Compiler Construction

Semester VI **Subject Code: BS61702** **Lectures: 60**

Objectives:

The syllabus aims in equipping students with,

- To understand the various phases of a compiler and to develop skills in designing a compiler
- To learn working of lexical analyzer and use of LEX tool
- To learn working of a parser and use of YACC tool
- To study techniques of code optimization

Unit 1: Introduction and Lexical Analysis **10**

Ch 1: Introduction

- Definition of Compiler
- The structure of Compiler.
- Introduction to Phases of Compiler – Lexical Analysis, Syntax Analysis, Semantic Analysis, Intermediate Code generation, code optimization, code generation.
- Introduction to one pass & Multi-pass compilers, cross compiler, Bootstrapping.

Ch 2: Lexical Analysis(Scanner)

- Review of Finite automata as a lexical analyzer,
- Applications of Regular Expressions and Finite Automata (lexical analyzer, searching using RE), Input buffering, Recognition of tokens
- LEX: A Lexical analyzer generator (Simple Lex Program)

Unit 2: Syntax Analysis **20**

Ch 3: Syntax Analysis(Parser)

- Definition , Types of Parsers
- Top-Down Parser –
  - Top-Down Parsing- Introduction
  - Drawbacks of Top-Down parsing
  - Elimination of Left Recursion(direct & indirect)
  - Need for Left Factoring & examples
- Recursive Descent Parsing : Definition Implementation of Recursive Descent Parser Using Recursive Procedures
- Predictive [LL(1)]Parser(Definition, Model)

- Implementation of Predictive Parser[LL(1)] using stack
- FIRST & FOLLOW
- Construction of LL(1) Parsing Table

- Bottom-Up Parsers
- Operator Precedence Parser -Basic Concepts
  - Operator Precedence Relations from Associativity & Precedence
  - Operator Precedence Grammar
  - Algorithm for LEADING & TRAILING (with ex.)
  - Algorithm for Operator Precedence Parsing (with ex.)
  - Precedence Functions
- Shift Reduce Parser
  - Reduction, Handle, Handle Pruning
  - Stack Implementation of Shift Reduce Parser (with examples)
- LRParser
  - Model
  - Types [SLR(1), Canonical LR, LALR] Method & examples.
- YACC –program sections, simple YA CC program for expression evaluation

Unit 3: Semantic Analysis **8**

Ch 4: Semantic Analysis : Syntax Directed Translation

- Semantic Analysis
- Syntax Directed Translation: Introduction, two notations for attaching semantic rules (1. Syntax Directed Definitions, 2. Translation Schemes),
- Syntax Directed Definitions: Attributes, Semantic Rules, Annotated Parse-Trees, Synthesized Attributes, Inherited Attributes, example
- Implementing Syntax Directed Definitions
  - Dependency Graphs
  - S-Attributed Definitions
  - L-Attributed Definitions
- Translation Schemes: Definition, Example

Unit 4: Code Generation and Optimization **10**

Ch 5: Code Generation and Optimization

- Compilation of expression –
  - Concepts of operand descriptors and register descriptors with example.
  - Intermediate code for expressions – postfix notations, Triples and quadruples, expression trees.
- Code Optimization – Optimizing transformations – compile time evaluation, elimination of common sub expressions, dead code elimination, frequency reduction, strength reduction
- Three address code
  - DAG for Three address code
  - The Value-number method for constructing DAG's.

- Definition of Basic Blocks and Flow Graphs
- Directed Acyclic Graph (DAG) representation of basic block
- Issues in design of code generator

***Contact hours – 12 hours**

Reference Books:

3. Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, *Compilers: Principles, Techniques, and Tools*
4. Alfred V. Aho, Jeffrey D. Ullman, *Principles of Compiler Design*, Narosa Publication House
5. LEX & YACC, O'reilly Publication