



DISCIPLINE ACADEMIC SHEET
ACADEMIC YEAR 2025 – 2026

1. PROGRAMME DATA

1.1. Higher Education Institution	UNIVERSITY OF CRAIOVA
1.2. Faculty	Automation, Computers and Electronics
1.3. Department	Computers and Information Technology
1.4. Field of Study	Computers and Information Technology
1.5. Study level	L(License / Undergraduate)
1.6. Organizational form	IF
1.7. Study Program (name/code) /Qualification	Computers / L2060101010

2. DISCIPLINE DATA

2.1 Discipline Name	Translator Design						
2.2 Course Activities Holder	Tănasie Răzvan						
2.3. Practical Activities Holder	Tănasie Răzvan						
2.4. Study Year	4	2.5. Semester	1	2.6. Evaluation type	E	2.7. Discipline regime	DOP

3. ESTIMATED TOTAL TIME (hours per semester of teaching activities)

ESTIMATED TOTAL TIME (hours per semester of teaching activities)					
3.1. Number of hours per week	4	in which: 3.2 course	2	3.3. sem./lab./pr.	2
3.4. Total hours of curriculum	56	in which: 3.5 course	28	3.6. sem./lab./pr.	28
3.7. Time distribution - hours/week					hours
Study after manual, course support, bibliography and notes					20
Additional documentation in library, on specialized electronic platforms and field					12
Training seminars / labs, homework, portfolios and essays					20
Tutorial					-
Examinations					3
Other activities.....					14
Total hours per individual activities					69
3.8. Total hours per semester					125
3.9. Number of credits					5

4. PRECONDITIONS (where appropriate)

4.1. of curriculum	• Computer Programming and Programming Languages, Algorithms and Data Structures, Formal Languages and Automata
4.2. of competence	• C1

5. CONDITIONS (where appropriate)

5.1. of the course	• The course is taught face to face, with physical presence, using the G Suite for Education platform as support.
--------------------	---

	Course support in electronic format and access to
5.2. of seminar/ laboratory/project	•

6. COURSE OBJECTIVES – expected learning outcomes that are supported by completing and passing the course

Knowledge	The student/graduate: 1. Understands the principles and structure of compilers and translators. 2. Understands the techniques and methods for building the main components of translators
Skills (Abilities)	The student/graduate develops the following practical skills: 1. Developing programming languages or DSLs (domain-specific languages); 2. Creating interpreters, transpilers and code analyzers; 3. Code optimization and machine code generation in embedded systems or JIT compilers; 4. Static and dynamic code analysis (verification and debugging tools); 5. Developing IDEs and intelligent software tools (autocomplete, highlighting, semantic analysis);
Responsibility and autonomy	The student/graduate: 1. The capacity for technical documentation and communication of abstract concepts; 2. A structured approach to software problems; 3. Logical rigor and intellectual discipline;

7. CONTENT

7.1. Course	The method of implementation	Teaching methods	Allocated time (hours)
1. 1. Introduction to compilers and translators	Online sincron (S 1)	The G Suite for Education platform will be used as technical support for transmitting, in electronic format,	2
2. 2. Lexical analysis	Online sincron (S - 2 și 3)		4
3. 3. Syntax analysis (W – 4, 5, 6)	Online sincron	the materials needed for the course. The course is delivered with the help of electronic devices such as the video projector or by videoconference when the situation requires it. The course is structured as follows: - 70% theoretical presentation, based on the course support; - 30% interactive activity, discussions with the students	6

4.	4. Syntax-directed translation	face to face S - 7,8		4
5.	5. Type checking	face to face S 9, 10		4
6.	6. Optimization and intermediate code generation	face to face, S - 11 la 14		8
Bibliography:				
1.	1. Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman - Compilers: Principles, Techniques, and Tools, Addison-Wesley, 2006. 2. Steven Muchnick - Advanced Compiler Design implementation, Morgan Kaufmann Publishers, 1997 3. Marius Brezovan - Limbaje formale si aplicatii in stiinta calculatoarelor, Editura SITECH, Craiova, 2006.			
2.				

7.2. Seminar/laboratory/project	The method of implementation	Teaching methods	Allocated time (hours)
1. 1. The software methods and tools of translator design	Online sincron (S - 1)	The laboratory works are carried out using the G Suite for Education platform. Tutorials, practical demonstrations and code writing are used. The emphasis is placed on understanding how to write lexical and syntax analyzers, interpreting the results and discussions with the students, with feedback being provided for the solved problems.	2
2. 2. Using JFlex to build a lexical analyzer		Synchronous online (W - 2, 3), Face to face (W - 4)	8
3. 3. Using JavaCup to build a syntax analyzer	face to face - S 5 - 8		8
4. 4. Optimization and machine code generation	face to face - S 9 - 14		10
Bibliography:			
1.	1. Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman - Compilers: Principles, Techniques, and Tools, Addison-Wesley, 2006. 2. CUP User's Manual, Scott E. Hudson (https://www.cs.princeton.edu/~appel/modern/java/CUP/manual.html) 3. JFlex User's Manual, 2023, Gerwin Klein, Steve Rowe, and Régis Décamps. (https://jflex.de/manual.pdf)		
2.			

8. Correlating the content of the discipline with the expectations of representatives from the epistemic community, professional associations and representative employers in the field related to the program

The content of the course and of the applications is in line with similar disciplines at universities in the country and abroad. Also, the acquired competences are in accordance with the requirements of representative employers in the field.

9. EVALUATION

9. EVALUATION			
Activity Type	9.1. Evaluation criteria	9.2. Evaluation methods	9.3. Final mark weight
9.4. Course	- Understanding the structure of compilers and translators - Understanding the specific methods and algorithms for lexical analysis, syntax analysis, syntax-directed translation and intermediate code generation - The ability to make connections between the taught notions - The ability to apply the methods specific to translator design for simple programming languages	Multiple-choice exam	60 %
9.5. Seminar/laboratory/project	- The ability to understand and apply specific algorithms and methods for the direct implementation of the modules of simple translators - The ability to understand and apply specific algorithms and methods for implementing the modules of simple translators using code generators	Continuous assessment during the semester	40%
9.6. Minimum standard of performance			

- Maximum 2 absences from the laboratory activities. - Mastery of the basic knowledge regarding the design of lexical analyzers. - Mastery of the basic knowledge regarding the design of syntax analyzers. - Mastery of the basic knowledge regarding syntax-directed translators - Mastery of the basic mechanisms for code optimization

Date of Completion	Signature of the Course Activities Holder,	Signature of the Practical Activities Holder,
23.08.2025	Tănasie Răzvan	Tănasie Răzvan

Date of Approval in the Department	Signature of the Department Director,
25.08.2025	Ș.l.dr.ing. Nicolae ENESCU

Date of Approval in the Faculty	Signature of the Dean,
	Prof.dr.ing. Dorin ȘENDRESCU