Actividad 1: Introducción a la Arquitectura de Computadoras (Material complementario)

1. El centro de control: interconectando componentes

La **Placa Madre** es el componente principal al que se conectan los demás. Imagina la placa madre como un centro de distribución donde se coordinan todas las funciones. Al igual que las

Actividad 1: Introducción a la Arquitectura de Computadoras

(Material complementario)

1.1. ¿Por qué es importante la Arquitectura de Computadoras?

La arquitectura de computadoras no solo se estudia para saber "cómo está hecha" una PC, sino porque:

- Condiciona el desarrollo de software (qué lenguaje, qué compilador, qué sistema operativo).
- Afecta el rendimiento de aplicaciones: los juegos, las apps móviles, los servicios web
- Define compatibilidades: por ejemplo, saber si un software corre en ARM o x86. Ejemplo real: Apple migró sus computadoras de procesadores Intel (x86) a Apple Silicon (ARM).

Muchos programas tuvieron que recompilarse o adaptarse.

1.2. Aspectos Relevantes (Profundización)

Aspecto	¿Por qué importa?
Rendimiento	Más instrucciones por segundo = tareas más rápidas. Se mide en FLOPS, IPC, frecuencia.
Eficiencia energética	Crucial en notebooks, celulares, servidores. Menor consumo = menor calor y mayor autonomía.
Expansión y actualización	Un diseño modular permite reemplazar RAM, CPU o disco sin cambiar toda la máquina.
Compatibilidad	La arquitectura debe permitir que el sistema operativo y los programas interactúen correctamente.

Consejo: Como desarrollador, siempre ten en cuenta las limitaciones de la plataforma destino (ej: no es lo mismo desarrollar para una notebook que para un microcontrolador).

1.3. Evolución de las Arquitecturas

Esta evolución no solo mejoró el hardware: también cambió cómo se programa.

Generación	Clave histórica	Qué cambió para el programador
Primera	Válvulas de vacío	Lenguaje máquina puro, sin abstracciones.
Segunda	Transistores	Aparecen lenguajes de alto nivel (FORTRAN, COBOL).
Tercera	Circuitos integrados	Multiprogramación, compiladores más complejos.
Cuarta	Microprocesadores	Nace la PC, desarrollo de sistemas operativos personales.
Quinta	Redes e Internet	Multiprocesamiento, software distribuido, programación orientada a objetos.
Actualidad	IA y especialización	Se desarrolla para GPUs, TPUs, y con modelos de ML en mente.

Dato: Hoy, una notebook común tiene más potencia que una supercomputadora de los 90.

1.4. Tipos de Arquitectura

Arquitectura von Neumann

- Ventaja: simple y económica.

- Desventaja: cuello de botella de memoria (bus compartido).
- Ejemplo: Intel Core i5.

Arquitectura Harvard

- Ventaja: lectura simultánea de instrucciones y datos.
- Desventaja: diseño más complejo.
- Ejemplo: Arduino (ATmega328), microcontroladores PIC.

CISC (Complex Instruction Set Computer)

- Instrucciones complejas que hacen mucho en una sola operación.
- Más fáciles de programar en ensamblador.
- Ejemplo: procesadores x86.

RISC (Reduced Instruction Set Computer)

- Instrucciones simples y rápidas.
- Más eficiente, especialmente en tareas móviles o embebidas.
- Ejemplo: ARM, MIPS.

Nota: RISC-V es una arquitectura abierta y libre, cada vez más usada en educación y nuevos diseños.

1.5. Instrucciones y Código Máquina

¿Qué es el código máquina?

Son instrucciones binarias que la CPU ejecuta directamente. Toda ejecución (inclusive Python) termina convirtiéndose en instrucciones de este tipo, mediante un intérprete o compilador.

ISA – Instruction Set Architecture

Define qué instrucciones reconoce el procesador.

ISA	**Uso habitual**
x86 / x64	PCs, notebooks, servidores.
ARM	Celulares, tablets, embebidos.
RISC-V	Proyectos educativos, hardware libre.

Compatibilidad entre ISAs

- Compilar para x86 ≠ ARM: no es intercambiable.
- Necesitás recompilar o usar un emulador (ej: QEMU, Rosetta 2 en Mac).

¿32 o 64 bits?

Sistema	Límite RAM	Compatibilidad
32 bits	~4 GB	Solo software 32 bits
64 bits	+ de 4 GB	Corre software 32 y 64 bits

Recomendación: Si programás para múltiples arquitecturas, usá lenguajes multiplataforma (Python, Java) o compiladores cruzados (cross-compilation).

Conclusión

Comprender la arquitectura de computadoras no es solo para ingenieros de hardware. Para quienes programan, esta comprensión:

- Ayuda a optimizar software.
- Permite resolver errores de compatibilidad.
- Mejora la toma de decisiones sobre qué herramientas usar.