(faculty stamp)

COURSE DESCRIPTION

1. Course title: INTRODUCTION TO COMPILERS	**2. Course code:** ItC
3. Validity of course description: 2017/2018	
4. Level of studies: 2nd cycle of higher education	
5. Mode of studies: intramural studies	
6. Field of study: MACROFACULTY	
7. Profile of studies: general academic	
8. Programme:	
9. Semester: II	
10. Faculty teaching the course: Faculty of Automatic Control, Electronics and Computer Science	
11. Course instructor: dr inż. Przemysław Szmal	
12. Course classification: specialization courses	
13. Course status: compulsory	
14. Language of instruction: English	
15. Pre-requisite qualifications: ability to program in C language	

16. Course objectives:

The goal of the course is to present selected problems connected to programming language description and compiler construction. The student masters algorithms and methods for lexical and syntactic analysis, as well as ways of extending them for translation purposes. Topics suitable for construction of simple translators met in programmer's practice – command interpreters, macro-generators, linkers and so on – are covered.

17. Description of learning outcomes:

Nr	Learning outcomes description	Method of assessment	Teaching methods	Learning outcomes reference code
1	Knowledge in the scope of specifying programming languages	Test	Lecture, laboratory classes	K2A_W05
2	Knowledge in the scope of compiler operation and construction	Test	Lecture, laboratory classes	K2A_W05
3	Knowledge of the lexical layer of programming languages	Test	Lecture, laboratory classes	K2A_W05
4	Ability to perform syntactic analysis of texts	Test, laboratory task	Lecture, laboratory classes	K2A_U06, K2A_U14
5	Ability to use lex and yacc analysers	Test, laboratory task	Laboratory classes	K2A_U06, K2A_U14

18. Teaching modes and hours

Lecture 30 h **/ BA/MA Seminar** – **/ Class** – **/ Project** – **/ Laboratory** 30 h

19. Syllabus description:

Lectures:

Essence of programming language machine translation: generating equivalent programs expressed in another language. Lexical, syntactic and semantic layers of programs. Connections with the construction of the language virtual machine. Characteristic stages in translating programs to target form: compilation and its phases, consolidation. Translation schema variants.
Language description methods and using them in the text analysis stage. Formal grammars, Chomsky's classification.
Lexical layer of programming languages – regular grammars. (Stack-less) nondeterministic, deterministic finite-state automata.
The syntactic layer – context-free grammars. Grammar transformations: left recursion elimination, (left) factorization, disambiguation.
Top-down syntax analysis: deterministic analysers based on the recursive descent principle, non-recursive predictive analysis; LL-grammars. Construction of parse-driving table.
Bottom-up syntax analysis. Simple- and operator-precedence grammars – analysis algorithm, construction of parse-driving table. Evaluating precedence functions. LR-grammars. Analysis algorithm, construction of simple (SLR) canonical (LR), look-ahead (LALR) LR-analysers.
Semantic analysis according to the principles of syntax directed translation.
Selected information on intermediate code, output code, and run-time organization.

Laboratory:
The aim of the laboratory classes is to get students practically acquainted with techniques of building text processors, whose action can be described by means of regular expressions and a subclass of context-free grammars. At the beginning, the students recognize *lex* and *yacc* – how to build simple and nuanced lex and yacc specifications and how to use and debug resulting lexers and parsers. Then the students are assigned individual projects involving parsing mechanisms; the projects should be completed at home and presented at the end of the semester.

20. Examination: no

21. Primary sources:

A.V.Aho, M.S. Lam, R. Sethi, J.D. Ullman: "Compilers. Principles, techniques, and tools. Second edition". Addison-Wesley, Reading, MA, 2006 (also available for free-download in a .pdf form at http://www.pdf-search-engine.com/aho-sethi-ullman-pdf.html)

22. Secondary sources:

Ch. Donnelly, R. Stallman: Bison. The YACC-compatible Parser Generator.
http://www.gnu.org/software/flex/manual ,
http://www.gnu.org/software/bison/manual/
T. Niemann, "A Compact Guide to Lex & Yacc", http://www.epaperpress.com/lexandyacc/
http://dinosaur.compilertools.net/

23. Total workload required to achieve learning outcomes

Lp.	Teaching mode :	Contact hours / Student workload hours
1	Lecture	30 / 15
2	Classes	- / -
3	Laboratory	30 / 45
4	Project	- / -
5	BA/MA Seminar	- / -
6	Other	- / -
	Total number of hours	60 / 60

24. Total hours: 120

25. Number of ECTS credits: 4

26. Number of ECTS credits allocated for contact hours: 2

27. Number of ECTS credits allocated for in-practice hours (laboratory classes, projects): 1

26. Comments: –

Approved:

…….....
(date, Instructor's signature)

…….....
(date, the Director of the Faculty Unit signature)