MUTHOOT INSTITUTE OF TECHNOLOGY AND SCIENCE
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
COURSE MANUAL

Course: Compiler Lab	**Credits:** 3

Instructor: Sneha Sreedevi/Fasila K A, Department of CSE

Academic Year: 2019-2020

Batch: 2016-2020

Number of students attending the course: 62

Location of instruction

Location	Type of instruction (L/T/P)
OS Lab	Practical

VISION

To be recognized as a socially accountable thought leader in applications of computational technologies and a centre of excellence in solving complex and cross-disciplinary problems.

MISSION

- Develop pedagogical practices that help equip students with the ability to self-learn and reason.
- Improve ability to communicate on complex engineering activities and interpersonal skills by developing proper training methods and professional networking.
- Improve problem solving ability by promoting computational thinking and working on cross-disciplinary projects.
- Create and sustain networks in professional, academic and start up environments to stay updated with changes in the field of Computer Science and Engineering.
- Instill social commitment in students and faculty by engaging in spreading computer literacy and other socially relevant services.
- Promote research towards developing insight into complex problems.

PROGRAMME EDUCATIONAL OBJECTIVES

Based on

1. Preparation
2. Core competence
3. Breadth
4. Professionalism
5. Lifelong learning

Course Context and Overview of Instructional Goals

Compiler design is a course that discusses ideas used in construction of programming language compilers. Students learn how a program written in high level programming language and designed for humans understanding is systematically converted into low level assembly language understood by machines. This course is intended to give the students to implement practically different phases of the compilers, test different optimization techniques and give an exposure to different compiler writing tools.
Industry relevance of the course: Students will be able to debug the the working of compilers for any generic language as per the requirements specified.

Course Objectives

- To implement the different Phases of compiler
- To implement and test simple optimization techniques
- To give exposure to compiler writing tools.

Course outcome

CS431.1	Students will able to **implement** the techniques of Lexical Analysis and Syntax Analysis.
CS431.2	Students will able to **apply** the knowledge of Lex & Yacc tools to develop programs
CS431.3	Students will be able to **generate** intermediate code
CS431.4	Students will be able to **implement** Optimization techniques and generate machine level code.

Course Prerequisites

- CS301 Theory of Computation
- CS303 System Software
- CS304 Compiler Design

Previous Instances of the course

Number of instances - 1

Instructor	Jisha James															
Year	2018-2019															
Outcomes	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
	3	3	3		3				3							
Feedback																

Skills and concepts expected to be acquired

Knowledge	**Comprehension**	**Application**	**Analysis**	**Synthesis**	**Evaluation**
1.Phases of compiler 2. Fundamentals of TOC	-	3. Use of C programs with files and pointers			

COURSE OUTCOMES

After successful completion of the course, students will be able to

	Course Outcomes	BL
CS431.1	Students will able to **implement** the techniques of Lexical Analysis and Syntax Analysis.	Application [3]
CS431.2	Students will able to **apply** the knowledge of Lex & Yacc tools to develop programs.	Application [3]
CS431.3	Students will be able to **generate** intermediate code	Application [3]
CS431.4	Students will be able to **implement** optimization techniques and generate machine level code.	Application [3]

Course Outcome – Program Outcome / Program Specific Outcome correlation Matrix

Course	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO 10	PO 11	PO 12	PSO1	PSO2
CS431.1	2		1		2			1						
CS431.2	2				2			1						
CS431.3	2	2	1					1						
CS4031.4	2	2						1						

Justification for CO-PO/PSO mapping

Mapping	L/M/H	Justification
CS431.1 - PO1,PO3,PO5,PO8	M	The outcome involves representation of language tokens using regular expressions, context free grammar and finite automata. Programming in LEX helps to design lexical analyzer. Programming in YACC helps to know the basic principles of parsing strategies.
CS431.2 – PO1,PO5,PO8	M	Student will be able to get familiarized with the programming concepts of lex and yacc there by applying the knowledge of LEX and YACC tools for developing the lexical and syntax analyzer. The outcome involves the application and analysis of LEX and YACC to develop programs.
CS431.3- PO1,PO2,PO8	M	Students gain the ability to learn about the Intermediate code generation and different representations used in compilation. The knowledge of various intermediate representations helps to generate efficient target code. Selection of suitable intermediated code representations helps the proper usage of registers allocated and used.
CS431.4- PO1,PO2 ,PO8	M	Students will be able to apply the basic knowledge of code optimization and its techniques. The outcome implements the different optimization strategies to develop the machine code.

REFERENCE BOOKS

1.	Aho A. Ravi Sethi and D Ullman. Compilers – Principles Techniques and Tools, Addison Wesley, 2006.
2.	D. M.Dhamdhare, System Programming and Operating Systems,Tata McGraw Hill & Company, 1996.
3.	Kenneth C. Louden, Compiler Construction – Principles and Practice, Cengage Learning Indian Edition, 2006.
4.	Tremblay and Sorenson, The Theory and Practice of Compiler Writing, Tata McGraw Hill & Company, 1984.
5.	John E Hopcroft, Rajeev Motwani and Jeffrey D Ullman, Introduction to Automata Theory, Languages and Computation,3/e,Pearson Education,2007

DETAILS OF WEB SOURCES

1) http://dinosaur.compilertools.net/
2) http://web.mit.edu/gnu/doc/html/flex_1.html
3) https://www.cs.utexas.edu/users/novak/yaccpaper.html

4) http://compilers.iecc.com/crenshaw/
5) https://www.cs.utexas.edu/users/novak/asg-lex.html

LIST OF EXPERIMENTS:

SL.NO	EXPERIMENT	COURSE OUTCOME
1	Design and implement a lexical analyzer for given language using C and the lexical analyzer should ignore redundant spaces, tabs and new lines.	**CS431.1**
2	Implementation of Lexical Analyzer using Lex Tool.	**CS431.2**
3	Generate YACC specification for a few syntactic categories. a) Program to recognize a valid arithmetic expression that uses operator +, – , * and /. b) Program to recognize a valid variable which starts with a letter followed by any number of letters or digits.	**CS431.2**
4	Generate YACC specification for a few syntactic categories c) Implementation of Calculator using LEX and YACC	**CS431.2**
5	Write program to find ε – closure of all states of any given NFA with ε transition.	**CS431.1**
6	Write program to convert NFA to DFA.	**CS431.1**
7	Develop an operator precedence parser for a given language.	**CS431.1**
8	Write program to find Simulate First and Follow of a given grammar	**CS431.1**
9	Construct a recursive descent parser for an expression.	**CS431.1**
10	Write a program to perform constant propagation.	**CS431.4**
11	Implement Intermediate code generation for simple expressions.	**CS431.3**
12	Implement the back end of the compiler which takes the three address code and produces the 8086 assembly language instructions that can be assembled and run using an 8086 assembler. The target assembly instructions can be simple move, add,sub etc	**CS431.4**