GUJARAT TECHNOLOGICAL UNIVERSITY

SUBJECT NAME: Compiler Design
SUBJECT CODE: 2170701
B.E. 7th SEMESTER

Type of course: Core

Prerequisite: Data Structures and Algorithms, Theory of Computation,

Rationale: Compiler Design is a fundamental/core subject of Computer Engineering. It teaches how Compiler of a Programming Language works. It also focuses on various designs of Compiler and structuring and optimizing various phases of a Compiler. It is also necessary to learn types of Grammar, Finite state machines, lex, yacc and related 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)	
					PA	ALA	ESE	OEP		
4	0	2	6	70	20	10	20	10	20	150

Content:

Sr. No.	Content	Total Hrs	% Weightag
1	Overview of the Translation Process, A Simple Compiler, Difference between interpreter, assembler and compiler. Overview and use of linker and loader, types of Compiler, Analysis of the Source Program, The Phases of a Compiler, Cousins of the Compiler, The Grouping of Phases, Lexical Analysis, Hard Coding and Automatic Generation Lexical Analyzers, Front-end and Back-end of compiler, pass structure	**08**	15
2	**Lexical Analyzer** Introduction to Lexical Analyzer, Input Buffering, Specification of Tokens, Recognition of Tokens, A Language for Specifying Lexical Analyzers, Finite Automata From a Regular Expression, Design of a Lexical Analyzer Generator, Optimization of DFA	**08**	15
3	**Parsing Theory** Top Down and Bottom up Parsing Algorithms, Top-Down Parsing, Bottom-Up Parsing, 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	**10**	22

4	**Error Recovery** Error Detection & Recovery, Ad-Hoc and Systematic Methods	**06**	**08**
5	**Intermediate Code Generation** Different Intermediate Forms, Syntax Directed Translation Mechanisms And Attributed Mechanisms And Attributed Definition.	**06**	**10**
6	**Run Time Memory Management** Source Language Issues, Storage Organization, Storage-Allocation Strategies, and Access to Non local Names, Parameter Passing, Symbol Tables, and Language Facilities for Dynamic Storage Allocation, Dynamic Storage Allocation Techniques.	**06**	**10**
7	**Code Optimization** Global Data Flow Analysis, A Few Selected Optimizations like Command Sub Expression Removal, Loop Invariant Code Motion, Strength Reduction etc.	**06**	**10**
8	**Code Generation** Issues in the Design of a Code Generator, The Target Machine, Run-Time Storage Management, Basic Blocks and Flow Graphs, Next-Use Information, A Simple Code Generator, Register Allocation and Assignment, The DAG Representation of Basic Blocks, Peephole Optimization, Generating Code from DAGs, Dynamic Programming Code-Generation Algorithm, Code-Generator Generators.	**06**	**10**

Suggested Specification table with Marks (Theory):

Distribution of Theory Marks					
R Level	U Level	A Level	N Level	E Level	C Level
20	**20**	**10**	**10**	**5**	**5**

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.

Reference Books:

1. Compilers: Principles, Techniques and Tools By Aho, Lam, Sethi, and Ullman, Second Edition, Pearson, 2014
2. Compilers: Principles, Techniques and Tools By Aho, Sethi, and Ullman, Addison-Wesley, 1986
3. Compiler Design in C By Allen I. Holub, Prentice-Hall/Pearson.
4. Advanced Compiler Design and Implementation By Muchnick, Morgan and Kaufmann, 1998.

Course Outcome:

After learning the course the students should be able to:

1. Understand the basic concepts and application of Compiler Design

2. Apply their basic knowledge Data Structure to design Symbol Table, Lexical Analyser , Intermediate Code Generation, Parser (Top Down and Bottom Up Design) and will able to understand strength of Grammar and Programming Language.

3. Understand various Code optimization Techniques and Error Recovery mechanisms.

4. Understand and Implement a Parser.

List of Experiments:

1. Design a lexical analyzer for given language and the lexical analyzer should ignore redundant spaces, tabs and new lines. It should also ignore comments. Although the syntax specification states that identifiers can be arbitrarily long, you may restrict the length to some reasonable value. Simulate the same in C language
2. Write a C program to identify whether a given line is a comment or not
3. Write a C program to test whether a given identifier is valid or not.
4. Write a C program to simulate lexical analyzer for validating operators
5. To Study about Lexical Analyzer Generator(LEX) and Flex(Fast Lexical Analyzer)
6. Implement following programs using Lex.
   a. Create a Lexer to take input from text file and count no of characters, no. of lines & no. of words.
   b. Write a Lex program to count number of vowels and consonants in a given input string.
7. Implement following programs using Lex.
   a. Write a Lex program to print out all numbers from the given file.
   b. Write a Lex program to printout all HTML tags in file.
   c. Write a Lex program which adds line numbers to the given file and display the same onto the standard output.
8. Write a Lex program to count the number of comment lines in a given C program. Also eliminate them and copy that program into separate file.
9. Write a C program for implementing the functionalities of predictive parser for the mini language.
10. Write a C program for constructing of LL (1) parsing.
11. Write a C program for constructing recursive descent parsing
12. Write a C program to implement LALR parsing.
13. Write a C program to implement operator precedence parsing.
14. To Study about Yet Another Compiler-Compiler(YACC).
15. Create Yacc and Lex specification files to recognizes arithmetic expressions involving +, -, * and / .
16. Create Yacc and Lex specification files are used to generate a calculator which accepts,integer and float type arguments.

Design based Problems (DP)/Open Ended Problem:

Students can do a mini project in C to implement various phases of a Compiler considering a simple set of Instructions and other assumptions. They can also practice on LEX and YACC for various applications involving different Grammars etc.

Major Equipment: PC, Unix Server/Client.

List of Open Source Software/learning website:

1. nptel.ac.in
2. https://en.wikipedia.org/wiki/Principles_of**_Compiler_Design**
3. https://en.wikipedia.org/wiki/**Compiler_**construction

ACTIVE LEARNING ASSIGNMENTS: Preparation of power-point slides, which include videos, animations, pictures, graphics for better understanding theory and practical work – The faculty will allocate chapters/ parts of chapters to groups of students so that the entire syllabus to be covered. The power-point slides should be put up on the web-site of the College/ Institute, along with the names of the students of the group, the name of the faculty, Department and College on the first slide. The best three works should submit to GTU.