NAME OF THE COURSE	**Compilers**					
Code	PMID60	Year of study				
Course teacher	prof.dr. sc. Marko Rosić dr. sc. Tonći Dadić	Credits (ECTS)	5,0			
Associate teachers		Type of instruction (number of hours)	L	S	E	F
			30		30	
Status of the course		Percentage of application of e-learning				

COURSE DESCRIPTION

Course objectives	Provide the main concepts related to the implementation of compilers of programming languages: lexical analysis, syntax analysis, semantic analysis, support the execution of programs and code generation program in the target language.
Course enrolment requirements and entry competences required for the course	Object oriented programming.
Learning outcomes expected at the level of the course (4 to 10 learning outcomes)	Students will be able to: 1. Explain the procedures for analysis and synthesis program 2. Understand lexical, syntactic and semantic properties of a programming language 3. Formally define a simple procedural programming language 4. Select the process of syntax analysis appropriate grammar language 5. Develop a language processor simple procedural programming language 6. Develop a virtual machine defined by programming language.
Course content broken down in detail by weekly class schedule (syllabus)	Week 1: Introduction to the subject. Definition compiler. Components compiler. Automat. Pressure machine. Turing machine. Exercises. Design and implementation of slot machines. Week 2: Regular expressions. Lexical unit. The classification of lexical individuals. The conflict lexical analysis. Creation of the lexical analyzers. Lexical errors and recovery. Exercise: Regular expressions. Using regex class. Week3: The definition of grammar. The formal presentation of grammar in BNF notation. Classification of languages by Chomsky. Practices: Implementation of lexical analyzers based on class RegEx. Week4: LL (1) and LR (1) grammar. Left and right generative syntax tree. The abstract syntax tree. Exercises: Design and Implementation: object models grammar and abstract syntax tree. Week 5: The introduction of a simple programming language: input, output and assign variables values algebraic-logical expressions with parentheses. LL (1) grammar of the language. Syntax analysis, recursive descent. Exercises: Design and implementation of a recursive descent parser. Week 6: Construction of syntax analyzers from top to bottom with the help of the pusher machine. Conferences begins, followed by the application. Table syntax analysis.

	Practices: Implementation process of building the table syntax analysis. Week 7: Parse the program from top to bottom using the pushdown automaton. Syntax error and recovery. Exercise: Preparing for the first midterm. Week 8: LR (0) syntax analyzer. Building tables GO TO / ACTION. LR syntax analyzer. Exercise: The first colloquium Week 9: Disadvantages of LR (0) grammar. LR (1) syntax analyzer. Building tables GO TO / ACTION LR (1) syntax analyzer. Practices: Implementation of syntax analyzers from top to bottom based on the discharge slot. Week 10: Expanding grammar simple language instruction decisions and repetition. Table identifiers. Semantic analysis program. Exercises: Design and implementation of LR syntax analyzer. Week 11: Virtual stogovno oriented machine. The introduction of intermediate instruction. Exercises: Design and implementation of LR syntax analyzer (continued). Week 12: Support the execution of the program. Calling procedures and functions. Support recursion. Exercises: Design and implementation of a table identifier and semantic analysis program. Week 13: Generating intermediate code as a linear program of the virtual machine. Exercises: Design and implementation stogovno oriented virtual machine. Week 14: Basic features of translating object-oriented programming languages. Exercise: Preparing for the second midterm. Week 15: Study examples of the virtual machine: Microsoft IL language. Exercise: The second colloquium.
Format of instruction	☒ lectures ☐ seminars and workshops ☒ exercises ☐ on line in entirety ☐ partial e-learning ☐ field work ☐ independent assignments ☐ multimedia ☐ laboratory ☐ work with mentor ☒ homework assignments
Student responsibilities	Attendance, active participation in the learning process, homework, a colloquium or practical / written examination, oral examination.

Screening student work *(name the proportion of ECTS credits for each activity so that the total number of ECTS credits is equal to the ECTS value of the course)*

Name	Ects	Name	Ects	Name	Ects
Class attendance	0.5	Research		Experimental work	
Oral exam	2	Report		Homework assignments	0.5
Seminar essay		Essay			

	Tests		Practical training			
	Written exam	2	Project			

Grading and evaluating student work in class and at the final exam	Class attendance (10%), two homework assignments (10%), practical / written exam (40%)

Required literature (available in the library and via other media)	**Title**	**Number of copies in the library**	**Availability via other media**
	Srbljić, S: Programming Language Translation, Element, Zagreb, 2007.	10	9

Optional literature (at the time of submission of study programme proposal)	Grune, D., Bal, H., E., Jacobs, C., J., H., Langendoen, K., G.: Modern Compiler Design, Wiley, 2000.
Quality assurance methods that ensure the acquisition of exit competences	Talk with students, student evaluation using the anonymous survey, the success of students in the exam, self-assessment.
Other (as the proposer wishes to add)	