## COMPILER DESIGN

# **Course Objectives:**

The course objectives of Compiler Design are to discuss and make student familiar with the

- 1. Knows and learn about various phases in the design of a compiler.
- 2. Study the design of top-down and bottom-up parsers.
- 3. Study about Syntax directed translation schemes.
- 4. Introduce LEX and YACC tools.
- 5. Learn different methods for both Machine Dependent and Independent Code Optimization.
- 6. Learn to develop algorithms to generate code for a target machine.

# **Course Outcomes:**

By the end of the course, the student will:

- 1. Design, develop, and implement a compiler. And also use LEX tool for developing a scanner for the given language.
- 2. Design and implement LL and LR parsers. And also use YACC tool for developing a parser for the given language.
- 3. Study about synthesized and inherited attributes and also generate different types of intermediate code forms.
- 4. Design the good symbol table, to access easily. And also apply machine dependent code optimization techniques.
- 5. Apply algorithm to generate machine code, and also apply machine dependent code optimization techniques.

# UNIT I

Language Processors: Introduction, the structure of a compiler, the science of building a compiler, programming language basics.

**Lexical Analysis:** The role of the lexical analyzer, input buffering, recognition of Tokens, the lexical analyzer generator *lex* program specification, finite automata, from regular expressions to automata, design of a lexical-analyzer generator, optimization of DFA-based pattern matchers.

## UNIT II

**Syntax Analysis:** Introduction, context-free grammars (CFG), derivation, top-down parsing, recursive and non recursive top down parsers, bottom-up parsing, Operator precedence parser, **Introduction to LR parsing:** simple LR parser, more powerful LR parsers, using ambiguous grammars, parser hierarchy, and automatic parser generator YACC tool.

#### UNIT III

**Syntax-Directed Definitions**: Introduction, evaluation orders for SDD's, applications of syntax-directed translation, syntax-directed translation schemes, and implementing L-attributed SDD's. **Intermediate-Code Generation:** variants of syntax trees, three-address code, types and declarations, type checking, control flow statements, switch-statement, and procedures.

#### **UNIT IV**

Run-Time Environments: Storage organization, stack allocation of space, access to nonlocal data on the stack, heap management, introduction to garbage collection, introduction to tracebased collection.

Machine Independent Code optimizations: The principal sources of optimization, introduction to data-flow analysis, foundations of data-flow analysis, constant propagation, partialredundancy elimination, and loop optimization in flow graphs.

## **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, a simple code generator. **Machine Dependent Code Optimizations:** peephole optimization, register allocation and assignment, code generation algorithm.

# **Text Books:**

- 1. Compilers: Principles, Techniques and Tools, Second Edition, Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffry D. Ullman, Pearson.
- 2. Compiler Construction-Principles and Practice, Kenneth C Louden, Cengage Learning.

#### **Reference Books:**

- 1. Modern compiler implementation in C, Andrew W Appel, Revised edition, Cambridge University Press.
- 2. Elements of Compiler Design, A.Meduna, Auerbach Publications, Taylor and Francis Group.
- 3. The Theory and Practice of Compiler writing, J. P. Tremblay and P. G. Sorenson, TMH.
- 4. Writing compilers and interpreters, R. Mak, 3rd edition, Wiley student edition.