COURSE UNIT DESCRIPTION

Course unit title	Course unit code
Computer Architecture	

Lecturer(s)	Department where the course unit is delivered
Coordinator: prof. dr. Saulius Gražulis **Other lecturers:** -	Department of Computer Science Faculty of Mathematics and Informatics Vilnius University

Cycle	Type of the course unit
1st (BA)	Compulsory

Mode of delivery	Semester or period when the course unit is delivered	Language of instruction
Face-to-face/online	3 semester	Lithuanian, English

Prerequisites
Prerequisites: -

Number of credits allocated	Student‘s workload	Contact hours	Individual work
5	134	66	68

Purpose of the course unit: programme competences to be developed		
Purpose of the course unit: to shape understanding of the real processing of computer programs as iterative transformation of memory data state using computer's instructions, to understand computer hardware implementation principles, to master the system of machine level notions, to learn read and write machine level software. ***Generic competences:*** • Ability to analyse and organise the information *(GK1)*. • Ability to apply the knowledge in practice *(GK2)*. • Ability to organise and plan the work, to work in a team as well as individually *(GK3)*. ***Specific competences:*** • Programming *(SK6)*. • Systems architecture (*SK7*).		
Learning outcomes of the course unit: students will be able to	**Teaching and learning methods**	**Assessment methods**
operate computer architecture concepts and notions fluently and focused	Teaching methods: • Lectures; • Laboratory works. Learning methods: • Actual knowledge gathering and accumulation; • Knowledge synthesis – generalization, abstraction and aggregation of actual knowledge; • Knowledge analysis – new knowledge matching with aggregated knowledge, their verification and correction is needed; • Application of aggregated and validated knowledge.	Examination. Laboratory works presentation. Report. Quiz. Criteria: • Ability to solve practical exercises; • Ability to develop, debug, trace, explain and modify programs in assembler; • Ability to explain operation principles of computer and CPU components ;
understand computer systems diagnostic messages done in machine oriented terms		
understand influence of computer architecture on program performance and correctness		
possess concepts needed to learn programming languages		

Course content: breakdown of the topics	Contact hours							Individual work: time and assignments	
	Lectures	**Tutorials**	**Seminars**	**Practice**	**Laboratory work**	**Practical training**	**Contact hours**	**Individual work**	**Assignments**
1. Introduction to Computer Architecture. Basic Computer structure. Switching circuits.	2				2		**4**	**4**	I, II. Investigating logic circuits and computer components on transistor and logic level (using Logisim or similar simulation software); III, IV. Writing programs in assembly language for various architectures and investigating their execution on simulation software.
2. Logic gates and combinational logic. Complete sets of logic functions, Post's theorem.	2				2		**4**	**4**	
3. Computer arithmetic. Positional systems and number representation.	2				2		**4**	**4**	
4. Stateful computer elements. Triggers and re-gisters. Memory.	2				2		**4**	**4**	
5. The CPU data tract and its control. Finite state automata. Microprogramming.	2				2		**4**	**4**	
6. Data representation in computers. Alternative integer and rational number representations. Char-acter data and character encodings. Unicode.	2				2		**4**	**4**	
7. Floating point numbers.	2				2		**4**	**4**	
8. Representation of variable size data. Advanced representations of numbers. Multiple precision arithmetic. Examples of CISC and RISC commands for number and character processing.	2				2		**4**	**4**	
9. Example of a CPU implementation. CPU control sequencer. Pipelines. Various types of computer architectures (Stack, Accumulator, Memory-Memory, Load-Store), CISC vs RISC. Zero, One, Two, Three address instructions.	2				2		**4**	**4**	
10. RISC-V ISA	2				2		**4**	**4**	
11. Assembler programming. Command mnemonics, operands, addressing modes, labels, sections, macroassembler. Compilation from high level languages (C).	2				2		**4**	**4**	
12. Pipelined architectures. Memory cache. RISC-V emulator. Examples and analysis.	2				2		**4**	**4**	
13. CISC CPUs. x86 architecture example.	2				2		**4**	**4**	
14. Virtual memory. Paging. Segmenting. Memory protection.	2				2		**4**	**4**	
15. Microcontrollers. Example: AVR. Interrupts and interrupt handling. Peripheral devices: timers, ADC.	2				2		**4**	**4**	
16. Future, exotic, non-standard architectures: ANN, tagged architectures, cell matrix, FPGA, FORTH machines. Hardware description languages.	2				2		**4**	**4**	
Self-preparation and exam							**2**	**4**	
Total	**32**				**32**		**66**	**68**	

Assessment strategy	**Weight %**	**Deadline**	**Assessment criteria**
Lecture quizzes	10	10 min. at the beginning of each practical.	4-question quiz covering several recent lectures (Bloom's 1 an 2 level questions) using an electronic teaching environment (Moodle, Open edX or similar).
Intermediate quiz	15	mid-term	Approx. 30-question quiz covering several recent lectures (Bloom's 1 to 9 level questions) using an electronic teaching environment (Moodle, Open edX or similar).
Evaluation of practical assignments	50	After each practical according to the schedule announced by the teacher of the practical	The practical work schedule and evaluation criteria are announced by the teacher of each group. The teacher who leads practical work grades it and communicates the assigned grade.
Analysis of an assigned computer architecture example	10	end of term	Students provide a written (4 pages, A4 format, 9pt) technical report on a computer architecture which they studied themselves, or 5 min oral presentation with slides on that architecture. The oral presentation is available for achieving students, with permission of the teaching professor, and may be accepted as a final exam.

Final exam	15	end of term	approx. 30-question quiz covering several recent lectures (Bloom's 1 to 9 level questions) using an electronic teaching environment (Moodle, Open edX or similar).
			
			To be eligible for the exam, students must fulfil all following criteria:
			1. carry out at least one practical work and get a positive grade for the practicals;
			2. have enough accumulated points to be able to pass the exam in principle if they score maximum points at the exam quiz;
			3. students who have shown excellent performance during the term may be freed from the final exam quiz and given the opportunity to present their research on computer architectures as the oral presentation, provided:
			    1. collect at least 60% of all possible points during the theory course (e.g. at least 150 points from the 250 possible points from the intermediate exam and the lecture quizzes);
			    2. they have carried out all practical assignments in the scheduled time;
			    3. they have positive recommendations from the teacher of their practical team.
			If there are more students wishing to make oral presentations than the allocated time permits, students with higher grades will be given priority.
			Unless you are allowed to take the oral exam (i.e. make the presentation), participation in the final exam quiz is obligatory to pass the course, regardless of the accumulated points. Students who do not show up in the final exam will be indicated as such in the exam grading report. To pass the exam, on must score at least 50% of possible points.

Total	100		The final mark is obtained summing up all points obtained for each task, quiz or assignment, dividing them by 100 and rounding to the *higher* integer (i.e. 0.001 is rounded to 1.0; 9.1 is rounded to 10).

Author	Publishing year	Title	Number or volume	Publisher or URL
Required reading				
Andrew S.Tanenbaum	2005	Structured computer organization		Prentice Hall PTR, Fifth Edition
D. A. Patterson and J. L. Hennessy	2017	Computer Organization and Design: The Hardware/ Software Interface. RISC-V edition.		Elsevier
A. Waterman, Y. Lee, D. Patterson, and K. Asanović	2011	The RISC-V instruction set manual. Volume I: base user-level ISA. Version 1.0.	Vol. 1, ver. 1.0	https://inst.eecs.berkeley.edu/~cs250/fa11/handouts/riscv-spec.pdf
Recommended reading				
Antanas Mitašiūnas	2016	Computer architecture. Teaching book (in Lithuanian Kompiuterių architektūra)		Vilnius, 126 p. http://www.mif.vu.lt/katedros/cs/Asmen/Kompiuteriu%20architektura.pdf
D. E. Knuth	2005	MMIX – A RISC Computer for the New Millennium	Vol. 1, Fasc. 1	Addison–Wesley, http://www.mmix.cs.hm.edu/doc/fasc1.pdf, https://www-cs-faculty.stanford.edu/~knuth/fasc1.ps.gz
C. W. Kann	2016	Implementing a One Address CPU in Logisim		Gettysburg College; https://open.umn.edu/opentextbooks/textbooks/implementing-a-one-address-cpu-in-logisim
C. W. Kann	2019	Digital Circuit Projects: An Overview of Digital Circuits Through Implementing Integrated Circuits	Second Edition	Gettysburg College; http://cupola.gettysburg.edu/oer/1
C. W. Kann	2019	Introduction To MIPS Assembly Language Programming		Gettysburg College; https://cupola.gettysburg.edu/oer/2
M. J. Murdocca and V. P. Heuring	1999	Principles of Computer Architecture		Prentice Hall
D. A. Patterson and J. L. Hennessy	2013	Computer Organization and Design: The Hardware/Software Interface. MIPS edition.		Elsevier
E. Upton	2016	Learning Computer Architecture with Raspberry Pi		John Wiley & Sons
A. P. Malvino and J. A. Brown	1999	Digital Computer Electronics		McGraw-Hill
R. E. Bryant and D. R. O'Hallaron	2001	Computer Systems: A Programmer's Perspective	3rd Edition	https://github.com/smellslikekeenspirit/an-askreddit-list-of-compsci-books/blob/master/Randal%20E.%20Bryant%2C%20David%20R.%20O%E2%80%99Hallaron%20-%20Computer%20Systems.%20A%20Programmer%E2%80%99s%20Perspective%20%5B3rd%20ed.%5D%20(2016%2C%20Pearson).pdf
D. Goldberg	1991	What every computer scientist should know about floating-point arithmetic		https://doi.org/10.1145/103162.103163
J. L. Gustafson	2015	The End of Error: Unum Computing		CRC Press