Nirma University
Institute of Technology
Computer Science & Engineering Department
<u>Course Policy</u>

B. Tech. Computer Science & Engineering   Sem-VII   Academic Year: 2022-23

Course Code & Name	2CS701 Compiler Construction		
Credit Details	Lectures-3 Hr., Praticals-2 Hr., Credits-4		
Course Coordinator	Prof Monika Shah		
Contact No. & Email	079-71652572, Email: monika.shah@nirmauni.ac.in		
Office	NF8, opposite to N511 lab		
Course Faculty			
	Name	**Contact**	**Office**
	Prof Monika Shah	monika.shah@nirmauni.ac.in 079-71652572	NF8 (opp. N511 lab)
	Dr Tejal Upadhyay	tejal.upadhyay@nirmauni.ac.in 079-71652565	NF43 (beside N506 lab)
	Dr Tarjni Vyas	tarjni.vyas@nirmauni.ac.in 079-71652563	NF3 (opp. N511 lab)
	Dr Sharada Valiveti	sharada.valiveti@nirmauni.ac.in	NN-Office (E building 11th floor)
	Prof Deepti Sarawat	deepti.sarawat@nirmauni.ac.in	NN-C (E building 11th floor)
	Dr Satyabrata Roy	satyabrata.roy@nirmauni.ac.in	NN-C (E building 11th floor)
Course Site	https://lms.nirmauni.ac.in/course/view.php?id=4849		

<u>Introduction to Course:</u>

The course will discuss how a program written in high level language designed for human is systematically translated into low level languages suitable to computer machines. It covers lexical analysis, syntax analysis, semantic analysis, syntax directed translation, intermediate languages, code optimization, and code generation. It will also help to understand design of computer programming language constructs, and its semantics.

Importance of the course

A computer professional write a program in high level programming language, which will be executed on computer hardware. Compiler connects programming language and hardware. Today in high computing era, every single performance percent improvement

matters and save expenditure on hardware. Therefore, understanding of compiler construction process is essential to find optimization scope and write efficient programs. Evolution in computer hardware architecture is continuous and hundreds of programming languages are available. Knowledge of compilation techniques help to design compiler to connect these programming languages and architectures.

Objective of the course:

The objective of this course is to make student understand programming language constructs, and give them hands-on experience with crafting a simple compiler using modern software tools.

Course Pre-requisite:
Data structure, C programming

Course Learning Outcomes:

After successful completion of the course, the students will be able to:
- summarize the functionalities of various phases of compiler
- apply language theory concepts to various phases of compiler design
- identify appropriate optimization technique for compilation process
- develop a miniature compiler using appropriate compiler design tool

Syllabus

Syllabus	Teaching Hours
Unit I **Introduction:** Overview of the Translation Process, Structure of a compiler, Types of compiler and applications, Symbol table	03
Unit II **Lexical Analysis:** The role of a Lexical Analyzer, Input Buffering, Specifications of Tokens, Recognition of tokens, Lexical Analyzer Generator, Finite Automata, Regular Expression to Automata,	06
Unit III **Syntax Analysis:** Context Free Grammar, Top-down Parsing, Bottom-up Parsing, LR Parsers, Error Recovery, Parsing for ambiguous grammars, Parsing Generator Tools	13
Unit IV **Syntax Directed Translation:** Syntax Directed Definition (SDD), Evaluation order of SDD, Syntax Directed Translation Schemes	07
Unit V **Intermediate Code Generation:** Variants of Syntax Trees, Three Address Codes, Type Checking, Control Flow, Back patching	07
Unit VI **Runtime Environment:** Storage Organization, Stack Allocation and Heap Management	02

Unit VII   **07**

Code Generation and Optimization: Issues in code generation, Data Flow and Control Flow, Peephole Optimization, Register Allocation, Machine independent optimization techniques

Self-Study:

Conversion of regular expression to NFA and from NFA to DFA**,**

Stack Allocation and Heap Management

Suggested Readings^:

1. Aho, Lam, Ullman, Sethi, Compilers, Principles, Techniques and Tools, Pearson
2. Keith D Cooper & Linda Torczon, Engineering a Compiler, Elsevier
3. Jean Paul Trembly & Paul G Sorenson, The theory and Practice of Compiler writing, McGraw Hill

^this is not an exhaustive list

Lesson Plan

Lecture No.	Topic	Mapped CLO
	Introduction	
1	Course overview and significance Translator , Compilers, Interpreter	1
2	Analysis-Synthesis Compiler Model	1
3	Single Pass vs Multi-pass Cousins of Compiler Types of compiler and applications Symbol Table	1
	Lexical Analysis	
4	Role of lexical analyzer Token, Lexeme Use of Regular Expression for token Look-ahead operator	1
5	Input Buffering, and Significance Sentinel	1
6	Finite Automata, Optimization of DFA based Pattern Matching Lexical analyzer generator	1,2
7	Regular expression to DFA	1,2
	Syntax Analysis	

8	Role of Parser Introduction to Error recovery Context Free Grammar Comparison of CFG and Regular Expressions	1
9	Context Free Grammar for programming constructs	1,2
10	Introduction to Top Down Parsing LL(1) Parsing	1,2
11	Recursive-Descent Parser	1,2
12	Need of Left Recursion elimination Left Recursion elimination : Direct, Indirect	1
13	Predictive parse table generation LL(1), LL(k) Grammar	1,2
14	Error recovery strategies	1,2

	Error recovery at LL(1) parsers	
15	Bottom-up Parsing Shift Reduce Parsing Shift/Reduce conflict, Reduce/Reduce conflict	1
16	LR Parsing Model	1
17	LR(0) Parse table Generation	1,2
18	SLR Parse table Generation	1,2,3
19	LR(1) Parse table Generation	1,2
20	LALR(1) Parse table Generation Error recovery at LR parsers	1,2
21	Comparison of LR Parsers and LL Parsers Dealing with ambiguous grammar Parsing Generator Tools	1,2
22	Operator Precedence Parsing Error recovery at operator precedence parsing	1,2
	Syntax Directed Translation	
23	Syntax Directed Definition (SDD) Annotated Tree S-attributed and L-attributed SDD	1
24	Type checking SDD	1,2,4
25	Bottom – Up Evaluation of S – Attributed Definitions	1,2
26	Bottom – Up Evaluation of L – Attributed Definitions	1,2
27	Translation scheme	1
28	Top – Down Translation	1,2

29	Recursive evaluators	1,2
	Run-time Environments	
30	Static Memory Allocation Stack Memory Allocation	1
31	Symbol Table Management	1
	Intermediate code generation	
32	Intermediate code representations	1
33	Types of three address statements	1
34	Three address code for declaration, assignment statements, Boolean expressions	1,2
35	Three address code for control(if..else) construct	1,2
36	Three address code for Loop construct	1,2
37	Back-patching	1,2
	Code Generation	
38	Functionalities of Code Generation phase Issues in design of a code generation	1
39	Basic code generation	1,2
40	Basic blocks and flow graph, Control flow	1
41	A simple code generator	1,2
42	Register allocation and assignment	1,3
	Code Optimization	
43	Machine independent optimization techniques	3
44	Local and Global Code optimization	3
45	Peephole Optimization	3

Laboratory details:

Each experiment will be of 10 marks. Evaluation for a 100 marks will be done throughout the semester as part of the Continuous Evaluation scheme. The assessment of Laboratory work is as under:

Total Marks	Continuous Evaluation			Semester End Evaluation	
	No. of experiments	Max. Marks	Weightage	Max Marks	Weightage
100 marks	10	100	75%	25	25%

List of Practical

Sr. No.	Title	Hour(s)	Mapped CLO
1	**To implement lexical analyse to recognize all distinct token classes:** use flex/lex tool to recognize all distinct token classes (Data type, Identifier, constant (Integer, Float, Char, String), Operator (Arithmetic, Relational, Assign, Unary +/-, Increment), Single line/Multi-line comments, Special symbol(;,{}())) . Generate Lexical error reports for invalid lexeme.	02	1,2,4
2	**To implement a Recursive Descent Parser Algorithm for the grammar.**	02	1,2,4
3	Write a program to find first( ), and follow() set for each non-terminal of given grammar.	04	1,4
4	**To Implement Left Recursion derivation removal algorithm** : Eliminate direct and indirect Left recursion from given grammar for LL(1) parser.	04	1,4
5	**To implement a calculator in YACC: Syntax Directed Translation** Extend practical assignment 1 to generate a Symbol Table for identifiers, and label in the code. (Symbol Table columns : Name, Value) Use YACC to Write a Grammar for multiple expression statements, and apply syntax directed translation for calculator.	04	1,2,4
6	**Intermediate Code Generation:** To generate Three Address code for assignment statement	02	1,2,4
7	**To implement grammar rules for control statements, and Loop control**.	04	1,2,4
8	**To implement a Type Checker.:** Extend experiment 5 to assign Data type to each identifier as per declaration statement. Verify Data type as per each programming construct and report appropriate error message	02	1,2,4
9	**To implement Assembly code generator.:** Extend practical 6 to generate an assembly code. (use getReg() algorithm)	02	1,2,4
10	**To implement Code Optimization techniques:** Implement any code optimization technique.	04	1,3,4
	Total	**[30]**	

Teaching Scheme

L	T	P	C
3	0	2	4

Course Assessment Schemes

Assessment scheme	CE				LPW		SEE
Component weightage	0.4				0.2		0.4
	Quiz –I 0.15	Quiz -II 0.15	Sessional Exam 0.4	Term Paper 0.30	Continuous Evaluation 75 marks	Viva Voce 25 marks	100 marks

It is mandatory to clear each component with minimum 40%.

Teaching-learning methodology:

- Lectures: Primarily Chalk and Black board will be used to conduct the course. However, where required, Power Point Presentations (PPTs), Video Lectures, Simulations / Animations etc. will be used to enhance the teaching-learning process.

- Laboratory: Explanation of Experiment to be performed along with co-relation with theory will be given. At the end of each session assessment will be carried out based on parameters like completion of lab work that includes timely submission, satisfying problem specification, performance, analysis, conclusion etc.

- Tutorial: NA

Active learning techniques

- Think-Pair-Share o Think-pair-share (TPS) is a collaborative learning strategy in which students work together to solve a problem or answer a question about an assigned reading. This technique requires students to (1) think individually about a topic or answer to a question; and (2) share ideas with classmates.

- Muddiest Points : It helps assess where students are having difficulties. Faculty asks students at end of session for 5 minutes to quickly identify what they find the "muddiest"—the most confusing or least clear. Collecting students' feedback and responses them in next session/topic.

- Think break : Ask a theoretical question, and then allow 20 seconds for students to think about the problem before the professor go on to explain.

- Online Forum – Students can post their queries as well as answer queries raised by other students.

Course Outcome Attainment:

Following means will be used to assess attainment of course learning outcomes:

- Use of formal evaluation components of continuous evaluation, laboratory work, semester end examination

- Informal feedback during course conduction

Course Material: https://lms.nirmauni.ac.in/course/view.php?id=4849

- Course Policy
- PPTs, Notes, other Material
- Assignments, Tutorials, Lab Manuals
- Previous year Question Papers
- Web-links, Blogs, Video Lectures, Journals
- Advanced Topics

Academic Integrity Statement:

Students are expected to carry out assigned work under Continuous Evaluation (CE) component and LPW component. Copying in any form is not acceptable and will invite strict disciplinary action. Evaluation of corresponding component will be affected proportionately in such cases. Academic integrity is expected from students in all components of course assessment.