**COMPILER DESIGN**

**COURSE OUTCOMES**

At the end of the course, the student will develop ability to

1. Understand the different phases of compiler, and implement practical aspects of automata theory.
2. Apply the knowledge of different phases of a compiler designing.
3. Apply the syntax and semantic rules to design an error free compiler.
4. Enhance the issues on source languages and storage allocation strategies for dynamic storage system.
5. Acquire the skills to design and program various kinds of compilers.
6. Enhance the code Generation and optimization technology.

**UNIT I**

**Introduction to Compiling**

Compiler, Analysis of the source program, The phases of a compiler, Cousins of the compiler, The grouping of phases, Compiler writing tools.

**Lexical Analysis**

The role of the lexical analyzer, Specification of tokens.

**UNIT II**

Recognition of tokens, A Language for specifying lexical Analyzers, Finite automata, Optimization of DFA-based pattern matchers.

**Syntax Analysis**

The role of a parser, Context-free grammars, Writing a grammar, Parsing.

**UNIT III**

Ambiguous grammar, Elimination of Ambiguity, Classification of parsing techniques – Top down parsing –Back Tracking, Recursive Descent parsing, First and Follow- LL Grammars, Non-Recursive descent parsing – Error recovery in predictive parsing.

**UNIT IV**

LR grammars, Bottom Up parsing – LR Parsers – Model of an LR Parsers – SR parsing, Operator Precedence Parsing. SLR parsing, CLR parsing, LALR parsing.

**UNIT V**

Error recovery in LR Parsing, handling ambiguous grammars.

**Syntax Directed Translation**

Syntax-directed definition, S-attributed definitions, L-attributed definitions, Attribute grammar, S-attributed grammar, L-attributed grammar.

**Semantic Analysis**

Type Checking, Type systems, Type expressions, Equivalence of type expressions.

**UNIT VI**

**Intermediate Code Generation**

Construction of syntax trees, Directed acyclic graph, three address codes.

**Runtime Environments**

Storage organization, Storage-allocation strategies, Symbol tables, Activation records.

**Code Optimization**

The principal sources of optimization, Basic blocks and Flow graphs, data-flow analysis of flow graphs.

**TEXT BOOKS**

1. Alfred V.Aho, Ravi Sethi and Jeffry D. Ullman “Compiler Principles, Techniques and Tools”16th Indian Reprint, Pearson Education Asia, ISBN No.81-7808-046-X.,2004.
2. D.M.Dhamdere ”Compiler Construction“, 2nd Edition ” Mac Mellon India Ltd”, ISBN No.0333 -90406-0,1997

**REFERENCE BOOKS**

1. Donovan,”Systems programming”, Mc. Graw Hill.
2. Leland L. Beck, “System Software – An Introduction to Systems Programming” Addison Wesley.

**WEB LINKS**

1. books.google.co.in [Computers](http://www.google.co.in/url?url=http://www.google.co.in/search%3Ftbo%3Dp%26tbm%3Dbks%26q%3Dsubject:%2522Computers%2522&rct=j&sa=X&ei=5IrMT56oIMbsrAemos2sDg&ved=0CE0Q6QUoADAA&q=Leland+L.+Beck,+%E2%80%9DSystem+Software+%E2%80%93+An+Introduction+to+Systems+Programming%E2%80%9DAddison++++++Wesley.+&usg=AFQjCNE-O0pueXVWCWTNaEhd58hQOF6P_w) [Programming](http://www.google.co.in/url?url=http://www.google.co.in/search%3Ftbo%3Dp%26tbm%3Dbks%26q%3Dsubject:%2522Computers%2BProgramming%2522&rct=j&sa=X&ei=5IrMT56oIMbsrAemos2sDg&ved=0CE4Q6QUoATAA&q=Leland+L.+Beck,+%E2%80%9DSystem+Software+%E2%80%93+An+Introduction+to+Systems+Programming%E2%80%9DAddison++++++Wesley.+&usg=AFQjCNHQhqfSKzIU-m2KVxsaqeQ9Lla1eQ) General
2. ww.amazon.com  Books Computers and Technology
3. <http://nptel.iitm.ac.in>

**COMPILER DESIGN LAB**

**COURSE OUTCOMES**

At the end of the course, the student will develop ability to

1. Apply tools, design the compiler and expose the different parsing paradigms
2. Enhance the code optimization techniques
3. Design the symbol tables; implement practical aspects of automata theory.
4. Gain Knowledge of powerful compiler generation tools.
5. Apply the syntax and semantic rules to design an error free compiler.
6. Enhance the issues on source languages and storage allocation strategies for dynamic storage system.
   * + 1. Programs using Lex Tool
   1. Lex specification to demonstrate different regular expressions.
   2. Lex specification to print two digit numbers in words.
   3. Lex specification to check the validity of given date.

* + - 1. Programs using Lex Tool
  1. Lex specification to convert given octal number into decimal equivalent.
  2. Lex specification to count no of vowels, consonants, characters, words and lines in a file.
     + 1. Programs using Yacc Tool
  3. Yacc specification to demonstrate different grammars.
  4. Yacc specification to find sentence validity.
  5. Yacc specification to evaluate expressions using precedence.

* + - 1. Programs using Yacc Tool
  1. Yacc specification to convert binary numbers to decimal numbers
  2. Yacc specification to check the validity of given date.

* + - 1. Program to find all meaningful words and generate the tokens for the given input program.
      2. Implementing lexical analyzer using C.
      3. Implementing Symbol Table for given HLL.
      4. Implementing Shift reduce parser.
      5. Implementing Simple LR parser.
      6. Implementing LALR Parser.
      7. Write a program to generate machine code for restricted programming expressions.
      8. Experiments on code optimization of programming expressions.