AGH UNIVERSITY OF SCIENCE AND TECHNOLOGY

Code:	UBPJO-122	Module name:	Compilation theory		
Academic year:	2013/2014	Semester:	Fall	ECTS credits:	4
Programme:	University Base of Courses in English				
Course homepage:	http://www.math.us.edu.pl/~pgladki/teaching/index.html	Lecture language:	English		
Responsible teacher:	dr Gładki Paweł (pawel.gladki@us.edu.pl)				
Academic teachers:	dr Gładki Paweł (pawel.gladki@us.edu.pl)				

Description of learning outcomes for module

MLO code	Student after module completion has the knowledge/ knows how to/is able to	Method of learning outcomes verification (form of completion)
Skills		
M_U001	The student knows how to implement an interpreter	Project
M_U002	The student knows how to program an abstract syntax and lexical analyzer.	Project
M_U003	The student knows how to design a parser.	Project
M_U004	The student knows how to devise a type checker.	Project
M_U005	The student knows how to proceed with code generation	Project
M_U006	The student knows how to perform liveness analysis	Project
Knowledge		
M_W001	The student is familiar with regular expressions, finite automata, Lex and ML-Lex	Test
M_W002	The student understands basic notions of context-free grammars, parsing and Yacc	Test
M_W003	The student has a working knowledge of symbol tables and type-checking	Test
M_W004	The student posseses a deep understanding of stack frames, intermediate trees, expressions to trees, declarations to trees, canonical trees, instruction selection, assemby code, liveness, register allocation and linking.	Test

M_W005	The student is able to comprehend garbage collection, object-oriented languages, higher-order functions and closures	Test
M_W006	The student knows about parser generation, just-in-time compilation, program analysis, optimization	Test

FLO matrix in relation to forms of classes

MLO code	Student after module completion has the knowledge/ knows how to/is able to	Form of classes										
		Lectures	Auditorium classes	Laboratory classes	Project classes	Conversation seminar	Seminar classes	Practical classes	Others	Fieldwork classes	Workshops	E-learning
Skills												
M_U001	The student knows how to implement an interpreter	-	-	+	-	-	-	-	-	-	-	-
M_U002	The student knows how to program an abstract syntax and lexical analyzer.	-	-	+	-	-	-	-	-	-	-	-
M_U003	The student knows how to design a parser.	-	-	+	-	-	-	-	-	-	-	-
M_U004	The student knows how to devise a type checker.	-	-	+	-	-	-	-	-	-	-	-
M_U005	The student knows how to proceed with code generation	-	-	+	-	-	-	-	-	-	-	-
M_U006	The student knows how to perform liveness analysis	-	-	+	-	-	-	-	-	-	-	-
Knowledge												
M_W001	The student is familiar with regular expressions, finite automata, Lex and ML-Lex	+	-	-	-	-	-	-	-	-	-	-
M_W002	The student understands basic notions of context-free grammars, parsing and Yacc	+	-	-	-	-	-	-	-	-	-	-
M_W003	The student has a working knowledge of symbol tables and type-checking	+	-	-	-	-	-	-	-	-	-	-
M_W004	The student posseses a deep understanding of stack frames, intermediate trees, expressions to trees, declarations to trees, canonical trees, instruction selection, assemby code, liveness, register allocation and linking.	+	-	-	-	-	-	-	-	-	-	-

M_W005	The student is able to comprehend garbage collection, object-oriented languages, higher-order functions and closures	+	-	-	-	-	-	-	-	-	-	-
M_W006	The student knows about parser generation, just-in-time compilation, program analysis, optimization	+	-	-	-	-	-	-	-	-	-	-

Module content

Lectures

Lectures on compilation theory
(1) Regular expressions, finite automata, Lex and ML-Lex.
(2) Context-free grammars, parsing and Yacc.
(3) Symbol tables, type-checking.
(4) Stack frames, intermediate trees, expressions to trees, declarations to trees, canonical trees.
(5) Instruction selection, assemby code, liveness, register allocation, linking.
(6) Garbage collection, object-oriented languages, higher-order functions, closured.
(7) Parser generation, just-in-time compilation, program analysis, optimization.

Laboratory classes

Labs in compilation theory
(1) Programming assignment 1: interpreter.
(2) Programming assignment 2: abstract syntax and lexical analyzer.
(3) Programming assignment 3: parser.
(4) Programming assignment 4: type checker.
(5) Programming assignment 5: code generation.
(6) Programming assignment 6: liveness analysis.

Method of calculating the final grade
6 programming assignment worth 60% total.
2 mid-term exams worth 40% total.

Prerequisites and additional requirements
Prerequisites and additional requirements not specified

Recommended literature and teaching resources
(1) Andrew W. Appel, Modern Compiler Implementation in ML. Cambridge University Press, 1998.
(2) Jeffrey D. Ullman, Elements of ML Programming, second edition, Prentice Hall.
(3) Robert Harper, Programming in Standard ML: http://www.cs.cmu.edu/%7Erwh/smlbook/book.pdf

Scientific publications of module course instructors related to the topic of the module
Additional scientific publications not specified

Additional information

None

Student workload (ECTS credits balance)

Student activity form	Student workload
Participation in lectures	14 h
Realization of independently performed tasks	17 h
Participation in laboratory classes	14 h
Completion of a project	15 h
Preparation for classes	15 h
Contact hours	25 h
Summary student workload	100 h
Module ECTS credits	4 ECTS