



COURSE DESCRIPTION

1. Program Information

1.1 University	Alexandru Ioan Cuza
1.2 Faculty	Faculty of Computer Science
1.3 Department	Computer Science
1.4 Study Domain	Computer Science
1.5 Study Cycle	Undergraduate studies
1.6 Study Program / Qualification	Computer Science / Bachelor of Science

2. Course Information

2.1 Course Name	Formal Languages, Automata and Compilers						
2.2 Course Teacher	LECTURER OANA OTILIA CAPTARENCU, PHD LECTURER ALEX MORUZ, PHD						
2.3 Seminary Teacher	LECTURER OANA OTILIA CAPTARENCU, PHD LECTURER ALEX MORUZ, PHD						
2.4 Study Year	2	2.5 Semester	1	2.6 Evaluation	E	2.7 Course Status	OB

* OB – Compulsory / OP – Optional

3. Total estimated hours (hours per semester and didactic activities)

3.1 Hours per week	4	in which: 3.2 course	2	3.3 seminary/laboratory	2
3.4 Hours in curriculum	56	in which: 3.5 course	28	3.6 seminary/laboratory	28
Time Distribution					hours
Manual study, Course support, Bibliography, and others					14
Supplementary Documentation in library, in electronic forums, and on the field					14
Seminaries/laboratories preparation, homeworks, reports, portfolios and essays					28
Tutoring					-
Evaluation					4
Other activities (consultations per student)					-
3.7 Total hours individual study					56
3.8 Total hours per semester					116
3.9 Credits					5

4. Preconditions (if necessary)

4.1 Of Curriculum	
4.2 Of Skills	

5. Conditions (if necessary)

5.1 For Course Operation	
5.2 For Seminary/Laboratory Operation	

**6. Specific Skills Acquired**

Professional Skills	<p>C1. The understanding of fundamental concepts, theories and models used in the design and implementation of the programming languages: formal language, finit automata, regular expression, grammer, pushdown automata, normal forms for grammers, lexical analysis, syntacic analysis (LL Syntactic Analysis, LR Syntactic Analysis), traslation to intermediate code</p> <p>C2. The use of models and mathematical and informatic tools for solving problems in the area of formal languages and compilers</p> <p>C3. The implementation of the informatic components for an application in the area of formal languages and compilers</p>
Transversal Skills	<p>CT1. The application of organized and efficient work rules, of responsible attitudes towards the area of the course, for the creative use of one's potential, respecting the principles and norms of the proffesional ethics.</p>

7. Course Objectives (from the grid of specific skills acquired)

7.1 General Objectives	<ul style="list-style-type: none"> - Teaching fundamental concepts and results on formal languages (especially those of type 2 and 3), grammers,finite automata, regular expressions and pushdown automata - The knowledge of the fundamental algorithms for syntactic analysis and traslation to intermediate code
7.2 Specific Objectives	<ul style="list-style-type: none"> - The ability to use specific mechanisms for generation/recognition/description of formal languages of type 2 and 3 (grammers, automata, regular expressions, pushdown automata) - The ability to apply the fundamental algorithms for lexical and syntax analysis - The ability to use tools such as Lex/Flex and Yacc for generating syntactic analyzors

8. General Description

8.1	Course	Teaching Methods	Observations (hours and bibliographic references)
1.	Formal languages, grammers, Chomsky hierarchy; type 3 languages and grammers	Exposure (video-projector)	References : 3, 4, 5
2.	Properties for the L3 family of languages; Finite automata (deterministic, non-deterministic, with ϵ -transitions), the equivalence of the 3 models	Exposure (video-projector)	References : 3, 4, 5



3.	The connection between type 3 grammars and finite automata; The minimal automata.	Exposure (video-projector)	References : 3, 4, 5
4.	Regular expressions. The automaton equivalent to a regular expression. Expresii regulate; Type 2 languages and grammars.	Exposure (video-projector)	References : 3, 4, 5
5.	Syntactic trees; Chomsky normal form and the CYK algorithm;	Exposure (video-projector)	References : 3, 4, 5
6.	Type 2 languages and pushdown automata.	Exposure (video-projector)	References : 3, 4, 5
7.	Lexical Analysis	Exposure (video-projector)	References : 1,2
8.	Syntactic Analysis – general presentation	Exposure (video-projector), debate, case studies	References: 1,2
9.	SLR Syntactic Analysis	Exposure (video-projector)	References: 1,2
10.	LR1 Syntactic Analysis	Exposure (video-projector)	References: 1,2
11.	LALR1 Syntactic Analysis	Exposure (video-projector)	References: 1,2
12.	LL Syntactic Analysis	Exposure (video-projector), debate	References: 1,2
13.	Traslation to intermediate code	Exposure (video-projector), debate	References: 1,2
14.	Traslation to intermediate code	Exposure (video-projector), debate	References: 1,2
Bibliography			
Main references:			
1. A. V. Aho, M. S. Lam, R. Sethi, J. D. Ullman: <i>Compilers: Principles, Techniques, and Tools</i> . Boston: Addison-Wesley, 2007			
2. Gh. Grigoras. <i>Constructia compilatoarelor - Algoritmi fundamentali</i> , Ed. Universitatii Al. I. "Cuza Iasi", ISBN 973-703-084-2, 274 pg., 2005			
3. Hopcroft, John E.; Motwani, Rajeev; Ullman, Jeffrey D. (2006). <i>Introduction to Automata Theory, Languages, and Computation</i> (3rd ed.). Addison-Wesley			
4. J. Toader - <i>Limbaje formale și automate</i> , Editura Matrix Rom, București, 1999.			
5. J. Toader, Ștefan Andrei – <i>Limbaje formale și teoria automatelor. Teorie și practică</i> , Editura Universității "Al. I. Cuza", Iași, 2002.			
Supplementary references:			
▪ Stoughton Alley, <i>Formal Language Theory</i> , Kansas State University, Draft of Fall 2007			
8.2	Seminary / Laboratory	Teaching methods	Observations (hours and bibliographic references)
1.	The construction of type 2 and 3 grammars for generating given formal languages	Problem solving , debate	References : 1



2.	The construction of type 2 and 3 grammars for generating given formal languages	Problem solving , debate	References : 1
3.	The construction of finite automata for the recognition of given formal languages	Problem solving	References: 1
4.	The algorithm for the transformation of a non-deterministic automaton into the equivalent deterministic automaton	Problem solving , debate, case studies	References : 1
5.	The description of formal languages using regular expressions; The construction of the automaton equivalent to a regular expression	Problem solving , debate, case studies	References : 1
6.	The Chomsky normal form for grammars and the use of CYK algorithm	Problem solving , debate, case studies	References : 1
7.	The construction of pushdown automata for the recognition of type 2 languages	Problem solving , debate,	References: 1
8.	Recapitulation, evaluation	Problem solving	References : 2
9.	The use of the tool Lex/Flex	Problem solving	References : 1,2
10.	The use of the tool Yacc	Problem solving , debate,	References : 3, 4
11.	Project requirements presentation	debate	References : 1, 2,3
12.	Project requirements solving	Problem solving , debate	References : 1, 2,3
13.	Project evaluation – part 1	Problem solving , debate	References : 1, 5
14.	Project evaluation – part 2	Problem solving , debate	References : 1, 5

Bibliography

1. Course bibliography
2. Flex manual: <http://dinosaur.compilertools.net/flex/index.html>
3. Yacc manual: <http://dinosaur.compilertools.net/yacc/index.html>
4. Bison manual: <http://dinosaur.compilertools.net/bison/index.htm>
5. Compiler construction using Flex and Bison: <http://foja.dcs.fmph.uniba.sk/kompilatory/docs/compiler.pdf>

**9. Course content synchronization with the expectations of the community representatives, professional associations and employers from the program domain**

--

10. Evaluation

Activity Type	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 The weight of each evaluation form (%)
10.4 Course	A written test formed of two parts, T1 and T2 The grades for both T1 and T2 should be at least 5.	Written test (examination session)	T1 – 25% T2 – 25%
10.5 Seminary/ Laboratory	The seminar activity (AS) is graded with a grade between 0 and 10. The laboratory activity (AL) is graded with a grade between 0 and 10. The grades for both AS and AL should be at least 3 and AS + AL at least 10.	20% of AS: problem solving 80% of AS: written test AL: the grade for a project	AS – 25% AL - 25%
10.6 Minimal performance standards			
<ul style="list-style-type: none">- The understanding of the notions, theories and models used for the design and implementation of programming languages- The solving of problems with a moderate degree of complexity, using computer science and mathematics knowledge (the ability to use specific mechanisms for generation/recognition/description of formal languages of type 2 and 3, the ability to apply the fundamental algorithms for lexical and syntax analysis and the ability to use tools such as Lex/Flex and Yacc)			

Date

Course Teacher

Seminary/Laboratory Teacher

Department Date of Approval

Director of the Department