Study programme(s): Teaching Informatics (IC), Informatics (IM)

Level: master

Course title: Compiler construction 1 (code IA111)

Lecturers: Mirjana K. Ivanović, Danijela N. Boberić-Krstićev

Status: obligatory for the *Computer Science* module; elective for other modules and study programme *"Teaching Informatics"*.

ECTS: 7

Requirements: none

Learning objectives

Training students to design and create a compiler for a procedural or object-oriented programming languages.

Learning outcomes

Minimum: Students should be able to write a compiler for a subset of a procedural programming language based on the specifications provided.

Optimal: Students should be able to develop software for the transformation of general input text to an output text, according to the specifications provided.

Syllabus

Theoretical instruction

Description of programming languages. Syntax diagrams. Backus normal form. Context-free grammars. LL, LR and related grammars. Compiler generators. The working principle of compilers. Attribute grammars. An example of a compiler generator.

Practical instruction

Examples of compilers for a subset of a procedural or object-oriented programming language. The main parts of a compiler. Management of the symbol table. The basic elements of lexical analysis. Syntax analysis – method of recursive descent. Semantic analysis of compliance types. Abstract machine. Code generation. Optimization of code.

Literature

Suggested:.

1. Hanspeter Mössenböck, Compiler Construction Slides, Institut für Systemsoftware, Johannes Kepler Universität
Linz, Austria
2. V. Aho, J. D. Ullman: "Principles of Compiler Design", Addison-Wesley, 1977.
3. V. Aho, R. Sethi, J. D. Ullman "Compilers, Principles, Techniques and Tools, Addison-Wesley, 1985.

Alternative:

Mirjana Ivanovic, Compilers and interpreters, script, University of Novi Sad,
Faculty of Sciences, Department of Mathematics and Informatics, Novi Sad, 2002.

Weekly teaching load				Other:
Lectures: 2	Exercises: 3	Other forms of teaching:	Student research:	

Teaching methodology

Classical methodology is applied on lectures with usage of a video-beam and slides projector. The principles and functionalities of a compiler are explained and illustrated by examples. On lectures, students' knowledge is checked through three tests. On exercises, the Java programming language is used for compiler implementation. The software package "Svetovid", developed at the Chair of Computer Science, is used to prevent cheating. On the exercises, students’

knowledge is verified through 5 assignments/seminars covering the appropriate stages of compiler development. At the oral exam, the students should demonstrate understanding of the principles and operations performed by the compiler.			
Grading (maximum number of points 100)			
Pre-exam obligations	**points**	**Final exam**	**points**
Active participation in lectures	**6**		
Practical instruction	**6**	Oral exam	**40**
Colloquia	**8, 8, 8**		
Seminar(s)	**5,5,5,4,5**		