NIRMA UNIVERSITY

Institute:	Institute of Technology
Name of Programme:	Master of Computer Application (2-Years Programme)
Course Code:	3MCAD355
Course Title:	System Software
Course Type:	Departmental Elective
Year of Introduction:	2021-22

Credit Scheme

L	T	Practical Component				C
		LPW	PW	W	S	
3	0	2	-	-	-	4

Course Learning Outcomes (CLO):

At the end of the course, students will be able to –

1. relate formal grammar with the concepts of language processing
2. illustrate functionalities of various system software like loader, linker, and device driver
3. analyze working of assembler and macro-preprocessor
4. design a simple compiler using code optimization techniques

Syllabus: **Total Teaching hours: 45**

Unit	Syllabus	Teaching hours
Unit-I	**Language Processor:** Fundamentals of Language Processing and language Specification, Grammar and Types of Grammar, Toy Compiler, Introduction to Data structures: Heap and heap allocation, sorting methods	06
Unit-II	**Scanning and Parsing:** Scanning, Finite Automata: DFA and NFA, Conversion of NFA into DFA, Top Down Parsing, Bottom up Parsing, Introduction to LEX and YACC tools	09
Unit-III	**Assemblers:** Elements of Assembly Language Programming, a Simple Assembly Language Scheme, Pass Structure of Assembler, Design of Two Pass Assembler	06
Unit-IV	**Macro Processors:** Macro Definition and Call, Macro Expansion, Nested Macro Calls, Advanced Macro Pre-processor, Design of Macro Pre-processor: Single Pass Algorithm, Two Pass Algorithm and Macro Calls within Macro Calls	07
Unit-V	**Loader and Linkers:** Relocation And Linking Concept, Design of a Linker, Various schemes of Loader	04
Unit-VI	**Introduction to Compilers:** Aspects of Compilation, Memory Allocation, Compilation of Expression, Compilation of Control Structure, Code Optimization, Interpreters	07
Unit -VII	**Editors and Debuggers:** Various types of Editors and Debuggers and its design	02
Unit-VIII	**Device Drivers:** Introduction to the Device Driver, Requirements of Device Driver, Types of Device Driver	04

Self-Study: The self-study contents will be declared at the commencement of semester. Around 10% of the questions will be asked from self-study contents.

Suggested Readings/ References:

1. Dhamdhere, Introduction to System Software, McGraw Hill.
2. Aho. A.V. Sethi R. and Ulman J.D, Compilers, Principles, Techniques and Tools, Pearson.
3. Srimanta Pal, Systems Programming, Oxford University Press.
4. John Donowan, System Programming, TMH.
5. Leland L. Beck and D. Manjula, System Software-An Introduction to Systems Programming, Pearson.
6. Das, Compiler Design Using Flex and Yacc, PHI.
7. Rajeshkumar Maurya, System Programming, Dreamtech

Suggested List of Experiments:

Sr. No.	Title	Hours
1	To study the structural and implementation process of LEX tool and prepare detail analysis report of all the phases.	02
2	To design a scanner for any programing language like C, C++: The scanner breaks the input (a small program of any language) into individual words or tokens. A token is sequence of characters that can be treated as a unit in the grammar of a programming language.	04
3-4	To generate symbol table in lex for the above code of the practical number 2.	04
5	To design a Parser using any parsing technique: The parser analyzes the phrase structure of the token stream produced by the scanner. The output of the parser is an abstract syntax tree (AST), a data structure that conveys the phrase structure of the source program.	04
6-7	To apply Assembler pass –I algorithm on the source program. The assembly-code generator produces various output tables and Intermediate code. (Source program can be any assembly language program).	04
8	To apply Assembler pass- II on the output produced by the pass-I of Assembler and generate a target program equivalent to the source program.	04
9	To apply Macro processor pass I on the suitable macro code and generate necessary tables.	06
10	To design text Editor in C/C++ programming language. Hint: To implement editor for any language.	02
11*	To implement Operator Precedence Parsing	-
12*	To implement Recursive Descent Parser	-

* optional

Suggested Case List: -NA-