SARVAJANIK UNIVERSITY
INCLUSIVE | INTEGRATED | INNOVATIVE

SARVAJANIK UNIVERSITY
Sarvajanik College of Engineering and Technology
Bachelor of Technology

Year: B. Tech IV (Semester VII)

Subject Name: Compiler Design **Subject Code:** BTCO13701
Type of course: Professional Core Course
Prerequisite (if any): Formal Languages and Automata Theory, Data Structures

Rationale: The objective of this course is to introduce the concepts of language translation and compiler design and to develop an awareness of the function and complexity of modern compilers. This course is a study of the theory and practice required for the design and implementation of compilers for programming languages.

Teaching and Examination Scheme:

Teaching Scheme				Theory Marks			Practical Marks		Total
L	T	P	C	TEE	CA1	CA2	TEP	CA3	
3	0	2	4	60	25	15	30	20	150

CA1: Continuous Assessment (assignments / projects / open book tests / closed book tests) CA2: Sincerity in attending classes / class tests / timely submissions of assignments / self-learning attitude / solving advanced problems TEE: Term End Examination TEP: Term End Practical Exam (Performance and viva on practical skills learned in course) CA3: Regular submission of Lab work / Quality of work submitted / Active participation in lab sessions / viva on practical skills learned in course.

Contents:

Sr. No.	Contents	Total Hours
1	Introduction to Compilers – Language processors and their applications, Role of Compiler, Assembler and Interpreter, Phases of a Compiler, Symbol table, Compilation of source code into target code	08
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, transition graph for NFA, Conversion of NFA to DFA	10
3	Syntax Analysis – Role of Parser, Context Free grammars, Parse trees and derivations, Writing a grammar – eliminating ambiguity, eliminating left recursion, left factoring; Top-down parsing- First and Follow, LL(1) parsing; Bottom-up parsing- shift reduce parsing, Simple LR, syntax directed translation	10

4	Intermediate code generation - Variants of Syntax Trees, Directed Acyclic Graph, Three-Address Code, Types and Declarations, Translation of Expressions, Type Checking, Intermediate code for procedures	05
5	Run-time environment - Source Language Issues, Storage Organization. Stack Allocation of Space, Access to Nonlocal Data on the Stack, Heap Management	04
6	Code generation and Optimization - Issues in the design of a Code Generator, Basic Blocks and Flow Graphs, Optimization of Basic Blocks, Machine dependent optimization, Machine independent optimization, simple code generator	08

Suggested Specification table with Marks (Theory): (For B. Tech only)

Distribution of Theory Marks					
R Level	**U Level**	**A Level**	**N Level**	**E Level**	**C Level**
10	15	20	5	5	5

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

Reference Books:

Sr No	Title of book /article	Author(s)	Publisher and details like ISBN	Year of publication	Publication Edition
1	Compilers: Principles, Techniques and Tools	Aho, Lam, Sethi & Ullman	Pearson Education	2007	2nd edition
2	Compiler Construction – Principles and Practice	Kenneth Louden	Cengage Learning	2012	
3	Compiler Design	O. G. Kakde	Laxmi Publications	2014	4th edition

Course Outcomes (CO):

Sr. No.	CO statements	Marks % weightage
CO-1	Explain the basic concepts of assembler, interpreter and compiler.	20
CO-2	Describe the phases of compiler and its run-time data structures such as symbol table	20
CO-3	Identify and select suitable parsing strategies for a compiler such as top-down or bottom-up	30
CO-4	Explain the issues related to target machine's run time environment, code generation and techniques used for code optimization.	20
CO-5	Demonstrate the use of tools like lex and yacc to generate a simple compiler	10

List of Open learning website:

- https://nptel.ac.in/courses/106104123

List of Experiments:

Sr. No	Practical
1	Write a C Program to Implement Scanner.
2	Write a C Program of Scanner that generates and reports lexical errors and also creates symbol table.
3	Write a C Program that implements symbol table using the concept of hashing.
4	Write a LEX program to check whether input symbol is number or character.
5	Write a LEX program to check whether input is multidigit number or string.
6	Write a LEX program that reads input file and copies only words starting with vowels to output file.

7	Write a LEX program that reads program from input file and counts number of letters,words, lines from it. Resulting count should be written to output file.
8	Write a LEX program that reads a c program from input file and eleminates single line as well as multiline comments from it and copies updates program to output file.
9	Write a LEX program that reads input file and performs following tasks. • generate valid tokens in output1 file • generate invalid token errors in Error file (along with line number) • generate symbol table file that contains only identifiers. • generates output2 file that will content program(from input file) as it is with following eliminate tab space, white space, newline eliminate comments
10	Create Yacc and Lex specification files to recognizes arithmetic expressions involving +, -, and / .
11	Write a C program for constructing of LL (1) parsing.