1. COURSE

CS221. Computer Systems Architecture (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS221. Computer Systems Architecture
2.2 Semester	:	3^{er} Semestre.
2.3 Credits	:	3
2.4 Horas	:	2 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Blended
2.8 Prerrequisites	:	CS1D2. Discrete Structures II. (2^{nd} Sem)

CS1D2. Discrete Structures II. (2^{nd} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

A computer scientist must have a solid knowledge of the organization and design principles of diverse computer systems, by understanding the limitations of modern systems they could propose next-gen paradigms. This course teaches the basics and principles of Computer Architecture. This class addreses digital logic design, basics of Computer Architecture and processor design (Instruction Set architecture, microarchitecture, out-of-order execution, branch prediction), execution paradigms (superscalar, dataflow, VLIW, SIMD, GPUs, systolic, multithreading) and memory system organization.

5. GOALS

- Provide a first approach in Computer Architecture.
- Study the design and evolution of computer architectures, which lead to modern approaches and implementations in computing systems.
- Provide fine-grained details of computer hardware, and its relation with software execution.
- Implement a simple microprocessor using Verilog language.

6. COMPETENCES

1) Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions. (**Usage**)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (**Assessment**)

7. TOPICS

Unit 1: Digital logic and digital systems (18)	
Competences Expected:	
Topics	**Learning Outcomes**
• Overview and history of computer architecture • Combinational and sequential logic/Field programmable gate arrays as a fundamental combinational + sequential logic building block • Abstraction models • Computer-aided design tools that process hardware and architectural representations • Register transfer notation/Hardware Description Language (Verilog/VHDL) • Physical constraints (gate delays, fan-in, fan-out, energy/power)	• Describe the progression of technology devices from vacuum tubes to VLSI, from mainframe computer architectures to the organization of warehouse-scale computers [Familiarity] • Comprehend the trend of modern computer architectures towards multi-core and that parallelism is inherent in all hardware systems [Usage] • Explain the implications of the "power wall" in terms of further processor performance improvements and the drive towards harnessing parallelism [Usage] • Articulate that there are many equivalent representations of computer functionality, including logical expressions and gates, and be able to use mathematical expressions to describe the functions of simple combinational and sequential circuits [Familiarity] • Design the basic building blocks of a computer: arithmetic-logic unit (gate-level), registers (gate-level), central processing unit (register transfer-level), memory (register transfer-level) [Usage] • Use CAD tools for capture, synthesis, and simulation to evaluate simple building blocks (eg, arithmetic-logic unit, registers, movement between registers) of a simple computer design [Familiarity] • Evaluate the functional and timing diagram behavior of a simple processor implemented at the logic circuit level [Assessment]
Readings : [**Harris12**], [**Sanjay05**], [**Patterson2004**], [**Ashenden07**], [HP06], [Par05], [**Stallings2010**], [**Pong06**]	

Unit 2: Machine level representation of data (8)	
Competences Expected:	
Topics	**Learning Outcomes**
• Bits, bytes, and words • Numeric data representation and number bases • Fixed- and floating-point systems • Signed and twos-complement representations • Representation of non-numeric data (character codes, graphical data) • Representation of registers and arrays	• Explain why everything is data, including instructions, in computers [Assessment] • Explain the reasons for using alternative formats to represent numerical data [Familiarity] • Describe how negative integers are stored in sign-magnitude and twos-complement representations [Usage] • Explain how fixed-length number representations affect accuracy and precision [Usage] • Describe the internal representation of non-numeric data, such as characters, strings, records, and arrays [Usage] • Convert numerical data from one format to another [Usage]
Readings : [**Harris12**], [**Sanjay05**], [**Patterson2004**], [**Ashenden07**], [HP06], [Par05], [**Stallings2010**], [**Pong06**]	

Unit 3: Assembly level machine organization (8)	
Competences Expected:	
Topics	**Learning Outcomes**
• Basic organization of the von Neumann machine • Control unit; instruction fetch, decode, and execution • Instruction sets and types (data manipulation, control, I/O) • Assembly/machine language programming • Instruction formats • Addressing modes • Subroutine call and return mechanisms • I/O and interrupts • Heap vs. Static vs. Stack vs. Code segments	• Explain the organization of the classical von Neumann machine and its major functional units [Familiarity] • Describe how an instruction is executed in a classical von Neumann machine, with extensions for threads, multiprocessor synchronization, and SIMD execution [Familiarity] • Describe instruction level parallelism and hazards, and how they are managed in typical processor pipelines [Familiarity] • Summarize how instructions are represented at both the machine level and in the context of a symbolic assembler [Familiarity] • Demonstrate how to map between high-level language patterns into assembly/machine language notations [Usage] • Explain different instruction formats, such as addresses per instruction and variable length vs fixed length formats [Usage] • Explain how subroutine calls are handled at the assembly level [Usage] • Explain the basic concepts of interrupts and I/O operations [Familiarity] • Write simple assembly language program segments [Usage] • Show how fundamental high-level programming constructs are implemented at the machine-language level [Usage]
Readings : [**Harris12**], [**Sanjay05**], [**Patterson2004**], [**Ashenden07**], [HP06], [Par05], [**Stallings2010**], [**Pong06**]	

Unit 4: Functional organization (8)	
Competences Expected:	
Topics	**Learning Outcomes**
• Implementation of simple datapaths, including instruction pipelining, hazard detection and resolution • Control unit: microprogrammed • Instruction pipelining • Introduction to instruction-level parallelism (ILP)	• Compare alternative implementation of datapaths [Assessment] • Discuss the concept of control points and the generation of control signals using hardwired or microprogrammed implementations [Familiarity] • Explain basic instruction level parallelism using pipelining and the major hazards that may occur [Usage] • Design and implement a complete processor, including datapath and control [Usage] • Determine, for a given processor and memory system implementation, the average cycles per instruction [Assessment]
Readings : [**Harris12**], [**Sanjay05**], [**Patterson2004**], [**Ashenden07**], [HP06], [Par05], [**Stallings2010**], [**Pong06**]	

Unit 5: Memory system organization and architecture (8)	
Competences Expected:	
Topics	**Learning Outcomes**
• Storage systems and their technology • Memory hierarchy: importance of temporal and spatial locality • Main memory organization and operations • Latency, cycle time, bandwidth, and interleaving • Cache memories (address mapping, block size, replacement and store policy) • Multiprocessor cache consistency/Using the memory system for inter-core synchronization/atomic memory operations • Virtual memory (page table, TLB) • Fault handling and reliability • Error coding, data compression, and data integrity	• Identify the main types of memory technology (eg, SRAM, DRAM, Flash, magnetic disk) and their relative cost and performance [Familiarity] • Explain the effect of memory latency on running time [Familiarity] • Describe how the use of memory hierarchy (cache, virtual memory) is used to reduce the effective memory latency [Usage] • Describe the principles of memory management [Usage] • Explain the workings of a system with virtual memory management [Usage] • Compute Average Memory Access Time under a variety of cache and memory configurations and mixes of instruction and data references [Assessment]
Readings : [**Harris12**], [**Sanjay05**], [**Patterson2004**], [**Ashenden07**], [HP06], [Par05], [**Stallings2010**], [**Pong06**]	

Unit 6: Interfacing and communication (8)	
Competences Expected:	
Topics	**Learning Outcomes**
• I/O fundamentals: handshaking, buffering, programmed I/O, interrupt-driven I/O • Interrupt structures: vectored and prioritized, interrupt acknowledgment • External storage, physical organization, and drives • Buses: bus protocols, arbitration, direct-memory access (DMA) • Introduction to networks: communications networks as another layer of remote access • Multimedia support • RAID architectures	• Explain how interrupts are used to implement I/O control and data transfers [Familiarity] • Identify various types of buses in a computer system [Familiarity] • Describe data access from a magnetic disk drive [Usage] • Compare common network organizations, such as ethernet/bus, ring, switched vs routed [Assessment] • Identify the cross-layer interfaces needed for multimedia access and presentation, from image fetch from remote storage, through transport over a communications network, to staging into local memory, and final presentation to a graphical display [Familiarity] • Describe the advantages and limitations of RAID architectures [Familiarity]
Readings : [**Harris12**], [**Sanjay05**], [**Patterson2004**], [**Ashenden07**], [HP06], [Par05], [**Stallings2010**], [**Pong06**]	

Unit 7: Multiprocessing and alternative architectures (8)	
Competences Expected:	
Topics	**Learning Outcomes**
• Power Law • Example SIMD and MIMD instruction sets and architectures • Interconnection networks (hypercube, shuffle-exchange, mesh, crossbar) • Shared multiprocessor memory systems and memory consistency • Multiprocessor cache coherence	• Discuss the concept of parallel processing beyond the classical von Neumann model [Assessment] • Describe alternative parallel architectures such as SIMD and MIMD [Familiarity] • Explain the concept of interconnection networks and characterize different approaches [Usage] • Discuss the special concerns that multiprocessing systems present with respect to memory management and describe how these are addressed [Familiarity] • Describe the differences between memory backplane, processor memory interconnect, and remote memory via networks, their implications for access latency and impact on program performance [Assessment]
Readings : [**Harris12**], [**Sanjay05**], [**Patterson2004**], [**Ashenden07**], [HP06], [Par05], [**Stallings2010**], [**Pong06**]	

Unit 8: Performance enhancements (8)	
Competences Expected:	
Topics	**Learning Outcomes**
• Superscalar architecture • Branch prediction, Speculative execution, Out-of-order execution • Prefetching • Vector processors and GPUs • Hardware support for multithreading • Scalability • Alternative architectures, such as VLIW/EPIC, and Accelerators and other kinds of Special-Purpose Processors	• Describe superscalar architectures and their advantages [Familiarity] • Explain the concept of branch prediction and its utility [Usage] • Characterize the costs and benefits of prefetching [Assessment] • Explain speculative execution and identify the conditions that justify it [Assessment] • Discuss the performance advantages that multithreading offered in an architecture along with the factors that make it difficult to derive maximum benefits from this approach [Assessment] • Describe the relevance of scalability to performance [Assessment]
Readings : [**Harris12**], [**Sanjay05**], [**Patterson2004**], [**Ashenden07**], [HP06], [Par05], [**Stallings2010**], [**Pong06**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[HP06] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. 4th. San Mateo, CA: Morgan Kaufman, 2006.

[Par05] Behrooz Parhami. *Computer Architecture: From Microprocessors to Supercomputers*. New York: Oxford Univ. Press, 2005. ISBN: ISBN 0-19-515455-X.