Presentation

ARQCP Course

Arquitetura de Computadores
Licenciatura em Engenharia Informática

2024/25

Paulo Baltarejo Sousa
pbs@isep.ipp.pt

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Outline

Overview

Course

- Computer Architecture, Arquitetura de Computadores (ARQCP)
- Year: 2^{st}
- Semester: 1^{st}
- Hours per week
  - T: 1
  - TP: 1
  - PL: 3
- European Credit Transfer and Accumulation System (ECTS): 5
  - Each ECTS corresponds **25** to **28** working hours.
    - Working hours include **classes** (contact) and (plus) **autonomous** (out of classes) hours.

Working hours

- Each ARQCP student must **dedicate between 125 to 140 working hours per semester**.
  - 5 ECTS correspond to a range between (5 * 125 =) **125** and (5 * 128 =) **140** working hours.
- Contact hours (classes): (12 + 12 + 45 =) **69**
  - T: 1 * 12 (weeks) = 12
  - TP: 1 * 12 (weeks) = 12
  - PL: 3 * 15 (weeks) = 45
- Autonomous working hours : (125 - 69 =) **56** to (140 - 69 =) **71**
  - 56 / 15 = **3.73 hours per week**.
  - 71 / 15 = **4.73 hours per week**.

Be aware

You must dedicate $\sim$4 hours per week out of classes

Week mapping [1]

Week	Dates	Week	Dates
1	16/09 – 22/09/2024	9	11/11 – 17/11/2024
2	23/09 – 29/09/2024	10	18/11 – 24/11/2024
3	30/09 – 06/10/2024	11	25/11 – 01/12/2024
4	07/10 – 13/10/2024	12	02/12 – 08/12/2024
5	14/10 – 20/10/2024	13	09/12 – 15/12/2024
6	21/10 – 27/10/2024	14	16/12 – 22/12/2024
7	28/10 – 03/11/2024	15	06/01 – 12/01/2025
8	04/11 – 10/11/2024		

- Public holidays:
  - 01/11/2024 (Dia de Todos os Santos), Friday;
- Christmas holiday: 22/12/2024 – 05/01/2025

[1] We consider the first week day is on Monday and the last week day is on Sunday.

Lecturers

- Carlos Gonçalves (CAG), cag@isep.ipp.pt
- Luís André (LAO), lao@isep.ipp.pt
- Luís Nogueira (LMN), lmn@isep.ipp.pt
- Orlando Sousa (OMS), oms@isep.ipp.pt
- Paulo Baltarejo Sousa (PBS), pbs@isep.ipp.pt
- Paulo Ferreira (PDF), pdf@isep.ipp.pt
- Rafael Silva (RVS), rvs@isep.ipp.pt
- Ricardo Severino (SEV), sev@isep.ipp.pt

Required previous knowledge

- ARQCP course assumes students have previously acquired knowledge and skills in:
  - Algorithms (APROG);
  - The principles of computer technology (PRCMP);
  - Binary and hexadecimal coding (PRCMP);
  - Representation of strings and arrays (PRCMP and APROG);
  - Boolean algebra (PRCMP).

Overview

- This course aims at providing students with **fundamental knowledge in computer architecture**, allowing them **to write better programs** (robust, more secure, and with better performance) **by learning what is going on "under the hood"** of a computer system.
  - It emphasis on the **programmer's perspective of a computer system**.
- It explores the **interactions between C programs** and their **machine-language counterpart on x86-64 architectures**.

Goals

- At the end of the course, the students should be able to:
  - **Explain** the hierarchy of abstractions and implementations that comprise a modern computer system and how these impact on the performance and correctness of applications
  - **Explain** what assembly language is and how it fits into the programming model
  - **Explain** how compilation systems work
  - **Systematize opportunities to optimize application programs** with techniques that exploit the designs of modern processors and memory systems
  - **Develop efficient and reliable software solutions** based on the knowledge of the underlying computer system and on how high-level language programs are mapped onto a machine and executed
  - **Organize solutions in a team** in order to achieve specific objectives constrained by time deadlines

Teaching/learning methodology and Planning

Lectures (T)

- Expository and interrogative

Week nr	Topic
1	High-level programs and data abstractions
2	Continue
3	Continue
4	Machine programming
5	Continue
6	Continue
7	Continue
8	Continue
9	Memory organization
10	Continue
11	Optimizing programs performance
12	Continue
13	Project support
14	Project support
15	Project evaluation

Practical (TP)

- Expository and interrogative

Week nr	Topic
1	Overview of compilation systems. C programming language basics
2	Static allocation of arrays and strings in C. Memory access through pointers
3	Assembly programming language basics
4	Arithmetic operations in Assembly
5	Control flow in Assembly
6	Working with strings and arrays in Assembly
7	Stack and functions without arguments in Assembly
8	Stack and functions with arguments in Assembly. Handling local variables
9	Bitwise operations (C and Assembly)
10	Heterogeneous data structures (C and Assembly)
11	Dynamic memory allocation (C and Assembly)
12	Dynamic memory allocation (C and Assembly)
13	Project support
14	Project support
15	Project evaluation

Labs (PL)

- Exercise- and Problem-Based Learning
- Project-Based Learning

Week nr	Topic
1	Module 0 – Introduction to C programming. Modules and Makefiles
2	Continue
3	Module 1 – C programming: static memory allocation, strings, arrays and pointers
4	Continue
5	Module 2 – Machine programming: basic Assembly programming
6	Continue
7	Module 3 – Machine programming: strings and arrays Integration project
8	Continue
9	Module 4 – Machine programming: functions with arguments and local variable Integration project
10	Continue
11	Module 5 – Complex data structures (C and Assembly) Integration project
12	Continue
13	Integration project
14	Integration project
15	Integration project evaluation

Materials and Support

Teaching Material (I)

- Support material available at the subject's Moodle page.
  - T pdf files are used for lectures.
  - TP pdf files are used for practical classes.
  - PL pdf files are used for lab classes.
  - **zip files are for students' self study** and support to PL classes' exercises.
- Books:
  - "Computer Systems: A Programmer's Perspective", 3rd Edition, Randal E. Bryant, David R. O'Hallaron, Prentice Hall, 2016
  - "Professional Assembly Language", Richard Blum, Wiley Publishing, 2005
  - "The C Programming Language, Second Edition", Brian W. Kerninghan and Dennis M. Ritchie, Prentice Hall, 1988

Teaching Material (II)

- Tools:
  - GCC, the GNU Compiler Collection (`http://gcc.gnu.org/`)
  - GDB, the GNU Project Debugger (`http://www.gnu.org/software/gdb`)
  - Make, the GNU tool to compile and link a program (`http://www.gnu.org/software/make/`)
  - Valgrind, a suite of tools for debugging and profiling Linux programs (`http://www.valgrind.org/`)
  - Bitbucket, GIT code management (`http://bitbucket.org`)
  - VirtualBox, machine virtualization (`http://www.virtualbox.org`)

Support (I)

- There is no specific schedule for students support.
  - Whenever students need support, must send an email to any lecturer (preferably to PL lecturer).

Email rule

- **Subject/Assunto**: **ARQCP**
  - If you do not follow, the email is ignored (i.e., it is not ”received”)

Support (II)

- For installing software
  - Help is provided until week 2.
  - Recall, **until week 2.**
  - **No help is provided after week 2.**

Assessment

Components and Rules (I)

- **Components**
  - **PAG** (Practical Assessment Grade)
    - M1 – Individual assessment exercise (High-level language)
    - M2 – Individual assessment exercise (Low-level language)
    - M3 – (Team) Integrating project (Phase 1), LAPR3 Sprint 2
    - M4 – (Team) Integrating project (Phase 2), LAPR3 Sprint 3
  - **E** (Final Exam), written
- **The PAG final grade is determined as follows**:
  - PAG = (M1 * 0.15 + M2 * 0.15 + M3 * 0.10 + M4 * 0.20) / 0.60
    - PAG is graded in the interval [0,20]
- **The ARQCP final grade (CF) is determined as follows**:
  - CF = PAG * 0.60 + E * 0.40.
    - CF is graded in the interval [0,20]
- **Minimal grades**
  - **To grant access to the E**:
    - PAG >= 7.50 (7.50/20,00).
  - **To get E grade**
    - The minimum E grade is 7.50/20.00.

Components and Rules (II)

- The M3 and M4 **teams are the same as in LAPR3**.
- For those that **are not enrolled in LAPR3, create a team with students from the same PL class**.
  - In this case, **all teams creation are supervisioned by PBS**, which means, that you have to send an email to `pbs@isep.ipp.pt` with the following information:
    - PL class identifier;
    - Team members (name and number);
    - Each team is composed by 4 students.

Individual Exercises Assessment

- **M1** – Individual assessment exercise (High-level language):
  - A programming problem to be solved individually by the student during one of the lab classes in week 5.
  - All the topics discussed in lab classes until that point can be assessed in M1, but it should focus on the topics of Module 0 and Module 1.
  - The expected duration is 1h 30m.
- **M2** – Individual assessment exercise (Low-level language):
  - A programming problem to be solved individually by the student during one of the lab classes in week 9.
  - All the topics discussed in lab classes until that point can be assessed in M2, but it should focus on the topics of Module 2 and Module 3.
  - The expected duration is 1h 30m.
- M1 and M2 are graded in the interval [0,20]

Integration Project Assessment

- **M3** and **M4** grades are computed as follows:
  - **M*x*** = **IMPL** * **Factor**;
- **IMPL**, project implementations (present into group repository)
  - Refers to the amount of Use Cases (UCs) implemented
  - UC implementations quality
  - The system is working?
  - Errors, warnings and etc.
  - **Graded in the interval [0,20]**;
- **Factor**, refers to the discussion with each group member (**individually**)
  - Infer (evaluate) the student's knowledge
  - The amount of work.
  - **Graded in the interval [0,100%]**;
  - Some questions could be:
    - Could you explain this functionality?
    - Why are you using this approach?
    - ...
- M3 and M4 are graded in the interval [0,20]

Schedule

Week nr	Moment
1	
2	
3	
4	
5	M1 – Individual assessment exercise (High-level language)
6	
7	
8	
9	M2 – Individual assessment exercise (Low-level language)
10	
11	M3 – Integrating project (Phase 1), LAPR3 Sprint 2
12	
13	
14	
15	M4 – Integrating project (Phase 2), LAPR3 Sprint 3