GUJARAT TECHNOLOGICAL UNIVERSITY

Bachelor of Computer Engineering
Subject Code: 3170701
Semester – VII
Subject Name : Compiler Design

Type of course: Compulsory/Core

Prerequisite: Algorithms, Data Structures, Assembly Language Program, Theory of Computation, C/C++ Programming Skills

Rationale: 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.

Teaching and Examination Scheme:

Teaching Scheme			Credits	Examination Marks				Total Marks
L	T	P	C	Theory Marks		Practical Marks		
				ESE (E)	PA (M)	ESE (V)	PA (I)	
3	0	2	4	70	30	30	20	150

Content:

Sr. No.	Content	Total Hrs
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.	05
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, 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	11

4	**Error Recovery** Error Detection & Recovery, Ad-Hoc and Systematic Methods	**04**
5	**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**
6	**Run-Time Environments:** Source Language Issues, Storage Organization. Stack Allocation of Space, Access to Nonlocal Data on the Stack, Heap Management,	**04**
7	**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**
8	**Instruction-Level Parallelism:** Processor Architectures, Code-Scheduling Constraints, Basic-Block Scheduling, Pass structure of assembler	**04**

Suggested Specification table with Marks (Theory):

Distribution of Theory Marks					
R Level	U Level	A Level	N Level	E Level	C Level
10	25	20	10	05	00

Legends: R: Remembrance; U: Understanding; A: Application, N: Analyze and E: Evaluate C: Create and above Levels (Revised Bloom's Taxonomy)

Note: This specification table shall be treated as a general guideline for students and teachers. The actual distribution of marks in the question paper may vary slightly from above table.

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
2. Compiler Construction - Waite W.N. And Goos G., Springer Verlag
3. Compiler Construction-Principles And Practices - D.M.Dhamdhere, Mcmillian
4. Principles of Compiler Design, V. Raghavan, McGrawHill

GUJARAT TECHNOLOGICAL UNIVERSITY

Bachelor of Computer Engineering
Subject Code: 3170701

Course Outcomes:

After learning the course the students should be able to:

Sr. No.	CO statement	Marks % weightage
CO-1	Understand the basic concepts; ability to apply automata theory and knowledge on formal languages.	35
CO-2	Ability to identify and select suitable parsing strategies for a compiler for various cases. Knowledge in alternative methods (top-down or bottom-up, etc).	25
CO-3	Understand backend of compiler: intermediate code, Code optimization Techniques and Error Recovery mechanisms	25
CO-4	Understand issues of run time environments and scheduling for instruction level parallelism.	15

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:

	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 Follow set of the given string.
11	Implement a C program for constructing LL (1) parsing.
12	Implement a C program to implement LALR parsing.
13	Implement a C program to implement operator precedence parsing.