Introducción a Arquitectura de las Computadoras

Diego Feroldi

Arquitectura del Computador*
Departamento de Ciencias de la Computación
FCEIA-UNR

*Actualizado 28 de agosto de 2025 (D. Feroldi, feroldi@fceia.unr.edu.ar)

Índice

1. Definición de computadora

> *"Una computadora digital es una máquina que puede resolver problemas ejecutando las instrucciones que recibe de las personas".*
>
> **Andrew S. Tanenbaum**
> Organización de computadoras: un enfoque estructurado, 2000.

La definición anterior es solo una entre muchas posibles de lo que es una computadora. Esta en particular sugiere que, a pesar de las enormes prestaciones de las computadoras actuales, en su nivel más elemental estas máquinas realizan operaciones muy simples. De hecho, es importante destacar que los circuitos integrados de una computadora solo pueden reconocer y ejecutar un conjunto muy limitado de operaciones básicas.

A lo largo de esta asignatura, exploraremos cómo se llevan a cabo dichas operaciones. Para ello, comenzaremos por analizar cómo están constituidas las computadoras. Además, haremos un breve repaso de su evolución histórica, desde sus orígenes hasta convertirse en las potentes herramientas que utilizamos en la actualidad.

2. Descripción básica de una computadora

Los componentes básicos de una computadora incluyen una Unidad Central de Procesamiento (CPU), almacenamiento primario o memoria de acceso aleatorio (RAM, por su sigla en inglés *Random Access Memory*), almacenamiento secundario (disco duro, disco de estado sólido, memoria USB, etc.), dispositivos de entrada/salida (pantalla, teclado, mouse, etc.) y una interconexión entre los diferentes componentes denominada buses. Un diagrama muy básico de la arquitectura de una computadora se muestra en la Figura 1.

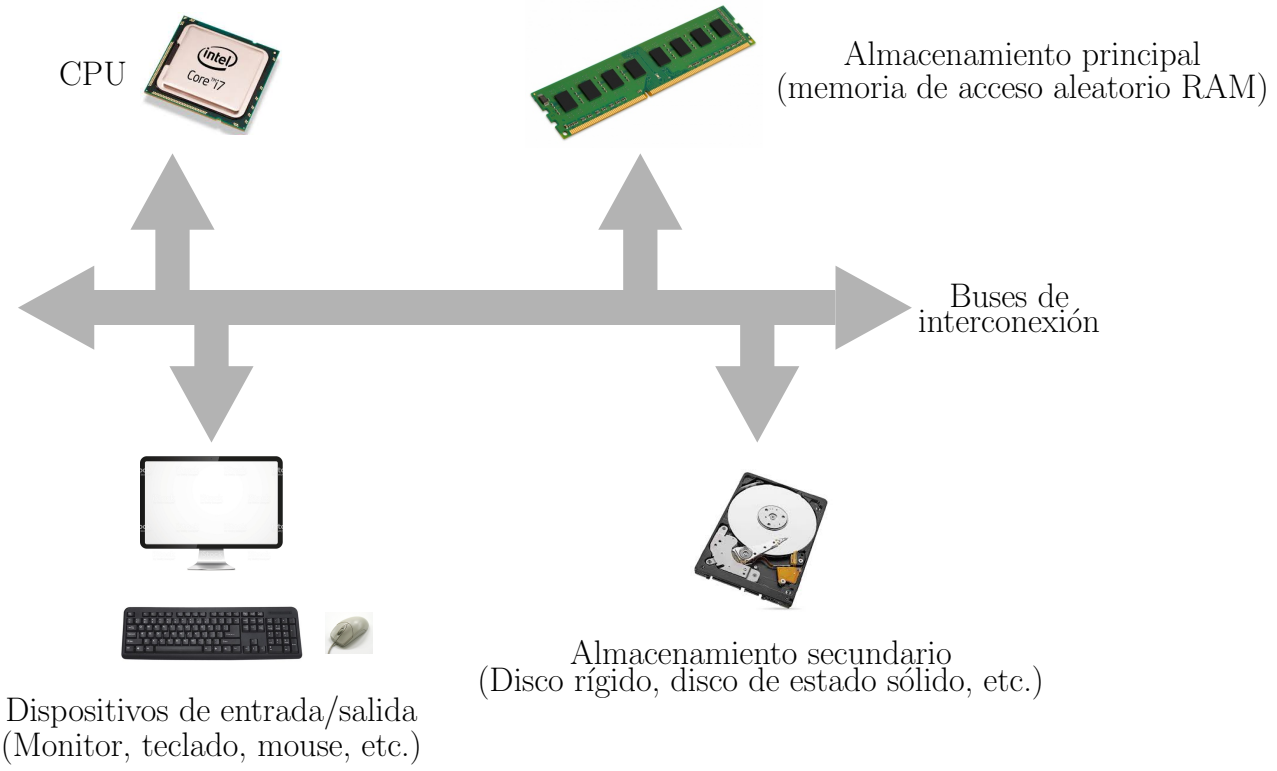


Figura 1: Componentes básicos de una computadora.

3. Componentes principales de una computadora

3.1. Unidad Central de Procesamiento (CPU)

La unidad central de procesamiento (CPU), o simplemente el procesador, es el "motor" que interpreta (o ejecuta) las instrucciones almacenadas en la memoria principal. Estos son algunos ejemplos de las operaciones simples que la CPU podría realizar a pedido de una instrucción:

- **Cargar**: Copiar información de la memoria principal en un registro, sobrescribiendo los contenidos anteriores del registro.
- **Almacenar**: Copiar información de un registro a una ubicación en la memoria principal, sobrescribiendo el contenido anterior de esa ubicación.
- **Operar**: Realizar operaciones con el contenido de registros de la ALU y almacenar el resultado en un registro, sobrescribiendo el contenido anterior de ese registro.
- **Saltar**: Modificar la secuencia de ejecución de instrucciones, modificando el contenido del contador de programa (PC).

La Figura 2 muestra de manera esquemática una CPU con sus principales componentes.

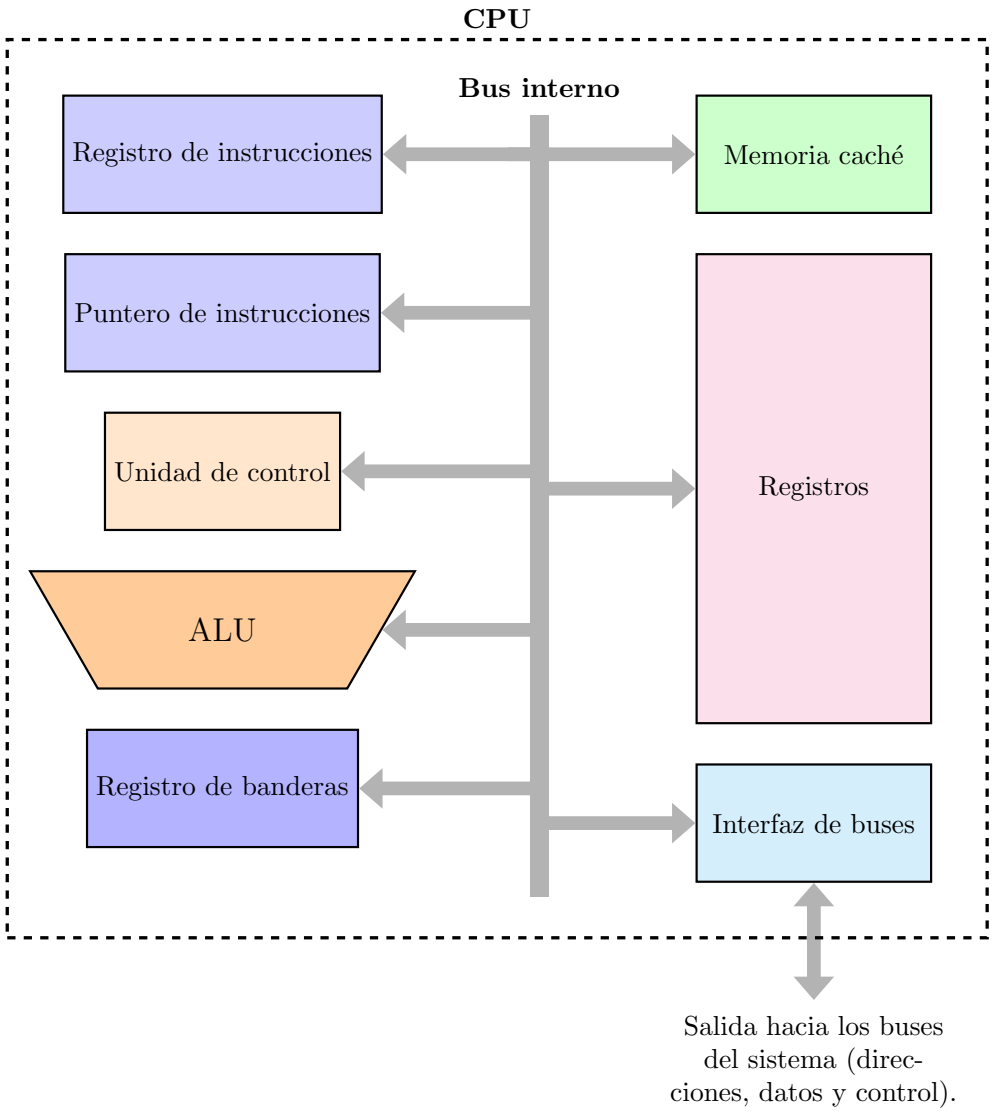


Figura 2: Vista general de los componentes principales de la CPU.

3.2. Unidad Aritmética Lógica (ALU)

La CPU incluye varias unidades funcionales, incluida la unidad aritmética lógica (ALU, por su sigla en inglés *Aritmetic Logic Unit*), que es la parte del chip que realmente realiza los cálculos aritméticos y lógicos. La función de la ALU es calcular nuevos datos y valores de dirección. A nivel de la ALU solo hay disponibles algunas pocas operaciones muy simple:

- Operaciones aritméticas entre operandos,
- Operaciones lógica de bits,
- Operaciones de desplazamiento de bits.

La Figura 3 presenta un esquema de la ALU, cuyas entradas incluyen los dos operandos y la señal de control que determina la operación a ejecutar. Las salidas comprenden el resultado de la operación y las banderas asociadas.

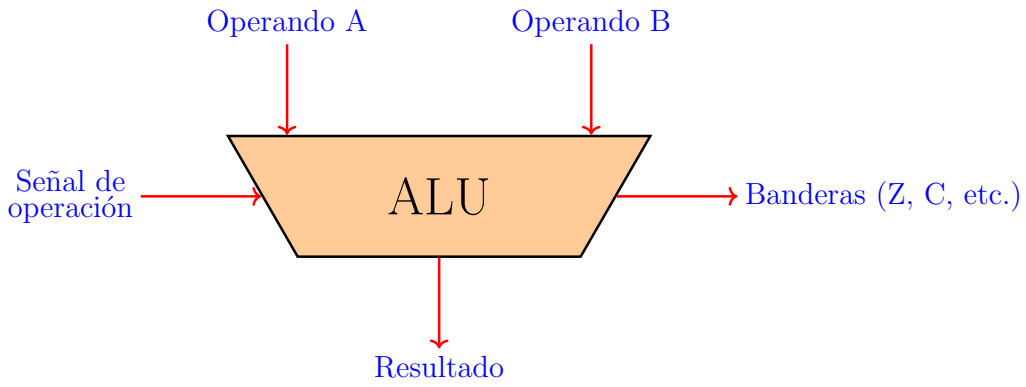


Figura 3: Esquema de la unidad aritmética lógica (ALU).

Por ejemplo, si la señal de operación corresponde a una suma, se realiza la suma de los operandos A y B, y el resultado se coloca en la salida junto con las banderas resultantes de la operación.

Ejemplo

Es interesante analizar, aunque sea a modo de ejemplo, cómo funciona la ALU, en particular en la realización de la operación de suma. Esta operación se implementa mediante compuertas lógicas, con las cuales pueden construirse sumadores que permiten sumar bit a bit los operandos. Cada sumador se encarga de sumar el n-ésimo bit de cada operando junto con el acarreo proveniente de la etapa anterior (c_{in}). Como resultado, se obtiene el bit de suma y un nuevo acarreo (c_{out}), que se utiliza como entrada en la siguiente etapa del sumador.

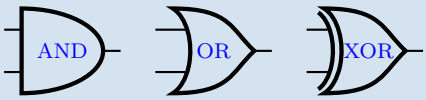
A continuación vemos la tabla de la verdad de dicha operación para el bit n-ésimo:

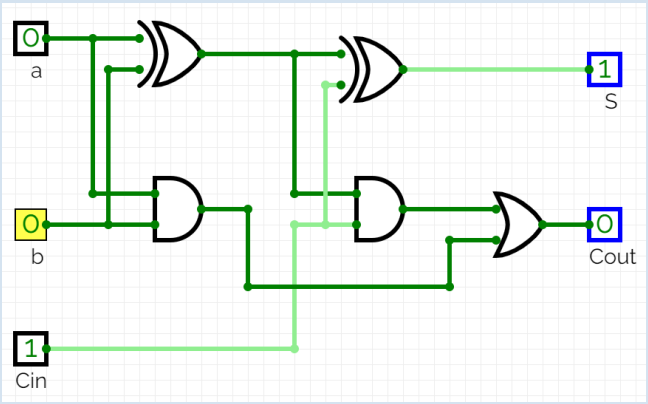
a	b	c_{in}	s	c_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

A partir de la tabla tabla anterior podemos llegar a la siguiente expresión lógica:

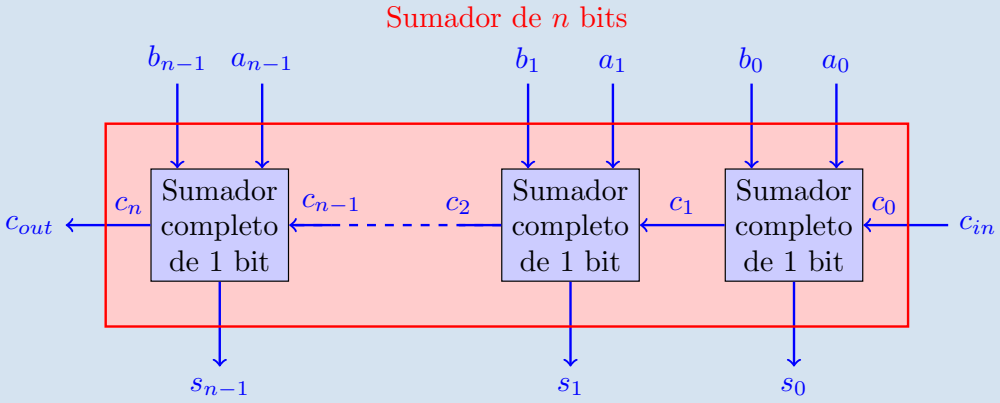
$$\begin{aligned} s &= (a \oplus b) \oplus c_{in} \\ c_{out} &= ab + c_{in}(a \oplus b) \end{aligned}$$

Podemos representar esta expresión como un circuito lógico. A continuación se presentan algunos de los símbolos de las compuertas lógicas más comunes que se utilizan en los diagramas de circuitos digitales:





Este circuito se conoce como "sumador completo" (*full-adder*). Para sumar dos números de n bits, es necesario conectar en cascada n sumadores completos, como se observa en la siguiente figura:



De manera análoga, la CPU implementa también operaciones lógicas, aritméticas, desplazamientos y rotaciones.

3.3. Registros

El intercambio de datos entre la CPU y la memoria es fundamental en una arquitectura von Neumann: las instrucciones y los operandos deben obtenerse de la memoria, y algunas instrucciones también almacenan resultados en ella. Este proceso puede generar un cuello de botella debido a la diferencia de velocidades entre la CPU y la memoria, lo que hace que la CPU pierda

tiempo mientras espera la respuesta del chip de memoria. Para mitigar este problema, el procesador cuenta con registros, celdas de memoria integradas que, aunque limitadas en cantidad, son extremadamente rápidas y mejoran significativamente el rendimiento del procesamiento.

Un registro es una memoria de alta velocidad y baja capacidad, integrada en el microprocesador, que permite almacenar temporalmente y acceder a valores frecuentemente utilizados, especialmente en operaciones matemáticas. La Figura 4 muestra un diagrama esquemático de un registro compuesto por n celdas de memoria. Cada celda consiste en un circuito electrónico biestable, conocido como *flip-flop*, que puede adoptar uno de dos estados posibles (0 o 1), lo que permite almacenar un bit de información[1].

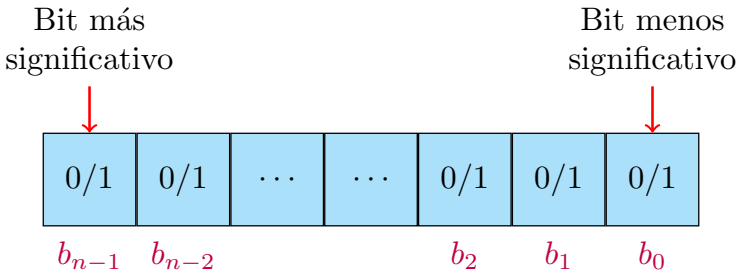


Figura 4: Esquema de un registro de n bits.

Las principales características de los registros son las siguientes:

- **Velocidad**: Son extremadamente rápidos en comparación con la memoria principal (RAM), ya que están integrados directamente en el procesador.
- **Capacidad**: Tienen una capacidad limitada, típicamente de 8, 16, 32 o 64 bits, según la arquitectura.
- **Función**: Se utilizan para almacenar datos temporales, como resultados de operaciones aritméticas, direcciones de memoria o valores intermedios durante la ejecución de un programa.

3.4. Memoria Caché

La memoria caché es un pequeño subconjunto del almacenamiento primario que se encuentra integrado en el chip de la CPU. Está diseñada para combinar la alta velocidad de acceso de los registros de la CPU, que son costosos y rápidos, con el gran tamaño de la memoria principal, que es menos costosa pero más lenta. De este modo, se establece una jerarquía en la que la memoria principal es relativamente grande y lenta, mientras que la memoria caché es más pequeña pero mucho más rápida.

La memoria caché contiene copias de porciones de la memoria principal. Cuando el procesador intenta leer una palabra de memoria, primero verifica si esa palabra está en la caché. Si es así, la palabra se entrega al procesador de inmediato. Si no está en la caché, se lee un bloque de la memoria principal, que consta de un número fijo de palabras, y se carga en la caché antes de entregar la palabra solicitada al procesador. Este proceso se beneficia del fenómeno de localidad, que sugiere que cuando se carga un bloque de datos en la caché para satisfacer

[1]Un **bit** (acrónimo de *binary digit*) es la unidad mínima de información en computación y sistemas digitales. Puede representar dos estados posibles, típicamente denotados como 0 y 1. El concepto de bit y su uso en el sistema de representación binario se abordan en detalle en el Apunte **Representación Computacional de Datos**.

una referencia, es probable que se realicen referencias futuras a esa misma ubicación o a otras palabras dentro del mismo bloque[2].

La memoria caché tiene su propia jerarquía, que se organiza en tres niveles:

- **Caché L1 (Nivel 1)**: Es la memoria caché más cercana al núcleo del procesador y, por lo tanto, la más rápida. Suele estar dividida en dos partes: una para instrucciones y otra para datos. Debido a su proximidad y alta velocidad, la caché L1 es relativamente pequeña, normalmente entre 16 KB y 64 KB por núcleo.
- **Caché L2 (Nivel 2)**: Se encuentra un nivel por debajo de la caché L1 y es más grande, generalmente entre 256 KB y 1 MB por núcleo. Aunque es más lenta que la L1, sigue siendo mucho más rápida que la memoria principal (RAM). La caché L2 puede estar compartida entre varios núcleos o ser exclusiva para cada uno, dependiendo del diseño del procesador.
- **Caché L3 (Nivel 3)**: Es el nivel más grande y lento de la jerarquía de caché. Suele estar compartida por todos los núcleos del procesador y su tamaño varía, pudiendo llegar a varios megabytes (de 4 MB a 64 MB o más). Aunque es más lenta que las cachés L1 y L2, la L3 sigue siendo más rápida que acceder a la RAM, por lo que juega un papel crucial en la mejora del rendimiento del sistema.

Cada nivel de caché actúa como un intermediario para el siguiente, con el objetivo de reducir los tiempos de acceso a los datos y mejorar la eficiencia general del procesamiento. La Figura 5 muestra esquemáticamente la jerarquía de una memoria caché típica en una CPU de dos núcleos.

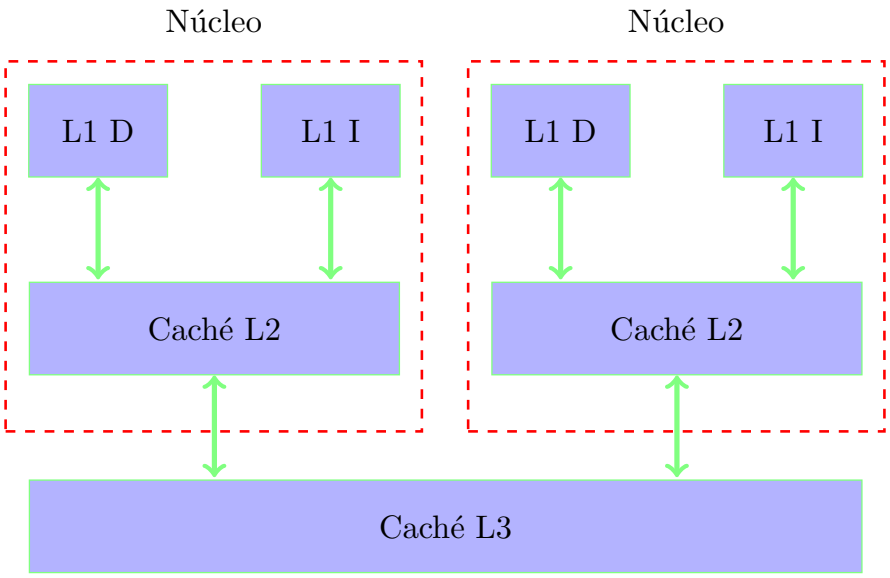


Figura 5: Esquema jerárquico de la memoria caché en una CPU de dos núcleos.

3.5. Contador de programa

El contador de programa (conocido en inglés como *Program Counter* o PC) es un registro en la CPU que contiene la dirección de memoria de la siguiente instrucción que debe ser ejecutada.

[2]Los registros de la CPU y la memoria caché se describirán con mayor detalle en la Sección 6 y otros apuntes. Sin embargo, cabe señalar que el diseño interno y la configuración de un procesador moderno son sumamente complejos, y esta asignatura proporciona un enfoque de alto nivel muy simplificado de algunas unidades funcionales clave dentro de una CPU.

Es uno de los registros más importantes en la arquitectura del computador, ya que controla la secuencia de ejecución de las instrucciones. En todo momento, el PC apunta a una instrucción en lenguaje máquina almacenada en la memoria principal. Desde que el sistema se enciende hasta que se apaga, el procesador ejecuta repetidamente la instrucción señalada por el contador de programa y actualiza el PC para que apunte a la siguiente instrucción.

El funcionamiento de un procesador se puede interpretar mediante un modelo de ejecución de instrucciones muy simple. En este modelo, las instrucciones se ejecutan en secuencia estricta, y la ejecución de una sola instrucción implica una serie de pasos: el procesador lee la instrucción de la memoria a la que apunta el contador de programa (PC), interpreta los bits de la instrucción, realiza la operación simple dictada por la instrucción y luego actualiza el PC para que apunte a la siguiente instrucción. Esta siguiente instrucción puede estar o no contigua en memoria a la instrucción que acaba de ejecutarse.

3.6. Memoria principal

La memoria principal es un dispositivo de almacenamiento temporal que contiene tanto el programa como los datos que se están manipulando mientras el procesador ejecuta el programa. Esta característica es fundamental en las computadoras basadas en el concepto de *Máquina de Von Neumann*. Físicamente, la memoria principal está constituida por una colección de chips de memoria dinámica de acceso aleatorio (DRAM, por sus siglas en inglés *Dynamic Random Access Memory*). A nivel lógico, la memoria está organizada como una matriz lineal de bytes[3], cada uno con su propia dirección única (dirección de memoria).

Los tamaños de los elementos de datos almacenados tienen correspondencia con las variables utilizadas en los lenguajes de alto nivel. Por ejemplo, en una máquina x86-64 que ejecuta Linux, los datos en lenguaje C de tipo `char` requieren 1 byte, los `short` requieren dos bytes, los `int` 4 bytes, los tipo `long` 8 bytes, los `float` 4 bytes y los `double` 8 bytes.

3.7. Memoria secundaria

El principal problema de las memorias RAM es su volatilidad. Esto significa que la memoria RAM pierde toda la información almacenada cuando la computadora se apaga o se reinicia. A diferencia de los dispositivos de almacenamiento no volátiles, como los discos duros o las SSD, que conservan los datos incluso cuando la energía se desconecta, la memoria RAM solo retiene los datos mientras está alimentada. Esta característica limita su uso a tareas que requieren acceso rápido a datos temporales durante el funcionamiento de la computadora, pero no es adecuada para el almacenamiento permanente de información.

Un disco duro es un dispositivo de almacenamiento de datos magnético que se utiliza en las computadoras para guardar permanentemente archivos y programas. Una de las principales diferencias entre el disco duro y la memoria RAM es la velocidad. La memoria RAM proporciona un acceso rápido y aleatorio a los datos y programas, lo que permite al procesador acceder a ellos de manera eficiente. En comparación, los discos duros son significativamente más lentos en términos de velocidad de acceso, ya que el brazo de lectura/escritura debe moverse físicamente para acceder a los datos almacenados en los platos giratorios.

Un SSD (Solid State Drive) es un dispositivo de almacenamiento de datos que utiliza memoria flash para guardar permanentemente archivos y programas en la computadora. A diferencia de los discos duros tradicionales, que emplean platos magnéticos giratorios y cabezales de lectura/escritura, una SSD no tiene partes móviles y utiliza chips de memoria flash para almacenar y acceder a los datos.

[3]Un byte es un conjunto ordenado de ocho bits. A su vez, un bit (acrónimo de ***bi**nary digi**t***) es la unidad mínima de información en un sistema binario. Un bit puede representar uno de dos valores: 0 o 1. Este tema se explorará con mayor detalle en el apunte **Representación Computacional de Datos**.

La tecnología SSD ofrece un rendimiento superior en comparación con los discos duros debido a la ausencia de partes mecánicas móviles. Los datos se almacenan en chips de memoria flash, que permiten un acceso y lectura mucho más rápidos. Como resultado, las SSD proporcionan un mejor rendimiento en términos de velocidad de lectura/escritura y tiempo de acceso.

3.8. Dispositivos de entrada/salida

Los dispositivos de entrada/salida (I/O *devices*) son la conexión del sistema con el mundo externo. La computadora del ejemplo en la Figura 1 tiene cuatro dispositivos de E/S: un teclado y un mouse para la entrada del usuario, una pantalla para la salida, y una unidad de disco (o disco rígido) para el almacenamiento a largo plazo de datos y programas.

Cada dispositivo de E/S está conectado al bus de E/S mediante un controlador o un adaptador. Los controladores son conjuntos de chips integrados en el propio dispositivo o en la placa de circuito impreso principal del sistema, conocida comúnmente como placa base. Un adaptador, en cambio, es una tarjeta que se conecta a una ranura en la placa base. En ambos casos, el propósito es transferir información de ida y vuelta entre el bus de E/S y un dispositivo de E/S.

3.9. Buses

A lo largo de todo el sistema existe una colección de conductores eléctricos llamados buses que transportan bytes de información de un lado a otro entre los componentes. Los buses suelen estar diseñados para transferir fragmentos de información de tamaño fijo conocidos como palabras. La Figura 6 muestra el modelo de buses de una computadora.

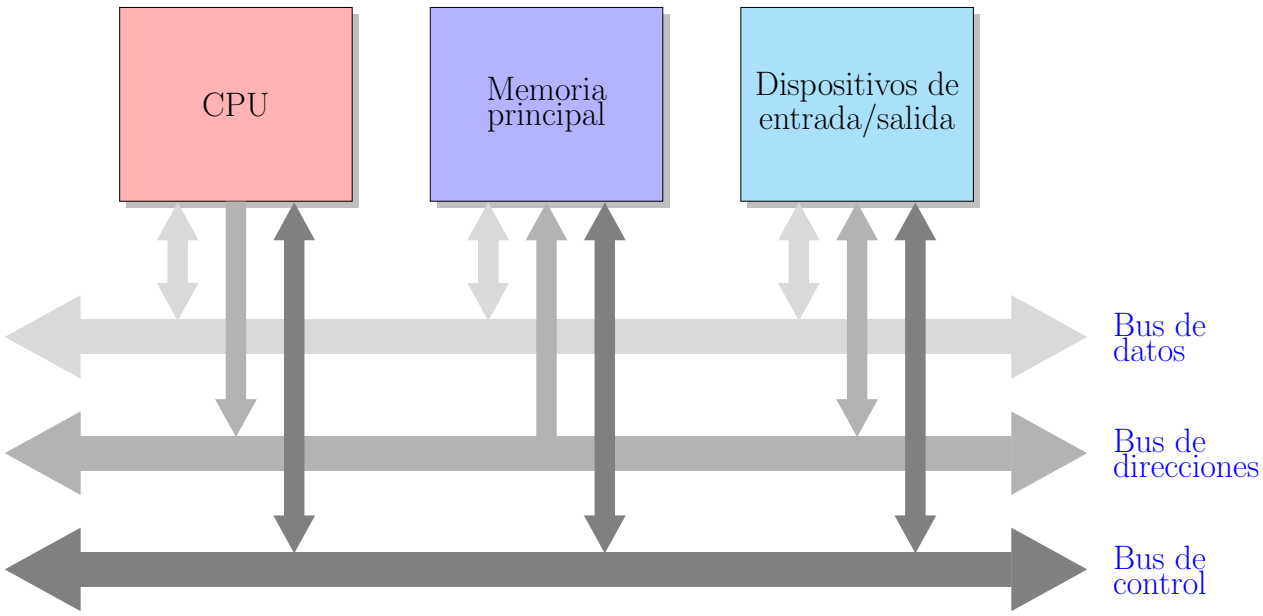


Figura 6: Modelo de buses del sistema.

- **Bus de Datos:** indica qué información se está transmitiendo.
- **Bus de Dirección:** señala hacia dónde se transfiere la información.
- **Bus de Control:** especifica cómo se transfiere la información y qué se hará con ella.
- La CPU genera direcciones que se transmiten a través del bus de direcciones.
- La memoria recibe esas direcciones mediante el mismo bus.

- La memoria nunca genera direcciones, y la CPU nunca las recibe.

El número de bytes en una palabra (es decir, el tamaño de palabra) es un parámetro fundamental del sistema que varía entre distintas arquitecturas y repercute directamente en el ancho de los buses[4]. La mayoría de las computadoras actuales utilizan tamaños de palabra de 4 bytes (32 bits) u 8 bytes (64 bits). Este tema se desarrollará con mayor detalle a lo largo de la asignatura.

4. Concepto de Arquitectura del Computador

Para comprender el funcionamiento de un sistema informático electrónico, es útil analizarlo desde distintos niveles de abstracción. Cada nivel permite enfocar la atención en ciertos aspectos específicos del sistema, ocultando temporalmente los detalles de los niveles inferiores. Desde el software que utiliza una persona usuaria hasta los circuitos electrónicos que procesan las señales, estos niveles abarcan desde el nivel de aplicación, pasando por el sistema operativo, la arquitectura del conjunto de instrucciones (ISA), la microarquitectura y, finalmente, el nivel físico. Esta organización en capas facilita el diseño, análisis y comprensión de sistemas complejos, permitiendo que especialistas en distintas áreas puedan trabajar de manera coordinada. La Figura 7 muestra los niveles de abstracción mencionados.

El nivel de la **arquitectura del computador** define el comportamiento visible del hardware desde la perspectiva del programador. En este nivel se especifican las instrucciones disponibles, los modos de direccionamiento, los registros, el formato de los datos y la forma en que se manejan las interrupciones, entre otros aspectos. En resumen, este nivel de abstracción describe cómo se presenta la computadora a quien la programa.

La arquitectura x86, utilizada por los microprocesadores en la mayoría de las computadoras personales, se define mediante un conjunto de instrucciones y registros (memoria utilizada para almacenar temporalmente variables) disponibles para el programador.

Por lo tanto, al abordar una nueva arquitectura, algunas de las preguntas clave son:

- ¿Cuál es la longitud de la palabra de datos?
- ¿Qué registros existen y cuántos son?
- ¿Cómo está organizada la memoria?
- ¿Cómo se accede a memoria?
- ¿Qué instrucciones están disponibles?

A lo largo de la asignatura, iremos profundizando en los conceptos clave de Arquitectura del Computador para comprender su funcionamiento y diseño.

5. Arquitectura de von Neumann

En 1946, el matemático y físico John von Neumann y sus colegas comenzaron el diseño de una nueva computadora, conocida como la computadora IAS (por sus siglas en inglés de *Institute for Advanced Study*), en el Instituto de Estudios Avanzados de Princeton. Aunque la computadora IAS no se completó hasta 1952, es el prototipo de todas las computadoras de propósito general posteriores. Un enfoque de diseño fundamental implementado por primera vez en la computadora IAS es conocido como el concepto de *programa almacenado*.

La Figura 8 muestra la estructura de la computadora IAS. Consiste en:

[4]El ancho de un bus se define como la cantidad de bits que puede transportar simultáneamente.

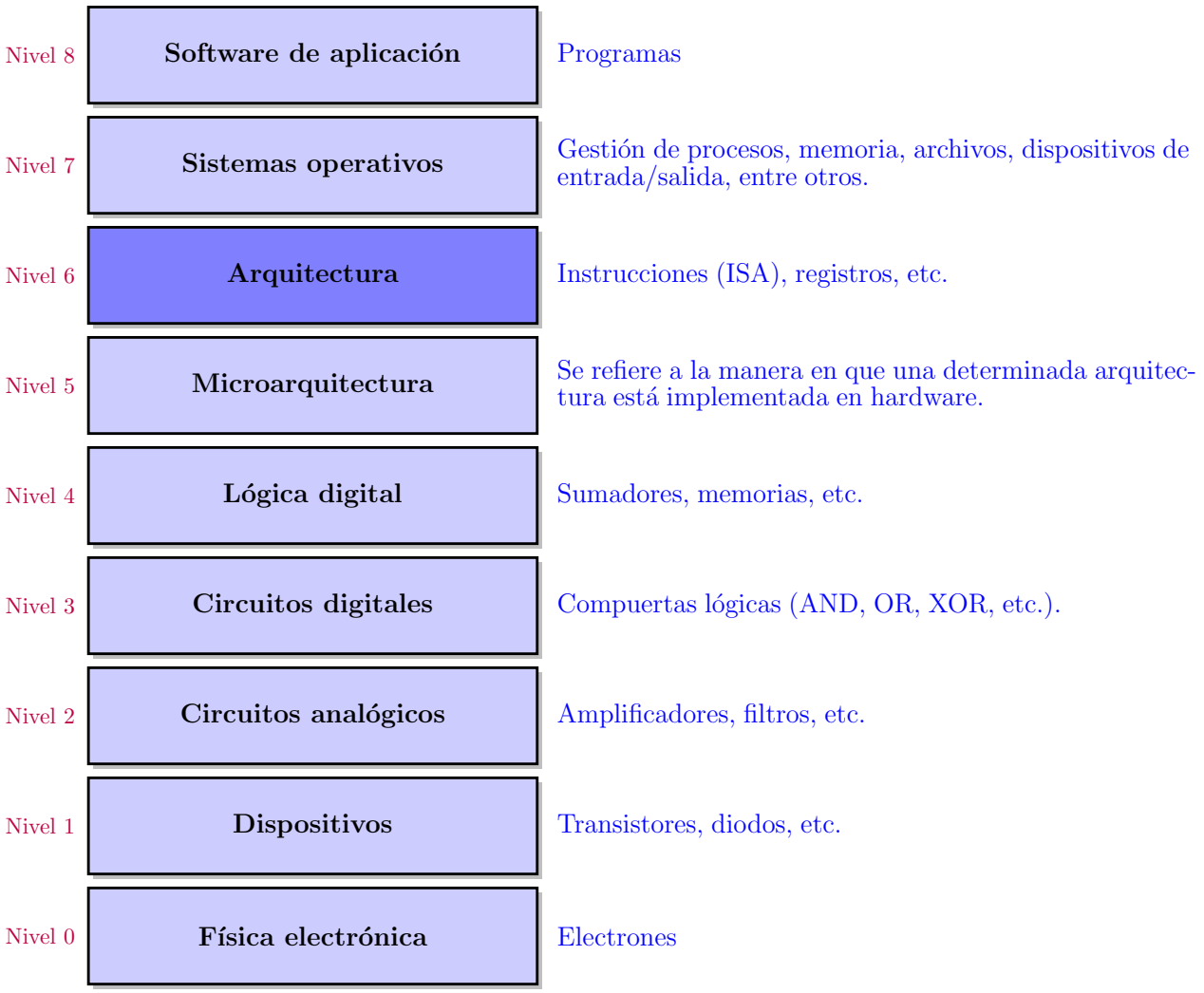


Figura 7: Niveles de abstracción de una computadora.

- Una memoria principal, que almacena tanto datos como instrucciones.
- Una unidad aritmética-lógica (ALU) capaz de operar con datos binarios.
- Una unidad de control, que interpreta las instrucciones en la memoria y las ejecuta.
- Equipos de entrada/salida (I/O) operados por la unidad de control.

Los programas y los datos se almacenan en un almacenamiento secundario (por ejemplo, un disco rígido o de estado sólido). Cuando se ejecuta un programa, tanto los programas como los datos deben copiarse desde el almacenamiento secundario hacia el almacenamiento primario o memoria principal (RAM). Luego, la CPU ejecuta el programa desde el almacenamiento primario (o memoria principal), realizando repetitivamente un ciclo de instrucciones. Con raras excepciones, todas las computadoras actuales tienen esta misma estructura y función general, por lo que se las conoce como **arquitectura de von Neumann**.

La principal ventaja de la arquitectura de Von Neumann es su simplicidad y flexibilidad, derivadas del uso de una única memoria para almacenar tanto instrucciones como datos. Esto permite que el programa pueda modificarse a sí mismo en tiempo de ejecución —lo que habilita técnicas como los bucles, los saltos condicionales y una programación más dinámica— y que el diseño del hardware resulte más sencillo, ya que solo se requiere un único bus de memoria y un mecanismo unificado para acceder a instrucciones y datos. Sin embargo, la demanda generada por el acceso simultáneo a instrucciones y datos desde la misma memoria pasó a conocerse

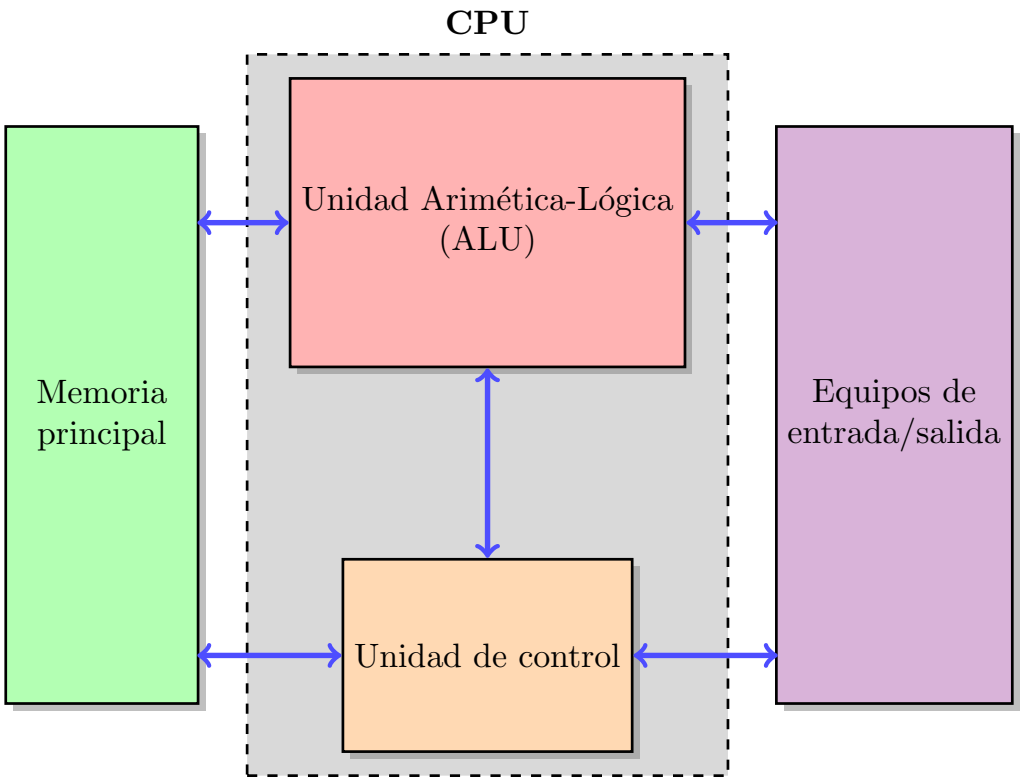


Figura 8: Estructura de la computadora con arquitectura *Von Neumann*.

más adelante como el *"cuello de botella"* de Von Neumann. Este inconveniente se analizará en detalle en la Sección 6.

6. Jerarquía de memoria

> *"Idealmente, se desearía una capacidad de memoria indefinidamente grande de manera que cualquier palabra en particular estuviera disponible de inmediato. Sin embargo, nos vemos obligados a reconocer la posibilidad de construir una jerarquía de memorias, cada una de las cuales tiene una capacidad mayor que la anterior, pero que es menos rápidamente accesible."*
>
> **A. W. Burks, H. H. Goldstine, y J. von Neumann**
> Discusión preliminar sobre el diseño lógico de un instrumento de cálculo electrónico, 1946.

Los dispositivos de almacenamiento en un sistema informático se organizan en una jerarquía de memoria, como se muestra en la Figura 9. El objetivo es reducir la brecha de velocidad entre el procesador y las memorias más lentas. Para ello, se procura maximizar el rendimiento del sistema manteniendo los datos más utilizados lo más cerca posible del procesador.

A medida que se desciende en la jerarquía, los dispositivos se vuelven más lentos, tienen mayor capacidad y resultan menos costosos por byte. En cambio, en los niveles superiores se encuentran dispositivos más rápidos y con mayor costo.

En la cúspide de la jerarquía se ubican los registros, seguidos por la memoria caché, que suele subdividirse en tres niveles: L1, L2 y L3. A continuación, se encuentra la memoria principal y, en los niveles inferiores, los dispositivos de almacenamiento masivo, como los discos rígidos y las cintas magnéticas.

El principio fundamental de esta jerarquía es que cada nivel actúa como un *buffer* del nivel inferior. De este modo, los registros funcionan como *buffer* de la caché L1; la memoria caché,

de la memoria principal; y esta, a su vez, del disco rígido.

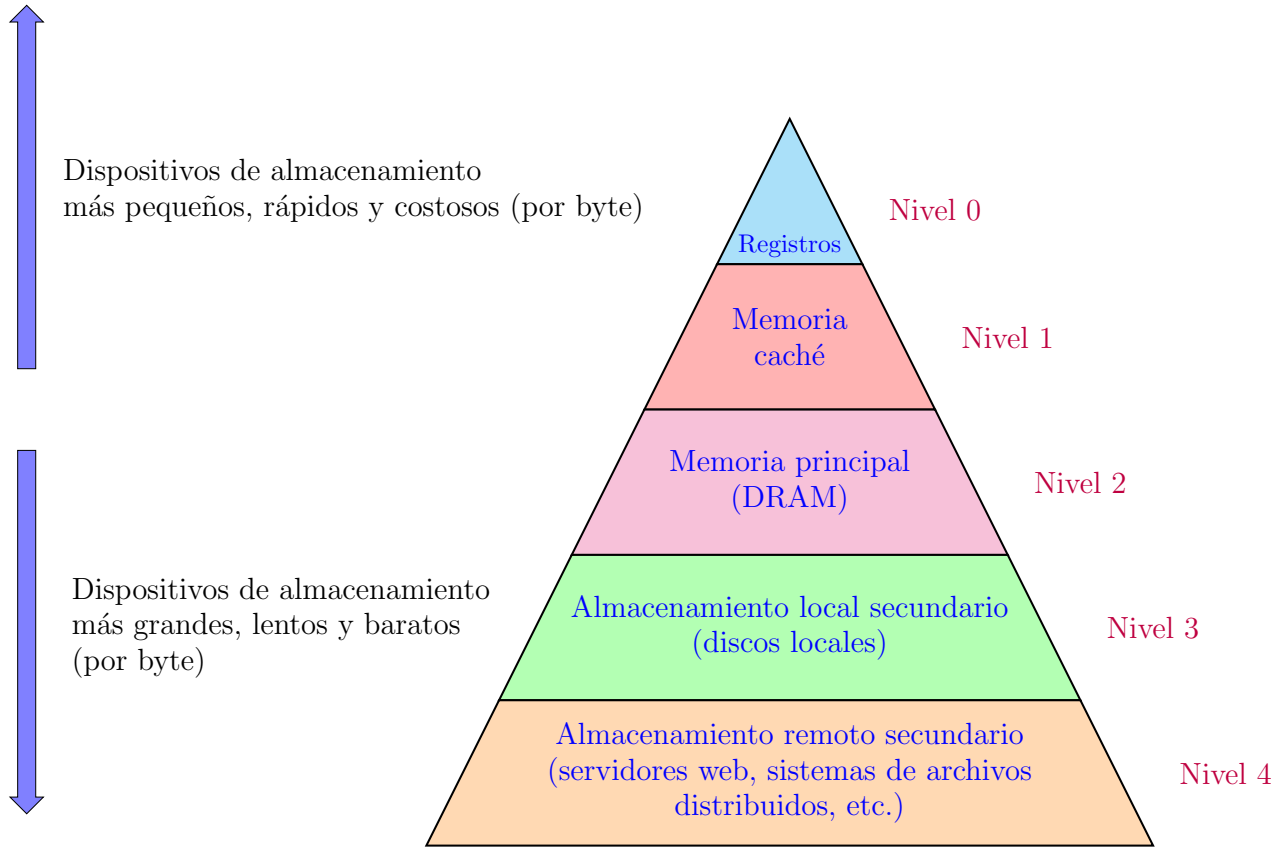


Figura 9: Diagrama piramidal de la jerarquía de memoria.

7. Clasificación de arquitecturas

Las arquitecturas **CISC** (Complex Instruction Set Computer) y **RISC** (Reduced Instruction Set Computer) representan dos enfoques distintos en el diseño de procesadores. La principal diferencia radica en la complejidad y la cantidad de instrucciones que cada una ofrece. El término conjunto de instrucciones de la arquitectura (ISA, por sus siglas en inglés Instruction Set Architecture) hace referencia al conjunto real de instrucciones disponibles para controlar la CPU.

El ISA actúa como una interfaz entre el hardware y el software, al especificar qué operaciones puede realizar el procesador y cómo se ejecutan. Define los tipos de datos admitidos, los registros disponibles, la forma en que el hardware gestiona la memoria principal, características clave como la memoria virtual, el conjunto de instrucciones ejecutables y el modelo de entrada/salida en distintas implementaciones. Además, el ISA puede ampliarse mediante la incorporación de nuevas instrucciones, funcionalidades adicionales o el soporte para direcciones y tamaños de datos más grandes.

Los procesadores CISC disponen de un conjunto amplio y variado de instrucciones, muchas de las cuales pueden ejecutar tareas complejas en una sola operación. Esto permite que los programas en lenguaje ensamblador sean más compactos, pero a costa de una mayor complejidad en el hardware y tiempos de ejecución variables.

En contraste, las arquitecturas RISC se basan en un conjunto reducido y uniforme de instrucciones, generalmente de ejecución rápida y duración fija. Esto simplifica el diseño del procesador y facilita la implementación de técnicas como la segmentación (pipelining), lo que mejora

el rendimiento y la eficiencia energética. Además, en RISC, el acceso a memoria está limitado a instrucciones específicas de carga y almacenamiento, lo que favorece una ejecución más predecible.

Estas diferencias hacen que CISC sea más común en arquitecturas tradicionales como x86, mientras que RISC ha ganado protagonismo en dispositivos móviles y sistemas embebidos, como los basados en ARM, debido a su bajo consumo y alto rendimiento.

La Tabla 1 presenta una comparación entre las arquitecturas CISC (Complex Instruction Set Computer) y RISC (Reduced Instruction Set Computer). Se destacan sus principales diferencias en cuanto al diseño del conjunto de instrucciones, el acceso a memoria, la complejidad del hardware y del compilador, así como su eficiencia y ámbitos de aplicación típicos.

Tabla 1: Comparación entre arquitecturas CISC y RISC

Característica	**CISC (Complex Instruction Set Computer)**	**RISC (Reduced Instruction Set Computer)**
Conjunto de instrucciones	Grande y complejo	Reducido y simple
Duración de instrucciones	Variable (dependiendo de la instrucción)	Fija (una instrucción por ciclo, típicamente)
Acceso a memoria	Las instrucciones en general pueden acceder a memoria directamente	Solo las instrucciones específicas de carga/almacenamiento acceden a memoria
Formato de instrucciones	Varios formatos y longitudes	Formato uniforme (tamaño fijo)
Uso de microcódigo	Frecuente (traducción interna a microinstrucciones)	Poco frecuente o inexistente
Tamaño del compilador	Más simple (mayor trabajo en hardware)	Más complejo (mayor trabajo en software)
Tamaño del hardware	Más grande y complejo	Más pequeño y eficiente
Tiempos de ejecución	Puede ser más lento por la complejidad de las instrucciones	Más rápido por simplicidad y paralelismo
Ejemplo típico de procesador	Intel x86	ARM, MIPS, RISC-V
Eficiencia energética	Menor (más consumo por instrucción compleja)	Mayor (mejor para dispositivos móviles)

En esta asignatura nos vamos a focalizar en primer lugar en la arquitectura x86-64. El x86-64 es un diseño de CPU de tipo CISC. Esto se refiere a la filosofía de diseño del procesador interno. Los procesadores CISC generalmente incluyen una amplia variedad de instrucciones (a veces superpuestas), diferentes tamaños de instrucciones y una amplia gama de modos de direccionamiento. El término se acuñó retroactivamente en contraste con las arquitruras tipo RISC. Una de las arquitecturas tipo RISC es la denominada ARM, la cual estudiaremos al final del curso en el apunte *Conceptos básicos del ensamblador y arquitectura ARM*.

8. Microarquitectura

La **microarquitectura** es el diseño interno de un procesador que determina cómo se implementan físicamente las instrucciones definidas por la arquitectura del conjunto de instrucciones (ISA). Incluye componentes como unidades funcionales, registros, buses, cachés, decodificadores de instrucciones, unidades de ejecución y mecanismos de control de flujo y paralelismo. En particular, la arquitectura del computador especifica qué instrucciones existen y cómo deben comportarse, mientras que la microarquitectura determina cómo se ejecutan esas instrucciones en una implementación concreta. La Figura 7 muestra la ubicación de la microarquitectura dentro de los distintos niveles de abstracción de una computadora.

Una arquitectura particular, como la x86-64, puede tener muchas microarquitecturas diferentes (como Intel Haswell vs. Intel Skylake), cada una con diferentes rendimiento, costo y complejidad. Todos las microarquitecturas de una determinada arquitectura ejecutan los mismos programas, pero sus diseños internos pueden variar ampliamente. En esta asignatura nos centraremos en las arquitecturas y no abordaremos las microarquitecturas.

9. Breve historia de la computación

Se han construido cientos de tipos diferentes de computadoras desde sus orígenes. Algunas han tenido un gran impacto y otras una menor incidencia. El objetivo de esta sección es dar un pantallazo muy breve sobre los orígenes de la computación, marcando algunos hitos relevantes.

9.1. Computadoras mecánicas

- Blaise Pascal (1623–1662). En 1642 construyó un dispositivo completamente mecánico, basado en engranajes, que solo podía realizar sumas y restas.
- Gottfried von Leibniz (1646–1716). Construyó otra máquina completamente mecánica que, además de sumar y restar, podía multiplicar y dividir.
- Charles Babbage (1792–1871). Construyó una máquina diseñada para ejecutar un único algoritmo con el fin de calcular tablas numéricas útiles para la navegación. Los resultados eran perforados en una placa de cobre.
- Posteriormente, desarrolló la máquina analítica, capaz de ejecutar distintos algoritmos. Contaba con cuatro componentes: el almacén (memoria), el molino (unidad de cómputo), la sección de entrada (lector de tarjetas perforadas, ver Figura 11) y la sección de salida (resultados impresos o perforados). Podía sumar, restar, multiplicar y dividir operandos.
- Ada Lovelace (1815–1852). Es considerada la primera programadora de computadoras del mundo. Colaboró con Charles Babbage en la descripción de la máquina analítica y escribió el primer algoritmo destinado a ser ejecutado por una máquina, anticipando conceptos fundamentales de la programación moderna, como los bucles y las subrutinas. Su visión iba más allá del simple cálculo numérico: comprendió que la máquina podía manipular símbolos y realizar tareas generales siguiendo instrucciones, lo que la llevó a imaginar aplicaciones más amplias, como la generación automática de música. A pesar de la originalidad y profundidad de sus ideas, su trabajo no fue valorado en su tiempo y solo comenzó a ser reconocido muchas décadas después.
- Konrad Zuse (1910–1995). Construyó una serie de máquinas calculadoras automáticas utilizando contactores electromagnéticos[5].

[5]También denominados relés o relevadores. Un relé es un dispositivo electromagnético que funciona como

- John Atanasoff (1903–1995). Diseñó una máquina que utilizaba aritmética binaria y una memoria basada en condensadores, empleando un proceso que denominó “refrescar la memoria”. Aunque el concepto era correcto, la máquina nunca llegó a funcionar por problemas de implementación.
- George Stibitz (1904–1995). Construyó una máquina más primitiva que la de Atanasoff, aunque sí llegó a funcionar.
- Howard Aiken (1900–1973). Construyó, utilizando relés, la máquina de propósito general que Babbage no había logrado construir con ruedas dentadas. Su primera máquina (Mark I) se completó en Harvard en 1944 (ver Figura 10). Contaba con 72 palabras de 23 dígitos decimales cada una y un tiempo de instrucción de 6 segundos. Las entradas y salidas se realizaban mediante cintas de papel perforadas (ver Figura 11). Para cuando se desarrolló la Mark II, las máquinas basadas en relés ya eran obsoletas.

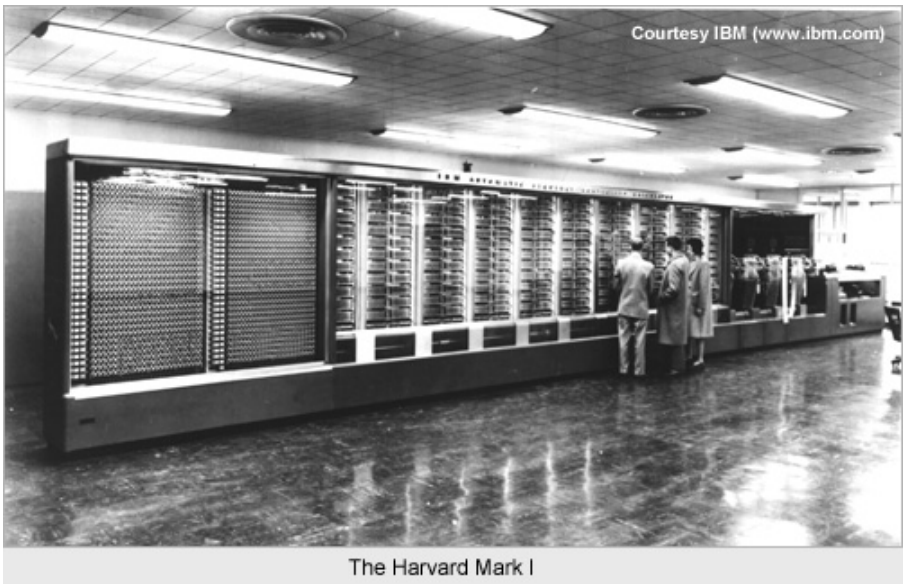


Figura 10: Imagen de la computadora Mark I (fuente:[1]).

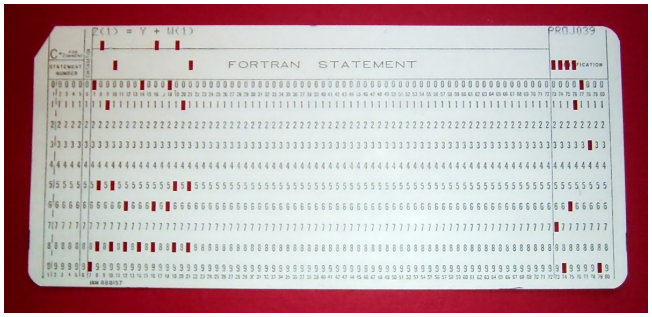


Figura 11: Imagen de una tarjeta perforada (Estándar de 80 columnas).

9.2. Computadoras con válvulas de vacío

La válvula electrónica —también conocida como válvula de vacío, tubo de vacío o bulbo— es un componente utilizado para amplificar, conmutar o modificar señales eléctricas mediante el

un interruptor controlado por un circuito eléctrico, en el que, por medio de una bobina y un electroimán, se acciona un juego de uno o varios contactos que permiten abrir o cerrar otros circuitos eléctricos independientes.

control del movimiento de electrones en un espacio "vacío" a muy baja presión, o en presencia de gases especialmente seleccionados. Fue el componente clave que posibilitó el desarrollo de la electrónica durante la primera mitad del siglo XX, impulsando el crecimiento y la comercialización de tecnologías como la radiodifusión, la televisión, el radar, el audio, las redes telefónicas, las computadoras analógicas y digitales, y los sistemas de control industrial. Si bien algunas de estas aplicaciones ya existían antes de la invención de la válvula, su desarrollo se aceleró notablemente gracias a ella [2].

A continuación se enumeran algunos de los avances más notables de la computación basados en el uso de válvulas electrónicas:

- Alan Turing (1912–1954). Contribuyó al desarrollo de COLOSSUS, la primera computadora electrónica, construida por el gobierno británico para descifrar códigos durante la Segunda Guerra Mundial.
- John Mauchly (1907–1980). Participó en la construcción de la ENIAC (Electronic Numerical Integrator And Computer) en 1946, considerada la primera computadora digital, electrónica, de propósito general y programable (ver Figura 12). Contaba con 18 000 válvulas y 1 500 relés, pesaba 30 toneladas y consumía 140 kW. Operaba en sistema decimal. En la Figura 13 se muestra una válvula de vacío.
- EDSAC (Electronic Delay Storage Automatic Calculator). Máquina sucesora de la ENIAC, construida en 1949. A diferencia de esta, operaba en sistema binario y fue una de las primeras computadoras en ejecutar programas almacenados en memoria.
- EDVAC (Electronic Discrete Variable Automatic Computer). Una de las primeras computadoras electrónicas, desarrollada en 1951. A diferencia de la ENIAC, usaba aritmética binaria y fue diseñada siguiendo el concepto de programa almacenado, inspirado en la Máquina de Turing.
- John von Neumann (1903–1957). Desarrolló en 1952 la máquina IAS, en la que se implementó el diseño arquitectónico que hoy se conoce como **máquina de Von Neumann**, compuesto por cinco partes básicas: memoria, unidad aritmético-lógica, unidad de control, entrada y salida. Esta arquitectura continúa siendo la base de casi todas las computadoras digitales actuales.
- Whirlwind I (1951). Diseñada en el MIT, fue una de las primeras computadoras en operar en tiempo real. Utilizaba palabras de 16 bits y representó un avance significativo en aplicaciones de control.

Observación

Las computadoras ENIAC, COLOSSUS, MARK I y MARK II son ejemplos de máquinas con *programa cableado*. Estas no contaban con memoria para almacenar instrucciones; por lo tanto, cada nuevo programa requería una reconfiguración manual del sistema mediante cables y conmutadores. Este proceso era lento y complejo, y limitaba considerablemente la flexibilidad de uso. La posterior adopción del concepto de *programa almacenado*, implementado en máquinas como la EDVAC y la EDSAC, representó un avance fundamental en la evolución de la computación, al permitir modificar y ejecutar programas directamente desde la memoria sin necesidad de realizar cambios físicos en el hardware.

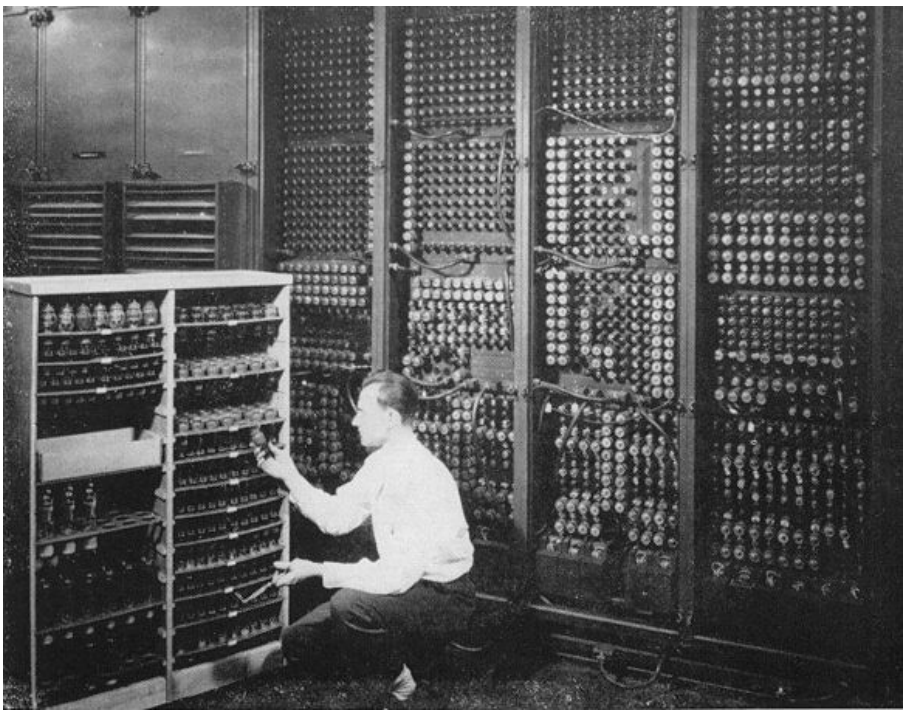


Figura 12: Imagen de la computadora ENIAC (fuente: [3]).

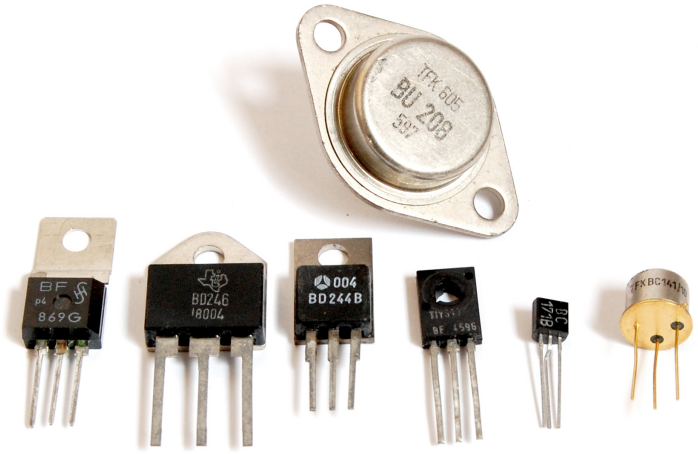
Figura 13: Imagen de una válvula electrónica (fuente: [2]).

9.3. Computadoras con transistores

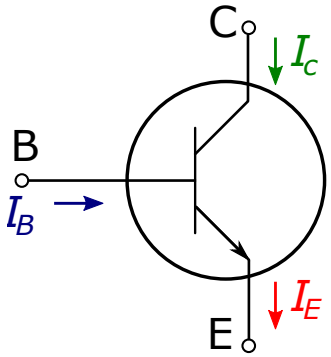
En 1948, John Bardeen, Walter Brattain y William Shockley inventaron el transistor mientras trabajaban en los Laboratorios Bell. El transistor es un dispositivo electrónico semiconductor que entrega una señal de salida en respuesta a una señal de entrada. Puede funcionar como amplificador, oscilador, interruptor o rectificador. Actualmente, está presente en prácticamente todos los aparatos electrónicos de uso diario, tales como radios, televisores, reproductores de audio y video, relojes de cuarzo, computadoras, lámparas fluorescentes, tomógrafos y teléfonos celulares, aunque casi siempre dentro de los llamados circuitos integrados [4].

Los transistores pueden usarse como interruptores, al igual que las válvulas, pero con un tamaño mucho menor, menor costo y menor disipación de calor. En la Figura 14(a) se muestran diferentes tipos de transistores, mientras que en la Figura 14(b) se ilustra el esquema de un transistor tipo NPN, donde se observan sus tres partes fundamentales: la *base*, el *colector* y el *emisor*.

El impacto del transistor ha sido enorme, ya que, además de dar origen a la industria de los



(a) Imagen de algunos tipos de transistores.



(b) Diagrama de Transistor tipo NPN.

Figura 14: Transistores (fuente: [4]).

semiconductores, fue el precursor de otros inventos como los circuitos integrados, los dispositivos optoelectrónicos y los microprocesadores. Sin embargo, los cambios más significativos se produjeron en el campo de la computación. Según Albert P. Malvino [5], *"el transistor no se creó para mejorar la computación, sino que fue él quien la creó"*.

Antes de 1950, una computadora ocupaba una sala entera y costaba millones de dólares. Con la incorporación del transistor, fue posible reducir notablemente tanto el tamaño como el costo de estas máquinas. Hoy en día, una computadora cabe en un bolsillo, tiene un costo muy bajo y ofrece enormes prestaciones.

A continuación, se enumeran algunos de los avances más notables en computación que utilizaron esta tecnología:

- TX-0 (1956). Primera computadora transistorizada, desarrollada en el Lincoln Laboratory del MIT.
- IBM 1401 (1959). Computadora desarrollada por IBM, orientada a aplicaciones comerciales y contables (ver Figura 15).
- PDP-1 (1960). Primera microcomputadora; contaba con 4K palabras de 18 bits y un tiempo de ciclo de 5 μs.
- IBM 7094 (1962). Computadora desarrollada por IBM, orientada a la computación científica.
- CDC 6600 (1964). Desarrollada por Control Data Corporation (CDC), considerada la primera supercomputadora científica.

9.4. Computadoras con circuitos integrados

En 1958, Robert Noyce inventó el circuito integrado de silicio, lo que permitió colocar decenas de transistores en un solo chip. Un circuito integrado, también conocido como chip o

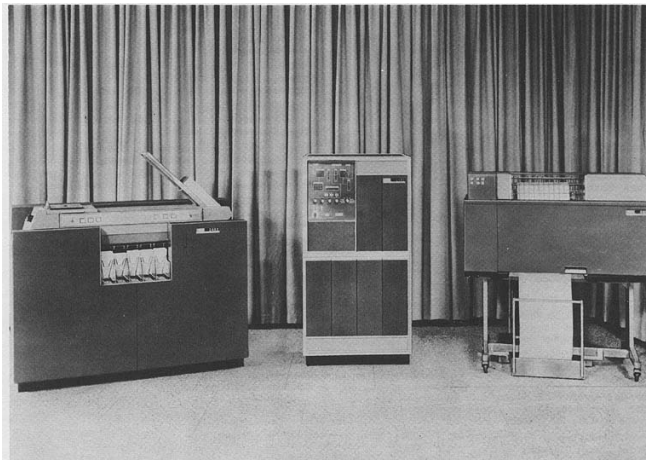


Figura 15: Imagen de la computadora IBM 1401 en el centro; a la izquierda, se observa la lectora/perforadora de tarjetas IBM 1402: a la derecha, la impresora 1403 (fuente: [6]).

microchip, es una estructura de pequeñas dimensiones hecha de material semiconductor —normalmente silicio— con una superficie de pocos milímetros cuadrados, sobre la cual se fabrican circuitos electrónicos generalmente mediante fotolitografía. Este está protegido dentro de un encapsulado de plástico o cerámica, que cuenta con conductores metálicos adecuados para conectar el circuito integrado a un circuito impreso [7].

La integración de grandes cantidades de transistores diminutos en un espacio reducido representó un avance significativo respecto a la elaboración manual de circuitos con componentes discretos. La producción masiva de circuitos integrados, junto con su fiabilidad y la posibilidad de diseñar circuitos cercanos a un diagrama a bloques, facilitó la rápida adopción de circuitos integrados estandarizados en lugar de los diseños basados en transistores discretos.

A continuación, se enumeran algunos de los avances más notables en computación que utilizaron esta tecnología:

- System/360 (1964). Familia de computadoras desarrollada por IBM, destinada tanto a aplicaciones científicas como comerciales. Una innovación destacada fue la multiprogramación (ver Figura 16).
- PDP-8 (1965). Primera minicomputadora con éxito comercial masivo, con unas 50 000 unidades vendidas.
- PDP-11 (1970). Dominó el mercado de las minicomputadoras durante la década de los setenta.

9.5. Computadoras con integración a muy gran escala

La integración a escala muy grande, o VLSI (siglas en inglés de *very-large-scale integration*), es el proceso de fabricar circuitos integrados que contienen cientos de miles de transistores en un único chip. Esta tecnología comenzó a utilizarse en la década de 1970, como parte de los avances en semiconductores y comunicaciones que se estaban desarrollando [9].

- Intel 8080 (1974). Primer microprocesador de propósito general de 8 bits en un solo chip. Funcionaba a 2 MHz y es considerado el primer diseño de microprocesador verdaderamente usable. En la Figura 17 se muestra una imagen del microprocesador Intel 8080.
- Apple II (1977). Primera serie de microcomputadoras de producción masiva desarrollada por Apple Computer (ver Figura 18). Contaba con una arquitectura de 8 bits.

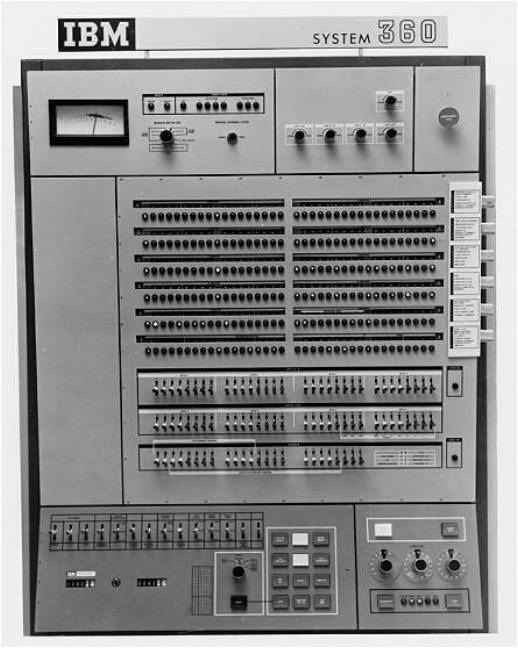


Figura 16: Imagen de la computadora IBM 360 (fuente: [8]).

Figura 17: Microprocesador Intel 8080 (fuente: [10]).

Figura 18: Computadora Apple II (fuente: [11]).

- 8086 (1978). Uno de los primeros microprocesadores de 16 bits en un solo chip, con 29 000 transistores. Su variante, el 8088, con un bus externo de 8 bits, fue utilizado en la computadora personal original de IBM.
- 8087 (1980). Intel introdujo el coprocesador de punto flotante 8087 (45 000 transistores), diseñado para trabajar junto al procesador 8086 o 8088 y ejecutar instrucciones de pun-

to flotante. El 8087 estableció el modelo de punto flotante para la línea x86, conocido comúnmente como "x87".

- IBM Personal Computer (1981). Se convirtió en la computadora más vendida de la historia. Venía equipada con el sistema operativo MS-DOS, provisto por Microsoft Corporation.
- 80286 (1982). Añadió más modos de direccionamiento de memoria y formó la base para la IBM PC-AT, la plataforma original para MS Windows. Contaba con registros de 16 bits y 134 000 transistores.
- RISC-I (1982). Desarrollada en el proyecto RISC de la Universidad de Berkeley, bajo la dirección de David A. Patterson. RISC (Reduced Instruction Set Computer) propone reemplazar arquitecturas complejas (CISC) por otras más sencillas y rápidas.
- R2000 (1985). Primer diseño MIPS creado por John L. Hennessy, que mejoró drásticamente el rendimiento mediante el uso de segmentación.
- i386 (1985). Expandió la arquitectura x86 a 32 bits. Contaba con 275 000 transistores.
- i486 (1989). Mejoró el desempeño de la línea x86 e integró la unidad de punto flotante en el chip, aunque sin cambios significativos en el conjunto de instrucciones. Tenía 1.2 millones de transistores.
- Pentium (1993). Mejoró el desempeño y añadió extensiones menores al conjunto de instrucciones. Contaba con 3.1 millones de transistores.
- Pentium Pro (1995). Introdujo una microarquitectura radicalmente nueva, conocida internamente como P6. Añadió instrucciones de movimiento condicional al conjunto. Poseía 5.5 millones de transistores.
- Pentium II (1997). Continuó con la microarquitectura P6. Contaba con 7 millones de transistores.
- Pentium III (1999). Introdujo SSE, un conjunto de instrucciones para manipular vectores de datos enteros o de punto flotante. Tenía 8.2 millones de transistores, con versiones posteriores que llegaron a 24 millones.
- Pentium 4 (2000). Extendió SSE a SSE2, incorporando nuevos tipos de datos, incluyendo punto flotante de doble precisión, junto con numerosas instrucciones adicionales. Contaba con 42 millones de transistores.
- Pentium 4E (2004). Añadió *hyperthreading*, que permite ejecutar dos programas simultáneamente en un solo procesador, así como EM64T, la implementación de Intel de una extensión de 64 bits para IA-32 desarrollada por AMD, conocida como x86-64. Tenía 125 millones de transistores.
- Core 2 (2006). Retornó a una microarquitectura similar a P6. Fue el primer microprocesador multinúcleo de Intel, integrando múltiples núcleos en un solo chip. No es compatible con *hyperthreading*. Poseía 291 millones de transistores.
- Core i7 (2008). Incorporó tanto *hyperthreading* como *multi-core*, con la versión inicial que permitía ejecutar dos hilos por núcleo y hasta cuatro núcleos por chip. Contaba con 781 millones de transistores.

9.6. Primeros lenguajes de programación

A continuación, se presentan los primeros lenguajes de programación junto con una descripción de sus características fundamentales:

Lenguaje	Año	Principales características
Assembly	1940s	Lenguaje de bajo nivel, cercano al código máquina; específico para cada arquitectura; manipulación directa de registros y memoria.
FORTRAN	1957	Primer lenguaje de alto nivel ampliamente usado; orientado a cálculos científicos y matemáticos; facilitaba la programación estructurada y la portabilidad.
LISP	1958	Primer lenguaje para inteligencia artificial; basado en listas; soporta recursión y manejo simbólico.
COBOL	1959	Orientado a aplicaciones de negocio; diseñado para ser legible y cercano al lenguaje inglés; usado en sistemas administrativos y financieros.
ALGOL	1958-1960	Introdujo conceptos de lenguajes estructurados y de bloques; influyó en muchos lenguajes posteriores como Pascal, C y Java.
BASIC	1964	Diseñado para principiantes; fácil de aprender y usar; orientado a educación y programación interactiva.
PL/I	1964	Intentó unificar características de FORTRAN y COBOL; diseñado para aplicaciones científicas y comerciales; lenguaje complejo.
Simula	1967	Primer lenguaje orientado a objetos; basado en ALGOL; usado para simulaciones y modelado.
Pascal	1970	Diseñado con fines educativos; promueve la programación estructurada y el uso riguroso de tipos de datos. Fue ampliamente utilizado en enseñanza y desarrollo de software en los años 80.
C	1972	Diseñado originalmente para escribir sistemas operativos como UNIX, es un lenguaje ampliamente utilizado que aún hoy se emplea de forma masiva. Ha influido en el diseño de muchos lenguajes modernos, como C++, Java y Python.

Referencias

[1] Historia de la Informática. Harvard MARK I (1944), 2015. [Internet; descargado 2-agosto-2023].

[2] Colaboradores de Wikipedia. Válvula termoiónica — Wikipedia, La enciclopedia libre. `https://es.wikipedia.org/w/index.php?title=V%C3%A1lvula_termoi%C3%B3nica&oldid=121814534`, 2019. [Internet; descargado 27-enero-2020].

[3] Historia de la Informática. Proyecto ENIAC, 2011. [Internet; descargado 31-enero-2022].

[4] Colaboradores de Wikipedia. Transistor — Wikipedia, La enciclopedia libre. `https://es.wikipedia.org/w/index.php?title=Transistor&oldid=122923620`, 2020. [Internet; descargado 27-enero-2020].

[5] Albert Paul Malvino and David J. Batesr. *Principios de Electrónica*. Mcgraw-Hill, 2007.

[6] Colaboradores de Wikipedia. IBM 1401 — Wikipedia, La enciclopedia libre, 2021. [Internet; descargado 31-enero-2022].

[7] Colaboradores de Wikipedia. Circuito integrado — Wikipedia, La enciclopedia libre. https://es.wikipedia.org/w/index.php?title=Circuito_integrado&oldid=122233649, 2019. [Internet; descargado 27-enero-2020].

[8] Colaboradores de Wikipedia. IBM System/360, 2015. [Internet; descargado 2-agosto-2021].

[9] Colaboradores de Wikipedia. Integración a muy gran escala — Wikipedia, La enciclopedia libre. https://es.wikipedia.org/w/index.php?title=Integraci%C3%B3n_a_muy_gran_escala&oldid=117810308, 2019. [Internet; descargado 27-enero-2020].

[10] Elprocus. Introduction to 8080 Microprocessor and its Architecture, 2015. [Internet; descargado 2-agosto-2023].

[11] Colaboradores de Wikipedia. Apple II, 2023. [Internet; descargado 2-agosto-2023].

[12] Ed Jorgenson. *X86-64 Assembly Language Programming with Ubuntu*. 2019.

[13] John L Hennessy and David A Patterson. *Computer architecture: A quantitative approach*. Elsevier, 2011.

[14] Randal E Bryant and David Richard O'Hallaron. *Computer systems: a programmer's perspective*, volume 281. Prentice Hall Upper Saddle River, second edition, 2003.

[15] Richard Blum. *Professional assembly language*. John Wiley & Sons, 2007.

[16] Andrew S Tanenbaum. *Organización de computadoras: un enfoque estructurado*. Pearson educación, 2000.

[17] Randall Hyde. *The art of assembly language*. No Starch Press, 2003.

[18] Igor Zhirkov. *Low-Level Programming: C, Assembly, and Program Execution on Intel 64 Architecture*. Apress, 2017.

[19] Sarah L Harris and David Harris. *Digital design and computer architecture*. Morgan Kaufmann, 2015.

[20] William Stallings. *Computer organization and architecture: designing for performance*. Pearson Education India, 11th edition, 2022.