Course Title:	**Languages theory and compilation**
Course Code:	CSE211
Program:	Master Degree In Computer Engineering
Department:	Computer Engineering
Course coordinator:	Dhekra CHERMITI
Institution:	Private Higher School of Engineers of Gafsa (ESIP)

A. Course Identification

1. Credit hours: **3 (2-1-0)**

2. Course type

a. College ☐ Department ■ Others ☐

b. Fundamental ■ Transversal ☐ Optional ☐

3. Level/year at which this course is offered: 1.2/3

4. Pre-requisites for this course : Basic Computer architecture, Programming languages, Formal logic, Data structures and algorithms, Discrete Mathematics

1. Mode of Instruction (mark all that apply)

No	Mode of Instruction	Contact Hours	Self-study	Total workload
1	**Traditional classroom**	…..	33	78
2	**Blended**	45		
3	**E-learning**	.....		
4	**Distance learning**	.....		
5	**Other ()**	.....		

2. Contact Hours (based on academic semester)

No	Activity	Contact Hours
1	**Lecture**	30
2	**Laboratory/Studio**	-
3	**Tutorial**	15
4	**Others** (specify)	-
	Total	45

B. Course Objectives and Learning Outcomes

Course Description

This course introduces formal languages, automata theory, and compiler design. Students will learn about language classification (regular, context-free, and Turing-complete languages), finite automata, grammars, and parsing techniques.
The course covers lexical analysis, syntax analysis, and semantic processing in compiler design, including finite automata, pushdown automata, and Turing machines. It also explores context-free grammars, parsing algorithms (LL, LR), and code-generation techniques used in modern compilers.
By the end of the course, students will be able to analyze formal languages, design parsers, and understand compiler structure, applying these concepts to programming language development.

Course Main Objective

This course aims to:

- ✓ Introduce formal language theory, including regular, context-free, and recursively enumerable languages.
- ✓ Develop an understanding of automata theory, covering finite automata, pushdown automata, and Turing machines.
- ✓ Explain the structure of compilers, focusing on lexical analysis, syntax analysis, and semantic processing.
- ✓ Teach parsing techniques, including top-down (LL) and bottom-up (LR) parsing methods.
- ✓ Provide an overview of code generation and optimization, preparing students for programming language implementation.

1. Course Learning Outcomes

CLOs		Aligned PLOs
	Knowledge and Understanding	
1.1	✓ Understand the different categories of programming languages, including high-level languages, intermediate languages, and low-level languages.	PLOK.1
2.1	✓ Explain the fundamental concepts of formal languages, automata theory, and their classification (regular, context-free, and Turing-complete languages).	
3.1	✓ Describe the key components of compiler architecture, including lexical analysis, syntax analysis, and code generation.	
	Skills	
2.1	✓ Develop structured problem-solving skills by analyzing formal languages and designing parsing techniques.	PLOS.1
2.2	✓ Construct and evaluate formal proofs and derivations in language theory and automata.	
2.5	✓ Implement and optimize parsing techniques, including LL and LR parsing, for compiler design.	PLO.S5

C. Course Content

No	List of Topics	Contact Hours
1	Chapter 1: Introduction to Language Theory 1. Introduction 2. Basic concepts of formal languages 3. Alphabets, words, and languages 4. Language classification (Chomsky hierarchy)	7
2	Chapter 2: Automata Theory 1. Introduction 2. Finite automata (DFA & NFA) 3. Regular expressions and regular languages 4. Pushdown automata and context-free languages 5. Turing machines and decidability	7
3	Chapter 3: Grammars and Parsing 1. Introduction 2. Context-free grammars (CFGs) 3. Syntax trees and derivations 4. LL and LR parsing techniques 5. Recursive descent and shift-reduce parsing	8
4	Chapter 4: Compiler Design and Code Generation 1. Compiler structure and phases 2. Lexical analysis and tokenization 3. Syntax analysis and semantic processing 4. Code generation and optimization	8
Total		30

Tutorials work Content

No	List of Topics	Contact Hours
1	Tutorial 1: Introduction to Formal Languages and Automata • Alphabets, words, and languages • Language classification (Regular, Context-Free, Turing-complete) • Regular expressions and finite automata (DFA, NFA)	3
2	Tutorial 2: Context-Free Grammars and Pushdown Automata • Context-Free Grammars (CFGs) • Parse trees and derivations	3
3	Tutorial 3: Parsing Techniques and Syntax Analysis	3

	• LL(1) and LR(1) parsing • Recursive descent and shift-reduce parsing • Syntax trees and ambiguity in parsing	
4	Tutorial 4: Turing Machines and Computability	3
5	Tutorial 5: Compiler Phases and Code Generation	3
Total		15

D. Teaching and Assessment

1. Alignment of Course Learning Outcomes with Teaching Strategies and Assessment Methods

Code	Course Learning Outcomes	TeachingStrategies	AssessmentMethods
1.0	**Knowledge and Understanding**		
PLOK.1	✓ Understand the different categories of programming languages, including high-level languages, intermediate languages, and low-level languages. ✓ Explain the fundamental concepts of formal languages, automata theory, and their classification (regular, context-free, and Turing-complete languages). ✓ Describe the key components of compiler architecture, including lexical analysis, syntax analysis, and code generation.	- Lecturing, Tutorial	- Assignments, Quizzes , Exams,
2.0	**Skills**		
PLOS.1	✓ Develop structured problem-solving skills by analyzing formal languages and designing parsing techniques.. ✓ Construct and evaluate formal proofs and derivations in language theory and automata.	- Lecturing, Tutorial	- Assignments, Quizzes , Exams,
PLO.S5	✓ Implement and optimize parsing techniques, including LL and LR parsing, for compiler design. compilation phases		

2. Assessment Tasks for Students

#	Assessment task*	Week Due	Percentage of Total Assessment Score
1	Practical Work (written or oral)	Weekly	00%
2	Quizzes, Homework assignments	Random	00%
3	First mid Term	8	35%

#	Assessment task*	Week Due	Percentage of Total Assessment Score
4	Final Exam	16	65%

E. Student Academic Counselling and Support

Arrangements for availability of faculty and teaching staff for individual student consultations and academic advice:
- Office hours - Blackboard interface - Academic advisor - Bibliotic

F. Learning Resources and Facilities

1. Learning Resources

Required Textbooks	1. Grune, Dick, and Ceriel J. H. Jacobs. *Modern Compiler Design.* Springer, 2012. 2. Sudkamp, Thomas A. *Languages and Machines: An Introduction to the Theory of Computer Science.* Addison-Wesley, 2006. 3. Hopcroft, John E., Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Pearson, 2006
Essential References Materials	▪ NA
Electronic Materials	1. MIT OpenCourseWare – Automata, Computability, and Complexity 2. CS50 Harvard – Introduction to Programming Languages and Compilers
Other Learning Materials	NA

1. Facilities Required

Item	Resources
Accommodation	**Classroom board**
Technology Resources	**Data projector**

G. Course Quality Evaluation

Evaluation Areas/Issues	Evaluators	Evaluation Methods
Effectiveness of teaching and assessment.	Students, course coordinator, Alumni, Employers	Direct/Indirect
Extent of achievement of course learning outcomes.	Faculty, Program Leaders, quality department	Direct
Quality of Learning resources	Faculty, Program Leaders,	Direct, Indirect
Teaching and learning quality and effectiveness.	Students, Faculty Program Leaders,	Direct, Indirect

H. Specification Approval Data

Council / Committee	Computer Engineering Council
Date	07/02/2024