CS450: DESIGN OF LANGUAGE PROCESSOR

# Credits and Hours:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Teaching Scheme** | **Theory** | **Practical** | **Tutorial** | **Total** | **Credit** |
| Hours/week | 3 | 2 | - | 5 | **4** |
| Marks | 100 | 50 | - | 150 |

**Pre-requisite courses:**

* Digital Electronics
* Operating System
* Theory of Computation

# Outline of the Course:

|  |  |  |
| --- | --- | --- |
| **Sr.**  **No.** | **Title of the unit** | **Minimum number**  **of hours** |
| 1. | Overview of Language Processors & Lexical Analysis | 08 |
| 2. | Syntax Analysis | 08 |
| 3. | Parsing Methods | 08 |
| 4. | Syntax-Directed Translation & Intermediate Code Generation | 08 |
| 5. | Runtime Environment & Code Generation | 08 |
|  | Total hours (Theory) : | 40 |
|  | Total hours (Lab) : | 30 |
|  | Total hours : | 70 |

**Detailed Syllabus:**

|  |  |  |  |
| --- | --- | --- | --- |
| **1.** | **Overview of Language Processors & Lexical Analysis** | **08 Hours** | **15%** |
|  | * Language Processors * The Structure if a Compiler * Application of Compiler Technology   **Lexical Analysis:**   * The Role of Lexical Analyzer * Specification of Tokens * Recognition of Tokens * Lexical Analyzer Generator LEX |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **2.** | **Syntax Analysis** | **08 Hours** | **15%** |
|  | * Role of the Parser * Representative Grammar * Syntax Error Handling * Error-recovery Strategies |  |  |
| **3.** | **Parsing Methods** | **08 Hours** | **30%** |
|  | * + Top Down Parsing: Recursive-Descent Parsing, FIRST and FOLLOW, LL(1) grammar   + Non-recursive Predictive Parsing   + Construction of Non-recursive Predictive Parsing Table   + Error Recovery in Predictive Parsing   + Bottom-up Parsing: Shift-Reduce Parsing, Conflicts during Shift-Reduce Parsing   + Introduction to LR Parsing, L-R Parsing Algorithm, Viable Prefixes   + Simple LR Parser (SLR), Construction of Simple LR Parsing Table   + Canonical LR(1), Construction of LR(1) Parsing Table   + Look Ahead LR (LALR), Construction of LALR Parsing Table   + Parser Generator – Yacc |  |  |
| **4.** | **Syntax-Directed Translation & Intermediate Code Generation** | **08 Hours** | **25%** |
|  | * + Syntax-Directed Definitions   + Dependency Graphs   + S-attributed Definitions   + L-attributed Definitions   + Application of Syntax Directed Translation   + Syntax Directed Translation Schemes   **Intermediate Code Generation:**   * Variants of Syntax Trees * Three Address Code * Control Flow |  |  |
| **5.** | **Runtime Environment & Code Generation** | **08 Hours** | **15%** |
|  | * + Storage Organization   + Activation Trees   + Activation Records   + Calling Sequence   + Heap Management   + Introduction to Garbage Collection   **Code Generation**   * + Issues in Code Generator   + The Target Language   + Basic Blocks and Flow Graphs   + Optimization of Basic Blocks   + A simple Code Generator   + Peephole Optimization |  |  |

# Course Outcome (COs):

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

|  |  |
| --- | --- |
| CO1 | Understand fundamentals of compiler and identify the relationships among different phases of the compiler and use the knowledge of the Lex tool |
| CO2 | Describe Role of Parser and the various error recovery strategies |
| CO3 | Develop the parsers and experiment with the knowledge of different parsers design. |
| CO4 | Design syntax directed translation schemes for a given context free grammar |
| CO5 | Develop semantic analysis scheme to generate intermediate code |
| CO6 | Summarize various optimization techniques used for dataflow analysis and generate machine code from the source code |

# Course Articulation Matrix:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 | PSO1 | PSO2 |
| CO1 | 1 | - | - | - | 2 | - | - | - | - | - | - | - | - | - |
| CO2 | - | 2 | - | 1 | - | - | - | - | - | - | - | - | - | - |
| CO3 | - | - | 3 | - | - | - | - | - | - | - | - | - | - | - |
| CO4 | - | - | 3 | - | - | - | - | - | - | - | - | - | - | - |
| CO5 | - | 1 | - | 2 | - | - | - | - | - | - | - | - | - | - |
| CO6 | - | - | 2 | 1 | - | - | - | - | - | - | - | - | - | - |

Enter correlation levels 1, 2 or 3 as defined below:

1: Slight (Low) 2: Moderate (Medium) 3: Substantial (High)

If there is no correlation, put “-”

# Recommended Study Material:

* **Text book:**
  1. Alfred Aho, Ravi Sethi, Jeffrey D Ullman, “Compilers Principles, Techniques

and- Tools”, Pearson Education Asia.

* 1. M. Dhamdhere, “System Programming and Operating Systems”, Tata

McGraw-Hill.

* 1. Steven S. Muchnick. Advanced Compiler Design and Implementation

# Reference book:

1. Allen I. Holub “Compiler Design in C”, Prentice Hall of India.
2. C. N. Fischer and R. J. LeBlanc, “Crafting a compiler with C”, Benjamin

Cummings.

1. J.P. Bennet, “Introduction to Compiler Techniques”, Second Edition, Tata

McGraw-Hill

1. HenkAlblas and Albert Nymeyer, “Practice and Principles of Compiler

Building with C”, PHI.

1. Kenneth C. Louden, “Compiler Construction: Principles and Practice”,

Thompson Learning.

1. Compiler Construction by Kenneth. C. Louden, Vikas Pub

# Web material:

1. <http://compilers.iecc.com/crenshaw>
2. [http://www.compilerconnection.com](http://www.compilerconnection.com/)
3. [http://dinosaur.compilertools.net](http://dinosaur.compilertools.net/)
4. <http://pltplp.net/lex-yacc>

# Software:

1. LEX
2. YACC