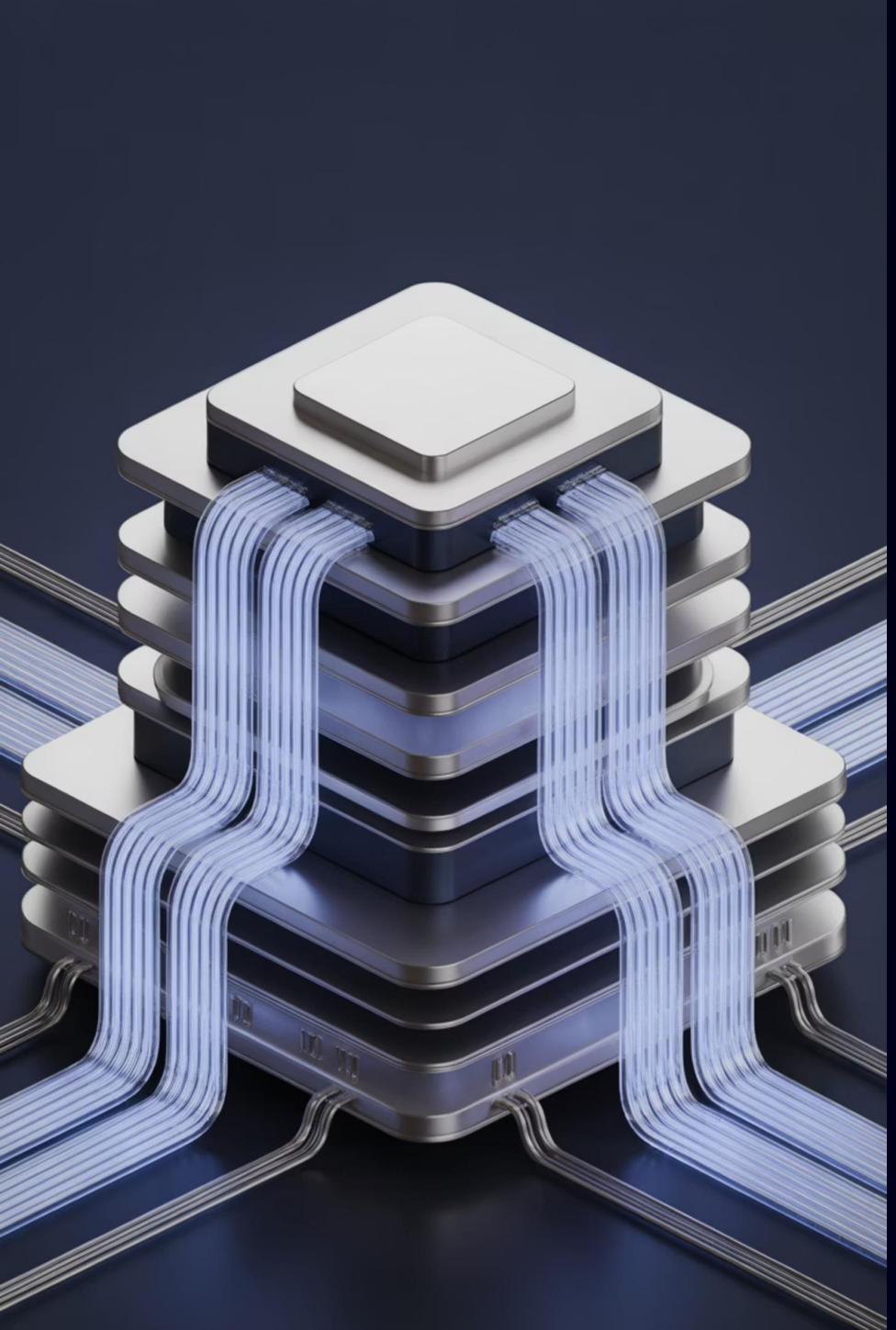
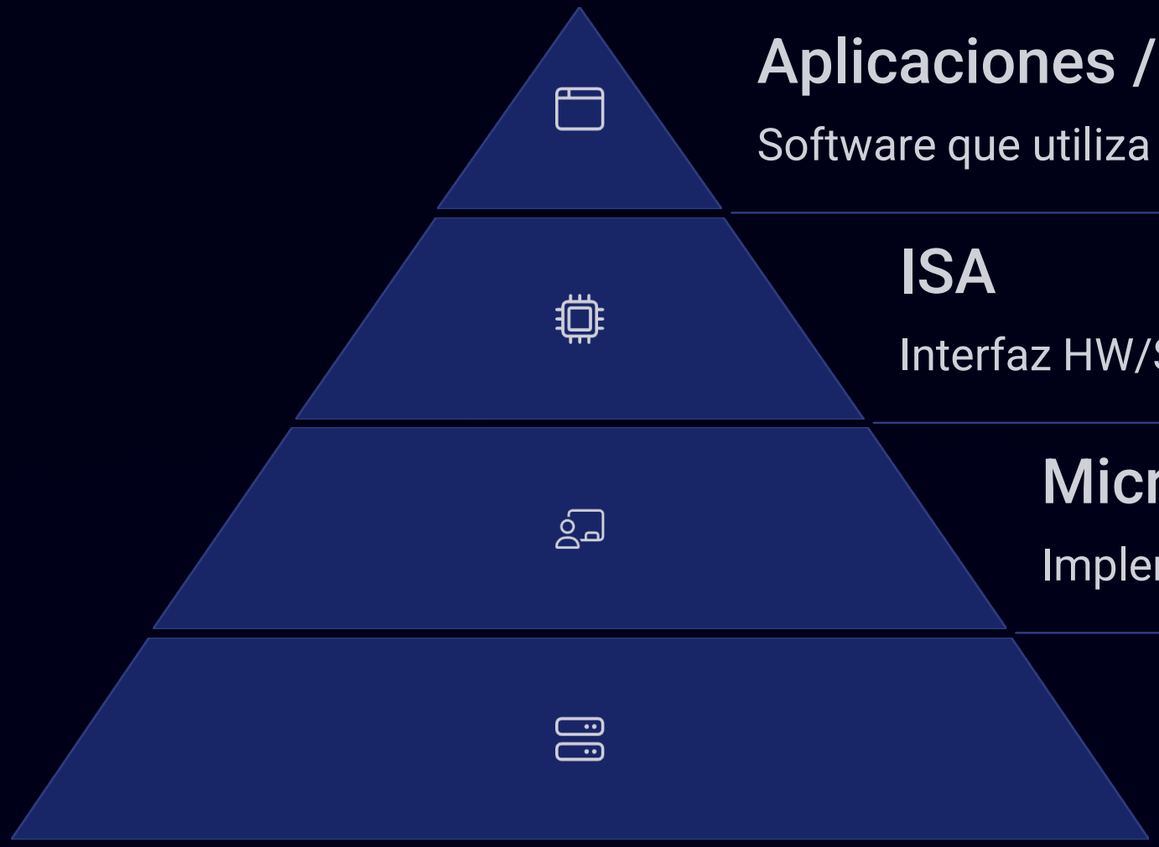


Arquitectura de Computadoras: Sistemas, ISA y Microarquitectura

Las Tres Capas de la Arquitectura



Aplicaciones / SO

Software que utiliza los recursos del hardware

ISA

Interfaz HW/SW (contrato visible al software)

Microarquitectura

Implementación de la ISA (pipelines, cachés, núcleos)

Arquitectura de sistemas

Organización global CPU–memoria–E/S

Estas capas representan diferentes niveles de abstracción en el diseño de computadoras, desde la organización física de los componentes hasta la interfaz que ve el software.

Arquitectura de Sistemas

Es la organización macro de un computador: cómo se conectan y coordinan la CPU, la(s) memoria(s), los buses, la E/S y los periféricos. Responde dónde residen instrucciones y datos, cómo circulan y con qué latencias/ancho de banda.

Objetivos Típicos

- Maximizar rendimiento y escalabilidad
- Minimizar latencia y consumo
- Garantizar coherencia/consistencia de memoria
- Simplificar la integración de periféricos

Métricas Clave

- Latencia y ancho de banda de memoria/buses
- Escalabilidad (núcleos, sockets)
- Consumo energético
- Eficiencia de E/S

Componentes y Decisiones Clave

Modelo de memoria

Unificada o separada (ej. Von Neumann vs Harvard)

Buses e interconexiones

Bus único/jerárquico, anillo, malla/torus, NoC (Network-on-Chip)

Multiprocesamiento

UMA/SMP (latencia uniforme), NUMA (latencia no uniforme), COMA; coherencia de caché

Jerarquía de memoria

Caches on-chip, memoria principal (DRAM), almacenamiento (SSD/HDD), memoria no volátil (NVRAM)

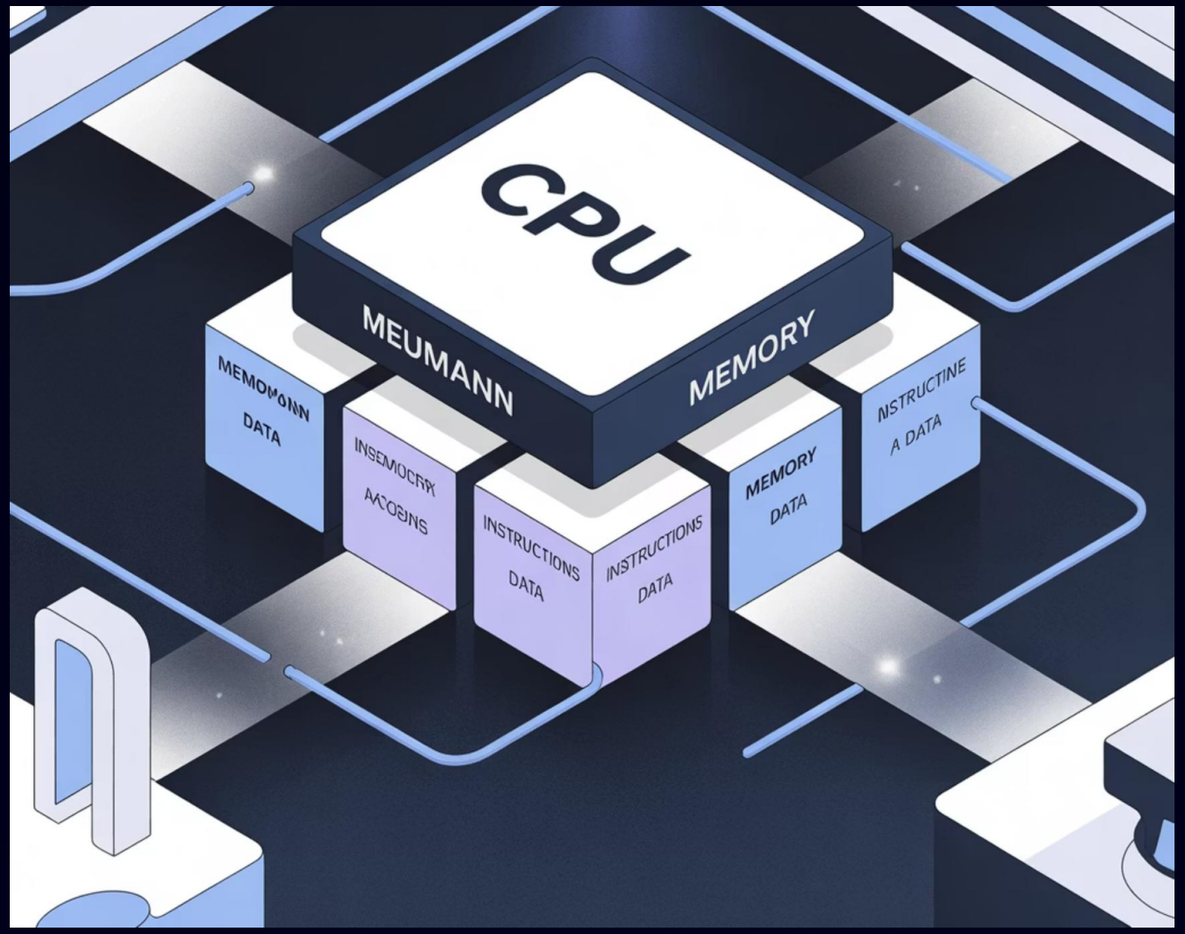
DMA y controladores

Canales de E/S, temporizadores, interrupciones, MMIO

Modelos Clásicos de Organización de Memoria

Von Neumann (Princeton)

Instrucciones y datos comparten misma memoria y bus. Sencillo y flexible, pero con posible cuello de botella.



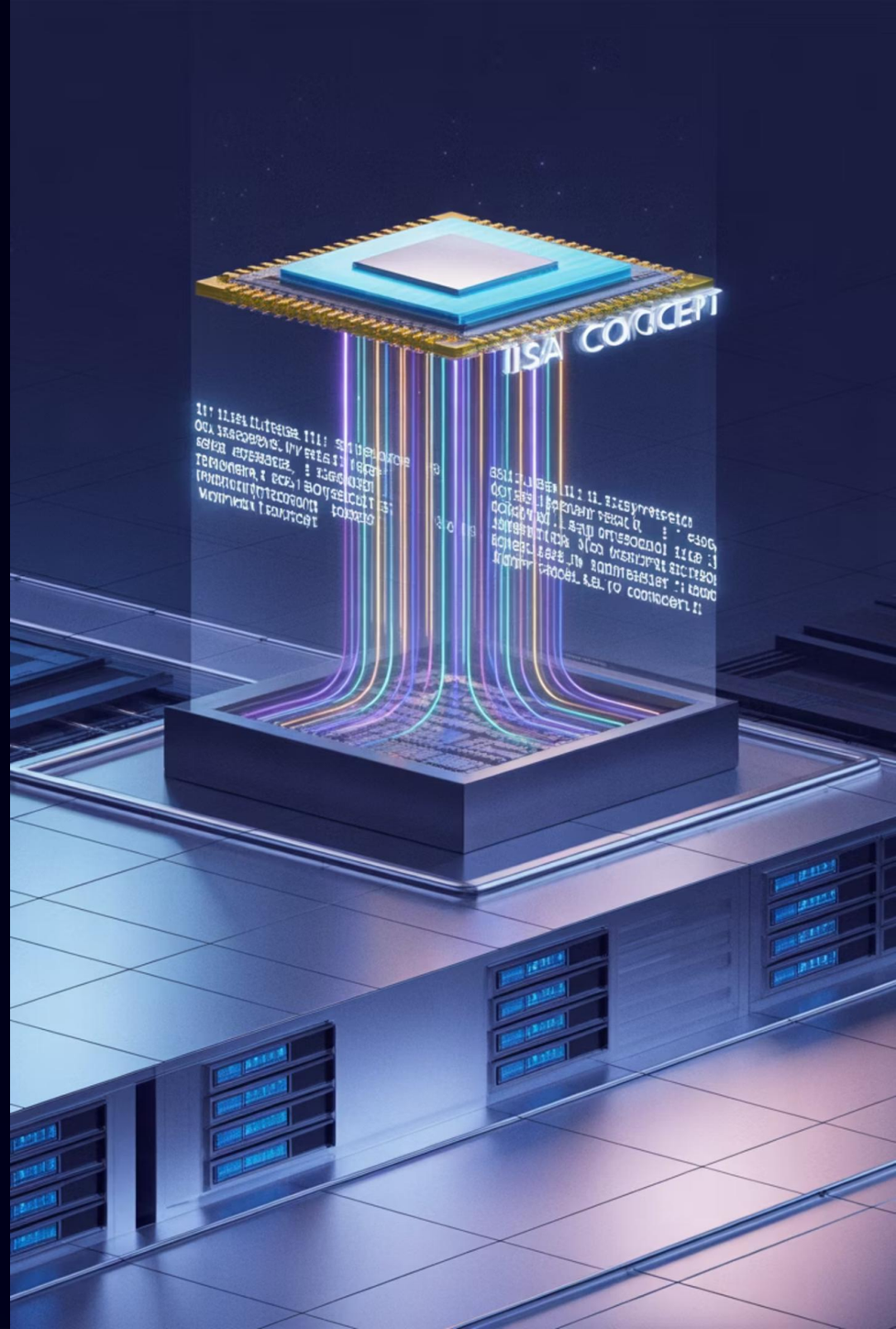
Harvard

Instrucciones y datos en memorias y buses separados, permitiendo accesos simultáneos. Común en microcontroladores y DSPs.

ISA — Instruction Set Architecture

Es el "contrato" visible entre hardware y software. Especifica qué puede pedir el software al procesador y cómo: conjunto de instrucciones, registros, modos de direccionamiento, formatos binarios, modelo de memoria y gestión de excepciones/interrupciones.

También incluye aspectos de ABI (convenciones de llamada, layout de pila) cuando se normalizan a nivel de plataforma.



Componentes de una ISA

Repertorio de instrucciones

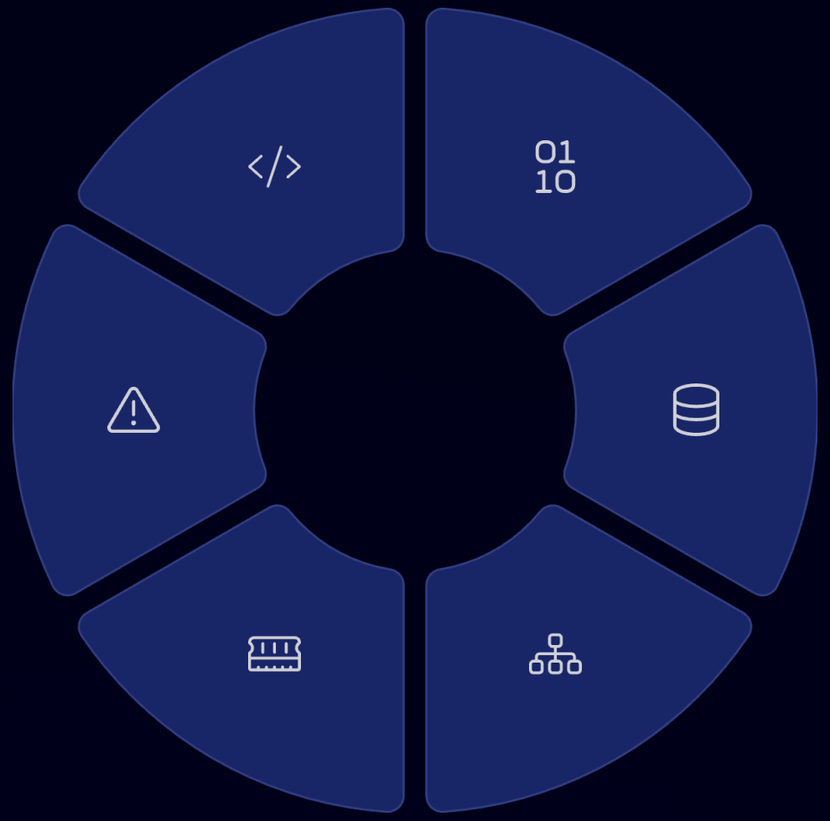
Aritméticas, control de flujo, memoria, SIMD, atómicas

Excepciones e interrupciones

Trampas, fallos de página, syscalls

Modelo de memoria

Orden/consistencia de accesos, alineación, atomics



Formato de instrucciones

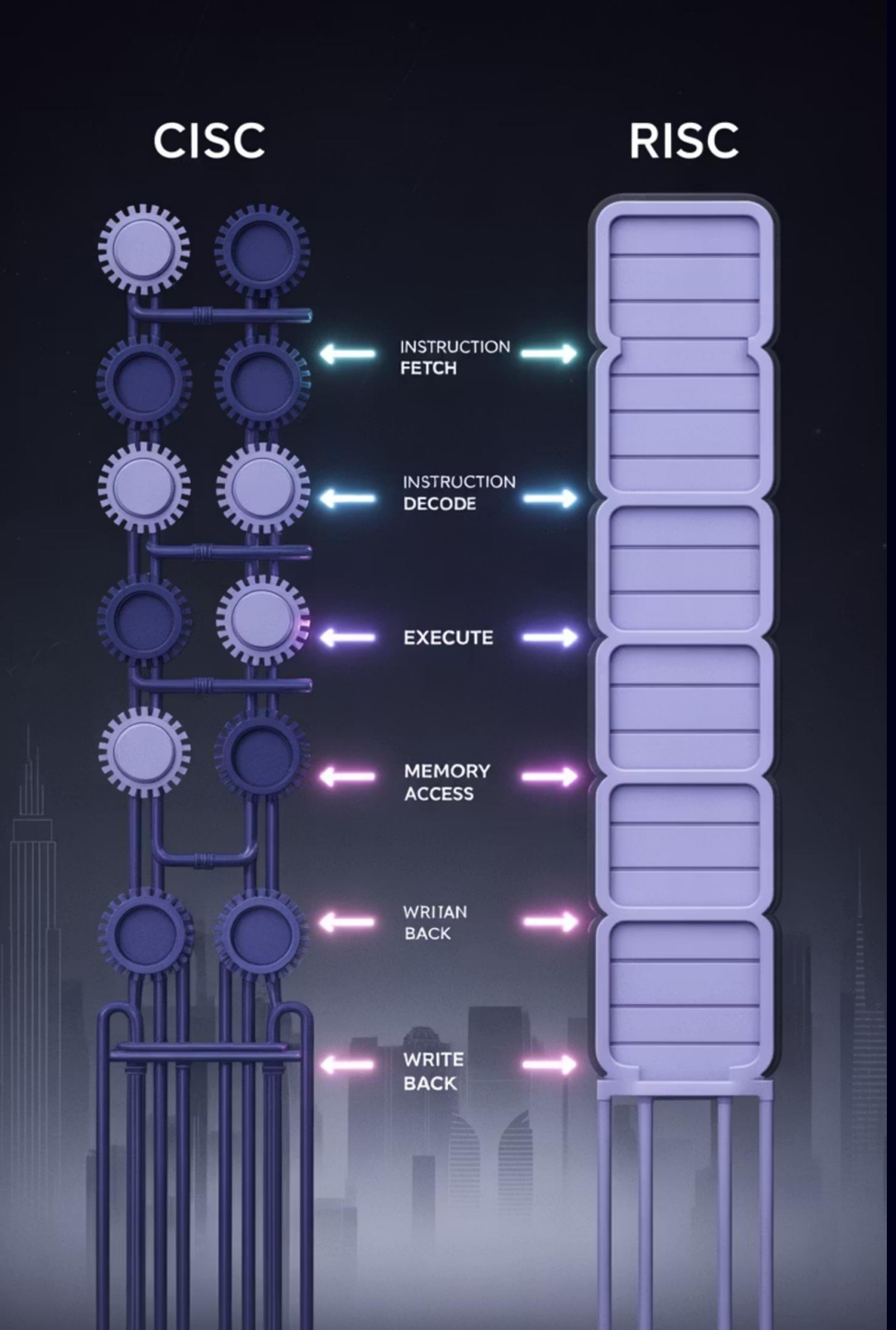
Longitud fija/variable, campos op-code/registro/inmediato

Conjunto de registros

Generales, punto flotante, vectoriales, especiales

Modos de direccionamiento

Inmediato, directo, indirecto, indexado, base+desplazamiento



CISC vs RISC: Filosofías de Conjunto de Instrucciones

CISC (Complex Instruction Set Computer)

Filosofía de diseño que prioriza instrucciones complejas y potentes que pueden realizar múltiples operaciones de bajo nivel en una sola instrucción.

- Objetivo: minimizar líneas de código y simplificar compiladores
- Instrucciones de longitud variable y ciclos múltiples
- Ejemplo emblemático: Familia x86 de Intel/AMD

RISC (Reduced Instruction Set Computer)

Enfoque que utiliza un conjunto más pequeño de instrucciones simples y optimizadas que se ejecutan típicamente en un solo ciclo.

- Objetivo: maximizar velocidad y eficiencia energética
- Instrucciones de longitud fija y formato consistente
- Arquitectura de carga-almacenamiento (load-store)
- Ejemplos destacados: ARM, MIPS, SPARC, PowerPC

Ejemplos y Extensiones de ISA

Ejemplos de ISA

- x86-64 (Intel, AMD)
- ARMv8-A (smartphones, tablets)
- RISC-V (abierta, personalizable)
- MIPS32/64 (educación, embebidos)
- Power (IBM, servidores)
- AVR-8 (ATmega328P, Arduino)

Extensiones habituales

- SIMD/Vectorial (AVX, NEON, SVE)
- Criptografía (AES/SHA)
- Compresión
- Control de memoria (MPA)
- Virtualización

Las ISA modernas suelen tener un conjunto base de instrucciones y múltiples extensiones opcionales para casos de uso específicos.

Métricas y Trade-offs en ISA

Densidad de código

Cuánto espacio ocupa el código compilado

Ortogonalidad

Regularidad y consistencia del conjunto de instrucciones

Facilidad de compilación

Qué tan sencillo es generar código eficiente

Complejidad de decodificación

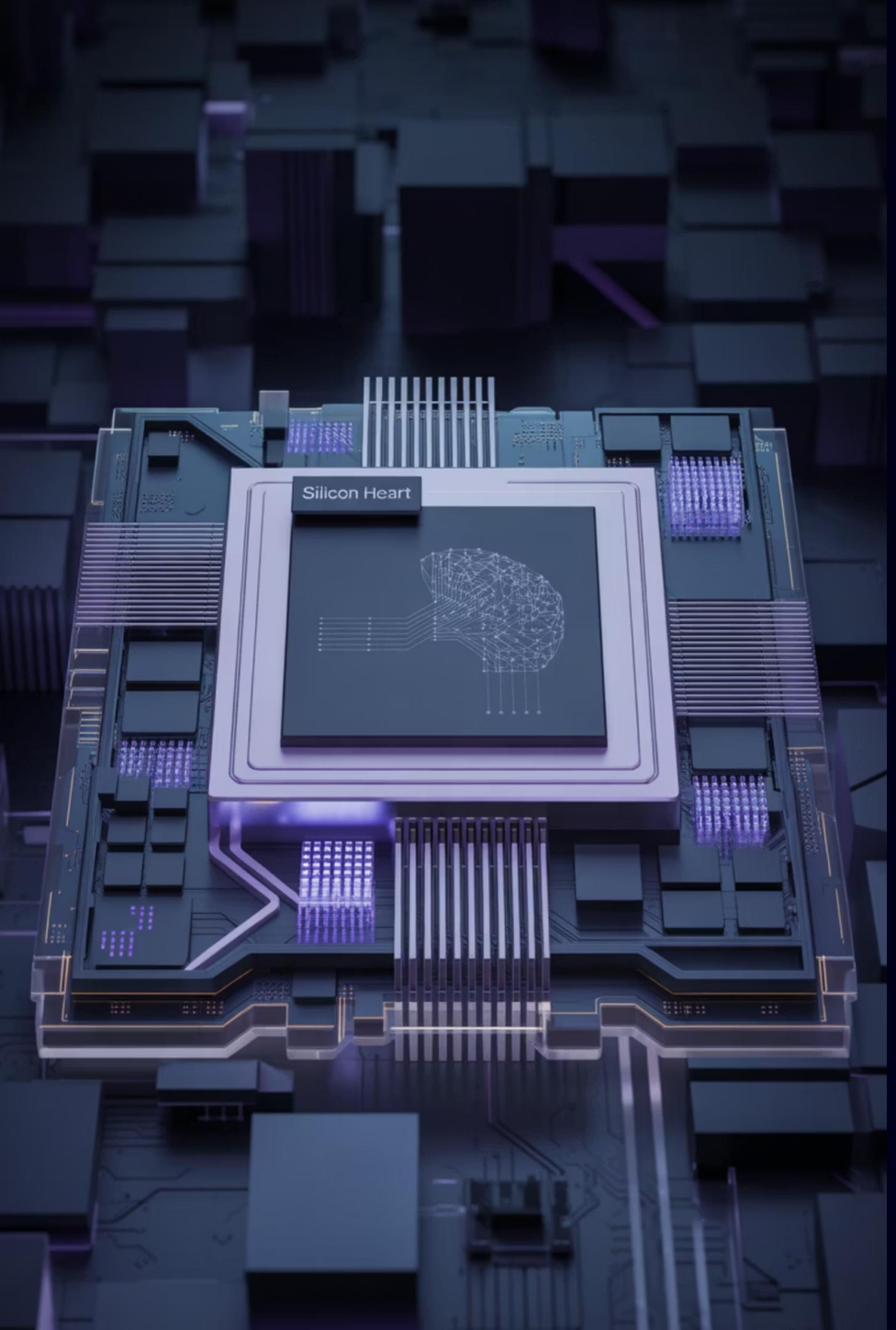
Recursos necesarios para interpretar instrucciones

Portabilidad binaria/ABI

Compatibilidad entre implementaciones

Estabilidad a largo plazo

Mantenimiento de compatibilidad con el tiempo



Microarquitectura

Es la implementación interna que ejecuta una ISA concreta. Dos microarquitecturas muy distintas pueden compartir la misma ISA (ej.: varios "Core" de Intel o varias "Zen" de AMD, todos x86-64).

La microarquitectura determina el rendimiento real, eficiencia energética y otras características prácticas del procesador, mientras mantiene la compatibilidad con el software existente.

Técnicas Avanzadas de Microarquitectura

Superscalar

Issue múltiple por ciclo y reordenamiento para ocultar latencias. Incluye renombrado de registros, ROB (reorder buffer), colas de reserva.

Predicción de Saltos

Anticipa el resultado de instrucciones condicionales mediante BTB (Branch Target Buffer) y BHT (Branch History Table).

Jerarquía de Caché

L1-I / L1-D separadas, L2 privada, L3 compartida; prefetchers; políticas de reemplazo y de coherencia (MESI/MOESI).

Paralelismo a Nivel de Hilo

SMT/Hyper-Threading, multinúcleo, clúster de núcleos heterogéneos (big.LITTLE).

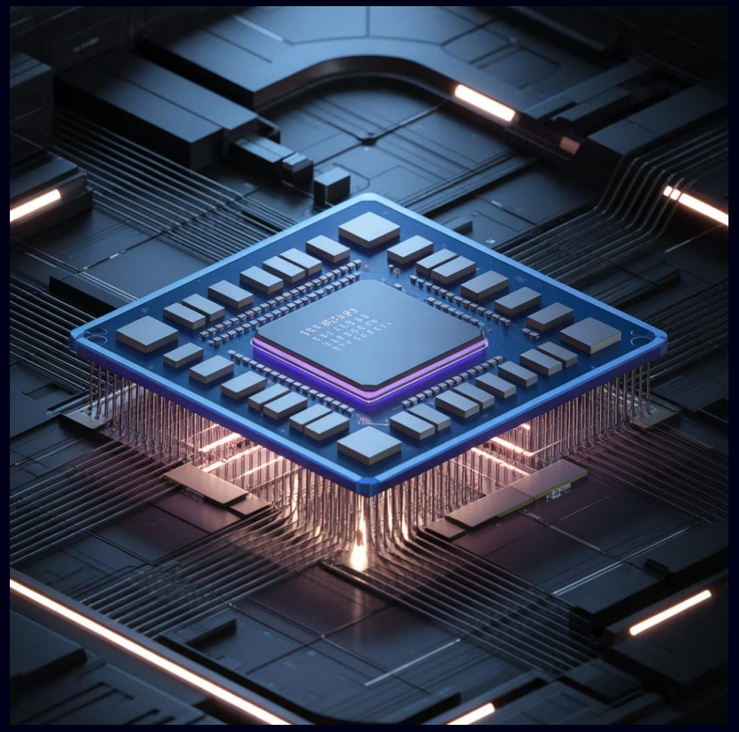
Gestión de Energía

DVFS, clock/power gating, dominios de voltaje, dark silicon.

Ejemplos de Microarquitecturas

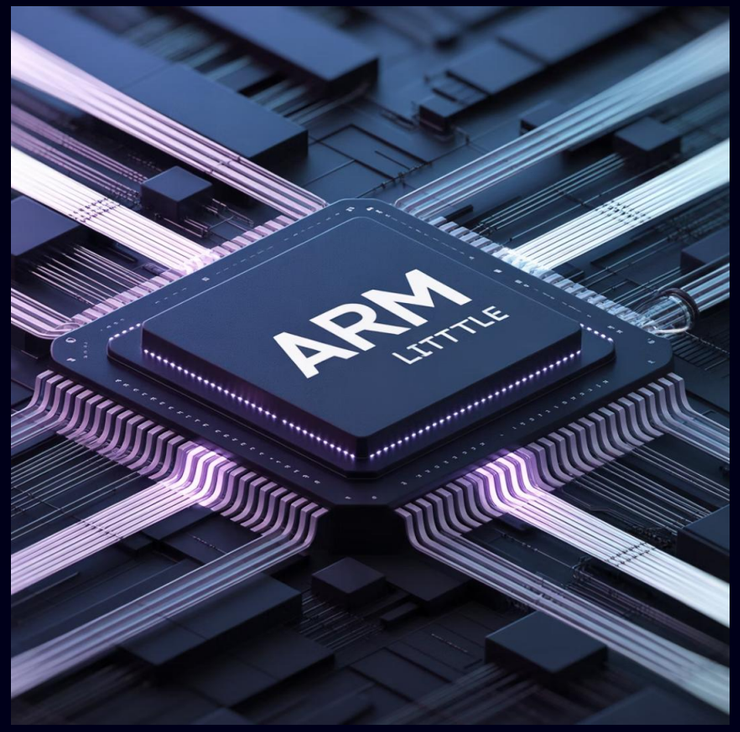
x86-64

- Intel "Skylake/Alder Lake"
- AMD "Zen 4/5"



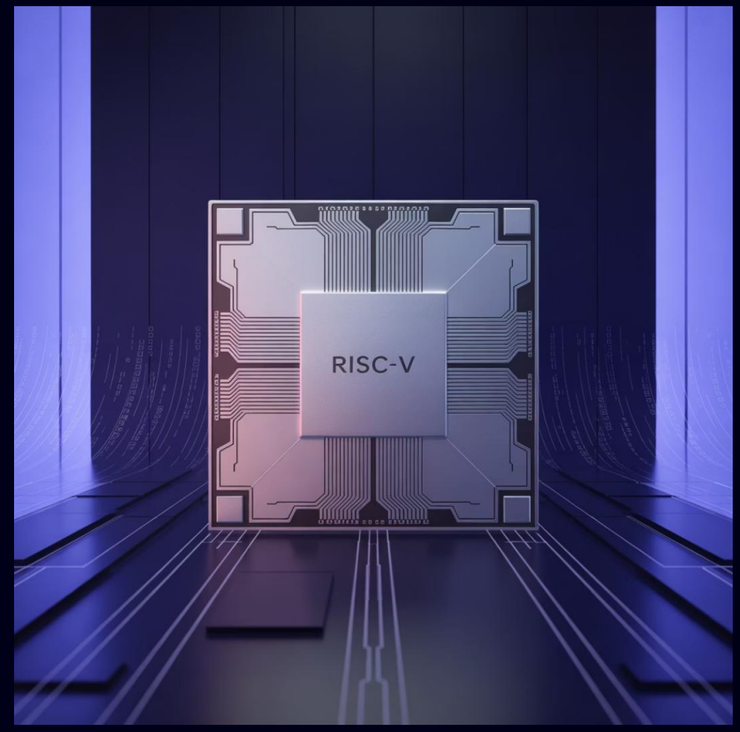
ARM

- ARM "Cortex-A76/A78"
- ARM "Cortex-M4/M7"



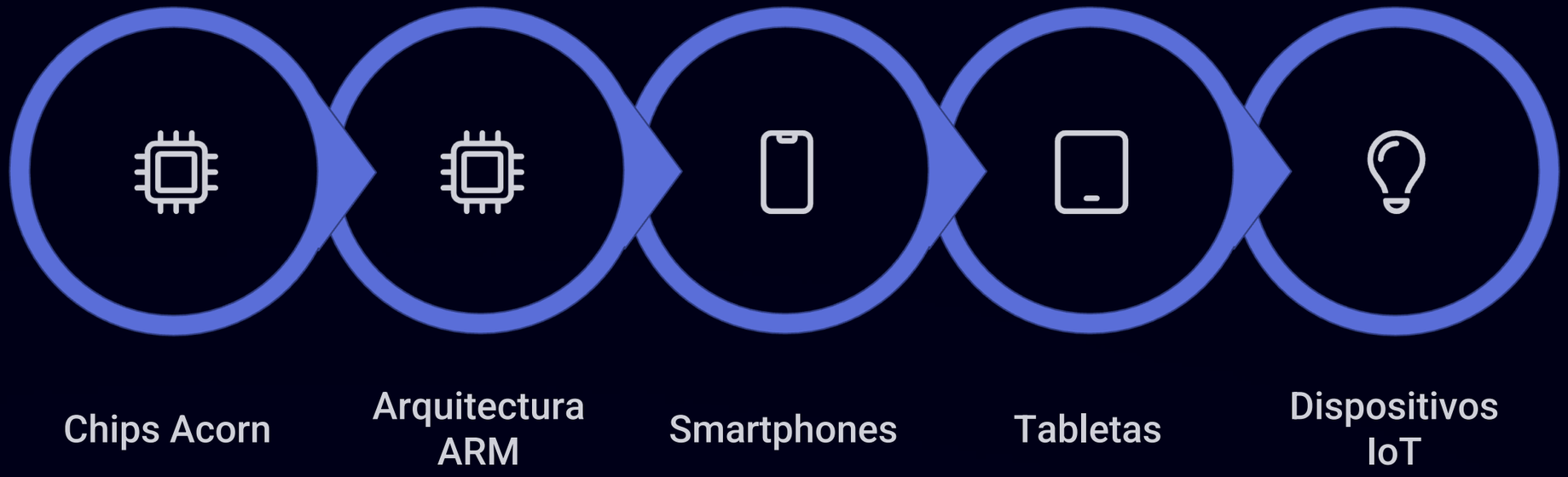
RISC-V

- Núcleos RISC-V Rocket
- Núcleos RISC-V BOOM



Todas estas implementan ISAs específicas pero con enfoques microarquitectónicos muy diferentes.

RISC y su Expansión: ARM y Más Allá



Características Clave de ARM

- Diseño RISC eficiente con bajo consumo energético
- Modelo de licenciamiento flexible que permite personalización
- Arquitectura de 32 bits (ARMv7) y 64 bits (ARMv8-A)

Implementaciones Destacadas

- Apple: Chips M1/M2 para MacBooks y iPads, A-series para iPhones
- Qualcomm: Procesadores Snapdragon para smartphones Android
- Samsung: Exynos para dispositivos Galaxy

Relación Entre las Capas

Independencia Relativa

Una ISA puede vivir en múltiples microarquitecturas (compatibilidad binaria a lo largo del tiempo). La arquitectura de sistemas puede variar (UMA/NUMA, bus/NoC) sin cambiar la ISA.

Influencia Bidireccional

La elección de ISA condiciona el decodificador, registros y excepciones; la microarquitectura condiciona la profundidad de pipeline, tamaño de cachés; la organización del sistema condiciona la coherencia y las latencias efectivas.

Ejemplo concreto: ATmega328P → ISA AVR-8 + microarquitectura simple de 8 bits + organización Harvard en chip.

Un Intel Core i9 → ISA x86-64 + microarquitectura superscalar + split-cache/DRAM compartida (Von Neumann mod.).

Arquitecturas de Computadoras

La Revolución de Von Neumann (1945)

Origen e Innovación

Propuesta por el matemático John von Neumann en el histórico informe EDVAC (Electronic Discrete Variable Automatic Computer) de 1945, estableciendo las bases de la computación moderna.

Características Revolucionarias

- Memoria única para almacenar tanto datos como instrucciones: el revolucionario concepto de "programa almacenado"
- Unidad central de procesamiento (CPU) con unidad aritmético-lógica, registros y unidad de control
- Procesamiento secuencial de instrucciones siguiendo un ciclo de búsqueda-decodificación-ejecución

Implementaciones Históricas

EDVAC (1949), UNIVAC I (1951), IBM 701 (1953) y prácticamente todas las computadoras personales modernas siguen los principios fundamentales de von Neumann.

Von Neumann: Ventajas y Limitaciones

Ventajas

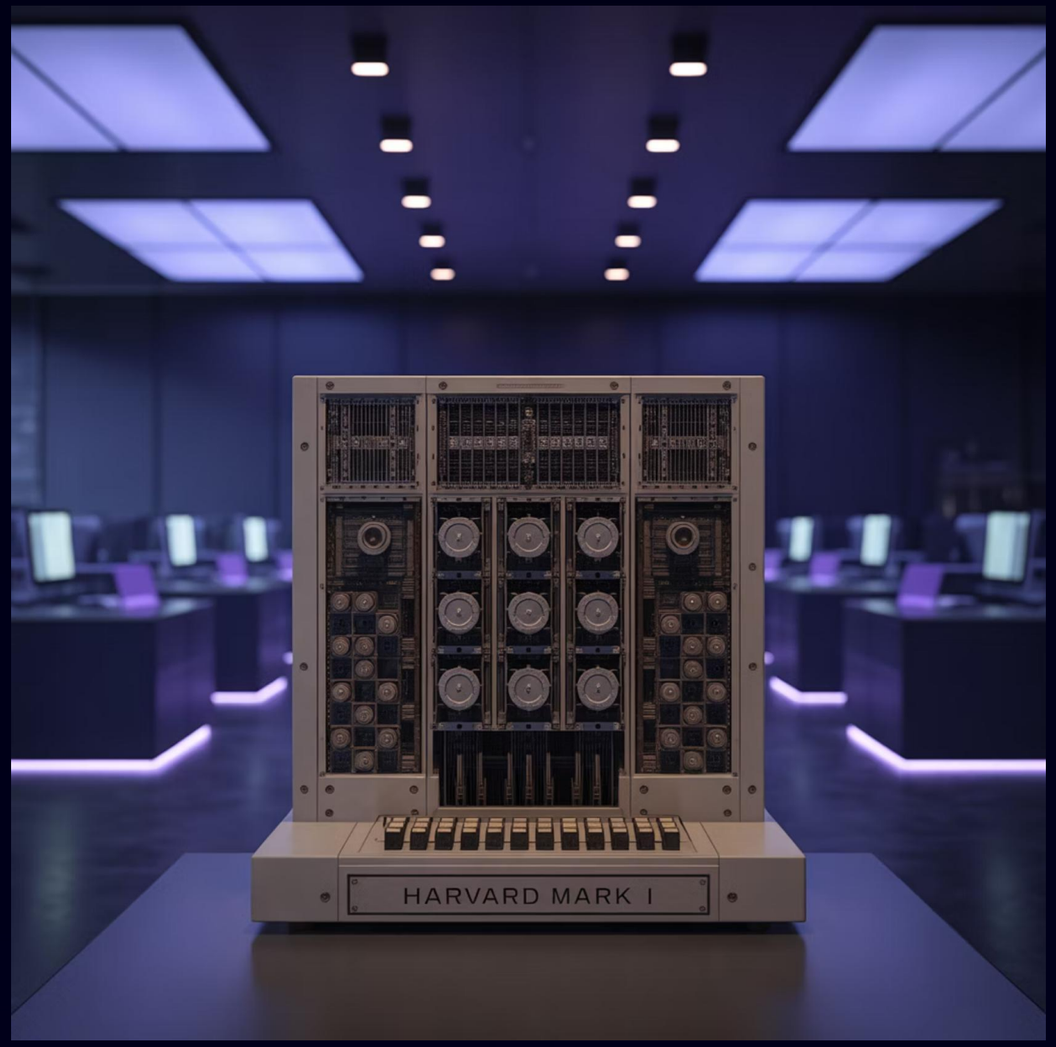
- Simplicidad conceptual y de implementación
- Programación flexible (software modificable)
- Adaptabilidad a diferentes tipos de aplicaciones
- Base para el desarrollo de lenguajes de programación

Limitaciones

- "Cuello de botella de Von Neumann": instrucciones y datos comparten el mismo bus
- Límites en el paralelismo de ejecución
- Velocidad restringida por acceso secuencial a memoria
- Ineficiencia energética en operaciones repetitivas

La arquitectura de von Neumann sigue siendo la base para la mayoría de computadoras personales y servidores actuales, incluyendo prácticamente todos los sistemas basados en procesadores x86 de Intel y AMD.

Arquitectura Harvard: Separación para Velocidad



Característica Principal

Memorias físicamente separadas para instrucciones y datos, con buses independientes que permiten accesos simultáneos.

Ventajas Clave

- Mayor velocidad por acceso paralelo a instrucciones y datos
- Optimización de ancho de banda de memoria
- Mejor rendimiento en aplicaciones de tiempo real

Implementaciones Modernas

Microcontroladores PIC de Microchip, procesadores digitales de señales (DSP) de Texas Instruments, y muchos sistemas embebidos donde la eficiencia energética y el rendimiento determinístico son cruciales.

La arquitectura Harvard, nombrada por la computadora Harvard Mark I (1944), fue desarrollada en la Universidad de Harvard bajo la dirección de Howard Aiken con apoyo de IBM.