Compiler Design Lab

COURSE CODE: 18B17CI672

COURSE CREDITS: 2

CORE/ELECTIVE: CORE

: 0-0-4

Pre-requisite: An understanding in Theory of computation, Introduction to any programming language (Preferably, C)

Course Objectives:

1. The lab course provides the complete description about inner working of a compiler.
2. The main focus is on the design of compilers and optimization techniques.
3. The course also aims to convey the language specifications, use of regular expressions and context free grammars behind the design of compiler.
4. It builds an understand ability of various parsing techniques like predictive parsing, LR parsing, LALR parsing. It also focuses on the design of Compiler writing tools.

Course Outcomes:

S. No.	Course Outcomes	Level of Attainment
CO-1	Gain an in-depth understanding of the principles underlying the design	Familiarity
CO-2	Construction of compilers	Familiarity
CO-3	Functioning of Compiler writing tools	Computational skills
CO-4	Building various parsing techniques	Technical skills

List of Experiments:

S. No.	Description	Hours
1	a. Write a program to read and translate integers into numbers. e.g. 1=ONE 12 = ONE TWO 856 = EIGHT FIVE SIX Generate an error if the number of digits is more than 3 b. Write a program to convert infix notation to postfix notation.	4
2	1. Implement a DFA which simulates the regular expression a + (aa)*b. 2. The following rules define the translation of an English word into pig Latin: a) If the word begins with a nonempty string of consonants, move the initial consonant string to the back of the word and add the suffix AY; e.g., pig comes igpay. b) If the word begins with a vowel, add the suffix YAY; e.g., owl becomes owlyay. c) U following a Q is a consonant.	8

	d) Y at the beginning of a word is a vowel if it is not followed by a vowel. e) One-letter words are not changed. Write a C program to generate pigLatin from an English word.	
3	Implementation of Lexical analysis	4
4	Program for computation of FIRST AND FOLLOW of non-terminals.	4
5	Write a program to check whether a grammar is left recursive or not, if it is remove left recursion.	4
6.	Implementation of Predictive Parsing Table Construction	6
7.	Implementation of Shift Reduce Parsing	6
8.	Implementation of Operator Precedence Parsing	6
9.	Implementation of LR Parsing	4
10	Intermediate Code Generation	4
11	Implementation of Code Generation	4
Total Lab hours		**56**

Suggested Books/Resources:

1. Compilers : Principles, techniques and tools, Aho, Sethi and Ullman
2. Compiler design in C, Holub
3. Advanced compiler design and implementation, Muchnick, Morgan and Kauffman

Evaluation Scheme:

1	Mid Sem. Evaluation	20 Marks
2	End Sem. Evaluation	20 Marks
3	Attendance	15 Marks
4	Lab Assessment	45 Marks
	Total	100 marks

Course Outcomes (COs) contribution to the Programme Outcomes(POs)

CO/PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	Average
CO-1	3	2	2	1	2	3	3	2	1	3	2	1	**2.1**
CO-2	3	3	2	1	1	3	3	1	3	3	3	1	**2.3**
CO-3	3	3	2	1	2	2	2	1	3	3	3	2	**2.3**
CO-4	3	3	3	2	2	2	3	2	2	1	2	2	**2.3**
Average	**3**	**2.8**	**2.3**	**1.3**	**1.8**	**2.5**	**2.8**	**1.5**	**2.3**	**2.5**	**2.5**	**1.5**	