COURSE OUTLINE

1. GENERAL

SCHOOL	School of Engineering, University of Patras		
DEPARTMENT	Computer Engineering and Informatics		
LEVEL OF COURSE	Undergraduate, Core		
COURSE CODE	CEID_23Y212	**SEMESTER OF STUDIES**	Spring, 4th
COURSE TITLE	Computer Architecture Laboratory		

INDEPENDENT TEACHING ACTIVITIES *if credits are awarded for separate components of the course, e.g. lectures, laboratory exercises, etc. If the credits are awarded for thewhole of the course, give the weekly teaching hours and the total credits*	**TEACHING HOURS PER WEEK**	**ECTS CREDITS**
Lectures	1	
Laboratory Exercises	2	2
Total	**3**	**2**
Add rows if necessary. The teaching organization and methods used are described in detail in (d).		

COURSE TYPE *general background, special background, specialized general knowledge, skills development*	General Background Specialized General Knowledge
PREREQUISITE COURSES:	Recommended prerequisite knowledge: • Fundamental Principles of Computer System Organization and Operation (CEID_22Y104) • Logic Design (CEID_23Y107) • Computer Architecture (CEID_23Y203)
TEACHING AND ASSESSMENT LANGUAGE:	Greek
THE COURSE IS OFFERED TO ERASMUS STUDENTS	No
COURSE WEBPAGE (URL)	https://eclass.upatras.gr/courses/CEID1493/

2. LEARNING OUTCOMES

Learning outcomes

The course learning outcomes, specific knowledge, skills, and competences of an appropriate level that the students will acquire with the successful completion of the course are described.

Consult Appendix A

- *Description of the level of learning outcomes for each qualifications cycle, according to the Qualifications Framework of the European Higher Education Area*
- *Descriptors for Levels 6, 7 & 8 of the European Qualifications Framework for Lifelong Learning and Appendix B*
- *Guidelines for writing Learning Outcomes*

Upon successful completion of the course, students will be able to:

1. Develop, debug, and execute RISC-V assembly language programs using the RARS simulator.
2. Apply basic and advanced arithmetic and logical instructions of the RISC-V instruction set, demonstrating an understanding of concepts such as carry, overflow, and internal data representation.
3. Manage the stack and heap to implement procedures and dynamic data structures, applying the calling conventions of the RISC-V architecture.

4. Understand and handle exceptions during program execution by utilizing the architectural support provided by RISC-V.
5. Observe and analyze the instruction flow within a processor using the visualization features of Ripes, identifying dependencies and applying optimization techniques.
6. Compare different microarchitectural implementations of processors (single-cycle and pipelined designs), recognizing their impact on overall program performance.

General Abilities

Taking into consideration the general competences that the degree-holder must acquire (as these appear in the Diploma Supplement and appear below), at which of the following does the course aim?

Search for, analysis and synthesis of data and information, with the use of the necessary technology	*Project planning and management*
Adapting to new situations	*Respect for difference and multiculturalism*
Decision-making	*Respect for the natural environment*
Working independently	*Showing social, professional and ethical responsibility and sensitivity to gender issues*
Team work	*Criticism and self-criticism*
Working in an international environment	*Production of free, creative and inductive thinking*
Working in an interdisciplinary environment	
Production of new research ideas	*Others…*

Decision-making
Working independently
Team work
Production of new research ideas
Production of free, creative and inductive thinking

3. COURSE CONTENT

The Computer Architecture Laboratory is a practice-oriented course that complements and reinforces the theoretical instruction of computer architecture. Through a series of targeted laboratory exercises, students gain hands-on experience in developing, executing, and debugging RISC-V assembly language programs, as well as in understanding the internal operation of modern processors. The course is built around two educational simulators, RARS and Ripes, offering students both programming practice at the Instruction Set Architecture (ISA) level and visual insight into processor internals.

In the initial exercises, students become familiar with the RARS simulator, focusing on assembly translation, monitoring registers and memory, and controlling program execution. They strengthen their understanding of basic arithmetic and logical operations, carry management, and algorithm implementation using RISC-V assembly language. The course then advances to more complex topics such as procedure calls using the stack, memory management with pointers and the heap, and exception handling.

The second part of the course shifts focus to the microarchitectural aspects of instruction execution, using the Ripes simulator. Students have the opportunity to observe step-by-step the flow of instructions through the stages of a processor (fetch, decode, execute, memory, writeback) and to compare single-cycle and pipelined processor implementations. Through exercises that combine data dependency analysis and instruction reordering, students develop a deep understanding of performance optimization techniques and the role of hardware design in execution speed.

Overall, the course strengthens the connection between low-level programming and hardware operation, preparing students for further study in areas such as compilers, operating systems, and digital system design.

4. TEACHING AND LEARNING METHODS – ASSESSMENT

<table>
<tr><td>TEACHING METHOD
Face-to-face, Distance learning, etc.</td><td>Face-to-face</td></tr>
<tr><td>USE OF INFORMATION AND COMMUNICATION TECHNOLOGIES
Use of ICT in teaching, laboratory education, communication with students</td><td>Powerpoint Slides
eClass</td></tr>
<tr><td>TEACHING ORGANIZATION
The manner and methods of teaching are described in detail.
Lectures, seminars, laboratory practice, fieldwork, study and analysis of bibliography, tutorials, placements, clinical practice, art workshop, interactive teaching, educational visits, project, essay writing, artistic creativity, etc.
The student's study hours for each learning activity are given, as well as the hours of non-directed study according to the principles of the ECTS</td><td>
<table>
<tr><th>Activity</th><th>Semester Workload</th></tr>
<tr><td>Lectures</td><td>13 hours</td></tr>
<tr><td>Laboratory Exercises</td><td>26 hours</td></tr>
<tr><td>Study and preparation of laboratory exercises</td><td>29 hours</td></tr>
<tr><td>Writing of laboratory reports</td><td>5 hours</td></tr>
<tr><td>Assessment of the laboratory component / Laboratory exams</td><td>2 hours</td></tr>
<tr><td>Total number of hours for the Course (25 hours of work-load per ECTS credit)</td><td>75 hours</td></tr>
</table>
</td></tr>
<tr><td>STUDENT ASSESSMENT
Description of the evaluation procedure

Language of evaluation, methods of evaluation, summative or conclusive, multiple-choice questionnaires, short-answer questions, open-ended questions, problem-solving, written work, essay/report, oral examination, public presentation, laboratory work, clinical examination of patient, art interpretation, other

Specifically defined evaluation criteria are given and if and where they are accessible to students.</td><td>Assessment is conducted in Greek.

Student performance is primarily evaluated through direct, in-person observation and verification of the correctness of the programs they implement during the laboratory exercises, as well as through the submission of corresponding reports. In addition, a final laboratory examination is held at the end of the semester, during which students are required to develop and execute programs in real time.</td></tr>
</table>

5. RECOMMENDED LITERATURE

Suggested bibliography:

- “Digital Design and Computer Architecture, RISC-V Edition”, S. L. Harris, D. Harris, Morgan Kaufmann, 2021.
- “Computer Organization and Design, RISC-V Edition”, D. A. Patterson, J. L. Hennessy, 2nd Edition, Morgan Kaufmann, 2020.
- “Computer Architecture: A Quantitative Approach”, J. L. Hennessy, D. A. Patterson, C. Kozyrakis, 7th Edition, Morgan Kaufmann, 2025.
- “Digital Design and Computer Architecture, ARM Edition”, S. L. Harris, D. Harris, Morgan Kaufmann, 2015.
- “Computer Organization and Architecture”, 11th edition, William Stallings, Published by Pearson, 2019.

Related academic journals:

- IEEE Micro
- IEEE Transactions on Computers
- IEEE Transactions on Dependable and Secure Computing
- ACM Transactions on Architecture and Code Optimization