III B.Tech II Semester

15ACS31-COMPILER DESIGN

L	T	P	C
3	1	0	3

Course Objective:

This course is a de facto capstone course in Computer Science, as it combines skills in software design, programming, data structures and algorithms, theory of computing, documentation, and machine architecture to produce a functional compiler.

- *Realize that computing science theory can be used as the basis for real applications Introduce the major concept areas of language translation and compiler design. Learn how a compiler works*
- *Know about the powerful compiler generation tools and techniques, which are useful to the other non-compiler applications*
- *Know the importance of optimization and learn how to write programs that execute faster*

UNIT – I

Overview of Compilation and Language processing:Preprocessor-Compiler-assembler-interpreters-pre-processors-linkers and loaders-structure of a compliler[1]- Phases of Compilation– Lexical Analysis, Regular Grammar andregular expression for common programming language features, pass and Phases of translation, interpretation, bootstrapping, data structures in compilation – LEX lexical analyzer generator.

UNIT – II

Top down Parsing: Context free grammars, Top down parsing–Backtracking, LL (1),recursive descent parsing, Predictive parsing, Preprocessing steps required for predictive parsing.

Bottom up Parsing: Shift Reduce parsing, LR and LALR parsing, Error recovery in parsing ,handling ambiguous grammar, YACC – automatic parser generator.

UNIT – III

Semantic analysis : Intermediate forms of source Programs–abstract syntax tree, polishnotation and three address codes. Attributed grammars, Syntax directed translation, Conversion of popular

Programming languages language Constructs into Intermediate code forms, Type checker.

UNIT – IV

Symbol Tables :Symbol table format, organization for block structures languages, hashing, treestructures representation of scope information. Block structures and non block structure storage allocation: static, Runtime stack and heap storage allocation, storage allocation for arrays, strings and records.

Intermediate code Generation: Intermediate languages, Declarations, Assignment statements, Boolean expressions, Backpatching

Code optimization: Consideration for Optimization, Scope of Optimization, local optimization,loop optimization, frequency reduction, folding, DAG representation.

UNIT – V

Data flow analysis: Flow graph, data flow equation, global optimization, redundant subexpression elimination, Induction variable elements, Live variable analysis, Copy propagation.

Object code generation: Object code forms, machine dependent code optimization, register allocation and assignment generic code generation algorithms, DAG for register allocation.

Course Outcomes :

- *Able to design a compiler for a simple programming language*
- *Able to use the tools related to compiler design effectively and efficiently*
  Can write an optimized code

TEXT BOOKS:

1. Principles of compiler design -A.V. Aho .J.D.Ullman; Pearson Education. (Text book edition)
2. Modern Compiler Implementation in C- Andrew N. Appel, Cambridge University Press.
3. Compilers Principles, Techniques and Tools-Alfred V.Aho, Ravi Sethi, JD Ullman, Pearson Education, 2007.

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 dreamtech.
3. Engineering a Compiler-Cooper & Linda, Elsevier. Compiler Construction, Louden, Thomson.