GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY
COMPILER DESIGN

Course Code: GR20A4048 **L/T/P/C: 3/0/0/3**
IV Year I Semester

Course Objectives:

1. Understand the fundamental principles in compiler design and to provide the skills needed for building compilers for various situations that one may encounter in a career in Computer Science.
2. Explore the algorithms and data structures involved in the design and construction of compilers.
3. Introduce the major concept in the areas of language translation and compiler design.
4. Develop an awareness of the function and complexity of modern compilers.
5. Enrich the knowledge in various phases of compiler ant its use, code optimization techniques, machine code generation, and use of symbol table.

Course Outcomes:

1. Understand the basic concepts of compiler design, and its different phases.
2. Understand the different types of parsing techniques and should be in a position to solve the problem.
3. Analyze the program and minimize the code by using optimizing techniques which helps in reducing the number of instructions in a program and also utilization of registers in an effective way.
4. Learn the process of translating a modern high-level language to executable code.
5. Construct new tools for compilation for small programming languages.

UNIT I

Overview of Compilation: Phases of Compilation – Lexical Analysis, Regular Grammar and regular expression for common programming language features, pass and phases of translation, interpretation, bootstrapping, data structures in compilation – LEX/ lexical analyzer generator.

UNIT II

Top down Parsing: Context-free grammars, Top down parsing – Backtracking, LL(1), Recursive Descent Parsing, Predictive parsing, preprocessing steps required for predictive parsing.

Bottom up Parsing: Shift Reduce parsing, LR and LALR parsing, Error recovery in parsing, handling ambiguous grammar, YACC – automatic parser/ generator.

UNIT III

Semantic Analysis: Intermediate forms of source programs – abstract syntax tree, polish notation and three address codes. Attributed Grammars, Syntax Directed Translation, Conversion of popular programming languages constructs into Intermediate code forms, Type checker.

Symbol Tables: Symbol table format, organization for block structures languages, hashing, tree structures representation of scope information.

UNIT IV

Block Structure and Non-Block Structure Storage Allocation: Static, Runtime stack and heap storage allocation, storage allocation for arrays, strings and records.

Code Optimization: Consideration for optimization, scope of optimization, local optimization, loop optimization, frequency reduction, folding, DAG representation.

UNIT V

Data Flow Analysis: Flow graph, data flow equation, global optimization, redundant sub expression elimination, Induction variable elements, Live variable analysis, Copy propagation.

Object Code Generation: Object code forms, machine dependent code optimization, register allocation and assignment, generic code generation algorithms, DAG for register allocation.

Text Books:

1. Principles of Compiler Design -A.V. Aho,J.D.Ullman, Pearson Education.
2. Modern Compiler Implementation in C-Andrew N. Appel, Cambridge University Press.

References:

1. Lex&Yacc – John R. Levine, Tony Mason, Doug Brown,O’reilly
2. Modern Compiler Design- Dick Grune, Henry E. Bal, Cariel T. H. Jacobs,Wileydreamtech.
3. Engineering a Compiler-Cooper & Linda,Elsevier.
4. Compiler Construction- Louden,Thomson.