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

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

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


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

UNIT – IV  CODE OPTIMIZATION


Lab Component: Exploring and customizing different types of optimizations supported by any open source compiler
UNIT – V PARALLELIZING COMPILER

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.

TEXT BOOKS:


REFERENCES: