Indian Institute of Information Technology, Allahabad

Compiler Design (COD - 632)
Theory –Syllabus (No. of Credits: 3)

Compiler Structure: analysis-synthesis model of compilation, various phases of compiler, other related basic concepts related to compilers such as interpreters, preprocessors, macros etc.

Lexical Analysis & concepts related to Regular expressions and Finite Automata relevant to Compiler construction Syntactic specification of Languages: Context Free Grammar/ language, ambiguity, associatively, precedence, basic parsing techniques, LEX.

Top-down parsing: Backtracking parser, drawbacks, Top-down parser without backtracking: LL (1) parsing, Problem of Left recursion, Left factoring of Common prefixes, problem solving.

Bottom-up parsing: Handle of a rightmost sentential form, Shift-reduce parsing, LR (0) parsing, Conflicts, SLR (1) parsing, limitations, LR(1) and LALR(1) parsing, problem solving, YACC.

Semantic Analysis and Syntax Directed Translation: Static & Dynamic Checks, Typical Semantic errors, Scoping, Type Checking; Syntax directed definitions (SDD) & Translation (SDT), Attribute Types: Synthesized & Inherited, Annotated Parse Tree, S-attributed and L-attributed grammar, Ordering the evaluation of Attributes, Applications of syntax directed translation.

Symbol Table Design: Function of Symbol Table (ST), Information provided by ST, Attributes of ST, Data Structures for ST: Unsorted list, Sorted list, Linked list, Search trees, Hash table; Scoping, Methods to deal with Scope.

Intermediate Code Generator: High-level and Low-level Intermediate representation, Syntax tree & DAG representations, Three-address code, Quadruples, Triples, Indirect-triples, SDT for intermediate code, Intermediate code generation for control flow, boolean expressions and procedure calls; Short-circuit code, Back patching and Introduction to run-time environments.

Code Optimization: Criteria for code improving transformation, Basic blocks, Flow graphs, Function-Preserving Transformations: common subexpression elimination, copy propagation, dead-code elimination and constant folding; Loop optimizations: Code motion, Induction-variable elimination and Reduction in strength; Peephole optimization e.g. Flow-of-Control optimization, Algebraic simplification; Data flow analysis.

References:

1. A.V. Aho, M.S. Lam, R. Sethi and J.D. Ullman, “Compilers: Principles, Techniques and Tools, 2nd Ed.,” Pearson.

2. K.C. Louden, “Compiler Construction: Principles and Practice,” CENGAGE Learning.

3. J.R. Levine, T. Mason and D. Brown, “Lex and Yacc”, O’Reilly.