National University of Engineering (UNI)
School of Computer Science
Syllabus 2026-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-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

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

7. TOPICS

Unit 1: Lógica Digital y Sistemas Digitales (14 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Lógica combinacional vs secuencial/arreglos de puertas programables en campo (FPGAs): – Lógica combinacional fundamental – Bloque de construcción de lógica secuencial • Herramientas de diseño asistido por computadora que procesan representaciones de hardware y arquitectura • Síntesis de alto nivel: – Notación de transferencia de registros – Lenguaje de descripción de hardware (por ejemplo, Verilog/VHDL/Chisel) • Flujo de diseño de sistema en chip (SoC) • Restricciones físicas: – Retardos de puertas – Fan-in y fan-out – Energía/potencia – Velocidad de la luz	• Discutir la progresión de los componentes de tecnología informática desde tubos de vacío hasta VLSI, desde arquitecturas de computadoras mainframe hasta la organización de computadoras a escala de almacén [Debatir] • Describir el paralelismo y las dependencias de datos entre y dentro de componentes en una arquitectura de computadora heterogénea moderna [Describir] • Explicar la relación entre paralelismo y consumo de energía [Explicar] • Construir el diseño de bloques básicos para una computadora: unidad aritmético-lógica (nivel de puertas), registros (nivel de puertas), unidad central de procesamiento (nivel de transferencia de registros) y memoria (nivel de transferencia de registros) [Crear] • Evaluar bloques básicos simples (por ejemplo, unidad aritmético-lógica, registros, movimiento entre registros) de un diseño de computadora simple [Evaluar] • Analizar el comportamiento de temporización de un procesador segmentado, identificando problemas de dependencia de datos [Analizar]
Readings : [PH20], [HH12]	

Unit 2: Representación de Datos a Nivel de Máquina (12 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Visión general e historia de la arquitectura de computadoras • Bits, bytes y palabras • Representaciones sin signo, con signo y complemento a dos • Representación de datos numéricos y bases numéricas: – Punto fijo – Punto flotante • Representación de datos no numéricos • Representación de registros, arreglos y tipos de datos UTF	• Discutir por qué todo en las computadoras son datos, incluyendo instrucciones [Debatir] • Explicar cómo las representaciones numéricas de longitud fija pueden afectar la exactitud y precisión [Explicar] • Describir cómo se almacenan los enteros negativos en representaciones de magnitud con signo y complemento a dos [Describir] • Discutir cómo diferentes formatos pueden representar datos numéricos [Debatir] • Explicar la representación a nivel de bits de datos no numéricos, como caracteres, cadenas, registros y arreglos [Explicar] • Traducir datos numéricos de un formato a otro [Traducir] • Describir cómo un sumador único (sin detección de desbordamiento) puede manejar tanto entrada con signo (complemento a dos) como sin signo (binario) sin ”saber” qué formato está usando una entrada dada [Describir]
Readings : [PH20], [Sta15]	

Unit 3: Organización de Máquina a Nivel de Ensamblador (20 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Arquitectura de máquina von Neumann • Unidad de control: captación, decodificación y ejecución de instrucciones • Introducción a SIMD vs MIMD y la taxonomía de Flynn • Organización de multiprocesadores/multinúcleo de memoria compartida • Arquitectura del conjunto de instrucciones (ISA) (por ejemplo, x86, ARM y RISC-V): – Conjuntos de instrucciones de ancho fijo vs variable – Formatos de instrucción – Manipulación de datos, control, E/S – Modos de direccionamiento – Programación en lenguaje de máquina – Programación en lenguaje ensamblador • Mecanismos de llamada y retorno de subrutinas • E/S e interrupciones • Segmentos de montículo, estáticos, pila y código	• Discutir cómo las unidades funcionales clásicas de von Neumann se implementan en sistemas embebidos, particularmente en memoria dentro y fuera del chip [Debatir] • Describir cómo se ejecutan las instrucciones en una máquina von Neumann clásica, con extensiones para hilos, sincronización de multiprocesadores y ejecución SIMD [Describir] • Evaluar un diagrama de ejemplo con paralelismo a nivel de instrucción y riesgos para describir cómo se manejan en las canalizaciones de procesadores típicos [Evaluar (valorar)] • Discutir cómo se representan las instrucciones a nivel de máquina y en el contexto de un ensamblador simbólico [Debatir] • Mapear un ejemplo de patrones de lenguaje de alto nivel a notaciones de lenguaje ensamblador/máquina [Mapear/Mapa conceptual] • Contrastar diferentes formatos de instrucción considerando aspectos como direcciones por instrucción y formatos de longitud variable vs longitud fija [Contrastar] • Analizar un diagrama de subrutina para comentar cómo se manejan las llamadas de subrutina a nivel de ensamblador [Analizar] • Describir conceptos básicos de interrupciones y operaciones de E/S [Describir] • Escribir un programa simple en lenguaje ensamblador para procesamiento y manipulación de cadenas/arreglos [Escribir]
Readings : [PH20], [HH12]	

Unit 4: Jerarquía de Memoria (18 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Jerarquía de memoria: la importancia de la localidad temporal y espacial • Organización y operaciones de memoria principal • Memoria persistente (por ejemplo, SSD, discos estándar) • Latencia, tiempo de ciclo, ancho de banda e intercalado • Memorias caché: – Mapeo de direcciones – Tamaño de bloque – Política de reemplazo y almacenamiento – Prebúsqueda • Coherencia de caché multiprocesador • Memoria virtual (soporte de hardware) • Manejo de fallas y confiabilidad • Confiabilidad: – Codificación de error – Compresión de datos – Integridad de datos • Procesamiento en Memoria (PIM)	• Usando un diagrama del sistema de memoria, identificar los principales tipos de tecnología de memoria (por ejemplo, SRAM, DRAM) y su costo y rendimiento relativos [Analizar] • Medir el efecto de la latencia de memoria en el tiempo de ejecución [Evaluar] • Enumerar las funciones de un sistema con gestión de memoria virtual [Enumerar] • Calcular el tiempo promedio de acceso a memoria bajo varias configuraciones de caché y memoria y mezclas de referencias de instrucciones y datos [Computar/Calcular]
Readings : [PH20], [Sta15]	

Unit 5: Arquitecturas Heterogéneas (10 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Arquitecturas SIMD y MIMD (por ejemplo, GPUs de Propósito General, TPUs y NPUs) • Sistemas de memoria heterogéneos: – Memoria compartida versus memoria distribuida – Memoria volátil vs no volátil – Protocolos de coherencia • Arquitecturas Específicas de Dominio (DSAs): – Acelerador de Aprendizaje Automático – Computación en red – Sistemas embebidos para aplicaciones emergentes – Computación neuromórfica – Dispositivos de computación perimetral • Soluciones de empaquetado e integración como 3DIC y chiplets • Aprendizaje automático en diseño de arquitectura: – Algoritmos de IA para análisis de carga de trabajo – Optimización de configuraciones de arquitectura para rendimiento y eficiencia energética	• Analizar un diagrama de sistema con arquitecturas paralelas alternativas, por ejemplo, SIMD y MIMD, e identificar las diferencias clave [Analizar] • Discutir qué problemas de gestión de memoria se encuentran en multiprocesadores que no están presentes en uniprocesadores y cómo estos problemas podrían resolverse [Debatir] • Indicar las diferencias entre placa base de memoria, interconexión de memoria del procesador y memoria remota a través de redes, sus implicaciones para la latencia de acceso y su impacto en el rendimiento del programa [Diferenciar] • Discutir cómo determinarías cuándo usar un acelerador específico de dominio en lugar de una CPU de propósito general [Debatir] • Enumerar diferencias clave en los principios de diseño arquitectónico entre una unidad de procesamiento basada en vector y basada en escalar [Enumerar] • Enumerar las ventajas y desventajas de una arquitectura PIM [Listar/Enumerar]
Readings : [PH20], [HH12]	

Unit 6: Arquitecturas de Procesador Seguras (8 hours)	
Competences Expected: AG-C03,AG-C09	
Topics	**Learning Outcomes**
• Principios de Hardware Seguro: – Análisis de Riesgos de Seguridad, Protección de Activos y Modelo de Amenazas – Aceleración Criptográfica con Hardware – Soporte para virtualización (por ejemplo, aislamiento de SO) • Raíces de confianza en hardware, Funciones Físicamente No Clonables (PUF) • Generadores de Números Aleatorios por Hardware • Extensiones de protección de memoria: – Verificación de límites de puntero en tiempo de ejecución (por ejemplo, desbordamiento de búfer) – Protección a nivel microarquitectónico – Protección a nivel de ISA • Entorno de Ejecución Confiable (TEE): – Protecciones de Base de Computadora Confiable – Protección de máquinas virtuales – Protección de contenedores – Módulos de software confiables (Enclaves) • Cifrado homomórfico para procesamiento de datos que preserva la privacidad	• Discutir principios de hardware seguro, explorando un marco para análisis de riesgos y protección de activos [Debatir] • Resumir cómo las Funciones Físicamente No Clonables (PUF) pueden ser un identificador único de dispositivo en aplicaciones de seguridad [Resumir] • Distinguir un generador de números aleatorios con soporte de hardware dedicado de generadores sin hardware dedicado a generar entropía [Distinguir] • Enumerar las ventajas y desventajas de la protección de memoria a nivel de ISA [Listar/Enumerar] • Describir problemas de diseño clave de un entorno de ejecución confiable (TEE) para soportar máquinas virtuales [Describir]
Readings : [PH20], [Sta15]	

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

[HH12] David Harris and Sarah Harris. *Digital Design and Computer Architecture*. 2nd. Morgan Kaufmann, 2012.

[Sta15] William Stallings. *Computer Organization and Architecture: Designing for Performance*. 10th. Pearson, 2015.

[PH20] David A. Patterson and John L. Hennessy. *Computer Organization and Design RISC-V Edition: The Hardware Software Interface*. 2nd. Morgan Kaufmann, 2020.