Curso Competências Transversais em Engenharia - Formação Pedagógica
2016/2017 - 1º semestre
1º Exercício
Pedro Miguel Salgueiro dos Santos

Page of Curricular Unit: **Computer Architecture and Operating Systems**

- Study cycle: B.Sc. in Electrical and Computer Engineering, 3rd year
- Type: Mandatory
- Teaching language: English
- ECTS/ number of hours: 6.5 ECTS ; 172 hours - 60 contact hours, 112 self-study hours (see ratios at end of “Competencies and Learning Objectives” table)
- Prerequisites: RTL (Verilog/VHDL); C/C++
- Competences and Learning Objectives:

Competency	*Competency Typology*	*Domain of L.O.*	*Level within domain*	*Learning objective*	*Hours **	
					C.T.	*S.S.*
Architecture and operation of a Personal Computer	Knowledge	Cognitive	Knowledge	List the components of a personal computer (PC); describe their function	2	4
			Understanding	Explain the dataflow between components	2	2
Architecture and operation of a Central Processing Unit	Knowledge	Cognitive	Knowledge	Describe the modules of a Central Processing Unit (CPU)	2	4
			Understanding	Summarize the dataflow inside a CPU	2	4
CPU operation in a practical context	Knowledge	Cognitive	Application	Apply knowledge about CPU operation to use cases	2	4
			Analysis	Analyse the modularity of a CPU architecture -- how composing modules may be replaced by others of similar function but different design	2	4
Design and implementation of CPU modules	Knowledge	Cognitive	Synthesis	Develop a design for a particular module, based on a requirements specification	6	18
Design and implementation of working CPU	Knowledge	Cognitive	Synthesis	Combine existing modules into a sequential organization that implements a fully-operational CPU	3	8

* C.T.: contact time; S.S.: self-study time

Competency	*Competency Typology*	*Domain of L.O.*	*Level within domain*	*Learning objective*	*Hours **	
					C.T.	*S.S.*
Operating Systems	Knowledge	Cognitive	Knowledge	State the elements of an operating system (OS) and their function	2	2
			Understanding	Explain interaction between OS modules, host hardware and guest processes	2	2
File Systems	Knowledge	Cognitive	Knowledge	List the different types of filesystems and distinguishing features	2	4
			Understanding	Explain the operation of a filesystem	2	2
Memory Segmentation and Page Files	Knowledge	Cognitive	Knowledge	Define memory segmentation and page files	2	2
			Understanding	Associate page files to memory areas of processes	2	2
Processes, Concurrency and Schedulers	Knowledge	Cognitive	Knowledge	Describe the concepts of pre-emption and scheduling policies	2	4
			Understanding	Explain the operation of a scheduler	2	2
			Application	Demonstrate scheduling policy enforcement in actual scenarios	2	4
Drivers and Concurrent Resource Access	Knowledge	Cognitive	Knowledge	Define the purpose of a driver and identify the problem of race-conditions	2	2
			Understanding	Describe the operation of semaphores	2	2
Compilers	Knowledge	Cognitive	Knowledge	Define the purpose of a compiler	2	2
			Understanding	Summarize the operation of a compiler	2	2
			Application	Demonstrate compiling procedures in actual scenarios	2	4
Design and implementation of a working OS	Knowledge	Cognitive	Analysis	Explain how OS modules interface with each other	2	2
			Synthesis	Develop OS modules: File system, Page File Manager, Scheduler, Driver and Compiler	6	18
			Synthesis	Combine existing modules into an operational OS	3	8
Total hours					60	112
Total hours (relative)					.35	.65

* C.T.: contact time; S.S.: self-study time

Program:

1. Internal archictecture of a PC and dataflow
    a. Central Processing Unit (CPU), RAM memory, hard drive, peripherals.
    b. Distinguish volatile and permanent storage devices.
    c. Data flow from hard drive to CPU and vice-versa
2. Internal architecture of a CPU and data flow
    a. Program and data registers, arithmetic and logic units, control flow, program stack
    b. Reading a program instruction and converting the instruction into control signals
    c. Flow of data into registers and arithmetic/logic units caused by the instruction
3. Examples of CPU usage and modularity of architecture
    a. Examples of CPU usage with actual values and various arithmetic/logic operations
    b. Evaluation of output of the said examples.
    c. Examples of CPU modularity: modules can be replaced by modules of similar function but different design
4. Design and implementation of individual or multiple CPU modules
    a. Selection of a particular module of CPU operation and creation of a requirements specification for that module
    b. Planning and design of module to fulfill the requirement specification
    c. Implementation in RTL and test of module; write design specification
5. Design and implementation of a fully-working CPU
    a. Planning of control and data lines to connect all CPU modules
    b. Integrate all CPU modules into a fully-operational CPU
    c. Design and run tests to evaluate CPU operationality and functionalities
6. Operating Systems
    a. Concept of operating system, kernel; OS as bridge between host hardware and guest processes
    b. Composing elements of operating systems
7. Memory Segmentation and Page Files
    a. Concept of dedicated memory area per process
    b. Purpose and operation of a Page File Manager
8. File Systems
    a. Purpose and operation of file systems
    b. Different types of file systems
9. Processes, Pre-emption and Schedulers
    a. Concept of process, parallel execution and pre-emption
    b. Purpose and operation of schedulers
    c. Scheduling policies
10. Drivers and Concurrent Resource Access
    a. Purpose and design of drivers; race-conditions
    b. Semaphores and their operation
11. Compilers
    a. Concept of human-readable and machine-readable programs
    b. Purpose and operation of compilers
    c. Design of compilers
12. Design and Implementation of OS modules
    a. Planning and implementation of OS modules
    b. Integrate all OS modules into fully-operational OS
    c. Design and run tests to evaluate CPU operationality and functionalities

Curso Competências Transversais em Engenharia - Formação Pedagógica
2016/2017 - 1º semestre
2º Exercício
Pedro Miguel Salgueiro dos Santos

Page of Curricular Unit: **Computer Architecture and Operating Systems**

1. Teaching methods and learning activities

Working Method (and why):

- **Presencial** -- explain fundamental concepts and operation in proximity with students; present use cases/exercises and discuss their solving strategy
- **Laboratorial work** -- in order to develop a hands-on course project
- **Moodle** -- for content distribution and to serve as contact channel between teacher and students

Type of classes: theoretical, theoretical/practical, practical

Methods (and why):

- **Lectures** -- List relevant elements, explain interaction between elements, describe operation at an abstract level
- **Use-cases presentation** -- concrete realizations of the high-level explanation taught; involve students in solving use-cases
- **Electronic support (simulator)** -- explanations (both high-level and concrete realizations) can be supported by a electronic simulator that allows to visualize the internal operation of the CPU
- **Exercises** -- for students to train the use of the concepts addressed in the course
- **Project** -- for students to implement the concepts addressed in the course

Amount of time allocated to each course component:

Name	Time (Hours)
Project	46
Report	6
Self-study	60
Lectures attendance	60

Total: 172 hours

2. Class Preparation

Class Name / Competencies: Architecture and Operation of a Central Processing Unit

Learning objectives *(replicated from 1st assignment)*:

- Describe the modules of a Central Processing Unit (CPU): arithmetic and logical unit, registers (program counter, MBR, etc.), buses
- Summarize the dataflow inside a CPU

Themes: Instruction Set Architecture, Integer Java Virtual Machine, Arithmetic/Logic Unit, Data Registers and Buses, Datapath, Program Memory, Memory Buffer Register, Program Counter, Instructions, Micro-commands, Control Lines

Number of hours: 2 hours in lecture; 4 hours of self-study

Type of class: Theoretical/practical

General Strategies:

- Explanation of the relevant concepts (previously addressed in theoretical lecture)
- Presentation of concrete use cases and applications of the concepts addressed
- Provide exercises for students to solve alone or in pairs, and provide solutions and solving strategy immediately afterwards for feedback
- Use of an electronic support (visualization tool) to show steps of use-case and exercises' solving strategies

Structure of class:

(Start of class)

Minutes 0-5
Overview of themes to be addressed in class:

- The Instruction Set Architecture (ISA) and a particular instance: the IJVM
- Datapath of IJVM and data and control flow for example operations
- Structure of a codeword and breakdown into control lines
- Composition of a program as a sequence of codewords

1st Key Point: Instruction Set Archicture and Datapath

Minutes 5-20
Topic/goal: Revisiting the concept of ISA, IJVM and composing elements
Strategy: Lecture, slides for support
Points to address:

- Explain what is the Instruction Set Architecture (ISA), and how the Integer Java Virtual Machine (IJVM) is a concrete realization of an ISA
- Revisit the purpose of the main elements of a Central Processing Unit (CPU): Arithmetic/Logic Units (ALU), Program Memory, Program Counter Register (PC), Memory Buffer Register (MBR), Data Registers and Buses
- Introduce the concept of datapath

Minutes 20 to 30
Topic/goal: Explanation of datapath; presentation of a concrete use-case
Strategy: Electronic support from visualization tool; involve students by asking about nature or outcome of observed processes
Points to address:

- Point-by-point explanation of an example use-case: adding the contents of two registers
    1. Present initial values contained in registers and operation to be made
    2. Show control line activation and corresponding register read operation
    3. Explain operation at ALU
    4. Describe storage of output value at recipient register

2nd Key Point: Instruction Decomposition Into Commands

Minutes 30 to 45
Topic/goal: Explanation of program, program-related architecture structures and instructions
Strategy: Lecture; slides for support
Points to address:

- Revisit the concept of Program as collection of instructions
- Review program-related structures: Program Memory, PC, MBR
- Introduction to the concept of an instruction as a collection of micro-commands
- Breakdown of a instruction/codeword into IJVM micro-commands: understand which bits in the codeword trigger which commands/operations
- Understand the role of instructions in the datapath / dataflow and CPU operation

Minutes 45 to 55
Topic/goal: Exercise by students to apply the concept of instruction and micro-commands
Strategy: Exercise provided in hand-out sheets; provide individual support to students
Points to achieve:

- Successfully breakdown an instruction into micro-commands
- Successfully understand which data registers and ALU are activated
- For a initial set of values in the data registers and a given instruction, successfully calculate the end result of the instruction

Minutes 55 to 60
Topic/goal: Present exercise result
Strategy: Ask students for their results; provide feedback about their reasoning; use electronic support to validate conclusions and details of solving strategy
Points to address:

- Breakdown of an instruction and identification of triggered command lines
- Present dataflow from input registers towards ALU and back to output register

(10 minute break)

3rd Key Point: Program Execution in the IJVM

Minutes 70 to 85
Topic/goal: Explanation of the program from high level perspective
Strategy: Lecture; slides for support
Points to address:

- Programs as composition of instructions/codewords
- Introducing additional operations of the IJVM and their encoding into instructions: logical operations, shift operations, move data between registers, fetch new codeword
- Presentation of a use case of a multi-instruction program

Minutes 90 to 105
Topic/goal: Exercise by students to apply the concepts of datapath, instruction and program
Strategy: Students can form pairs; exercise provided in hand-out sheets; provide support to students on a group-level
Points to achieve:

- Read an Assembly program and successfully understand its goal in the context of the IJVM
- Successfully identify the used datapath and operations triggered by an instruction
- Successfully calculate the outcome of each instruction
- For a initial set of values in the data registers and a given program, successfully calculate the end result of the program

Minutes 115 to 125
Topic/goal: Present exercise result
Strategy: Ask students for their results; provide feedback about their reasoning; use electronic support to validate conclusions and details of solving strategy
Points to address:

- Summary of actions of each instruction in the program and prediction of expected output
- Identification of triggered operations per instruction and presentation of the values in relevant registers at the end of each instruction
- Presentation of the final output value of the program

Minutes 115 to 120

Summary of themes addressed in class and acquired competencies:

- The Instruction Set Architecture (ISA) and the IJVM as a particular instance
- The IJVM datapath and how it operates for concrete instructions/input values
- The structure of a codeword and how it triggers CPU operations
- To compute the data flow and operations in the CPU for an arbitrary instruction
- The Program and how it is a collection of codewords
- To compute the output of a program for the IJVM

(End of class)

Curso Competências Transversais em Engenharia - Formação Pedagógica
2016/2017 - 1º semestre
3º Exercício
Pedro Miguel Salgueiro dos Santos

Page of Curricular Unit: **Computer Architecture and Operating Systems**

Assessment type

Typology: Final and periodic assessment throughout the semester

Types of assessment: Final, distributed

FEUP/UP Format: Distribuida com exame final

Definition and justification:
Note: italicized segments correspond to *"Pros and Cons"* from slides.

Final assessment:

- This assessment allows the teacher to check if a minimal set of competences were acquired by the students during the course, by *concentrating all the aproached subjects*.
- The exam should include exercises that require combining knowledge and reasoning, to avoid *tendency to the use of questions that only incite memorization*.
- The final assessment is to be complemented with distributed assessment, so it *does not* need to *assess deeply all themes*.

Distributed assessment:

- The distributed assessment provides a chance to review and ask detailed questions about recently taught subjects, contributing to the students' solidification of aquired competences and supporting *evaluation of different topics with more depth along the UC.*
- This assessment also provides the teacher with feedback on how well the students are understanding the subjects, as it *allows each subject/module to be assessed closer to its teaching*.
- If the assessment moments are not in exagerated number and are well planned in advance, neither the students will suffer great stress in their preparation (thus, not *implying a greater investment of time and resources* from the students), nor the teachers will endure considerable burden (i.e. will not *imply a greater investment of time in teaching*).
- Finally, distributed assignment can be oriented for the teacher to evaluate specific competences across the range that the students are supposed to acquire, supporting a *diversification of the assessment techniques and the types of instruments used.*

Information gathering instruments

Name	*Instants (w.r.t. semester duration)*	*Frequency*	*Weight*	*Mode*	*Instruments*	*Mandatory*
Mini-test	40% and 80%	Twice a semester	50% (counts whichever is higher)	Written	Exam with exercises	Yes
Exam	End of semester	Once		Written	Exam with exercises	
Projects	50% and 90%	Twice a semester	50%	Written	Report	Yes

Final grade calculation

Final Grade = 25% x <Project 1 grade + 25% x <Project 2 grade> +
50% x max (50% x <Mini-test 1 grade> + 50%+ <Mini-test 2 grade>, Exam)

Failure to pass

Students that have failed to get the minimum grade can re-take the course in full in the next edition, or re-take only a specific evaluation component:

- *Final:* re-take exam on the semester's second exam phase or on the course's next edition
- *Distributed:* on the course's next edition

Grade Improvement

Students wishing to must improve their grades can do so separately for each evaluation component:

- *Final:* re-take exam on the semester's second exam phase or on the course's next edition
- *Distributed:* on the course's next edition

Working Students

Working students must do one of the following: the two mini-tests or the exam. As with regular students, whichever has a higher grade overrides the other.

Projects can be done off-class and are mandatory.