|  |  |
| --- | --- |
| MONO | **DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING****NATIONAL INSTITUTE OF TECHNOLOGY PATNA** Ashok Raj Path, PATNA 800 005 (Bihar), India |
|  Phone No.: 0612 – 2372715, 2370419, 2370843, 2371929, 2371930, 2371715 Fax – 0612- 2670631 Website: [www.nitp.ac.in](http://www.nitp.ac.in/) |

***CS5404 Compiler Design***

**L-T-P-Cr: 3-0-0-3**

**Pre-requisites:** Basics of Fundamental Language and Automata Theory

**Objectives/Overview:**

The Objectives of this course is to explore the principles, algorithms, and data structures involved in the design and construction of compilers. Topics include context-free grammars, lexical analysis, parsing techniques, symbol tables, error recovery, code generation, and code optimization.

**Course Outcomes:**

At the end of the course, a student should:

|  |  |  |
| --- | --- | --- |
| **Sl.No** | **Outcome** | **Mapping to PO** |
|  1 | Have basic knowledge of different phases of a compiler | PO1 |
|  2 | Design of a compiler including its phases and components. | PO2, PO3, PO5 |
|  3 | Develop a large, complex, but well-structured software system that implements various phases of a compiler such as the scanner, parser, code generator, and optimizer.  | PO2, PO3, PO5 |

**UNIT I: Lectures: 13**

**Introduction to Compilers:** Compilers and translators, the phases of a compiler, Compiler writing tools, The Lexical and Syntactic structure of a language, operators, Assignment statements and parameter translation.

**Lexical Analysis:** The role of the lexical analyzer, Specification of tokens, lexical analysis tool.

**Syntax Analysis:** Role of Parser, CFG, Top-down parsing, bottom-up parser, Operator-precedence parsing, LR Parsers, The Canonical Collection of LR (0) items, Constructing SLR, Canonical LR, and LALR parsing tables, Use of ambiguous grammars in LR parsing, An automatic parser generator, Implementation of LR parsing tables, and constructing LALR sets of items.

**UNIT II: Lectures: 10**

**Syntax Directed Translation:** Syntax tree, Bottom-up evolution of S-attributed definitions, L-attributed definition, top-down translation, Bottom-up evaluation of inherited attributed, Recursive evaluators.

**Type Checking:** Static vs. Dynamic Checking, Type expression, Type Checking, Type Equivalence, Type Conversion.

**UNIT III: Lectures: 09**

**Intermediate Code Generation:** Intermediate Language, Intermediate representation Technique, Three-address code, quadruples and triples, Translation of assignment statements, Boolean expressions, Control Flow, Case Statement, and Function Call.

**Code Generation:** Factors affecting code generation, Basic Block, Code generation for tree, Register Allocation and assignment, DAG representation, Code generation using dynamic programming, code-generator generators.

**UNIT IV: Lectures: 10**

**Error Detection and Recovery:** Errors, Lexical- Phase errors, Syntactic- Phase errors, Semantic errors.

**Code Optimization:** Need for optimization, Optimization of Basic Blocks, Loops in flow graph, Optimizing transformation (Compile time evaluation, common sub-expression elimination, Variable Propagation, Code Movement Optimization, Strength Reduction, Dead code optimization, Loop Optimization), Local Optimization, Global Optimization, Computing Global data flow equation, Setting up data flow Equations, Iterative Data Flow Analysis.

**Text/Reference Books**

1. Compilers: Principles, Techniques, and Tools by Alfered V. Aho, Ravi Sethi, Jeffery D. Ullman, Pearson Education.