La ALU

La Unidad Aritmética Lógica (ALU, por sus siglas en inglés: Arithmetic Logic Unit) es un componente fundamental del procesador (CPU) responsable de ejecutar operaciones aritméticas y lógicas requeridas por las instrucciones de un programa y se encuentra ubicado en el circuito digital combinacional. Actúa como el "cerebro matemático" del computador, permitiendo desde cálculos básicos (suma, resta) hasta operaciones complejas (comparaciones, desplazamientos de bits). Su diseño y eficiencia impactan directamente en el rendimiento del sistema.

Componentes Principales de la ALU

La ALU está estructurada en tres bloques funcionales:

- **Unidad Aritmética**
  - Funciones:
    - Operaciones básicas: suma, resta.
    - Operaciones avanzadas (según complejidad): multiplicación, división (en ALUs modernas).
    - En ALUs simples, multiplicación/división pueden implementarse mediante software o unidades especializadas (e.g., FPU).
- **Unidad Lógica**
  - Funciones:
    - Operaciones booleanas: AND, OR, NOT, XOR.
    - Desplazamientos de bits (*shift*): lógicos (relleno con ceros) y aritméticos (conserva el signo). (https://www.youtube.com/watch?v=FphdXXiHNPE)
    - Rotaciones de bits.
- **Circuitos de Control**
  - Funciones:
    - Interpreta el *opcode* (código de operación) de la instrucción en ejecución.
    - Genera señales para seleccionar la operación a realizar (e.g., activar sumador en lugar de comparador).
- **Entradas y Salidas**
  - Entradas:
    - Dos operandos (desde registros o memoria).
    - Señales de control (definen la operación a ejecutar).
  - Salidas:

- Resultado de la operación.
- Banderas (Flags): Indicadores de estado como:
  - Carry (acarreo): Desbordamiento en operaciones aritméticas.
  - Zero (cero): Resultado igual a cero.
  - Overflow: Desbordamiento en operaciones con signo.
  - Negative: Resultado negativo (bit más significativo).

Ejemplos de Opcode:

Operaciones Aritméticas:

Opcode (nombre)	**Código (ejemplo)**	**¿Qué hace?**	**Ejemplo en uso**
ADD	`0000`	Suma dos números	`5 + 3 = 8`
SUB	`0001`	Resta dos números	`5 - 3 = 2`
MUL	`0010`	Multiplica dos números	`5 * 3 = 15`
DIV	`0011`	Divide dos números	`15 / 3 = 5`
INC	`0100`	Incrementa en 1	`5 → 6`
DEC	`0101`	Decrementa en 1	`5 → 4`
NEG	`0110`	Negativo (cambia signo)	`5 → -5`

Operaciones Logicas (Booleanas):

Opcode	**Código**	**¿Qué hace?**	**Tabla de verdad**
AND	`1000`	Y lógico	`1 AND 1 = 1`, resto `0`
OR	`1001`	O lógico	`0 OR 0 = 0`, resto `1`
XOR	`1010`	O exclusivo	`1 XOR 1 = 0`, `1 XOR 0 = 1`
NOT	`1011`	Inversión	`NOT 1 = 0`, `NOT 0 = 1`

Entre otras operaciones.

Operación de la ALU

Flujo de Trabajo

1. **Fetch**: La CPU carga la instrucción desde memoria.
2. **Decode**: La unidad de control identifica el opcode y configura la ALU.
3. **Execute**:
    1. Los operandos se cargan en los registros de entrada de la ALU.
    2. La ALU procesa la operación y genera el resultado + flags.
4. **Writeback**: El resultado se almacena en un registro o memoria.

Ejemplo de Operación

- Instrucción: `ADD R1, R2, R3` (suma R2 + R3 y guarda en R1).
- La unidad de control envía señal "suma" a la ALU.
- La ALU recibe R2 y R3, realiza la suma, actualiza flags (e.g., carry si hay desbordamiento).
- El resultado se guarda en R1.

Aplicaciones de la ALU

- Procesamiento de datos: Cálculos matemáticos en programas.
- Toma de decisiones: Comparaciones (e.g., `if (a > b)`) mediante resta y flags.
- Manipulación de bits: Encriptación, compresión, gráficos.
- Ejecución de algoritmos: Desde operaciones básicas hasta IA y simulaciones.

Tendencias Modernas

- Pipelining: División de operaciones en etapas para paralelismo.
- Múltiples ALUs: Procesadores superscalares ejecutan varias operaciones simultáneas.
- Integración con FPU/GPUs: Unidades especializadas para operaciones en coma flotante o gráficos.

ALUs en Arquitecturas Actuales

- Procesadores x86 (Intel/AMD): Múltiples ALUs con soporte para SIMD (SSE, AVX).
- ARM: ALUs optimizadas para bajo consumo en dispositivos móviles.
- RISC-V: Diseño modular permitiendo ALUs personalizadas.

Futuro y Desafíos

- ALU Cuánticas: En investigación para realizar operaciones cuánticas (qubits).
- IA Integrada: Aceleradores de operaciones matriciales (e.g., TPUs de Google).
- Nanotecnología: ALUs en escala atómica para mayor eficiencia.

La Unidad Aritmético-Lógica (ALU) y la Unidad de Punto Flotante (FPU) son componentes esenciales de la CPU que ejecutan cálculos. La ALU maneja números enteros y operaciones lógicas binarias, mientras que la FPU gestiona números reales decimales complejos, cruciales para gráficos y ciencia. Ambas trabajan juntas para procesar instrucciones.

Unidad Aritmético-Lógica (ALU)

- *Función: Realiza operaciones aritméticas (suma, resta) y lógicas (AND, OR, NOT, XOR) a nivel de bit.*
- *Datos: Opera principalmente con números enteros, a menudo en complemento a dos.*
- *Funcionamiento: Recibe operandos de los registros, la instrucción de la unidad de control, y devuelve el resultado, generando indicadores (flags) como cero o desbordamiento.*

Unidad de Punto Flotante (FPU)

- *Función: Especializada en cálculos matemáticos de alta precisión con números reales (coma flotante), como los estándares IEEE 754.*
- *Datos: Maneja números con coma decimal, fundamentales para juegos 3D, multimedia y simulaciones científicas.*
- *Funcionamiento: Ajusta exponentes y mantisas antes de operar para gestionar grandes rangos dinámicos.*

Diferencias Principales

- *Complejidad: La FPU es más compleja y rápida para números reales que la ALU emulada por software.*
- *Especialización: La ALU es básica para el control del sistema; la FPU es para alto rendimiento matemático.*
- *Integración: Actualmente, la mayoría de los procesadores modernos integran la FPU dentro del núcleo de la CPU*

Bus de datos

Los parámetros más importantes de un *bus de datos* determinan su capacidad para transferir información entre componentes de un sistema computacional. Estos son los aspectos clave:

1. **Ancho del Bus (Bus Width)**
    1. Número de bits que puede transmitir simultáneamente.
    2. Ejemplo: Un bus de 64 bits transfiere 8 bytes en paralelo.
    3. La iImportancia radica que a Mayor ancho = Mayor rendimiento (más datos por ciclo).
    4. Limitado por el diseño físico del hardware (ej: buses en CPUs de 64 bits).

2. **Velocidad de Reloj (Clock Speed)**
    1. Frecuencia a la que el bus opera, medida en Hz (Megahertz, Gigahertz).
    2. Ejemplo: Un bus a 1600 MHz realiza 1.6 mil millones de ciclos por segundo.
    3. Importancia radica que a mayor frecuencia, más operaciones por segundo.
    4. Debe sincronizarse con los componentes conectados (ej: RAM y CPU).

3. **Ancho de Banda (Bandwidth)**
    1. Cantidad máxima de datos que el bus puede transferir por segundo.
    2. Cálculo:
       Ancho de banda = Ancho del bus / Velocidad de reloj
    3. Ejemplo: Un bus de 64 bits a 1600 MHz tiene un ancho de banda de 64 x 1600 x 10^6 = 12.8 GB/s
    4. Importancia: Define el límite teórico de transferencia (ej: PCIe 4.0 x16 = 31.5 GB/s).

4. **Latencia**
    1. Tiempo que tarda una operación en completarse desde que se inicia.
    2. Baja latencia = Respuesta rápida (crítico en aplicaciones en tiempo real).
    3. Alta latencia = Cuellos de botella (ej: acceso a memoria lento).

5. **Modo de Operación (Síncrono vs. Asíncrono)**
    1. Síncrono: Todos los dispositivos siguen un reloj común. Ejemplo: Buses en sistemas integrados.
    2. Asíncrono: Sin reloj global; usa señales de control (ej: handshaking). Ejemplo: USB.
    3. Importancia: La sincronización afecta la complejidad y la eficiencia.

6. **Protocolo de Comunicación**
    1. Reglas para enviar/recibir datos (formato, detección de errores, control de flujo).
    2. Ejemplos:
        1. PCIe: Paquetes con prioridades y corrección de errores.
        2. I²C: Protocolo maestro-esclavo para dispositivos embebidos.
        3. Bluetooth, WiFi entre otros.
    3. Importancia: Garantiza integridad y compatibilidad entre dispositivos.

7. **Escalabilidad**
    1. Capacidad de soportar más dispositivos o mayores velocidades sin degradar el rendimiento.
    2. Ejemplo: USB-C admite hasta 100 W de potencia y 40 Gbps.
    3. Importancia: Facilita actualizaciones y expansiones del sistema.

8. **Consumo de Energía**
    1. Potencia eléctrica requerida para operar el bus.
    2. Importancia: Clave en dispositivos móviles (ej: buses de bajo consumo en smartphones).
    3. Altos requisitos pueden generar calor y limitar el diseño.

9. **Topología del Bus**
    1. Cómo están conectados los dispositivos al bus.
    2. Tipos:
        1. Paralelo: Múltiples líneas físicas (ej: buses antiguos como ISA).
        2. Serial: Un solo canal de datos (ej: PCIe, SATA).
    3. Importancia: Los buses seriales modernos son más rápidos y menos propensos a interferencias.

10. **Compatibilidad y Estándares**
    1. Cumplimiento con normas industriales (ej: USB, PCIe, SATA).
    2. Importancia: Permite interoperabilidad entre fabricantes.
    3. Ejemplo: Un dispositivo USB 3.2 funciona en cualquier puerto USB.

Ejemplo Práctico:

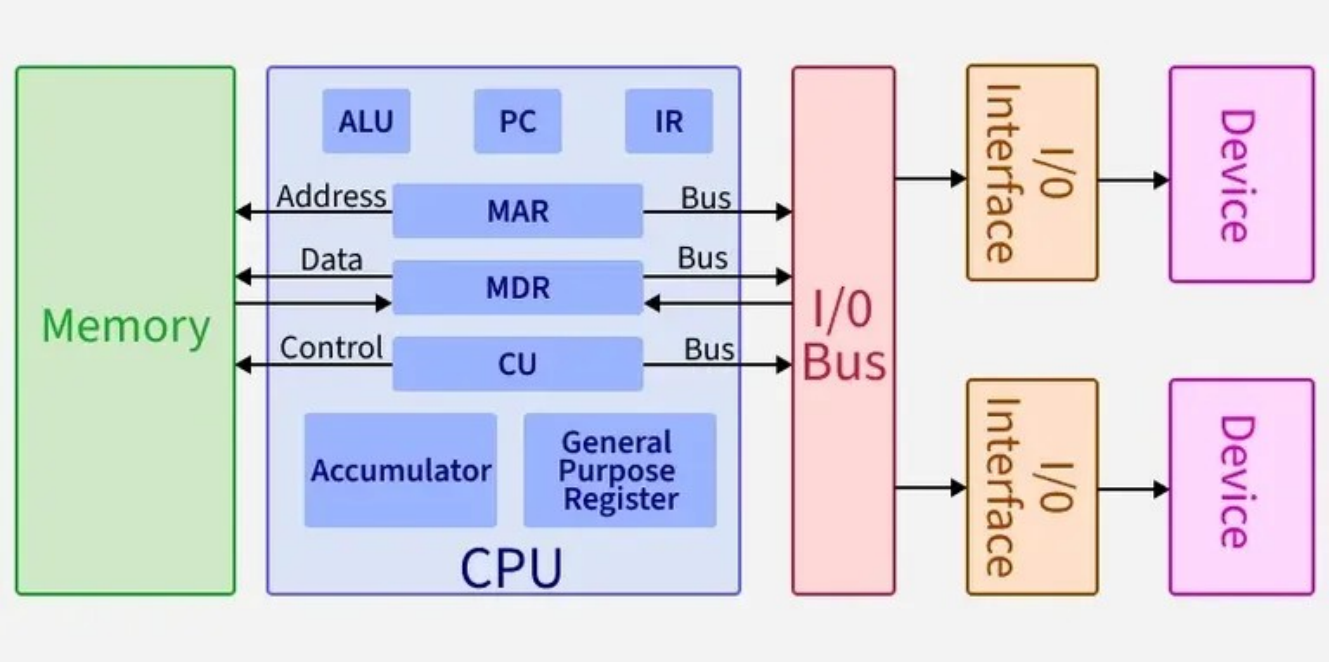
PCI Express (PCIe)

Parámetro	**PCIe 4.0 x16**
Ancho del bus	16 carriles (lanes)
Velocidad	16 GT/s (Gigatransfers/s)
Ancho de banda	31.5 GB/s por dirección
Protocolo	Paquetes con prioridad
Topología	Serial punto a punto

Los parámetros de un bus de datos determinan su eficiencia, velocidad y adaptabilidad. En sistemas modernos, el diseño óptimo equilibra ancho de banda, latencia y consumo energético, siguiendo estándares como PCIe o USB para garantizar compatibilidad. Estos aspectos son fundamentales en aplicaciones críticas como servidores, GPUs para IA o dispositivos IoT.

Arquitecturas de Memoria

Arquitectura de von Neumann



CU (Unidad de Control) : La unidad de control gestiona el funcionamiento del procesador mediante el envío de señales de control. Decide cómo deben moverse los datos dentro del ordenador, controla las operaciones de entrada y salida, y obtiene las instrucciones de la memoria para su ejecución.
ALU (Unidad Aritmético-Lógica)

La unidad aritmético-lógica: Es la parte de la CPU que se encarga de los cálculos y la toma de decisiones. Realiza operaciones aritméticas como la suma y la resta, operaciones lógicas como las comparaciones y tareas como el desplazamiento de bits en los datos.

Registros

Los registros son el tipo de memoria más rápido dentro de la CPU. Almacenan temporalmente la información con la que el procesador está trabajando, lo que hace que la ejecución de programas y las operaciones sean más rápidas y eficientes. Los registros funcionan como la memoria de trabajo principal de la CPU.

- PC (Contador de Programa) : Registra la dirección de la siguiente instrucción que se ejecutará.
- IR (Registro de Instrucciones) : Contiene la instrucción que se está ejecutando actualmente.
- MAR (Registro de Dirección de Memoria) : Almacena la dirección de la ubicación de memoria a la que se está accediendo.

- MDR (Registro de Datos de Memoria) : Almacena temporalmente los datos que se transfieren hacia o desde la memoria.
- Acumulador : Un registro que almacena resultados intermedios de operaciones aritméticas y lógicas.
- Registros de uso general : Se utilizan para el almacenamiento temporal de datos durante el procesamiento.

Bus de datos

El bus es un sistema de comunicación que transfiere datos, direcciones y señales de control entre la CPU, la memoria y los dispositivos de entrada/salida. En la arquitectura Von Neumann, se comparte un único bus tanto para datos como para instrucciones, lo que puede generar un cuello de botella (conocido como el cuello de botella de Von Neumann).

Bus de E/S

Interfaz de E/S : Conecta la CPU y la memoria a los dispositivos de entrada/salida.

Dispositivo: Se refiere a hardware externo como teclados, monitores o dispositivos de almacenamiento.

Características clave

- **Memoria única para datos e instrucciones** : Tanto los datos como las instrucciones del programa se almacenan en la misma memoria.
- **Bus compartido** : Se utiliza un único bus para transferir datos, direcciones y señales de control, lo que puede limitar el rendimiento.
- **Ejecución secuencial** : Las instrucciones se ejecutan una a una de forma secuencial.

Cuello de botella de Von Neumann

Por mucho que intentemos mejorar el rendimiento, no podemos ignorar que las instrucciones solo se pueden ejecutar de una en una y de forma secuencial. Ambos factores limitan la capacidad de la CPU.

Esto se conoce comúnmente como el *cuello de botella de Von Neumann*. Podemos dotar a un procesador Von Neumann de más caché, más RAM o componentes más rápidos, pero si queremos obtener mejoras significativas en el rendimiento de la CPU, es necesario un análisis exhaustivo de su configuración.

Aplicaciones

La arquitectura Von Neumann es la base de la mayoría de los sistemas informáticos modernos, donde tanto las instrucciones como los datos se almacenan en la misma memoria. A continuación, se presentan algunas aplicaciones:

- **Ordenadores personales y portátiles**: La mayoría de los ordenadores de sobremesa y portátiles (Intel/AMD) utilizan una memoria compartida que almacena tanto programas como datos, lo que permite un funcionamiento eficiente de software como Windows y los navegadores. Ejemplo: procesadores Intel Core i7 y AMD Ryzen.
- **Smartphones y tabletas**: Los dispositivos basados en ARM utilizan el modelo de memoria compartida para admitir la multitarea y la eficiencia energética en las aplicaciones de iOS y Android. Ejemplo: Qualcomm Snapdragon, chips de la serie A de Apple.
- **Sistemas embebidos:** Los microcontroladores en automóviles, dispositivos IoT y electrodomésticos utilizan la arquitectura Von Neumann para el procesamiento sencillo y rentable de instrucciones y datos en la misma memoria. Ejemplos: Arduino Uno (ATmega328), diseños ARM Cortex-M más antiguos.
- **Servidores y computación en la nube:** Los servidores utilizan sistemas de memoria compartida para gestionar tareas de software y datos a gran escala para servicios como AWS y Netflix.
- **Consolas de videojuegos:** Consolas como PS5 y Xbox utilizan memoria unificada para ejecutar el código del juego y gestionar las entradas, lo que permite una experiencia de juego fluida.

Tradicionalmente los sistemas con microprocesadores se basan en esta arquitectura, en la cual la unidad central de proceso (CPU), está conectada a una memoria principal única (casi siempre sólo RAM) donde se guardan las instrucciones del programa y los datos. A dicha memoria se accede a través de un sistema de buses único (control, direcciones y datos).

En un sistema con arquitectura Von Neumann el tamaño de la unidad de datos o instrucciones está fijado por el ancho del bus que comunica la memoria con la CPU. Así un microprocesador de 8 bits con un bus de 8 bits, tendrá que manejar datos e instrucciones de una o más unidades de 8 bits (bytes) de longitud. Si tiene que acceder a una instrucción o dato de más de un byte de longitud, tendrá que realizar más de un acceso a la memoria.

El tener un único bus hace que el microprocesador sea más lento en su respuesta, ya que no puede buscar en memoria una nueva instrucción mientras no finalicen las transferencias de datos de la instrucción anterior.

Las principales limitaciones que nos encontramos con la arquitectura Von Neumann son:

- La limitación de la longitud de las instrucciones por el bus de datos, que hace que el microprocesador tenga que realizar varios accesos a memoria para buscar instrucciones complejas.

- La limitación de la velocidad de operación a causa del bus único para datos e instrucciones que no deja acceder simultáneamente a unos y otras, lo cual impide superponer ambos tiempos de acceso.

La arquitectura Von Neumann describe a la computadora con 4 secciones principales: la unidad lógica y aritmética (ALU), la unidad de control, la memoria, y los dispositivos de entrada y salida (E/S).

Características

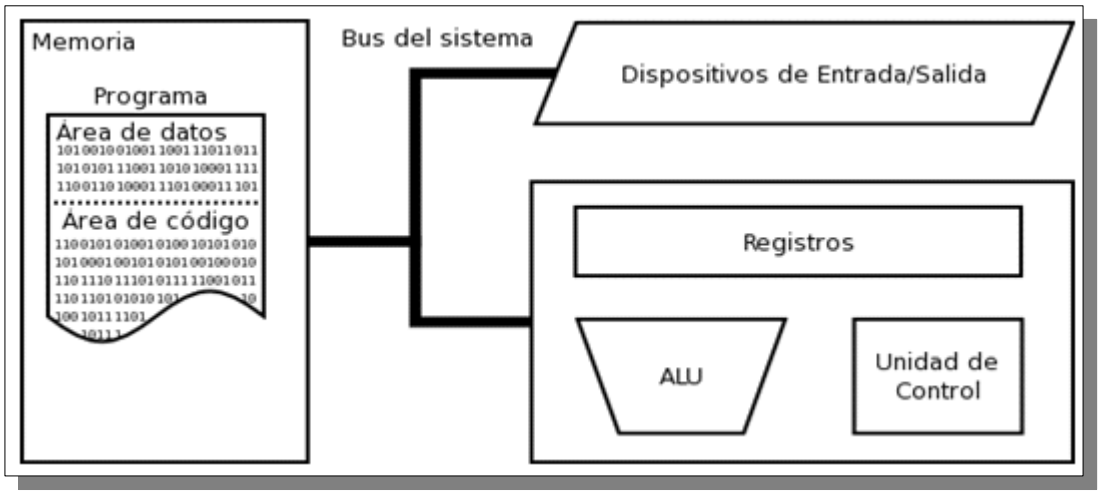
- **Memoria unificada**: Almacena datos y programas en la misma memoria.
- **Buses compartidos**: Un único bus para transferir instrucciones y datos.
- **Secuencialidad**: Las operaciones se ejecutan en serie (puede generar cuellos de botella).

Componentes

1. **Unidad Central de Proceso (CPU)**: Ejecuta instrucciones.
2. **Memoria Principal**: Almacena datos e instrucciones.
3. **Buses de Datos y Control**: Facilitan la comunicación.

Ejemplo: Computadoras personales, servidores (x86, ARM en modo general).

Los ordenadores con arquitectura Von Neumann constan de las siguientes partes:



La arquitectura Von Neumann realiza o emula los siguientes pasos secuencialmente:

1. Obtiene la siguiente instrucción desde la memoria en la dirección indicada por el contador de programa y la guarda en el registro de instrucción.
2. Aumenta el contador de programa en la longitud de la instrucción para apuntar a la siguiente.
3. Descodifica la instrucción mediante la unidad de control. Ésta se encarga de coordinar el resto de componentes del ordenador para realizar una función determinada.
4. Se ejecuta la instrucción. Ésta puede cambiar el valor del contador del programa, permitiendo así operaciones repetitivas.
5. Regresa al paso N° 1.

La mayoría de las computadoras todavía utilizan la arquitectura Von Neumann, propuesta a principios de los años 40 por John Von Neumann.

https://www.geeksforgeeks.org/computer-organization-architecture/computer-organization-von-neumann-architecture/

Arquitectura Harvard

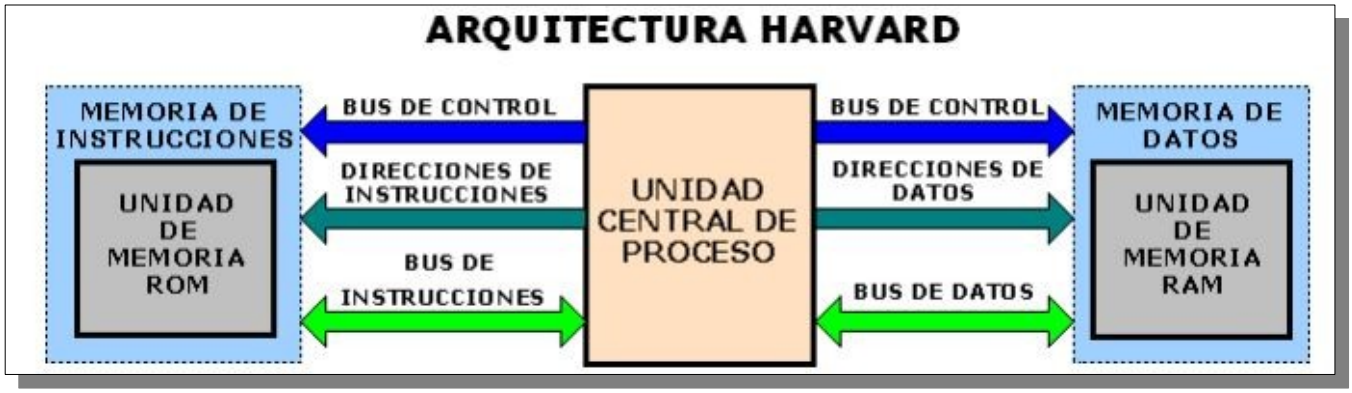
Este modelo, que utilizan los Microcontroladores PIC (**Peripheral Interface Controller** - Controlador de Interfaz Periférico), tiene la unidad central de proceso (CPU) conectada a dos memorias (una con las instrucciones y otra con los datos) por medio de dos buses diferentes.

*PIC (**Peripheral Interface Controller** - Controlador de Interfaz Periférico) es un chip que controlaba periféricos (teclados, displays, sensores, motores) conectados a una computadora más grande o sistema embebido.*

Una de las memorias contiene solamente las instrucciones del programa (Memoria de Programa o Memoria de Instrucciones), y la otra sólo almacena datos (Memoria de Datos). La memoria de programa o de instrucciones lo diferencia de la arquitectura Vonn Neumann.

Ambos buses son totalmente independientes lo que permite que la CPU pueda acceder de forma independiente y simultánea a la memoria de datos y a la de instrucciones. Como los buses son independientes estos pueden tener distintos contenidos en la misma dirección y también distinta longitud.

También la longitud de los datos y las instrucciones puede ser distinta, lo que optimiza el uso de la memoria en general.



Para un procesador de Set de Instrucciones Reducido, o RISC (Reduced Instrucción Set Computer), el set de instrucciones y el bus de memoria de programa pueden diseñarse de tal manera que todas las instrucciones tengan una sola posición de memoria de programa de longitud.

Además, al ser los buses independientes, la CPU puede acceder a los datos para completar la ejecución de una instrucción, y al mismo tiempo leer la siguiente instrucción a ejecutar.

Ventajas de esta arquitectura:

- El tamaño de las instrucciones no está relacionado con el de los datos, y por lo tanto puede ser optimizado para que cualquier instrucción ocupe una sola posición de memoria de programa, logrando así mayor velocidad y menor longitud de programa.
- El tiempo de acceso a las instrucciones puede superponerse con el de los datos, logrando una mayor velocidad en cada operación.

Recibe el nombre por el ordenador Harvard Mark I, desarrollado en esa universidad de Massachussetts por Howard Aiken

Características

- **Memorias separadas**: Una para datos y otra para instrucciones.
- **Buses independientes**: Permiten acceso simultáneo a datos y código.
- **Paralelismo**: Mayor velocidad en aplicaciones específicas.

Componentes

1. **CPU**: Con buses separados para datos e instrucciones.
2. **Memoria de Instrucciones**: Almacena el programa.
3. **Memoria de Datos**: Almacena variables y resultados.

Ejemplo: Sistemas embebidos (Microcontroladores PIC, Arduino), DSPs. (digital signal processor)

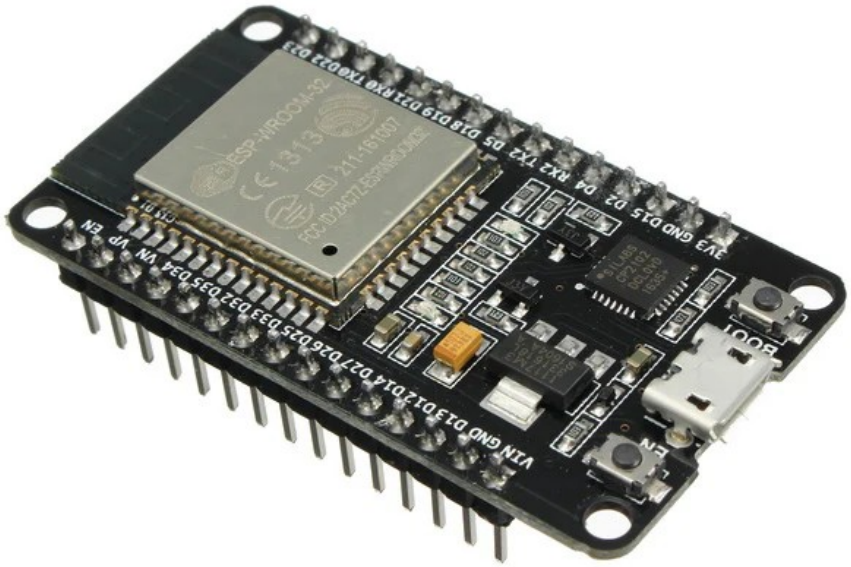
Cuadro Comparativo

Característica	**Arquitectura Von Neumann**	**Arquitectura Harvard**
Memoria	Unificada (datos + instrucciones)	Separada (datos ≠ instrucciones)
Buses	Único bus para datos/instrucciones	Buses independientes
Rendimiento	Cuellos de botella por secuencialidad	Mayor paralelismo
Complejidad/Costo	Menor complejidad y costo	Mayor complejidad y costo
Aplicaciones Típicas	Computadoras generales	Sistemas embebidos, tiempo real

Característica	Arquitectura Von Neumann	Arquitectura Harvard
Ejemplos	Intel Core, AMD Ryzen, ARM	Microcontroladores PIC, Arduino

Ejemplos Modernos

Arquitectura	Aplicación	Ejemplo Concreto
von Neumann	Computación general	Laptops (Procesadores Intel Core, RYZEN AMD)
Harvard	Sistemas embebidos	Arduino Uno, ESP32
Paralela (SIMD)	Procesamiento gráfico/IA	NVIDIA GeForce RTX 4090
Multicore (MIMD)	Servidores y supercomputadoras	AMD EPYC, IBM POWER10



ESP32

https://www.geeksforgeeks.org/computer-organization-architecture/harvard-architecture/

Arquitecturas Híbridas y Evoluciones Modernas

Muchos sistemas actuales combinan ambas arquitecturas:

- Arquitectura Harvard Modificada: Usada en procesadores ARM (cachés separados para datos/instrucciones, pero memoria principal unificada).
- GPUs: Utilizan buses paralelos para alta velocidad en procesamiento gráfico.

Arquitectura Harvard Modificada (Modified Harvard)

La solución híbrida que usan prácticamente todos los sistemas actuales:

- Nivel de sistema (Von Neumann):
  - El sistema operativo y programas ven un espacio de memoria unificado
  - Permite flexibilidad: código puede tratarse como datos (compiladores JIT, sistemas operativos)
- Nivel de CPU (Harvard):
  - La caché L1 se divide físicamente en:
    - L1 Instruction Cache (I-Cache): Solo almacena instrucciones
    - L1 Data Cache (D-Cache): Solo almacena datos
  - Permite acceso simultáneo a instrucciones y datos (paralelismo)
  - Elimina el cuello de botella de Von Neumann a nivel del procesador
- Ventaja: Combina la flexibilidad de Von Neumann con la velocidad de Harvard

Arquitecturas NUMA (Non-Uniform Memory Access)

- Cada procesador/core tiene su propia memoria local
- Acceso a memoria remota (de otro procesador) es más lento que acceso local
- Usada en servidores multicore y sistemas de alto rendimiento
- Requiere gestión sofisticada por el sistema operativo

ARQUITECTURAS ESPECIALIZADAS

Arquitecturas VLIW (Very Long Instruction Word)

- Concepto: El compilador agrupa múltiples operaciones independientes en una "palabra de instrucción" muy larga
- Características:
  - Paralelismo explícito definido por software (compilador), no por hardware
  - Hardware más simple (no necesita detectar dependencias en tiempo de ejecución)
  - Instrucciones de longitud fija pero muy anchas
  - Desafío: Requiere compiladores extremadamente sofisticados; binarios dependen de la implementación específica

Arquitecturas EPIC (Explicitly Parallel Instruction Computing)

- Evolución de VLIW desarrollada por Intel y HP
- Características:
  - Especificación explícita de paralelismo en el código máquina
  - Ejecución predicada (instrucciones condicionales sin saltos)
  - Carga especulativa de datos
  - Registros rotativos para optimización de loops
- Implementación: Intel Itanium (IA-64)
- Lección aprendida: Dificultad de extraer paralelismo de código general llevó a su discontinuación

ARQUITECTURAS HETEROGÉNEAS Y SoC

System on a Chip (SoC)

- Integración de múltiples componentes en un único chip:
  - CPU (generalmente multicore)
  - GPU (procesamiento gráfico y paralelo)

- NPU (Neural Processing Unit para IA)
- Controladores de memoria, E/S, conectividad

- Ejemplos: Apple Silicon (M1/M2/M3/M4), Qualcomm Snapdragon, Samsung Exynos

Arquitecturas big.LITTLE (ARM) / Hybrid (Intel)

- Combinación de núcleos de diferente potencia en el mismo chip:
  - Núcleos "big": Alto rendimiento, mayor consumo (tareas exigentes)
  - Núcleos "LITTLE": Eficiencia energética, menor rendimiento (tareas simples)
  - El sistema operativo migra tareas dinámicamente entre tipos de núcleos
- Implementaciones:
  - ARM: big.LITTLE (desde 2011)
  - Intel: P-cores (Performance) + E-cores (Efficient) desde 12ª generación

CUADRO COMPARATIVO: EVOLUCIÓN DE ARQUITECTURAS

Arquitectura	**Época**	**Característica Principal**	**Limitación Superada**	**Uso Actual**
Von Neumann	1945-presente	Memoria unificada, bus único	Simplicidad de diseño	Sistemas de propósito general
Harvard	1940s-presente	Memorias separadas, buses independientes	Cuello de botella de Von Neumann	Microcontroladores, DSPs, sistemas embebidos
Harvard Modificada	1990s-presente	Caché L1 separada, memoria unificada	Inflexibilidad de Harvard pura	Todos los procesadores modernos
SIMD	1970s-presente	Procesamiento vectorial paralelo	Bajo rendimiento en tareas homogéneas	Gráficos, IA, multimedia
MIMD	1980s-presente	Múltiples procesadores independientes	Limitación de un solo flujo de ejecución	Servidores, supercomputadoras

NUMA	1990s-presente	Memoria no uniforme, escalabilidad	Limitaciones de memoria compartida	Servidores multicore, HPC
VLIW/EPIC	1990s-2000s	Paralelismo explícito por compilador	Complejidad de detección de paralelismo en hardware	DSPs, nichos específicos
SoC Heterogéneo	2010s-presente	Integración CPU+GPU+NPU+controladores	Consumo energético, tamaño físico	Móviles, laptops, IoT

CONCLUSIÓN: LA REALIDAD ACTUAL

Ningún sistema moderno usa puramente Von Neumann o Harvard. La arquitectura dominante es el híbrido:

"Harvard por dentro, Von Neumann por fuera"

Por fuera: El sistema operativo y los programas ven una memoria unificada (Von Neumann)

Por dentro: El procesador usa cachés separadas y técnicas de paralelismo (Harvard, SIMD, MIMD)

Esta convergencia permite:

- Compatibilidad con software existente
- Rendimiento optimizado mediante paralelismo
- Eficiencia energética en dispositivos móviles
- Escalabilidad en servidores y supercomputadoras

El procesamiento vectorial es la evolución natural que resuelve la limitación fundamental de Von Neumann:

Problema Von Neumann	**Solución del procesamiento vectorial**
Un bus, un dato a la vez	Múltiples datos procesados en paralelo
CPU inactiva esperando memoria	Unidades vectoriales mantienen el pipeline ocupado
Instrucciones secuenciales	Una instrucción vectorial reemplaza muchas escalares

- SISD (Taxonomía Flynn) = Procesamiento escalar
- SIMD (Taxonomía Flynn) = Procesamiento vectorial

Nota clave: Las CPUs modernas tienen ambas unidades: ALU escalar para trabajo general + Unidad Vectorial para procesamiento masivo de datos.

Herramientas:

- **Profiler de procesadores**: es una herramienta que analiza el uso de la CPU de una aplicación para identificar qué funciones consumen más tiempo y cómo se asignan los recursos. https://es.wikipedia.org/wiki/An%C3%A1lisis_de_rendimiento_de_software
- **Profiler de memoria**: Herramienta que mide el uso de la memoria de una aplicación y detecta fugas de memoria.
- **Profiler de E/S**: Herramienta que mide el tiempo que tarda una aplicación en realizar operaciones de entrada y salida.
- **Profiler de red**: Herramienta que mide el tiempo que tarda una aplicación en enviar y recibir datos a través de una red.
- **Profiler de rendimiento web:** Herramienta que mide el tiempo de carga de una página web y detecta cuellos de botella en el código.

https://elchapuzasinformatico.com/2024/03/intel-continuous-profiler-aumentar-rendimiento-cpu/

https://newsroom.intel.com/software/intel-releases-continuous-profiler-for-cpu-performance

https://granulate.io/continuous-profiling/

https://github.com/intel/gprofiler-performance-studio

Leer:

https://es.wikipedia.org/wiki/Ley_de_Moore

https://www.unir.net/revista/ingenieria/computacion-paralela/

Referencias

- Patterson, D. A., & Hennessy, J. L. (2017). *Computer Organization and Design
- IEEE Standard for Binary Floating-Point Arithmetic (IEEE 754).
- Tanenbaum, A. S. (2016). *Structured Computer Organization
- Hennessy, J. & Patterson, D. *Computer Architecture: A Quantitative Approach
- Tanenbaum, A. *Structured Computer Organization
- Documentación oficial de NVIDIA CUDA y Arduino.

Anexo:

intrínsecos

Los intrínsecos (del inglés intrinsics) son funciones especiales proporcionadas por el compilador que actúan como interfaz entre el código de alto nivel (C/C++) y las instrucciones de ensamblador de bajo nivel.

En esencia: Son "funciones mágicas" que el compilador traduce directamente a instrucciones de máquina específicas del procesador, sin necesidad de escribir ensamblador manualmente.

¿Por qué se llaman "intrínsecos"?

El término proviene de que estas funciones son intrínsecas al compilador — es decir, el compilador las conoce internamente y sabe exactamente cómo traducirlas a código máquina. No son funciones de biblioteca externa, sino extensiones propias del compilador.