**ITM(SLS) Baroda University**

**School of Computer Science Engineering and Technology**

**BTech Engineering - Semester VI**

**Course Name : Compiler Design**

**Course Code: C2710C2**

**Course Type: Core**

**Teaching Scheme:**

| **Teaching Scheme** | | | **Credits** | **Examination Marks** | | | | | | **Total Marks** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **L** | **T** | **P** | **C** | **Theory Marks** | | | **Practical marks** | | |  |
| **External** | **Internal** | | **External** | **Internal** | |  |
| **3** |  | **2** | **4** | **40** | | **60** | **0** | | **50** | **150** |

**Preamble**-Compiler Design is a fundamental subject of Computer Engineering. Compiler design

principles provide an in-depth view of translation, optimization and compilation of the

entire source program. It also focuses on various designs of compiler and structuring of

various phases of compiler. It is inevitable to grasp the knowledge of various types of

grammar, lexical analysis, yacc, FSM(Finite State Machines) and correlative concepts of

languages.

**Prerequisite:** Algorithms, Data Structures, Assembly Language Program, Theory of Computation,

C/C++ Programming Skills

**Course Objective:**

1. To explore the principles, algorithms, and data structures involved in the design and construction of compilers
2. To understand the design of top-down and bottom-up parsers.
3. To understand syntax directed translation schemes.
4. To introduce LEX and YACC tools.

**Course Learning Outcome:**

|  | **Course Outcome** | **Bloom’s Level** |
| --- | --- | --- |
| CO1 | Understand the basic concepts; ability to apply automata theory and knowledge on  formal languages. | Understanding |
| CO2 | Ability to identify and select suitable parsing strategies for a compiler for various  cases. Knowledge in alternative methods (top-down or bottom-up, etc). | Application |
| CO3 | Understand backend of compiler: intermediate code, Code optimization Techniques and Error Recovery mechanisms | Understanding |
| CO4 | Understand issues of run time environments and scheduling for instruction level parallelism. | Understanding |

After completing the course, the student shall be able to :

**Course Competency:**

**1.** Use compiler construction tools and describes the Functionality of each stage of compilation process

2. Construct Grammars for Natural Languages and find the Syntactical Errors/Semantic errors during the compilations using parsing techniques

**Course Contents:**

| Units | Contents | ***Teaching Hours*** |
| --- | --- | --- |
| 1 | **Overview of the Compiler and its Structure:**  Language processor, Applications of language processors, Definition-Structure-Working of compiler, the science of building compilers, Basic understanding of interpreter and assembler. Difference between interpreter and compiler. Compilation of source code into target language, Cousins of compiler, Types of compiler | 03 |
| 2 | **Lexical Analysis:**  The Role of the Lexical Analyzer, Specification of Tokens, Recognition of Tokens, Input Buffering, elementary scanner design and its implementation (Lex), Applying concepts of Finite Automata for recognition of tokens. | 06 |
| 3 | **Syntax Analysis:**  Understanding Parser and CFG(Context Free Grammars), Left Recursion and Left  Factoring of grammar Top Down and Bottom up Parsing Algorithms, Operator-Precedence Parsing, LR Parsers, Using Ambiguous Grammars, | 07 |
| 4 | **Syntax Analysis :**  Parser Generators, Automatic Generation of Parsers. Syntax-Directed Definitions, Construction of Syntax Trees, Bottom-Up Evaluation of S-Attributed Definitions, L-Attributed Definitions, syntax directed definitions and translation schemes | 05 |
| 5 | **Error Recovery**  Error Detection & Recovery, Ad-Hoc and Systematic Methods | 04 |
| 6 | Intermediate-Code Generation:  Variants of Syntax Trees, Three-Address Code, Types and Declarations, Translation of Expressions, Type Checking, Syntax Directed Translation Mechanisms, Attributed Mechanisms And Attributed Definition. | 05 |
| 7 | **Run-Time Environments:**  Source Language Issues, Storage Organization. Stack Allocation of Space, Access to Nonlocal Data on the Stack, Heap Management, | 04 |
| 8 | **Code Generation and Optimization:**  Issues in the Design of a Code Generator, The Target Language, Addresses in the Target Code, Basic Blocks and Flow Graphs, Optimization of Basic Blocks, A Simple Code Generator, Machine dependent optimization, Machine independent optimization Error detection of recovery | 06 |
|  | **Total** | **40** |

**Reference:**

**1.**

**Text Books:**

|  |  |
| --- | --- |
| **1** | Compiler Tools Techniques - A.V.Aho, Ravi Sethi, J.D.Ullman, Addison Wesley |
| **2** | The Theory And Practice Of Compiler Writing - Trembley J.P. And Sorenson P.G.  Mcgraw-Hill |

**Reference Books:**

|  |  |
| --- | --- |
| **1** | Modern Compiler Design - Dick Grune, Henri E. Bal, Jacob, Langendoen, WILEY India |
|  |  |
|  |  |

**Case Studies:**

| **Sr.No** | **Case Studies** | **Evaluation** |
| --- | --- | --- |
| C1 | Create a Program and try to optimize that code to increase the time complexity. | 1. Identification of algorithm. 2. Report preparation. 3. Presentation with VIVA |

**Simulation and Animation: NA**

**NPTEL Video:**

| **Sr. No** | **About Video** | **Link** | **Topic** |
| --- | --- | --- | --- |
| 01 | This NPTEL video contains the knowledge required for learning basics of Compiler Design | https://archive.nptel.ac.in/courses/106/105/106105190/ | Introduction to Compiler Design |
| 02 | Prof. Sanjeev K Aggarwal, IIT Kanpur | https://archive.nptel.ac.in/courses/106/104/106104123/ | All about Compiler Design |

**Related MOOCs courses**

| **Sr.No** | **MOOC Courses** |
| --- | --- |
| M1 | Introduction to Compiler Construction & Design - Learn all phases of Compiler Construction in a very easy & simple approach by enrolling in this course - Udemy Course |
| M2 | Learn Compiler Construction & Design from Scratch - Udemy Course |

**Sample List of Experiments**

| **Sr No** | **Title of Experiment** |
| --- | --- |
| **1** | Implementation of Finite Automata and String Validation |
| **2** | Introduction to Lex Tool. |
| **3** | Implement following Programs Using Lex  a. Generate Histogram of words  b. Ceasor Cypher  c. Extract single and multiline comments from C Program |
| **4** | Implement following Programs Using Lex  a. Convert Roman to Decimal  b. Check weather given statement is compound or simple  c. Extract html tags from .html file |
| **5** | Implementation of Recursive Descent Parser without backtracking  Input: The string to be parsed.  Output: Whether string parsed successfully or not. Explanation:  Students have to implement the recursive procedure for RDP for a typical grammar. The production no. are displayed as they are used to derive the string. |
| **6** | Finding “First” set  Input: The string consists of grammar symbols.  Output: The First set for a given string.  Explanation:  The student has to assume a typical grammar. The program when run will ask for the string to be entered. The program will find the First set of the given string. |
| **7** | Generate 3-tuple intermediate code for given infix expression |
| **8** | Extract Predecessor and Successor from given Control Flow Graph |
| **9** | Introduction to YACC and generate Calculator Program |
| **10** | Finding “Follow” set  Input: The string consists of grammar symbols.  Output: The Follow set for a given string.  Explanation: |