****

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**Lesson Plan for the Academic year 2019-20**

**Faculty: Dr. R GURU Semester Starting: 06.01.2020**

**Class & Section: IV Sem ‘A & B’ Section Semester Ending: 25.04.2020**

**Subject with Code: Complier Design (CS420)**

**Pre-requisite:** **Theory of Computations.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl. No** | **Date** | **Topic covered** | **Course Outcomes** | **Remarks** |
| **Unit - 1: INTRODUCTION, LEXICAL ANALYSIS** | | | |  |
| 1 |  | Introduction to Language processors. | CO1  CO2 |  |
| 2 |  | The structure of Compilers. |  |
| 3 |  | **LEXICAL ANALYSIS:** The role of Lexical Analyzer, |  |
| 4 |  | Input Buffering. |  |
| 5 |  | Specifications of Tokens. |  |
| 6 |  | Recognition of Tokens. |  |
| 7 |  | LEX—its structure and usage. |  |
| 8 |  | Problems on Specifications and Recognition of Tokens. |  |
| **Unit -2: SYNTAX ANALYSIS-I** | | | |  |
| 9 |  | Introduction. The role of the parser. | CO1  CO2  CO3 |  |
| 10 |  | Writing a Grammar. |  |
| 11 |  | **TOP-DOWN PARSING**: Introduction to top-down parser. |  |
| 12 |  | Recursive descent parser. |  |
| 13 |  | Predictive parser. |  |
| 14 |  | Nonrecursive predictive parser. |  |
| 15 |  | LL(1) grammar. |  |
| 16 |  | **BOTTOM-UP PARSING:** Introduction to Bottom-up parser, handles. |  |
| 17 |  | Handle pruning, shift reduce parser. |  |
| 18 |  | **LR PARSERS:** Introduction to LR parser. |  |
| 19 |  | Items and the LR(0) Automaton. |  |  |
| 20 |  | The LR-Parsing Algorithm, Constructing SLR-Parsing Tables. |  |
| **Unit -3: SYNTAX ANALYSIS-II** | | | |  |
| 21 |  | Canonical LR (1) Items, Constructing LR (1) Sets of Items. | CO1  CO2  CO3 |  |
| 22 |  | Canonical LR (1) Parsing Tables. |  |
| 23 |  | Constructing LALR Parsing Tables. |  |
| 24 |  | LEX and YACC. |  |
| 25 |  | **SYNTAX-DIRECTED DEFINITIONS** Syntax directed definitions. |  |
| 26 |  | Evaluating an SDD at the Nodes of a Parse Tree. |  |
| 27 |  | Evaluation order for SDDs. |  |
| 28 |  | S-Attributed Definitions, L-Attributed Definitions |  |
| 29 |  | Applications of Syntax-directed translation, Construction of Syntax Trees. |  |
| 30 |  | The Structure of a Type. |  |
| 31 |  | Syntax-directed translation schemes. Postfix Translation Schemes. |  |
| 32 |  | Parser-Stack Implementation of Postfix SDT's. |  |
| **Unit -4 : INTERMEDIATE CODE GENERATION** | | | |  |
| 33 |  | Variants of syntax trees. | CO1  CO2  CO4 |  |
| 34 |  | Three-address code. |  |
| 35 |  | Types and declarations. |  |
| 36 |  | Translation of expressions. |  |
| 37 |  | Type checking. |  |
| 38 |  | Control flow: Boolean Expressions, Short-Circuit Code. |  |
| 39 |  | Flow-of-Control Statements, Control-Flow Translation of Boolean Expressions. |  |
| 40 |  | Back patching. |  |
| 41 |  | Switch statements. |  |
| 42 |  | Intermediate code for procedures. |  |
| **Unit -5** : **CODE GENERATION** | | | |  |
| 43 |  | Issues in the design of Code Generator, | CO1  CO5 |  |
| 44 |  | The Target language. |  |
| 45 |  | Addresses in the target code: Static Allocation, Stack Allocation. |  |
| 46 |  | Run-Time Addresses for Names. |  |
| 47 |  | Basic blocks and |  |
| 48 |  | Flow graphs, |  |
| 49 |  | Optimization of basic blocks: The DAG Representation of Basic Blocks, Finding Local Common Sub expressions, Dead Code Elimination. |  |
| 50 |  | The Use of Algebraic Identities, Representation of Array References, Pointer Assignments and Procedure Calls. |  |
| 51 |  | A Simple Code Generator: Register and Address Descriptors, The Code-Generation Algorithm |  |
| 52 |  | Design of the Function *getReg.* |  |

**Course Outcomes:** After completing this course, students should be able to:

**CO1:** Understand the fundamentals of compiler.

**CO2:** Analyze and evaluate the given CFG and Syntax-Directed Definitions.

**CO3:** Design paring table for the given grammar using various parsing approaches.

**CO4:** Generate various forms of intermediate code for the given language constructs.

**CO5:** Apply concepts of intermediate code for generating assembly code.

**Text Book:**

1. Alfred V Aho, Monica S. Lam, Ravi Sethi, Jeffrey D Ullman: Compilers- Principles, Techniques and Tools, 2nd Edition, Pearson education, 2014.

**Reference Books:**

1. Kenneth C Louden: Compiler Construction - Principles & Practice, First Edition, Brooks/Cole, CENGAGE learning, 1997.
2. Andrew W Appel: Modern Compiler Implementation in C, First Edition, Cambridge University Press, 2010.

Signature of staff Signature of H.O.D