SYLLABUS

Computer systems architecture / Arhitectura sistemelor de calcul

University year 2025-2026

1. Information regarding the programme

1.1. Higher education institution	Babeș-Bolyai University
1.2. Faculty	Mathematics and Computer Science
1.3. Department	Computer Science
1.4. Field of study	Computer Science
1.5. Study cycle	Bachelor
1.6. Study programme/Qualification	Computer Science
1.7. Form of education	Full time

2. Information regarding the discipline

2.1. Name of the discipline	**Computer systems architecture (Arhitectura sistemelor de calcul)**					Discipline code	**MLRE004**
2.2. Course coordinator					Assoc.prof.phd. Mihai SUCIU		
2.3. Seminar coordinator					Assoc.prof.phd. Mihai SUCIU		
2.4. Year of study	1	2.5. Semester	1	2.6. Type of evaluation	E	2.7. Discipline regime	Mandatory

3. Total estimated time (hours/semester of didactic activities)

3.1. Hours per week	**5**	of which: 3.2 course	**2**	3.3 seminar/laboratory/project	**1/2/0**
3.4. Total hours in the curriculum	70	of which: 3.5 course	28	3.6 seminar/laboratory/project	**42**
Time allotment for individual study (ID) and self-study activities (SA)					**hours**
Learning using manual, course support, bibliography, course notes (SA)					20
Additional documentation (in libraries, on electronic platforms, field documentation)					10
Preparation for seminars/labs, homework, papers, portfolios and essays					20
Tutorship					10
Evaluations					20
Other activities:					
3.7. Total individual study hours	**80**				
3.8. Total hours per semester	**150**				
3.9. Number of ECTS credits	**6**				

4. Prerequisites (if necessary)

4.1. curriculum	-
4.2. competencies	-

5. Conditions (if necessary)

5.1. for the course	course room with video projector and whiteboard
5.2. for the seminar /lab activities	course room with video projector and whiteboard and workstations

6.1. Specific competencies acquired [1]

Professional/essential competencies	• development and maintenance of computer applications • use of computer tools in an interdisciplinary context
Transversal competencies	• applying rules of organized and efficient work, of responsible attitudes towards the didactic-scientific field, for the creative valorization of one's own potential, while respecting the principles and norms of professional ethics • efficiently carrying out activities organized in an interdisciplinary group and developing empathetic interpersonal communication, relationship and collaboration capacities with diverse groups

6.2. Learning outcomes

Knowledge	The student has the necessary knowledge to use computers, develop software programs and applications, process information. The graduate has knowledge related to programming, mathematics, engineering and technology and has the necessary skills to use them in creating complex computer systems.
Skills	The student can apply general rules to specific problems and produce relevant solutions. The graduate can develop, design and create new applications, systems or products using good practices in the field.
Responsibility and autonomy:	The student can work independently to identify complex problems and examine related issues to develop solution options and implement solutions. The graduate is able to combine diverse information to formulate solutions and generate development ideas for new products and applications.

[1] One can choose either competences or learning outcomes, or both. If only one option is chosen, the row related to the other option will be deleted, and the kept one will be numbered 6.

7. Objectives of the discipline (outcome of the acquired competencies)

7.1 General objective of the discipline	• Knowledge of computer architectural models, processor operation, use of computer information representation systems.
7.2 Specific objective of the discipline	• The student learns about computer architectural models, processor operation, and the use of computer information representation systems. • Introduction to assembly language programming, which ensures understanding of the architecture and functioning of a microprocessor. • Understanding the impact of the 80x86 processor architecture on the Windows operating system and its limitations. • Awareness of the architecture - operating system - programming languages triad and the interactions between them as the basic core of computer science. • Awareness of the influence that the basic functional principles of the von Neumann architecture have on the implementation of high-level programming languages. • Awareness of the architectural impact on the design and implementation techniques of high-level programming languages.

8. Content

8.1 Course	Teaching methods	Remarks
1. Data representation: Elementary data types, binary representations and placement orders, data organization and storage.	Exposition, conversation, debate, problematization, discovery	
2. Character encoding, integer encoding, signed and unsigned convention, sign bit, complement code, arithmetic operations, concept of overflow, conversion to a location of other dimensions.		
3. CS performance, 80x86 microprocessor architecture – structure, registers, address calculation, addressing modes, FAR and NEAR addresses		
4. Executive unit (EU) of 80x86 microprocessor.		
5. BIU unit of 80x86 microprocessor		
6. Assembly language elements.		
7. Standard directives for defining segments.		
8. Assembly language instructions.		
9. Impact of little endian representation on data access.		
10. Conditional and unconditional jump instructions.		
11. Representation of machine instructions.		
12. ASM-ASM multimodule programming:		
13. Implementing subprogram calls.		
14. Linking NASM modules with modules written in high-level languages.		
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. Boian F.M. Vancea A. Arhitectura calculatoarelor, suport de curs. Facultatea de Matematica si Informatica, Centrul de Formare Continua si Invatamânt la Distanta,. Ed. Centrului de Formare Continua si Invatamânt la Distanta, Cluj, 2002 6. Irvine, K.R., 2015. Assembly language for x86 processors. 7. Kusswurm, D., 2014. Modern X86 Assembly Language Programming. Springer. 8. Carter, P.A., 2004. PC Assembly Language. Github: (http://pacman128.github.io/static/pcasm-book.pdf) 9. Cavanagh, J., 2013. X86 Assembly Language and C Fundamentals. CRC Press. 10. 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) 11. BitDefender internal documentations – materiale postate pe pagina cursului 12. Cursuri si materiale suport postate pe site-ul cursului		
8.2 Seminar / laboratory	Teaching methods	Remarks
S1: Introducere în limbajul de asamblare IA-32.		
S2: Obtaining the offset / value of a variable.		
S3: Comparison instructions.		
S4: String instructions.		
S5: Function calls and text file operations.		
S6: Multi-module programming using assembly language.		
S7: Exam preparation: discussions and case studies.		
L1: Conversion between different number bases.		
L2: Simple arithmetic expressions.		
L3: Complex arithmetic expressions.		
L4: Bitwise instructions.		
L5: Simple string operations.		
L6: Complex string operations.		
L7: Function calls.		
L8: test		
L9: Text file operations.		
L10: Discussions, analysis and evaluation of laboratory work. Teaching the latest assignments.		
L11: Multimodule programming (asm + asm).		
L12: Multimodule programming (asm + C).		
L13: Preparation for practical exams: discussions and case studies.		
L14: 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. Boian F.M. Vancea A. Arhitectura calculatoarelor, suport de curs. Facultatea de Matematica si Informatica, Centrul de Formare Continua si Invatamânt la Distanta,. Ed. Centrului de Formare Continua si Invatamânt la Distanta, Cluj, 2002
6. Irvine, K.R., 2015. Assembly language for x86 processors.
7. Kusswurm, D., 2014. Modern X86 Assembly Language Programming. Springer.
8. Carter, P.A., 2004. PC Assembly Language. Github: (http://pacman128.github.io/static/pcasm-book.pdf)
9. Cavanagh, J., 2013. X86 Assembly Language and C Fundamentals. CRC Press.
10. 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)
11. BitDefender internal documentations – materiale postate pe pagina cursului
12. Cursuri si materiale suport postate pe site-ul cursului

9. Corroborating the content of the discipline with the expectations of the epistemic community, professional associations and representative employers within the field of the program

- This course is included in the curriculum of all major universities in Romania and abroad.
- This course provides the basic knowledge that any programmer should have.

10. Evaluation

Activity type	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Percentage of final grade
10.4 Course	Knowledge of the basic principles of the field.	written exam	60
	Understanding assembly language concepts.		
10.5 Seminar/laboratory	Problem solving by applying 32-bit assembly language programming principles.	Average grades obtained on laboratory assignments.	10%
	Develop and implement solutions in assembly language for a given problem.	Average of grades for laboratory tests.	30%
10.6 Minimum standard of performance			

- Grade: minimum 5 at each grading activity.
- Attendances: 75% attendance at seminar activities, 90% attendance at laboratory activities.
- Students with more than 2 unmotivated absences at the seminar/laboratory activities will not be able to take the exam in the normal session and or in the retake examination session (seminar activities are activities that take place on the following principle “activity during the semester ”, and they cannot be recovered or repeated for a

possible retake examination (these students will have to repeat this course in the next academic year)). Students with medical certificates for each of their absences are exempted from this rule.

11. Labels ODD (Sustainable Development Goals)[2]

Not applicable.

Date:	Signature of course coordinator	Signature of seminar coordinator
	Assoc.prof.phd. Mihai SUCIU	Assoc.prof.phd. Mihai SUCIU

Date of approval:	Signature of the head of department
	Assoc.prof.phd. Adrian STERCA

[2] Keep only the labels that, according to the *Procedure for applying ODD labels in the academic process*, suit the discipline and delete the others, including the general one for *Sustainable Development* – if not applicable. If no label describes the discipline, delete them all and write „*Not applicable*.”.