|  |  |  |  |
| --- | --- | --- | --- |
| Lecture plan  COMPILER DESIGN (RCS – 602) | | | |
| Unit | Topic | Lecture |  |
| Unit I | Introduction to Compiler | Lecture 1 |  |
| Phases and passes | Lecture 2 |  |
| Finite state machines and regular expressions | Lecture 3 |  |
| Optimization of DFA-Based Pattern Matchers implementation of lexical analyzers | Lecture 4 |  |
| lexical-analyzer generator, LEX compiler | Lecture 5 |  |
| Bootstrapping, CROSS COMPILERS | Lecture 6 |  |
| Formal grammars and their application to syntax analysis, BNF notation, ambiguity | Lecture 7 |  |
| YACC, Context free grammars | Lecture 8 |  |
| Derivation and parse trees, capabilities of CFG. | Lecture 9 |  |
| Unit II | Basic Parsing Techniques: Parsers, Shift reduce parsing | Lecture 10 |  |
| operator precedence parsing | Lecture 11 |  |
| top down parsing, predictive parsers | Lecture 12 |  |
| Automatic Construction of efficient Parsers: LR parsers | Lecture 13 |  |
| the canonical Collection of LR(0) items | Lecture 14 |  |
| constructing SLR parsing tables | Lecture 15 |  |
| constructing Canonical LR parsing tables | Lecture 16 |  |
| Constructing LALR parsing tables | Lecture 17 |  |
| using ambiguous grammars, an automatic parser generator, | Lecture 18 |  |
|  | implementation of LR parsing tables | Lecture 19 |  |
| Unit III | Syntax-directed Translation schemes | Lecture 20 |  |
| Implementation of Syntax-directed Translators | Lecture 21 |  |
| Intermediate code, postfix notation, Parse trees & syntax trees | Lecture 22 |  |
| three address code, quadruple & triples | Lecture 23 |  |
| translation of assignment statements, Boolean expressions | Lecture 24 |  |
| statements that alter the flow of control, postfix translation | Lecture 25 |  |
| translation with a top down parser | Lecture 26 |  |
| Array references in arithmetic expressions, procedures call | Lecture 27 |  |
| declarations and case statements | Lecture 28 |  |
| Unit IV | Data structure for symbols tables | Lecture 29 |  |
| representing scope information | Lecture 30 |  |
| Implementation of simple stack allocation scheme | Lecture 31 |  |
| storage allocation in block structured language | Lecture 32 |  |
| Error Detection & Recovery | Lecture 33 |  |
| Lexical Phase errors, syntactic phase errors semantic errors | Lecture 34 |  |
| Unit V | Code Generation: Design Issues, the Target Language | Lecture 35 |  |
| Addresses in the Target Code | Lecture 36 |  |
| Basic Blocks and Flow Graphs | Lecture 37 |  |
| Optimization of Basic Blocks, Code Generator | Lecture 38 |  |
| Code optimization: Machine-Independent Optimizations | Lecture 39 |  |
| Loop optimization, DAG representation of basic blocks | Lecture 40 |  |
| value numbers and algebraic laws | Lecture 41 |  |
| Global Data-Flow analysis. | Lecture 42 |  |
| References:  1. K. Muneeswaran,Compiler Design,First Edition,Oxford University Press.  2. J.P. Bennet, “Introduction to Compiler Techniques”, Second Edition, Tata McGraw-Hill,2003.  3. Henk Alblas and Albert Nymeyer, “Practice and Principles of Compiler Building with C”, PHI, 2001.  4. Aho, Sethi & Ullman, "Compilers: Principles, Techniques and Tools”, Pearson Education  5. V Raghvan, “ Principles of Compiler Design”, TMH  6. Kenneth Louden,” Compiler Construction”, Cengage Learning.  7. Charles Fischer and Ricard LeBlanc,” Crafting a Compiler with C”, Pearson Education | | |  |