Unidad 1 – Introducción a los Microprocesadores (μP)

1. Voltaje

Señales eléctricas, se representan en binario:

- **0** → Voltaje **bajo**
- **1** → Voltaje **alto**

Con estas dos posibilidades los circuitos electrónicos pueden ejecutar operaciones lógicas. Una señal digital simulada es la **base del cómputo.**

2. Componentes electrónicos

- **Resistor**: Controla el paso de la corriente
- **Capacitor**: Almacena carga eléctrica
- **Diodo**: Deja pasar corriente en una sola dirección
- **Transistor**: Funciona como un interruptor, encendiendo o apagando el paso de corriente. Es la base de la lógica y de los microprocesadores modernos.
- **Bobina**: Almacena energía en forma de campo magnético.
- **Tubo** de vacío: Usado en los primeros equipos electrónicos, antes de los transistores.

Dato clave: El transistor permitió la miniaturización y construcción de procesadores modernos.

3. Decimal vs binario

Decimal (Base 10): Cada posición representa una potencia de 10
Binario (Base 2): Cada posición representa una potencia de 2

4. Puertas lógicas (GATES)

Tipos de puertas:

- **Buffer**: Deja pasar la señal sin cambios
- **NOT** (inversor): Convierte 0 en 1 y 1 en 0
- **AND**: La salida es 1 solo si ambas entradas son 1
- **OR**: La salida es 1 si al menos una entrada es 1
- **NAND**: Es el inverso de la AND
- **NOR**: Es el inverso de la OR
- **XOR** (OR exclusivo): La salida es 1 solo si las entradas son diferentes
- **XNOR**: Es el inverso del XOR

5. La compuerta NAND

Un ejemplo muy usado en la enseñanza es el circuito integrado **7400**, que contiene **4 compuertas NAND de 2 entradas.**

Con solo compuertas **NAND** se puede construir **cualquier otro tipo** de puerta lógica

En una **protoboard**, se conectan los **pines** de **entrada** (A y B) y se observa la salida a través de un **LED**

6. NOR Gate

La compuerta **NOR** es simplemente una **OR invertida**

Su salida es **1** solo cuando **todas** las **entradas** son **0**

En cualquier **otro caso**, la **salida** es **0**

Al igual que NAND, NOR **es universal**. Combinando **varias NOR** se pueden construir **todas las demás puertas lógicas**.

7. Exclusive OR (XOR) y Exclusive NOR (XNOR)

- **XOR** (OR exclusivo): La **salida** es **1** solo si las **entradas** son **diferentes**
- **XNOR**: Es lo **opuesto** al **XOR**, su **salida** es **1** solo si las **entradas** son **iguales**

La XOR es la **base** de los **sumadores binarios**, que se usan en **operaciones aritméticas**.

8. Flip-Flop

Es un circuito **diferente** a las puertas lógicas combinacionales **porque tiene memoria**.

Puede **guardar** un **valor de 0 o 1** hasta que recibe una **señal** de reloj (clock)

Es la **base** de los **registros** y de la **memoria** dentro de un **procesador**

Mientras que la lógica combinacional calcula, **los flip-flops recuerdan**.

Permiten crear **circuitos secuenciales**, que son **fundamentales** para que el **procesador** siga una **secuencia** de **instrucciones**.

9. Adder (Sumador)

El **sumador** es un circuito **construido** con **compuertas lógicas** (AND, XOR, OR). Sirve para **sumar números binarios**, **bit a bit.**

- **Carry** (**acarreo**): Aparece cuando la suma de dos bits genera un "**1 extra**" que se debe pasar a la siguiente posición.
- **Overflow**: Ocurre cuando el resultado **no cabe** en el **número** de **bits disponibles**.

Los **sumadores** forman parte de la **Unidad Aritmético-Lógica** (**ALU**), el corazón de cualquier microprocesador.

10. Multiplier (Multiplicador)

La multiplicación en hardware se construye a partir de sumadores y compuertas.
Más compleja que la suma, pero con la misma lógica: Descomponer el cálculo en operaciones con 0 y 1.

11. Multiplexer & Demultiplexer

- **Multiplexer** (**MUX**): Selecciona una de varias entradas y la envía a una sola salida. Depende de las señales de control. Ej: Control de TV (eliges qué canal ver)
- **Demultiplexer** (**DEMUX**): Hace lo contrario, envía una señal de entrada hacia una de varias salidas. Ej: Distribuidor de agua que envía flujo hacia una tubería específica

En un procesador, sirven para dirigin información y gestionar datos entre distintos módulos.

12. State machine (Máquina de estados)

Si combinamos flip-flops con lógica combinacional, obtenemos una máquina de estados.
Se trata de un sistema que cambia entre diferentes estados según sus entradas y un reloj
Son la base de los algoritmos en hardware, donde se siguen pasos secuenciales según condiciones.
Ej: Un semáforo que cambia verde → amarillo → rojo siguiendo un orden establecido

13. Dynamic RAM (DRAM)

La DRAM es un tipo de memoria muy usada porque es barata y densa (mucha información en poco espacio)

- Organizada en una matriz de filas y columnas
- Necesita señales especiales (CAS y RAS) para seleccionar posiciones
- Es volátil: Pierde los datos si no se refresca constantemente

14. Static RAM (SRAM)

- Usa mas transistores por bit (unos 6)
- Es mucho más rápida que la DRAM
- No necesita refresco constantemente
- Es más cara y ocupa más espacio

Se usa en las cachés del procesador, donde la velocidad es crítica. Mientras que la DRAM se usa para la RAM principal, la SRAM acelera operaciones dentro del procesador.

15. Flash Memory

Es un tipo de memoria no volátil, es decir, conserva los datos aunque no tenga energía.

- Está construida con puertas NAND
- El acceso es más lento que la RAM
- Los datos deben borrarse en bloques completos, no bit a bit.

Ej: Memorias USB, discos SSD, tarjetas SD.

Es ideal para almacenamiento masivo y portátil, aunque no para la memoria de trabajo principal.

16. Arquitectura Von Neumann

Es la base de las computadoras modernas

- Existe un bus del sistema que conecta:
  - La memoria
  - La CPU
  - Los dispositivos de entrada/salida(I/O)
- La CPU se divide en dos partes:
  - Unidad de Control (CU): Interpreta y coordina instrucciones
  - Unidad Aritmético-Lógica(ALU): Realiza cálculos y comparaciones
- La memoria principal guarda tantos datos como instrucciones en el mismo espacio.

17. Computing System in layers (Capas del sistema computacional)

- **Hardware**: Lo físico (CPU, memoria, periféricos)
- **Firmware**: Instrucciones básicas que controlan directamente el hardware
- **Software**: Programas de usuario y aplicaciones
- El compilador/intérprete traduce el código humano (C, Python, etc.) a binario para que lo entienda la máquina

Si el software cambia, ¿cómo sabe el CPU qué hacer al encenderse? La respuesta está en el BIOS

18. CISC vs RISC

CISC (Complex Instruction Set Computing)

- Muchas instrucciones, algunas muy complejas

- Más fácil para programadores
- Tiende a ser más lento por ciclo
- Ejemplo: Intel x86

RISC (Reduced Instruction Set Computing)

- Instrucciones más simples y rápidas
- Requiere más instrucciones para lograr una tarea
- Optimiza la velocidad
- Ejemplo: ARM (Muy usado en móviles)

Hoy en día la mayoría de procesadores combinan características de ambos.

19. Lenguajes de programación y niveles

- **Lenguaje de máquina**: 0s y 1s, directamente ejecutado por la CPU
- **Assembler**(**ASM**): Lenguaje de bajo nivel, cercano al hardware
- **Lenguajes de alto nivel** (C, Python, Java, etc.): fáciles de usar por humanos, pero necesitan un compilador o intérprete.

20. BIOS (Basic Input Output System)

Es el firmware básico que se ejecuta al encender la computadoras

Funciones:

- Inicializar hardware
- Realizar pruebas de arranque (POST: Power On Self Test)
- Cargar el sistema operativo en memoria

Antes se almacenaba en ROM, hoy en día en memoria Flash, lo que permite actualizarlo.

Sin BIOS, la CPU no sabría qué hacer al encender.

21. The Bootloader

Es un programa muy pequeño que se ejecuta después del BIOS

- Copia el sistema operativo desde el disco (u otro medio) a la memoria RAM
- Le da control al sistema operativo para que inicie
- Usa hexadecimal (base 16: 0-9, A-F) para representar direcciones y datos
- Ejemplo: La dirección 0x7C00 es el lugar donde se carga el bootloader

Es el puente entre el BIOS y el SO, el que le dice a la computadora: “ya puedes empezar”

22. Operative System (OS)

Es el software que permite al usuario interactuar con el hardware

- **Sistema de archivos:** Organiza datos en memoria y disco como un árbol
- **Gestión de procesos:** Controla los programas en ejecución
- **Drivers** (**controladores**): Permiten la comunicación entre hardware y software
- **Interfaz gráfica** (**GUI**): Entorno visual para el usuario
- **Aplicaciones**: Los programas que usamos a diario

Ejemplos: Linux, Windows, macOS, Android

23. Linux, why not?

- Su kernel está programado en bajo nivel, lo que lo hace eficiente y cercano al hardware
- Es código abierto, lo que permite aprender y modificar cómo funciona un sistema operativo
- Es ampliamente usado en servidores, supercomputadoras y móviles (Android está basado en Linux)

Existes muchas distribuciones (Ubuntu, Debian, Fedora, etc.) y cada una adapta diferentes necesidades.

Conclusión global

- Todo empieza con voltajes (0 y 1)
- Con componentes electrónicos básicos y puertas lógicas se construyen circuitos más complejos
- Estos bloques permiten crear la ALU, la memoria y la unidad de control, que juntos forman el microprocesador
- La arquitectura von Neunmann organiza cómo se conectan memoria, CPU e I/O
- Firmware y software permiten que el hardware cobre vida y se adapte a distintas tareas
- El arranque del sistema (BIOS → Bootloader → SO) convierte una máquina en una herramienta interactiva
- Ejemplos como Linux muestran la potencia y versatilidad de estos sistemas

Frase clave: Un microprocesador es la combinación de electrónica, lógica y software que transforma simples 0 y 1 en toda la tecnología que usamos cada día.