L	T	P	C
3	0	0	3

Course Code: CSE320

COMPILER DESIGN

Course Objective:
This course will help the learner to explain various phases in translating source language to target language construct scanner and parser, intermediate code generation and identify the opportunities for optimization.

UNIT - I **11 Periods**

Introduction: Languages Processors – Structure of Compiler – Applications of Compiler Technology – Programming Language basics - **Lexical Analysis:** Role of Lexical Analyzer - Input Buffering- Specifications and recognition of tokens – Lexical – Analyzer generator Lex- Finite Automata – From regular expressions to Automata – Design of a Lexical Analyzer Generator – Optimization of DFA Based pattern.

UNIT - II **11 Periods**

Syntax Analysis – Introduction – Context Free grammars - Writing a Grammar – Top Down Parsing – Bottom up parsing – Simple LP – Canonical LR parsers – Parsers generators YACC **Symbol Table -** Basic structure – use of Symbol table.

UNIT - III **12 Periods**

Syntax Directed Translation: Syntax Directed Definitions – Evaluation orders for SDD's -Applications of Syntax Directed translation – Syntax Directed Translation Schemes **Intermediate Code generation**: Variants of Syntax trees – Three Address code – types and Declarations – Translation of Expression – Type Checking - **Runtime Environments**: Stack allocation of space – Access to Non local Data on the stack – Heap Management.

UNIT - IV **11 Periods**

Code Generation: Issues in code generator – Basic Blocks and Flow graphs – Optimization for Basic Blocks – Peephole Optimization – Register Allocation and assignment – Machine Independent Optimizations: Principal Sources of optimization - Introduction to Data flow analysis – Foundation of Data Flow analysis.

TEXTBOOKS
1. Alfred V.Aho, Ravi Sethi, Jeffrey D. Ullman, Monica S. Lam. *Compilers: Principles, Techniques and Tools*, Pearson Education, Second Edition, 2006.
2. Levine, John R., Tony Mason, and Doug Brown. *Lex & yacc*, O'Reilly Media, Inc., Second Edition, 2013.

REFERENCES

1. Dick Grune, Kees Van Reewijk, Henry E.Bal, C. J.H. Jacobs, Koen G. Langendoen, *Modern Compiler Design*, Springer, Second Edition, 2012.
2. Das, Vinu V. *Compiler Design using FLEX and YACC*, Prentice Hall of India Learning Pvt.Ltd, 2007.
3. Keith D.Cooper and Linda Torczon. *Engineering a Compiler*, Morgan Kauffman Publishers, Second Edition, 2013.

ONLINE MATERIALS

1. http://nptel.ac.in/courses/Webcourse-contents/IIT-KANPUR/compiler-desing/ui/TOC.html
2. http://nptel.ac.in/courses/106108052/

LEARNING OUTCOMES

Upon successful completion of each unit, the learner will be able to

Unit I	• Describe the phases of compiler • Design and develop scanners using Lex
Unit II	• Construct LL and LR parsers • Use of Symbol table in all phases of compiler
Unit III	• Describe the significance of attribute grammars • Development of intermediate code generation
Unit IV	• Design dependent code generation • Identify the different techniques for code optimization for compiler construction • Elucidate the register allocation process in the backend phase of a compiler

COURSE LEARNING OUTCOMES

Upon successful completion the course, the learner will be able to

- Demonstrate the scanner construction from using Lex
- Develop parser using Lex & YACC
- Apply context sensitive analysis for type Inferencing
- Construct intermediate code representation for a given source code
- Identify appropriate techniques for code optimization
- Explain about the code generation and register allocation components in the backend phase of a compiler