COMPILER DESIGN **3 0 2 4**

OBJECTIVE

To design the front end of the compiler, scanner, parser, intermediate code generator, object code generator, and the parallel compilation strategies.

UNIT – I FRONT END OF COMPILERS **9+6**

The structure of Compiler – Lexical analysis: Role of Lexical analyzer, Specification and recognition of tokens, Syntax Analysis: Top down parsing, Bottom up parsing, LR Parsers: SLR, CLR, and LALR.

Lab Component: Lexical analyzer generators, Parser generators

UNIT – II INTERMEDIATE CODE GENERATION **9+6**

Syntax Directed Definitions, Evaluation orders for syntax directed definitions, Syntax Directed Translation schemes, Intermediate languages : Three address code, Syntax tree, Postfix code – Declarations – Type checking – Expression translation – Back patching

Lab Component: Intermediate code generation of Expressions, Assignment statements with arrays, Control flow statements, Switch statements.

UNIT – III OBJECT CODE GENERATION **9+6**

Storage organization, Stack allocation space, Access to non-local data on the stack, Heap management - Issues in code generation - Design of code generator - Register allocation and assignment – Instruction selection by tree rewriting – Optimal code generation for expressions – Dynamic programming code generation.

Lab Component: Code generation for any specific architecture supported by open source compilers

UNIT – IV CODE OPTIMIZATION **9+6**

Basic blocks and Flow graphs – Optimization of basic blocks – Principal sources of optimizations – Data flow analysis – Constant propagation – Partial redundancy elimination - Peephole optimizations.

Lab Component: Exploring and customizing different types of optimizations supported by any open source compiler

UNIT – V PARALLELIZING COMPILER 9+6

Basic concepts and examples – Iteration spaces – Affine array indexes – Data reuse – Array data dependence - Finding synchronization free parallelism – Synchronization between parallel loops, Locality optimizations.

Case study : Open source parallelizing compilers.

TOTAL: 45 + 30

TEXT BOOKS:

1. Alfred V. Aho, Monica S.Lam, Ravi Sethi, Jeffrey D.Ullman, “Compilers : Principles, Techniques and Tools”, Second Edition, Pearson Education, 2008.

REFERENCES:

1. Randy Allen, Ken Kennedy, “Optimizing Compilers for Modern Architectures: A Dependence-based Approach”, Morgan Kaufmann Publishers, 2002.
2. Steven S. Muchnick, “Advanced Compiler Design and Implementation”, Morgan Kaufmann Publishers - Elsevier Science, India, Indian Reprint 2003.
3. Keith D Cooper and Linda Torczon, “Engineering a Compiler”, Morgan Kaufmann Publishers Elsevier Science, 2004.
4. V. Raghavan, “Principles of Compiler Design”, Tata McGrawHill Education Publishers, 2010.
5. Allen I. Holub, “Compiler Design in C”, Prentice-Hall software series, 1993