
6. Bottom up parsers.
7. Parser Generation using YACC.
8. Intermediate Code Generation.
9. Target Code Generation.
10. Code optimization.