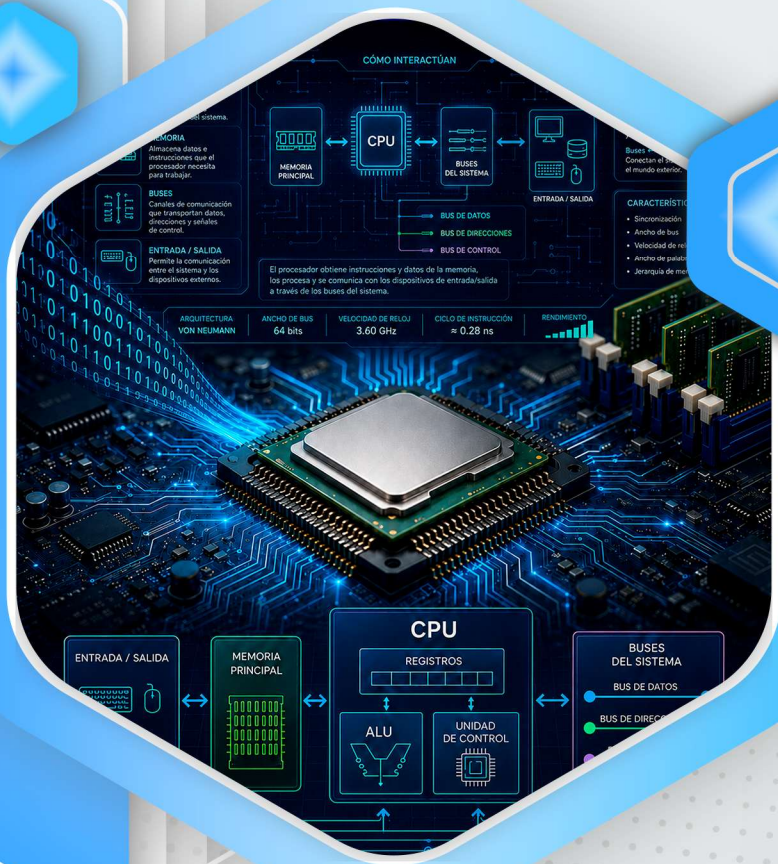
1era Ed.
2026

PUERTO MADERO EDITORIAL

Escuela
Superior Politécnica
de Chimborazo

ARQUITECTURA DE COMPUTADORES:
UN ENFOQUE PRÁCTICO

DIEGO FERNANDO AVILA PESANTEZ



EDITORES:
JUAN CARLOS SANTILLÁN LIMA
SANDRA ISABEL GONZÁLEZ POLO

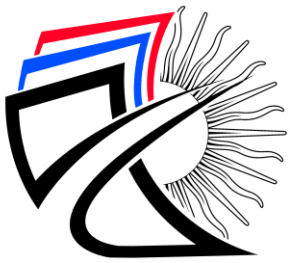
puertomaderoeditorial.com.ar
La Plata - Argentina

ARQUITECTURA DE COMPUTADORES: UN ENFOQUE PRÁCTICO

AUTORES: Diego Fernando Avila Pesantez
ISBN: 978-631-6557-81-0

ISBN: 978-631-6557-81-0

ISBN: 978-631-6557-81-0



PUERTO MADERO
EDITORIAL

ARQUITECTURA DE COMPUTADORES: UN ENFOQUE PRÁCTICO

COMPUTER ARCHITECTURE: A PRACTICAL APPROACH

AUTOR:
Diego Fernando Avila Pesantez

ISBN: 978-631-6557-81-0

Diego Fernando Avila Pesantez
Arquitectura de computadores: un enfoque práctico / Diego Fernando Avila Pesantez; Editado por Juan Carlos Santillán Lima ; Sandra Isabel González Polo. - 1a ed. - La Plata : Puerto Madero Editorial Académica, 2026.
Libro digital, PDF

Archivo Digital: descarga y online
ISBN 978-631-6557-81-0

1. Computación. I. Santillán Lima, Juan Carlos, ed. II. Sandra Isabel, González Polo, ed. III. Título.
CDD 004.071

ISBN: 978-631-6557-81-0

ISBN: 978-631-6557-81-0

DEDICATORIA

A la **Escuela Superior Politécnica de Chimborazo** (ESPOCH), por ser el lugar donde los conocimientos se construyen día a día y se transforma la sociedad.

A mis hijos, **Jennifer y Daniel**, mi mayor motivación y orgullo, quienes, con su alegría, amor y apoyo, dan sentido a cada meta alcanzada.

A mi **madre Milita**, que hoy me acompaña desde el cielo. Gracias por su amor y por las enseñanzas que guían cada paso de mi vida.

A mi **padre Victitor**, por su fortaleza, su apoyo incondicional, por enseñarme que la vida es como una escalera, que siempre tiene un peldaño para crecer como persona y cumplir los sueños.

ISBN: 978-631-6557-81-0

Primera Edición, Mayi 2026

TITULO: ARQUITECTURA DE COMPUTADORES: UN ENFOQUE PRÁCTICO
ISBN: 978-631-6557-81-0
DOI: https://doi.org/10.55204/pmea.133

Editado por:
Sello editorial: ©Puerto Madero Editorial Académica
Nº de Alta: 933832

Editorial: © Puerto Madero Editorial Académica
CUIL: 20630333971
Calle 45 N491 entre 4 y 5
Dirección de Publicaciones Científicas Puerto Madero Editorial Académica
La Plata, Buenos Aires, Argentina
Teléfono: +54 9 221 314 5902
+54 9 221 531 5142
Código Postal: AR1900

Este libro se sometió a arbitraje bajo el sistema de doble ciego (peer review)

Corrección y diseño: Puerto Madero Editorial Académica
Diseñador Gráfico: José Luis Santillán Lima

Diseño, Montaje y Producción Editorial:
Puerto Madero Editorial Académica
Diseñador Gráfico: Santillán Lima, José Luis

Director del equipo editorial: Santillán Lima, Juan Carlos

Editor: Santillán Lima, Juan Carlos
González Polo, Sandra Isabel

Hecho en Argentina
Made in Argentina

ISBN: 978-631-6557-81-0

ISBN: 978-631-6557-81-0

AUTOR:

Diego Fernando Avila Pesantez

Escuela Superior Politécnica de Chimborazo, Facultad de Informática y Electrónica, Grupo de Investigación e Innovación Científica y Tecnológica **(GIITYC)**, Panamericana Sur km 1 1/2, EC06022, Riobamba, Chimborazo, Ecuador

davila@espoch.edu.ec

https://orcid.org/0000-0001-8394-5621

ISBN: 978-631-6557-81-0

ISBN: 978-631-6557-81-0

CONTENIDO

RESUMEN _____ *1*

ABSTRACT _____ *2*

INTRODUCCIÓN _____ *3*

CAPÍTULO 1

1 FUNDAMENTOS A LA ARQUITECTURA DE COMPUTADORES _____ *5*

1.1 Definición _____ **5**

1.2 Organización y arquitectura _____ **5**

1.3 Fuerzas que actúan sobre la Arquitectura de Computadores _____ **7**

1.4 La evolución de los equipos informáticos automatizados _____ **9**

1.5 Arquitectura de Von Neumann _____ **12**

1.6 Arquitectura de Harvard _____ **15**

1.7 Clasificación de los computadores por su capacidad de procesamiento _____ **16**

1.7.1 Supercomputadores _____ 17

1.7.2 Mainframes _____ 18

1.7.3 Minicomputadores _____ 20

1.7.4 Microcomputadores _____ 21

1.8 Ley de Moore _____ **22**

1.9 Representación numérica de la información _____ **23**

1.9.1 Representación numérica de la información _____ 24

1.9.2 Representación numérica de enteros _____ 28

CAPÍTULO 2

2 ORGANIZACIÓN FUNCIONAL _____ *35*

2.1 Estructura de un sistema de cómputo _____ **35**

2.1.1 Mainboard o placa base _____ 37

2.1.2 Carcasa o Case _____ 39

2.1.3 Fuente de poder _____ 42

2.1.4 Socket y CPU _____ 45

ISBN: 978-631-6557-81-0

2.1.5 Sistema de enfriamiento (cooling systems) 47
2.1.6 Memorias 48
2.1.7 Chipset 58
2.1.8 Buses de conexión 61
2.1.9 Slot de expansión 63
2.1.10 Almacenamiento 65
2.1.11 Tarjeta gráfica 73
2.1.12 Interface de Red 75
2.1.13 Dispositivos de entrada y salida 76

CAPÍTULO 3.

3 Microprocesador 79

3.1 Microprocesadores y controladores 79

3.2 Estructura y función del microprocesador 79

3.2.1 Ciclo de ejecución de las instrucciones 83
3.2.2 Registros 86
3.2.3 Unidad Aritmético-Lógica (ALU) 89
3.2.4 Unidad de control 90

3.3 Evolución de los microprocesadores 94

3.4 Arquitecturas CISC/RISC 101

3.5 Arquitecturas contemporáneas 102

CAPÍTULO 4.

4 Lenguaje Ensamblador 107

4.1 Elementos básicos del lenguaje ensamblador 107

4.2 Arquitectura del procesador x86 108

4.3 Formato general de las instrucciones 110

4.4 Definiendo los datos 113

4.5 Instrucciones Aritméticas y Operaciones 115

4.6 Directivas 117

4.7 Desplazamiento 119

4.8 Saltos condicionales e incondicionales 119

4.9 Bucles 120

4.10 Ejemplos de programación 123

ISBN: 978-631-6557-81-0

BIBLIOGRAFÍA _____ ***131***

AUTOR _____ ***135***

DIEGO FERNANDO AVILA PESANTEZ _____ ***135***

ISBN: 978-631-6557-81-0

ISBN: 978-631-6557-81-0

ÍNDICE DE FIGURAS

Figura 1.1 Relación entre la arquitectura del computador y la organización funcional 7
Figura 1.2 Fuerzas externas que permiten el avance de la arquitectura 8
Figura 1.3 Componentes de la arquitectura de Neumann. 14
Figura 1.4 Base de la arquitectura de Harvard 15
Figura 1.5 Supercomputador “El Capitan” 18
Figura 1.6 Mainframe de IBM z16 20
Figura 1.7 Equipo IBM AS/400, ampliamente utilizado en entornos empresariales. 21
Figura 1.8 PC gaming 22
Figura 1.9 Evolución de los sistemas de cómputo enfocados a operaciones por segundo 23
Figura 2.1 Modelo general de la arquitectura de una computadora 36
Figura 2.2. a) Mainboard ASUS PRIME H810M-K-CSM 38
Figura 2.3 Case de PC Gaming 41
Figura 2.4 Fuente de poder de PC y laptop, respectivamente. 43
Figura 2.5 Principales conectores de la fuente de poder 44
Figura 2.6 a) CPU para socket PGA; b) socket LGA para procesador Intel. 45
Figura 2.7 Sistemas de enfriamiento 48
Figura 2.8 Organización jerárquica de la memoria y los dispositivos de almacenamiento. 49
Figura 2.9 Organización básica de celdas de memoria DRAM y SRAM 51
Figura 2.10 Módulos de memoria RAM. 53
Figura 2.11 DIMM y SODIMM 54
Figura 2.12 Uso de CPU-Z para visualizar los niveles de la memoria caché de un laptop. 56
Figura 2.13 Formas de realizar el Clear CMOS 58
Figura 2.14 Modelo clásico de chipset con Northbridge y Southbridge. 59
Figura 2.15 Estructura del chipset Intel 800 Series y flujo de comunicación. 60
Figura 2.16 Líneas de comunicación interna de un sistema de cómputo 62
Figura 2.17 Componentes internos de un HDD 65
Figura 2.18 Conectores para discos duros. 67
Figura 2.19 Form factor de los SSD 68
Figura 2.20 Tamaños de SSD con interface M.2 NVMe 69

ISBN: 978-631-6557-81-0

Figura 2.21 Sistemas RAID para discos. 72
Figura 2.22 Componentes principales de la tarjeta gráfica. 73
Figura 2.23 Principales interfaces de conexión de video. 75
Figura 2.24 Elementos de interconexión de red. 76
Figura 3.1 Arquitectura interna del CPU y sus bloques funcionales. 80
Figura 3.2 Bloques funcionales del procesador actual. 83
Figura 3.3 Ciclo de ejecución de una instrucción. 85
Figura 3.4 Segmentación de instrucciones 86
Figura 3.5 Esquema del uso de registros de acceso a memoria 88
Figura 3.6 Esquema lógico de los registros internos del CPU 89
Figura 3.7 La ALU y conexiones internas. 91
Figura 3.8 Lógica del trabajo de la unidad de control. 92
Figura 3.9 Descripción de las características clave de CISC y RISC. 103
Figura 4.1 Registros de memoria en arquitectura x86 109

ISBN: 978-631-6557-81-0

ÍNDICE DE TABLAS

Tabla 1.1 Evolución histórica y avances tecnológicos. 13
Tabla 1.2 Top 5 de supercomputadores en noviembre de 2025. 19
Tabla 2.1 Principales form factor desarrollados en la industria 41
Tabla 2.2 Principales criterios para seleccionar un case de computador. 42
Tabla 2.3 Tipos de fuente de poder según el equipo. 43
Tabla 2.4 Desarrollo y cambios en los sockets de procesadores 46
Tabla 2.5 Características y clasificación de los sistemas de memoria en un sistema de cómputo 50
Tabla 2.6 Diferencias clave entre DRAM y SRAM. 51
Tabla 2.7 Detalles de las Memorias DDR 53
Tabla 2.8 Evolución de los Slots de Expansión del Computador. 64
Tabla 2.9 Interfaces de discos duros y unidades de almacenamiento. 69
Tabla 2.10 Evolución histórica y características técnicas principales de la memoria gráfica GDDR. 74
Tabla 2.11 Características de las interfaces de video. 75
Tabla 3.1 Registros de propósito general de la arquitectura x86-64 87
Tabla 3.2 Componentes y características destacadas de los procesadores Intel 98
Tabla 3.3 Descripción cronológica breve de los procesadores de Intel y AMD 101
Tabla 4.1 Tipos de datos intrínsecos. Elaborada por el autor. 114

ISBN: 978-631-6557-81-0

ISBN: 978-631-6557-81-0

RESUMEN

Este libro presenta los fundamentos de la arquitectura de computadores, dirigido a estudiantes universitarios que necesitan comprender el funcionamiento del hardware de un sistema informático. El primer capítulo introduce los conceptos básicos de arquitectura y organización de computadores, analiza las fuerzas que influyen en su evolución, revisa el desarrollo histórico de los sistemas de cómputo y explica las arquitecturas de Von Neumann y de Harvard. Como cierre, se incluye la representación numérica de la información, esencial para comprender cómo se tratan los datos internamente en el computador. El segundo capítulo aborda la organización funcional de un sistema de cómputo, explicando de manera detallada los principales componentes físicos, como la placa base, el procesador, las memorias, el almacenamiento, los buses y los dispositivos de entrada/salida. Con ello, el estudiante puede identificar el rol de cada componente y su relación con el rendimiento del sistema computacional. Luego, en el capítulo 3 se estudia el microprocesador, incluyendo la estructura interna, el ciclo de ejecución de instrucciones, los registros, la ALU y la unidad de control, las arquitecturas CISC y RISC, así como las tendencias actuales. Finalmente, en el último capítulo se introduce el lenguaje ensamblador, destacando su relación directa con la arquitectura del procesador, especialmente en x86, y se explican las instrucciones básicas, saltos, bucles y se incluyen ejemplos prácticos que vinculan el hardware con el software de bajo nivel.

Palabras clave: Arquitectura de computadores, organización funcional, microprocesador, lenguaje ensamblador, sistema de cómputo.

ISBN: 978-631-6557-81-0

ABSTRACT

This book presents the fundamentals of computer architecture and is intended for college students who need to understand how computer hardware works. The first chapter introduces the basic concepts of computer architecture and organization, analyzes the forces that influence their evolution, reviews the historical development of computer systems, and explains the Von Neumann and Harvard architectures. Finally, it covers the numerical representation of information, which is essential for understanding how data is processed internally within the computer. The second chapter addresses the functional organization of a computer system, providing a detailed explanation of the main physical components, such as the motherboard, the processor, memory, storage, buses, and input devices. This enables students to identify the role of each component and its relationship to the performance of the computer system. Chapter 3 then examines the microprocessor, including its internal structure, the instruction execution cycle, registers, the ALU and control unit, CISC and RISC architectures, as well as current trends. Finally, the last chapter introduces assembly language, highlighting its direct relationship with processor architecture, especially in x86, and explains basic instructions, jumps, and loops, while including practical examples that link hardware with low-level software.

Keywords: Computer architecture, functional organization, microprocessor, assembly language, computer system.

ISBN: 978-631-6557-81-0

INTRODUCCIÓN

El libro de Arquitectura de Computadores: Un enfoque práctico está diseñado para que los estudiantes de la carrera de software comprendan de manera clara y práctica cómo funcionan los sistemas de cómputo desde su interior. En su contenido se abordan los conceptos esenciales de la arquitectura y la organización de computadores, comenzando por sus fundamentos teóricos y avanzando progresivamente hacia aspectos más concretos como los componentes físicos, el microprocesador y el lenguaje ensamblador. La intención no es hacerlo demasiado teórico ni complejo, sino ofrecer una guía accesible que permita fortalecer la base técnica del estudiante y facilitar la comprensión del funcionamiento interno de los computadores, de modo que puedan aplicar estos conocimientos en su formación académica y en su desarrollo profesional.

El capítulo 1, Fundamentos de la Arquitectura de Computadores, constituye el pilar teórico de este libro, abordando definiciones clave de organización y arquitectura de computadores. Mediante un análisis, se explican las arquitecturas clásicas de Von Neumann y de Harvard, la clasificación de los computadores según su capacidad de procesamiento, la ley de Moore y se ofrece una visión general de la evolución de los equipos informáticos hasta la actualidad. El capítulo finaliza con el tema de representación numérica de la información para una comprensión sobre operaciones básicas a nivel binario y de punto.

En el capítulo 2, denominado Organización Funcional, se analiza en detalle la estructura de un sistema de cómputo desde una perspectiva práctica. A lo largo de este capítulo, se explican los componentes principales de una computadora, como la placa base, el chipset, el procesador, las memorias, los buses, el almacenamiento y los dispositivos de entrada y salida. Este contenido permite que el estudiante identifique cada componente, comprenda su función y entienda cómo interactúan con otros elementos del equipo computacional.

ISBN: 978-631-6557-81-0

El capítulo 3 detalla el funcionamiento del microprocesador, se sumerge en el cerebro del computador y analiza la estructura y el funcionamiento interno, el ciclo de ejecución de las instrucciones, los registros, la ALU y la unidad de control. Además, se detallan los componentes esenciales de la CPU: la Unidad de Control, los registros y la Unidad Aritmético-Lógica (ALU). Mediante el estudio del ciclo de ejecución de instrucciones, el lector comprenderá la dinámica interna del procesamiento de datos. Por último, se revisa la evolución de los microprocesadores y las arquitecturas CISC y RISC, así como, algunas arquitecturas contemporáneas, clave para entender los procesadores modernos.

Finalmente, el capítulo 4, Lenguaje Ensamblador, aborda una introducción a la programación de bajo nivel, permitiendo interactuar directamente con el hardware. Se explican los elementos básicos del lenguaje ensamblador, la arquitectura del procesador x86, el formato de las instrucciones y las principales operaciones, como el manejo de registros, la aritmética, los saltos y los bucles. Finalmente, la unidad integra ejemplos prácticos, proporcionando una comprensión técnica sobre cómo se ejecutan las instrucciones a nivel de registros, memoria y el código que interactúa con los recursos del sistema.

ISBN: 978-631-6557-81-0

CAPÍTULO 1

1 FUNDAMENTOS A LA ARQUITECTURA DE COMPUTADORES

La arquitectura de computadores es una asignatura esencial dentro de la informática, ya que analiza los principios de diseño y organización, funcionamiento de los sistemas que procesan datos, lo que permite entender cómo el hardware trabaja junto con los programas e influye en la velocidad de procesamiento. Con la evolución de los computadores, sus componentes y estructuras se pueden procesar información más rápido, almacenar grandes cantidades de datos y ofrecer sistemas más confiables.

1.1 Definición

La arquitectura de computadores es el conjunto de atributos de un sistema informático, visibles al programador y que tienen un impacto directo en la ejecución lógica de un programa, tales como el conjunto de instrucciones, los tipos de datos, los modos de direccionamiento y la organización de la memoria (Tanenbaum, 2016) . Según Hennessy and Patterson (2017), hablar de arquitectura de computadores es señalar un puente que conecta el hardware y el software, describiendo la estructura funcional y el comportamiento del computador, incluyendo el conjunto de instrucciones, las técnicas de diseño y los principios de rendimiento. Desde otra perspectiva, Stallings (2019) define la arquitectura de computadores como los atributos del sistema computacional para la ejecución lógica del programa, y se distinguen de la organización interna del hardware, que incluye el diseño del procesador, la estructura de instrucciones, las memorias y los mecanismos de control.

1.2 Organización y arquitectura

La arquitectura y la organización de computadores pueden verse como dos miradas distintas, pero se complementan para entender cómo funciona un computador actual (Yadin, 2016). La manera en que un sistema de cómputo se diseña y cómo se estructuran

ISBN: 978-631-6557-81-0

sus partes ayudan a entender cómo funciona. La arquitectura de computadores se refiere a los atributos del sistema visibles para el programador al ejecutar los programas. Incluye el conjunto de instrucciones (ISA), los tipos de datos soportados, los modos de direccionamiento, el manejo de excepciones, la estructura lógica del procesador y, en general, todo aquello que define la interfaz entre el hardware y el software (Englander & Wong, 2021). Desde esta perspectiva, la arquitectura establece el "qué hace" del computador: cómo interpreta instrucciones, qué operaciones puede ejecutar y cómo se comporta ante distintas situaciones.

Por otro lado, la organización de computadores abarca los detalles internos de la implementación del hardware, es decir, cómo está construido físicamente el sistema para cumplir con una arquitectura definida. Este proceso involucra el diseño de circuitos digitales, la jerarquía de memoria, el diseño de canales de comunicación, el funcionamiento del pipeline (flujo interno), el uso de unidades funcionales, los mecanismos de control y la observación de cómo trabaja cada pieza junto al resto. La organización determina cómo el computador ejecuta las funciones definidas por la arquitectura (Stallings, 2019).

La Figura 1.1 muestra la vinculación entre la arquitectura y la organización interna. La parte izquierda presenta la arquitectura mediante los modelos conceptuales de Von Neumann y de Harvard, así como los componentes elementales que conforman el diseño lógico del sistema (CPU y memoria principal). Esta arquitectura debe tomarse como base, en la parte derecha del diagrama, los dispositivos de entrada y salida, la CPU, la memoria y el almacenamiento, todos ellos entrelazados por buses de datos y control. La figura da a entender que la arquitectura define qué componentes existen y cómo deben relacionarse, mientras que la organización materializa cómo esos componentes se implementan y cómo colaboran para que la unidad procese datos y proporcione resultados (Dongarra & Duff, 2020).

ISBN: 978-631-6557-81-0

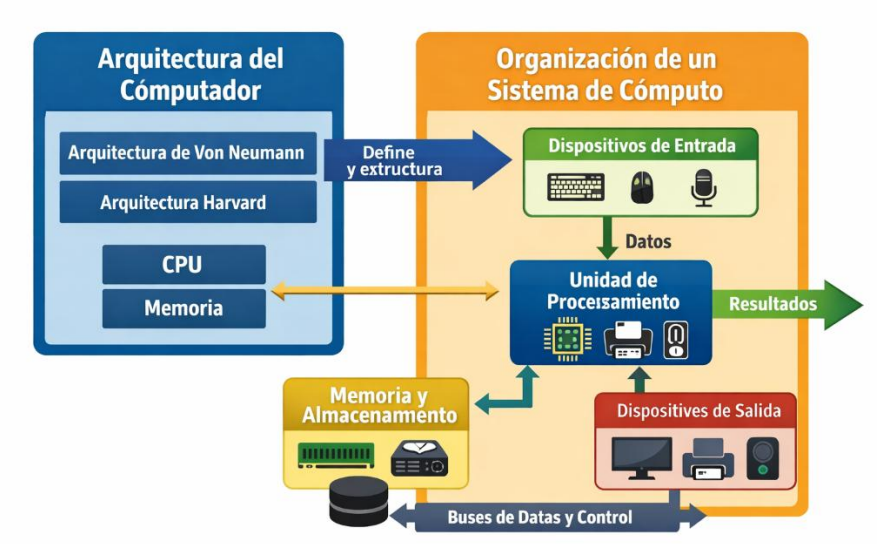


Figura 1.1 Relación entre la arquitectura del computador y la organización funcional

Elaborado por el Autor

1.3 Fuerzas que actúan sobre la Arquitectura de Computadores

El diseño de la arquitectura de computadores está condicionado por un conjunto de fuerzas, restricciones y tendencias tecnológicas que influyen en las decisiones de los arquitectos de hardware. Las fuerzas surgen del entorno tecnológico, de las necesidades del usuario y de la industria del software, lo que condiciona el tipo de componentes que se diseñan, cómo se construyen y hacia qué futuro están orientados (Figura 1.2). Entender estas fuerzas es esencial para interpretar los cambios en las arquitecturas modernas.

a) Avances en la tecnología de fabricación

La forma en que se fabrican los chips cambió mucho con el tiempo, sobre todo al reducir el tamaño de los transistores. Durante años, la ley de Moore junto con la de Dennard hicieron posible incrementar más componentes y la reducción del consumo energético; esto ayudó a construir procesadores más veloces y avanzados.

b) Demanda del software y los modelos de programación

ISBN: 978-631-6557-81-0

A medida que las aplicaciones requieren un mayor procesamiento (IA, procesamiento de big data, entornos virtuales o computación en la nube), la arquitectura evoluciona hacia la aceleración y el uso de paralelismo. Por ello, la arquitectura actúa como enlace entre la capacidad de procesamiento del equipo físico y la dotación de recursos que el software necesita para funcionar de manera eficiente.

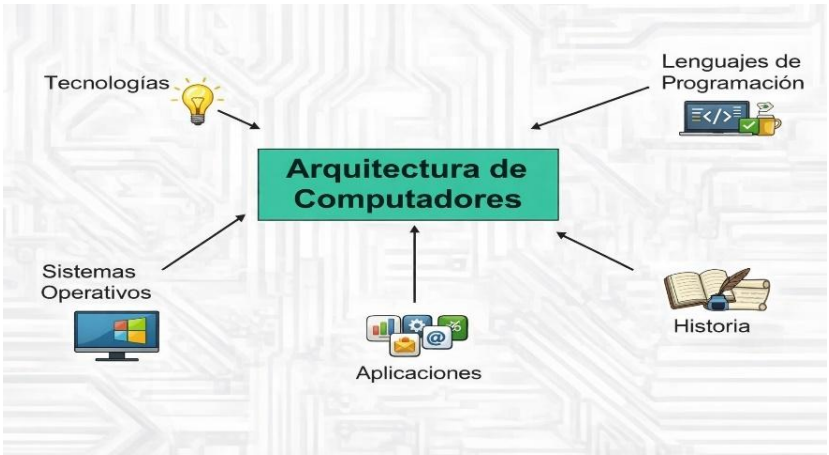


Figura 1.2 Fuerzas externas que permiten el avance de la arquitectura

Elaborado por el autor

c) Requerimientos de rendimiento

Históricamente, la búsqueda de un mayor rendimiento ha sido una de las fuerzas clave en la evolución de los sistemas de cómputo. Las técnicas como el pipelining, el superescalar, la predicción de saltos, las jerarquías de memoria complejas y los buses de alta velocidad buscan mejorar el rendimiento, no solo aumentando la velocidad del procesador, sino también optimizando el rendimiento por watt, el rendimiento por transistor y la eficiencia de la comunicación entre unidades de cómputo.

d) Consumo energético y eficiencia

El consumo de energía ha pasado a ser una fuerza fundamental gracias al incremento del uso de smartphones, de sistemas embebidos y de centros de datos a gran

ISBN: 978-631-6557-81-0

escala. La eficiencia energética condiciona el diseño del conjunto de instrucciones, el número de núcleos y las técnicas de gestión de la energía. Gracias a un equilibrio entre el gasto energético y el desempeño, las arquitecturas aprovechan mejor los recursos sin aumentar el costo operativo ni el consumo térmico.

e) Costos de producción y mercado

El costo de producción es una de las restricciones más determinantes en el diseño de la CPU. Cada transistor adicional, cada capa de caché o cada bus especializado hace que el costo del chip aumente. Estas restricciones obligan a los arquitectos a equilibrar la funcionalidad y la viabilidad económica. Además, las tendencias del mercado —como la demanda de procesadores para móviles, servidores, videojuegos o IA— orientan las características que deben priorizarse en su fabricación.

f) Fiabilidad, seguridad y escalabilidad

Los sistemas computacionales actuales necesitan ser seguros y confiables, ya que los riesgos de ataques reales o digitales se multiplican exponencialmente. Para ello, se añaden funciones al conjunto de instrucciones para proteger datos, detectar fallos, cifrar información y separar tareas. Asimismo, la arquitectura debe diseñarse de manera escalable para adaptarse al crecimiento de las aplicaciones cada vez más exigentes en rendimiento y seguridad.

1.4 La evolución de los equipos informáticos automatizados

La historia de los computadores suele organizarse en eras o generaciones, ya que cada etapa refleja un cambio en la forma de construirlos y en su utilidad. Al inicio, la palabra computador no tenía que ver con la electrónica; se trataba de dispositivos mecánicos capaces de facilitar los cálculos mediante engranajes y palancas. Durante la conocida era mecánica, que se extendió aproximadamente hasta mediados del siglo XX, los avances se centraron en máquinas como las diseñadas por Charles Babbage, que introdujeron las bases de la programación y la separación entre datos e instrucciones,

ISBN: 978-631-6557-81-0

aunque muchas de ellas nunca llegaron a funcionar plenamente. Estas ideas fueron muy importantes para el desarrollo posterior (Stallings et al., 2006).

La siguiente gran etapa fue la primera generación de computadores electrónicos, en la que se reemplazaron los mecanismos físicos por tubos de vacío. Máquinas como el ENIAC (Electronic Numerical Integrator and Computer) marcaron este período. Eran equipos que ocupaban grandes espacios físicos, con un consumo energético muy alto y que se recalentaban con facilidad. Programarlos no era algo flexible: cambiar un programa significaba mover cables o ajustar interruptores a mano. A pesar de sus limitaciones, demostraron que el cómputo electrónico era mucho más rápido que cualquier sistema mecánico, lo que impulsó su uso en contextos científicos y militares (Patterson & Hennessy, 2000).

La segunda generación trajo consigo el desarrollo de los transistores, lo que disminuyó el tamaño, el consumo y la fiabilidad. Los computadores se volvieron más compactos y menos frágiles, y comenzaron a utilizarse los primeros lenguajes de alto nivel como FORTRAN o COBOL, lo que cambió la forma de programar. Por primera vez, el desarrollador no tenía que centrarse únicamente en el hardware, los bits o las señales eléctricas, sino que podía escribir programas más cercanos al lenguaje humano (Ramírez, 2009).

Con la tercera generación llegaron los circuitos integrados. En lugar de ensamblar transistores uno por uno, ahora muchos podían colocarse en un solo chip. Esto permitió crear sistemas más potentes y complejos. En esta etapa surgieron los sistemas operativos, la multiprogramación y el uso compartido del tiempo de CPU. También se empezaron a explorar ideas como el paralelismo y la segmentación de instrucciones, conceptos que siguen siendo fundamentales en la arquitectura de procesadores modernos (Denning, 1971).

La cuarta generación se asocia con la llegada del microprocesador y el nacimiento del computador personal. El procesador pasó a integrarse en un único chip, lo que redujo

ISBN: 978-631-6557-81-0

costos y facilitó la producción a gran escala. Para el software, este cambio fue enorme: surgieron sistemas operativos más amigables, interfaces gráficas y lenguajes como C, que todavía se usan ampliamente. El computador dejó de ser una máquina exclusiva de laboratorios y empresas grandes y pasó a ser un elemento cotidiano de la vida (Moto-Oka, 2012).

Finalmente, desde los años noventa hasta hoy, hablamos de una etapa más continua, a veces llamada quinta o sexta generación, según el enfoque. No hubo un solo salto tecnológico, sino muchos avances acumulados: multinúcleo, arquitecturas paralelas, redes de alta velocidad y dispositivos móviles. Además, los computadores ya no siempre se ven como PCs; están integrados en teléfonos, autos y electrodomésticos mediante microcontroladores. Para quienes estudian software, esta es clave porque el reto ya no es solo programar, sino hacerlo de forma eficiente sobre arquitecturas cada vez más paralelas y diversas (Blaauw & Brooks Jr, 1997).

Con esta revisión general, la evolución de los equipos computacionales se ha visto influida por los avances tecnológicos, las necesidades del entorno científico y empresarial y la búsqueda constante de aumentar la capacidad de procesamiento. Desde los primeros sistemas electromecánicos y máquinas de cálculo programables hasta los actuales computadores de arquitectura multinúcleo y dispositivos inteligentes; cada fase ha incorporado mejoras significativas en velocidad, miniaturización, confiabilidad y eficiencia energética. Esta transformación ha permitido que los sistemas informáticos evolucionen de máquinas especializadas y de gran tamaño a dispositivos versátiles, compactos y accesibles, capaces de ejecutar tareas complejas de manera autónoma y en tiempo real, así como la utilización de la IA para mejorar sus procesos (Laudon & Laudon, 2025). En la Tabla 1.1 se presenta, de manera resumida, la evolución de los dispositivos computacionales.

ISBN: 978-631-6557-81-0

1.5 Arquitectura de Von Neumann

La arquitectura de Von Neumann es fundamental en la informática moderna; gracias a ella se estableció una forma ordenada y lógica de construir computadoras capaces de ejecutar diferentes programas sin necesidad de modificar físicamente sus componentes (Vázquez Gómez, 2012). Neumann planteó un diseño para que el hardware se volviera más flexible y el software desarrollara su pleno potencial. La idea central de este enfoque es sencilla, pero tuvo un impacto enorme en la forma en que se construyen y se programan los computadores (Villarreal, 2017). En este modelo, la unidad de control toma las instrucciones de la memoria, las interpreta y las ejecuta una por una, de forma secuencial, lo que permite cambiar los programas sin alterar el hardware. Gracias a esto, fue posible crear lenguajes de programación más flexibles y sistemas operativos que gestionan mejor los recursos del computador (Arriarán, 2015).

En este esquema, el sistema se organiza básicamente en tres partes: la CPU, la memoria principal y los dispositivos de entrada y salida. La CPU es el “cerebro” del sistema y, dentro de ella, se encuentran la unidad aritmético-lógica, que se encarga de realizar cálculos y comparaciones, y la unidad de control, que indica qué operación debe ejecutarse en cada momento. La memoria principal almacena tanto las instrucciones de los programas como los datos, todos almacenados de forma lineal en el mismo espacio, lo que simplifica mucho el diseño (López & Rueda, 2009).

ISBN: 978-631-6557-81-0

Época / Periodo	Dispositivo / Hito	Fecha aproximada	Características clave	Impacto en la computación
Antigüedad	Ábaco	≈ 2400 a.C.	Primer instrumento mecánico de cálculo; uso manual.	Base del cálculo sistemático.
Siglo XVII	Regla de cálculo	1620	Operaciones logarítmicas; herramienta analógica.	Fundamental en ciencia e ingeniería durante siglos.
1642	Pascalina	1642	Máquina de sumar/restar de Pascal; engranajes mecánicos.	Primer dispositivo mecánico automático.
Siglo XVII	Máquina de Leibniz	1672	Realizaba multiplicaciones y divisiones.	Amplió el concepto de cálculo mecánico.
Siglo XIX	Máquina diferencial de Babbage	1822	Diseñada para tablas matemáticas; totalmente mecánica.	Precursor conceptual de la computadora moderna.
Siglo XIX	Máquina analítica de Babbage	1837	Programa mediante tarjetas perforadas; unidad aritmética y de control.	Primer diseño de computadora programable.
Finales XIX	Telar de Jacquard	1804	Tarjetas perforadas para control de patrones.	Inspiró la programación mediante tarjetas.
Década de 1890	Máquinas tabuladoras de Hollerith	1890	Procesamiento de datos mediante tarjetas perforadas.	Base de IBM y del procesamiento automático de datos.
Década 1930	Z1 (Konrad Zuse)	1938	Primera computadora binaria electro-mecánica.	Hito en automatización del cálculo.
Década 1940	Colossus	1943	Usada en criptografía; electrónica basada en válvulas.	Primera computadora electrónica programable.
Década 1940	ENIAC	1945	Miles de tubos de vacío; programación manual.	Considerada la primera computadora electrónica de propósito general.
Década 1940	EDVAC	1949	Arquitectura de programa almacenado (Von Neumann).	Modelo dominante en computadoras modernas.
Década 1950	UNIVAC I	1951	Primera computadora comercial.	Expansión de la computación a la industria.
Década 1950	Transistores	1947/1950s	Sustituyen válvulas; menor tamaño, menos calor.	Revolución en miniaturización y fiabilidad.
Década 1960	Circuitos integrados (IC)	1958/1960s	Miles de transistores en un chip.	Inicio de la microelectrónica moderna.
Década 1970	Microprocesador Intel 4004	1971	Primer CPU comercial en un solo chip.	Nacimiento de las computadoras personales.
Década 1970	Altair 8800	1975	Primer microcomputador popular.	Impulso a la informática doméstica.
Década 1980	IBM PC	1981	Arquitectura abierta; uso de MS-DOS.	Estándar dominante de PCs.
Década 1980	Computadoras Macintosh	1984	Interfaz gráfica (GUI).	Popularizó las interfaces visuales.
Décadas 1990–2000	Computadoras portátiles	1990s	Movilidad; baterías mejoradas; miniaturización.	Computación personal móvil.
Décadas 2000–2010	Smartphones	2007	Computación completa en dispositivos móviles.	Transformación del acceso a la información.
Actualidad	Computación en la nube	2010→	Procesamiento remoto masivo.	Cambio de modelo computacional global.
Actualidad	IA y aceleradores (GPU/TPU)	2015→	Paralelismo masivo; redes neuronales.	Impulso a la computación inteligente.

Tabla 1.1 Evolución histórica y avances tecnológicos.

Elaborado por el autor y mejorado con IA generativa

Por su parte, los dispositivos de entrada y salida permiten la comunicación con el exterior, por ejemplo, al ingresar información o al recibir resultados. Un concepto clave en esta arquitectura es el ciclo de instrucción, que se repite una y otra vez: primero se toma la instrucción de la memoria, luego se interpreta y, finalmente, se ejecuta. Este proceso tan sencillo permitió que las computadoras evolucionaran durante años sin cambiar su

ISBN: 978-631-6557-81-0

estructura básica, facilitando, además, el diseño del hardware y la construcción de los sistemas (Figura 1.3).

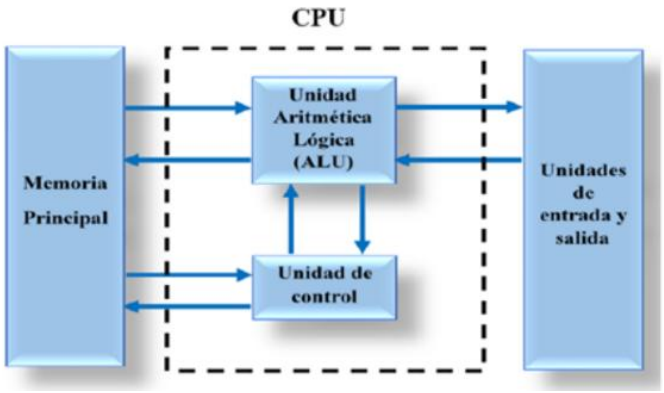


Figura 1.3 Componentes de la arquitectura de Neumann.

Elaborado por el autor

La arquitectura de Von Neumann es uno de los modelos más importantes en la historia de la computación, pero también presenta una limitación conocida como el "cuello de botella de Von Neumann". Este problema aparece porque tanto los datos como las instrucciones viajan por el mismo canal entre la CPU y la memoria, lo que hace que el procesador muchas veces tenga que esperar a que la información llegue, sin importar cuán rápido sea (Ibanez, 1999). En la práctica, esto significa que el rendimiento del sistema no depende solo de la potencia del CPU, sino también de la velocidad con la que se pueden mover los datos. Con el tiempo, esta limitación se fue reduciendo mediante técnicas como el uso de la memoria caché, la ejecución paralela de instrucciones y la incorporación de sistemas multiprocesador. Aun así, el modelo de Neumann sigue siendo la base de la mayoría de las computadoras actuales, principalmente por su diseño simple y flexible. Para los estudiantes de software, entender esta arquitectura ayuda mucho a comprender cómo se ejecutan los programas, por qué muchas instrucciones se procesan de forma secuencial y cómo el software interactúa realmente con el hardware (Arikpo et al., 2007).

ISBN: 978-631-6557-81-0

1.6 Arquitectura de Harvard

La arquitectura de Harvard es un modelo de organización interna de los sistemas computacionales que se caracteriza por la separación física entre la memoria destinada al almacenamiento de instrucciones y la memoria destinada al almacenamiento de datos (ver Figura 1.4). Aunque hoy existen muchas variantes y extensiones, la idea central sigue siendo la misma: el procesador no usa un único espacio de memoria para todo, sino que dispone de dos espacios de memoria distintos, con buses separados, y cada uno cumple una función muy clara (Murdocca et al., 2002). Esta separación permite que el procesador lea instrucciones y datos simultáneamente, lo que mejora el rendimiento en muchas aplicaciones, especialmente en sistemas embebidos y en computadores que ejecutan tareas muy específicas. Para entender mejor por qué surgió esta arquitectura, conviene recordar que los primeros computadores electrónicos eran muy lentos para mover información hacia y desde el procesador y tenían un único espacio de almacenamiento tanto para instrucciones como para datos (Null & Lobur, 2014).

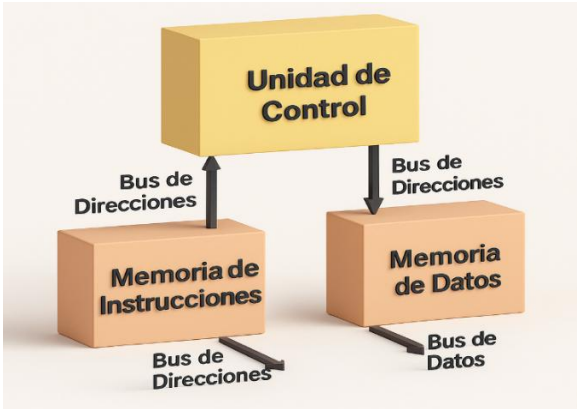


Figura 1.4 Base de la arquitectura de Harvard

Elaborada por el autor

Una ventaja de la arquitectura de Harvard es que el procesador puede obtener una instrucción mientras accede, en paralelo, a los datos que requiere. Este paralelismo básico reduce los tiempos de espera y permite que el sistema funcione con mayor eficiencia. Esta

ISBN: 978-631-6557-81-0

característica hace que Harvard sea útil en sistemas en los que el tiempo es crítico, como controladores industriales, microcontroladores de automóviles, dispositivos médicos y sensores inteligentes. No obstante, sus ventajas, la arquitectura de Harvard presenta algunas limitaciones. Una de ellas es que el software no puede modificar directamente sus propias instrucciones durante la ejecución. Esto dificulta que técnicas como la automodificación de código, comunes en sistemas más flexibles, resulten imposibles de utilizar. Además, programar para una arquitectura Harvard pura puede resultar más complejo, ya que el compilador debe determinar con precisión en qué tipo de memoria se ubicará cada instrucción o dato. Este detalle no es visible para la mayoría de los estudiantes que trabajan con lenguajes de alto nivel, pero sí lo es para quienes programan microcontroladores o desarrollan sistemas de bajo nivel (Hamacher et al., 1984).

1.7 Clasificación de los computadores por su capacidad de procesamiento

Cuando se habla de computadores, no todos tienen la misma potencia ni están diseñados para resolver los mismos tipos de problemas. A lo largo del tiempo, y también por razones prácticas, se ha establecido una clasificación que ayuda a entender para qué sirve cada tipo de máquina y en qué escenarios se utiliza. Esta clasificación no es rígida porque la tecnología avanza con rapidez y, a veces, los límites se vuelven un poco difusos, pero sigue siendo muy útil para estudiantes y profesionales de software.

En el estudio de la arquitectura de los computadores, evaluar el rendimiento es fundamental para entender cómo responde un sistema ante cargas de trabajo. La forma más importante de medir el rendimiento de un procesador es observar qué tan rápido ejecuta instrucciones. Esta idea suele expresarse como el producto de la frecuencia del reloj del procesador (f, normalmente medida en MHz o GHz) y el IPC (Instructions Per Cycle), que indica cuántas instrucciones, en promedio, se completan en cada ciclo del reloj (MIPS rate = f × IPC). Durante mucho tiempo se utilizó como indicador el MIPS (millones de instrucciones por segundo), que mide cuántas instrucciones puede ejecutar una máquina por segundo. En teoría, una computadora con mayor valor de MIPS debería ser más rápida,

ISBN: 978-631-6557-81-0

aunque en la práctica no es tan simple, ya que depende del tipo de instrucciones, del programa que se ejecuta y el diseño interno del procesador (Mir et al., 2019).

Otro factor importante es el FLOPS, que mide las operaciones de punto flotante por segundo. Este indicador es especialmente útil en áreas como la simulación, la ingeniería y los cálculos científicos, donde las operaciones matemáticas resultan complejas. A diferencia de MIPS, que es más general, los FLOPS se centran más en el rendimiento numérico que en la ejecución de instrucciones en general. Para evaluar todos estos factores de forma más realista, se utilizan programas de benchmarking, que consisten en pruebas estandarizadas diseñadas para medir el rendimiento bajo determinadas condiciones. En esta sección se detallan cuatro grandes categorías: supercomputadores, mainframes, minicomputadores y microcomputadores o PC. La idea es entender cuál es la diferencia y cuál es el papel que desempeñan en el mundo real (Quiroga, 2010).

1.7.1 Supercomputadores

Los supercomputadores representan el nivel más alto de capacidad de procesamiento. Son máquinas enormes, instaladas casi siempre en centros de investigación, universidades grandes, agencias militares o empresas que requieren cálculos intensivos. Su propósito principal no es ejecutar programas comunes sino resolver problemas matemáticos gigantescos: simulaciones climáticas, modelos de física cuántica, análisis genéticos avanzados, cálculos para energías renovables, predicción de terremotos y cosas así. Estos equipos están compuestos por miles de procesadores trabajando en paralelo, y su rendimiento se mide en FLOPS. Para que se entienda mejor: un supercomputador moderno puede realizar trillones de operaciones por segundo, algo totalmente fuera del alcance de los equipos personales. También consumen muchísima energía, requieren sistemas de enfriamiento especiales y son tan costosos que solo unos pocos países o instituciones pueden mantener uno. Aunque pueden parecer lejanas a la vida diaria de un programador, muchas tecnologías actuales, como nuevas medicinas o modelos de inteligencia artificial, se han desarrollado gracias a simulaciones en supercomputadores. Por ejemplo, el supercomputador “El CAPITAN” figura en la lista

ISBN: 978-631-6557-81-0

Top500 (www.top500.org) como el más poderoso del mundo (noviembre de 2025). Este se utiliza en el Lawrence Livermore National Laboratory como herramienta multifuncional en proyectos de investigación en energía, clima y fusión nuclear (figura 1.5). En la Tabla 1.2 se presenta el listado de los 5 supercomputadores y sus principales características.

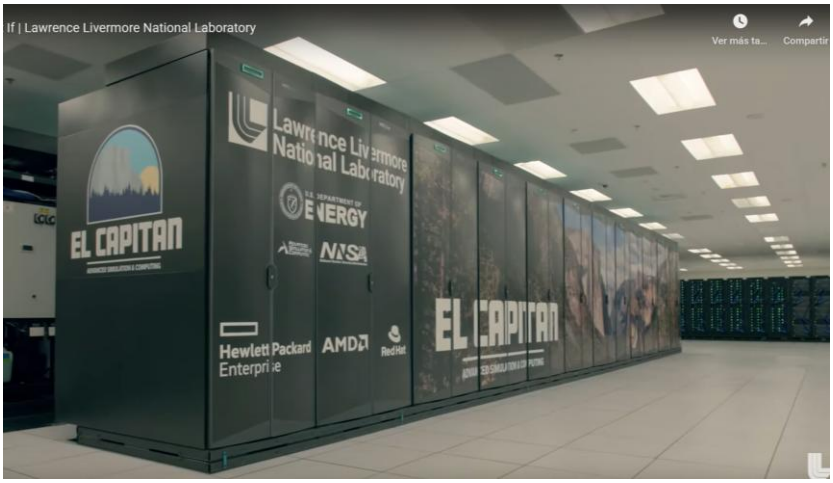


Figura 1.5 Supercomputador "El Capitan"

Fuente: https://www.hpe.com/lamerica/es/compute/hpc/lawrence-livermore-national-laboratory.html

1.7.2 Mainframes

Debajo de los supercomputadores se encuentran los mainframes (equipos poderosos, pero con un propósito distinto). Mientras un supercomputador se enfoca en cálculos científicos, un mainframe está diseñado para procesar enormes volúmenes de transacciones y datos con la mayor confiabilidad posible. Se utilizan en entidades bancarias, aerolíneas, grandes cadenas de tiendas, aseguradoras, gobiernos y empresas que requieren una disponibilidad casi absoluta. El mainframe es capaz de manejar miles de solicitudes simultáneamente sin fallas y con un nivel de seguridad muy alto. Un ejemplo común es cuando un mainframe procesa las transacciones de los usuarios al realizar compras con la tarjeta de crédito o mediante transferencia bancaria.

ISBN: 978-631-6557-81-0

Aunque hoy en día, los mainframes pueden ser reemplazados por clústeres de servidores más modernos, siguen siendo populares por su estabilidad y capacidad de manejar grandes volúmenes transaccionales. En muchos casos, estos sistemas llevan décadas en funcionamiento, lo que demuestra su robustez. Un ejemplo es el mainframe de IBM z16 (https://www.ibm.com/es-es/products/z1), lanzado en 2022, diseñado para trabajar en la nube híbrida con altas cargas de trabajo críticas, y destacado por integrar un acelerador de IA en el chip Telum para analizar datos en tiempo real (Figura 1.6).

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**El Capitan** - HPE Cray EX255a, AMD 4th Gen EPYC 24C 1.8GHz, AMD Instinct MI300A, Slingshot-11, TOSS, HPE DOE/NNSA/LLNL United States	11,340,000	1,809.00	2,821.10	29,685
2	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE Cray OS, HPE DOE/SC/Oak Ridge National Laboratory United States	9,066,176	1,353.00	2,055.72	24,607
3	**Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	9,264,128	1,012.00	1,980.01	38,698
4	**JUPITER Booster** - BullSequana XH3000, GH Superchip 72C 3GHz, NVIDIA GH200 Superchip, Quad-Rail NVIDIA InfiniBand NDR200, RedHat Enterprise Linux, EVIDEN EuroHPC/FZJ Germany	4,801,344	1,000.00	1,226.28	15,794
5	**Eagle** - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Azure Microsoft Azure United States	2,073,600	561.20	846.84	

Tabla 1.2 Top 5 de supercomputadores en noviembre de 2025.

Fuente: https://top500.org/lists/top500/2025/11/

Con su capacidad para procesar millones de transacciones por segundo y mantener tiempos de respuesta muy bajos, el IBM z16 sigue siendo fundamental en sectores como el bancario, el de seguros, el retail y el comercio masivo, las aerolíneas, los gobiernos y agencias públicas, las empresas de telecomunicaciones y el sector de la salud.

ISBN: 978-631-6557-81-0

Figura 1.6 Mainframe de IBM z16

Fuente: https://developer.ibm.com/blogs/a-tour-inside-the-ibm-z16/

1.7.3 Minicomputadores

Los minicomputadores, que surgieron entre las décadas de 1960 y 1980, fueron una categoría intermedia entre los mainframes y las computadoras personales. No eran tan gigantes ni tan costosos como un mainframe, pero sí lo suficientemente potentes como para manejar tareas empresariales importantes. Su trabajo funcional destacaba en universidades, laboratorios y compañías que necesitaban sistemas multiusuario. Con el tiempo, muchos minicomputadores desaparecieron o fueron reemplazados por servidores modernos, pero su legado se encuentra en la idea de sistemas medianos capaces de atender a varios usuarios simultáneamente. Uno de los minicomputadores más utilizados en los 90 fue el IBM AS/400 (luego iSeries) por su arquitectura robusta, con procesadores Power RISC (Figura 1.7).

ISBN: 978-631-6557-81-0

1.7.4 Microcomputadores

En el nivel más cercano a todos están los microcomputadores, también conocidos como PC (Personal Computer). Desde los años 80, estos computadores han transformado el acceso a la informática, porque antes de ellos la computación estaba reservada casi exclusivamente a las grandes instituciones.

Figura 1.7 Equipo IBM AS/400, ampliamente utilizado en entornos empresariales.

Fuente :https://arquitcomputadoras.wordpress.com/2013/05/28/servidor-ibm-as400/

Un microcomputador es básicamente una máquina que integra un microprocesador como unidad central de procesamiento, junto con memoria, almacenamiento y periféricos. Aunque hoy parecen “normales”, en su momento fueron revolucionarios porque redujeron el tamaño y el costo de la computación. Actualmente, las PCs pueden realizar tareas avanzadas de programación, diseño, edición de video e incluso entrenamiento de modelos pequeños de inteligencia artificial (Figura 1.8).

Además, los computadores portátiles y las tablets se consideran parte de esta misma familia porque usan arquitecturas similares basadas en microprocesadores. Su evolución

ISBN: 978-631-6557-81-0

ha sido tan significativa que hoy un PC integra capacidades impensables en sus inicios, lo que evidencia la miniaturización y la fabricación de chips integrados.

Figura 1.8 PC gaming

Fuente: Gemini

1.8 Ley de Moore

La Ley de Moore es una observación clave en la industria tecnológica, propuesta por Gordon Moore allá por los años 60; dice que la cantidad de transistores encapsulados en un chip se duplica cada cierto tiempo, originalmente cada dos años. Esto venía observándose en la industria mientras los fabricantes mejoraban sus procesos, reducían el tamaño de los transistores, disminuían el consumo de energía y aumentaban los requerimientos de enfriamiento (cooling); así, los procesadores se volvían más rápidos y eficientes casi de manera "automática". No es una ley física, sino más bien una predicción que terminó guiando a toda la industria del hardware durante décadas, como si fuera una especie de meta que todos intentaban alcanzar (Ortega et al., 2019).

Con el desarrollo tecnológico actual, cumplir la Ley de Moore se vuelve cada vez más difícil. Aunque los fabricantes utilizan múltiples núcleos o nuevas arquitecturas, ya no es tan sencillo seguir duplicando la cantidad de transistores al mismo ritmo. Aun así, la ley dejó una huella enorme en cómo se diseñan las computadoras y todavía hoy se usa para

ISBN: 978-631-6557-81-0

hablar de la evolución del rendimiento computacional, aunque ya no se cumpla tan estrictamente como antes (Figura 1.9).

1.9 Representación numérica de la información

Los circuitos que hay dentro de una CPU no manejan números en absoluto. En su lugar, siguen reglas eléctricas básicas y responden automáticamente a las señales que reciben. Esas señales provienen de software escrito por programadores o de las entradas de los usuarios en una aplicación. Cuando el resultado de un programa aparece como dígitos en una tabla o como letras en un documento, se trata simplemente de una representación numérica de la información. Se vincula el voltaje bajo con 0 y el voltaje alto con 1: ahí es donde comienza el significado. La unidad más pequeña en una máquina digital es el bit, que solo puede tomar dos valores (0 o 1). Cuando varios bits se agrupan, pueden representar números más variados. Un byte está compuesto por 8 bits unidos para formar un valor. Un byte es la unidad de información más pequeña que la mayoría de las CPU actuales pueden capturar o almacenar.

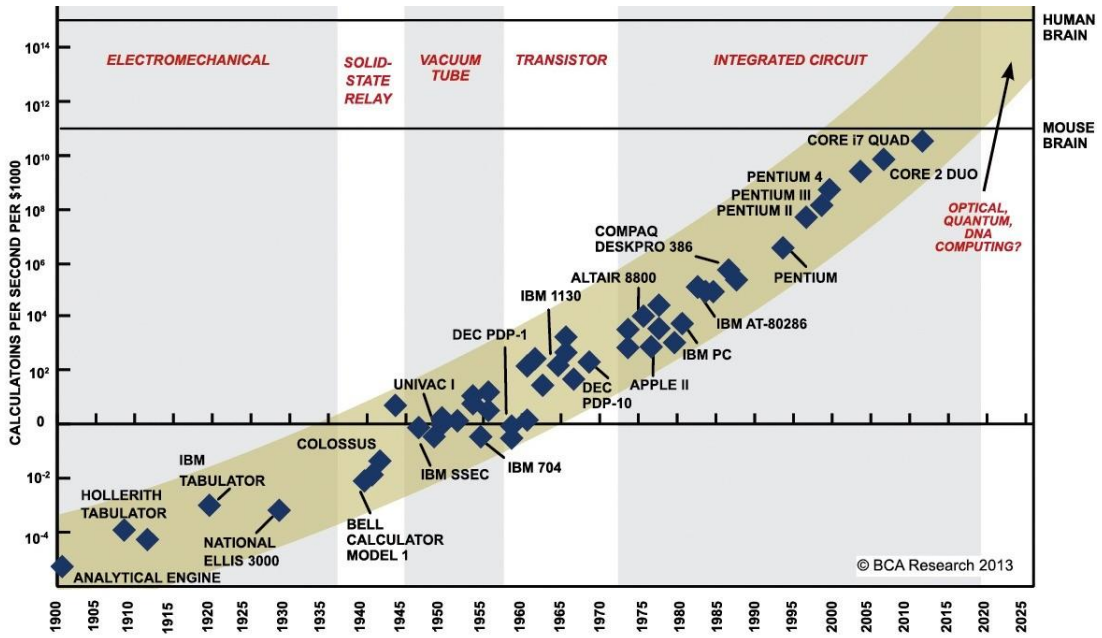


Figura 1.9 Evolución de los sistemas de cómputo enfocados a operaciones por segundo
Fuente: Trabajo de Kurzweil (2005).

ISBN: 978-631-6557-81-0

1.9.1 Representación numérica de la información

Los circuitos que hay dentro de una CPU no manejan números en absoluto. En su lugar, siguen reglas eléctricas básicas y responden automáticamente a las señales que reciben. Esas señales provienen de software escrito por programadores o de las entradas de los usuarios en una aplicación. Cuando el resultado de un programa aparece como dígitos en una tabla o como letras en un documento, se trata simplemente de una representación numérica de la información. Se vincula el voltaje bajo con 0 y el voltaje alto con 1: ahí es donde comienza el significado. La unidad más pequeña en una máquina digital es el bit, que solo puede tomar dos valores (0 o 1). Cuando varios bits se agrupan, pueden representar números más variados. Un byte está compuesto por 8 bits unidos para formar un valor. Un byte es la unidad de información más pequeña que la mayoría de las CPU actuales pueden capturar o almacenar.

a) Sistema Binario

El sistema binario es el lenguaje más básico que entienden las computadoras. A diferencia del sistema decimal, que usa diez dígitos, aquí solo existen dos: el 0 y el 1. Según el material, cada posición tiene un peso que es una potencia de 2, algo así como 2^0, 2^1, 2^2, etc., y también potencias negativas cuando hablamos de fracciones. Por ejemplo, un número como 10111.101_2 se descompone multiplicando cada dígito por su potencia de 2 correspondiente (Pérez et al., 2005). Para un número binario con parte entera y parte fraccionaria, se usa:

$$\sum_{i=0}^{n} b_i \cdot 2^i \; + \; \sum_{j=1}^{m} b_{-j} \cdot 2^{-j}$$

Donde:

b_ison los bits de la parte entera, desde el menos significativo hasta el más significativo.

ISBN: 978-631-6557-81-0

b_(-j) son los bits de la parte fraccionaria, desde el primer bit después del punto hacia la derecha.

Ejemplo:

$$10111.1012=1\cdot24+0\cdot23+1\cdot22+1\cdot21+1\cdot20+1\cdot2-1+0\cdot2-2+1\cdot2-3$$

b) Sistema Octal

El sistema octal es un sistema de numeración que utiliza la base 8, es decir, trabaja únicamente con los dígitos del 0 al 7. Aunque hoy no es tan visible como el sistema decimal o el hexadecimal, el octal tuvo bastante importancia en los primeros sistemas de computación y todavía se usa en algunos contextos específicos, como los permisos de archivo en sistemas operativos tipo Unix. Su principal ventaja es que resulta más fácil de relacionar con el sistema binario, ya que cada dígito octal equivale exactamente a tres bits. Esto hace que la conversión entre binario y octal sea un proceso más directo y menos propenso a errores cuando se realiza a mano, lo cual es útil para entender cómo la computadora maneja la información internamente.

Para representar un número en el sistema octal se sigue el mismo principio posicional que en otros sistemas numéricos. Cada dígito se multiplica por una potencia de 8 según su posición, comenzando por la derecha con la potencia 0. Por ejemplo, un número octal como 5742.36_8 se puede descomponer sumando cada dígito multiplicado por 8 elevado a su posición correspondiente, incluyendo potencias negativas para la parte fraccionaria (Angulo Usategui, 2007).

$$Número = \sum \ (d_i \cdot 8^{posición})$$

Ejemplo:

$$5742.368 = (5\times83) + (7\times182) + (4\times81) + (2\times80) + (3\times8-1) + (6\times8-2)$$

ISBN: 978-631-6557-81-0

c) Sistema Hexadecimal

El sistema hexadecimal usa 16 símbolos: 0 a 9 y luego las letras A a F, que equivalen a 10, 11, 12, 13, 14 y 15, respectivamente. Este sistema se utiliza con frecuencia en computación porque es más sencillo leer estos valores que el sistema binario: un solo dígito hexadecimal representa cuatro bits binarios. En programación y en temas de arquitectura del computador es muy común ver direcciones de memoria o códigos en hexadecimal, ya que quedan más cortos y ordenados, aunque por dentro la máquina sigue trabajando en binario. Por ejemplo, el número $ACF4.96_{16}$ se descompone utilizando:

$$Número = \sum (d_i \cdot 16^{posición})$$

Ejemplo:

$$ACF4.9616 = (10\times16^3) + (12\times16^2) + (15\times16^1) + (4\times16^0) + (9\times16^{-1}) + (6\times16^{-2})$$

Conversión de sistemas numéricos

Cuando se quiere convertir un número decimal a binario (o, en general, a otra base), se puede utilizar el método de divisiones y multiplicaciones: una para la parte entera y otra para la parte fraccionaria. Por ejemplo, para convertir 3219.6875_{10} → binario, se realizan divisiones sucesivas:

División	Cociente	Resto
3219 ÷ 2	1609	**1**
1609 ÷ 2	804	**1**
804 ÷ 2	402	**0**
402 ÷ 2	201	**0**
201 ÷ 2	100	**1**
100 ÷ 2	50	**0**
50 ÷ 2	25	**0**
25 ÷ 2	12	**1**
12 ÷ 2	6	**0**
6 ÷ 2	3	**0**
3 ÷ 2	1	**1**
1 ÷ 2	0	**1**

ISBN: 978-631-6557-81-0

Se toman los restos de abajo hacia arriba: **$1\,1\,0\,0\,1\,0\,0\,1\,0\,0\,1\,1_2$** en la parte entera.

Para convertir la parte fraccionaria 0.6875_{10} → binario, se efectúan multiplicaciones sucesivas:

Multiplicación	Resultado	Parte Entera
0.6875 × 2	1.375	**1**
0.375 × 2	0.75	**0**
0.75 × 2	1.5	**1**
0.5 × 2	1.0	**1**

Se toman la parte entera del resultado de arriba hacia abajo: 1 0 1 1

Resultado: 3219.6875_{10} → **$1\,1\,0\,0\,1\,0\,0\,1\,0\,0\,1\,1\,.\,1\,0\,1\,1_2$**

Otro ejemplo, conversión de 3219.39_{10} → hexadecimal

a) Parte entera: 3219 → hex

División	Cociente	Resto
3219 ÷ 16	201	**3**
201 ÷ 16	12	**9**
12 ÷ 16	0	**C**

b) Parte fraccionaria: 0.39 → hex

Multiplicación	Resultado	Parte Entera
0.39 × 16	6.24	**6**
0.24 × 16	3.84	**3**
0.84 × 16	13.44	**D**
0.44 × 16	7.04	**7**

Resultado: **3219.3910 ≈ C93.63D716**

En los sistemas de numeración utilizados en la arquitectura de computadores, el rango de representación depende directamente de la cantidad de bits disponibles. Con n bits, es posible representar 2^n valores distintos, lo que establece un intervalo de 0 a 2^n−1. Por ejemplo, con 4 bits pueden representarse 16 valores diferentes, comprendidos entre 0 y 15. Si se quiere saber cuántos bits hacen falta para representar un valor numérico (N), se

ISBN: 978-631-6557-81-0

puede usar la fórmula $n = \log_2 (N)$. Por ejemplo, si se necesita representar el número 16, bastan 4 bits, ya que $\log_2 (16)$ es igual a 4. En el caso de números negativos, existen distintas formas de representarlos, como el sistema de signo y magnitud o los métodos de complemento a 1 y complemento a 2.

1.9.2 Representación numérica de enteros

Para representar números enteros con signo en binario se pueden utilizar distintos métodos (signo-magnitud, complemento a uno y complemento a dos). Cada uno permite representar valores positivos y negativos de un número limitado de bits, lo que facilita su interpretación y procesamiento por los componentes de hardware. Además, resulta esencial comprender cómo se realizan las operaciones aritméticas básicas (suma o resta), utilizando estos formatos, así como sus características, beneficios y limitaciones, que influyen en el diseño y el funcionamiento de los sistemas digitales.

a) Signo-magnitud

El sistema de signo-magnitud es uno de los métodos más simples para representar números enteros con signo en binario. Básicamente, el primer bit (el más a la izquierda) se utiliza para indicar el signo (0 significa positivo y 1 negativo). Los demás bits se encargan de mostrar el valor absoluto del número. Para poder representar ciertos valores, se necesita una cantidad suficiente de bits; por ejemplo, si se quiere representar el número 7, se requieren tres bits para la magnitud y uno adicional para el signo, dando un total de cuatro bits. Aunque este método es fácil de entender, no resulta muy práctico cuando se realizan operaciones aritméticas más complejas (Entrialgo Castaño et al., 2023).

SIGNO	Magnitud (n-1 bits)

0 → número positivo

1 → número negativo

ISBN: 978-631-6557-81-0

Por ejemplo, $+78_{10}$ =

Signo	Magnitud (n-1 bits)
0	1 0 0 1 1 1 0

-1239_{10} =

Signo	Magnitud (n-1 bits)
1	0 0 0 0 1 0 0 1 1 0 1 0 1 1 1

En este caso, el rango de representación va desde $-(2n-1-1)$ hasta $+(2n-1-1)$, lo que implica un total de 2n combinaciones. Un ejemplo con 3 bits ilustra que existen dos maneras de representar el cero (000 y 100), lo que constituye una de las limitaciones más conocidas de este sistema. Seguidamente, se detallan las posibles combinaciones con 3 bits:

Representación	Signo–Magnitud	Sin signo
000	0	0
001	1	1
010	2	2
011	3	3
100	–0	4
101	–1	5
110	–2	6
111	–3	7

En esta tabla se evidencia la doble representación del cero en el signo–magnitud (+0 y -0) y la interpretación de los patrones binarios.

b) Complemento a 1

El complemento a 1 es una forma de representar números enteros con signo en binario y simplifica las operaciones aritméticas con números negativos. En esta representación, los números positivos se representan igual que en signo-magnitud; en cambio, un número negativo se invierte cada bit del número en la representación positiva,

ISBN: 978-631-6557-81-0

cambiando los 0 por 1 y viceversa. Este proceso se denomina complementar a 1 (Entrialgo Castaño, 2019). Ejemplo: Representación de –1987 en complemento a 1.

1. Determinar el tamaño total en bits (en escala de base 2)

Para representar signo-magnitud en complemento a 1, se agrega un bit adicional para el signo. Si se utiliza 16 bits en total (1 bit de signo + 15 bits de magnitud).

2. Convertir 1987 a binario

Primero convertimos el valor absoluto:

198710 = 00001111100001112. Este número requiere 15 bits para la magnitud.

3. Obtener el complemento a 1

El complemento a 1 se obtiene invirtiendo todos los bits: (Cambiar 0→1 o 1→ 0)

0 000011111000011 (representación positiva)

↓ invertir bits

1 111100000111100 (representación en complemento a 1) = -1987

c) Complemento a 2

El complemento a 2 (C2) es el sistema de representación de números enteros con signo más utilizado en la arquitectura de computadores y en sistemas digitales. Su principal ventaja es que permite realizar operaciones aritméticas —como suma y resta— utilizando el mismo hardware diseñado para números sin signo, evitando tratar por separado los signos y simplificando notablemente el diseño de las unidades aritmético-lógicas (ALU). Para obtener el C2 se invierten todos los bits (complemento a 1) y se suma 1 al resultado (Angulo Usategui, 2007). Otro proceso para calcular C2 consiste en copiar de derecha a izquierda todos los bits hasta encontrar el primer "1" y, a partir de él, complementar el resto de los bits (C1).

ISBN: 978-631-6557-81-0

Método 1: C2(número) = C1(número) + 1

Para calcular C2(−90)

C2(−90) = C1(90) + 1 = 1 0 1 0 0 1 0 1 + 1 = 1 0 1 0 0 1 1 0

Método 2: Búsqueda del primer '1' desde la derecha

↓

C2(−90) = 1 1 0 1 1 0 1 0 = 1 0 1 0 0 1 1 0

Ejercicios de suma y resta con C1 y C2

Para estos ejercicios, primero se va a recordar la suma básica de un bit:

Operación	Resultado	Acarreo
0 + 0	0	0
0 + 1	1	0
1 + 0	1	0
1 + 1	0	1

Enunciado: A = 213 + 14; B = A – 1237; y C = -A – 993. Resolver utilizando C1.

A	=	8	1	9	2		+	1	4												
		S	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1				
	+	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	→	8	1	9	2
		0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0					
		0	0	1	0	0	0	0	0	0	0	0	0	1	1	1	0	→	8	2	0	6
B	=	8	2	0	6	-	1	2	3	7												
			1	1	1										1	1	1					
	+	0	0	1	0	0	0	0	0	0	0	0	0	1	1	1	0	→	8	2	0	6
		1	1	1	1	1	0	1	1	0	0	1	0	1	0	1	0	→	1	2	3	7
		0	0	0	1	1	0	1	1	0	0	1	1	1	0	0	0					
																	1					
		0	0	0	1	1	0	1	1	0	0	1	1	1	0	0	1	→	6	9	6	9

ISBN: 978-631-6557-81-0

C	=	-	A	-	9	9	3														
	1	1	1	1	1	1	1	1	1	1	1	1									
	+	1	1	0	1	1	1	1	1	1	1	1	1	0	0	0	1	→ -	8	2	0	6
		1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	→ -		9	9	3
		1	1	0	1	1	1	0	0	0	0	0	0	1	1	1	1					
																	1					
		1	1	0	1	1	1	0	0	0	0	0	1	0	0	0	0	→ -	9	1	9	9

Enunciado: Calcular el C2 de -8459 - 589

A	=	-	8	4	5	9	-	5	8	9											
		S	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1					
	1	1	1	1	1	1	1	1	1	1	1	1		1	1	1						
	+	1	1	1	1	1	1	0	1	1	0	1	1	0	0	1	1	→ -	8	4	5	9
		1	1	0	1	1	1	1	0	1	1	1	1	0	1	0	1	→ -		5	8	9
		1	1	0	1	1	1	0	0	1	0	1	0	1	0	0	0	→ -	9	0	4	8

Tareas complementarias.

1. Elabora un mapa conceptual de las eras de desarrollo de las computadoras.

2. Haz una línea de tiempo de los dispositivos electrónicos que han evolucionado hasta nuestros días.

3. Efectúa un ensayo de 2 páginas sobre la vida de Von Neumann.

4. Determina las principales características de tu PC, laptop o smartphone.

5. Resuelve los siguientes ejercicios:

    a) Convierte a binario, octal y hexadecimal, el número 502,562510

    b) Convierte a binario, octal y decimal, el número F1D9,AE416

ISBN: 978-631-6557-81-0

c) Resuelve el C2 de:

A=212+1234; B=A-897; y C=-A -B.

d) Resuelve el C1 de:

A=3987+915; B=-A-678; y C=-A – B.

6. Desarrolle una calculadora en cualquier lenguaje de programación que realice el Complemento a 1, el Complemento a 2 y las conversiones binarias, octales, decimales y hexadecimales entre números enteros y decimales, visualizando en detalle cada proceso y los resultados intermedios.

ISBN: 978-631-6557-81-0

ISBN: 978-631-6557-81-0

CAPÍTULO 2

2 ORGANIZACIÓN FUNCIONAL

En este capítulo se exploran los componentes más importantes y las funcionalidades de un sistema de cómputo. Esto incluye el mainboard, la unidad de procesamiento, la memoria y los dispositivos de entrada y salida, así como las interfaces de comunicación. Además, se detalla que la organización funcional es la manera en que todas las partes internas se coordinan para ejecutar instrucciones y almacenar datos, lo que permite realizar operaciones tanto sencillas como complejas.

2.1 Estructura de un sistema de cómputo

La estructura de un sistema de cómputo se basa principalmente en cómo interactúan el hardware y el software para que el equipo funcione correctamente. El hardware está formado por todos los componentes físicos que se pueden ver y tocar (el gabinete, la placa base, el procesador, la memoria, los dispositivos de almacenamiento, el teclado, el monitor o la impresora). Cada uno de estos elementos tiene una función clara: algunos procesan información, otros la almacenan y otros permiten la entrada y la salida de datos. Sin embargo, el hardware necesita software para funcionar. El software abarca dos elementos principales: el sistema operativo y las aplicaciones. El primero se encarga de controlar el uso de los recursos del computador y de coordinar el trabajo entre los distintos componentes, y las aplicaciones permiten al usuario realizar tareas concretas como programar, escribir, comunicarse o analizar datos. En conjunto, esta interacción entre hardware y software convierte el sistema de cómputo en una herramienta útil y flexible, capaz de adaptarse a las necesidades del usuario (Jara Castro, 2015).

Dentro de su estructura, la placa base es el punto de conexión entre los componentes esenciales. Su diseño, conocido como form factor, puede variar según el tipo de equipo y determina tanto su tamaño como la distribución de los conectores. Sobre esta placa se instalan elementos fundamentales, como el microprocesador (CPU), la memoria RAM y diversos dispositivos de almacenamiento (discos duros, unidades de estado sólido,

ISBN: 978-631-6557-81-0

memorias USB y tarjetas de memoria). Cada uno tiene sus propias características de velocidad, durabilidad e interfaz de conexión. En conjunto, todos estos componentes forman una estructura funcional compleja pero bien organizada que permite que los procesos se ejecuten de forma continua y coordinada en el computador. En la Figura 2.1 se muestra la organización funcional de un sistema de cómputo basado en una CPU, memoria principal y un subsistema de E/S.

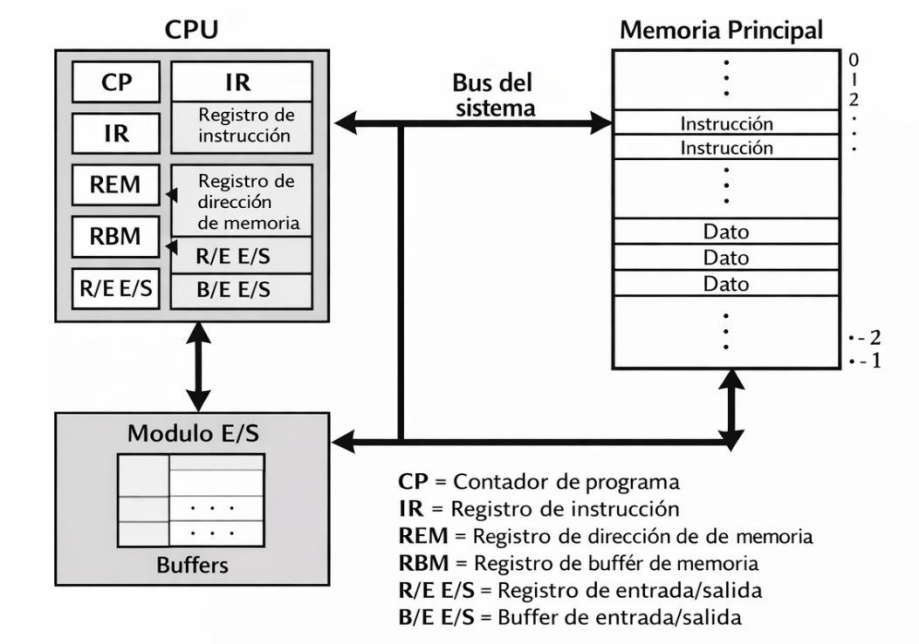


Figura 2.1 Modelo general de la arquitectura de una computadora

Fuente: Adaptado de (Stallings, 2019)

En la arquitectura básica de un computador, hay varios componentes que trabajan juntos para que los programas se ejecuten correctamente. El Program Counter (PC) es un registro muy importante porque almacena la dirección de la siguiente instrucción que la CPU debe ejecutar; básicamente le dice al procesador "qué sigue", y esa dirección se envía luego al registro de direcciones de memoria para acceder a la instrucción correcta. La Unidad Aritmética-Lógica (ALU) es la parte de la CPU encargada de realizar cálculos, desde operaciones simples como sumas y restas hasta comparaciones, operaciones lógicas

ISBN: 978-631-6557-81-0

y desplazamientos de bits, muy comunes en el bajo nivel. Por otro lado, la Unidad de Control se encarga de coordinar todo el proceso, ya que genera las señales de control necesarias para mover datos, ejecutar instrucciones y comunicarse con los dispositivos de entrada y salida; sin esta unidad, cada parte del sistema funcionaría de forma desordenada.

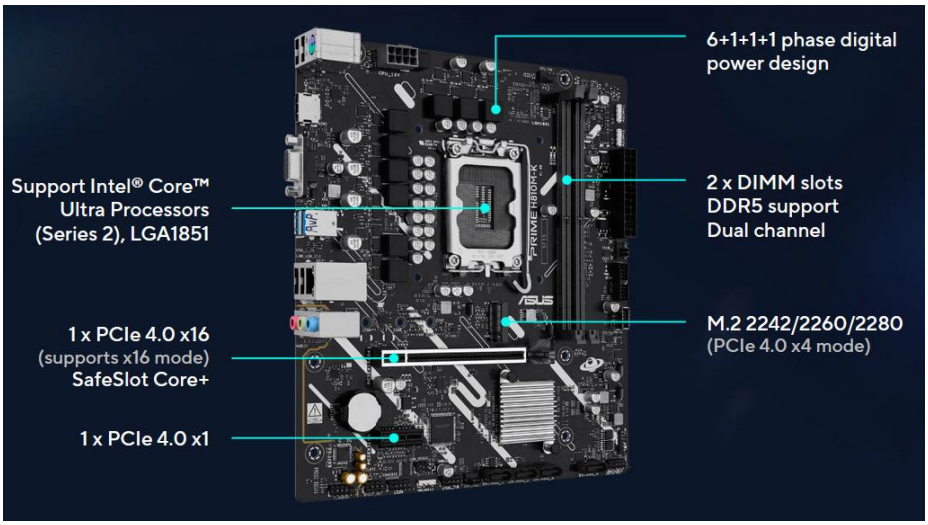
El sistema de Entrada/Salida permite que la computadora se comunique e interactúe con el mundo exterior, ya sea recibiendo datos mediante dispositivos de entrada (típicamente el teclado o el ratón) o mostrando los resultados del procesamiento a través de dispositivos de salida (como el monitor o la impresora). Aunque cada uno de estos dispositivos tiene una función específica, todos ellos trabajan juntos para que el sistema operativo funcione correctamente y de manera eficiente (Stallings, 2019).

2.1.1 Mainboard o placa base

El mainboard, también llamado placa base o motherboard, es la tarjeta principal que permite que todos los componentes de un sistema de cómputo trabajen de forma coordinada. Es como el "esqueleto electrónico", compuesto por circuitos, buses y conectores, que permite que las distintas partes del computador se comuniquen entre sí (Figura 2.2). Su funcionalidad principal es proporcionar la infraestructura eléctrica y lógica para que cada componente reciba energía, pueda enviar y recibir datos y funcione conforme a los estándares del sistema.

Además, el mainboard integra el chipset, un conjunto de controladores que gestionan la comunicación entre el CPU, la memoria RAM, los gráficos y los demás dispositivos. También determina la compatibilidad del sistema, ya que define qué tipo de procesador se puede instalar, qué cantidad y tipo de memoria admite, así como las interfaces de conexión disponibles (USB, SATA, PCIe, DisplayPort, entre otras).

ISBN: 978-631-6557-81-0



a)

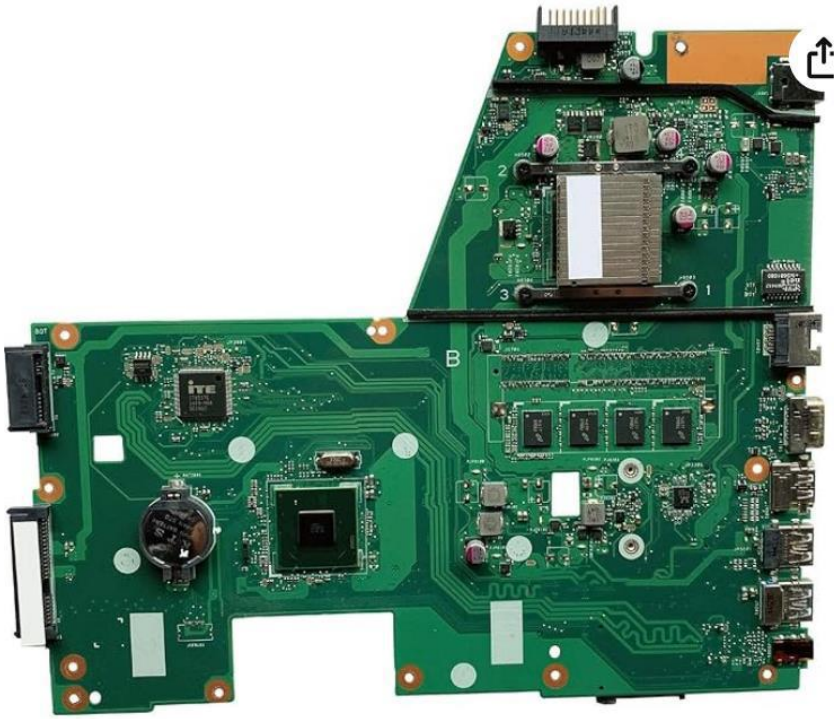


Figura 2.2. a) Mainboard ASUS PRIME H810M-K-CSM.

b) Mainboard de una laptop HP.

Fuente: https://www.asus.com/motherboardscomponents/motherboards/csm/prime-h810m-k-csm/

ISBN: 978-631-6557-81-0

El factor de forma (form factor) es una especificación que ayuda a determinar el tamaño físico interno y se obtiene a partir de la forma de la placa base, sus dimensiones y su posición de anclaje. Este estándar define dónde se ubican elementos clave como el procesador, las ranuras de memoria, los conectores de alimentación y los puertos de expansión, y también determina con qué tipo de gabinete y con qué fuente de alimentación es compatible la placa, lo cual facilitará el ensamblaje de un computador que se adapte a las necesidades del usuario. Con el paso de los años, los form factors se han adaptado a las necesidades de espacio, rendimiento y eficiencia. En los años 80 surgió el formato AT (Advanced Technology), que marcó un hito importante en el desarrollo del PC (Wessels & Weaver, 2008).

Luego, durante la década de los 90, aparecieron los estándares LPX (Low Profile Extended) y NLX (New Low Profile Extended), pensados para equipos más compactos, que optimizaron la ventilación y la accesibilidad interna. Posteriormente, el estándar BTX (Balanced Technology Extended) surgió como una propuesta orientada a mejorar la distribución térmica y la circulación del aire en el sistema. Su diseño físico influyó en el rendimiento y estabilidad del PC.

Actualmente, los form factors más conocidos son los de ATX (Advanced Technology Extended), que estableció los estándares de diseño para computadoras de escritorio, y sus variantes reducidas, como Mini-ATX y Micro-ATX, ofrecen un tamaño más compacto sin perder funcionalidad esencial. Cada uno de estos factores representa una evolución en la organización física y funcional de los sistemas de cómputo modernos. En la Tabla 2.1 se presentan algunas especificaciones de los tipos de placa base.

2.1.2 Carcasa o Case

La carcasa, también conocida como case, es la estructura que sostiene y protege todos los componentes internos del equipo. Comúnmente fabricados en plástico, acero o aluminio, estos gabinetes se diseñan en diferentes estilos y tamaños (form factors), lo que influye en la organización y compatibilidad de los componentes internos. Además de servir

ISBN: 978-631-6557-81-0

como soporte físico, ayuda a la gestión térmica, ya que incorpora aberturas, ventiladores y guías de aire que permiten mantener los componentes a temperaturas adecuadas (Figura 2.3). También ayuda a minimizar los riesgos asociados a la electricidad estática, lo que proporciona un entorno más seguro y estable para el hardware. Las computadoras de escritorio comunes están disponibles en los siguientes formatos: Horizontal case, Full-Size Tower, Compact Tower y All-in-one. En la tabla 2.2 se presentan algunos factores a tener en cuenta al comprar las carcasas.

Formato de Placa Base	Año de Tecnología	Dimensiones (cm)	Aplicaciones
ATX 	1995 (Intel)	30.5 × 24.4	Estación de trabajo, sistemas embebidos, PC de cine en casa, automatización industrial, entre otros.
Micro ATX 	1997 (Intel)	24.4 × 24.4	Multimedia, centros de entretenimiento, sistemas de firma digital, etc.
Mini ITX 	2001 (VIA)	17 × 17	PC de cine en casa, sistemas de información, etc.

ISBN: 978-631-6557-81-0

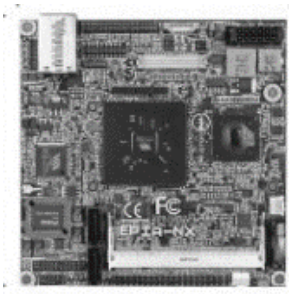
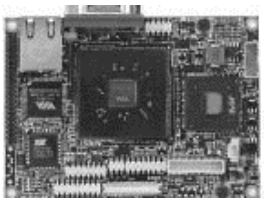
Nano ITX 	2003 (VIA)	12 × 12	Automatización industrial, sistemas de información, etc.
Pico ITX 	2007 (VIA)	9.9 × 7.1	IoT, sistemas embebidos, etc.

Tabla 2.1 Principales form factor desarrollados en la industria

Elaborado por el autor

Figura 2.3 Case de PC Gaming

Fuente: https://www.corsair.com

ISBN: 978-631-6557-81-0

Factores	Descripción
Modelo	El tipo de placa madre determina el tipo de chasis. El tamaño y la forma deben coincidir exactamente.
Tamaño	Si una computadora tiene muchos componentes, necesitará más espacio para el flujo de aire que mantenga el sistema frío.
Espacio disponible	Los gabinetes o case de escritorio pueden ahorrar espacio en áreas reducidas, ya que el monitor podría ubicarse sobre la unidad. El diseño del gabinete puede limitar la cantidad y el tamaño de los componentes que pueden añadirse al computador.
Fuente de poder	La fuente de poder debe coincidir en capacidad y tipo de conexión con el mainboard elegido.
Apariencia	Existen muchos diseños de gabinetes para elegir si es necesario tener una apariencia estética específica.
Pantalla	Los indicadores LED en la parte frontal del gabinete muestran si el sistema recibe energía, cuándo se usa el disco duro, y cuándo la computadora está en reposo o suspensión.
Ventilación	Todos los gabinetes tienen una ventilación en la fuente de poder. Algunos tienen más ventilaciones para disipar el calor.

Tabla 2.2 Principales criterios para seleccionar un case de computador.

2.1.3 Fuente de poder

Uno de los elementos importantes es la fuente de poder, encargada de transformar la corriente alterna (AC) proveniente del enchufe en corriente directa (DC), necesaria para alimentar todos los dispositivos internos del equipo. Esta energía viaja a través de cables y conectores especialmente diseñados para encajar con precisión en cada componente, evitando errores de conexión y asegurando un suministro estable. Por esta razón, es fundamental manipular los elementos del hardware con cuidado, sin aplicar fuerza excesiva, ya que cada pieza está fabricada para ajustarse únicamente en la posición correcta dentro del sistema.

Existen varios formatos de fuentes de alimentación, como el AT (Advanced Technology), utilizado en sistemas más antiguos. Posteriormente, surgió el ATX (AT Extended), que modernizó la tecnología y mejoró la compatibilidad con nuevas tecnologías. Actualmente, el ATX12V es el modelo más común en el mercado, especialmente en computadoras de escritorio, y ofrece una mayor eficiencia energética. A diferencia de otras fuentes más comunes, el EPS12V (Entry-Level Power Supply) fue diseñado inicialmente para servidores de red, pero hoy en día se ha expandido a modelos

ISBN: 978-631-6557-81-0

de escritorio de gama alta, ya que ofrecen más potencia para soportar tarjetas gráficas avanzadas y múltiples unidades de almacenamiento (Figura 2.4). En conjunto, estos formatos garantizan que los sistemas de cómputo operen de manera eficiente y estable (Tabla 2.3).

Figura 2.4 Fuente de poder de PC escritorio y laptop, respectivamente.

Fuente:https://www.amazon.com/Voltage-100-240V-Computer-Supplies-Non-Modular/dp/B0CNJQ36H6.

Tipo de Equipo	Tipo de Power Supply	Rango de Potencia (W)	Factor de Forma	Descripción
Desktop	ATX	400–1200W	ATX	El más común en PCs de escritorio
Desktop	SFX	300–700W	SFX	Versión compacta para equipos pequeños
Desktop	TFX	200–350W	TFX	Usado en desktops delgados
Laptop	Adaptador Externo	45–240W	Cargador	Transforma AC a DC fuera del equipo
Laptop	USB-C PD	30–140W	USB-C	Cargadores modernos compatibles con Power Delivery
Laptop	Propietario (Dell/HP/Lenovo)	45–240W	Varía	Conectores específicos según fabricante

Tabla 2.3 Tipos de fuente de poder según el equipo.

La fuente de alimentación incluye diversos tipos de conectores, diseñados específicamente para cada dispositivo que debe energizar. Entre los más importantes se

ISBN: 978-631-6557-81-0

encuentran: a) el conector ranurado ATX de 24 pines, utilizado para alimentar la placa madre, b) EPS (4 + 4) de 4 a 8 pines, que suministran energía al CPU, c) PCIe de 6 u 8 pines, necesarios para tarjetas gráficas de alto rendimiento, d) conector SATA, que proporciona alimentación a unidades como discos duros, unidades de estado sólido y algunas unidades ópticas, e) Molex, que aún se utiliza en algunos ventiladores y dispositivos antiguos (Figura 2.5).

Cada conector de la fuente de poder se distingue por su forma y el voltaje que suministra, lo cual es vital para su uso correcto. Los principales voltajes que brindan son 3,3 V, 5 V y 12 V, cada uno destinado a funciones específicas dentro del computador. Los primeros voltajes se utilizan en la mayoría de los circuitos digitales de la placa madre como chipset y memorias, mientras que los 5V se emplean para puertos USB y algunos dispositivos de almacenamiento y lógica interna del mainboard. Finalmente, 12 V se emplean en dispositivos que requieren mayor potencia, como ventiladores, tarjetas gráficas, bombas de refrigeración líquida y motores de unidades de disco. El VRM (Voltage Regulator Module) se encarga de transformar y regular el voltaje que llega desde la fuente de alimentación para que los componentes lo utilicen de forma segura. Mediante este mecanismo, el VRM puede reducir los voltajes a valores bajos, como 1 V, ajustándolos en tiempo real para adaptarse a la carga de trabajo del procesador (Kunysz et al., 2016).

Figura 2.5 Principales conectores de la fuente de poder

Fuente: Gemini

ISBN: 978-631-6557-81-0

2.1.4 Socket y CPU

El socket del CPU es el punto de conexión físico y eléctrico entre el procesador y la placa base. Este define la compatibilidad, los límites de energía y hasta las posibilidades de actualización. Existen sockets con pocas decenas de contactos y otros muy complejos que superan fácilmente los mil pines, según la potencia y las necesidades del procesador. En los diseños PGA (Pin Grid Array), los pines estaban en el propio CPU y se insertaban en el socket, algo común en procesadores más antiguos o en ciertas plataformas específicas. En cambio, los LGA (Land Grid Array), más usados hoy, colocan los pines en el socket y dejan al procesador con superficies de contacto simples, lo que reduce el riesgo de daño al CPU y mejora la precisión eléctrica (Figura 2.6).

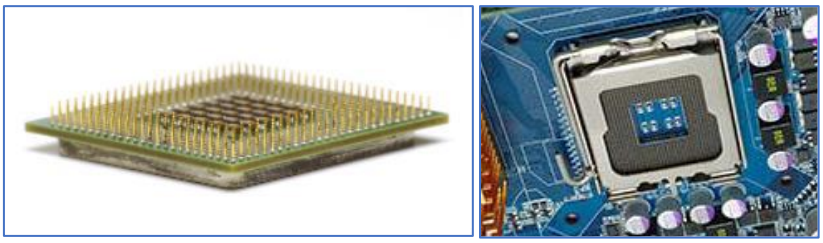


Figura 2.6 a) CPU para socket PGA; b) socket LGA para procesador Intel.

Fuente: Gemini

La evolución de los sockets de CPU ha sido un proceso constante por la necesidad de mayor rendimiento y eficiencia. En las primeras etapas, los procesadores se conectaban a la placa madre mediante zócalos simples, como el Socket 1 de Intel, utilizado en los procesadores 80486. Con el avance tecnológico, los CPU se volvieron más potentes y complejos y los sockets se adaptaron a estos cambios. Un ejemplo claro es el Socket 7, utilizado en procesadores Pentium, que marcó un hito al ofrecer mayor flexibilidad y rendimiento. Más adelante, Intel introdujo el Socket 478 y el LGA775, diseñados para CPUs de doble núcleo con mejores mecanismos de refrigeración. En generaciones recientes, los sockets han seguido evolucionando; el LGA 1200 o el AM4 de AMD han permitido integrar CPUs de última generación, con un enfoque particular en la

ISBN: 978-631-6557-81-0

compatibilidad con nuevas tecnologías, como el soporte para memoria DDR5, las interfaces PCIe de alta velocidad y un uso más eficiente de la energía. La tabla 2.4 resume dicha evolución.

Nombre Socket	Usado por CPU Family	Descripción
Socket 7	Intel Pentium, AMD K6	Socket antiguo compatible con múltiples proveedores, 321 pines.
Socket 370	Intel Pentium III, Celeron	Socket PGA para CPU Intel de finales de la década de 1990.
Socket A (462)	AMD Athlon, Duron	Socket PGA ampliamente utilizado para CPU AMD a principios de la década de 2000.
Socket 478	Intel Pentium 4	Socket PGA con 478 pines para la arquitectura NetBurst.
LGA 775	Intel Core 2 series	Primer Socket LGA convencional de Intel.
Socket 754	Athlon 64, Sempron	Primeros chips AMD con controlador de memoria integrado.
Socket 939	Athlon 64, Athlon 64 X2, Opteron	Plataforma popular para entusiastas; soporte dual channel.
AM2	AMD Athlon 64, Phenom	Socket PGA compatible con memoria DDR2.
AM3	AMD Phenom II, FX	Socket PGA compatible con versiones anteriores y compatible con DDR3.
FM1/FM2/FM2+	APUs Llano, Trinity, Kaveri	Diseño orientado a APUs con GPU integrada potente.
LGA 1156	Intel Core i3/i5/i7 (1st gen)	Controlador PCIe y de memoria integrado.
LGA 1155	Intel Core (2nd–3rd gen)	Utilizado por las CPU Sandy Bridge e Ivy Bridge.
LGA 1150	Intel Core (4th–5th gen)	Compatible con Haswell y Broadwell.
AM4	Ryzen 1000 → Ryzen 5000	Uno de los sockets con más soporte histórico; DDR4 y PCIe 4.0.
LGA 1151	Intel Core (6ª a 9ª gen)	Popular por su uso masivo; versiones con chipsets distintos.
LGA 1200	Intel Core (10th–11th gen)	Suministro de energía actualizado, Comet/Rocket Lake.
LGA 1700	Intel Core (12th–14th gen)	Compatible con arquitectura híbrida (Alder/Lake).
AM5	AMD Ryzen 7000–9000	Socket LGA compatible con DDR5 y PCIe 5.0.
LGA 1851	Intel Core (2024–2025)	Sucesor del LGA1700 con diseño térmico y de alimentación mejorado.
AM5+	Future AMD Ryzen 5/6	Evolución prevista del AM5 con compatibilidad ampliada.

Tabla 2.4 Desarrollo y cambios en los sockets de procesadores

Elaborado por el autor

En las computadoras de escritorio, el socket del CPU está diseñado para ofrecer un alto rendimiento, ya que dispone de mayor espacio físico para integrar disipadores de calor y ventiladores más grandes o incluso sistemas de enfriamiento líquido. Esto permite que los procesadores de escritorio operen a altas velocidades y mantengan un rendimiento sostenido incluso bajo cargas de trabajo intensivas (videojuegos, edición de video o ejecución de máquinas virtuales). Además, las plataformas de escritorio suelen facilitar la actualización del CPU, ya que los sockets y las placas base están diseñados para admitir una variedad de modelos y generaciones de procesadores. Por esta razón, las computadoras

ISBN: 978-631-6557-81-0

de escritorio son la opción preferida para usuarios que buscan potencia, escalabilidad y personalización.

En cambio, en las computadoras portátiles, el concepto de socket casi desaparece en la práctica, ya que la mayoría de las CPU vienen soldadas directamente a la placa base mediante técnicas como BGA (Ball Grid Array). Este enfoque reduce el tamaño, mejora la eficiencia energética y facilita diseños más delgados, pero también elimina la posibilidad de reemplazar o actualizar el procesador.

2.1.5 Sistema de enfriamiento (cooling systems)

Los sistemas de enfriamiento mantienen los componentes a una temperatura segura durante su operación. Cada vez que la CPU ejecuta instrucciones, especialmente en tareas intensivas como la compilación de código o la ejecución de videojuegos, se genera calor debido al consumo de energía. Si ese calor no se controla, el procesador reduce su velocidad automáticamente o incluso puede apagarse para evitar daños. Por eso, lo más común es encontrar un disipador de calor acompañado de un ventilador, lo que se conoce como enfriamiento activo. El disipador absorbe el calor del procesador y el ventilador se encarga de mover el aire caliente hacia afuera, lo que ayuda a mantener un rendimiento estable durante más tiempo.

Las tarjetas gráficas (GPU) siguen el mismo principio, pero suelen necesitar soluciones más potentes porque generan aún más calor. Es habitual ver GPUs con dos o tres ventiladores, e incluso con sistemas propios de control térmico. En equipos de alto rendimiento, como estaciones de trabajo o PCs gaming avanzadas, el enfriamiento líquido se presenta como una alternativa. En este caso, un bloque metálico se coloca sobre el procesador y un líquido circula por él, llevando el calor hacia un radiador, donde se disipa con ayuda de ventiladores. Aunque es más caro y algo más complejo de instalar, este sistema resulta muy eficiente y silencioso (Figura 2.7).

ISBN: 978-631-6557-81-0

Figura 2.7 Sistemas de enfriamiento

Fuente: https://primetechsupport.com/products/gaming-pc-hardware-upgrades-and-computer-parts-sales

2.1.6 Memorias

Los computadores usan distintos tipos de chips de memoria que almacenan la información en bloques, según cómo se cargue eléctricamente la celda del chip. Esta forma básica de almacenar datos es común en cualquier tipo de memoria, ya sea simple o muy avanzada. Entre las memorias más conocidas están la ROM (Read Only Memory), que almacena datos de forma permanente, y la RAM (Random Access Memory), que funciona como un espacio temporal en el que la CPU trabaja con programas y datos mientras la máquina está encendida. La RAM es volátil, lo que significa que pierde todo su contenido al apagar el computador. Cuando un equipo tiene poca RAM, se siente lento y, por eso, agregar más suele mejorar bastante el rendimiento. Claro, siempre existe un límite marcado por la placa madre, que determina la cantidad máxima que se puede instalar.

Los sistemas de memoria pueden clasificarse por su ubicación: interna dentro del CPU (como los registros), principal (en la memoria RAM) o secundaria, como ocurre con unidades externas (Figura 2.8). Cada nivel cumple un rol diferente dentro de la jerarquía

ISBN: 978-631-6557-81-0

de memoria: los registros son extremadamente rápidos, pero con capacidad limitada; la RAM equilibra tamaño y velocidad; y las memorias secundarias ofrecen gran capacidad, pero menor velocidad. Esta forma de organización ayuda a que el sistema funcione sin cuellos de botella.

Otra forma de clasificar la memoria es según el método de acceso. Algunas trabajan de manera secuencial, como las cintas magnéticas, en las que hay que avanzar paso a paso hasta llegar al dato buscado. Otras permiten el acceso directo mediante direcciones físicas, como los discos duros. Y la más rápida de todas es la memoria de acceso aleatorio, la RAM o la caché, que puede acceder a cualquier dato casi al instante.

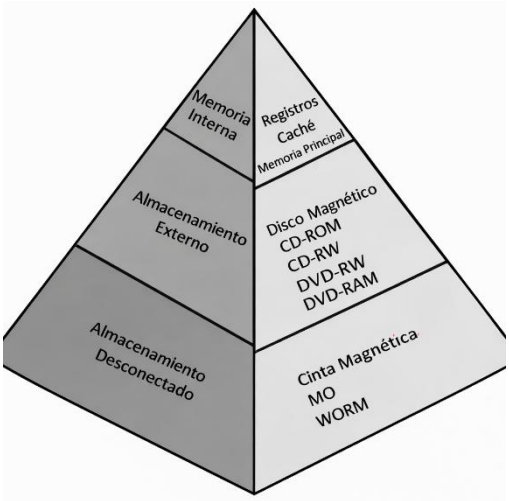


Figura 2.8 Organización jerárquica de la memoria y los dispositivos de almacenamiento.

Fuente: Stallings et al. (2006)

Esta diferencia en cómo se accede a la información es clave para entender por qué unas tecnologías sobreviven y otras quedan para usos más específicos. En cuanto a las memorias semiconductoras, pueden ser volátiles o no volátiles y borrables o no borrables. La RAM, que permite la lectura y la escritura eléctricas por bytes, es volátil y se utiliza para el almacenamiento temporal. La familia ROM incluye variantes como PROM,

ISBN: 978-631-6557-81-0

EPROM y EEPROM, así como las memorias flash, que pueden borrarse mediante electricidad o incluso con luz ultravioleta, según el tipo. Cada una de estas apareció para cubrir necesidades distintas, como facilitar actualizaciones o guardar firmware de forma segura. En la Tabla 2.5 se presentan las categorías y sus descripciones (Stallings et al., 2006).

Categoría	Descripción / Ejemplos
Ubicación	Registros (CPU), Memoria Principal (RAM), Memoria Secundaria (SSD/HDD), Memoria Externa (USB, Nube)
Capacidad	Tamaño de palabra (32/64 bits), Número total de palabras o celdas
Método de acceso	Secuencial (cintas), Directo (discos), Aleatorio (RAM, caché)
Dispositivo físico	Semiconductores (RAM, Flash), Magnético (HDD), Óptico (CD/DVD), Estado sólido (SSD)
Características físicas	Volátil / No volátil, Borrable / No borrable
Velocidad	Latencia y tiempo de acceso (Registros > Caché > RAM > SSD > HDD)
Consumo energético	Bajo (SSD, RAM), Medio (HDD), Alto (memorias antiguas)

Tabla 2.5 Características y clasificación de los sistemas de memoria en un sistema de cómputo

Elaborado por el autor

Memoria RAM

La memoria RAM es uno de los componentes clave de un computador, porque almacena temporalmente los datos y programas que el procesador necesita para ejecutar acciones. Se trata de una memoria volátil, lo que significa que pierde su contenido cuando al equipo no se le suministra energía. Gracias a esta memoria, el sistema operativo y las aplicaciones pueden ejecutarse de manera fluida, ya que el procesador no tiene que buscar los datos constantemente en dispositivos de almacenamiento más lentos. En términos simples, mientras más RAM tenga un equipo, mejor podrá manejar varias tareas al mismo tiempo, aunque esto también depende del resto del hardware. En la memoria RAM existen principalmente dos tipos: SRAM y DRAM (Figura 2.9). La SRAM (Static RAM) almacena cada bit mediante flip-flops y no necesita refresco; por eso es muy rápida y suele usarse

ISBN: 978-631-6557-81-0

como memoria caché, aunque su construcción es más compleja y costosa. En cambio, la DRAM (Dynamic RAM) es más lenta que la SRAM, pero más barata y densa; por eso se utiliza como memoria principal en computadoras de escritorio y laptops (Tabla 2.6).

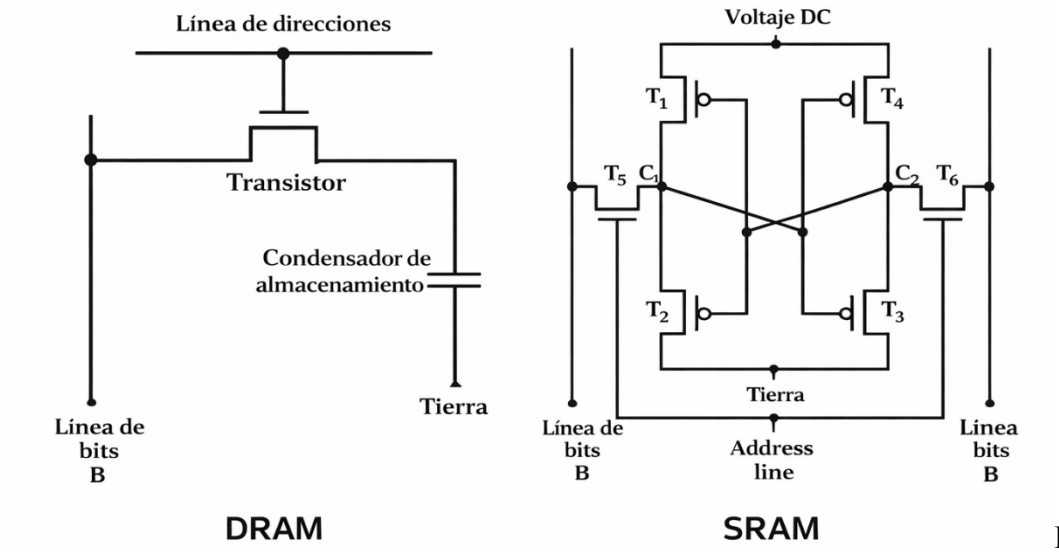


Figura 2.9 Organización básica de celdas de memoria DRAM y SRAM

Fuente: (Tanenbaum, 2016)

Característica	DRAM	SRAM
Tipo de memoria	Dinámica	Estática
Volatilidad	Volátil	Volátil
Estructura básica	1 transistor + 1 capacitor	Varios transistores (flip-flops)
Necesita refresco	Sí, periódicamente	No
Velocidad	Más lenta	Más rápida
Densidad	Alta	Baja
Costo	Más económica	Más costosa
Consumo energético	Menor por bit, mayor por refresco	Mayor por bit
Uso principal	Memoria RAM principal	Memoria caché (L1, L2, L3)
Capacidad típica	Alta (GB)	Baja (KB–MB)
Complejidad de diseño	Simple	Más compleja
Ubicación	Módulos DIMM/SO-DIMM	Dentro del CPU

Tabla 2.6 Diferencias clave entre DRAM y SRAM.

Elaborado por el autor

ISBN: 978-631-6557-81-0

Módulos de Memoria RAM

Las ranuras de memoria RAM son los espacios físicos de la placa base donde se conectan los módulos de memoria principal del computador. A estas ranuras también se las conoce como bancos de memoria y cumplen una función clave, ya que permiten que el procesador acceda rápidamente a los datos e instrucciones. A lo largo del tiempo han existido distintos tipos de conectores, que han ido cambiando según la generación del hardware. Por ejemplo, los primeros módulos fueron los SIP (Single In-line Package), usados en equipos con procesadores 286, en los que los chips de memoria venían soldados sobre una pequeña placa con pines individuales. Luego aparecieron los SIMM (Single Inline Memory Module), que ya tenían los conectores en el borde del módulo, y se usaron en sistemas 386 y 486, lo que mejoró la facilidad de instalación y la capacidad.

En una etapa inicial se popularizaron los módulos SIMM de 72 contactos, que eran físicamente más largos y tenían una ranura en el centro para evitar colocarlos al revés, algo bastante común en aquellos años. Estos módulos usaban memorias DRAM como FPM (Fast Page Mode) o EDO (Extended Data Out), que ofrecían capacidades de entre 8 MB y 64 MB. Luego aparecieron los módulos DIMM (Dual In-line Memory Module) de 168 pines, que operaban con un bus de 64 bits y soportaban mayores frecuencias, como 66, 100 y 133 MHz. Además, incorporaron memoria SDRAM (Synchronous DRAM), que era mucho más eficiente y rápida que las DRAM anteriores, lo que permitió aumentar tanto el rendimiento como la capacidad total del sistema. De forma paralela, aparecieron los módulos RIMM (Rambus Inline Memory Module), desarrollados con la tecnología Rambus y promovidos por Intel (principios del 2000); aunque ofrecían excelentes prestaciones, su alto costo y complejidad hicieron que no fueran ampliamente aceptados, siendo reemplazados por tecnologías más económica como DDR.

Los módulos DIMM DDR (Double Data Rate) han evolucionado con el tiempo para responder a las mayores exigencias de rendimiento de los computadores. La primera versión de DDR permitió transferir datos dos veces por ciclo de reloj, superando a SDRAM tradicional. Luego apareció DDR2, que aumentó las velocidades de bus y redujo el

ISBN: 978-631-6557-81-0

consumo de energía, aunque con una ligera mayor latencia. Con la llegada de DDR3, se dio un salto significativo en la densidad de memoria y la velocidad, menor voltaje y una mejor eficiencia general; por lo que fue muy utilizado tanto en computadoras de escritorio como en portátiles durante varios años. Más adelante, DDR4 introdujo un ancho de banda superior, un menor consumo de energía y una mayor capacidad por módulo, para manejar el procesamiento moderno. Finalmente, DDR5 marca el estándar actual, ya que duplica el ancho de banda respecto a su antecesora e integra la gestión de energía directamente en el módulo (PMIC). Esto permite manejar mayores volúmenes de datos, lo cual es clave para aplicaciones modernas como la virtualización, los videojuegos y el procesamiento intensivo de datos (Figura 2.10 y tabla 2.7).

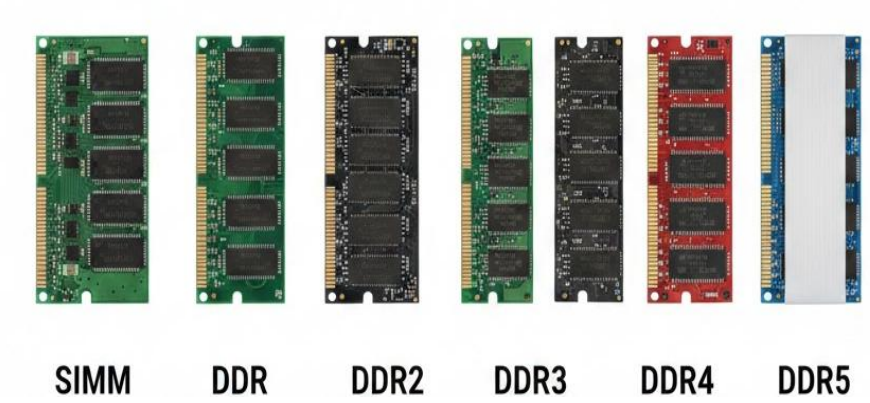


Figura 2.10 Módulos de memoria RAM.

Fuente: Gemini

Módulo	Frecuencia típica (MHz)	Velocidad (MT/s)	Voltaje (V)	Año de aparición
DDR	200–400	200–400	2,5	2000
DDR2	400–800	400–800	1,8	2003
DDR3	800–2133	800–2133	1,5	2007
DDR4	2133–3200	2133–3200	1,2	2014
DDR5	4800–8400+	4800–8400+	1,1	2020

Tabla 2.7 Detalles de las Memorias DDR

Elaborado por el autor

ISBN: 978-631-6557-81-0

La velocidad de la memoria RAM influye directamente en el rendimiento general del sistema. Si la RAM es lenta, el procesador pasa más tiempo esperando datos y eso se nota en aplicaciones pesadas o cuando se trabaja con varias tareas a la vez. En computadoras de escritorio, lo habitual es encontrar módulos DIMM, con mayor capacidad de almacenamiento y frecuencias más altas. En los sistemas portátiles se utilizan los módulos SODIMM (Small Outline DIMM), una versión más compacta diseñada para ahorrar espacio. Aunque cumplen la misma función, no son intercambiables entre sí. Por eso, al ampliar la RAM de una PC o portátil, se debe verificar la capacidad y el tipo exacto de módulo compatible (Figura 2.11).

Otro aspecto por analizar es el manejo de errores de memoria, que pueden aparecer por fallos eléctricos, desgaste del hardware o interferencias, y aunque no son frecuentes, pueden provocar bloqueos o corrupción de datos. Las memorias sin paridad no realizan comprobaciones, mientras que las memorias con paridad añaden un bit adicional para detectar errores simples, aunque no pueden corregirlos. Las memorias ECC (Error-Correcting Code) van un paso más allá, ya que pueden detectar errores de varios bits y corregir automáticamente los más simples sin que el sistema se detenga. Por esta razón se usan sobre todo en servidores y estaciones de trabajo, donde la fiabilidad es clave. En cualquier caso, el usuario puede consultar cuánta memoria RAM tiene instalada durante el arranque del sistema (POST), desde la BIOS o directamente en el sistema operativo.

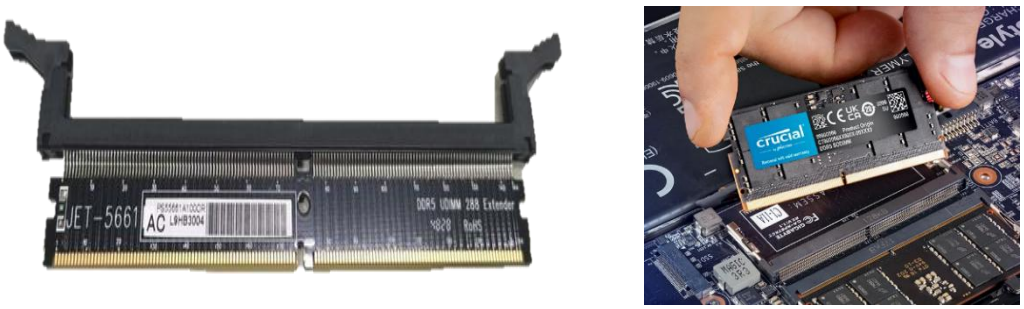


Figura 2.11 DIMM y SODIMM

Fuente: https://mfactors.com/jet-5661ac-ddr5-udimm-extender-4800mhz/

ISBN: 978-631-6557-81-0

Memoria Caché

La memoria caché es una memoria semiconductora cuyo objetivo principal es reducir al máximo el tiempo de acceso a los datos que requiere el procesador. Para lograrlo, utiliza tecnología SRAM (Static RAM); gracias a esta, la caché puede almacenar y leer información casi de forma inmediata, lo que permite que la CPU no tenga que esperar constantemente a la memoria principal. Esta memoria puede estar integrada directamente dentro del procesador o ubicada en módulos cercanos, y suele ser entre cinco y seis veces más rápida que la RAM dinámica. Sin embargo, su precio es elevado, hasta veinte veces mayor que la RAM por la misma cantidad de almacenamiento, por lo que se usa en tamaños reducidos y solo para los datos más críticos.

La memoria caché está organizada en varios niveles según su proximidad al CPU y su velocidad. El nivel L1 es el más rápido y próximo al CPU, ya que se encuentra dentro de cada núcleo del CPU y responde en tiempos de acceso mínimos; por esta razón, su tamaño es muy pequeño, normalmente alrededor de 32 KB para la caché de datos y 32 KB para la caché de instrucciones (total de 64 KB por núcleo). Esto puede variar según la arquitectura del procesador. El nivel L2, anteriormente conocido como caché externa, hoy en día está integrado en el procesador, aunque no siempre en cada núcleo, y ofrece capacidades que van desde 512 KB, 1 MB hasta varios megabytes por núcleo. En arquitecturas modernas, el nivel L3 aparece y sirve como una caché compartida que apoya a los núcleos y ayuda a reducir los accesos a la memoria principal cuando los datos no se encuentran en L1 o L2, con tamaños de entre 8 MB y 64 MB, o incluso mayores. Finalmente, el nivel L4 no es muy común y puede encontrarse en pocos procesadores que trabajan de forma compartida entre todos los núcleos. Actúa como apoyo adicional cuando se requiere manejar grandes volúmenes de datos con mayor eficiencia. En la Figura 2.12 se muestra la información de la memoria caché de una laptop.

ISBN: 978-631-6557-81-0

ROM y BIOS UEFI

La memoria ROM es un tipo de memoria diseñada para almacenar información básica y crítica del computador, que debe estar siempre disponible, incluso cuando el equipo está apagado. A diferencia de la RAM, que pierde su contenido al cortar la alimentación, la ROM no es volátil. Su función principal es almacenar las primeras instrucciones que se ejecutan al encender el equipo, encargadas de comprobar el hardware y preparar el sistema para cargar el sistema operativo. En los primeros computadores, esta memoria venía en chips físicos soldados o conectados a la placa base (en formato DIP) y no podía modificarse fácilmente. Hoy en día, en cambio, suele implementarse como memoria flash (EEPROM), lo que permite actualizar su contenido mediante software, por ejemplo, al actualizar el BIOS o el firmware.

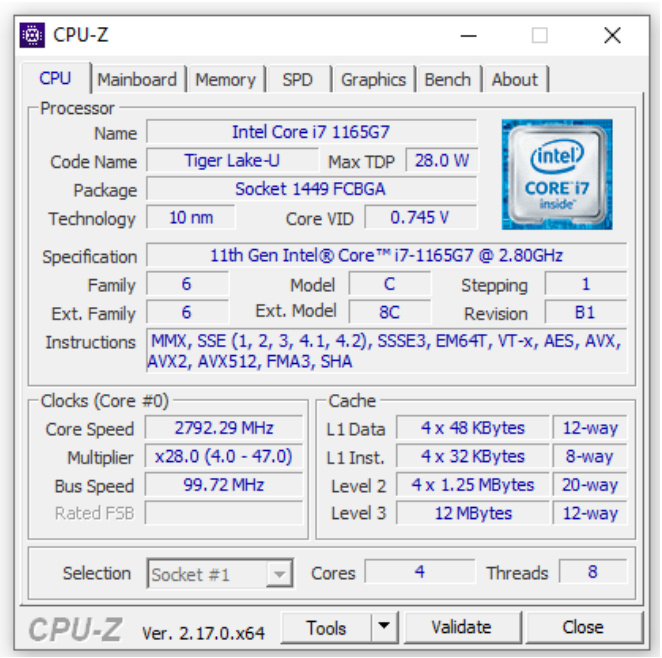


Figura 2.12 Uso de CPU-Z para visualizar los niveles de la memoria caché de un laptop.

En la memoria ROM se almacena el **BIOS** (Basic Input and Output System), un pequeño programa que realiza varias funciones importantes durante el arranque del computador. Una de sus primeras tareas es ejecutar la **rutina POST** (Power On Self Test),

ISBN: 978-631-6557-81-0

que verifica que los componentes principales (memoria, procesador, GPU y dispositivos esenciales) funcionen correctamente para cargar el sistema operativo. Los códigos POST son señales que utiliza la computadora para indicar qué ocurre durante el proceso de arranque, justo al encender el equipo. Se pueden presentar de varias formas; a) pitidos (beep codes), en los que una combinación de sonidos largos y cortos indica el tipo de falla detectada, por ejemplo, un error de memoria o de video; b) en placas base más recientes, especialmente de gama media o alta, es común encontrar pantallas LED de dos dígitos o luces indicadoras (LEDs) en la motherboard, que muestran códigos alfanuméricos específicos o registros de eventos en el firmware UEFI. Cada fabricante (ASUS, Gigabyte, MSI, entre otros.) tiene su propia tabla de códigos, por lo que es necesario consultar el manual de la placa base para interpretar correctamente el mensaje.

Otra funcionalidad de la BIOS es el conocido **SETUP**, que mediante un menú de configuración permite al usuario ajustar parámetros (el orden de arranque, la fecha y la hora del sistema o ciertas opciones del hardware). Estas configuraciones se guardan en una memoria llamada **CMOS** (Complementary Metal-Oxide-Semiconductor), que se mantiene activa gracias a una pequeña batería en la placa base. Cada vez que el equipo se enciende, el BIOS lee los datos del CMOS para configurar el sistema. En relación con esto, aparece el concepto de Clear CMOS, un procedimiento que consiste en borrar la memoria CMOS y restaurar la configuración del BIOS o UEFI a sus valores de fábrica. Esto se produce, por ejemplo, cuando el computador no arranca debido a una configuración incorrecta o tras un intento fallido de overclocking. El Clear CMOS puede realizarse mediante un jumper, un botón específico en la placa madre o retirando temporalmente la batería (Figura 2.13).

En los computadores actuales, el BIOS clásico ha sido reemplazado en gran medida por UEFI (Unified Extensible Firmware Interface), un firmware que actúa como intermediario entre el hardware y el sistema operativo; este reside en memoria no volátil, similar a la ROM. UEFI funciona en sistemas de 32 y 64 bits, permite arrancar desde discos de gran capacidad (más de 2,2 TB) e incorpora mejoras importantes en seguridad, como

ISBN: 978-631-6557-81-0

Secure Boot, que evita que se cargue software malicioso o controladores no firmados durante el arranque del sistema.

Figura 2.13 Formas de realizar el Clear CMOS

Fuente:https://hardzone.es/tutoriales/mantenimiento/clear-cmos-placa-base-procedimiento/

2.1.7 Chipset

El chipset es un conjunto de circuitos integrados diseñados según la arquitectura de un procesador específico y cuya función principal es actuar como puente de comunicación entre la CPU y el resto de los componentes del sistema (memoria RAM, tarjetas de expansión, puertos USB, teclado, ratón y otros dispositivos). Sin este conjunto de circuitos, los distintos elementos del computador no podrían coordinarse entre sí. Por esta razón, el chipset define en gran medida las capacidades y limitaciones de una placa base, como el tipo de memoria que admite o las interfaces disponibles.

Durante muchos años, el chipset fue el corazón organizador de la placa base y estaba dividido en dos chips bien diferenciados: el Northbridge y el Southbridge. El Northbridge se ubicaba físicamente cerca del procesador porque se encargaba de las tareas más críticas y rápidas del sistema, como la comunicación directa con la memoria RAM y la tarjeta gráfica (por ejemplo, mediante AGP en arquitecturas más antiguas), además de servir como puente hacia el resto de la placa. Este chip mantenía un intercambio constante de datos con la CPU a través de un bus de 64 bits, operando a frecuencias que, según la generación, podían ir desde los 400 MHz hasta rozar el 1 GHz, algo bastante alto para su época (figura 2.14).

ISBN: 978-631-6557-81-0

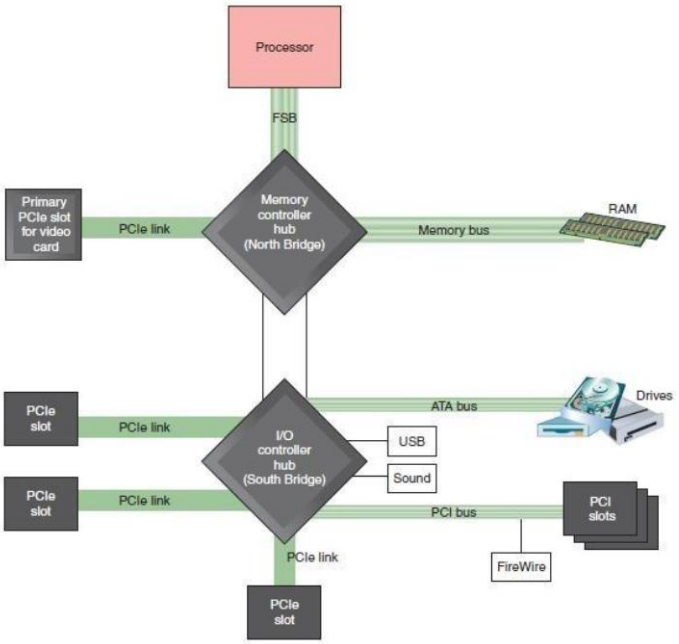


Figura 2.14 Modelo clásico de chipset con Northbridge y Southbridge.

Fuente: Gemini

Con el paso del tiempo y la evolución de los procesadores, muchas de estas funciones se fueron integrando directamente en la CPU, especialmente el controlador de memoria, lo que hizo que el Northbridge dejara de existir como chip independiente. Aun así, su concepto es clave para entender cómo se organizaba el hardware y por qué los sistemas actuales son más compactos y eficientes. Por otro lado, el Southbridge estaba orientado al control de los dispositivos de entrada y salida (también conocido como I/O Controller Hub). Este chip gestionaba elementos como el bus PCI, los controladores IDE o SATA, los puertos USB, el sonido, la red Ethernet, el soporte RAID y el sistema de interrupciones, entre otros. También cumple funciones importantes relacionadas con la administración de energía, el reloj en tiempo real y el acceso al BIOS/UEFI. Aunque en las placas base actuales el Southbridge ha evolucionado o se ha integrado parcialmente en otros componentes, su concepto sigue siendo fundamental para entender cómo se organizan y comunican los dispositivos dentro de un sistema de cómputo (Figura 2.15).

ISBN: 978-631-6557-81-0

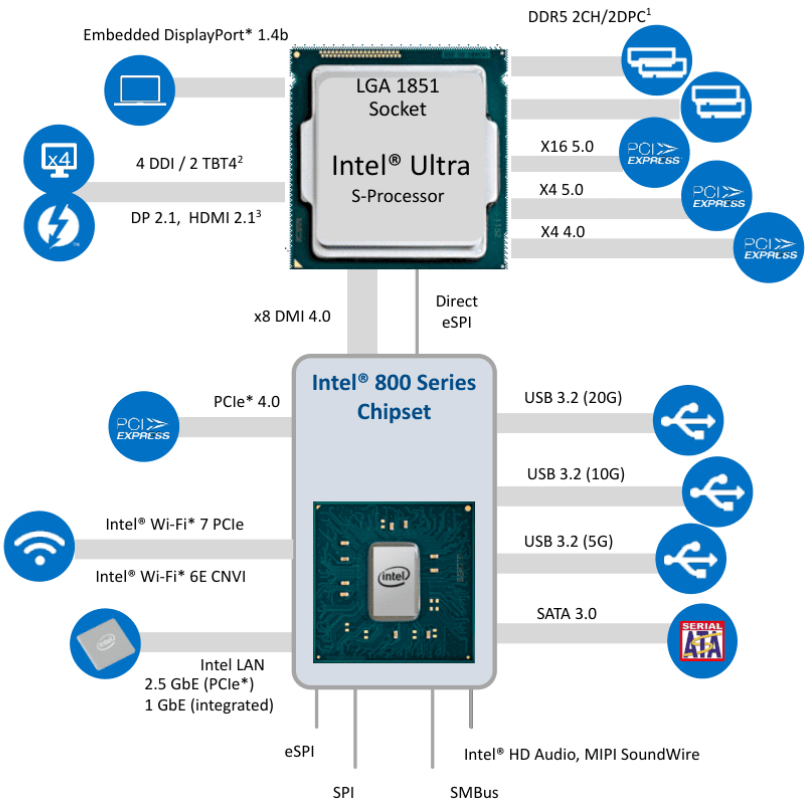


Figura 2.15 Estructura del chipset Intel 800 Series y flujo de comunicación.
Fuente: https://www.hardwareluxx.de/community/threads/intel-lga1851-mainboard-laberthread-z890-w880-q870-b860-h810.1354258/

De forma similar a lo que ocurre con los microprocesadores, el mercado de los chipsets está concentrado en pocos fabricantes. VIA y NVIDIA, SiS, ITE o Maxwell, que en su momento ofrecieron alternativas muy utilizadas en placas base para equipos de escritorio y estaciones de trabajo. Hoy en día, Intel y AMD han tenido un papel dominante durante años, ya que diseñan chipsets específicamente para sacar el mejor provecho de sus propios procesadores. Esta concentración se debe al alto costo de diseñar un chipset y a la necesidad de compatibilidad precisa con el hardware, algo que solo unas pocas empresas pueden mantener a lo largo del tiempo.

ISBN: 978-631-6557-81-0

2.1.8 Buses de conexión

Durante muchos años, el bus fue la principal forma de interconectar los componentes de un computador. En esencia, un bus es un medio de comunicación compartido que permite que el procesador, la memoria y los dispositivos de entrada/salida se comuniquen entre sí. Todos los dispositivos conectados "escuchan" ese mismo medio, lo que simplifica el diseño, pero también introduce una limitación importante: solo uno puede transmitir a la vez. Si dos lo hacen al mismo tiempo, los datos se mezclan y se pierde información. Por eso, aunque hoy en día las computadoras de propósito general usan más conexiones punto a punto, los buses siguen siendo muy comunes en sistemas embebidos y microcontroladores, donde el diseño sencillo y el bajo costo siguen siendo prioritarios.

Un bus no debe entenderse como un solo cable, sino como un conjunto de líneas que trabajan en conjunto y cada una transporta bits (0 o 1). Al agrupar varias líneas, se pueden enviar varios bits en paralelo. Por ejemplo, un bus de datos de 32 bits puede transferir todos estos bits de forma simultánea, lo que resulta más eficiente. En un sistema de cómputo existen varios tipos de buses, pero el más importante es el bus del sistema, que conecta los componentes principales, como el procesador, la RAM y otros dispositivos. De forma conceptual, este bus se organiza en tres grupos: bus de datos, bus de direcciones y bus de control (Figura 2.16). Los buses de datos transportan información útil, como datos e instrucciones. Es bidireccional y la información circula desde la memoria o los dispositivos hacia el procesador cuando realiza una operación de lectura, y desde el procesador hacia la RAM o los dispositivos cuando se escriben resultados.

Las líneas de dirección indican de dónde vienen o a dónde van los datos, ya sea una posición de memoria o un puerto de entrada/salida. Cuantas más líneas de dirección haya, mayor será la memoria que el sistema puede manejar y tiene sentido unidireccional (CPU hacia memoria/dispositivos). Cada vez que la CPU necesita leer o escribir un dato, primero coloca una dirección en este bus. Esa dirección funciona como una referencia clara: le indica al sistema exactamente dónde debe realizarse la operación.

ISBN: 978-631-6557-81-0

Por su parte, las líneas de control coordinan la gestión del proceso, indicando si se trata de una lectura o escritura, si el dato ya fue aceptado, o si un dispositivo está solicitando el uso del bus. Señales como lectura de memoria, escritura de E/S o interrupciones permiten que todos los módulos trabajen de forma ordenada. En la práctica, el funcionamiento del bus sigue una secuencia bien definida: primero, un módulo solicita el bus; luego, espera a que se le conceda y recién entonces puede enviar o recibir datos. Combina líneas bidireccionales y unidireccionales según el proceso a gestionar.

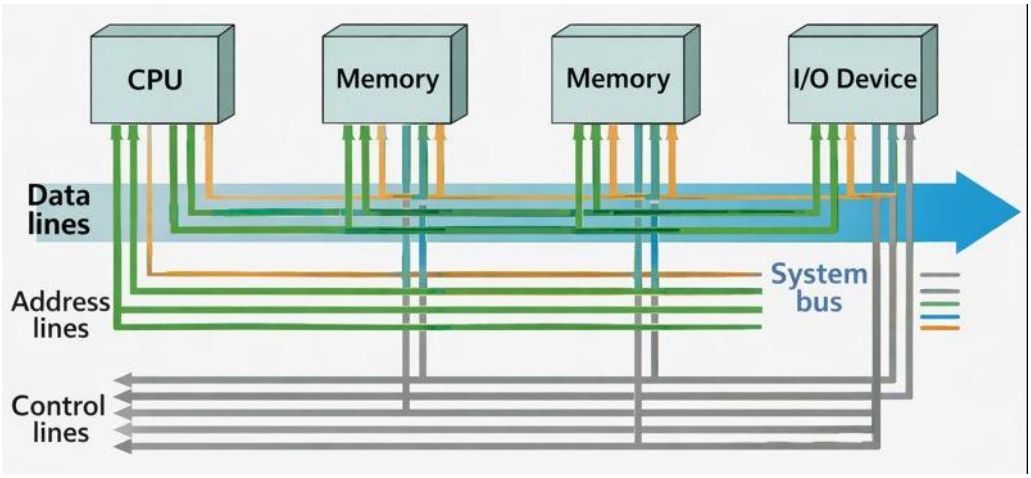


Figura 2.16 Líneas de comunicación interna de un sistema de cómputo

Fuente: Gemini

Buses de datos

En los primeros computadores personales, la comunicación entre componentes se realizaba principalmente mediante buses de datos paralelos. En este tipo de bus, varios bits se transmiten simultáneamente, cada uno por un conductor distinto. Por ejemplo, si el bus es de 8 bits, se necesitan al menos 8 líneas físicas solo para los datos. Durante muchos años esto funcionó bien, como en los antiguos puertos de impresora, pero con el tiempo empezaron a notarse sus límites. Más cables implican un mayor costo, más espacio ocupado y una mayor probabilidad de fallos físicos. Además, aunque los bits se envíen "a la vez", no siempre llegan exactamente al mismo tiempo, y eso empieza a ser un problema cuando se busca mayor velocidad. Esa desviación en la llegada de las señales, conocida

ISBN: 978-631-6557-81-0

como skew, se debe a pequeñas diferencias en la longitud o en las características eléctricas de los cables. A bajas velocidades no importa demasiado, pero cuando se intenta aumentar la frecuencia del bus, estas diferencias se vuelven críticas. Llega un punto en el que el sistema ya no puede distinguir con claridad qué bits pertenecen a un mismo dato. A esto se suma otra limitación importante: la mayoría de los buses paralelos son half-duplex, es decir, solo pueden enviar datos en un sentido a la vez, a menos que se dupliquen las líneas. Todo esto hace que escalar un bus paralelo resulte complicado y poco eficiente.

Los buses de datos seriales transmiten bit a bit a frecuencias muy altas. A primera vista parece un proceso poco eficiente y lento, pero en la práctica ocurre lo contrario. Los buses seriales modernos usan señalización diferencial, en la que la información se representa como la diferencia de voltaje entre dos conductores muy similares y próximos entre sí. Este diseño permite cancelar gran parte del ruido eléctrico del entorno, ya que las interferencias suelen afectar a ambos conductores con la misma intensidad. El receptor mide la diferencia absoluta de voltaje entre ellos, lo que mejora significativamente la fiabilidad de la transmisión.

Este enfoque permite que un bus serial transfiera miles de millones de bits por segundo con muy pocos cables y conexiones sencillas. Además, se pueden utilizar varios enlaces seriales en paralelo, (conocidos como lanes) para aumentar el ancho de banda total. En los sistemas actuales, la comunicación entre el procesador y otros dispositivos se basa en este modelo. Interfaces como PCI Express, SATA, M.2, USB o Thunderbolt están basadas en enlaces seriales independientes y full-duplex, es decir, pueden enviar y recibir datos simultáneamente. Esto simplifica el diseño del hardware y permite alcanzar velocidades antes difíciles de imaginar.

2.1.9 Slot de expansión

Los slots de expansión son ranuras físicas ubicadas en la placa base que permiten conectar tarjetas adicionales para ampliar las funciones del computador, como tarjetas de video, de red o de sonido. Estas ranuras son de plástico e incluyen conectores eléctricos

ISBN: 978-631-6557-81-0

que permiten la comunicación entre la tarjeta y el sistema. A lo largo del tiempo, han existido distintos tipos de slots, cada uno asociado a una tecnología específica y fácilmente reconocible por su tamaño, forma o incluso por su color. Por ejemplo, las ranuras ISA (Industry Standard Architecture) fueron de las primeras en utilizarse en las computadoras personales; operaban a bajas velocidades, alrededor de 8 MHz, y ofrecían un rendimiento muy limitado, adecuado solo para los primeros sistemas PC. Más adelante apareció VLB (VESA Local Bus), una tecnología de transición usada principalmente en procesadores 486, que mejoraba el ancho de banda respecto a ISA, aunque su uso fue breve debido a problemas de compatibilidad y estabilidad.

Evolución	Bus de sistema	Tipo de bus	Data path (bits)	Líneas de direccionamiento	Frecuencia del bus	Throughput aproximado
ISA	ISA	Paralelo	16	24	8 MHz	16 MB/s
VLB	VESA Local Bus	Paralelo	32	32	33–40 MHz	160 MB/s
PCI	PCI	Paralelo	0,5	32	33 MHz	133–266 MB/s
PCI-X	PCI-X	Paralelo	64	36	66–133 MHz	1 GB/s
PCIe 1.0	PCI Express	Serial	1 lane	Serial	2.5 GT/s	250 MB/s por línea
PCIe 2.0	PCI Express	Serial	1 lane	Serial	5 GT/s	500 MB/s por línea
PCIe 3.0	PCI Express	Serial	1 lane	Serial	8 GT/s	~1 GB/s por línea
PCIe 4.0	PCI Express	Serial	1 lane	Serial	16 GT/s	~2 GB/s por línea
PCIe 5.0	PCI Express	Serial	1 lane	Serial	32 GT/s	~4 GB/s por línea
PCIe 6.0	PCI Express	Serial	1 lane	Serial	64 GT/s	~8 GB/s por línea

Tabla 2.8 Evolución de los Slots de Expansión del Computador.

Elaborado por el autor

Con la evolución del hardware surgieron soluciones más eficientes como PCI (Peripheral Component Interconnect), que se convirtió en un estándar durante muchos años. Las ranuras PCI ofrecían mayores velocidades de transferencia, una mejor organización del bus y soporte para dispositivos de 32 y 64 bits, además de operar a distintos voltajes según el dispositivo. Sin embargo, el crecimiento de las necesidades gráficas y de procesamiento llevó al desarrollo de PCI Express, que reemplazó el bus compartido por un sistema de líneas dedicadas, lo que permitió mayor velocidad y escalabilidad. PCI Express mantiene compatibilidad con PCI a nivel de software, pero ofrece un rendimiento muy superior, ya que cada dispositivo dispone de su propio ancho de banda. Gracias a configuraciones como x1, x4, x8 o x16, este tipo de ranura se adapta

ISBN: 978-631-6557-81-0

a distintos dispositivos y hoy es la base de tarjetas gráficas modernas y de otros componentes de alto rendimiento dentro de la arquitectura del computador. En la Tabla 2.8 se resume la evolución mencionada.

2.1.10 Almacenamiento

El almacenamiento del PC ha sido, desde hace muchos años, una parte clave de la arquitectura del computador, y uno de los dispositivos más representativos es el disco duro, o Hard Disk Drive (HDD). Este componente es un dispositivo de almacenamiento no volátil, que graba de forma magnética información digital en uno o varios platos rígidos que giran a gran velocidad dentro de una carcasa metálica cerrada. En cada cara de estos platos se ubican los cabezales de lectura y escritura; cualquier contacto físico podría dañar la información. Cuando el sistema necesita leer o guardar datos, el cabezal se mueve hasta una pista y un sector específicos, donde se realiza la operación correspondiente (Figura 2.17). Aunque en la actualidad existen tecnologías más modernas, el disco duro tradicional sigue siendo importante para entender cómo se almacenan y se recuperan los datos en el hardware.

Figura 2.17 Componentes internos de un HDD.

Fuente:https://www.researchgate.net/figure/Figura-211-Partes-del-disco-duro_fig1_356904531

ISBN: 978-631-6557-81-0

Uno de los factores que influyen en el rendimiento de un disco duro es el tiempo medio de acceso, que indica cuánto tarda el disco en localizar un dato y usarlo. Este tiempo no es único, sino que se obtiene al sumar varios factores: el tiempo medio de búsqueda, el tiempo medio de lectura o de escritura y la latencia media. El tiempo de búsqueda corresponde al desplazamiento del cabezal hasta la pista correcta, mientras que la latencia depende de la rotación del disco hasta que el sector deseado pasa bajo el cabezal. Todos estos valores determinan la rapidez con la que se accede a la información almacenada.

Otro parámetro importante es la velocidad de rotación, expresada en revoluciones por minuto (RPM). A mayor velocidad de giro, menor será la latencia media, ya que el sector deseado tarda menos en posicionarse. También destaca la tasa de transferencia, que indica la velocidad con la que los datos se envían desde el disco a la computadora una vez que el cabezal está correctamente ubicado. Para que el disco duro pueda comunicarse con el resto del sistema, necesita una interfaz que actúe como medio de conexión con la placa base. Entre las primeras interfaces se encuentra ATA (Advanced Technology Attachment) también conocida como IDE (Integrated Drive Electronics), ampliamente utilizada en computadoras personales durante muchos años por su sencilla forma de implementar. En cambio, SCSI (Small Computer System Interface) se diseñó para entornos más exigentes, como servidores y clústeres de almacenamiento, lo que permitió mayores prestaciones en términos de velocidad y la conexión de varios discos a un mismo controlador.

Luego surgieron tecnologías más eficientes como SATA (Serial ATA), que utiliza un bus serie y ofrece mayores velocidades de transferencia, menor consumo energético y cada dispositivo tiene su propio canal dedicado hacia el controlador (conexión punto a punto). SATA cuenta con varias versiones, desde SATA 1 con rendimiento real de 150 MB/s, SATA II con 300 MB/s, hasta SATA III, con incrementos notables en el ancho de banda y rendimiento (600 MB/S). También apareció SAS (Serial Attached SCSI), una evolución del SCSI orientada a sistemas profesionales y servidores (velocidad teórica de 12 Gb/s). Finalmente, los SSD representan un cambio importante, ya que eliminan las

ISBN: 978-631-6557-81-0

partes mecánicas y utilizan memoria flash, lo que permite velocidades mucho más altas y tiempos de acceso más cortos, con interface SATA III (Figura 2.18).

Figura 2.18 Conectores para discos duros.
Fuente: Gemini

Unidad de estado sólido

Las unidades de estado sólido, conocidas como SSD (Solid State Drive) son dispositivos de almacenamiento de datos que utilizan memoria flash (celdas de memoria NAND), que les permite ofrecer tiempos de acceso mucho más rápidos. Estas unidades se han convertido en un componente clave de los sistemas de cómputo modernos, tanto en computadoras de escritorio como en laptops. Al no tener partes móviles, los SSD generan menos ruido, consumen menos energía y son menos propensos a fallos por golpes o vibraciones, lo cual es muy importante en equipos portátiles.

Los SSD ofrecen distintos formatos e interfaces, que presentan diferentes capacidades de rendimiento. Los SSD SATA III suelen ser comunes y económicos para sustituir los discos duros tradicionales, mientras que los SSD NVMe (Non-Volatile Memory Express) funcionan sobre el bus PCI Express o ranura M.2, alcanzan velocidades más altas y se usan en equipos de mayor rendimiento, lo que ha facilitado su adopción masiva. Con los discos SSD, el almacenamiento dejó de ser el principal cuello de botella del rendimiento general (Figura 2.19).

ISBN: 978-631-6557-81-0

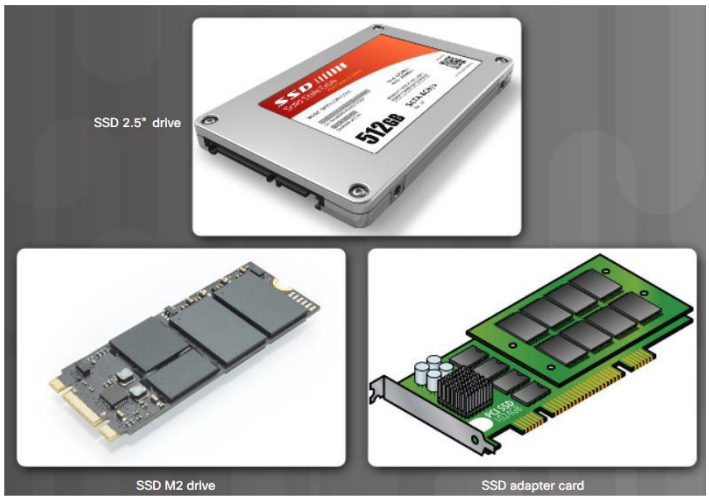


Figura 2.19 Form factor de los SSD

Fuente: Academia de redes CISCO NetAcad

Por su parte, la interfaz M.2 es un estándar moderno que define tanto el tamaño físico del SDD como la forma de conexión al sistema (laptops y computadoras de escritorio). A diferencia de los discos HDD, un SSD M.2 se conecta directamente a la placa base, sin necesidad de cables de datos ni de alimentación, lo que simplifica el montaje, reduce el espacio interno y mejora el flujo de aire dentro del equipo. Además, esta interfaz es muy flexible, ya que puede operar con los buses SATA o PCI Express. Los SSD M.2 se presentan en diferentes tamaños, que siguen un formato estandarizado que combina ancho y largo (con cuatro o cinco dígitos). Existen tamaños como 2230 (22 × 30 mm), 2242 (22 × 42 mm), 2260 (22 × 60 mm), 2280 (22 × 80 mm) y 22110 (22 × 110 mm). Los modelos más cortos suelen encontrarse en laptops muy delgadas o en dispositivos compactos, mientras que los más largos permiten incluir más chips de memoria (mayor capacidad) y, en algunos casos, ofrecen una mejor disipación térmica; el 2280 es el más utilizado en computadoras de escritorio y laptops modernas, la compatibilidad depende de la placa base (Figura 2.20).

ISBN: 978-631-6557-81-0

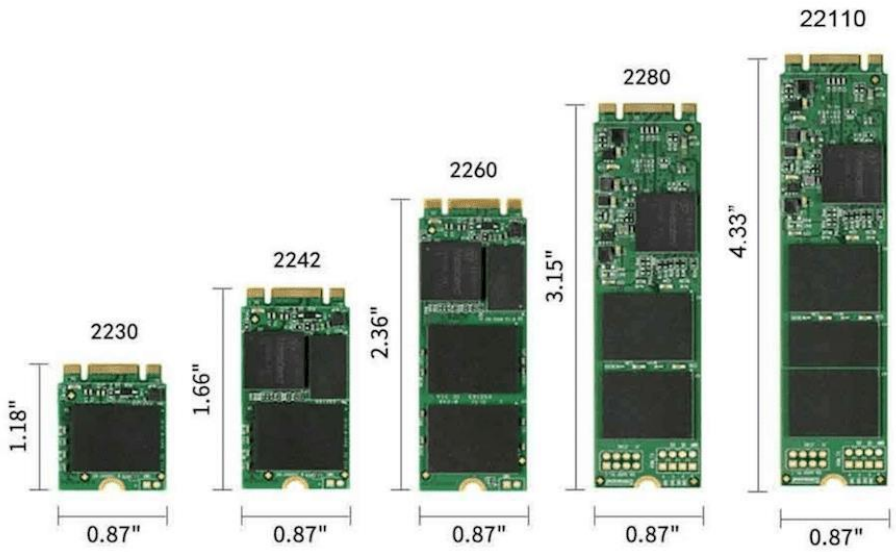


Figura 2.20 Tamaños de SSD con interface M.2 NVMe

Fuente:https://powertech.com.co/Blog/tipos-y-formatos-de-discos-ssd-m-2-pcie-y-nvme/?srsltid=AfmBOop51J1XhVGYQ7b3vk0od4fPfzTrik7_VfAJKIQRgtj7IRF77zPR

La interface U.2, técnicamente llamada SFF-8639, es una solución de conexión utilizada principalmente para unidades de estado sólido (SSD) de alto rendimiento, diseñada para entornos profesionales (servidores y centro de datos). Al combinar el protocolo NVMe sobre el bus PCI Express, ofrece un ancho de banda masivo y baja latencia, ideales para gestionar grandes volúmenes de datos con máxima confiabilidad.

Interfaz	Tipo de conexión	Año aproximado	Velocidad máxima	Tipo de bus	Uso principal	Observaciones
IDE (PATA)	Paralelo	1986	133 MB/s	Paralelo	PCs antiguos	Interfaz obsoleta
SATA I	Serial	2003	150 MB/s	Serial	PC de escritorio	Primera versión SATA
SATA II	Serial	2004	300 MB/s	Serial	PC y laptops	Retrocompatible
SATA III	Serial	2009	600 MB/s	Serial	HDD y SSD	Muy usado
SCSI	Paralelo	1980	320 MB/s	Paralelo	Servidores	Alta confiabilidad
SAS	Serial	2004	12 Gb/s	Serial	Servidores	Evolución de SCSI
NVMe	Serial	2013	Hasta 7 GB/s	PCIe	SSD modernos	Baja latencia
M.2	Serial	2013	Depende de PCIe	PCIe/SATA	Laptops y PCs	Formato compacto
U.2	Serial	2015	32 Gb/s	PCIe	Servidores	Uso profesional

Tabla 2.9 Interfaces de discos duros y unidades de almacenamiento.

Elaborado por el autor

ISBN: 978-631-6557-81-0

Estructura física

El funcionamiento interno de un SSD se basa en celdas de memoria flash, normalmente de tipo NAND, que almacenan los datos en forma de cargas eléctricas, incluso cuando el dispositivo está apagado. Existen varios tipos de celdas según la cantidad de bits que pueden almacenar. Las celdas SLC (Single-Level Cell) almacenan un solo bit por celda, lo que las hace muy rápidas y con una gran durabilidad, aunque su costo es elevado; por eso se usan más en entornos industriales o empresariales. Las MLC (Multi-Level Cell) almacenan dos bits por celda, ofreciendo un equilibrio entre precio, velocidad y resistencia. Las TLC (Triple Level Cell) permiten guardar tres bits en cada celda de memoria y son muy comunes en SSD de consumo (mayor capacidad a menor costo), aunque sacrifican algo de rendimiento y de vida útil. Finalmente, las QLC (Quad Level Cell) almacenan cuatro bits por celda, lo que permite densidades muy altas y precios más bajos, pero con menor durabilidad y un rendimiento más limitado en escrituras prolongadas.

Para su funcionamiento, la memoria NAND flash se organiza en **celdas** (unidad mínima de bits), **página** (grupo de celdas de 4KB a 16KB) y **bloque** (128 o 256 páginas). Se puede leer y escribir a nivel de página, solo se escribe en una página vacía y no se puede borrar una página individualmente. Cuando el SSD necesita almacenar nueva información, primero debe borrar un bloque completo y luego escribir los datos, lo que convierte el proceso en su interior en algo complejo. Para ello, utiliza un controlador (pequeño procesador) encargado de distribuir los datos, corregir los errores y equilibrar el desgaste de las celdas. Este proceso, conocido como wear leveling, permite que el disco tenga una vida útil más prolongada, ya que cada celda puede soportar un número limitado de escrituras.

La métrica TBW (TeraBytes Written) determina la vida útil del SDD, es decir, cuántas veces se puede escribir en el SSD antes de que pierda la capacidad de retener información de forma confiable. Por ejemplo, un SDD de 1 TB tiene un TBW de 600, significa que el fabricante garantiza que se puede escribir 600 TeraBytes de datos antes de

ISBN: 978-631-6557-81-0

que falle. Otra medida de rendimiento es las IOPS (Input/Output Operations per Second) definidas por el fabricante, que indican cuántas tareas de lectura o escritura puede realizar en un segundo. Por ejemplo, un SSD SATA puede tener entre 70.000 - 100.000 IOPS; y un M.2 NVME entre 500.000 – 1.000.000 IOPS.

Arreglo redundante de discos (RAID)

RAID (Redundant Array of Independent Disks) es una tecnología que combina varios discos físicos (HDD o SSD) en una sola unidad lógica y mejora el rendimiento, la seguridad y la protección contra fallos. En los sistemas de almacenamiento actuales, RAID 0 (conocido como striping) es una configuración de discos para maximizar la velocidad de lectura y escritura. Su funcionamiento no duplica los datos, sino que se divide la información en bloques y se reparte entre dos o más discos, escribiendo bloques alternados en cada uno de ellos. Esto mejora el rendimiento, ya que varias unidades trabajan simultáneamente, pero no hay tolerancia a fallos. Si uno de los discos se daña, se pierde todo el contenido. Por eso, RAID 0 se utiliza solo cuando la velocidad es prioritaria y los datos no son críticos, como archivos temporales o entornos de pruebas.

En el extremo opuesto está RAID 1, cuyo objetivo es proteger la información por encima del rendimiento. En este esquema, los datos se copian exactamente en dos discos, lo que se conoce como espejo o "mirroring". Si uno falla, el sistema sigue funcionando con el otro disco sin pérdida de información. Desde la perspectiva del software, esto resulta transparente, ya que el sistema ve un único disco lógico, aunque por debajo haya redundancia. La desventaja es que solo se aprovecha la mitad del espacio total. A pesar de esto, RAID 1 es muy utilizado en servidores pequeños y estaciones de trabajo donde la fiabilidad es más importante que la capacidad o el rendimiento.

Por su parte, RAID 5 busca un equilibrio entre rendimiento, capacidad y tolerancia a fallos. Requiere al menos tres discos y distribuye los datos junto con información de paridad que permite reconstruirlos si falla uno de ellos. La ventaja es que se aprovecha

ISBN: 978-631-6557-81-0

mejor el espacio que en RAID 1 y se mantiene cierta protección. Sin embargo, el cálculo de la paridad genera una sobrecarga, especialmente en las operaciones de escritura. Las arquitecturas modernas y los SSD pueden afectar tanto al rendimiento como a la vida útil del almacenamiento, por lo que RAID 5 debe analizarse con cuidado según el escenario de uso.

Por último, RAID 10 combina ideas de RAID 1 y RAID 0, suele considerarse una solución equilibrada. Primero se crean pares de discos en espejo (RAID 1) y luego se distribuyen los datos entre dichos pares (RAID 0). El resultado es un sistema rápido y tolerante a fallos, capaz de soportar la caída de uno o incluso varios discos, siempre que no pertenezcan al mismo espejo. El punto negativo es el costo: se necesitan al menos cuatro discos y solo se aprovecha el 50 % del espacio. Aun así, en entornos donde el rendimiento y la fiabilidad son críticos, RAID 10 suele ser una de las mejores opciones (Figura 2.21).

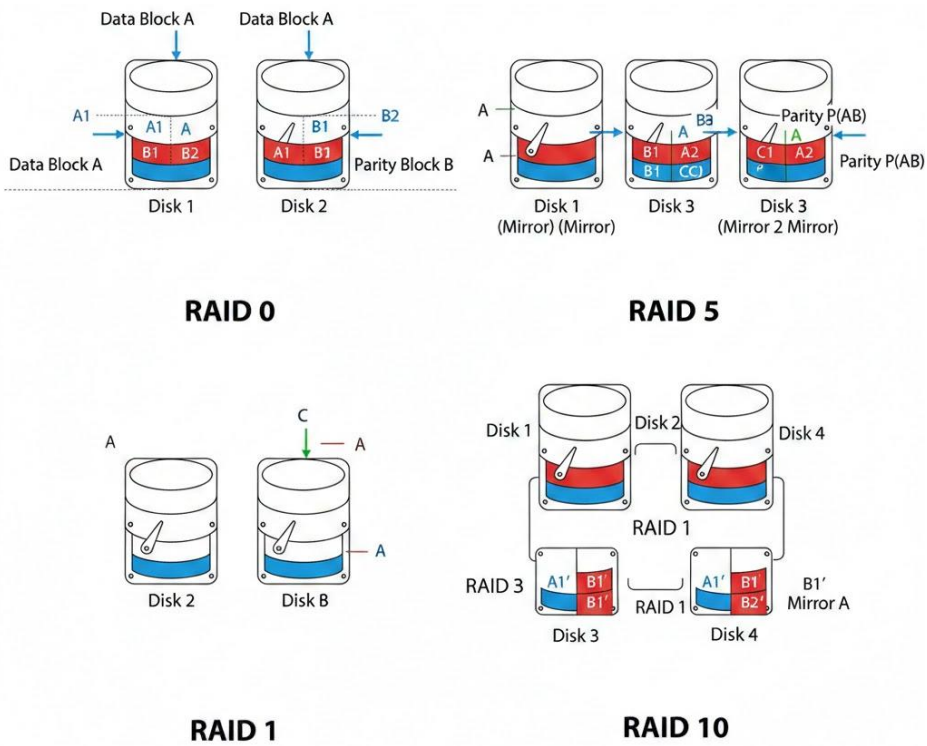


Figura 2.21 Sistemas RAID para discos.

Fuente: Gemini

ISBN: 978-631-6557-81-0

2.1.11 Tarjeta gráfica

La tarjeta gráfica, también conocida como tarjeta de video o adaptador gráfico, es un componente diseñado para procesar datos relacionados con imágenes y videos. Su función principal es recibir los datos que envía la CPU y transformarlos en información visual que pueda mostrarse en un monitor. Al ser una tarjeta de expansión, se instala directamente en la placa base y actúa como un acelerador gráfico, liberando al procesador central de una carga de trabajo muy intensa relacionada con el procesamiento visual (Figura 2.22).

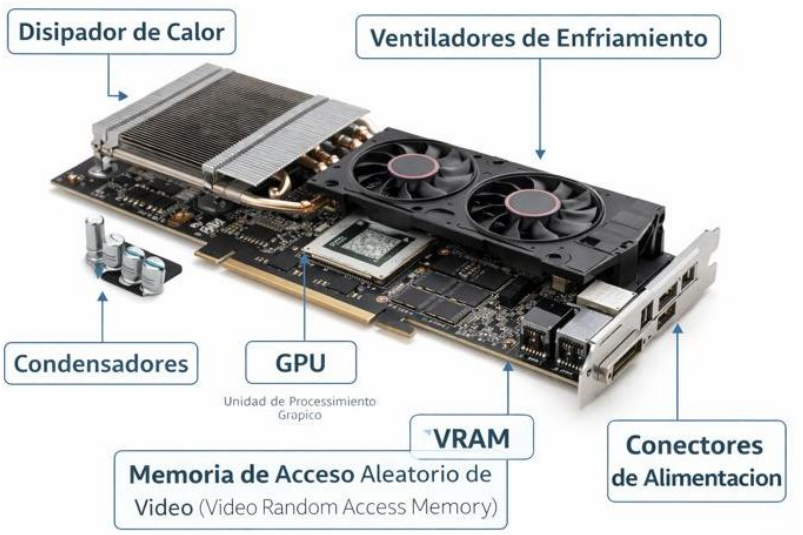


Figura 2.22 Componentes principales de la tarjeta gráfica.
Fuente: Gemini

El componente principal es la GPU (Graphics Processing Unit), que funciona como un procesador especializado (similar a la CPU) y optimizado para realizar cálculos en coma flotante (muy comunes en aplicaciones 3D), lo que permite generar imágenes. Entre sus características más importantes se encuentran la frecuencia de reloj del núcleo, el número de núcleos y shaders, entre otras. Estos elementos trabajan juntos para convertir escenas tridimensionales formadas por vértices y líneas en imágenes bidimensionales compuestas por píxeles, que se observan en la pantalla. Para trabajar con rapidez, la tarjeta gráfica

ISBN: 978-631-6557-81-0

contiene memoria gráfica (VRAM, por sus siglas en inglés, Video Random Access Memory), que almacena temporalmente texturas, imágenes, colores y otros datos visuales que la GPU necesita procesar. Existieron varios tipos de VRAM, como SAM, WRAM, SGRAM, MDRAM y la más utilizada en la actualidad, la GDDR SDRAM, diseñada para ofrecer altas velocidades de transferencia (Tabla 2.10)

Generación	Año de introducción	Velocidad por pin (MT/s)	Capacidad por chip (GB)	Características principales
GDDR2	2003	1500–2000	0.125–0.25	Primer estándar GDDR para gráficos
GDDR3	2004	1600–2000	0.25–0.5	Menor consumo y latencia que GDDR2
GDDR5	2008	3500–7000	0.5–1	Gran salto de rendimiento y eficiencia
GDDR5X	2016	10000–14000	1–2	Optimizada para GPUs de gama alta
GDDR6	2018	12000–16000	1–2	Estándar actual, mejor eficiencia energética
GDDR6X	2020	19000–21000	1–2	Uso de señalización PAM4
GDDR7	2024	24000–30000+	2–4	Mayor ancho de banda y menor consumo

Tabla 2.10 Evolución histórica y características técnicas principales de la memoria gráfica GDDR.

Elaborado por el autor.

En cuanto a los sistemas de interconexión, las tarjetas gráficas han evolucionado junto con las tecnologías de visualización. Las primeras interfaces, como VGA (Video Graphics Array) permitían imágenes simples y resoluciones básicas en monitores antiguos. Posteriormente, surgieron conexiones digitales como DVI (Digital Visual Interface), que mejoraron notablemente la calidad de la imagen en pantallas LCD (Liquid Crystal Display). Más adelante, HDMI (High-Definition Multimedia Interface) incorporó la transmisión conjunta de audio y video en alta definición y se ha vuelto muy popular en equipos modernos y dispositivos multimedia. Finalmente, DisplayPort se consolidó como una interfaz de alto rendimiento, capaz de manejar resoluciones elevadas y múltiples flujos de datos, complementando a HDMI, dominando los actuales sistemas de cómputo (Figura 2.23 y Tabla 2.11).

ISBN: 978-631-6557-81-0

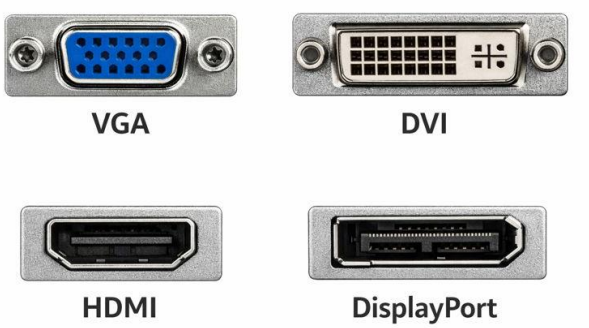


Figura 2.23 Principales interfaces de conexión de video.

Fuente: Gemini

Interfaz de video	Tipo de señal	Resolución típica	Resolución máxima aproximada	Comentarios
VGA	Analógica	1024x768	1920x1080 @ 60 Hz	Interfaz obsoleta, depende del cable
DVI-D	Digital	1920x1080	2560x1600 @ 60 Hz	No transporta audio
DVI-I	Digital / Ana	1920x1080	2560x1600 @ 60 Hz	Compatible con VGA mediante adaptador
HDMI 1.4	Digital	1920x1080	3840x2160 @ 30 Hz	Muy común en televisores
HDMI 2.0	Digital	1920x1080	3840x2160 @ 60 Hz	Buen soporte para 4K
HDMI 2.1	Digital	3840x2160	7680x4320 @ 60 Hz	Soporta 8K y altas tasas de refresco
DisplayPort 1.2	Digital	2560x1440	3840x2160 @ 60 Hz	Muy usado en PCs
DisplayPort 1.4	Digital	3840x2160	7680x4320 @ 60 Hz (con compresión)	Ideal para monitores modernos

Tabla 2.11 Características de las interfaces de video.
Elaborado por el autor.

2.1.12 Interface de Red

Las interfaces de red son los componentes de hardware que permiten que una computadora se comunique con una red. Desde el punto de vista práctico, una red LAN (Local Area Network) es un conjunto de dispositivos digitales (PC, laptop, smartphone, impresoras, consola de juegos, entre otros) que intercambian datos mediante un medio compartido, ya sea por cable o de forma inalámbrica en un área geográfica pequeña (casa, oficina o edificio). En redes WAN (Wide Area Network) enlazadas por grandes distancias suelen ser gestionadas por proveedores de servicios (compañías de telefonía o de cable) que permiten conectarnos a Internet. En un entorno doméstico, el router actúa como dispositivo de interconexión entre el proveedor de Internet y los dispositivos locales. A través de la conectividad Wi-Fi (Wireless Fidelity), permite la interconexión de

ISBN: 978-631-6557-81-0

computadores, laptops y teléfonos móviles de forma sencilla mediante una clave compartida (Figura 2.24).

En una LAN, Ethernet es la tecnología estándar para las conexiones cableadas punto a punto (par trenzado o fibra óptica), en las que cada computadora se conecta a un switch. Las velocidades de Gigabit Ethernet (1000 Mbps) son suficientes para la mayoría de las aplicaciones de software. En cambio, el Wi-Fi prioriza la movilidad, la comodidad y la flexibilidad en dispositivos móviles, ya que elimina cables y permite desplazarse libremente dentro de un espacio, pero se ve afectado por interferencias del entorno y presenta mayores desafíos de seguridad, ya que los datos se propagan a través del aire. Con el avance tecnológico, los estándares Wi-Fi han mejorado en velocidad, eficiencia y protección de datos, hasta el punto de que hoy es normal encontrar esta conectividad integrada directamente en las placas base y la interconexión cableada mediante una tarjeta de red Ethernet.

Figura 2.24 Elementos de interconexión de red.
Fuente: Gemini.

2.1.13 Dispositivos de entrada y salida

Los dispositivos de entrada, salida y E/S son componentes encargados de la comunicación directa entre el usuario y la computadora. Los dispositivos de entrada permiten introducir datos e instrucciones desde el exterior hacia la memoria del sistema. Entre los más comunes se encuentran el teclado, el ratón, el escáner, el micrófono, las

ISBN: 978-631-6557-81-0

cámaras, los lectores, los gamepads, entre otros. Cada uno está diseñado para capturar distintos tipos de información (texto, movimientos, imágenes o sonido) y procesar órdenes y datos de manera clara y ordenada, lo que facilita su funcionamiento interno.

Los dispositivos de salida permiten mostrar o transmitir al usuario los resultados del procesamiento de forma visual, auditiva o física. Entre los más comunes se encuentran el monitor o la pantalla, la impresora, los parlantes, el plotter y los proyectores. Su función es transformar la información digital en formas comprensibles para las personas, como imágenes, texto impreso o sonido. Además, existen dispositivos de E/S que pueden almacenar, transferir o intercambiar información, como por ejemplo, discos HDD/SSD, las tarjetas de red, las pantallas táctiles o las memorias USB. En conjunto, estos elementos hacen posible una comunicación fluida entre una persona y el computador de forma efectiva, lo cual es fundamental para el uso diario de la tecnología.

Tareas complementarias.

1. Elabore un esquema visual que identifique y explique los principales componentes de una PC y de una laptop de tecnología actual. Deben incluirse elementos como CPU, memoria RAM, tarjeta madre, fuente de alimentación, dispositivos de entrada y salida y tarjeta gráfica.

2. Describa el funcionamiento de dos métodos de enfriamiento modernos en sistemas de cómputo (PC, laptop, smartphone).

3. Elabore una lista de 10 unidades de entrada, 6 unidades de salida y 5 unidades de E/S; escoge dos dispositivos de cada tipo que menos conozca y explica detalladamente su funcionamiento, sus componentes y su mecanismo de interconexión con el sistema de cómputo.

4. Construya un cuadro comparativo de las tarjetas gráficas de los dos principales fabricantes, considerando aspectos clave como el año de fabricación, los componentes, el rendimiento y las características técnicas.

ISBN: 978-631-6557-81-0

5. Elabore una actividad de gamificación utilizando una plataforma digital para reforzar los contenidos correspondientes a la unidad dos.

ISBN: 978-631-6557-81-0

CAPÍTULO 3.

3 Microprocesador

Este capítulo se centra en el microprocesador, el rendimiento y la eficiencia. A lo largo de las siguientes secciones se analiza qué son los microprocesadores, cómo se organiza y trabaja la CPU, y cómo han evolucionado desde diseños simples hasta arquitecturas más complejas. También, cómo se comunican los distintos componentes mediante buses, cómo se gestiona la memoria mediante direccionamiento y por qué existen diferentes enfoques de diseño, como CISC, RISC e híbridos. Finalmente, se da una mirada a las arquitecturas contemporáneas que hoy dominan el mercado.

3.1 Microprocesadores y controladores

Un microprocesador, también llamado CPU (Central Processing Unit), es el cerebro de un sistema de cómputo, ya que ejecuta instrucciones, toma decisiones y coordina el uso de los recursos del hardware. En un PC es el chip más potente, diseñado para ejecutar sistemas operativos complejos y aplicaciones de todo tipo. En cambio, los controladores (o microcontroladores) están diseñados para tareas más específicas y definidas, por ejemplo, el control de un sensor, de un motor o de una pantalla. A diferencia del microprocesador, el controlador integra en un solo chip todos los componentes, como la unidad de procesamiento, la memoria, los temporizadores y los periféricos básicos, lo que reduce costos y el consumo de energía. Por eso son tan comunes en sistemas embebidos y en dispositivos inteligentes.

3.2 Estructura y función del microprocesador

La Unidad Central de Procesamiento es el componente principal del computador y su función principal es ejecutar instrucciones. Para lograrlo de manera eficiente, el CPU está organizado internamente en varias unidades que trabajan de forma coordinada, lo que influye significativamente en el rendimiento de los programas que se ejecutan. Esta organización no es visible directamente para el programador, pero entender cómo se

ISBN: 978-631-6557-81-0

estructura el CPU ayuda a comprender por qué ciertas instrucciones son más rápidas que otras y por qué algunas decisiones de diseño de software tienen un impacto directo en la eficiencia y la eficacia del sistema.

De forma clásica, el CPU se divide en tres grandes bloques: el conjunto de registros, la unidad aritmética y lógica (ALU) y la unidad de control (Figura 3.1). Estos bloques están interconectados mediante buses internos de datos y de señales de control. Todo el funcionamiento del procesador está sincronizado por una señal de reloj, que marca el ritmo de las operaciones internas. Cada pulso del reloj define el tiempo mínimo disponible para ejecutar una acción en el CPU, aunque una instrucción completa suele necesitar varios ciclos para ejecutarse.

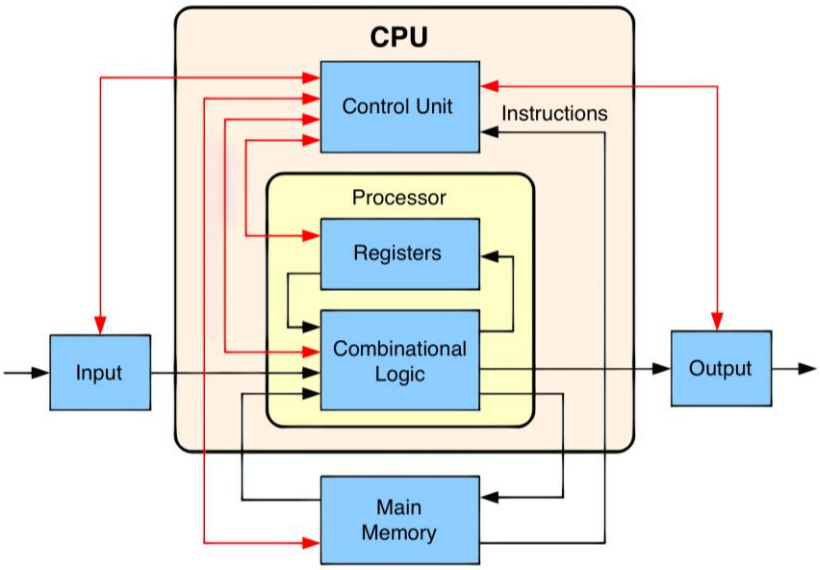


Figura 3.1 Arquitectura interna del CPU y sus bloques funcionales.
Fuente: Gemini.

Los registros son pequeñas unidades de almacenamiento ubicadas dentro del CPU. Son mucho más rápidos que la memoria principal y se usan para almacenar datos temporales, direcciones y resultados intermedios. Desde el punto de vista del software, los registros son críticos porque muchas instrucciones operan directamente sobre ellos.

ISBN: 978-631-6557-81-0

Cuantos más accesos se hagan a los registros y menos a la memoria RAM, mejor será el rendimiento del programa (Bonifacio Ceferino, 2018).

Existen distintos tipos de registros, cada uno con una función concreta. Algunos almacenan datos de uso general, otros guardan direcciones de memoria, como el contador de programa (PC), que indica cuál es la siguiente instrucción para ejecutar. También está el registro de instrucción (IR), que mantiene la instrucción actual mientras se decodifica y se ejecuta. Además, hay registros de estado que almacenan banderas o bits de control. Por ejemplo, para indicar si una operación produjo un desbordamiento o si el resultado fue cero. Aunque desde fuera parezca que todo ocurre "de golpe", internamente el CPU depende en gran medida de estos registros para mantener el orden y la coherencia de la ejecución (Ocrospoma Callupe, 2021).

La **Unidad Aritmética y Lógica** (ALU) se encarga de realizar las operaciones matemáticas y lógicas. Aquí se ejecutan sumas, restas, comparaciones, operaciones booleanas y, en muchos procesadores, también cálculos más complejos. La ALU no trabaja sola: toma los operandos de los registros, realiza la operación indicada por la instrucción y devuelve el resultado, normalmente a otro registro. En los procesadores actuales no se trabaja con una sola ALU aislada, ya que es común encontrar unidades de ejecución especializadas (de punto flotante o vectoriales) que permiten procesar múltiples datos en paralelo. Gracias a ellas, el procesador puede trabajar con varios datos simultáneamente y ejecutar aplicaciones como los gráficos, la inteligencia artificial o el procesamiento multimedia con un alto grado de paralelismo.

La unidad de control es, en cierto modo, el "director de orquesta" del CPU. Su función es generar las señales necesarias para que cada componente actúe en el momento adecuado. La ejecución de una instrucción se divide en varias fases, conocidas como el ciclo de instrucción: captación de la instrucción de memoria, decodificación, obtención de operandos, ejecución y, finalmente, comprobación de interrupciones. Cada una de estas fases se descompone en microoperaciones más simples que la unidad de control coordina paso a paso.

ISBN: 978-631-6557-81-0

Existen dos enfoques clásicos para implementar la unidad de control: **el control cableado y el control microprogramado**. El primero es más rápido, pero menos flexible, ya que las señales están definidas directamente por el hardware. El segundo utiliza microinstrucciones almacenadas en una memoria interna, lo que facilita cambios y extensiones en el conjunto de instrucciones. En los procesadores actuales, estos enfoques se combinan con técnicas avanzadas como segmentación (*pipeline*), ejecución fuera de orden y paralelismo a nivel de instrucción, que buscan aumentar el rendimiento sin incrementar excesivamente la frecuencia del reloj, como se describe en textos modernos de organización de computadores. Finalmente, los **buses internos** son el sistema de "autopistas" que permite que la información fluya entre estos tres bloques.

Dentro de la estructura interna del CPU, además de la ALU y la unidad de control que suelen explicarse primero, existen otras unidades especializadas que hacen posible el rendimiento de los procesadores actuales (Figura 3.2). Como se describe en los textos clásicos sobre la organización del procesador, el aumento del nivel de integración ha permitido que muchas de estas unidades se encuentren en el mismo CPU, cooperando entre sí para aprovechar cada ciclo de reloj (Stallings, 2019; Tanenbaum, 2016).

Una de estas unidades clave es la unidad de ejecución SIMD (Single Instruction, Multiple Data), cuya función es aplicar una sola instrucción a varios datos simultáneamente, mediante registros vectoriales en lugar de registros escalares. Por ejemplo, una suma puede realizarse en paralelo sobre varios valores en un único ciclo. Este tipo de ejecución es muy común en gráficos, audio, video, cifrado y aprendizaje automático. Aunque el programador no siempre las utiliza explícitamente, los compiladores modernos intentan aprovecharlas automáticamente cuando el código presenta un paralelismo de datos claro, como bucles simples sobre arreglos.

Un elemento clave del procesador es la memoria caché, integrada en el CPU y organizada en varios niveles, comúnmente denominados L1, L2 y L3. Su función es reducir el tiempo de acceso a datos e instrucciones de uso frecuente, aprovechando la localidad temporal y espacial. La caché L1 es la más rápida, pero tiene una capacidad de

ISBN: 978-631-6557-81-0

almacenamiento pequeña, mientras que los niveles superiores son más grandes, aunque más lentos en las operaciones de lectura y escritura.

Otros componentes, la **Unidad de Gestión de Memoria (MMU)** y la **Unidad de Punto Flotante (FPU),** cumplen roles muy específicos. La MMU se encarga de traducir direcciones virtuales a direcciones físicas, lo que permite que cada proceso trabaje con su propio espacio de memoria protegido. Esto facilita la ejecución de multitareas y la seguridad del sistema operativo. Por otro lado, la FPU está optimizada para operar con números reales (más costosos que los enteros) como un bloque independiente, con sus propios registros e interconexiones.

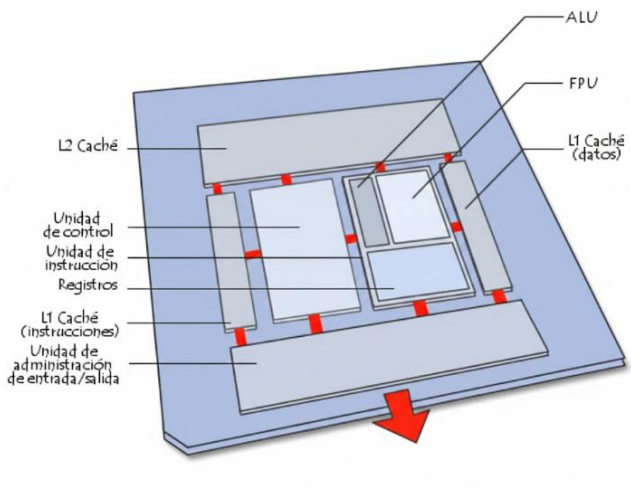


Figura 3.2 Bloques funcionales del procesador actual.
Fuente: Gemini

3.2.1 Ciclo de ejecución de las instrucciones

El ciclo de ejecución de las instrucciones describe el conjunto de pasos que sigue el microprocesador cada vez que ejecuta una instrucción de un programa. Aunque desde el punto de vista del programador todo parece continuo y casi instantáneo, internamente la CPU trabaja de forma ordenada y repetitiva. Este ciclo se repite millones de veces por segundo y es la base del funcionamiento de cualquier computador moderno. Entender este

ISBN: 978-631-6557-81-0

ciclo es clave porque permite comprender por qué ciertas instrucciones son más costosas que otras y cómo el hardware influye directamente en el rendimiento del código.

La primera etapa del ciclo es la búsqueda de la instrucción (fetch). En este punto, el procesador utiliza el contador de programa (PC) para determinar en qué dirección de memoria se encuentra la siguiente instrucción a ejecutar. Esa instrucción se trae de memoria y se almacena en un registro interno, normalmente denominado registro de instrucción. Al mismo tiempo, el PC se actualiza para apuntar a la siguiente instrucción. Este paso es común a todas las instrucciones y ocurre casi de forma automática, gobernado por la señal de reloj del procesador. Una vez cargada la instrucción, comienza la fase de decodificación. Aquí, la unidad de control analiza la instrucción para determinar qué operación debe ejecutarse y qué operandos intervienen en ella. Puede tratarse de una suma, un salto, una lectura de memoria, entre otras posibilidades. En esta etapa también se decide si será necesario acceder a registros, a memoria o a ambos. Aunque parezca simple, esta fase puede variar bastante en complejidad según el tipo de instrucción y el diseño del conjunto de instrucciones del procesador.

A continuación, viene la fase de ejecución, en la que la instrucción cumple su función. Si se trata de una operación aritmética o lógica, la ALU es la encargada de realizarla; si la instrucción implica leer o escribir datos, se accede a la memoria; si se trata de un salto, se modifica el flujo de ejecución del programa. Cuando existe un resultado, se guarda en un registro o directamente en memoria. Además, durante esta fase pueden actualizarse ciertos indicadores del procesador, como los bits de estado que indican si el resultado fue cero o negativo, o si se produjo un desbordamiento.

El ciclo se completa con la comprobación de interrupciones, una etapa fundamental que a menudo pasa inadvertida. El procesador verifica si algún dispositivo externo requiere un servicio (por ejemplo, un teclado o un temporizador) o si hay alguna interrupción pendiente. Si hay una interrupción activa, el procesador guarda la información necesaria sobre el estado actual y transfiere el control a una rutina especial encargada de gestionarla. En caso contrario, continúa con la siguiente instrucción del programa. En los procesadores

ISBN: 978-631-6557-81-0

actuales, este ciclo básico puede solaparse mediante técnicas como la segmentación (pipeline), lo que permite ejecutar varias instrucciones simultáneamente y así mejorar el rendimiento, sin cambiar la lógica del ciclo de instrucciones (Figura 3.3).

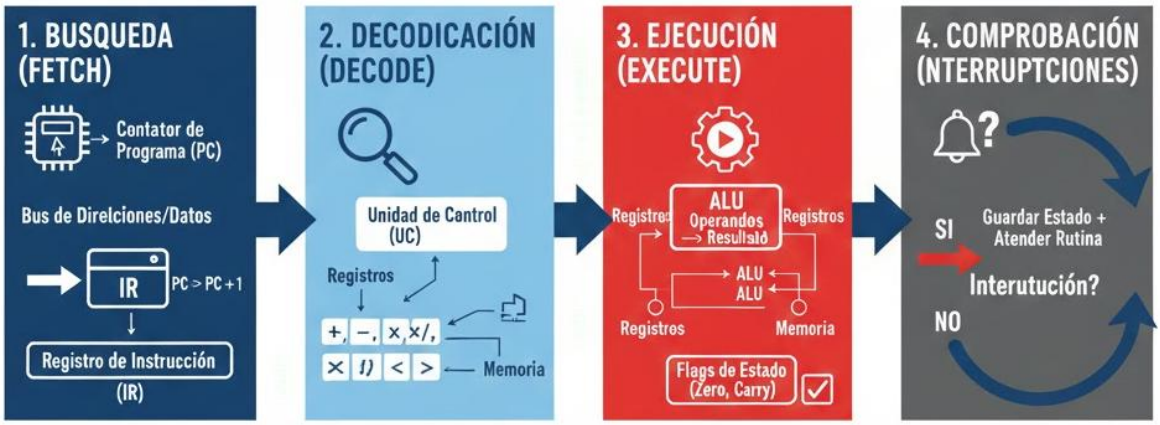


Figura 3.3 Ciclo de ejecución de una instrucción.

Fuente: Gemini.

Segmentación de las instrucciones

La segmentación de las instrucciones, también conocida como pipeline, es una técnica de diseño del procesador que busca mejorar el rendimiento sin necesidad de hacer el hardware más rápido. La idea central es que el procesador divida la ejecución en varias etapas: lectura de la instrucción, decodificación, ejecución y acceso a la memoria. Mientras una instrucción se está ejecutando, la siguiente puede estar decodificándose y otra más, leyéndose de memoria. Así, varias instrucciones avanzan simultáneamente, cada una en una fase distinta, lo que permite aprovechar mejor los recursos internos del procesador y aumenta el rendimiento general.

Una forma sencilla de entenderlo es pensar en una línea de montaje: no se fabrica un producto completo antes de empezar otro, sino que cada etapa trabaja en paralelo en productos distintos. Es importante notar que la segmentación no reduce el tiempo de ejecución de una instrucción individual, sino que aumenta el número de instrucciones completadas por unidad de tiempo. Eso sí, no todo es perfecto: pueden aparecer problemas

ISBN: 978-631-6557-81-0

llamados hazards o riesgos, como dependencias entre instrucciones o saltos condicionales que rompen el flujo normal. Por esta razón, los procesadores modernos incorporan mecanismos adicionales, como la detección de riesgos y la predicción de saltos, para que el pipeline funcione de manera eficiente en programas reales (Figura 3.4).

Figura 3.4 Segmentación de instrucciones

Fuente: Apuntes de (Orenga & Manonellas, 2011)

3.2.2 Registros

En la arquitectura de un computador, los **registros** son uno de los elementos más pequeños, pero también de los más importantes dentro del procesador. Se trata de espacios de almacenamiento muy rápidos, físicamente ubicados dentro de la CPU, que se utilizan para almacenar datos e información temporal mientras se ejecutan instrucciones. A diferencia de la memoria principal, acceder a un registro toma muy poco tiempo: normalmente, un solo ciclo del reloj. Por eso, muchas decisiones de diseño del procesador giran en torno a cuántos registros hay y cómo se utilizan.

Desde un enfoque práctico, los registros existen porque la CPU necesita trabajar con datos de forma inmediata. Si una instrucción tiene que sumar dos números, estos valores deben estar almacenados en los registros internos, no en memoria. Acceder directamente desde la RAM en cada operación sería demasiado lento y ralentizaría el

ISBN: 978-631-6557-81-0

sistema. Así, los registros actúan como una especie de "mesa de trabajo" del procesador, donde se colocan los datos que se van a usar de inmediato.

Dentro de un procesador encontramos distintos tipos de registros, cada uno con una función concreta. Los más conocidos son los **registros de propósito general**, que pueden almacenar operandos, resultados intermedios o direcciones. En las arquitecturas modernas, estos registros suelen ser visibles para el programador o el compilador, especialmente cuando se programa en ensamblador (Tabla 3.1). Cuantos más registros de este tipo tenga una arquitectura, menos accesos a la memoria serán necesarios, lo que normalmente mejora el rendimiento.

REGISTROS				
64 bits	**32 bits**	**16 bits**	**8 bits (alto)**	**8 bits (bajo)**
RAX	EAX	AX	AH	AL
RBX	EBX	BX	BH	BL
RCX	ECX	CX	CH	CL
RDX	EDX	DX	DH	DL
RSI	ESI	SI	No disponible	No disponible
RDI	EDI	DI	No disponible	No disponible
RSP	ESP	SP	No disponible	No disponible
RBP	EBP	BP	No disponible	No disponible

Tabla 3.1 Registros de propósito general de la arquitectura x86-64

Además de los registros de propósito general, existen **registros de instrucción** con tareas muy específicas. Un ejemplo claro es el contador de programa (PC), que almacena la dirección de la siguiente instrucción a ejecutar. Otro es el registro de instrucción (IR), donde se almacena la instrucción que acaba de ser leída de memoria y está a punto de ejecutarse. Estos registros son fundamentales para el ciclo de ejecución de las instrucciones y suelen funcionar de forma casi automática, sin intervención directa del programador

También están los **registros de acceso a la memoria**, como el registro de dirección de memoria (MAR) y el registro de datos de memoria (MBR o MDR). Estos registros sirven como intermediarios cuando la CPU necesita leer o escribir en la memoria principal. Aunque el programador no los utilice directamente, son esenciales para coordinar la

ISBN: 978-631-6557-81-0

comunicación entre la CPU y la memoria. Sin ellos, el diseño interno del procesador sería mucho más complejo y menos ordenado (Figura 3.5).

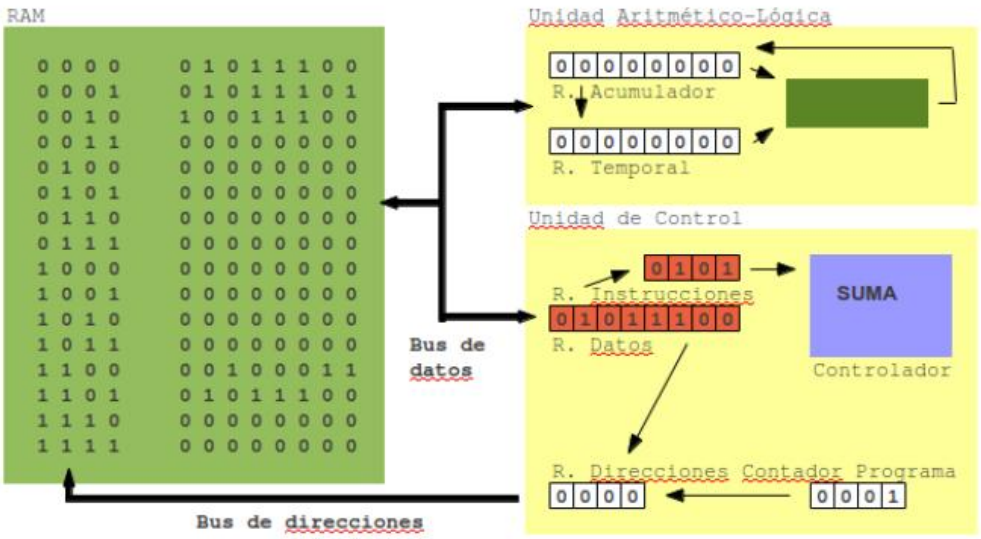


Figura 3.5 Esquema del uso de registros de acceso a memoria

Elaborado por el autor

Otro grupo importante son los **registros de estado y control.** El registro de estado se actualiza automáticamente mientras se ejecutan las instrucciones, especialmente las aritméticas y lógicas, y su contenido influye directamente en el control del programa. Por ejemplo, algunos bits indican si el resultado de una operación fue cero, si hubo acarreo o desbordamiento, o si fue negativo, lo cual es fundamental para instrucciones condicionales como saltos o comparaciones. Otros bits no están relacionados con cálculos, sino con el control del sistema, como el que indica si las interrupciones están habilitadas o si el procesador está ejecutando código en modo usuario o en modo supervisor. Incluso puede incluir información sobre el nivel de privilegio del programa en ejecución, lo que permite al sistema operativo tomar el control cuando sea necesario. Aunque el programador no siempre accede directamente a todos estos bits, muchos lenguajes de bajo nivel y sistemas operativos dependen de ellos para funcionar correctamente, ya que reflejan el estado real del procesador en tiempo de ejecución (Figura 3.6).

ISBN: 978-631-6557-81-0

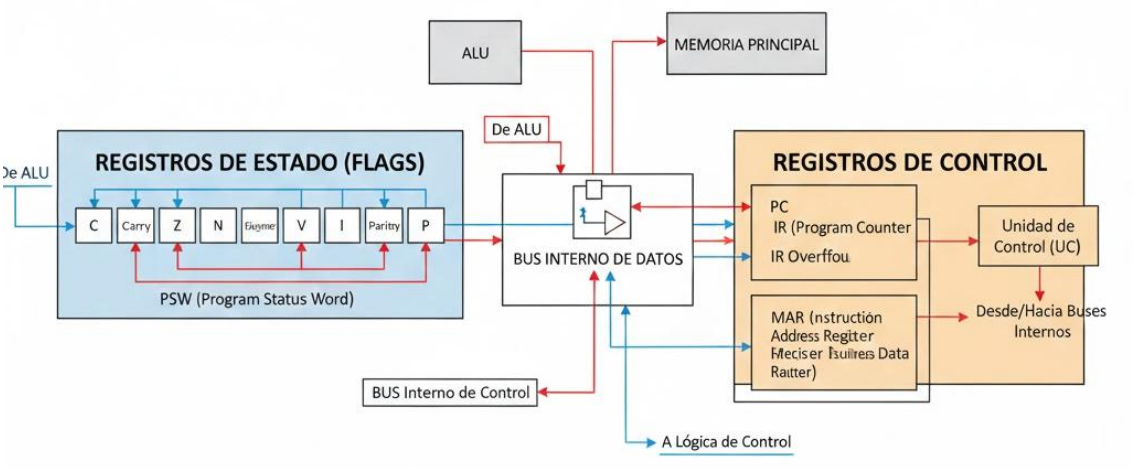


Figura 3.6 Esquema lógico de los registros internos del CPU

Fuente: Gemini

Por último, es importante destacar que el papel de los registros ha crecido con el tiempo. En arquitecturas RISC modernas, por ejemplo, se apuesta por contar con muchos registros y un conjunto de instrucciones más simple, lo que facilita la segmentación y el paralelismo. Esto conecta directamente con el rendimiento y el trabajo del compilador, que debe decidir qué valores van a los registros y cuáles se envían a memoria.

3.2.3 Unidad Aritmético-Lógica (ALU)

La unidad aritmética y lógica (ALU) es uno de los componentes centrales del procesador. La ALU es el circuito encargado de realizar operaciones matemáticas simples, como sumas y restas, y operaciones lógicas, como AND, OR, NOT, XOR, así como operaciones de desplazamiento de bits a la izquierda y a la derecha y de rotación de bits. Cada vez que en un programa se evalúa una condición, se incrementa una variable o se compara un valor, la ALU entra en acción. Desde el punto de vista del software, estas operaciones parecen instantáneas, pero, por debajo, implican señales eléctricas y puertas lógicas que operan de forma coordinada.

La ALU toma sus operandos de los registros del procesador y devuelve el resultado también a un registro. Al ejecutar muchas instrucciones, la ALU modifica ciertos

ISBN: 978-631-6557-81-0

indicadores, conocidos como bits de estado (como el bit de cero, el de signo o el de acarreo). Estos bits son fundamentales para la toma de decisiones y el flujo de control de los programas, ya que instrucciones como los saltos condicionales dependen directamente de ellos. Por ejemplo, una instrucción "if" en un lenguaje de alto nivel acaba convirtiéndose en una comparación en la ALU y en una decisión basada en esos bits de estado.

En los procesadores modernos, la idea de "una sola ALU" se queda corta. Hoy en día es común encontrar varias ALU, unidades especializadas y, en muchos casos, una unidad de punto flotante (FPU) separada para realizar cálculos más complejos. Incluso existen ALU vectoriales que operan sobre varios datos a la vez, muy utilizadas en gráficos y multimedia. Aun así, el concepto básico sigue siendo el mismo que en los primeros diseños: la ALU es el lugar donde los datos realmente se transforman (Figura 3.7).

3.2.4 Unidad de control

La unidad de control de la CPU se encarga de que todo funcione en orden. No realiza cálculos ni guarda datos como tal, pero decide qué se hace, cuándo y cómo dentro del procesador. Su tarea principal es interpretar las instrucciones provenientes de la memoria y generar las señales necesarias para que los registros, la ALU y otros componentes actúen correctamente. Sin esta unidad, el procesador sería solo un conjunto de circuitos descoordinados, incapaz de ejecutar un programa de forma coherente (Figura 3.8). En cada una de las fases del ciclo de instrucción, la unidad de control va activando microoperaciones muy concretas, como cargar un registro, mover datos por los buses internos o indicar a la ALU qué operación debe realizar, mediante un mecanismo sincronizado con la señal de reloj. Desde el punto de vista del software, estos detalles no se ven, pero influyen directamente en el rendimiento y en cómo se ejecutan las instrucciones que escribimos en alto nivel o incluso en ensamblador.

ISBN: 978-631-6557-81-0

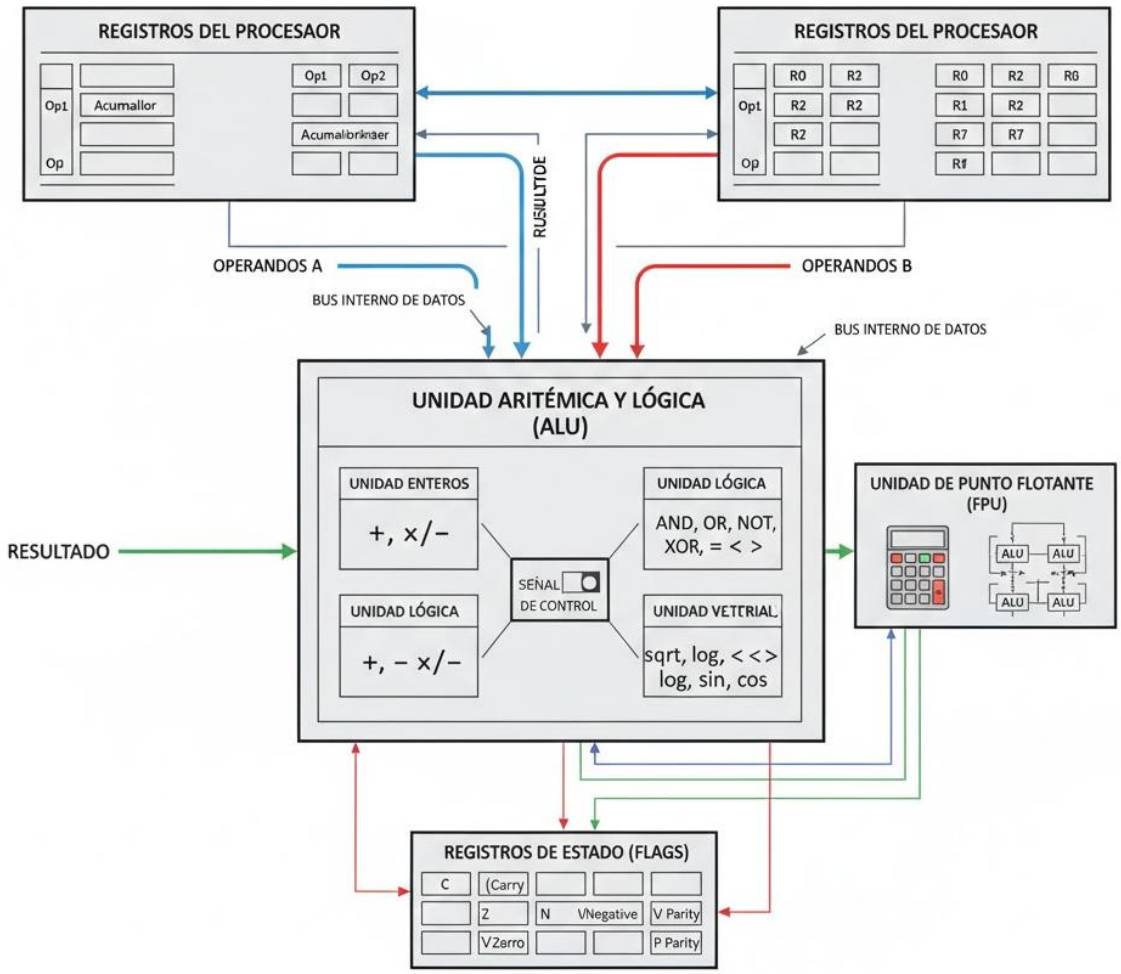


Figura 3.7 La ALU y conexiones internas.

Fuente: Gemini

Dentro de la unidad de control, las microoperaciones son las acciones más pequeñas y básicas que el procesador puede realizar durante la ejecución de una instrucción. No hablamos todavía de instrucciones completas como un ADD o un LOAD, sino de pasos internos mucho más simples, como mover un dato de un registro a otro o activar una suma en la ALU.

Según los textos clásicos de organización de computadores, estas microoperaciones suelen agruparse en categorías para comprender mejor su función en el ciclo de ejecución.

ISBN: 978-631-6557-81-0

Esta clasificación no es solo teórica: ayuda a diseñar la unidad de control y a razonar sobre cómo el hardware coordina todas las partes del procesador.

Una primera clasificación relevante corresponde a las microoperaciones de transferencia de datos, que se encargan de desplazar información entre registros internos o entre registros y la memoria principal. Ejemplos típicos son cargar el valor del contador de programa en un registro de dirección de memoria o copiar el contenido de un registro a otro. Otro grupo importante son las microoperaciones aritméticas y lógicas, que activan la ALU para realizar cálculos de operaciones básicas, comparaciones u operaciones lógicas como AND u OR. Finalmente, están las microoperaciones de control, que no mueven datos ni calculan resultados, sino que modifican el flujo de ejecución, como actualizar el contador del programa o comprobar condiciones para saltos e interrupciones.

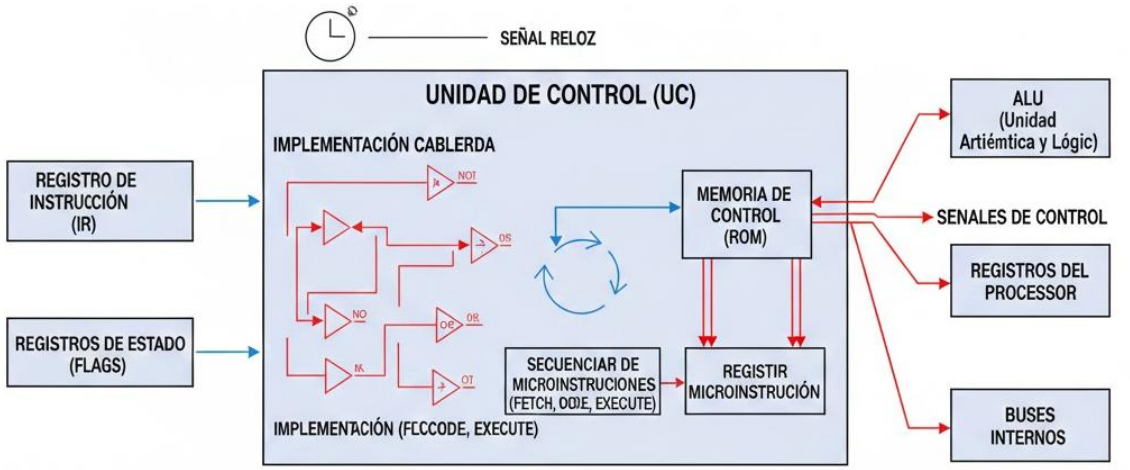


Figura 3.8 Lógica del trabajo de la unidad de control.

Fuente: Gemini

Tradicionalmente, la unidad de control se ha construido en dos formas: cableada y microprogramada. En el primer caso, el control se implementa con circuitos lógicos fijos, que permiten una ejecución muy rápida, pero también resultan poco flexibles ante cambios. Esto significa que el comportamiento del procesador está definido directamente por el hardware. En este enfoque, las señales de control que gobiernan los registros, el ALU, los buses internos y los accesos a memoria se generan directamente mediante lógica combinacional y secuencial, que se activa cuando una instrucción es decodificada y emite

ISBN: 978-631-6557-81-0

un conjunto concreto de señales que se propagan por el procesador y se ejecutan en el orden correcto. Esto la hace especialmente atractiva en procesadores en los que el rendimiento es crítico y el conjunto de instrucciones es relativamente simple, como ocurre en las arquitecturas RISC. Sin embargo, esta rapidez puede limitar la flexibilidad, ya que cualquier cambio en el conjunto de instrucciones o en la forma de ejecutarlas implica rediseñar el hardware, lo cual resulta costoso y poco práctico.

La unidad de control microprogramada, en cambio, utiliza un programa interno (llamado microprograma) almacenado en la memoria de control, lo que facilita la incorporación de instrucciones complejas sin rediseñar todo el hardware. Se la concibe como una capa intermedia entre el hardware y las instrucciones de máquina. Cada instrucción se descompone en una secuencia de microinstrucciones, y cada una de ellas activa señales concretas que mueven datos, controlan la ALU o acceden a registros. Esta idea permite que el funcionamiento interno del procesador sea más ordenado.

Desde el punto de vista práctico, la unidad de control microprogramada aporta flexibilidad. Cambiar o corregir el comportamiento de una instrucción puede implicar únicamente modificar el contenido de la memoria de control, sin rediseñar el hardware completo. Por eso, históricamente, muchos procesadores optaron por este enfoque, ya que simplificaba el diseño y reducía los errores, aunque a veces con una ligera pérdida de velocidad.

Sin embargo, esta técnica también tiene sus límites. Al depender de la lectura de microinstrucciones desde una memoria de control, la ejecución puede ser más lenta que en una unidad de control cableada, donde las señales se generan directamente. En los procesadores modernos, donde el rendimiento es crítico, la microprogramación ha perdido protagonismo en la ejecución de instrucciones complejas. En los procesadores modernos se combinan ambas técnicas, que constituyen la base para comprender cómo la CPU traduce instrucciones en una secuencia concreta de acciones.

ISBN: 978-631-6557-81-0

3.3 Evolución de los microprocesadores

Inicialmente, la CPU estaba formada por varios circuitos separados, lo que hacía que los computadores fueran grandes, costosos y difíciles de escalar. La historia de los microprocesadores comienza cuando la CPU se integra en un único chip. Con la llegada de los primeros microprocesadores en los años setenta, de 4 y 8 bits, se pudieron construir computadores más pequeños y accesibles. Estos primeros diseños eran simples, con pocos registros y sin mecanismos avanzados de control, pero marcaron el punto de partida de la evolución posterior.

Con el avance de la tecnología de fabricación, los microprocesadores comenzaron a manejar más bits por instrucción. Al pasar de 8 a 16 y luego a 32 bits, mejoraron la capacidad de procesamiento. Se ampliaron los registros, los espacios de direccionamiento de memoria y los conjuntos de instrucciones. En este período se afianzó el modelo clásico de procesador con registros, ALU y unidad de control, que ejecutaba instrucciones de forma secuencial, siguiendo el ciclo de búsqueda, decodificación y ejecución. Para quienes desarrollaban software, esto permitió crear programas y sistemas operativos más complejos.

Más adelante, el principal desafío ya no era solo ejecutar instrucciones, sino hacerlo más rápido sin aumentar indefinidamente la frecuencia del reloj. Para resolver esto surgieron técnicas como la segmentación de instrucciones (pipeline), en la que varias instrucciones se procesan parcialmente de forma simultánea, como en una cadena de montaje. Este enfoque fue importante para los microprocesadores RISC, que apostaron por instrucciones simples y una ejecución rápida, en contraste con las arquitecturas CISC más complejas. Con el aumento de la complejidad del software, los microprocesadores empezaron a incorporar más lógica interna. Se añadieron memorias caché, unidades de punto flotante y mecanismos de predicción de saltos para evitar tiempos muertos. El procesador dejó de ser solo un ejecutor de instrucciones y pasó a tomar decisiones internas para optimizar el flujo de ejecución. Esto hizo que la arquitectura interna se volviera más

ISBN: 978-631-6557-81-0

compleja, aunque el modelo de programación siguiera siendo bastante estable, de modo que no rompiera la compatibilidad con el software existente.

Cuando IBM lanzó el primer PC en 1981, incorporó el microprocesador Intel 8088. Este procesador operaba a 4,77 MHz para las tareas de oficina de la época. El equipo incluía 16 KB de memoria RAM, que podía ampliarse hasta 256 KB. El almacenamiento se hacía mediante disquetes como principal medio, y se podía conectar un monitor a color. Como complemento, el PC podía incorporar el coprocesador matemático Intel 8087, capaz de realizar operaciones con números de coma flotante y precursor directo del estándar IEEE 754 que seguimos usando hoy. Con el paso del tiempo, se añadieron más capacidades de memoria RAM, pero la arquitectura reservó parte del espacio de direcciones para vídeo y ROM. Para su funcionamiento contaba con 14 registros de 16 bits, organizados de forma bastante lógica. Cuatro eran registros de propósito general (AX, BX, CX y DX), utilizados para cálculos y el movimiento de datos. Otros cuatro eran registros de segmento (CS, DS, SS y ES), cuya función era ampliar el espacio de direcciones. Lo interesante de este enfoque fue que los procesadores x86 conservaban esta estructura, aunque con registros más grandes y extensiones diseñadas para sistemas operativos multitarea.

El Intel 80286 se incorporó al IBM PC AT en 1984. Este procesador de 16 bits y 24 líneas de dirección permitía acceder a hasta 16 MB de memoria. Esto, en la práctica, podía ejecutar casi el doble de instrucciones que su antecesor. A 6 MHz alcanzaba alrededor de 0,9 MIPS, una cifra notable para la época. Otro aporte fue la introducción del modo protegido, pensado para sistemas multiusuario y multitarea, para evitar que un programa afectara a otros programas o al sistema operativo. Sin embargo, esta idea generó un alto costo de memoria, lo que la hacía poco viable para su uso real durante varios años. Luego llegó el Intel 80386, con un procesador de 32 bits que permitió trabajar con un modelo de memoria plano, capaz de direccionar hasta 4 GB sin depender tanto de la segmentación.

En 1986, Compaq lanzó el DeskPro, un PC compatible con IBM basado en el 80386, que incorporó una versión de Microsoft Windows adaptada a dicha arquitectura.

ISBN: 978-631-6557-81-0

Una ventaja fue que el procesador 80386 era compatible con el software de los 80286 y 8088, lo que contribuyó significativamente a su adopción. Su diseño sentó las bases de la arquitectura x86 que aún se utiliza hoy. Las primeras versiones alcanzaron frecuencias de hasta 33 MHz y rendimientos superiores a 11 MIPS. Lo más relevante se centró en el uso intensivo de la caché y en técnicas más eficientes de ejecución de instrucciones. Gracias a estos aportes, los procesadores actuales son cientos de veces más rápidos.

Un buen punto de partida es el Intel 486, que apareció a comienzos de los años noventa. Este procesador marcó un salto importante porque integró en un solo chip elementos que antes estaban separados, como la unidad de punto flotante y la memoria caché básica. Además, introdujo el pipeline de forma más clara, lo que permitió ejecutar simultáneamente varias etapas de instrucciones distintas. Para el software, esto significó programas más rápidos sin cambiar el código, algo que empezaba a ser clave.

Con la llegada del Pentium, Intel marcó un hito importante con la arquitectura superscalar, que podía ejecutar más de una instrucción por ciclo de reloj. También se mejoró la predicción de saltos, lo que evitó esperas innecesarias, y se aumentó el ancho de banda del bus de datos. Se demostró que el rendimiento ya no dependía solo de la frecuencia, sino también del diseño interno del procesador.

Muchos años después, Intel reorganizó su línea de productos con la familia Core, empezando por el procesador Core i3, diseñado para tareas generales y de consumo moderado, con énfasis en la eficiencia energética. A nivel arquitectónico, se construyó con múltiples núcleos en un mismo chip y con jerarquías de caché más complejas. El paralelismo dejó de ser algo exclusivo de los servidores y pasó a los computadores personales, lo que obligó al software a adaptarse mediante la programación en varios hilos de ejecución para aprovechar realmente el hardware disponible.

El Core i5 se posicionó como un punto intermedio, equilibrando el rendimiento y el precio. Gracias a la tecnología Turbo Boost, se consolidó el uso de múltiples núcleos con mejores frecuencias dinámicas, ajustando su velocidad según la carga de trabajo. En

ISBN: 978-631-6557-81-0

la práctica, esto resultó muy útil para aplicaciones como editores, navegadores o de oficina, para beneficiarse sin estar completamente paralelizadas. El Core i7 mejoró su rendimiento al incorporar más núcleos y una mayor caché gracias al soporte de hyper-threading. Esta técnica permite que cada núcleo gestione dos hilos de ejecución, lo que mejora el uso de los recursos internos del procesador y permite ejecutar muchos procesos en paralelo.

El Core i9 representa el enfoque actual en el alto rendimiento. Estos procesadores están orientados a cargas de trabajo exigentes, como la compilación, el renderizado, las simulaciones complejas o los videojuegos de última generación. Para lograrlo, combinan muchos núcleos, grandes cantidades de caché y frecuencias elevadas, aunque con un consumo energético considerable. En esta etapa, la evolución ya no se basa solo en hacer núcleos más rápidos, sino en coordinar muchos núcleos de manera eficiente. En la tabla 3.2 se describen las principales características de los procesadores actuales de Intel.

En los últimos años, en lugar de seguir aumentando la frecuencia de los microprocesadores, los fabricantes empezaron a integrar más núcleos en un mismo chip (microprocesadores multinúcleo). Cada núcleo funciona como un procesador independiente, aunque comparte recursos como la memoria caché de niveles superiores.

Hoy en día, los microprocesadores combinan muchas de estas ideas: múltiples núcleos, ejecución fuera de orden, paralelismo a nivel de instrucciones y soporte para tareas especializadas. Además, no solo están en computadores personales, sino también en servidores, teléfonos y sistemas embebidos. La evolución no ha sido una línea recta, sino una adaptación constante a los límites físicos y a nuevas necesidades.

ISBN: 978-631-6557-81-0

Característica	Beneficio
Performance-core (P-core)	Un núcleo potente diseñado para cargas de trabajo de un solo thread, de threads ligeros o que presentan grandes exigencias, como gaming 4K y diseño 3D.
Efficient-core (E-core)	Optimizados para tareas de múltiples threads y en segundo plano, como las pestañas de navegador minimizadas, los servicios informáticos y la sincronización en la nube, lo que deja a los P-cores libres para ofrecer un desempeño increíble sin interrupciones.
Arquitectura híbrida de desempeño	Integra dos microarquitecturas de núcleo en un solo chip que priorizan y distribuyen cargas de trabajo, a fin de optimizar el desempeño.
Intel® Thread Director	Optimiza las cargas de trabajo al ayudar al programador de SO a distribuir de forma inteligente las cargas de trabajo a los núcleos óptimos.
PCIe 5.0 de hasta 16 carriles	Ofrece disponibilidad para hasta 32 GT/s para acceder rápidamente a gráficos independientes, almacenamiento y dispositivos periféricos con hasta 16 carriles PCI Express 5.0.
Hasta 4 carriles PCIe 4.0	Ofrece hasta 16 GT/s para acceder rápidamente al almacenamiento y dispositivos periféricos con hasta 4 carriles PCI Express 4.0.
Hasta DDR5 5600 MT/s	Ofrece la innovación más reciente, líder de la industria en capacidades de memoria para velocidades elevadas, ancho de banda alto y una mejora en la productividad del flujo de trabajo.
Hasta DDR4 3200 MT/s	Compatibilidad constante de tecnología de memoria y velocidades existentes.
Memoria caché L3 y L2	El aumento de los tamaños de caché inteligente Intel® (L3) y L2 permite a los usuarios trabajar más rápido, con conjuntos de datos más grandes.
Intel® Deep Learning Boost	Acelera la inferencia de IA para mejorar el desempeño de las cargas de trabajo de aprendizaje profundo.
Gaussian & Neural Accelerator 3.0 (GNA 3.0)	Procesa las aplicaciones de voz y audio de IA, como la cancelación de ruido neuronal, mientras que libera recursos de CPU para un desempeño y una capacidad de respuesta general del sistema.
Tecnología Intel® Adaptive Boost	Intel® ABT mejora el desempeño, ya que permite de manera oportuna mayores frecuencias turbo de múltiples núcleos, mientras funciona dentro de las especificaciones de energía y temperatura del sistema siempre que existan corriente, energía y margen térmico.
Intel® Thermal Velocity Boost	Intel® Thermal Velocity Boost aumenta de manera oportuna y automática la frecuencia de reloj de ciertos procesadores Intel® Core™ de 13ª generación para equipos de desktop hasta 100 MHz si el procesador está a una temperatura de 70 °C o menos y cuando el consumo de energía turbo esté disponible.
Tecnología Intel® Turbo Boost Max 3.0	Identifica los núcleos más rápidos del procesador y dirige a ellos las cargas de trabajo principales.
Gráficos UHD Intel® impulsados por arquitectura X	Las capacidades de gráficos inteligentes y medios enriquecidos posibilitan una complejidad visual amplificada, un desempeño 3D mejorado y un procesamiento de imágenes más rápido.
Características y capacidades de overclocking	Cuando se combinan con el chipset Intel® Z790 o Z690, los P-cores, los E-cores, los gráficos y la memoria del procesador pueden configurarse para funcionar a frecuencias superiores a la especificación del procesador, lo que da lugar a un mayor desempeño.
Intel® Extreme Tuning Utility	Un conjunto de herramientas de precisión para ajustar y hacer overclocking que cuenta con capacidad para overclocking del procesador híbrido y de la memoria, de manera que ofrece a los usuarios nuevos y experimentados más beneficios de sus procesadores desbloqueados.[6]
Intel® Extreme Memory Profile 3.0	Simplifica la experiencia de overclocking de memoria, gracias a una mayor flexibilidad, perfiles adicionales y controles de voltaje ampliados.
Intel® Dynamic Memory Boost	Desempeño de overclocking de memoria inteligente a pedido que optimiza el desempeño de la plataforma basado en el uso.

Tabla 3.2 Componentes y características destacadas de los procesadores Intel

Fuente: www.intel.com

ISBN: 978-631-6557-81-0

Por otro lado, la trayectoria de los microprocesadores de AMD (Advanced Micro Devices) comienza con un proceso gradual de aprendizaje y adaptación. Durante las décadas de 1980 y 1990, AMD fabricó procesadores compatibles con la arquitectura x86 de Intel (como los procesadores Am386 y Am486) ofreciendo alternativas compatibles y más económicas que funcionaran con el mismo software, utilizando una arquitectura con diseños bastante clásicos, con ejecución secuencial y un paralelismo muy limitado.

Un cambio significativo llegó a finales de los noventa con la familia K6 y, sobre todo, con el Athlon (K7); AMD empezó a competir seriamente en términos de diseño interno y rendimiento, mejorando el pipeline, el manejo de la caché y el soporte a instrucciones de punto flotante. De hecho, en varias pruebas de rendimiento, el Athlon logró resultados superiores a los de Intel, lo que demostró que la arquitectura interna del procesador era tan importante como la frecuencia de reloj.

La evolución continuó con la arquitectura K8, comercializada como Athlon 64 y Opteron. Este paso fue clave para la compañía, ya que diseñó el soporte nativo para el procesamiento de 64 bits, antes de que Intel lo adoptara de manera masiva. También AMD integró el controlador de memoria en el procesador, mejorando los tiempos de acceso a la RAM (latencia). Los sistemas operativos y aplicaciones 64 bits dejaron de ser exclusivos de servidores y empezaron a comercializarse en PCs de escritorio.

Sin embargo, las arquitecturas Bulldozer, Piledriver y sus derivados marcaron una etapa complicada, ya que fueron diseñadas de manera modular en la que algunos recursos se compartían entre los núcleos lógicos. Sobre el papel, parecía una forma eficiente de escalar el paralelismo, pero en la práctica el rendimiento por núcleo resultó bajo. Muchas aplicaciones no lograban aprovechar bien este diseño, lo que dejó claro que el paralelismo mal explotado no siempre se traduce en un mejor desempeño real.

La recuperación llegó en 2017 con el desarrollo de la arquitectura Zen, junto con los primeros Ryzen, que ofrecían un buen equilibrio entre el rendimiento por núcleo, el paralelismo y la eficiencia energética, mediante Simultaneous MultiThreading (SMT). Además, se implementó una jerarquía de caché más sólida y un diseño del núcleo más

ISBN: 978-631-6557-81-0

limpio, enfocados en chiplets de cómputo y de E/S, lo que permitió escalar el número de núcleos sin aumentar los costos, algo muy relevante tanto para servidores como para PC de escritorio.

En este contexto, apareció Ryzen 3, que ofrecía procesadores con pocos núcleos, pero con un rendimiento bastante sólido para tareas cotidianas y programación básica, basado en conceptos modernos de arquitectura (núcleos eficientes, cachés bien organizados y soporte para la multitarea real). Más adelante llegó Ryzen 5, que ayudó a popularizar el uso de más núcleos en PC y potenció a los procesadores de AMD frente a generaciones anteriores, ofreciendo un buen equilibrio entre rendimiento, consumo energético y precio.

Con Ryzen 7 y Ryzen 9, la evolución fue todavía más evidente. Estas gamas llevaron el enfoque multinúcleo al siguiente nivel, incorporando arquitecturas cada vez más refinadas (Zen 2, Zen 3 y posteriores), mejoras en el pipeline, mayor cantidad de caché y una mejor comunicación interna entre núcleos. Ryzen 7 se consolidó como una opción potente para tareas intensivas, como virtualización o edición, mientras que Ryzen 9 se posicionó directamente en el terreno del alto rendimiento, con muchos núcleos e hilos pensados para cargas de trabajo pesadas y paralelas.

Hoy en día, Intel y AMD compiten no solo en potencia, sino también en aceleración de IA, donde aparecen NPUs (Neural Processing Units) y unidades especializadas que impactan directamente en el software moderno, especialmente en aplicaciones locales de IA, en la compilación y en la multitarea pesada. La tabla 3.3 resume la evolución de los principales procesadores.

ISBN: 978-631-6557-81-0

Año aprox.	Intel (modelo clave)	AMD (modelo clave)	Arquitectura / tecnología	Núcleos	Enfoque principal
1982–1984	Intel 80286	AMD Am286	16 bits, sin caché	1	PCs iniciales
1989–1992	Intel 80486	AMD Am486	32 bits, FPU integrada	1	Mejor rendimiento matemático
1995–1999	Pentium / Pentium II	AMD K5 / K6	Superscalar, caché L1	1	Uso doméstico
2000–2003	Pentium 4	AMD Athlon	Altas frecuencias	1	Rendimiento por GHz
2005–2008	Core 2 Duo	Athlon 64 X2	64 bits, doble núcleo	2	Multitarea
2010–2013	Core i5 / i7	Phenom II	Multinúcleo, caché L3	4–6	Juegos y escritorio
2017	Core i7 (7ª gen)	Ryzen 7 (Zen)	SMT, 14 nm	8	Competencia directa
2019	Core i9 (9ª gen)	Ryzen 9 (Zen 2)	Chiplets, 7 nm	12–16	Alto rendimiento
2022	Core i9 (12ª–13ª)	Ryzen 7000	Núcleos híbridos / DDR5	16–24	Eficiencia + potencia
2024–2025	Core Ultra / Core i9	Ryzen AI / Ryzen 9	IA integrada (NPU)	16–24	IA local

Tabla 3.3 Descripción cronológica breve de los procesadores de Intel y AMD

Elaborado por el autor

3.4 Arquitecturas CISC/RISC

Cuando se habla de arquitecturas CISC y RISC, en realidad estamos comparando dos formas distintas de diseñar el conjunto de instrucciones que comprende un procesador. En las arquitecturas CISC (Complex Instruction Set Computer), la idea original fue que el hardware contuviera instrucciones muy potentes y variadas, capaces de realizar tareas complejas en una sola instrucción. Esto permitió, durante muchos años, escribir programas más breves en ensamblador, algo valioso cuando la memoria era escasa. Ejemplos clásicos de este enfoque son los procesadores de la familia x86, en los que una instrucción puede combinar acceso a memoria, operación aritmética y almacenamiento del resultado en un solo paso (Stallings, 2019). Desde el punto de vista práctico, el funcionamiento de CISC conlleva una mayor complejidad en el hardware, pero también una gran flexibilidad. Las instrucciones suelen tener longitud variable, múltiples modos de direccionamiento y pueden acceder directamente a la memoria, lo que reduce la cantidad de instrucciones visibles para el programador.

Las arquitecturas RISC (Reduced Instruction Set Computer) surgieron como respuesta a la creciente complejidad y a la mejora del rendimiento. Muchas instrucciones complejas se utilizaban poco y resultaban difíciles de ejecutar de manera eficiente. En este entorno, RISC define un conjunto de instrucciones pequeñas, más simples y de longitud fija, diseñadas para ejecutarse rápidamente. Esto simplifica el diseño interno de los

ISBN: 978-631-6557-81-0

procesadores mediante pipelines más limpios y eficientes, en los que casi cada instrucción se completa en uno o pocos ciclos del reloj. Además, estas arquitecturas suelen emplear muchos registros internos, de modo que la mayoría de las operaciones se realizan dentro del procesador sin necesidad de acceder constantemente a la memoria, lo que ahorra tiempo y energía. Para compensar esta simplicidad, se apoya en compiladores inteligentes y en un gran número de registros. Arquitecturas como MIPS (Microprocessor without Interlocked Pipeline Stages), ARM (Advanced RISC Machine) o PowerPC siguen este enfoque y son un buen ejemplo de cómo menos puede ser más cuando se diseña bien el hardware (Granda Candás, 2023).

Desde el punto de vista de la microarquitectura, RISC facilitó técnicas como la segmentación (pipeline) y el paralelismo a nivel de instrucciones. Al contar con instrucciones predecibles y simples, resulta más fácil dividir su ejecución en etapas y mantener ocupadas todas las unidades del procesador. En cambio, en CISC esto resulta más complicado porque las instrucciones tienen longitudes y tiempos de ejecución variables. Por esta razón, muchos procesadores CISC modernos internamente traducen sus instrucciones complejas en operaciones más simples, muy parecidas a las de los procesadores RISC, antes de ejecutarlas realmente (Englander & Wong, 2021).

Hoy en día, la frontera entre CISC y RISC ya no es tan clara como en los libros clásicos. Los procesadores actuales mezclan ideas de ambos mundos: los x86 siguen siendo CISC a nivel de programación, pero por dentro funcionan como máquinas RISC altamente optimizadas; mientras tanto, ARM ha incorporado instrucciones más complejas para mejorar el rendimiento (Figura 3.9).

3.5 Arquitecturas contemporáneas

Las arquitecturas contemporáneas de computadores parten de la idea de ejecutar más trabajo en menos tiempo, sin depender únicamente de aumentar la frecuencia del reloj. Hoy, el funcionamiento de un procesador moderno se apoya en una microarquitectura compleja, donde el ciclo de instrucción se divide en etapas y se ejecutan varias

ISBN: 978-631-6557-81-0

instrucciones al mismo tiempo mediante un pipeline y la ejecución fuera de orden. Para un estudiante de software, esto significa que el código ya no se ejecuta de forma estrictamente secuencial, aunque el modelo de programación lo oculte. El hardware se encarga de reordenar, adelantar y paralelizar instrucciones siempre que sea posible, buscando aprovechar cada ciclo de reloj.

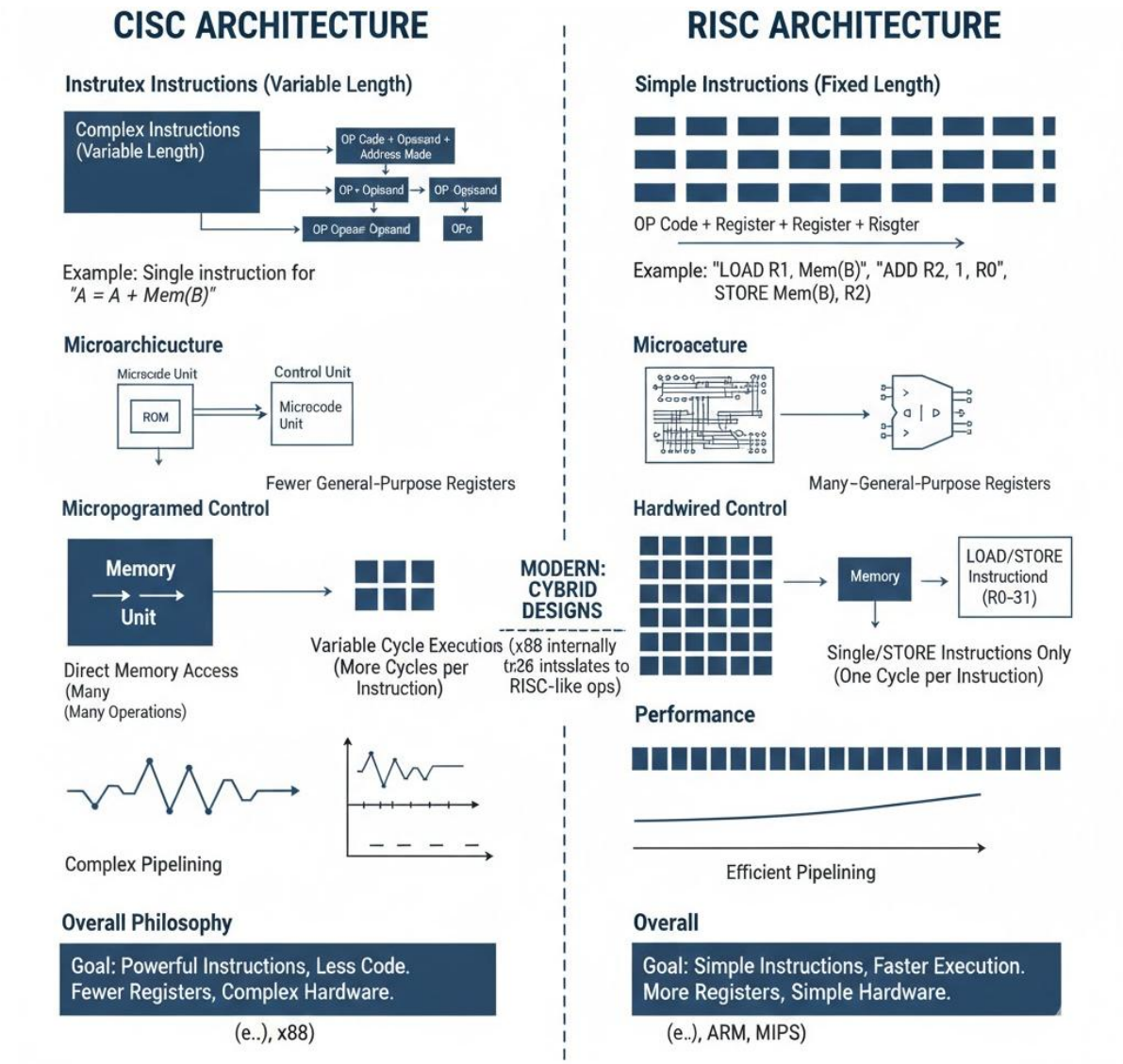


Figura 3.9 Descripción de las características clave de CISC y RISC.

Fuente: Gemini

ISBN: 978-631-6557-81-0

Otro rasgo clave de las arquitecturas actuales es el multinúcleo. En lugar de un único núcleo muy rápido, los procesadores integran varios núcleos que trabajan en paralelo y comparten niveles de memoria caché. Esto cambia la forma en que se entiende el rendimiento: ya no basta con un buen algoritmo, sino que también importa si el programa puede dividir su trabajo en hilos o procesos concurrentes. Conceptos como la coherencia de caché, la comunicación entre núcleos y la sincronización pasan a ser fundamentales, aunque muchas veces el programador solo los percibe cuando algo no escala como esperaba.

La tecnología superescalar en los procesadores surge como una respuesta directa a una limitación clara: ejecutar una sola instrucción por ciclo ya no era suficiente para mejorar el rendimiento. En un procesador superescalar, la idea principal es bastante intuitiva: aprovechar que muchas instrucciones de un programa son independientes entre sí y pueden ejecutarse simultáneamente. Para lograrlo, el procesador incorpora varias unidades de ejecución (por ejemplo, aritméticas, lógicas o de acceso a la memoria) y es capaz de despachar más de una instrucción en un mismo ciclo de reloj. Desde el punto de vista del software, el programa no cambia, pero internamente el hardware trabaja de forma más agresiva y en paralelo.

Para que esto funcione, el procesador debe analizar el flujo de instrucciones y detectar las dependencias. No todas las instrucciones pueden ejecutarse en paralelo, ya que algunas necesitan los resultados de otras. Aquí entran en juego mecanismos como el instruction dispatch, el out-of-order execution y los registros renombrados. Estos mecanismos permiten que el procesador reorganice el orden de ejecución sin alterar el resultado final del programa. Es un proceso complejo y, a veces, costoso en términos de hardware, pero muy efectivo para mejorar el rendimiento de aplicaciones comunes, incluso aquellas que no fueron diseñadas pensando en el paralelismo.

Por otro lado, ARM (Advanced RISC Machines*)* se ha convertido en la arquitectura dominante en dispositivos móviles y, cada vez más, en laptops y servidores. ARM sigue una filosofía RISC más clara: instrucciones más simples, menor consumo de energía y

ISBN: 978-631-6557-81-0

diseños más eficientes. Esto la hace ideal para smartphones, tablets y sistemas embebidos. En los últimos años, con procesadores como los Apple Silicon, ARM ha demostrado que también puede competir en alto rendimiento, no solo en eficiencia. Su crecimiento es clave hoy en día y muchos sistemas operativos y herramientas ya la tratan como una plataforma principal, no secundaria.

Una arquitectura más reciente y abierta es RISC-V. A diferencia de x86 y ARM, RISC-V no pertenece a una sola empresa y puede implementarse libremente. Esto la hace muy atractiva para la investigación, la educación y los nuevos diseños personalizados. Aunque todavía no domina el mercado de consumo, su adopción crece rápidamente en sistemas embebidos y aceleradores especializados. En conjunto, x86-64, ARM y RISC-V representan el núcleo de las arquitecturas de procesadores contemporáneas, cada una con fortalezas distintas, pero todas respaldadas por ideas modernas como el paralelismo, la jerarquía de memoria y la optimización agresiva del hardware.

Tareas complementarias.

1. Realiza una revisión de los modelos de procesadores más destacados de Intel y AMD desde sus inicios hasta la actualidad; incluye información sobre las innovaciones tecnológicas más importantes, como arquitecturas, tamaños de transistores, velocidades de reloj y funcionalidades introducidas. Representa esta información con una línea de tiempo realizada a mano.
2. Describe el significado de las letras y los números utilizados en la nomenclatura de las CPU actuales de las tecnologías Core y Zen. Por ejemplo, H: alto rendimiento para laptops de gaming o de productividad.
3. Realiza un análisis de las características funcionales de un microprocesador moderno de Intel y otro de AMD (el número de núcleos e hilos, los niveles de memoria caché, el tipo de arquitectura multinúcleo, la presencia de unidades especializadas (SIMD, IA, FPU), entre otras).
4. Responde las siguientes preguntas:
   ¿Qué beneficios tiene una CPU con multinúcleo?

ISBN: 978-631-6557-81-0

¿Cómo funciona la tecnología Turbo Boost?

Explica el funcionamiento de la técnica Pipeline

ISBN: 978-631-6557-81-0

CAPÍTULO 4.

4 Lenguaje Ensamblador

En este capítulo se conocerá cómo el procesador entiende y ejecuta un programa en lenguaje ensamblador, programando directamente en bajo nivel. Cada instrucción tiene un efecto directo sobre los registros, la memoria y el flujo de ejecución del programa. A diferencia de los lenguajes de alto nivel, cada instrucción corresponde a una operación concreta del hardware.

4.1 Elementos básicos del lenguaje ensamblador

El lenguaje ensamblador es un tipo de programación de bajo nivel que se convierte en código de máquina mediante la compilación al ensamblador. Los programas escritos en lenguaje ensamblador se conocen como **código fuente** (*.asm) y cada instrucción en ensamblador representa una operación específica del procesador y, simbólicamente, corresponde a una instrucción binaria en **código de máquina**. El ensamblador (por ejemplo, NASM o MASM) genera el código de máquina correspondiente a partir de este archivo y produce un archivo intermedio llamado archivo objeto, con la extensión *.obj. Después actúa el **enlazador** (linker), que toma los archivos obj y los une con las librerías del sistema para generar un archivo ejecutable *.exe.

Un código en ensamblador está formado por un conjunto de elementos básicos que permiten comunicarse casi directamente con el hardware. A diferencia de los lenguajes de alto nivel, aquí cada instrucción suele corresponder a una operación muy concreta del procesador. Uno de los primeros elementos que se aprenden son los mnemónicos, que son palabras cortas como MOV, ADD o JMP. Estas palabras representan instrucciones reales de la CPU y están pensadas para que el programador pueda recordarlas con facilidad, aunque al inicio cueste un poco acostumbrarse a su significado.

Los registros son otro componente clave del procesador y funcionan como pequeñas áreas de almacenamiento dentro del CPU. En el lenguaje ensamblador se trabaja

ISBN: 978-631-6557-81-0

constantemente con ellos, ya que la mayoría de las operaciones se realizan mediante registros (memoria intermediaria). Esto hace que el programador tenga que pensar más en cómo se mueven los datos, algo que no suele ser tan visible en lenguajes como Java o Python.

En el lenguaje ensamblador también se utilizan las directivas, que son instrucciones especiales que no ejecuta la CPU. Estas son indicaciones para el ensamblador que traduce el código al lenguaje de máquina. Con ellas se define, por ejemplo, dónde comienza el código, dónde se almacenan los datos o cómo se reserva espacio en memoria. Directivas como **.data, .code o .text** ayudan a organizar el programa y a separar claramente las instrucciones de los datos.

Otro elemento clave del ensamblador son las **etiquetas**, que permiten señalar puntos específicos dentro del código y se utilizan para controlar el flujo de ejecución mediante saltos y bucles. En lugar de trabajar con direcciones de memoria difíciles de leer, el programador utiliza nombres que hacen el código un poco más comprensible.

4.2 Arquitectura del procesador x86

Los procesadores x86 de 32 bits marcaron durante muchos años el estándar en los computadores personales. En este tipo de arquitectura, los registros principales del procesador, como EAX, EBX o ECX, tienen un tamaño de 32 bits, lo que permite manejar datos y direcciones de ese tamaño en una sola operación. Esto definía, entre otras cosas, el límite práctico de memoria direccionable, que era de 4 GB. En el modo de operación de 32 bits, el procesador trabaja normalmente en modo protegido, lo que permite utilizar mecanismos como la segmentación y la paginación para gestionar la memoria de forma segura. Esto fue clave para sistemas operativos como Windows y Linux, ya que evitaba que un programa accediera libremente a la memoria de otro. Desde el punto de vista del programador, el entorno de ejecución estaba bastante bien definido: un conjunto limitado de registros, direcciones de 32 bits y convenciones claras para las llamadas a funciones y el uso de la pila.

ISBN: 978-631-6557-81-0

Con la aparición de los procesadores x86 de 64 bits, la arquitectura dio un avance muy importante, que incluyó el aumento del tamaño de los registros generales a 64 bits, como RAX, RBX y RCX. Esto permitió trabajar con números de mayor longitud y amplió enormemente el espacio de direcciones de memoria, incorporando más registros generales (R8 a R15). Esto tiene un impacto directo en el rendimiento (Irvine, 2019). Al disponer de más registros, el compilador puede mantener más datos directamente en el procesador y reducir los accesos a la memoria, resultando que los programas se ejecuten más rápido (Figura 4.1).

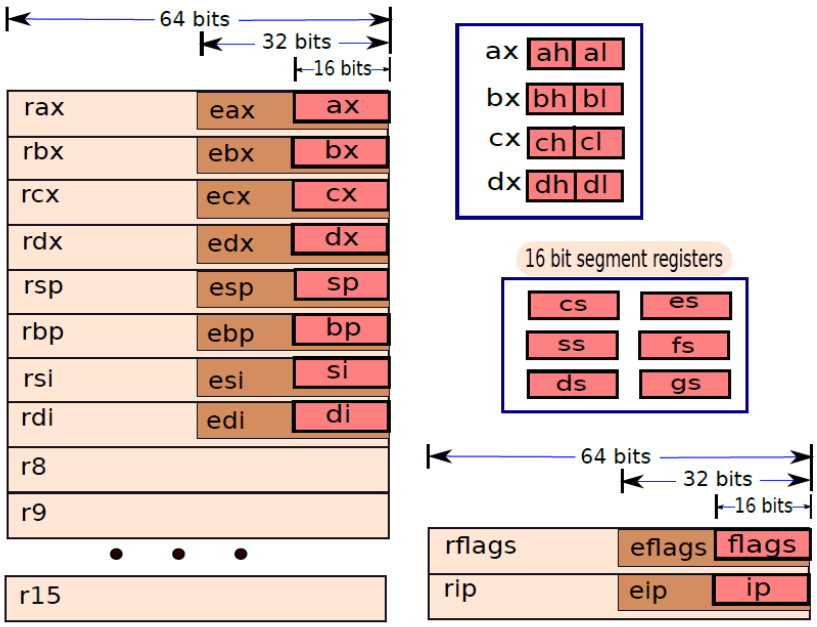


Figura 4.1 Registros de memoria en arquitectura x86

Fuente: https://www.cse.iitd.ac.in/~srsarangi/archbooksoft.html

Desde la perspectiva del desarrollo de software, las arquitecturas de 64 bits están diseñadas para mantener la compatibilidad con el código de 32 bits. Es decir, un mismo procesador puede ejecutar aplicaciones antiguas sin problema, siempre que el sistema operativo lo permita. Esto facilitó mucho la transición, ya que no fue necesario reescribir todos los programas. Aun así, el software compilado específicamente para 64 bits suele

ISBN: 978-631-6557-81-0

aprovechar mejor el hardware disponible, logrando un mayor rendimiento en tareas más demandantes.

4.3 Formato general de las instrucciones

Cuando uno va a programar por primera vez en el lenguaje ensamblador, lo primero que suele llamar la atención es lo distinto que se ve frente a lenguajes como C, Java o Python. No hay estructuras complejas ni palabras clave largas. En ensamblador, todo se reduce a instrucciones simples y directas. Justamente por eso, entender bien el formato general de una instrucción es clave antes de intentar escribir programas más complejos. En términos generales, una instrucción en ensamblador representa una operación específica que el procesador puede ejecutar directamente.

Cada **instrucción** se escribe en una sola línea e indica qué acción realizar y con qué datos ejecutarla. Aunque el formato puede variar según el ensamblador (MASM, NASM, GAS, entre otros). La forma general de una **instrucción** se compone de tres elementos principales: una **etiqueta** opcional, el **mnemónico de la instrucción** y uno o más **operandos**. La etiqueta (cuando existe) se escribe al inicio de la línea y sirve como nombre simbólico asociado a una dirección de memoria. Estas etiquetas son muy comunes en saltos condicionales y bucles. Por ejemplo, cuando el programa necesita volver a una instrucción previa o saltar a otra parte del código, se emplean etiquetas en lugar de direcciones numéricas. Esto mejora la claridad del código y facilita su mantenimiento. Aunque no todas las instrucciones requieren una etiqueta.

A continuación, aparece el **mnemónico**, que es el nombre abreviado de la instrucción. Estos nombres se derivan del inglés y describen la acción que se va a realizar, como mover datos, sumar, comparar, saltar, etc. Ejemplos clásicos son MOV, ADD, SUB, JMP o CMP. El campo mnemónico es obligatorio y, en esencia, es el corazón de la instrucción, ya que define la operación que el procesador realizará. Después del mnemónico se encuentran los **operandos**, que indican sobre qué datos actúa la instrucción. Según la operación, una instrucción puede no tener operandos, tener uno solo o tener dos.

ISBN: 978-631-6557-81-0

En las arquitecturas x86, lo más común es encontrar instrucciones con dos operandos, donde el primero suele ser el destino y el segundo, la fuente. El orden en que se colocan es fundamental y este punto suele generar confusión entre quienes provienen de lenguajes de alto nivel (Irvine, 2019). En ensamblador no hay mucha tolerancia a la ambigüedad: si los operandos están en el orden incorrecto, el resultado será erróneo o el ensamblador directamente rechazará la instrucción.

Los operandos pueden ser de distintos tipos. Un **operando** puede ser un **registro**, como EAX o RAX; una posición de memoria; o un valor inmediato, es decir, una constante escrita directamente en la instrucción. No todas las combinaciones son válidas y esto depende de la arquitectura y del tipo de instrucción. Por ejemplo, muchas instrucciones no admiten que ambos operandos sean posiciones de memoria al mismo tiempo, una restricción que no siempre resulta evidente al principio, pero que forma parte del funcionamiento interno del procesador. Un aspecto importante del ensamblador es el uso de comentarios, que suelen escribirse al final de la línea y comienzan con un símbolo especial: «;». Aunque el procesador los ignora por completo, su incorporación facilita la comprensión y explica la intención de la instrucción.

Es importante resaltar que el formato de las instrucciones en ensamblador está estrechamente ligado al hardware del procesador. Existen diferencias al escribir en ensamblador para x86 de 32 bits que para x64, y aún más para ARM u otras arquitecturas. Se cambian los registros disponibles, los tamaños de datos y, en ciertos casos, la forma de definir los operandos. A pesar de estas diferencias, el esquema mnemónico, más los operandos, se mantiene como patrón general.

Formato:

[etiqueta:] mnemónico operando(s) [; comentario]

Ejemplos:

a) Sin etiqueta, dos registros

```
MOV EAX, EBX
```

Mnemónico: MOV (mueve/asigna un valor)

ISBN: 978-631-6557-81-0

Operandos: EAX y EBX, ambos son registros

Qué hace: copia el contenido de EBX a EAX.

Idea clave: en x86 normalmente el operando izquierdo es el destino y el derecho es el origen

b) Con comentario al final

```
ADD EAX, 10    ; suma 10 al acumulador
```

Mnemónico: ADD (suma)

Operandos: EAX (registro) y 10 (constante inmediata)

Comentario: todo lo que va después de ; lo ignora el procesador, es para quien lea el código

Qué hace: suma 10 a EAX y guarda el resultado en EAX.

c) Etiqueta sola en la línea (marca de salto)

```
inicio:
```

Etiqueta: inicio termina en **:,** porque es una etiqueta de código

Qué significa: es un "nombre simbólico" para la dirección de esa instrucción (o del punto del código).

Para qué sirve: típicamente para saltos (JMP) y bucles

d) Etiqueta + instrucción en la misma línea

```
L1: MOV AX, BX
```

Etiqueta: L1:

Mnemónico: MOV

Operandos: AX, BX (registros de 16 bits)

Qué hace: marca esa línea con un nombre (L1) y, además, copia BX en AX.

Cuando se usa: cuando se desea que el código quede compacto. A veces se usa en ejemplos o en código corto.

e) Salto incondicional a etiqueta

```
JMP inicio
```

Mnemónico: JMP (salto)

Operando: inicio (una etiqueta de código)

ISBN: 978-631-6557-81-0

Qué hace: el procesador cambia el flujo y continúa ejecutando desde la dirección indicada por el inicio.

Uso típico: saltos a otra parte del código.

f) Salto condicional (depende de banderas)

```
JE L1     ; salta si es igual
```

Mnemónico: JE (Jump if Equal). Es una forma de Jcond (salto condicional)

Operando: L1 (etiqueta de destino).

Qué decide el salto: se basa en el estado de banderas, normalmente puestas por CMP

Lectura típica en código:

1. comparas con CMP
2. saltas con JE/JNE/JL/JG... según el caso.

g) Restricción típica: memoria a memoria no se permite con MOV

```
MOV var2, var1  ; Esto NO se permite en una sola instrucción MOV:
```

Regla importante: MOV no permite que ambos operandos sean memoria

Cómo se hace entonces: usando un registro intermedio (dos instrucciones):

```
MOV AX, var1
MOV var2, AX
```

4.4 Definiendo los datos

En ensamblador, definir datos es uno de los primeros pasos para que un programa tenga sentido. Antes de ejecutar instrucciones, el programa necesita saber qué datos va a manejar, dónde están y cuánto espacio ocupan. Esto se hace mediante directivas de definición de datos, que no generan instrucciones ejecutables, pero sí reservan espacio en memoria. En las arquitecturas x86, estas definiciones suelen colocarse en la sección de datos y permiten declarar constantes, variables simples o incluso estructuras más complejas. Se basan en **tipos de tamaño fijo**, lo cual es muy importante a bajo nivel. En lugar de tipos abstractos como en lenguajes de alto nivel, aquí trabajamos con tamaños concretos: **byte, word, doubleword, quadword**, entre otros (Tabla 4.1). Por ejemplo, un BYTE ocupa 8 bits (para caracteres o valores pequeños), mientras que un DWORD ocupa

ISBN: 978-631-6557-81-0

32 bits (para enteros). Se debe seleccionar el tipo de dato correcto para que no afecte directamente cómo el procesador accede y manipula la información.

Una variable **inicializada** tiene un valor definido desde el principio del código, mientras que una **no inicializada** solo reserva espacio en memoria. Esto es útil cuando se sabe que el valor se asignará más adelante durante la ejecución. En aplicaciones grandes, esta selección de variables ayuda a organizar mejor la memoria y reducir errores comunes, como el uso de datos sin valor. Otro aspecto clave al definir los datos es entender cómo se almacenan en memoria. En los sistemas x86 se utiliza el formato **little-endian**, por lo que el byte menos significativo se guarda primero. Esto no siempre es evidente al principio, pero es fundamental cuando se inspecciona memoria con un depurador o cuando se trabaja con datos de varios bytes. Saber cómo están realmente almacenados los valores evita confusiones al interpretar direcciones y contenidos de memoria.

En este contexto, aparecen estructuras comunes como los arreglos y las **cadenas**, que ocupan bloques de memoria contiguos. En un arreglo se almacenan varios datos del mismo tipo y una cadena suele ser una secuencia de bytes que representan texto. En estos casos, el programador es responsable de definir el tamaño y los límites de estos datos, lo que exige mayor atención, pero también proporciona un control preciso de la memoria.

TIPO	UTILIZACIÓN
BYTE	Entero sin signo de 8 bits. *B* significa byte.
SBYTE	Entero con signo de 8 bits. *S* significa signed (con signo).
WORD	Entero sin signo de 16 bits.
SWORD	Entero con signo de 16 bits.
DWORD	Entero sin signo de 32 bits. *D* significa double (doble palabra).
SDWORD	Entero con signo de 32 bits. *SD* significa signed double.
FWORD	Entero de 48 bits (puntero lejano en modo protegido).
QWORD	Entero de 64 bits. *Q* significa quad.
TBYTE	Entero de 80 bits (10 bytes). *T* significa ten-byte (diez bytes).
REAL4	Número real IEEE de 32 bits (4 bytes), precisión simple.
REAL8	Número real IEEE de 64 bits (8 bytes), precisión doble.
REAL10	Número real IEEE de 80 bits (10 bytes), precisión extendida.

Tabla 4.1 Tipos de datos intrínsecos. Elaborada por el autor.

ISBN: 978-631-6557-81-0

Ejemplos:

```
value1 BYTE 'A'              ; caracter literal
value2 BYTE 0                ; byte sin signo más pequeño
value3 BYTE 255              ; byte sin signo más grande
value4 SBYTE −128            ; byte con signo más pequeño
value5 SBYTE +127            ; byte con signo más grande
value6 BYTE ?                ; byte sin inicializar, el valor se va a asignar más
adelante
value7 BYTE 10h              ; byte inicializado con un valor hexadecimal
value8 SWORD -15             ; SWORD de 16 bits que almacena valores
                               positivos y negativos
value9 TBYTE 1234567890123456789    ; TBYTE ocupa 10 bytes, se usa para
                                      enteros muy grandes
list BYTE 10,20,30,40        ; arreglo de bytes
list1 BYTE 10, 32, 41h, 00100010b ; arreglo de bytes con distintas bases
numéricas
```

En lenguaje ensamblador, el operador DUP permite repetir una definición de dato varias veces sin tener que escribirla una y otra vez. En lugar de declarar cada elemento por separado, se indica cuántas veces se repite un valor o un espacio de memoria. Por ejemplo, se puede usar para crear un arreglo de varios bytes inicializados en cero o simplemente para reservar un bloque de memoria cuyo contenido se definirá más adelante.

```
BYTE 20 DUP(10)    ; 20 bytes, todos igual a 10.
BYTE 4 DUP("STACK")      ; 20 bytes, “STACKSTACKSTACKSTACK”
```

4.5 Instrucciones Aritméticas y Operaciones

Las instrucciones aritméticas permiten al procesador realizar operaciones matemáticas básicas, como sumas, restas, multiplicaciones o divisiones. Estas instrucciones trabajan directamente con los registros y, en algunos casos, con datos almacenados en memoria. A diferencia de los lenguajes de alto nivel, aquí no existen

ISBN: 978-631-6557-81-0

expresiones complejas en una sola línea: cada operación se descompone en pasos simples. Esto puede parecer tedioso al inicio, pero ayuda a entender con claridad qué hace realmente el procesador en cada ciclo. Una de las instrucciones más usadas es la suma, representada normalmente por el mnemónico ADD. Su formato típico sigue la idea de destino y fuente. Por ejemplo, una instrucción como ADD EAX, EBX indica que el contenido de EBX se suma al de EAX y el resultado se guarda en EAX. Algo importante que a veces confunde es que el operando de destino también participa como entrada; no es solo un lugar donde cae el resultado. Además, estas operaciones afectan a las banderas del procesador, como la bandera de cero o la de acarreo, que luego pueden usarse en decisiones condicionales.

La resta utiliza la instrucción SUB de la siguiente manera: SUB EAX, 5; en la que se descuenta un valor inmediato al contenido del registro EAX. También existen instrucciones más pequeñas como **INC y DEC**, que incrementan o decrementan un registro en una unidad. Aunque parecen detalles menores, estas instrucciones se utilizan para sumar o restar uno, ya que suelen ser más eficientes en estos casos. Para operaciones más complejas, como la multiplicación y la división, se realizan con MUL o IMUL y con **DIV o IDIV**. En la práctica, estas instrucciones muestran una idea clave del lenguaje ensamblador: nada es automático. Cada operación aritmética tiene reglas claras y entenderlas bien es fundamental para escribir códigos eficientes.

Formatos:

ADD destino, Fuente

SUB destino, Fuente

MUL operando (sin signo)

IMUL operando (con signo)

DIV operando (sin signo)
IDIV operando (con signo)

Ejemplos:

a) **ADD** EAX, EBX
Este ejemplo suma el contenido del registro EBX al de EAX. El resultado final queda almacenado en EAX.

ISBN: 978-631-6557-81-0

b) **SUB** EAX, 10

Aquí se resta el valor inmediato 10 al contenido de EAX. Nuevamente, el resultado se guarda en el mismo registro EAX.

c) **IMUL** EAX, ECX

En este caso, el contenido de EAX se multiplica por el de ECX y el resultado se almacena en EAX. Esta forma de IMUL es más clara y fácil de usar que la versión con registros implícitos.

d) **IDIV** EBX

Este ejemplo divide el valor almacenado en los registros que forman el dividendo entre el contenido de EBX. El cociente se guarda en EAX y el residuo en EDX, por lo que antes de usar esta instrucción es necesario preparar correctamente esos registros

e) **MOV** EAX, 20
CDQ

MOV EBX, 4

IDIV EBX

En este caso, primero se carga el valor 20 en el registro EAX. La instrucción CDQ (Convert Doubleword to Quadword) se usa para preparar una división con signo en ensamblador x86 y extiende el signo de EAX a EDX, preparando correctamente el dividendo. Luego se coloca el valor 4 en EBX y se ejecuta IDIV EBX.

Después de la división, el cociente queda en EAX (20 ÷ 4 = 5) y el residuo en EDX. Este ejemplo muestra por qué la división en ensamblador requiere más pasos que la suma o la resta, y por qué es importante preparar bien los registros antes de usar IDIV.

f)
```
MOV EAX, 12     ; Cargamos el primer número
ADD EAX, -18    ; Sumamos -18 ; Resultado de la suma: EAX = -6

MOV EBX, 12     ; Volvemos a cargar 12

IMUL EBX, -18   ; Multiplicamos 12 por -18 ; Resultado de la multiplicación:
EBX = -216
```

En este código se suman y se multiplican dos números: 12 y -18.

4.6 Directivas

Las **directivas** son órdenes dirigidas al ensamblador que indican cómo organizar el programa, qué espacio de memoria se reserva, qué constantes se definen o en qué sección

ISBN: 978-631-6557-81-0

del código va cada parte. No producen instrucciones ejecutables por sí mismas, pero son esenciales para que el programa tenga sentido y pueda ensamblarse correctamente. Las directivas más comunes son **.data, .code o .stack,** que sirven para separar claramente las zonas de datos, de código y de pila, lo que ayuda mucho a mantener el programa ordenado y legible. Por ejemplo, permiten definir variables (**BYTE, WORD, DWORD**), crear constantes simbólicas o indicar el inicio y el fin de un procedimiento. También existen directivas más avanzadas para control condicional o macros que ayudan a reducir la repetición y hacer el código más claro y manejable.

Ejemplo:

```
.data                    ; Directiva: sección de datos
num1 DWORD 20            ; Definición de una variable
num2 DWORD 8
resultado DWORD ?
.code                    ; Directiva: sección de código
main PROC                ; Directiva: inicio del procedimiento principal
        mov eax, num1    ; Carga el primer número en el registro
        sub eax, num2    ; Resta el segundo número
        mov resultado, eax ; Guarda el resultado en memoria
        ret              ; Fin del programa
main ENDP                ; Directiva: fin del procedimiento
END main                 ; Directiva: indica el punto de inicio del programa
```

La directiva **.data** señala la sección donde se definen las variables, constantes y espacios de memoria que utilizará el programa, dejando claro que en ella no hay instrucciones ejecutables. Por su parte, **.code** indica el inicio de la sección de código, donde se colocan las instrucciones que el procesador sí ejecuta, como operaciones aritméticas o movimientos de datos. Finalmente, **PROC** se usa para marcar el comienzo de un procedimiento, que puede entenderse como una función, lo que permite agrupar instrucciones en bloques bien definidos y facilita que el ensamblador y el enlazador identifiquen el punto principal de ejecución.

ISBN: 978-631-6557-81-0

4.7 Desplazamiento

Las **instrucciones de desplazamiento** se utilizan para mover los bits de un dato hacia la izquierda o hacia la derecha dentro de un registro o en una posición de memoria. A simple vista parecen operaciones muy básicas, pero en realidad son bastante potentes. Al desplazar bits a la izquierda, el valor binario se multiplica por potencias de dos; al desplazarlos a la derecha, se divide. Por ejemplo, un desplazamiento a la izquierda de una posición equivale, en muchos casos, a multiplicar por dos. Por eso, estas instrucciones se usan mucho cuando se busca eficiencia, ya que suelen ser más rápidas que la multiplicación o la división tradicionales.

En arquitecturas como x86 aparecen con frecuencia instrucciones de desplazamiento de bits como **SHL**, **SHR** o **SAR**, cada una con un rol concreto. SHL desplaza los bits a la izquierda, mientras que SHR los desplaza a la derecha, rellenando con ceros. En cambio, SAR desplaza a la derecha, pero conserva el bit de signo, lo cual es importante al trabajar con este tipo de números. Estas instrucciones se usan comúnmente para manipular bits individuales, trabajar con campos dentro de una palabra de datos y activar o desactivar banderas.

Ejemplos:

```
SHL AX, 1     ; todos los bits del registro AX se mueven una posición a la izquierda. Este
                desplazamiento equivale a multiplicar el valor por 2.
SHR BX, 1     ; los bits del registro BX se desplazan una posición a la derecha y el bit más
                significativo se rellena con 0. Se interpreta como una división entera entre
                2, pero solo funciona correctamente con números sin signo.
SAR CX, 1     ; Desplaza los bits hacia la derecha, pero mantiene el signo del número.
```

4.8 Saltos condicionales e incondicionales

Los saltos permiten controlar el flujo de ejecución de un código en lenguaje ensamblador. A diferencia de lenguajes de alto nivel, que ofrecen estructuras de control como if, while o for, en el ensamblador la instrucción decide si continúa en secuencia o se

ISBN: 978-631-6557-81-0

transfiere la ejecución a otra dirección concreta. Por eso, entender con claridad el funcionamiento de los saltos es clave para leer, escribir y depurar programas.

Los saltos incondicionales son los más simples. Cuando el procesador encuentra uno de estos, la ejecución siempre se desvía a la dirección indicada, sin evaluar ninguna condición previa. Se utilizan, por ejemplo, para repetir bloques de instrucciones, salir de una sección del programa o reorganizar el flujo de ejecución de forma directa. En cambio, los saltos condicionales dependen del estado de las banderas del procesador, que se actualizan normalmente después de una comparación (CMP) o de una operación aritmética. Estas banderas indican cosas como si el resultado fuera cero o negativo, o si hubo desbordamiento. Según su valor, el salto se ejecuta o no. De esta forma se construyen decisiones lógicas, equivalentes a un if en lenguajes de alto nivel. Aunque al principio puede parecer más complejo, este mecanismo ofrece un control muy preciso del comportamiento del programa.

Ejemplos:

```
JMP etiqueta   ; salto incondicional: la ejecución continúa siempre en etiqueta.
JE etiqueta    ; salto condicional: se ejecuta si el resultado anterior fue igual
               (bandera Z  activada).
```

4.9 Bucles

En el lenguaje ensamblador, el manejo de bucles es esencial para repetir tareas sin duplicar código. A diferencia de lenguajes de alto nivel como C o Java, que tienen bucles de repetición, como for o while, de forma explícita. En su lugar, el ensamblador construye los bucles combinando instrucciones simples, como comparaciones, saltos condicionales o incondicionales, y, en algunos casos, contadores. Cada repetición del bucle queda claramente definida por las instrucciones que controlan el flujo de ejecución, lo que facilita comprender, paso a paso, cómo el hardware decide si continuar o finalizar una operación repetitiva (Irvine & Das, 2011).

Por ejemplo, se ejecuta un bloque de instrucciones, luego se evalúa una condición y, según el resultado, el programa decide si vuelve al inicio del bloque o continúa con la

ISBN: 978-631-6557-81-0

siguiente parte del código. Todo este control se realiza explícitamente mediante registros y banderas del procesador, lo que obliga al programador a pensar con cuidado en el orden de las instrucciones.

En muchos casos, los bucles utilizan un **registro como contador**. En cada iteración del bucle, el procesador va disminuyendo o aumentando ese registro de forma controlada. Cuando el contador alcanza cierto valor, normalmente cero, el bucle finaliza, lo que resulta sencillo y eficiente de implementar. También es importante entender que los bucles en el ensamblador dependen en gran medida de las **banderas** del procesador, como la bandera de cero (ZF), que se activan o desactivan según el resultado de operaciones previas (comparaciones o restas). Luego, las instrucciones de salto consultan esas banderas para decidir si deben regresar al inicio del bucle o continuar con la ejecución normal del programa.

Formato:

LOOP etiqueta	; disminuye el registro CX y salta a la etiqueta si CX no es cero.
JNZ etiqueta	; salta a la etiqueta si el resultado anterior no fue cero.
CMP AX, BX	; compara dos valores y ajusta las banderas para decidir un salto posterior.

Ejemplos:

a) Dame un código que sume dos números ingresados por teclado y repita esta actividad 5 veces utilizando el bucle for.
Idea en un lenguaje de alto nivel (para compararlo)

```
for (i = 0; i < 5; i++) {
      leer a;
      leer b;
      suma = a + b;
}
```

En ensamblador x86

ISBN: 978-631-6557-81-0

```
MOV CX, 5        ; CX será el contador del bucle (5 repeticiones)
BUCLE:
   ; Leer primer número (simulado)
   CALL LEER_NUM
   MOV AX, BX   ; guardamos el primer número en AX
   ; Leer segundo número (simulado)
   CALL LEER_NUM
   ADD AX, BX   ; AX = AX + BX (suma de los dos números)
   ; Aquí AX contiene el resultado de la suma
 LOOP BUCLE   ; CX = CX - 1, si CX != 0 vuelve a BUCLE
LEER_NUM:
 MOV AH, 01h          ; Leer un carácter del teclado
 INT 21h
SUB AL, '0'           ; Convertir ASCII a número
 RET
```

Explicación: El programa comienza inicializando el segmento de datos y luego carga el registro CX con el valor 5, que actúa como contador del bucle. Ese bloque, etiquetado como BUCLE, se ejecuta mientras CX sea distinto de cero, lo que imita el comportamiento de un bucle for. Dentro del bucle se llama dos veces a la subrutina LEER_NUM. Cada llamada lee un número del teclado y lo devuelve en el registro AL. El primer valor se guarda en BL y el segundo en BH. Luego, se realiza la suma con ADD BL y BH. La instrucción DEC CX reduce el contador y CMP, junto con JNZ, decide si el bucle debe repetirse.

b) Dame un código que reste dos números ingresados por teclado y repita esta actividad 3 veces utilizando el bucle while.
Idea en un lenguaje de alto nivel (referencia mental)

```
int contador = 3;
while (contador > 0) {
      leer a;
```

ISBN: 978-631-6557-81-0

```
            leer b;
            resultado = a - b;
            contador--;
        }
```

En ensamblador x86

```
    MOV CX, 3       ; contador del while (3 repeticiones)
    WHILE_LOOP:
      CALL LEER_NUM         ; Leer primer número
      MOV AX, BX            ; guardar primer número en AX
      CALL LEER_NUM         ; Leer segundo número
      SUB AX, BX   ; AX = primer número - segundo número
                    ; aquí AX contiene el resultado de la resta
      DEC CX        ; contador--
      CMP CX, 0
    JG WHILE_LOOP           ; mientras CX > 0, repetir
```

Explicación: Primero se carga el valor 3 en el registro CX, que actuará como el contador del bucle, igual que una variable en un while. Luego comienza el bucle en la etiqueta WHILE_LOOP. Dentro del bucle se simula la lectura de dos números desde el teclado mediante una rutina llamada LEER_NUM. El valor leído se asume en el registro BX. El primer número se guarda en AX; luego se lee el segundo y se realiza la resta con la instrucción SUB. El resultado queda almacenado en AX. Después, el contador se decrementa con DEC CX. Finalmente, se compara el contador con cero y, si aún es mayor que cero, el programa vuelve a saltar al inicio del bucle.

4.10 Ejemplos de programación

Ejemplo 1: “if” con CMP + salto condicional

Generar un código que simula un if simple: si EAX es igual a EBX, se guarde 1; en caso contrario, se almacene 0.

ISBN: 978-631-6557-81-0

```
.data
flag DWORD ?
.code
main PROC
   mov eax, 5
   mov ebx, 5
   cmp eax, ebx          ; compara: ajusta banderas
   je IGUALES            ; si son iguales, salto

NO_IGUALES:
   mov flag, 0
   jmp FIN               ; salto incondicional para salir

IGUALES:
   mov flag, 1

FIN:
   RET           ; sale de la función
main ENDP
END main
```

Ejemplo 2: Genera el código para leer 2 números de 4 cifras por teclado, realizar las 4 operaciones básicas y visualizar los resultados de la suma, resta, multiplicación y división.

```
.data
msg1 db "Ingrese el primer numero (4 cifras): $"
msg2 db 13,10,"Ingrese el segundo numero (4 cifras): $"
```

ISBN: 978-631-6557-81-0

```
msgS db 13,10,"Suma: $"
msgR db 13,10,"Resta: $"
msgM db 13,10,"Multiplicacion: $"
msgD db 13,10,"Division: $"
num1 dw ?                ; Variable para el primer número
num2 dw ?                ; Variable para el segundo número
.code
main PROC
   mov ax, @data          ; Inicializa el segmento de datos
   mov ds, ax

   ; ---- Leer primer número ----
   mov ah, 09h            ; Servicio DOS: imprimir cadena
   lea dx, msg1           ; Dirección del mensaje
   int 21h
   call LeerNumero        ; Llama a la rutina de lectura
   mov num1, ax           ; Guarda el número leído en num1

   ; ---- Leer segundo número ----
   mov ah, 09h
   lea dx, msg2
   int 21h
   call LeerNumero
   mov num2, ax           ; Guarda el número leído en num2

   ; ---- SUMA ----
   mov ax, num1           ; AX = num1
   add ax, num2           ; AX = num1 + num2
```

ISBN: 978-631-6557-81-0

```
mov ah, 09h
lea dx, msgS
int 21h
call ImprimirNumero        ; Imprime el resultado en AX

; ---- RESTA ----
mov ax, num1               ; AX = num1
sub ax, num2               ; AX = num1 - num2
mov ah, 09h
lea dx, msgR
int 21h
call ImprimirNumero

; ---- MULTIPLICACION ----
mov ax, num1               ; AX = num1
mov bx, num2               ; BX = num2
mul bx                     ; AX = AX * BX (resultado en AX)
mov ah, 09h
lea dx, msgM
int 21h
call ImprimirNumero

; ---- DIVISION ----
mov ax, num1               ; AX = dividendo
xor dx, dx                 ; Limpia DX (requerido para DIV)
mov bx, num2               ; BX = divisor
div bx                     ; AX = cociente, DX = residuo
mov ah, 09h
```

ISBN: 978-631-6557-81-0

```
    lea dx, msgD
    int 21h
    call ImprimirNumero

    ; ---- Finalizar programa ----
    mov ah, 4Ch             ; Servicio DOS: terminar programa
    int 21h
main ENDP
; ------------------------------------------------
; Rutina: LeerNumero
; Lee un número decimal desde el teclado
; Devuelve el valor en AX
; ------------------------------------------------
LeerNumero PROC
    xor ax, ax              ; AX = 0 (acumulador del número)
    xor bx, bx              ; BX = 0 (dígito temporal)

LeerDigito:
    mov ah, 01h             ; Leer un carácter del teclado
    int 21h
    cmp al, 13              ; ¿Es ENTER?
    je FinLeer
    sub al, '0'             ; Convierte ASCII a número
    mov bl, al              ; BL = dígito
    mov cx, ax              ; Guarda AX temporalmente
    mov ax, 10
    mul cx                  ; AX = AX * 10
    add ax, bx              ; AX = AX + dígito
```

ISBN: 978-631-6557-81-0

```
    jmp LeerDigito          ; Leer siguiente carácter

FinLeer:
    ret
LeerNumero ENDP

; ------------------------------------------------
; Rutina: ImprimirNumero
; Imprime el número almacenado en AX
; ------------------------------------------------
ImprimirNumero PROC
    mov bx, 10              ; Divisor decimal
    xor cx, cx              ; Contador de dígitos
Convierte:
    xor dx, dx              ; Limpia DX antes de DIV
    div bx                  ; AX = AX/10, DX = resto
    push dx                 ; Guarda el dígito
    inc cx                  ; Aumenta contador
    cmp ax, 0
    jne Convierte
Imprime:
    pop dx                  ; Recupera dígito
    add dl, '0'             ; Convierte a ASCII
    mov ah, 02h             ; Imprime carácter
    int 21h
    loop Imprime
    RET
ImprimirNumero ENDP
```

ISBN: 978-631-6557-81-0

END main

Tareas Complementarias:

Genere el código en ensamblador que:

a) Lea 5 números enteros desde el teclado, determine si cada número es par o impar, acumule en dos contadores y muestre la cantidad de números pares e impares.

b) Lea 4 números enteros, sume todos los valores ingresados, calcule el promedio y muestre el resultado.

c) Lea un número entero positivo y muestre la tabla de multiplicar del 1 al 12. En cada línea deben mostrarse el multiplicador y el resultado. Por ejemplo, 5 * 1 = 5.

ISBN: 978-631-6557-81-0

ISBN: 978-631-6557-81-0

BIBLIOGRAFÍA

Angulo Usategui, J. A. M., Ignacio García Zubia. (2007). *Sistemas digitales y tecnología de computadores*. Ediciones Paraninfo, SA.

Arikpo, I., Ogban, F., Eteng, I. (2007). Von neumann architecture and modern computers. *Global Journal of Mathematical Sciences*, *6*(2), 97-103.

Arriarán, S. S. (2015). Respuestas a las preguntas del capítulo 2 [Todo Sobre Sistemas Embebidos].

Blaauw, G. A., Brooks Jr, F. P. (1997). *Computer architecture: Concepts and evolution*. Addison-Wesley Longman Publishing Co., Inc.

Bonifacio Ceferino, E. (2018). ARQUITECTURA DEL COMPUTADOR.

Denning, P. J. J. A. C. S. (1971). Third generation computer systems. *3*(4), 175-216.

Dongarra, J., Duff, I. S. (2020). Advanced architecture computers. In *Supercomputing in Engineering Analysis* (pp. 19-62). CRC Press.

Englander, I., Wong, W. (2021). *The architecture of computer hardware, systems software, and networking: An information technology approach*. John Wiley & Sons.

Entrialgo Castaño, J. (2019). Computadores y redes.

Entrialgo Castaño, J., Granda Candás, J., López López, J., Molleda Meré, J., Arias García, J., Usamentiaga Fernández, R., . . . Díaz de Arriba, J. (2023). Computadores. In: Ediuno, Ediciones de la Universidad de Oviedo.

Granda Candás, J. C. (2023). Arquitectura de computadores.

Hamacher, V. C., Vranesic, Z. G., Zaky, S. G., Vransic, Z., Zakay, S. (1984). *Computer organization*. McGraw-Hill New York.

Hennessy, J. L., Patterson, D. A. (2017). *Computer architecture: a quantitative approach*. Elsevier.

Ibanez, N. (1999). El problema de von Neumann sobre la medibilidad de ciertas álgebras de Boole.

Irvine, K. R. (2019). *Assembly language for x86 processors 8th edition*. Prentice Hall.

Irvine, K. R., Das, L. B. (2011). *Assembly language for x86 processors*. Prentice Hall.

Jara Castro, S. (2015). *Taller de Cómputo*. Ediciones Umbral.

Kunysz, E. J., Morales, D. M., Osio, J. R., Rapallini, J. (2016). Análisis de eficiencia en sistemas de computo de alta performance reconfigurable.

Kurzweil, R. (2005). The singularity is near. In *Ethics and emerging technologies* (pp. 393-406). Springer.

López, M. J. A. P., Rueda, M. K. H. (2009). Lecturas de Apoyo de Arquitectura de Computadoras.

Mir, S. B., Catalán, M. C., Llueca, G. F., Fernández, J. C. F., Navarro, G. L., Avilés, J. V. M., . . . Colás, R. M. (2019). Introducción a la arquitectura de computadores con QtARMSim y Arduino. In.

Moto-Oka, T. (2012). *Fifth generation computer systems*. Elsevier.

Murdocca, M., Heuring, V. P., Szklanny, F., de María, E. (2002). *Principios de arquitectura de computadoras*. Pearson Educación.

Null, L., Lobur, J. (2014). *The essentials of computer organization and architecture*. Jones & Bartlett Publishers.

Ocrospoma Callupe, B. M. (2021). ARQUITECTURA DE COMPUTADORAS Hardware: El Microprocesador, Las Memorias, Controladores de Hardware, Puertos de Comunicación, dispositivos de Almacenamiento, unidades de Disco. Ensamblaje y reparación de computadoras.

Orenga, M. A., Manonellas, G. E. (2011). *Estructura de computadores*. Universitat Oberta de Catalunya.

Ortega, J., Anguita, M., Prieto, A., Cañas, A., Damas, M., Díaz, A. F., . . . González, J. (2019). La hoja de ruta de la ingeniería de computadores al final de la ley de Moore y el escalado de Dennard. *Enseñanza y aprendizaje de ingeniería de computadores: Revista de Experiencias Docentes en Ingeniería de Computadores*, *9*, 5-28.

Patterson, D. A., Hennessy, J. L. (2000). *Estructura y diseño de computadores* (Vol. 1). Reverté.

Pérez, C., Facchini, H., Argüello, D. M. (2005). Arquitectura de computadoras. *Editorial Mc-Graw Hill 2a Edición, México*.

Quiroga, P. (2010). *Arquitectura de computadoras*. Alpha Editorial.

Ramírez, L. I. L. (2009). Introducción a las computadoras. *Apuntes , Recinto Universitario de Mayagüez, Universidad de Puerto Rico*. http://www.uprm.edu/cti/docs/manuales/manuales-espanol/vax-vms/manuales/Intcomp.pdf

ISBN: 978-631-6557-81-0

Stallings, W. (2019). *Computer organization and architecture: designing for performance*. Pearson Education India.

Stallings, W., Vargas, A. C., Espinosa, A. P. (2006). *Organización y arquitectura de computadores*. Pearson Educación.

Tanenbaum, A. S. (2016). *Structured computer organization*. Pearson Education India.

Vázquez Gómez, J. B. (2012). Arquitectura de computadoras. In: Red Tercer Milenio.

Villarreal, P. A. (2017). Visión moderna de los cálculos básicos en una computadora con estructura de Jhon Von Neumann. *Ingeniería UVM. Revista Electrónica Científico-Técnica*, *11*(1).

Wessels, D., Weaver, M. (2008). *Make Projects: Small Form Factor PCs*. " O'Reilly Media, Inc.".

Yadin, A. (2016). *Computer systems architecture*. Chapman and Hall/CRC.

ISBN: 978-631-6557-81-0

ISBN: 978-631-6557-81-0

AUTOR

DIEGO FERNANDO AVILA PESANTEZ

Doctor en Ingeniería de Sistemas e Informática por la Universidad Nacional Mayor de San Marcos (UNMSM) en Perú. Máster en Informática Aplicada por la Escuela Superior Politécnica de Chimborazo (ESPOCH). Ingeniero en Sistemas Informáticos por la Universidad Católica de Cuenca. Complementa su formación académica con diversos diplomados en educación superior e investigación científica. Cuenta con estancias posdoctorales en la Universidad de Castilla-La Mancha (UCLM), España, y en la Universidade Estadual do Norte Fluminense Darcy Ribeiro (UENF), Brasil, donde fortaleció su experiencia en investigación y desarrollo académico. Ha publicado más de 30 artículos científicos en Scopus y en revistas indexadas. En el ámbito profesional, ha ejercido como analista de sistemas en el Banco Centro Mundo, sucursal Riobamba, y en el Banco de Fomento. Posee más de 30 años de experiencia como docente universitario, participando en la formación de profesionales de las áreas de redes, sistemas e informática. Actualmente se desempeña como docente en la ESPOCH e investigador del grupo GIITYC, con trabajos enfocados en tecnología, sistemas de información e innovación educativa.

PUERTO MADERO
EDITORIAL

ISBN 978-631-6557-81-0