COMPILER DESIGN

III B. Tech. - II Semester	L	T	P	C
Course Code: A3CS25	4	1	-	4

COURSE OVERVIEW:

This course deals with the basic techniques of Compiler Construction and tools that can used to perform Syntax-directed translation of a high-level programming language into an executable code. This will provide deeper insights into the more advanced semantics aspects of programming languages, code generation, machine independent optimizations, dynamic memory allocation, types and their inferences, object orientation. The course is presented to the students by using power point projections, lecture notes, subjective and objective tests, assignments, and laboratory

COURSE OBJECTIVES:

1. To teach concepts of language translation and phases of compiler design
2. To describe the common forms of parsers
3. To inculcate knowledge of parser by parsing LL parser and LR parser
4. To demonstrate intermediate code using technique of syntax directed translation
5. To Illustrate the various optimization techniques for designing various optimizing compilers

COURSE OUTCOMES:

At the end of the course students will be able to:

1. Use compiler construction tools and describes the Functionality of each stage of compilation process
2. Construct Grammars for Natural Languages and find the Syntactical Errors/Semantic errors during the compilations using parsing techniques
3. Analyze different representations of intermediate code.
4. Construct new compiler for new languages.
5. Participate in GATE, PGECET and other competitive examinations

SYLLABUS

UNIT - I

INTRODUCTION TO COMPILERS: Definition of compiler, interpreter and its differences, the phases of a compiler, role of lexical analyzer, regular expressions, finite automata, from regular expressions to finite automata, pass and phases of translation, bootstrapping, LEX-lexical analyzer generator.

PARSING: Parsing, role of parser, context free grammar, derivations, parse trees, ambiguity, elimination of left recursion, left factoring, eliminating ambiguity from dangling-else grammar, classes of parsing, top down parsing - backtracking, recursive descent parsing, predictive parsers, LL(1) grammars.

UNIT - II

BOTTOM UP PARSING: Definition of bottom up parsing, handles, handle pruning, stack implementation of shift-reduce parsing, conflicts during shift-reduce parsing, LR grammars, LR parsers-simple LR, canonical LR(CLR) and Look Ahead LR (LALR) parsers, error recovery in parsing, parsing ambiguous grammars, YACC-automatic parser generator.

UNIT - III

SYNTAX DIRECTED TRANSLATION: Syntax directed definition, construction of syntax trees, S-attributed and L-attributed definitions, translation schemes, emitting a translation.

INTERMEDIATE CODE GENERATION: intermediate forms of source programs– abstract syntax tree, polish notation and three address code, types of three address statements and its implementation, syntax directed translation into three-address code, translation of simple statements, Boolean expressions and flow-of-control statements.

UNIT - IV

TYPE CHECKING: Definition of type checking, type expressions, type systems, static and dynamic checking of types, specification of a simple type checker, equivalence of type expressions, type conversions, overloading of functions and operators.

RUN TIME ENVIRONMENTS: Source language issues, Storage organization, storage-allocation strategies, access to non-local names, parameter passing, symbol tables and language facilities for dynamic storage allocation.

UNIT - V

CODE OPTIMIZATION: Organization of code optimizer, basic blocks and flow graphs, optimization of basic blocks, the principal sources of optimization, the directed acyclic graph (DAG) representation of basic block, global data flow analysis.

CODE GENERATION: Machine dependent code generation, object code forms, the target machine, a simple code generator, register allocation and assignment, peephole optimization.

TEXT BOOKS:

1. Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman (2007), Compilers Principles, Techniques and Tools, 2nd edition, Pearson Education, New Delhi, India.

REFERENCE BOOKS:

1. Alfred V. Aho, Jeffrey D. Ullman (2001), Principles of compiler design, Indian student edition, Pearson Education, New Delhi, India.
2. Kenneth C. Louden (1997), Compiler Construction– Principles and Practice, 1st edition, PWS Publishing.
3. K. L. P Mishra, N. Chandrashekaran (2003), Theory of computer science- Automata Languages and computation, 2nd edition, Prentice Hall of India, New Delhi, India.
4. Andrew W. Appel (2004), Modern Compiler Implementation C, Cambridge University Press, UK.