SYLLABUS

1. Data about the program of study

1.1 Institution	The Technical University of Cluj-Napoca
1.2 Faculty	Faculty of Automation and Computer Science
1.3 Department	Computer Science
1.4 Field of study	Computer Science and Information Technology
1.5 Cycle of study	Bachelor of Science
1.6 Program of study/Qualification	Computer science/ Engineer
1.7 Form of education	Full time
1.8 Subject code	38.

2. Data about the subject

2.1 Subject name	***Formal Languages and Translators***				
2.2 Course responsible/lecturer	Assoc.prof. dr.eng. Emil Şt. Chifu – emil.chifu@cs.utcluj.ro				
2.3 Teachers in charge of seminars/ laboratory/ project	Assoc.prof. dr.eng. Emil Şt. Chifu – emil.chifu@cs.utcluj.ro				
2.4 Year of study	III	2.5 Semester	2	2.6 Type of assessment (E - exam, C - colloquium, V - verification)	E
2.7 Subject category	*DF – fundamentală, DD – în domeniu, DS – de specialitate, DC – complementară*				DD
	DI – Impusă, DOp – opțională, DFac – facultativă				DI

3. Estimated total time

3.1 Number of hours per week	4	of which:	Course	2	Seminars		Laboratory	2	Project	
3.2 Number of hours per semester	56	of which:	Course	28	Seminars		Laboratory	28	Project	
3.3 Individual study:										
(a) Manual, lecture material and notes, bibliography										7
(b) Supplementary study in the library, online and in the field										5
(c) Preparation for seminars/laboratory works, homework, reports, portfolios, essays										4
(d) Tutoring										
(e) Exams and tests										3
(f) Other activities:										0

3.4 Total hours of individual study (suma (3.3(a)...3.3(f)))	19
3.5 Total hours per semester (3.2+3.4)	75
3.6 Number of credit points	3

4. Pre-requisites (where appropriate)

4.1 Curriculum	Computer Programming, Data Structures and Algorithms
4.2 Competence	Basic knowledge of programming and data structures (preferably in the C language)

5. Requirements (where appropriate)

5.1. For the course	N/A
5.2. For the applications	Computers, specific software

6. Specific competence

6.1 Professional competences	**C1 –** Operating with basic Mathematical, Engineering and Computer Science concepts (2 credits) **C1.1 –** Recognizing and describing concepts that are specific to the fields of calculability, complexity, programming paradigms, and modeling computational and communication systems **C1.2 –** Using specific theories and tools (algorithms, schemes, models, protocols, etc.) for explaining the structure and the functioning of hardware,

	software and communication systems **C1.3 –** Building models for various components of computing systems **C1.4 –** Formal evaluation of the functional and non-functional characteristics of computing systems **C1.5 –** Providing a theoretical background for the characteristics of the designed systems **C3** – Problems solving using specific Computer Science and Computer Engineering tools (2 credits) **C3.1 –** Identifying classes of problems and solving methods that are specific to computing systems **C3.2 –** Using interdisciplinary knowledge, solution patterns and tools, making experiments and interpreting their results **C3.3 –** Applying solution patterns using specific engineering tools and mehods **C3.4 –** Comparatively and experimentaly evaluation of the alternative solutions for performance optimization **C3.5 –** Developing and implementing informatic solutions for concrete problems
6.2 Cross competences	N/A

7. Discipline objective (as results from the *key competences gained*)

7.1 General objective	- To know the phases, components, and algorithms used by typical language translators. - To provide a formal basis for the development of concepts relating to lexical and syntactic processors in translators.
7.2 Specific objectives	- To know the underlying formal models such as finite state automata and push-down automata, and to understand their connection to language definition through regular expressions and grammars. - To understand the relationships between formal descriptions of the automata in the formal language theory and their practical implementations as lexical and syntactic analyzers in translators. - To know the classes of languages for which a deterministic parser can be implemented. - To describe the syntax of languages to be implemented by using grammars and regular expressions. - To design, develop and test a software project, by utilizing specialized software tools (parser generators), in order to arrive at a translator for an artificial language. - To master and control the phenomena of ambiguity and nondeterminism (conflicts) which occur when using parser generators and lexical analyzer generators.

8. Contents

8.1 Lectures	Hours	Teaching methods	Notes
Descriptive tools: strings and rewriting systems, grammars.	2	- The main ideas with multimedia tehniques - Details and examples at the blackboard, in interaction with the students - There are consultation hours - Students are invited to collaborate in research projects	
Descriptive tools: derivations and parse trees.	2		
Regular grammars and finite automata: finite automata.	2		
Regular grammars and finite automata: state diagrams and regular expressions.	2		
Context-free grammars and pushdown automata: pushdown automata.	2		
Top-down analysis and LL(*k*) grammars: LL(*k*) grammars	2		
Top-down analysis and LL(*k*) grammars: the LL(*k*) algorithm	2		
Top-down analysis and LL(*k*) grammars: elimination of left	2		

recursion, left factoring.			
LL parsers: strong LL(*k*) grammars, the LL(1) parsing algorithm.	2		
LL parsers: the LL(1) parsing algorithm, computation of FIRST and FOLLOW sets.	2		
Bottom-up analysis and LR(*k*) grammars: situations and closure of a nonterminal, the LR(*k*) algorithm.	2		
Bottom-up analysis and LR(*k*) grammars:the LR(*k*) algorithm.	2		
LR parsers: the LR(0) parsing algorithm.	2		
LR parsers: LR(0) states.	2		
Bibliography 1. W.M. Waite and G. Goos, Compiler Construction, Springer-Verlag, 1984. 2. I.A. Leţia and E.Şt. Chifu, Limbaje formale şi translatoare, Ed. Casa cărţii de ştiinţă, 1998. 3. A.V. Aho, R. Sethi, and J.D. Ullman, Compilers: Principles, Techniques and Tools, Addison-Wesley, 1986.			
8.2 Applications – Seminars/Laboratory/Project	Hours	Teaching methods	Notes
Lexical analyzer for C.	2	Brief presentation at the blackboard, implementing and testing homeworks on the computer, individual assignment on the computer.	
The generator of lexical analyzers Lex: Lex source, Lex regular expressions, Lex actions, ambiguous rules, Lex source definitions.	2		
Lex generator: left context sensitivity, examples.	2		
The bottom-up parser generator Yacc: basic specifications, Yacc syntax, actions, lexical analysis, how the parser works.	2		
Yacc generator: ambiguity and conflicts, precedence and associativity, error handling, the Yacc environment, hints for preparing specifications.	2		
Yacc generator: support for arbitrary value types, examples (expression evaluator).	2		
Yacc/ Lex applications: interpreter for a language operating on lists.	2		
Yacc/ Lex applications: interpreter for a language operating on binary trees.	2		
Yacc/ Lex applications::interpreter for a language operating on matrices.	2		
Yacc/ Lex applications: code generator for an imperative language.	2		
Yacc/ Lex test	2		
Building recursive-descent (RD) parsers: expression parser.	2		
RD parsers: parser for a language operating on binary trees.	2		
RD parsers: parser for a language operating on lists.	2		
Bibliography 1. The Lex & Yacc Page, http://www.combo.org/lex_yacc_page/ 2. I.A. Leţia, D. Marcu, B. Ungureanu, Procesoare de limbaje. Îndrumător de laborator, Universitatea Tehnică din Cluj-Napoca, 1995.			

* *Se vor preciza, după caz: tematica seminariilor, lucrările de laborator, tematica și etapele proiectului.*

9. Bridging course contents with the expectations of the representatives of the community, professional associations and employers in the field

It is a specialty course in Computer Science, its syllabus being both classical and modern. It teaches the students with the basic principles in the design of interpreters and translators for artificial languages. The syllabus of the course has been discussed with other important universities and companies from Romania, Europe, and USA. This syllabus has been evaluated by Romanian governmental agencies (CNEAA and ARACIS).

10. Evaluation

Activity type	Assessment criteria	Assessment methods	Weight in the final grade
Course	- Problem-solving skills	- Written exam	55%

	- Attendance, Activity		
Seminar			
Laboratory	- Problem-solving skills - Attendance, Activity	- Assessment of the Yacc/ Lex activity and test - Assessment of the RD activity and written exam	30% 15%
Project			
Minimum standard of performance: Modeling a typical engineering problems using the domain specific formal apparatus. Grade calculus: 45% lab + 55% final exam Conditions for participating in the final exam: lab ≥ 5 Conditions for promotion: grade ≥ 5			

Date of filling in:	**Titulari**	**Titlu Prenume NUME**	**Semnătura**
	Course	Assoc. prof. dr. eng. Emil Chifu	
	Applications	Assoc.prof. dr.eng. Emil Şt. Chifu	

Date of approval in the department	Head of department Prof.dr.ing. Rodica Potolea
Date of approval in the Faculty Council	Dean Prof.dr.ing. Liviu Miclea