# Compiler Design and Construction

**Course Title:** Compiler Design and Construction  **Full Marks:** 60 + 20 + 20
**Course No:** CSC365  **Pass Marks:** 24 + 8 + 8
**Nature of the Course:** Theory + Lab  **Credit Hrs:** 3
**Semester:** VI

**Course Description:**
This course is designed to develop acquaintance with fundamental concepts of compiler design. The course starts with the basic concepts and also includes different phases of compilers like lexical analysis, syntax analysis, syntax-directed translation, type checking etc. in detail.

**Course Objectives:**

- To develop knowledge in compiler design
- To develop lexical analyzers, parsers, and small compilers using different tools
- To develop lexical analyzers, parsers, and small compilers by using general purpose programming languages.

**Course Contents**:

**Unit 1:** **(3 hrs)**

1.1 Compiler Structure: Analysis and Synthesis Model of Compilation, different sub-phases within analysis and synthesis phases
1.2 Basic concepts related to Compiler such as interpreter, simple One-Pass Compiler, preprocessor, macros, symbol table and error handler.

**Unit 2:** **(22 hrs)**

2.1 Lexical Analysis: Its role, Specification and Recognition of tokens, Input Buffer, Finite Automata relevant to compiler construction syntactic specification of languages, Optimization of DFA based pattern matchers
2.2 Syntax Analysis: Its role, Basic parsing techniques: Problem of Left Recursion, Left Factoring, Ambiguous Grammar, Top-down parsing, Bottom-up parsing, LR parsing
2.3 Semantic Analysis: Static & Dynamic Checks, Typical Semantic errors, Scoping, Type Checking; Syntax directed definitions (SDD) & Translation (SDT), Attribute Types: Synthesized & Inherited, Annotated Parse Tree, S-attributed and L-attributed grammar, Applications of syntax directed translation, Type Systems, Type Checking and Conversion

**Unit 3:** **(4hrs)**

3.1 Symbol Table Design: Function of Symbol Table, Information provided by Symbol Table, Attributes and Data Structures for symbol table
3.2 Run–time storage management

**Unit 4:** **(16 hrs)**

4.1 Intermediate Code Generator: High-level and Low-level Intermediate representation, Syntax tree & DAG representations, Three-address code, Quadruples, Triples, SDT for intermediate code, Intermediate code generation for Declarations, Assignments, Control Flow, Boolean Expressions and Procedure Calls; Back patching
4.2 Code Generator: Factors affecting a code generator, Target Language, Basic blocks and flow graphs, Dynamic programming code-generation algorithm

4.3 Code Optimization: Need and criteria of Code Optimization, Basic optimization techniques

4.4 Case Studies of some compilers like C compiler, C++ complier

**Laboratory Works:**

The laboratory work develops practical knowledge on different concepts of compiler design. Students should

- Create a project by using lexical analyzer generator or any high level language
- Create a parser by using parser generator or any high level language
- Write programs for intermediate code generation and machine code generation
- Create front end of a compiler and using general purpose programming languages

**Recommended Books:**

1. Compilers Principles, Techniques, and Tools, Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman; Pearson Education
2. Introduction to Automata Theory, Languages, and Computation, Johne E. Hopcroft, Rajeev Motwani, Jeffrey D. Ulman, Pearson Education
3. Advanced Compiler Design and Implementation, Steven Muchnick, Morgan Kaufman Publication