RAMAKRISHNA MISSION VIDYAMANDIRA

CBCS Syllabus B.Sc. Computer Science Honours

Semester-V

Credit: 6
Course Type: Discipline Specific Elective

Course Outcome:

i) Define the phases of a typical compiler, including the front- and backend.
ii) To have an in depth understanding of data structure and design principal of a compiler.
iii) To be able to understand various parsing techniques.
iv) Explain the role of a semantic analyzer and type checking; create a syntax-directed definition and an annotated parse tree; describe the purpose of a syntax tree.
v) Explain the role of different types of runtime environments and memory organization for implementation of typical programming languages.
vi) To have a practical idea about designing a compiler using specific tools.

CMSA DSE T: Compiler Design

Credit: 4 **Marks: 50**

Introduction to Compiler and Review of Automata: Compilers – Analysis of the source program, Phases of a compiler, Grouping of Phases – Compiler construction tools, Role of Lexical Analyzer, Input Buffering – Specification of Tokens- design of lexical analysis (LEX), Finite automation, Conversion of regular expression of NDFA – Thompson's Conversion, Derivation - parse tree – ambiguity [10 L]

Syntax Analysis- Parsing: Definition - role of parsers - top down parsing - bottom-up parsing; Left recursion - left factoring - Handle pruning , Shift reduce parsing; LEADING- TRAILING- Operator precedence parsing; FIRST- FOLLOW; Predictive parsing; Recursive descent parsing; LR parsing – LR (0) items - SLR parsing; Canonical LR parsing; LALR parsing. [15 L]

Syntax Directed Translation: SDT definitions; Dependency graph; Attribute Grammar, Synthesized attributes – Inherited attributes; L attribute, S attribute, Semantic rules. Annotated Parse Tree. [5 L]

Intermediate Code Generation: Intermediate Languages - prefix - postfix - Quadruple - triple - indirect triples; Assignment Statements; Boolean Expressions; Case Statements; Back patching – Procedure calls. [10 L]

Code Generation: Issues in the design of code generator; The target machine – Runtime Storage management; Basic Blocks and Flow Graphs; Next-use Information – A simple Code generator; DAG representation of Basic Blocks; Peephole Optimization. [10 L]

Code Optimization: Introduction– Principal Sources of Optimization; Optimization of basic Blocks; Loop Optimization; Introduction to Global Data Flow Analysis; Runtime Environments – Source Language issues; Storage Organization; Storage Allocation strategies – Access to non-local names; Parameter Passing. [10 L]

CMSA DSE P: Compiler Design Laboratory

Credit: 2 **Marks: 25**

Writing programs to recognize numbers, identifiers, token; [10 L]

Introduction to with lex; bison, yacc. [15 L]

Designing lexical analyzer using lex or flex; Designing predictive parser, LALR Parser; Generating machine codes. [15 L]

Recommended Books:

1. Compilers: Principles and Tools by Aho, Ullman, Sethi, Lam; 2nd Edition; Pearson.
2. Compiler Design in C by Holub; 1st Edition; Pearson.
3. Theory and Practice of Compiler Writing by Tremblay, Sorenson; 2nd Edition; MacGrawHill.
4. Lex and Yacc by Levine, Mason; 2nd Edition; O'reilly/SPD.
5. Flex and Bison- Text Processing Tools by Levine 1st Edition; O'reilly/SPD.