COURSE DESCRIPTION

Computer Systems Architecture

Academic year 2026-2027

1. Programme-related data

1.1. Higher Education Institution	Babeş-Bolyai University
1.2. Faculty	Faculty of Mathematics and Computer Science
1.3. Department	Department of Computer Science
1.4. Field	Mathematics and Computer Science
1.5. Level of study	Bachelor
1.6. Degree programme / Qualification	Mathematics and Computer Science (in English)
1.7. Form of education	Full time

2. Course-related data

2.1. Course title	**Analysis and synthesis of circuits**			Course code	**MLE5004**
2.2. Course coordinator	Prof. dr. ing. Octavian Creţ				
2.3. Seminar coordinator	Prof. dr. ing. Octavian Creţ				
2.4. Year of study	2	2.5. Semester	3	2.6. Type of assessment	Exam
2.7. Course status	Compulsory			2.8. Course type	Specialisation subject

3. Total estimated time (hours per semester of teaching activities)

3.1. Number of hours per week	4	of which: 3.2. course	2	3.3. seminar/ laboratory/ ~~project~~	1 lab 1 sem
3.4. Total of hours in the curriculum	56	of which: 3.5. course	28	3.6. seminar/ laboratory/ ~~project~~	28
Time allocation for individual study (IS) and self-taught activities (ST)					**hours**
Learning from textbooks, course materials, bibliography, and notes (IS)					21
Additional research in the library, on subject-specific electronic platforms, and on-site					20
Preparing seminars/ laboratories/ projects, assignments, reports, portfolios, and essays					21
Tutoring (professional guidance)					4
Examinations					3
Other activities					
3.7. Total hours of individual study (IS) and self-taught activities (ST)				69	
3.8. Total hours per semester				125	
3.9. Number of credits				5	

4. Prerequisites (where applicable)

4.1. curriculum-related	
4.2 skills-related	

5. Specific conditions (where applicable)

5.1. course-related	Class room with projector
5.2. seminar/laboratory-related	Laboratory with computers

6.1. Competencies resulting from the completion of the degree programme ***(as referred to in the curriculum)***

Professional competencies	
Competency code	**Competency**

PC3	Development and analysis of algorithms for solving problems
PC6	Analysis, testing and using of software system
Transversal competencies	
Competency code	**Competency**
TC1	Application of organized and efficient work rules, a responsible attitude towards the didactic-scientific field, to bring creative value to own potential respecting professional ethics principles.
TC2	Ability to adopt and integrate in different environments from education, research and economy

6.2. Learning outcomes relevant to the degree programme (as referred to in the curriculum)

Learning outcomes targeted by the subject		
Competency code	**Knowledge and comprehension**	**Specific academic skills**
PC1	The graduate knows and understands the basic concepts, theories and methods of Mathematics and Computer Science and is able to use them appropriately in professional communication.	• Note-Taking: Summarizing and recording key information from lectures or texts. • Critical Thinking: Analyzing, evaluating, and synthesizing information rather than just • Reading Comprehension: Active reading, scanning, skimming, and understanding complex texts.
PC2	The graduate is able to design / implement hardware, software and communications components using design methods, languages, algorithms, data structures, protocols and technologies, and evaluate their functional and non-functional characteristics based on metrics.	• Time Management: Prioritizing tasks, meeting deadlines, and organization. • Academic Writing: Constructing clear, structured, and evidence-based arguments. • Study Skills: Revision techniques, test-taking strategies, and memory aids.
PC3	The graduate performs the testing and qualitative evaluation of the functional and nonfunctional characteristics of the information systems, based on specific criteria.	• Problem-Solving: Applying logical reasoning to solve academic problems.
PC5	The graduate is able to use electronic tools to characterize and evaluate the performance of electronic circuits.	• Digital Literacy: Using databases, software, and online tools effectively for research. • Communication & Presentation: Public speaking, presenting arguments, and group work. • Collaboration: Working effectively in teams and providing constructive critique.

7. Subject-specific learning outcomes

Knowledge and comprehension
1. Understanding computer architectural models, processor operation, and the use of information representation systems in computer systems
2. Mastering computer architectural models, processor operation, and the use of information representation systems in computers.
3. Introduction to assembly language programming, which ensures an understanding of the architecture and operation of a microprocessor.
4. Understanding the impact of 80x86 processor architecture on the operating system.
5. Knowledge of methods for assessing the performance of computer systems
6. Knowledge of architectural methods for improving the performance of computer systems
Specific academic skills
1. The graduate is able to design and optimize a computer system at architectural level
2. The graduate is able to test and evaluate the functional and non-functional characteristics of computer systems based on specific criteria.

8. Contents

8.1. Course	**Teaching and learning methods**	**Remarks**
1. Introduction. Number systems and codes	Presentations, discussions	N/A
2. Number representation in computers		
3. Binary representations and place orders		
4. Organization of a computer system		
5. The Central Processing Unit – CPU		
6. The memory		
7. Peripherals		
8. Architecture of x86 (IA-32) microprocessors		
9. RISC and CISC architectures. Assessing the performance of computer systems. Amdahl's law		
10. Basics of assembly language (on 32 bits)		
11. Instructions of assembly language		
12. Operations in assembly language		
13. Multi-module programming in assembly language		
14. Recap. Final Q&A		

Bibliography

1. Al. Vancea, F. Boian, D. Bufnea, A. Andreica, A. Darabant, A. Navroschi – Arhitectura calculatoarelor. Limbajul de asamblare 80x86., Editura Risoprint, Cluj-Napoca, 2014.
2. Al. Vancea, F. Boian, D. Bufnea, A. Gog, A. Darabant, A. Sabau – Arhitectura calculatoarelor. Limbajul de asamblare 80x86., Editura Risoprint, Cluj-Napoca, 2005.
3. A. Gog, A. Sabau, D. Bufnea, A. Sterca, A. Darabant, Al. Vancea – Programarea în limbaj de asamblare 80x86. Exemple si aplicatii., Editura Risoprint, Cluj-Napoca, 2005.
4. Randal Hyde – The Art of Assembly Programming, No Starch Press, 2003. (http://homepage.mac.com/randyhyde/webster.cs.ucr.edu/www.artofasm.com/DOS/index.html)
5. Irvine, K.R., 2015. Assembly language for x86 processors.
6. Kusswurm, D., 2014. Modern X86 Assembly Language Programming. Springer.
7. Carter, P.A., 2004. PC Assembly Language. Github: (http://pacman128.github.io/static/pcasm-book.pdf)
8. Cavanagh, J., 2013. X86 Assembly Language and C Fundamentals. CRC Press.
9. Guide, P., 2011. Intel® 64 and ia-32 architectures software developer's manual. Volume 3B: System programming Guide, Part, 2, p.11. (http://www.facweb.iitkgp.ac.in/~goutam/compiler/readingMaterial/intelXeon/253665.pdf)
10. Bartlett, Jonathan. "Nasm (Intel) Assembly Language Syntax." In Learn to Program with Assembly: Foundational Learning for New Programmers, pp. 271-273. Berkeley, CA: Apress, 2021.
11. Zhirkov, Igor, and Igor Zhirkov. "Assembly Language." Low-Level Programming: C, Assembly, and Program Execution on Intel® 64 Architecture, pp 17-38, 2017

8.2. Seminar/ laboratory	**Teaching and learning methods**	**Remarks**
1. Conversions and operations in different number bases. Familiarization with laboratory equipment	Hands-on exercises using educational test boards, FPGA boards, specialized software (simulators), blackboard presentations, additional explanations, and discussions	N/A
2. Arithmetic instructions (1). Declaring variables/constants		
3. Arithmetic instructions (2). Signed conversion instructions		
4. Bitwise operations. Comparison, conditional jump, and looping instructions. String operations.		
5. Operations on arrays of bytes/words/doublewords/quadwords		
6. System function calls. Multi-module programming (asm+asm)		
7. Lab test		

Bibliography

1. Al. Vancea, F. Boian, D. Bufnea, A. Andreica, A. Darabant, A. Navroschi – Arhitectura calculatoarelor. Limbajul de asamblare 80x86., Editura Risoprint, Cluj-Napoca, 2014.
2. Al. Vancea, F. Boian, D. Bufnea, A. Gog, A. Darabant, A. Sabau – Arhitectura calculatoarelor. Limbajul de asamblare 80x86., Editura Risoprint, Cluj-Napoca, 2005.

3. A. Gog, A. Sabau, D. Bufnea, A. Sterca, A. Darabant, Al. Vancea – Programarea în limbaj de asamblare 80x86. Exemple si aplicatii., Editura Risoprint, Cluj-Napoca, 2005.
4. Randal Hyde – The Art of Assembly Programming, No Starch Press, 2003. (http://homepage.mac.com/randyhyde/webster.cs.ucr.edu/www.artofasm.com/DOS/index.html)
5. Irvine, K.R., 2015. Assembly language for x86 processors.
6. Kusswurm, D., 2014. Modern X86 Assembly Language Programming. Springer.
7. Carter, P.A., 2004. PC Assembly Language. Github: (http://pacman128.github.io/static/pcasm-book.pdf)
8. Cavanagh, J., 2013. X86 Assembly Language and C Fundamentals. CRC Press.
9. Guide, P., 2011. Intel® 64 and IA-32 architectures software developer's manual. Volume 3B: System programming Guide, Part, 2, p.11. (http://www.facweb.iitkgp.ac.in/~goutam/compiler/readingMaterial/intelXeon/253665.pdf)
10. Zhirkov, Igor, and Igor Zhirkov. "Assembly Language." Low-Level Programming: C, Assembly, and Program Execution on Intel® 64 Architecture, pp 17-38, 2017

9. Evaluation

Type of activity	9.1 Evaluation criteria	9.2 Evaluation methods	9.3 Percentage in the final grade
9.4. Course	Problem-solving skills. Presence, (inter)activity	Written exam	70%
9.5. Seminar/ laboratory	Problem-solving skills.	In-person and/or written exam, or via the TEAMS platform, if necessary	30%
9.6 Minimum standard for passing			
• Requirements for taking the final written exam: grade on practical assignments ≥ 5 • Requirements for passing the exam: grade on practical assignments ≥ 5 AND grade on the written exam ≥ 5; • Formulating and solving typical logic design problems using the formal tools specific to the field.			

10. SDG labels (Sustainable Development Goals)

	○	Sustainable Development Generic Label

								
○	○	○	○	○	○	○	○	◉
								No label applies
○	○	○	○	○	○	○	○	○

Date of entry:
May 9, 2026

Signature of course coordinator

.....

Signature of seminar coordinator

.....

Date of approval in the department:
...

Signature of the head of department

.....