Universidad Nacional de Ingeniería (UNI)
Escuela Profesional de
Ciencia de la Computación
Sílabo 2026-I

1. CURSO

CS221. Arquitectura de Computadores (Obligatorio)

2. INFORMACIÓN GENERAL

2.1 Curso	:	CS221. Arquitectura de Computadores
2.2 Semestre	:	4^{to} Semestre.
2.3 Créditos	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duración del periodo	:	16 semanas
2.6 Condición	:	Obligatorio
2.7 Modalidad de aprendizaje	:	Presencial
2.8 Prerrequisitos	:	CS1D1. Estructuras Discretas. (2^{do} Sem)

3. PROFESORES

Atención previa coordinación con el profesor

4. INTRODUCCIÓN AL CURSO

Una comprensión profunda de la arquitectura de computadoras es esencial para desarrollar software de alto rendimiento y entender la interfaz hardware-software. Este curso explora cómo las computadoras ejecutan instrucciones, gestionan memoria y se comunican con dispositivos externos. Al estudiar la organización interna de los procesadores y la jerarquía de los sistemas de memoria, los estudiantes adquieren la capacidad de optimizar código para arquitecturas de hardware específicas y comprender las compensaciones en el diseño de sistemas modernos.

5. OBJETIVOS

- Comprender los principios de la lógica digital y la representación de datos.
- Comprender la organización de la Unidad Central de Procesamiento (CPU).
- Dominar los conceptos básicos del lenguaje ensamblador y la arquitectura del conjunto de instrucciones.
- Analizar la jerarquía de memoria, incluyendo memoria caché y memoria virtual.
- Evaluar mejoras de rendimiento mediante segmentación y paralelismo.

6. RESULTADOS DEL ESTUDIANTE

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Usage)

AG-C03) Trabajo Individual y en Equipo: Se desempeña efectivamente como individuo y como miembro o líder en equipos diversos. (Usage)

7. TEMAS

Unidad 1: Lógica Digital y Sistemas Digitales (14 horas)	
Resultados esperados: AG-C03,AG-C09	
Temas	**Aprendizaje esperado (*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]
Lecturas : [PH20], [HH12]	

Unidad 2: Representación de Datos a Nivel de Máquina (12 horas)	
Resultados esperados: AG-C03,AG-C09	
Temas	**Aprendizaje esperado (*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]
Lecturas : [PH20], [Sta15]	

Unidad 3: Organización de Máquina a Nivel de Ensamblador (20 horas)	
Resultados esperados: AG-C03,AG-C09	
Temas	**Aprendizaje esperado (*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]
Lecturas : [PH20], [HH12]	

Unidad 4: Jerarquía de Memoria (18 horas)	
Resultados esperados: AG-C03,AG-C09	
Temas	**Aprendizaje esperado (*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]
Lecturas : [PH20], [Sta15]	

Unidad 5: Arquitecturas Heterogéneas (10 horas)	
Resultados esperados: AG-C03,AG-C09	
Temas	**Aprendizaje esperado (*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]
Lecturas : [PH20], [HH12]	

Unidad 6: Arquitecturas de Procesador Seguras (8 horas)	
Resultados esperados: AG-C03,AG-C09	
Temas	**Aprendizaje esperado (*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]
Lecturas : [PH20], [Sta15]	

8. PLAN DE TRABAJO

8.1 Metodología

Se fomenta la participación individual y en equipo para exponer sus ideas, motivándolos con puntos adicionales en las diferentes etapas de la evaluación del curso.

8.2 Sesiones Teóricas

Las sesiones de teoría se llevan a cabo en clases magistrales donde se realizarán actividades que propicien un aprendizaje activo, con dinámicas que permitan a los estudiantes interiorizar los conceptos.

8.3 Sesiones Prácticas

Las sesiones prácticas se llevan en clase donde se desarrollan una serie de ejercicios y/o conceptos prácticos mediante planteamiento de problemas, la resolución de problemas, ejercicios puntuales y/o en contextos aplicativos.

9. SISTEMA DE EVALUACIÓN

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

10. BIBLIOGRAFÍA BÁSICA

[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.