***CS113 Compiler Design***

**Objectives –** The Objectives of this course is to explore the principles, algorithms, and data structures involved in the design and construction of compilers. Topics include context-free grammars, lexical analysis, parsing techniques, symbol tables, error recovery, code generation, and code optimization.

**Prerequisites** – Brief knowledge of programming languages, data structure, and algorithm design

**Outcome**- After completion of this course each student will implement a compiler for a small programming language.

**UNIT I Lectures: 12**

**Introduction to Compilers:** Compilers and translators, the phases of a compiler, Compiler writing tools, The Lexical and Syntactic structure of a language, operators, Assignment statements and parameter translation.

**Lexical Analysis:** The role of the lexical analyzer, Specification of tokens, lexical analysis tool.

**Syntax Analysis:** Role of Parser, CFG, Top-down parsing, bottom-up parser, Operator-precedence parsing, LR Parsers, The Canonical Collection of LR (0) items, Constructing SLR, Canonical LR, and LALR parsing tables, Use of ambiguous grammars in LR parsing, An automatic parser generator, Implementation of LR parsing tables, and constructing LALR sets of items.

**UNIT II Lectures: 10**

**Syntax Directed Translation:** Syntax tree, Bottom-up evolution of S-attributed definitions, L-attributed definition, top-down translation, Bottom-up evaluation of inherited attributed, Recursive evaluators.

**Type Checking:** Static vs. Dynamic Checking, Type expression, Type Checking, Type Equivalence, Type Conversion.

**Symbol Tables:** Structure of Symbol Table, Simple Symbol Table (Linear Table, Ordered List, Tree, Hash Table), Scoped Symbol Table (Nested Lexical Scoping, One Table per Scope, One Table for all Scopes).

**UNIT III Lectures: 8**

**Intermediate Code Generation:** Intermediate Language, Intermediate representation Technique, Three-address code, quadruples and triples, Translation of assignment statements, Boolean expressions, Control Flow, Case Statement, and Function Call.

**Code Generation:** Factors affecting code generation, Basic Block, Code generation for tree, Register Allocation and assignment, DAG representation, Code generation using dynamic programming, code-generator generators.

**UNIT IV Lectures: 12**

**Error Detection and Recovery:** Errors, Lexical- Phase errors, Syntactic- Phase errors, Semantic errors.

**Code Optimization:** Need for optimization, Optimization of Basic Blocks, Loops in flow graph, Optimizing transformation (Compile time evaluation, common sub-expression elimination, Variable Propagation, Code Movement Optimization, Strength Reduction, Dead code optimization, Loop Optimization), Local Optimization, Global Optimization, Computing Global data flow equation, Setting up data flow Equations, Iterative Data Flow Analysis.

**Text Books:**

1. Compilers: Principles, Techniques, and Tools by Alfered V. Aho, Ravi Sethi, Jeffery D. Ullman, Pearson Education.

2. Compiler Design by Santanu Chattopadhyay, PHI

**Reference Book**:

1. Modern Compiler Design by Dick Grune, E. Bal, Ceriel J.H. Jacobs, and Koen G. Langendoen, Wiley Dreamtech