II/IV B. TECH. SECOND SEMESTER

COMPILER DESIGN (Required)

Course Code : CS 4T1	**Credits: 3**
Lecture: 3 periods/week	**Internal assessment: 30 Marks**
Tutorial: 1period/week	**Semester end examination: 70 Marks**

Prerequisites: Formal Language & Automata Theory

Course Objectives:

1. An ability to use of formal attributed grammars for specifying the syntax and semantics of programming languages.
2. Working knowledge of the major phases of compilation, particularly lexical analysis, parsing, semantic analysis, and code generation Course
3. An ability to design and implement a significant portion of a compiler for a language chosen by the instructor.

Course Outcomes:
At the end of this course student will:

CO1) Understand about language processors and its phases.

CO2) Demonstrate about scanning of tokens and perform the syntax analysis by using parsing techniques

CO3) Perform Symantec analysis using attribute grammar and compare different memory management techniques in runtime environment

CO4) Ascertain optimization techniques for intermediate code forms and code generation

Syllabus:

UNIT 1
Overview of language processing: – preprocessors – compiler – assembler – Linkers & loaders, difference between compiler and interpreter- structure of a compiler –phases of a compiler.**Lexical Analysis**: - Role of Lexical Analysis – Input Buffering – Specification of Tokens – Recognition of Token – The Lexical Analyzer Generator Lex.

UNIT 2
Syntax Analysis: – Role of a parser – Context Free Grammar – Top Down Parsing
– Recursive Descent Parsing — Non recursive Predictive Parsing- FIRST and FOLLOW –

LL(1) Grammar – Error Recovery in Predictive Parsing.

UNIT 3

Bottom up Parsing: – Reductions – Handle Pruning - Shift Reduce Parsing - Introduction to simple LR – Why LR Parsers – Model of an LR Parsers — Construction of SLR Tables.

More powerful LR parsers: - Construction of CLR (1) - LALR Parsing tables.

UNIT 4

Runtime Environment: - Storage organization - Stack allocation – Static allocation - Heap management - Parameter passing mechanisms.

Intermediate code: - DAG - Three address code – Quadruples - Triples - Indirect Triples.

UNIT 5

Basic Blocks: – DAG representation of Block. Machine independent code optimization - Common sub expression elimination - Constant folding - Copy propagation -Dead code elimination - Strength reduction - Loop optimization.

Machine dependent code optimization: - Peephole optimization – Register allocation - Instruction scheduling - Inter Procedural Optimization - Garbage collection via reference counting.

Learning Resource

Text Books

1. Compilers: Principles, Techniques and Tools: 2nd Edition, Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ulman; 2nd Edition, Pearson Education.
2. Modern Compiler Implementation in C- Andrew N. Appel, Cambridge University Press

References

1. lex &yacc – John R. Levine, Tony Mason, Doug Brown, O'reilly
2. Modern Compiler Design- Dick Grune, Henry E. Bal, Cariel T. H. Jacobs, Wiley reamtech.
3. Engineering a Compiler-Cooper & Linda, Elsevier.
4. Compiler Construction, Louden, Thomson.
5. Principles of compiler design, V. Raghavan, 2nd ed, TMH, 2011.
6. http://www.nptel.iitm.ac.in/downloads/106108052/