FACULTAD DE INGENIERIA

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

1. NOMBRE DE LA UNIDAD CURRICULAR

Arquitectura de Computadoras

2. CRÉDITOS

10 créditos.

3. OBJETIVOS DE LA UNIDAD CURRICULAR

El objetivo de esta unidad curricular es presentar los conceptos fundamentales de la arquitectura de una computadora, tanto a nivel de los conceptos sobre la que se construye como su aplicación en una arquitectura real de ejemplo. Específicamente se busca que el estudiante identifique y domine los conceptos y principios fundamentales que permiten que los programas sean ejecutados en máquinas físicas, así como sea capaz de modelar el control de procesos físicos mediante computadoras..

A nivel de los objetivos de aprendizaje, el estudiante será capaz de:

Sistemas de numeración [Uso]:

- Realizar conversiones de números entre diferentes bases, aplicando los algoritmos de divisiones/multiplicaciones sucesivas y polinomio característico. [Aplicar]
- Utilizar el método de expansión de dígitos para realizar conversiones entre bases B^K y B. [Aplicar]
- Realizar operaciones aritméticas con números en diferentes bases (en particular en base 2 y 16). [Aplicar]

Códigos y errores [Familiaridad]:

- Explicar el concepto de distancia entre códigos y de un sistema de codificación binario. [Comprender]

- Describir las características de los sistemas de codificación con capacidad de detección y corrección de errores. [Recordar]
- Identificar cuándo un sistema de codificación binario corrige y/o detecta errores, y calcular cuántos bits de error puede detectar o corregir. [Analizar]

Representación interna de datos [Evaluación]:

- Describir las características de los sistemas de representación de caracteres (ASCII y Unicode). [Recordar]
- Indicar algoritmos de codificación y características fundamentales (rango de representación, unicidad del cero, conservación del orden), de los siguientes sistemas de representación de enteros: BCD, valor absoluto y signo, desplazamiento, complemento a uno y complemento a dos. [Comprender]
- Realizar operaciones aritméticas con números representados en los distintos sistemas de codificación de enteros. [Aplicar]
- Describir los algoritmos de codificación para las representaciones de punto fijo, y punto flotante IEEE 754. [Recordar]
- Calcular el error cometido al representar un número en punto fijo o punto flotante. [Aplicar]
- Realizar sumas, restas, multiplicaciones y divisiones en punto flotante. [Aplicar]
- Realizar conversiones entre diferentes representaciones de enteros y racionales. [Aplicar]

Álgebra de Boole y Circuitos Combinatorios [Familiaridad]:

- Indicar las tablas de verdad de las conectivas booleanas elementales (not, and, or, xor, nand, nor, xnor) y expresar sus tablas de verdad. [Recordar]
- Calcular la expresión booleana de un circuito lógico combinatorio en función de sus entradas [Aplicar]
- Describir el método de Karnaugh y fundamentar por qué logra hallar un mínimo en dos niveles. [Comprender]
- Aplicar el método de Karnaugh para minimizar funciones booleanas en dos niveles. [Aplicar]
- Escribir programas en C que manipulen la representación binaria de variables, utilizando operadores binarios bit a bit. [Crear]
- Describir circuitos combinatorios paradigmáticos tales como decodificador, multiplexor y demultiplexor. [Recordar]
- Definir memoria ROM y describir su organización interna. [Recordar].
- Explicar la utilidad de las entradas chip select (CS) y output enable (OE) en memorias ROM [Comprender]
- Calcular tamaño y organización de una ROM apropiada para guardar la tabla de verdad de una función lógica dada. [Analizar]

- Escribir programas para inicializar el contenido de una memoria ROM dada la función lógica que resuelve [Crear]

Circuitos secuenciales [Familiaridad]
Máquinas de estados [Evaluación
Máquina lógica general [Familiaridad]:

- Reconocer la diferencia entre circuitos combinatorios y secuenciales. [Recordar]
- Describir circuitos secuenciales paradigmáticos tales como flip-flops, memoria y, contadores. [Comprender]
- Modelar un problema a través de una Máquina de Estados. [Crear]
- Explicar el concepto de Máquina Lógica General. [Recordar]

Arquitectura Von Neumann [Uso] y Organización de CPU [Familiaridad]

- Detallar los bloques constructivos de la Arquitectura Von Neumann. [Recordar]
- Describir los elementos que caracterizan a una implementación particular de Arquitectura Von Neumann. [Recordar]
- Explicar la diferencia entre Arquitectura y Organización de una computadora. Identificar qué elementos y características de una CPU son definidos a nivel de Arquitectura y cuáles a nivel de Organización. [Comprender]
- Definir RISC y CISC y explicar las diferencias fundamentales entre ambas filosofías de diseño. [Comprender]
- Dado un set de instrucciones de una arquitectura de computadoras RISC o CISC, diseñar un formato de instrucción apropiado. [Aplicar]
- Escribir programas en Assembler simples en un set de instrucciones dado [Crear]
- Identificar y definir los subsistemas fundamentales de una CPU. [Recordar]
- Describir el ciclo de instrucción clásico de cinco etapas (Fetch, Decode, Read Execute, Write). [Recordar]
- Describir los modos de direccionamiento típicos. [Recordar]
- Definir registro visible, invisible y parcialmente visible. Identificar ejemplos de cada uno dentro de una arquitectura de ejemplo. [Comprender]

E/S [Uso] e Interrupciones [Evaluación]

- Identificar y describir los diferentes componentes de hardware que intervienen en la interacción con un dispositivo de entrada/salida. [Recordar]
- Escribir programas en lenguaje C que interactúen con dispositivos de E/S a través de las sentencias IN y OUT. [Crear]
- Describir y comparar los distintos mecanismos de identificación de interrupciones. [Comprender]
- Explicar las diferencias entre las técnicas de interacción con la E/S de polling e interrupciones. [Analizar]

- Programar rutinas de atención a la interrupción e integrarlas de manera de resolver problemas que impliquen dispositivos de E/S e interacciones con el mundo físico. [Crear]

Arquitectura de estudio [Uso]

- Describir las características principales del lenguaje ensamblador de una arquitectura tomada como ejemplo. [Recordar]
- Describir el mecanismo de atención a la interrupción en la arquitectura presentada en el curso. [Recordar]
- Leer e interpretar la cartilla de instrucciones de la arquitectura presentada en el curso [Comprender]
- Reconocer las diferencias más importantes entre la programación de bajo nivel y la programación de alto nivel. [Comprender]
- Compilar programas a lenguaje ensamblador de la arquitectura presentada en el curso, incluyendo el manejo de la entrada/salida y las interrupciones. [Aplicar]
- Calcular el consumo de stack de un programa recursivo en lenguaje ensamblador [Analizar]

Jerarquía de memoria [Uso]

- Explicar el concepto de jerarquía de memoria. [Recordar]
- Diferenciar los diferentes niveles de memoria disponibles en una jerarquía de memoria estándar de una computadora moderna. Reconocer las diferencias en precio por bit de almacenamiento y velocidad de acceso, entre los diferentes niveles. [Comprender]
- Definir los principios de localidad espacial y temporal y explicar cómo se explotan en la jerarquía de memoria. [Comprender]
- Calcular hit, miss, hit rate, miss rate, hit time, miss penalty y tiempo promedio de acceso a memoria en la ejecución de programas simples. [Aplicar]
- Describir los diferentes tipos de funciones de correspondencia utilizados en memorias caché. [Comprender]
- Determinar la estructura y el tamaño de los campos que componen a una dirección de memoria para acceder a la caché, dadas las características de la jerarquía de memoria y de la CPU con la que se opera. [Aplicar]
- Definir y comparar los algoritmos de escritura write-through y write-back. [Comprender]
- Describir el problema de coherencia de caché y cómo resolverlo. [Recordar]

Pipeline [Familiaridad] y Superescalaridad [Familiaridad]

- Describir la técnica de pipelining [Recordar]
- Explicar por qué el uso de la técnica de pipelining típicamente mejora el rendimiento de los programas. [Comprender]

- Definir los diferentes tipos de obstáculos que ocurren durante la ejecución de un programa en una CPU con pipeline y describir soluciones para cada uno de los problemas. [Comprender]
- Describir diferentes tipos de técnicas de predicción de saltos y sus estructuras de hardware asociadas. [Recordar]
- Indicar los requerimientos de hardware mínimos de una CPU superescalar. [Recordar]
- Describir las estructuras de hardware, ventana de instrucciones y buffer de reorden, y explicar su utilidad en el contexto de la ejecución fuera de orden. [Comprender]
- Describir la técnica de renombrado de registros y las estructuras de hardware asociadas a su implementación. [Recordar]
- Definir paralelismo a nivel de máquina y paralelismo a nivel de instrucción y explicar la relación entre ambos conceptos. [Comprender]

4. TEMARIO

• Sistemas de numeración.

• Códigos y errores

• Representación interna de datos.

• Algebra de Boole: Funciones y compuertas lógicas.

• Circuitos Combinatorios.

• Circuitos Secuenciales y Memoria..

• Máquina de estados. Máquina Lógica General.

• Arquitecturas Von Neumann.

• Organización de la CPU.

• Sistema de Entrada / Salida.

• Interrupciones.

• Arquitectura comercial de ejemplo.

• Programación de bajo nivel: lenguaje ensamblador, estructuras de datos, pasaje de parámetros, recursividad, interacción con la E/S.

• Jerarquía de memoria. Memoria Caché.

• Pipelining

• Superescalaridad.

5. CONOCIMIENTOS PREVIOS Y UBICACIÓN EN LA CARRERA

- **Conocimientos previos exigidos:** lógica, programación, ciencias experimentales**.**
- **Conocimientos previos recomendados:** estructuras de datos y algoritmos, matemática discreta, lenguaje C.