

Study Programme: Computer Science – Master			
Course Unit Title: Compiler Construction			
Course Unit Code: CS705			
Name of Lecturer(s): Mirjana Ivanović			
Type and Level of Studies: Master Academic Degree			
Course Status (compulsory/elective): Elective			
Semester (winter/summer): Winter			
Language of instruction: Serbian (primary), English (secondary)			
Mode of course unit delivery (face-to-face/distance learning): Face-to-face			
Number of ECTS Allocated: 8			
Prerequisites: Object-Oriented Programming 1, Data Structures and Algorithms 1			
Course Aims: The main objective of the course is to learn students about essential work of different phases of compilers and make them skilled to participate at bigger project and implement compilers for simple procedural and object-oriented languages.			
Learning Outcomes: <i>Minimum:</i> Successful students should be able to implement compiler for subset of procedural programming language based on given grammar for language specification. <i>Desirable:</i> At the end of the course it is expected that successful students are able to develop adequate software for translation from input text to output text/form based on given specification (grammar rules).			
Syllabus: <i>Theory</i> Techniques for programming languages specifications. Syntax diagrams, Backus and Extended Backus normal forms for programming language grammar specification. Context-free grammars, LL, LR and attributed grammars. Essential principles, tasks and phases of compilers: lexical analysis, syntax analysis using recursive descent technique, semantic analysis (type checking) and symbol table maintenance, code generation (using Virtual machine). Description of complete implementation of compiler for simple procedural (including some basic object-oriented concepts) programming language. Compiler generators. <i>Practice</i> Practical part is oriented to the gradual development and extension of existing parts of code for a real compiler. During laboratory classes students have task to fully implement parts of compiler adding their own code. They gradually develop compiler following theoretical classes.			
Required Reading: 1. Hanspeter Mössenböck, Compiler Construction Slides, Institut für Systemsoftware, Johannes Kepler Universität Linz, Austria 2. Aho V., Ullman J. D.: "Principles of Compiler Design", Addison-Wesley, 1977. 3. Aho V., Sethi R., Ullman J. D. "Compilers, Principles, Techniques and Tools, Addison-Wesley, 1985.			
Weekly Contact Hours: 5		Lectures: 2	Practical work: 3
Teaching Methods: Theoretical classes are based on the classical teaching model involving a projector and .ppt presentations. Basic principles and compiler functions are presented by illustrative examples. Students are expected to pass 3 theoretical tests. At theoretical exercises, existing parts of code for compiler implementation will be explained in more details. During laboratory classes students will work on implementation of particular parts of their own versions of the compiler. All together 5 tasks-seminars are foreseen for complete implementation of the compiler and grades depend on number of completed tasks. At the oral exam students demonstrate understanding of principles of compilers.			
Knowledge Assessment (maximum of 100 points):			
Pre-exam obligations	points	Final exam	points
Active class participation		written exam	
Tests and practical tasks	60	oral exam	40
Preliminary exam(s)		
Seminar(s)			
The methods of knowledge assessment may differ; the table presents only some of the options: written exam, oral exam, project presentation, seminars, etc.			