**COMPILER DESIGN (UCT 502)**

|  |  |  |  |
| --- | --- | --- | --- |
| **L** | **T** | **P** | **Cr** |
| **3** | **0** | **2** | **4.0** |

**Course Objectives**: To Gain the working knowledge of the major phases of compilation and develop the ability to use formal attributed grammars for specifying the syntax and semantics of programming languages. Learn about function and complexities of modern compilers and design a significant portion of a compiler.

**Introduction:**Phases of compilation and overview. Lexical Analysis (scanner): Regular languages, finite automata, regular expressions, relating regular expressions and finite automata, scanner generator (lex, flex).

**Syntax Analysis (Parser):**Context-free languages and grammars, push-down automata, LL(1) grammars and top-down parsing, operator grammars, LR(O), SLR(1), LR(1), LALR(1) grammars and bottom-up parsing, ambiguity and LR parsing, LALR(1) parser generator (yacc, bison)

**Semantic Analysis:**Attribute grammars, syntax directed definition, evaluation and flow of attribute in a syntax tree.

**Symbol Table:**Basic structure, symbol attributes and management. Run-time environment: Procedure activation, parameter passing, value return, memory allocation, scope.

**Intermediate Code Generation:**Translation of different language features, different types of intermediate forms.

**Code Improvement (optimization):**control-flow, data-flow dependence etc.; local optimization, global optimization, loop optimization, peep-hole optimization etc.

**Architecture dependent code improvement:**instruction scheduling (for pipeline), loop optimization (for cache memory) etc. Register allocation and target code generation.

**Advanced topics**: Type systems, data abstraction, compilation of Object Oriented features and non-imperative programming languages.

**Lab**

Assignments using Lex and Yaac

**Course Learning Outcomes (CLOs) / Course Objectives (COs):**

After the completion of the course, the student will be able to:

1. In-depth knowledge of working of major phases of compiler.

2. Parser construction using top-down and bottom-up parsing techniques.

3. Classify various parameters passing scheme, explain memory management techniques.

4. Apply code optimization techniques on HLL.

**Text Books:**

1. *Compilers: Principles, Techniques and Tools, V. Aho, R. Sethi and J. Ullman.*
2. *Lex & Yacc, Levine R. John, Tony Mason and Doug Brown*

**Reference Books:**

1. *The Design and Evolution of C++, Bjarne Stroustrup.*