**National University of Engineering (UNI)**
School of Artificial Intelligence
Syllabus 2027-I

## 1. COURSE

CS221. Computer Systems Architecture (Mandatory)

## 2. GENERAL INFORMATION

| | | |
|---|---|---|
| **2.1 Course** | : | CS221. Computer Systems Architecture |
| **2.2 Semester** | : | $4^{th}$ Semester. |
| **2.3 Credits** | : | 4 |
| **2.4 Hours** | : | 2 HT; 4 HP; |
| **2.5 Duration of the period** | : | 16 weeks |
| **2.6 Type of course** | : | Mandatory |
| **2.7 Learning modality** | : | Face to face |
| **2.8 Prerrequisites** | : | CS1D1. Discrete Structures. ($2^{nd}$ Sem) |

## 3. PROFESSORS

Meetings after coordination with the professor

## 4. INTRODUCTION TO THE COURSE

A deep understanding of computer architecture is essential for developing high-performance software and understanding the hardware-software interface. This course explores how computers execute instructions, manage memory, and communicate with external devices. By studying the internal organization of processors and the hierarchy of memory systems, students gain the ability to optimize code for specific hardware architectures and understand the trade-offs in modern system design.

## 5. GOALS

- Understand the principles of digital logic and data representation.
- Understand the organization of the Central Processing Unit (CPU).
- Master the basics of assembly language and instruction set architecture.
- Analyze the memory hierarchy, including cache memory and virtual memory.
- Evaluate performance improvements through pipelining and parallelism.

## 6. COMPETENCES

**AG-C03)** Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Usage)

**AG-C09)** Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

## 7. TOPICS

| **Unit 1: Assembly Level Machine Organization (20 hours)** | |
|---|---|
| **Competences Expected: AG-C03,AG-C09** | |
| **Topics** | **Learning Outcomes** |
| • von Neumann machine architecture<br>• Control unit: instruction fetch, decode, and execution<br>• Introduction to SIMD vs MIMD and the Flynn taxonomy<br>• Shared memory multiprocessors/multicore organization<br>• Instruction set architecture (ISA) (e.g., x86, ARM and RISC-V):<br>  – Fixed vs variable-width instruction sets<br>  – Instruction formats<br>  – Data manipulation, control, I/O<br>  – Addressing modes<br>  – Machine language programming<br>  – Assembly language programming<br>• Subroutine call and return mechanisms<br>• I/O and interrupts<br>• Heap, static, stack, and code segments | • Discuss how the classical von Neumann functional units are implemented in embedded systems, particularly on-chip and off-chip memory [Debate]<br>• Describe how instructions are executed in a classical von Neumann machine, with extensions for threads, multiprocessor synchronization, and SIMD execution [Describe]<br>• Assess an example diagram with instruction-level parallelism and hazards to describe how they are managed in typical processor pipelines [Assess]<br>• Discuss how instructions are represented at the machine level and in the context of a symbolic assembler [Debate]<br>• Map an example of high-level language patterns into assembly/machine language notations [Map]<br>• Contrast different instruction formats considering aspects such as addresses per instruction and variable-length vs fixed-length formats [Contrast]<br>• Analyze a subroutine diagram to comment on how subroutine calls are handled at the assembly level [Analyze]<br>• Describe basic concepts of interrupts and I/O operations [Describe]<br>• Write a simple assembly language program for string/array processing and manipulation [Write] |
| **Readings :** [**Patterson2020**], [**Harris2012**] | |

| **Unit 2: Memory Hierarchy (18 hours)** | |
|---|---|
| **Competences Expected: AG-C03,AG-C09** | |
| **Topics** | **Learning Outcomes** |
| • Memory hierarchy: the importance of temporal and spatial locality<br>• Main memory organization and operations<br>• Persistent memory (e.g., SSD, standard disks)<br>• Latency, cycle time, bandwidth, and interleaving<br>• Cache memories:<br>– Address mapping<br>– Block size<br>– Replacement and store policy<br>– Prefetching<br>• Multiprocessor cache coherence<br>• Virtual memory (hardware support)<br>• Fault handling and reliability<br>• Reliability:<br>– Error coding<br>– Data compression<br>– Data integrity<br>• Processing In-Memory (PIM) | • Using a memory system diagram, identify the main types of memory technology (e.g., SRAM, DRAM) and their relative cost and performance [Analyze]<br>• Measure the effect of memory latency on running time [Evaluate]<br>• Enumerate the functions of a system with virtual memory management [Enumerate]<br>• Compute average memory access time under various cache and memory configurations and mixes of instruction and data references [Compute] |
| **Readings :** [**Patterson2020**], [**Stallings2015**] | |

| **Unit 3: Interfacing and Communication (8 hours)** | |
|---|---|
| **Competences Expected: AG-C03,AG-C09** | |
| **Topics** | **Learning Outcomes** |
| • I/O fundamentals:<br>– Handshaking and buffering<br>– Programmed I/O<br>– Interrupt-driven I/O<br>• Interrupt structures: vectored and prioritized, interrupt acknowledgment<br>• I/O devices (e.g., mouse, keyboard, display, camera, sensors, actuators)<br>• External storage, physical organization, and drives<br>• Buses fundamentals:<br>– Bus protocols<br>– Arbitration<br>– Direct-memory access (DMA) | • Analyze an interrupt control diagram to comment on how interrupts are used to implement I/O control and data transfers [Analyze]<br>• Enumerate various types of buses in a computer system [Enumerate]<br>• List the advantages of magnetic disks and contrast them with those of solid-state disks [List] |
| **Readings :** [**Patterson2020**], [**Tanenbaum2012**] | |

| **Unit 4: Functional Organization (8 hours)** | |
|---|---|
| **Competences Expected: AG-C03,AG-C09** | |
| **Topics** | **Learning Outcomes** |
| • Implementation of simple datapaths, including instruction pipelining, hazard detection, and resolution (e.g., stalls, forwarding)<br>• Control unit:<br>– Hardwired implementation<br>– Microprogrammed realization<br>• Instruction pipelining<br>• Introduction to instruction-level parallelism (ILP) | • Compare alternative implementation of datapaths in modern computer architectures [Compare]<br>• Produce a set of control signals for adding two integers using hardwired and microprogrammed implementations [Produce]<br>• Discuss instruction-level parallelism using pipelining and significant hazards that may occur [Debate]<br>• Design a complete processor, including datapath and control [Design]<br>• Compute the average cycles per instruction for a given processor and memory system implementation [Compute] |
| **Readings :** [**Harris2012**], [**Patterson2020**] | |

| **Unit 5: Heterogeneous Architectures (10 hours)** | |
|---|---|
| **Competences Expected: AG-C03,AG-C09** | |
| **Topics** | **Learning Outcomes** |
| • SIMD and MIMD architectures (e.g., General-Purpose GPUs, TPUs, and NPUs)<br>• Heterogeneous memory systems:<br>– Shared memory versus distributed memory<br>– Volatile vs non-volatile memory<br>– Coherence protocols<br>• Domain-Specific Architectures (DSAs):<br>– Machine Learning Accelerator<br>– In-networking computing<br>– Embedded systems for emerging applications<br>– Neuromorphic computing<br>– Edge computing devices<br>• Packaging and integration solutions such as 3DIC and chiplets<br>• Machine learning in architecture design:<br>– AI algorithms for workload analysis<br>– Optimization of architecture configurations for performance and power efficiency<br>• Data-Oriented Design (DOD): organizing data in memory (array-of-structures vs. structure-of-arrays) to maximize cache locality and exploit SIMD vectorization; contrast and complementarity with the object-oriented paradigm. | • Analyze a system diagram with alternative parallel architectures, e.g., SIMD and MIMD, and identify the key differences [Evaluar]<br>• Discuss what memory-management issues are found in multiprocessors that are not present in uniprocessors and how these issues might be resolved [Evaluar]<br>• Indicate the differences between memory backplane, processor memory interconnect, and remote memory via networks, their implications for access latency, and their impact on program performance [Familiarizarse]<br>• Discuss how you would determine when to use a domain-specific accelerator instead of a general-purpose CPU [Evaluar]<br>• Enumerate key differences in architectural design principles between a vector and scalar-based processing unit [Familiarizarse]<br>• List the advantages and disadvantages of a PIM architecture [Familiarizarse]<br>• Restructure a collection of objects (AoS) into a DOD layout (SoA) and measure the performance impact [Usar]. |
| **Readings :** [**Patterson2020**], [**Stallings2015**] | |

| Unit 6: Secure Processor Architectures (8 hours) | |
|---|---|
| **Competences Expected: AG-C03,AG-C09** | |
| **Topics** | **Learning Outcomes** |
| • Principles of Secure Hardware:<br>– Security Risk Analysis, Asset Protection, and Threat Model<br>– Cryptographic Acceleration with Hardware<br>– Support for virtualization (e.g., OS isolation)<br>• Roots of trust in hardware, Physically Unclonable Functions (PUF)<br>• Hardware Random Number Generators<br>• Memory protection extensions:<br>– Runtime pointer bounds checking (e.g., buffer overflow)<br>– Protection at the microarchitectural level<br>– Protection at the ISA level<br>• Trusted Execution Environment (TEE):<br>– Trusted Computer Base Protections<br>– Protecting virtual machines<br>– Protecting containers<br>– Trusted software modules (Enclaves)<br>• Homomorphic encryption for privacy-preserving data processing | • Discuss principles of secure hardware, exploring a framework for risk analysis and asset protection [Debate]<br>• Summarize how Physically Unclonable Functions (PUF) can be a unique device identifier in security applications [Summarize]<br>• Distinguish a random number generator with dedicated hardware support from generators without hardware dedicated to generating entropy [Distinguish]<br>• List the advantages and disadvantages of memory protection at the ISA level [List]<br>• Describe key design issues of a trusted execution environment (TEE) to support virtual machines [Describe] |
| **Readings :** [**Patterson2020**], [**Anderson2020**] | |

## 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