Universitatea Națională de Știință și Tehnologie Politehnica București
Facultatea de Electronică, Telecomunicații și
Tehnologia Informației

COURSE DESCRIPTION

1. Program identification information

1.1 Higher education institution	**National University of Science and Technology Politehnica Bucharest**
1.2 Faculty	**Electronics, Telecommunications and Information Technology**
1.3 Department	**Electronic Devices, Circuits and Architectures**
1.4 Domain of studies	Electronic Engineering, Telecommunications and Information Technology
1.5 Cycle of studies	Masters
1.6 Programme of studies	Advanced Computing in Embedded Systems

2. Date despre disciplină

2.1 Course name (ro) (en)				Compilatoare Compilers			
2.2 Course Lecturer				Conf. Dr. Radu Hobincu			
2.3 Instructor for practical activities				Conf. Dr. Radu Hobincu			
2.4 Year of studies	2	2.5 Semester	I	2.6. Evaluation type	E	2.7 Course regime	Ob
2.8 Course type		DA	2.9 Course code	UPB.04.M3.O.26-23		2.10 Tipul de notare	Nota

3. Total estimated time (hours per semester for academic activities)

3.1 Number of hours per week	4	Out of which: 3.2 course	2.00	3.3 seminary/laboratory	2
3.4 Total hours in the curricula	56.00	Out of which: 3.5 course	28	3.6 seminary/laboratory	28
Distribution of time:					hours
Study according to the manual, course support, bibliography and hand notes Supplemental documentation (library, electronic access resources, in the field, etc) Preparation for practical activities, homework, essays, portfolios, etc.					63
Tutoring					0
Examinations					6
Other activities (if any):					0

3.7 Total hours of individual study	69.00
3.8 Total hours per semester	125
3.9 Number of ECTS credit points	5

4. Prerequisites (if applicable) (where applicable)

4.1 Curriculum	• Computer programming • Data Structures and Algorithms • Architecture of Microprocessors
4.2 Results of learning	Average to good knowledge of at least one programming language between C, C++ and Java, preferably C.

5. Necessary conditions for the optimal development of teaching activities (where applicable)

5.1 Course	Projector, workstation/laptop with Windows operating system (at least 8.1) with at least 4 GB RAM and processor in the performance range, newer than 2014, to be able to run a virtual machine.
5.2 Seminary/ Laboratory/Project	Projector, workstations with Linux operating systems (newer than 2014), with at least 8 GB of RAM and processor in the performance range, newer than 2014, one server in the middle performance class, to coordinate the infrastructure.

6. General objective *(Reffering to the teachers' intentions for students and to what the students will be thought during the course. It offers an idea on the position of course in the scientific domain, as well as the role it has for the study programme. The course topics, the justification of including the course in the currcula of the study programme, etc. will be described in a general manner)*

Acquiring knowledge specific to the development and implementation of a compiler.

7. Competences *(Proven capacity to use knowledge, aptitudes and personal, social and/or methodological abilities in work or study situations and for personal and proffesional growth. They refflect the empolyers requirements.)*

Specific Competences	The student will understand how a compiler works and be able to design and implement a compiler for a new or existing processor architecture.
Transversal (General) Competences	The student will gain knowledge about optimizing software applications by exploiting the functionalities that a compiler provides.

8. Learning outcomes *(Synthetic descriptions for what a student will be capable of doing or showing at the completion of a course. The learning outcomes reflect the student's acomplishments and to a lesser extent the teachers' intentions. The learning outcomes inform the students of what is expected from them with respect to performance and to obtain the desired grades and ECTS points. They are defined in concise terms, using verbs similar to the examples below and indicate what will be required for evaluation. The learning outcomes will be formulated so that the correlation with the competences defined in section 7 is highlighted.)*

Knowledge	*The result of knowledge aquisition through learning. The knowledge represents the totality of facts, priciples, theories and practices for a given work or study field. They can be theoretical and/or factual.* • Knowing the structure of a compiler • Knowledge of the implementation of each component in a compiler architecture • Knowledge of optimization techniques that a compiler routinely uses • Gaining knowledge about using a compiler effectively

Skills	*The capacity to apply the knowledge and use the know-how for completing tasks and solving problems. The skills are described as being cognitive (requiring the use of logical, intuitive and creative thinking) or practical (implying manual dexterity and the use of methods, materials, tools and intrumentation).* The master's student will be able to write a backend for a new processor for one of the compilers used in practice (GCC or LLVM).
Responsability and autonomy	*The student's capacity to autonomously and responsably apply their knowledge and skills.* -

9. Teaching techniques *(Student centric techniques will be considered. The means for students to participate in defining their own study path, the identification of eventual fallbacks and the remedial measures that will be adopted in those cases will be described.)*

The course will be supported by Powerpoint presentations on which to discuss and students to collect notes.

Students will have a lab sheet and the codes needed for the applications as support. Each student will work individually at a workstation. At the beginning of the laboratory, its objective will be explained as well as the relationship with the course material.

10. Contents

COURSE		
Chapter	**Content**	**No. hours**
1	Microprocessor architectures and instruction sets	2
2	The elements of a programming language, Taxonomies and Paradigms	2
3	Generalities about compilers and an example of the architecture of a compiler	2
4	Lexical and Syntactic Analysis	2
5	Semantic Analysis	2
6	Machine code generation	2
7	Data flow analysis	2
8	Techniques for optimizing the generated code	2
9	Techniques for optimizing the source code of applications 1	2
10	Techniques for optimizing the source code of applications 2	2
11	Description of Just-in-time compilers	2
12	Research directions in the field of design and implementation of a compiler	4
13	Evaluare finală	2
	Total:	28

Bibliography:

1. Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, “Compilers – Principles, Techniques and Tools”, Ullman Publisher, ISBN 0321486811
2. Steven Muchnick, “Advanced Compiler Design and Implementation”, Morgan Kaufman Publishers, 1997, ISBN 0321486811
3. Randy Allen, Ken Kennedy, “Optimizing Compilers for Modern Architectures”, Morgan Kaufman Publishers, 2001, ISBN 1-55860-286-0
4. John L. Hennessy, David A. Patterson, “Computer Architecture: A Quantitative Approach”, Morgan Kaufmann; 4 edition (September 27, 2006), ISBN: 0123704901

LABORATORY

Crt. no.	Content	No. hours
1	Analysis of the use of a processor's resources and the assembly code	2
2	Experiments to analyze the performance of a programming language and the default compiler	2
3	Data structures and algorithms used in the implementation of compilers	2
4	Implementation of lexical and syntactic analysis	2
5	Implementation of semantic analysis	2
6	Generating machine code for a simple processor	2
7	Implementation of the data flow analysis step	2
8	Implementation of techniques to optimize the generated code	2
9	Experiment with a program's build options	2
10	Examples of source code optimization techniques 1	2
11	Examples of source code optimization techniques 2	2
12	Analysis of just-in-time compilers	2
13	Recapitulation and Evaluation	4
	Total:	28

Bibliography:

1. Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, “Compilers – Principles, Techniques and Tools”, Ullman Publisher, ISBN 0321486811
2. John L. Hennessy, David A. Patterson, “Computer Architecture: A Quantitative Approach”, Morgan Kaufmann; 4 edition (September 27, 2006), ISBN: 0123704901
3. Documentație Intel vTune Amplifier (https://software.intel.com/en-us/intel-vtune-amplifier-xe-support/documentation)

11. Evaluation

Activity type	11.1 Evaluation criteria	11.2 Evaluation methods	11.3 Percentage of final grade

11.4 Course	Good understanding of the theoretical notions presented in the course will be pursued.	Written exam	35%
11.5 Seminary/laboratory/project	A good understanding of the application of the knowledge presented in the course and during the labs will be sought.	Project, colloquium, evaluations along the way	65%
11.6 Passing conditions			
The student will have to obtain a minimum of 50% of the total score (course and laboratory), at the end of the evaluation. There are no conditions for obtaining a minimum of 50% of the score for each activity (course or laboratory).			

12. Corroborate the content of the course with the expectations of representatives of employers and representative professional associations in the field of the program, as well as with the current state of knowledge in the scientific field approached and practices in higher education institutions in the European Higher Education Area (EHEA)

The course covers a field very important to both the scientific and industrial communities: the generation of efficient machine code, using a compiler, from high-level description. The task of compilers at this moment is becoming more and more difficult as hybrid, parallel and reconfigurable systems become more often present in electronic devices. The course thus allows students to understand and use modern computing systems.

Date	Course lecturer	Instructor(s) for practical activities
11.10.2024	Conf. Dr. Radu Hobincu	Conf. Dr. Radu Hobincu

Date of department approval	Head of department
31.10.2024	Prof. Dr. Claudius DAN

Date of approval in the Faculty Council	Dean
01.11.2024	Prof. Dr. Mihnea Udrea