UNIVERSITÀ
DEGLI STUDI DI MILANO-BICOCCA

COURSE SYLLABUS

Computer Architecture

2627-1-E3102Q102

Aims

By the end of the course, students will have: knowledge of the architecture of a simple computer and the basics of assembly programming; the ability to design small modifications to a computer's internal structure and write simple assembly programs; and the ability to evaluate the most suitable technologies, in terms of performance, for different processing areas.
The reference architecture for the course is MIPS32.

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

- understand and describe the main components of a computer's hardware architecture and their main functions (**Knowledge and Understanding**)
- understand (**Knowledge and Understanding**) and be able to develop (**Applying Knowledge and Understanding**) simple programs using the Assembly programming language
- understand (**Knowledge and Understanding**) and simulate (**Applying Knowledge and Understanding**) the behavior of the various components of the programming chain that brings a program written in a high-level language to its execution
- understand the datapath in the execution of individual elementary instructions and add new instructions to the Instruction Set Architecture (**Applying Knowledge and Understanding**)
- understand how to handle processing exceptions (**Knowledge and Understanding**) and be able to write source code in Assembly to implement it (**Applying Knowledge and Understanding**)
- understand the different ways of managing interaction with I/O devices (**Knowledge and Understanding**) and be able to write Assembly source code to implement them (**Applying knowledge and understanding**)
- understand and be able to simulate the behavior of different types of memory classes (**Applying knowledge and understanding**);

Contents

- Information representation (integer numbers, floating point numbers and text)
- Main components of the hardware architecture of a computer
- Instruction set architecture
- Programming toolchain
- Datapath control
- Exception handling
- I/O techniques
- Memory hierarchies: cache

(MIPS32 is the reference architecture for the course)

Detailed program

1. Information representation in digital computers

- representation of non numeric information
- representation of positive and negative integer numbers
- fixed and floating point representation of numbers

2. Logic circuits

- combinatorial circuits
- sequential circuits and FSMs (Finite State Machines)
- overview of relevant circuits: decoder, multiplexer, register file, ALU, etc.

3. Instruction Set Architecture

- von Neumann architecture
- CPU, registers, ALU and memory
- fundamental cycle of instruction execution (fetch/decode/execute)
- types and formats of MIPS instructions
- addressing modes

4. Assembly language

- Symbolic format of instructions
- Software development toolchain (compiler, assembler, linker, loader, debugger, etc.)
- Pseudo-instructions and assembler directives
- Development of simple assembly programs
- Programming conventions (memory, register names, etc.)

5. Datapath

- Datapath for each type of instruction
- Datapath control with FSM (multi-cycle implementation)

6. Exception handling

- Taxonomy of exceptions in MIPS32
- Cause and EPC registers and modifications to the Control Unit FSM to manage exceptions
- Exception handler

7. Techniques for I/O handling

- Polling (transfers under program control)
- Interrupt
- Direct Memory Access

8. Memory classess : cache

- Direct mapping cache
- Fully associative cache
- N-way set associative cache

Prerequisites

No prerequisites

Teaching form

- 14 frontal lessons of 2 hours each held by the teacher in presence
- 2 frontal lessons of 2 hours each held by the teacher remotely in asynchronous mode
- 10 sessions for exercises of 2 hours each, held by the teacher in presence (50% frontal 50% interactive)
- 8 interactive laboratory lessons of 3 hours each held by the teacher in presence

Textbook and teaching resource

- Textbook: David Patterson, John Hennessy: Computer Organization and Design, The Hardware/Software Interface. Fifth edition. Morgan Kaufmann (Elsevier)
- Teaching material available on the elearning platform concerning lectures (slides), practices (exercices with solutions), and laboratory (Moodle quiz with solutions)

Semester

First semester

Assessment method

There are two ongoing tests, one mid-semester and one at the end of the semester, covering the content taught up to the time of the ongoing test. The ongoing test can be taken only once, and the average of the scores obtained in the two tests will constitute the final grade.
Alternatively, exams are scheduled throughout the academic year covering the entire course content.

Each test (both the ongoing partial test and the comprehensive exam) is a computer session, including a "filter" section with closed-ended questions (2-3 questions for each major topic covered in the course) and a section with open-ended questions (one for each major topic). A grade is obtained only by completing the open-ended section. To access the open-ended questions, students must correctly answer at least one closed-ended question for each major topic. For each closed-ended question, students have the option of not answering (in which case they will be assigned a zero score), while an incorrect answer will result in a penalty being added to their final grade. Open questions are evaluated taking into account both the completeness of the content and the quality of the presentation.

Office hours

By appointment to be agreed via email

Sustainable Development Goals