

Uttarakhand Technical University, Dehradun
New Scheme of Examination as per AICTE Flexible Curricula
Computer Science and Engineering, VI-Semester
CS 602 Compiler Design

Course Objectives:

1. This course is designed to provide a comprehensive knowledge of Compiler Construction.
2. To learn how to construct compiler to translate High Level Languages to Machine Language.
3. To learn different phases of compiler and how to implement them.
4. To learn efficient machine Language Code Generation using the techniques of Optimization.

Course Learning Outcomes:

1. Understands compiler and various phases in compilation.
2. Understands Lexical Analysis and implement it using LEX tool.
3. Understands LL, LR, and SLR parsing techniques.
4. Implement parsing using YACC tool.
5. Understands Syntax Directed Translation, Symbol Tables and their applications.
6. Understands Intermediate Code Generation and Code Optimization.

Unit-I Introduction to compiling & Lexical Analysis

Introduction of Compiler, Major data Structure in compiler, types of Compiler, Front-end and Back-end of compiler, Compiler structure: analysis-synthesis model of compilation, various phases of a compiler, Lexical analysis: Input buffering , Specification & Recognition of Tokens, Design of a Lexical Analyzer Generator, LEX.

Unit-II Syntax Analysis & Syntax Directed Translation

Syntax analysis: CFGs, Top down parsing, Brute force approach, recursive descent parsing, transformation on the grammars, predictive parsing, bottom up parsing, operator precedence parsing, LR parsers (SLR, LALR, LR), Parser generation. Syntax directed definitions: Construction of Syntax trees, Bottom up evaluation of S-attributed definition, L-attribute definition, Top down translation, Bottom Up evaluation of inherited attributes Recursive Evaluation, Analysis of Syntax directed definition.

Unit-III Type Checking & Run Time Environment

Type checking: type system, specification of simple type checker, equivalence of expression, types, type conversion, overloading of functions and operations, polymorphic functions. Run time Environment: storage organization, Storage allocation strategies, parameter passing, dynamic storage allocation, Symbol table, Error Detection & Recovery, Ad-Hoc and Systematic Methods.

Unit –IV Code Generation

Intermediate code generation: Declarations, Assignment statements, Boolean expressions, Case statements, Back patching, Procedure calls Code Generation: Issues in the design of code generator, Basic block and flow graphs, Register allocation and assignment, DAG representation of basic blocks, peephole optimization, generating code from DAG.

Unit –V Code Optimization

Introduction to Code optimization: sources of optimization of basic blocks, loops in flow graphs, dead code elimination, loop optimization, Introduction to global data flow analysis, Code Improving transformations ,Data flow analysis of structure flow graph Symbolic debugging of optimized code.

References:

1. A. V. Aho, R. Sethi, and J. D. Ullman. Compilers: Principles, Techniques and Tools , Pearson Education
2. Raghavan, Compiler Design, TMH Pub.
3. Louden. Compiler Construction: Principles and Practice, Cengage Learning
4. A. C. Holub. Compiler Design in C , Prentice-Hall Inc., 1993.
5. Mak, writing compiler & Interpreters, Willey Pub.