OPOSICIONES TAI · ADMINISTRACIÓN GENERAL DEL ESTADO

Bloque II · Tema II.01

Informática básica: representación de la información y arquitectura de ordenadores

GUÍA INTENSIVA DE ESTUDIO · ORIENTADA A EXAMEN

Contenido actualizado a agosto 2026 · Preguntas 2016–2026

1. Conceptos básicos: el ordenador y la información

Un **ordenador** es una máquina electrónica capaz de recibir datos, procesarlos siguiendo una serie de instrucciones (un **programa**) y devolver un resultado. Todo lo que maneja un ordenador —números, texto, imágenes o sonido— se transforma internamente en **información digital**, es decir, en combinaciones de dos únicos valores: **0 y 1**. Entender cómo se representa esa información y cómo está organizado el ordenador que la procesa es la base de todo el temario técnico.

Un **sistema de información** es el conjunto organizado de elementos que recogen, procesan, almacenan y distribuyen información para apoyar las operaciones, la toma de decisiones y el control en una organización. Sus **elementos constitutivos** son cinco: el **hardware** (los equipos físicos), el **software** (los programas), los **datos** (la materia prima que, una vez procesada, se convierte en información), las **personas** (usuarios y personal técnico) y los **procedimientos** (las normas y métodos de trabajo que regulan su funcionamiento). Sus **funciones básicas** son cuatro: **entrada** (captura de los datos), **procesamiento** (transformación de los datos en información útil), **almacenamiento** (conservación de datos e información) y **salida** (distribución de la información a quien la necesita). Para que la información resultante sea útil debe reunir características como exactitud, completitud, relevancia y disponibilidad en el momento oportuno.

Este tema se divide en dos grandes bloques. Primero, la **representación de la información**: cómo se codifican los datos en binario (unidades, sistemas de numeración, números con signo y códigos de caracteres). Segundo, la **arquitectura de ordenadores**: cómo se organizan físicamente los componentes que procesan esa información (CPU, memorias, placa base y buses).

2. Representación de la información

2.1 El bit, el byte y las unidades de medida

El **bit** (dígito binario, del inglés binary digit) es la **unidad mínima de información**. Solo puede tomar dos valores, **0 o 1**, que representan dos estados opuestos (encendido/apagado, verdadero/falso). Agrupando bits se obtiene mayor capacidad de representación: con 8 bits se forma un **byte**, la unidad básica de almacenamiento con la que se mide la información en la práctica.

Agrupación	Nº de bits
Bit	1 bit
Crumb	2 bits
Nibble	4 bits
Byte (octeto)	8 bits
Palabra (word)	16 bits
Palabra doble (double word)	32 bits

Para medir cantidades grandes se usan **múltiplos del byte**. Existen dos escalas distintas que conviene no confundir. Los **prefijos decimales** (kilo, mega, giga…) son potencias de 10 y se emplean en redes y discos duros: 1 kilobyte (KB) = 1.000 bytes. Los **prefijos binarios** (kibi, mebi, gibi…) son potencias de 2 y se usan en memoria RAM: 1 kibibyte (KiB) = 1.024 bytes. El prefijo con «i» siempre es algo mayor que su equivalente decimal.

Escala decimal (×1000) — redes y discos duros

Unidad	Equivalencia (base 1000)
Kilobyte (KB)	10^3 bytes = 1.000 bytes
Megabyte (MB)	10^6 bytes = 1.000 KB
Gigabyte (GB)	10^9 bytes = 1.000 MB
Terabyte (TB)	10^{12} bytes = 1.000 GB
Petabyte (PB)	10^{15} bytes = 1.000 TB
Exabyte (EB)	10^{18} bytes = 1.000 PB
Zettabyte (ZB)	10^{21} bytes = 1.000 EB
Yottabyte (YB)	10^{24} bytes = 1.000 ZB

Escala binaria (×1024) — memoria RAM

Unidad	Equivalencia (base 1024)
Kibibyte (KiB)	2^{10} bytes = 1.024 bytes
Mebibyte (MiB)	2^{20} bytes = 1.024 KiB
Gibibyte (GiB)	2^{30} bytes = 1.024 MiB
Tebibyte (TiB)	2^{40} bytes = 1.024 GiB
Pebibyte (PiB)	2^{50} bytes = 1.024 TiB
Exbibyte (EiB)	2^{60} bytes = 1.024 PiB
Zebibyte (ZiB)	2^{70} bytes = 1.024 EiB
Yobibyte (YiB)	2^{80} bytes = 1.024 ZiB

El orden de menor a mayor es: **byte < KB < MB < GB < TB < PB < EB < ZB < YB**. Cada salto multiplica por 1.000 (decimal) o por 1.024 (binario). Así, **1 Exabyte (EB) = 1.000 Petabytes (PB)**, y el **Yottabyte** es la mayor de las unidades habituales. Por encima, el Sistema Internacional añadió recientemente el **Ronnabyte (RB, 10^{27})** y el **Quettabyte (QB, 10^{30})**.

💡 **TRUCO MNEMOTÉCNICO**

Para recordar las unidades grandes: **«PEZ»** → **P**eta, **E**xa, **Z**etta, y por encima el **Y**ottabyte (piensa en «**Y**oda»). La secuencia completa es **KB · MB · GB · TB · PB · EB · ZB · YB**.

⚠️ **CLAVE DE EXAMEN · 2016 · P53 (R) · 2018 · P23**

PREGUNTA ORIGINAL *"1 ExaByte (EB) se corresponde con… / ¿cuál de las siguientes unidades de medida de almacenamiento tiene mayor capacidad?"*

Muy preguntado: **1 EB = 1000 PB** (cada salto de la escala decimal multiplica por 1000) y el orden **YB > ZB > EB > PB** (el **Yottabyte** es el mayor). No confundir la escala decimal (×1000, discos y redes) con la binaria (×1024, memoria RAM).

2.2 Sistemas de numeración y conversiones

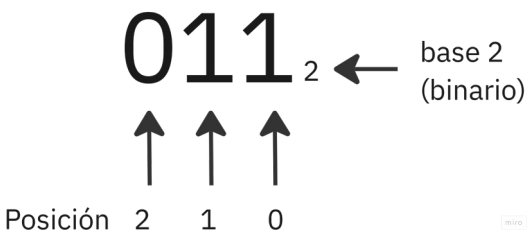
Un **sistema de numeración** es un conjunto de símbolos y reglas para representar números. El que usamos a diario es el **decimal** (base 10, dígitos 0–9). Los ordenadores trabajan internamente en **binario** (base 2, dígitos 0 y 1), pero para escribir esos valores de forma más compacta se emplean también el **octal** (base 8, dígitos 0–7) y, sobre todo, el **hexadecimal** (base 16, dígitos del 0–9 y las letras A–F, donde A=10, B=11, C=12, D=13, E=14 y F=15).

Sistema	Base	Dígitos válidos
Binario	2	0, 1
Octal	8	0, 1, 2, 3, 4, 5, 6, 7
Decimal	10	0–9
Hexadecimal	16	0–9 y A, B, C, D, E, F

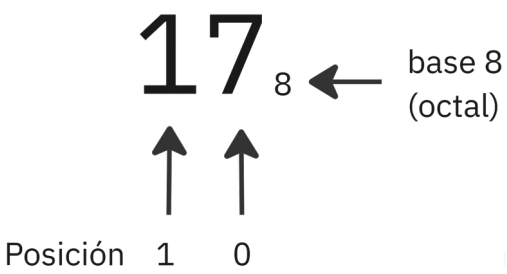
Conversión a decimal

Para pasar un número de cualquier base a decimal se multiplica cada dígito por la base elevada a la posición que ocupa (empezando por 0 a la derecha) y se suman los resultados.

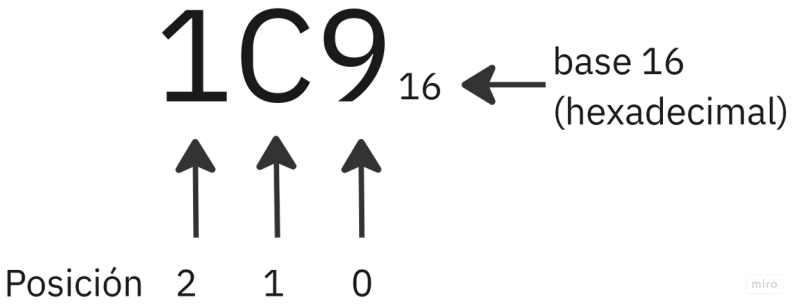
Ejemplo con el binario **011**: $0\times2^2 + 1\times2^1 + 1\times2^0 = 0 + 2 + 1 =$ **3** en decimal



Ejemplo con el octal **17**: $1\times8^1 + 7\times8^0 = 8 + 7 =$ **15** en decimal



Ejemplo con el hexadecimal **1C9**: $1\times16^2 + C(12)\times16^1 + 9\times16^0 = 256 + 192 + 9 =$ **457** en decimal.



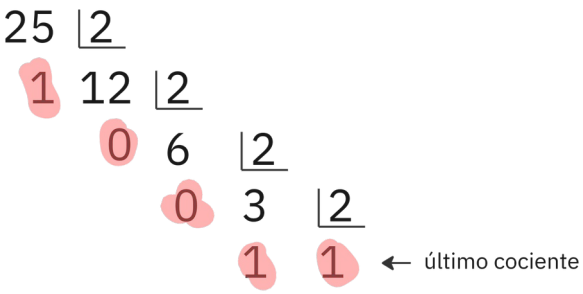
Conversión de decimal a otra base

Se emplea el **método de divisiones sucesivas**: se divide el número decimal entre la base de destino de forma repetida, anotando los restos, y el resultado se lee **de abajo arriba** (del último cociente al primer resto).

- Ejemplo, pasar **25 a binario**:

25÷2 = 12 (resto 1), 12÷2 = 6 (0), 6÷2 = 3 (0), 3÷2 = 1 (1), 1÷2 = 0 (1).

Leyendo los restos al revés: **11001** (que en decimal es 16+8+1 = 25).

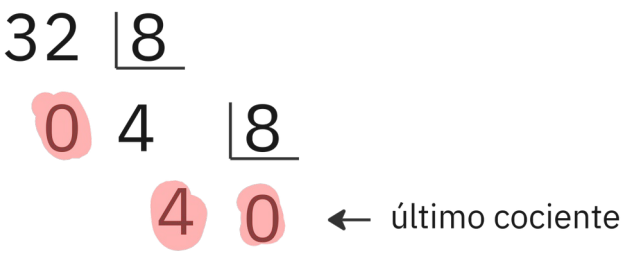


leído de abajo hacia arriba: 11001

El mismo método sirve para octal (dividiendo entre 8) y hexadecimal (entre 16).

- Ejemplo, pasar 32 a **octal**:

32÷8 = 4 (resto 0), 4÷8 = 0 (resto 4).
Leyendo los restos al revés: 40 (que en decimal es 4×8 + 0 = 32).



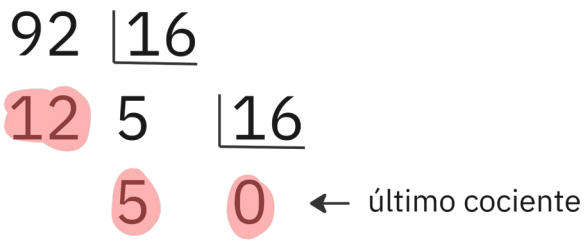
leído de abajo hacia arriba: 40_8

Al ser el último cociente cero, este se puede omitir ya que 040 = 40.

- Ejemplo, pasar 92 a **hexadecimal**:

92÷16 = 5 (resto 12), 5÷16 = 0 (resto 5).

Leyendo los restos al revés y traduciendo a símbolos: 5C (que en decimal es 5×16 + 12 = 80 + 12 = 92).

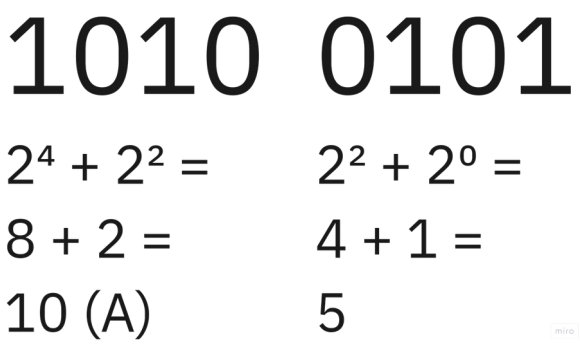


leído de abajo hacia arriba: $5(12) = 5C_{16}$

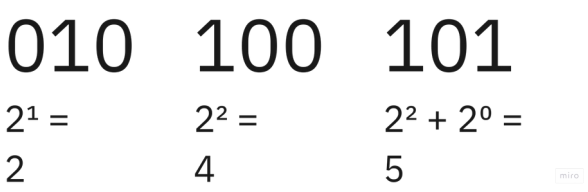
Conversión entre binario, octal y hexadecimal

Como 8 = 2^3 y 16 = 2^4, **cada dígito octal equivale a 3 bits y cada dígito hexadecimal a 4 bits**. Por eso es posible convertir estos, agrupando los bits (desde la derecha) y traduciendo cada grupo, sin pasar por decimal.

Ejemplo binario→hexadecimal: **1010 0101** → 1010 = A y 0101 = 5 → **A5** (=165 en decimal).



Ejemplo binario→octal: **010 100 101** → 2, 4, 5 → **245_8**.



El proceso inverso (hexadecimal u octal → binario) consiste en sustituir cada dígito por sus 4 o 3 bits correspondientes.

Al final de este documento tenéis varios ejercicios con los que poder practicar las conversiones.

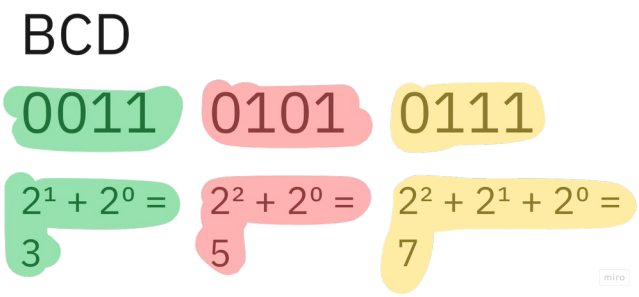
⚠️ **CLAVE DE EXAMEN · 2016 · P23 · 2016 · P26 · 2017 · P22 · 2024 · P20**

PREGUNTA ORIGINAL *"¿Qué palabra es un hexadecimal válido? · valor hexadecimal de 10100101 · valor decimal de 1C9 · ¿cuál de los números es el menor?"*

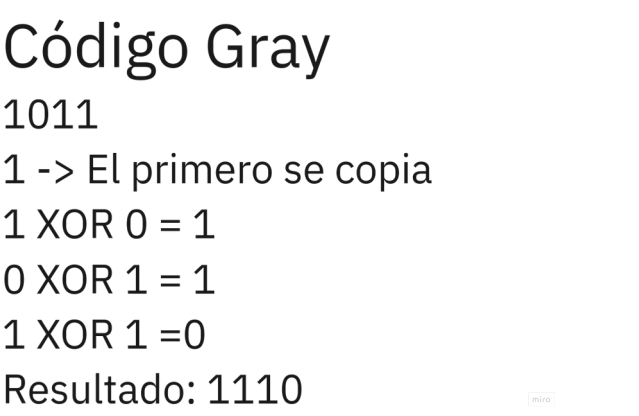
Cuatro tipos de pregunta sobre numeración, todos habituales.
(1) **Validez hexadecimal**: solo son dígitos válidos 0–9 y A–F; «ACCEDA» es hex válido, pero «CEGADA» (G), «EBOCA» (O) o «BECADAS» (S) no lo son, ni tampoco «FR» (la R no existe en hex).
(2) **Binario → hexadecimal**: se agrupan los bits de 4 en 4 → 1010 0101 = **A5** (=165 en decimal).
(3) **Hexadecimal → decimal**: 1C9 = 1×16^2 + C(12)×16 + 9 = **457**.
(4) **Comparar bases**: pasa todo a decimal antes de comparar; p. ej. 1226_8 = 662 es el **menor** frente a $2A0_{16}$ = 672, 690 y 1010110100_2 = 692.

2.3 Otros códigos: BCD, Gray y Base64

El **BCD (Binary-Coded Decimal, decimal codificado en binario)** representa cada dígito decimal (0–9) con su propio grupo de 4 bits, en lugar de convertir el número completo a binario puro. Evita errores de redondeo y facilita la conversión, por lo que se usa en calculadoras, relojes y sistemas financieros. Ejemplo: 357 en BCD es **0011 0101 0111**.



El **código Gray** es una codificación binaria en la que **dos valores consecutivos se diferencian en un único bit**. Esto reduce los errores de lectura en sensores y codificadores de posición (encoders), donde el cambio simultáneo de varios bits podría provocar lecturas erróneas. Para convertir un binario a Gray se **copia el bit más significativo** y cada bit siguiente se obtiene haciendo la operación **XOR** entre ese bit del binario y el anterior. XOR es un operador lógico que da 1 si los valores son distintos. Ejemplo: 1011 → el primer bit se copia (1); luego 1 XOR 0 = 1; 0 XOR 1 = 1; 1 XOR 1 = 0 → resultado **1110**.



Base64 convierte datos binarios en texto usando 64 caracteres imprimibles (A–Z, a–z, 0–9, «+» y «/», con «=» como relleno). Permite transportar archivos binarios (imágenes, adjuntos) por medios que solo aceptan texto, como el correo electrónico. **No es cifrado ni compresión**: no aporta seguridad y es totalmente reversible. Agrupa los bits de 6 en 6 ($2^6 = 64$) y añade «=» al final cuando el número de bytes no es múltiplo de 3. Ejemplo: la palabra «Hola» (4 bytes) se codifica como **SG9sYQ==**, donde los dos «=» indican que faltaban 2 bytes para completar el último bloque de 3.

2.4 Representación de números con signo: complemento a dos

Para representar números **negativos** en binario, el método estándar es el **complemento a dos**. En él, el **bit más significativo** (el de más a la izquierda) indica el signo: **0 para positivo y 1 para negativo**. Su gran ventaja es que la resta se realiza con el mismo circuito que la suma (se suma el opuesto del sustraendo), lo que ahorra circuitería en el hardware.

Antes del complemento a dos se usaron otras dos representaciones que hoy conviene conocer solo para diferenciarlas: **signo-magnitud** (un bit para el signo y el resto para el valor) y **complemento a uno** (se invierten todos los bits). Ambas tienen el inconveniente del **doble cero** (+0 y −0), que el **complemento a dos** elimina, además de simplificar las operaciones; por eso es el estándar actual.

Para obtener el complemento a dos de un número se **invierten todos sus bits y se suma 1**. Ejemplo, el −3 con 4 bits: 3 = 0011 → invertir = 1100 → +1 = **1101**. Con 4 bits el rango representable va del **−8 (1000) al +7 (0111)**.

La gran ventaja práctica es que **restar equivale a sumar el opuesto**. Ejemplo, 6 − 5 con 4 bits: 6 = 0110 y −5 en complemento a dos = 1011; su suma es 0110 + 1011 = 1**0001**, se **descarta el acarreo**

del quinto bit y queda **0001 = 1**, el resultado correcto. En cambio, si el resultado real se sale del rango [−8, +7] se produce **desbordamiento** (overflow): 6 + 5 = 0110 + 0101 = 1011, que en complemento a dos es −5, no 11. La señal es clara: **sumar dos positivos y obtener un negativo** —o dos negativos y obtener un positivo— indica desbordamiento.

2.5 Representación de caracteres

Un **código de caracteres** asigna a cada letra, dígito o símbolo un valor numérico que el ordenador pueda almacenar. La codificación ha evolucionado para admitir cada vez más alfabetos y símbolos.

Código	Bits / caracteres	Descripción
EBCDIC	8 bits	Código de IBM para grandes computadoras (mainframes).
ASCII estándar	7 bits · 128 caract.	Alfabeto inglés, dígitos y símbolos básicos.
ASCII extendido	8 bits · 256 caract.	Añade caracteres nacionales y de dibujo.
Unicode	> 1 millón caract.	17 planos de 65.536 caracteres; todos los alfabetos.
UTF-8	1–4 bytes/carácter.	Codificación dominante de Unicode; compatible con ASCII.
UTF-16 / UTF-32	2 o 4 bytes	Alternativas de Unicode, menos eficientes en general.

3. Lógica digital: puertas lógicas

Toda la circuitería de un ordenador se construye con **puertas lógicas**: pequeños circuitos que reciben uno o varios bits de entrada y producen un bit de salida según una regla fija. Combinando millones de ellas se forman sumadores, memorias y procesadores. Se dividen en **básicas** (AND, OR, NOT) y **secundarias**, que son las básicas negadas (NAND, NOR, XNOR).

Puerta	Salida = 1 cuando…	0·0	0·1	1·0	1·1
AND	las dos entradas son 1	0	0	0	1
OR	al menos una entrada es 1	0	1	1	1
XOR	las entradas son distintas	0	1	1	0
NAND	NO se cumple AND	1	1	1	0
NOR	NO se cumple OR	1	0	0	0
XNOR	las entradas son iguales	1	0	0	1

La puerta **NOT** actúa sobre una sola entrada e invierte su valor (0→1, 1→0). Las secundarias son la negación de las básicas: **NAND** = NOT(AND), **NOR** = NOT(OR) y **XNOR** = NOT(XOR).

⚠ **CLAVE DE EXAMEN · 2018 · P21**

PREGUNTA ORIGINAL *"Suponiendo que a=0 y b=1, ¿cuál de las siguientes sentencias es INCORRECTA? (a XOR b=1 · a XNOR b=0 · a NOR b=1 · a NAND b=1)"*

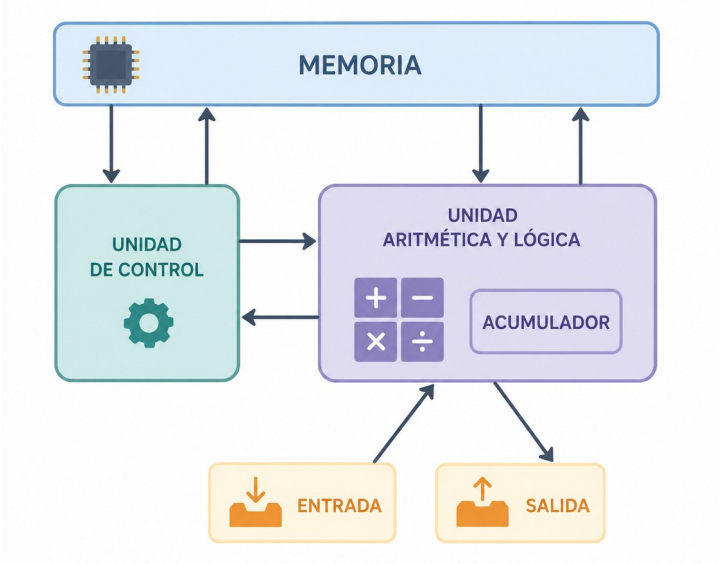
Con a=0 y b=1: **a XOR b = 1** (distintas), **a XNOR b = 0**, **a NAND b = 1** (NOT de AND(0,1)=0) y **a NOR b = 0** (NOT de OR(0,1)=1). Por eso afirmar que «a NOR b = 1» es **INCORRECTO**: es el error que busca la pregunta.

4. Arquitectura de ordenadores

La **arquitectura de computadoras** es el diseño conceptual y la estructura operativa de un sistema informático: define cómo se organizan e interactúan sus componentes de hardware (conjunto de instrucciones, registros, gestión de memoria, entrada/salida). Su componente central es la **unidad de procesamiento (CPU)**, acompañada de la memoria y los dispositivos de entrada/salida.

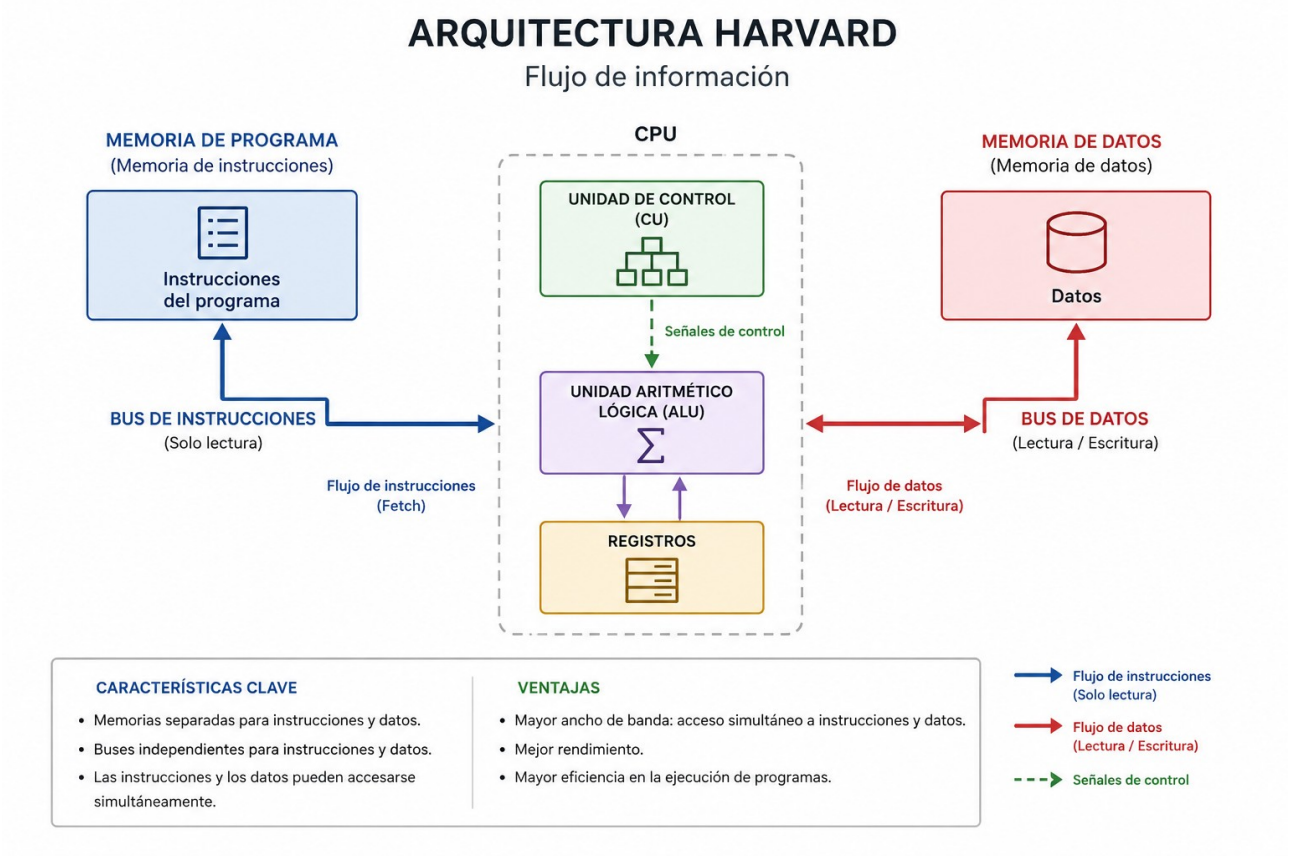
4.1 Arquitectura de Von Neumann

Es la arquitectura **más extendida**. Emplea **una única memoria y un solo bus** compartidos para almacenar y transportar tanto los datos como las instrucciones. Esto **simplifica el diseño**, pero al compartir un mismo bus puede generar un **cuello de botella** entre la CPU y la memoria (el llamado «cuello de botella de Von Neumann»). En este modelo, la **Unidad de Control** supervisa la transferencia de información e indica a la ALU qué operación ejecutar.



4.2 Arquitectura Harvard

Utiliza **memorias y buses físicamente separados** para las instrucciones y para los datos. Al disponer de dos vías independientes, la CPU puede **leer una instrucción y acceder a un dato de forma simultánea**, eliminando el cuello de botella de Von Neumann y mejorando el rendimiento. Su origen está en computadores tempranos como el Harvard Mark I. La **arquitectura Harvard modificada** mantiene la separación solo en la caché pero comparte la memoria principal, y es la más común en los procesadores modernos.



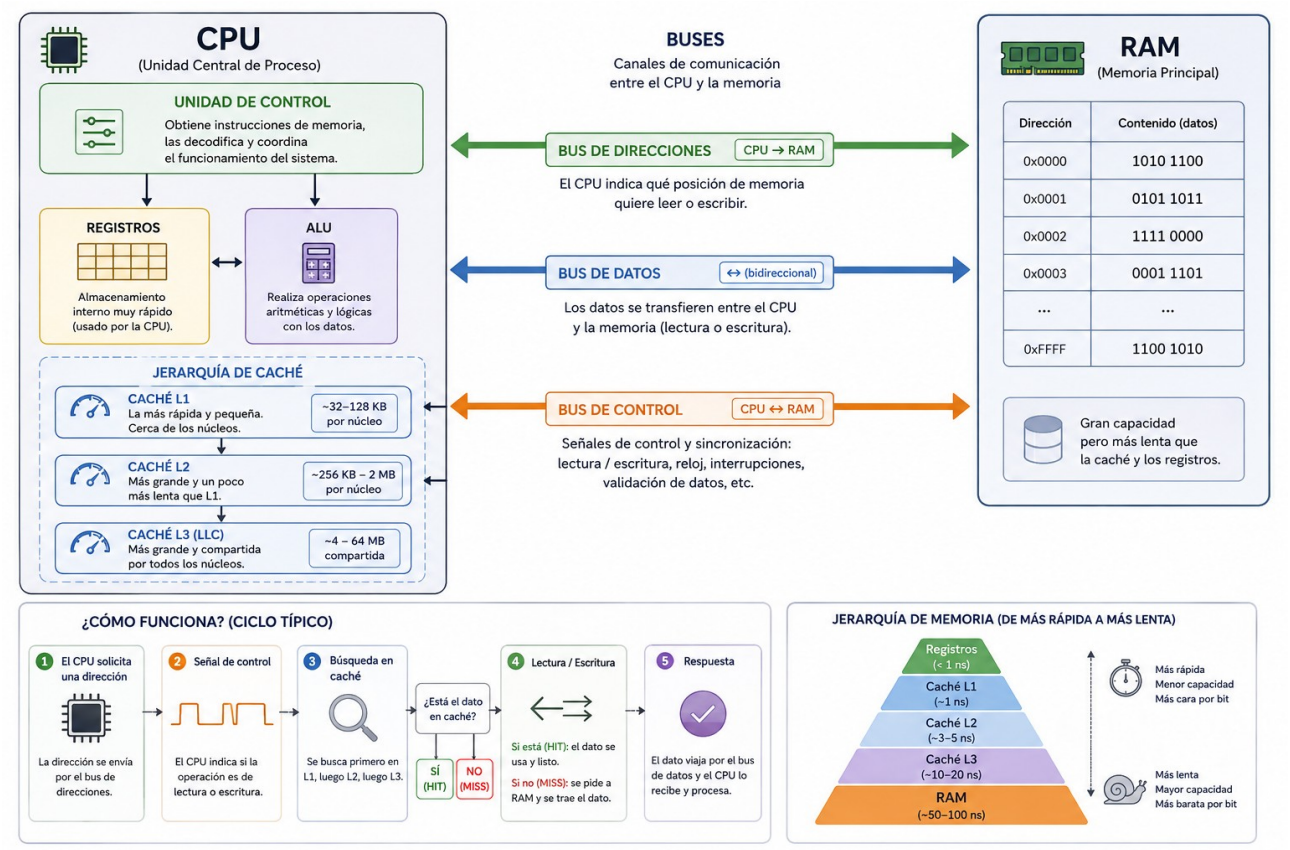
⚠️ **CLAVE DE EXAMEN · 2019 · P81 (R)**

PREGUNTA ORIGINAL *"La arquitectura de ordenadores Harvard... (tiene dos espacios de memoria separados, uno para datos y otro para instrucciones)."*

Contraste clave: **Von Neumann = una sola memoria y un solo bus** para datos e instrucciones; **Harvard = dos espacios de memoria separados** (uno para datos y otro para instrucciones) con buses independientes. Harvard **no** es un tipo de Von Neumann.

5. La unidad central de proceso (CPU)

El **microprocesador o CPU (Unidad Central de Proceso)** es el encargado de ejecutar todos los programas, desde el sistema operativo hasta las aplicaciones. Sus dos componentes principales son la **Unidad de Control (UC)** y la **Unidad Aritmético-Lógica (ALU)**, a los que se suman los **registros** internos. Puede incluir además una **FPU** (unidad de coma flotante, para operaciones con decimales).



5.1 Unidad Aritmético-Lógica (ALU)

La **ALU** es el componente de la **CPU** que **realiza las operaciones aritméticas (sumar, restar...) y lógicas (AND, OR...)** sobre los datos. Recibe como datos de entrada operandos (ej. En 5+3, los operandos son 5 y 3, el + es la operación que le indica la Unidad de Control). No decide qué operación hacer —eso lo indica la Unidad de Control— sino que la ejecuta.

⚠️ **CLAVE DE EXAMEN · 2022 · P24**

PREGUNTA ORIGINAL *"La ALU es una parte de... (la CPU)."*

Pregunta directa y frecuente: **la ALU forma parte de la CPU** (no de la memoria, ni es un bus ni un multiplexor).

5.2 Unidad de Control (UC) y el contador de programa

La **Unidad de Control** dirige el flujo de información entre los registros y **coordina la ejecución de las instrucciones**: las extrae de la memoria, las interpreta e indica a la ALU qué debe hacer. Está formada

por el **contador de programa (PC)**, el **registro de instrucciones (RI)**, el **decodificador** y el **reloj** del sistema.

Elemento de la UC	Función
Contador de programa (PC)	Guarda la DIRECCIÓN de memoria de la SIGUIENTE instrucción a ejecutar.
Registro de instrucciones (RI)	Almacena la instrucción que se está ejecutando en ese momento.
Decodificador	Interpreta la instrucción que hay en el registro de instrucciones.
Reloj del sistema	Marca el ritmo (la sincronización) de funcionamiento del procesador.

El ciclo de instrucción

La CPU ejecuta los programas repitiendo continuamente un **ciclo de instrucción** de tres fases: **búsqueda (fetch)** —la Unidad de Control lee de la memoria la instrucción cuya dirección marca el contador de programa y la lleva al registro de instrucciones—, **decodificación (decode)** —el decodificador interpreta qué debe hacerse— y **ejecución (execute)** —la ALU u otra unidad realiza la operación—. Tras cada instrucción, el contador de programa avanza a la siguiente. Este ritmo lo marca el **reloj**, medido en hercios (un procesador de 3 GHz genera 3.000 millones de pulsos por segundo).

⚠️ **CLAVE DE EXAMEN · 2016 · P21 · 2017 · P19**

PREGUNTA ORIGINAL *"El «contador de programa» de una CPU contiene la… (dirección de la instrucción a ejecutar)."*

El **contador de programa (PC)** contiene la **dirección** de la siguiente instrucción a ejecutar, **no** la instrucción en sí (esa se carga en el registro de instrucciones) ni la dirección donde dejar el resultado. Y en Von Neumann, es la **Unidad de Control** la que supervisa la transferencia e indica a la ALU la operación.

5.3 Registros

Los **registros** son unidades de almacenamiento temporal de **muy alta velocidad** situadas dentro de la propia CPU. Se dividen en **no accesibles** por el programador (como el registro de instrucción) y **accesibles**, que a su vez pueden ser de uso específico (contador de programa, puntero de pila, acumulador, registros de estado o flags) o de uso general (para datos y direcciones durante la ejecución).

5.4 Memoria caché

La **memoria caché** es una capa de almacenamiento **muy rápida** que guarda un subconjunto de datos de uso frecuente, de modo que la CPU los obtiene más deprisa que si tuviera que acudir a la memoria principal. Se organiza por niveles: cuanto más cerca del núcleo, más rápida pero más pequeña.

Nivel	Ubicación	Velocidad y tamaño
Caché L1	Dentro del núcleo del procesador	La MÁS RÁPIDA y la de MENOR capacidad (32–128 KB/núcleo).
Caché L2	Propia de cada núcleo	Mayor y algo más lenta que L1

		(256 KB – varios MB).
Caché L3	Compartida entre los núcleos	Mayor capacidad y menor velocidad (varios MB – decenas de MB).

⚠ CLAVE DE EXAMEN · 2026 · P23

PREGUNTA ORIGINAL *"¿Qué nivel de caché está integrado físicamente dentro del núcleo del procesador, es el más rápido y tiene la menor capacidad?"*

La **caché L1** está integrada físicamente **dentro del núcleo**, es la **más rápida** y la de **menor capacidad**. La L2 es mayor y más lenta; la L3 se comparte entre núcleos y es la mayor de las tres.

5.5 Núcleos del procesador

Un **núcleo** es una unidad de procesamiento completa (con su propia ALU) dentro de la CPU. Un **procesador de dos núcleos** (dual-core) integra **dos unidades de procesamiento independientes que trabajan al mismo tiempo**, lo que permite ejecutar tareas en **paralelo real**. Los núcleos **no** se reparten particiones del disco ni bloques fijos de RAM (comparten la memoria bajo el sistema operativo), ni se especializa uno en lógica y otro en aritmética: cada núcleo puede ejecutar cualquier instrucción.

⚠ CLAVE DE EXAMEN · 2016 · P24

PREGUNTA ORIGINAL *"Cuando se habla de un procesador de dos núcleos significa que... (hay dos chips trabajando independientemente y al mismo tiempo)."*

«Procesador de dos núcleos» significa **dos unidades de proceso independientes trabajando simultáneamente** (en paralelo). No implica repartirse el disco, la RAM ni las operaciones lógicas/aritméticas.

5.6 Medida de la potencia: FLOPS

La **potencia de cálculo** de un microprocesador se mide en **FLOPS** (Floating Point Operations Per Second, operaciones en coma flotante por segundo). No debe confundirse con el **gigabyte** (capacidad de almacenamiento) ni con los **FPS** (fotogramas por segundo en vídeo y juegos).

⚠ CLAVE DE EXAMEN · 2017 · P21

PREGUNTA ORIGINAL *"La unidad de medida de la potencia de un microprocesador es... (FLOPS)."*

La unidad de la **potencia de un microprocesador** son los **FLOPS**. «FBS» no existe; los FPS miden fotogramas, no potencia de cálculo.

5.7 Otras unidades de proceso: FPU y GPU

Junto al procesador principal existen unidades especializadas. La **FPU (unidad de coma flotante)**, hoy integrada en la propia CPU, se encarga de las operaciones con números **decimales** (coma flotante), mucho más costosas que las de enteros. La **GPU (unidad de procesamiento gráfico)** es un procesador con **cientos o miles de núcleos** más simples, diseñado para hacer muchas operaciones en

paralelo; nació para los gráficos, pero hoy es clave en inteligencia artificial y cálculo científico (encaja en el modelo **SIMD** de la taxonomía de Flynn). También se usa el **multihilo** (**Hyper-Threading** en Intel, SMT en general), que permite a un mismo núcleo físico ejecutar dos hilos de instrucciones de forma más eficiente.

6. Arquitecturas de procesador: CISC, RISC y ARM

La **arquitectura del procesador** define cómo está organizado internamente y cómo interpreta las instrucciones. Los dos modelos clásicos son **CISC** y **RISC**, que se diferencian en el tamaño y la complejidad de su juego de instrucciones.

Aspecto	CISC	RISC
Nombre	Complex Instruction Set Computing	Reduced Instruction Set Computing
Juego de instrucciones	Amplio y complejo	Reducido, simple y optimizado
Implementación	Microprogramación (característica esencial)	Instrucciones directamente en hardware
Ciclos por instrucción	Variable; una instrucción hace mucho	Pocos ciclos por instrucción
Consumo / uso típico	x86 (Intel, AMD); sobremesa y servidores	Bajo consumo; móviles y tabletas

La arquitectura **ARM (Advanced RISC Machines)** es una **arquitectura avanzada de microprocesadores de tipo RISC**. Su juego de instrucciones reducido necesita menos transistores y favorece el **bajo consumo**, por lo que domina en dispositivos alimentados por batería como **móviles y tabletas**.

⚠️ **CLAVE DE EXAMEN · 2022 · P25 · 2019 · P25**

PREGUNTA ORIGINAL *"Señale la respuesta correcta sobre el modelo CISC / ¿Qué es la arquitectura ARM? (una arquitectura avanzada para microprocesadores RISC)."*

En **CISC**, la **microprogramación es una característica esencial**. Reducir el número de instrucciones, implementarlas directamente en hardware o predominar en móviles por su bajo consumo son rasgos de **RISC**, no de CISC. **ARM** es una arquitectura **RISC** avanzada.

Taxonomía de Flynn

La **taxonomía de Flynn** clasifica las arquitecturas de computadoras según cuántos flujos de **instrucciones** y de **datos** pueden procesar a la vez. Es la clasificación de referencia para hablar de procesamiento paralelo.

Tipo	Instrucciones · Datos	Ejemplo típico
SISD	Una instrucción · un dato	CPU secuencial clásica de un solo núcleo.
SIMD	Una instrucción · varios datos	GPU; instrucciones vectoriales AVX/SSE de Intel y AMD.
MISD	Varias instrucciones · un dato	Muy raro; sistemas redundantes

		de alta seguridad.
MIMD	Varias instrucciones · varios datos	Procesadores multinúcleo, servidores y supercomputadoras.

Las dos más utilizadas hoy son **SIMD** (por el auge de las GPU y el cálculo vectorial) y **MIMD** (por los procesadores multinúcleo).

Ley de Moore

La **Ley de Moore**, enunciada en 1965 por **Gordon Moore** (cofundador de Intel), no es una ley física sino una observación empírica: predijo que el **número de transistores** que caben en un chip se **duplicaría aproximadamente cada dos años** con un coste mínimo añadido. Ha marcado el ritmo de la industria durante décadas, aunque en los últimos años su cumplimiento se ha ido ralentizando por los límites físicos de la miniaturización.

7. Memorias

Además de la caché y los registros, el ordenador cuenta con dos grandes familias de memoria: la **ROM**, no volátil y de solo lectura, y la **RAM**, volátil y de lectura/escritura. Ambas son de **acceso aleatorio**: se puede acceder directamente a cualquier posición sin recorrer las anteriores.

7.1 Memoria ROM y sus variantes

La **ROM (Read Only Memory, memoria de solo lectura)** es **no volátil**: conserva su contenido al apagar el equipo. Almacena datos que no cambian o cambian poco, como el programa de arranque (**BIOS**). Sus variantes se diferencian en cómo se pueden borrar y regrabar.

Tipo de ROM	Característica
PROM	Se suministra virgen y se programa UNA sola vez (mediante un dispositivo especial).
EPROM	Se puede borrar y regrabar; se borra exponiéndola a luz ULTRAVIOLETA.
EEPROM	Se puede borrar y regrabar repetidas veces ELÉCTRICAMENTE, sin extraer el chip.
EAROM	Se borra aplicando una tensión eléctrica mediante impulsos idénticos a todas las celdas.

⚠️ **CLAVE DE EXAMEN · 2017 · P20**

PREGUNTA ORIGINAL *"Respecto a los tipos de memoria ROM: EEPROM puede ser borrada y grabada repetidas veces eléctricamente."*

La **EEPROM** se borra y regraba **eléctricamente** cuantas veces se quiera. Cuidado con las trampas: la **PROM** solo se programa **una vez**; la **EPROM** se borra con **luz ultravioleta** (no infrarrojos); y «EPPROM» **no existe**.

7.2 Memoria RAM: DRAM y SRAM

La **RAM (Random Access Memory)** es de **gran velocidad** y **volátil**: los datos se almacenan solo de forma temporal y se pierden al apagar. Según su tecnología se divide en dos grandes tipos:

Tipo	Tecnología	Necesita refresco	Uso típico
DRAM (dinámica)	Condensadores (pierden carga)	SÍ, refresco continuo	Memoria principal del sistema
SRAM (estática)	Transistores / biestables	NO (mientras esté alimentada)	Memoria caché (más rápida)

La **SDRAM (RAM dinámica síncrona)** mejora la DRAM funcionando en sincronía con el reloj del procesador. De ella derivan la **SDR SDRAM** (una lectura y una escritura por ciclo de reloj) y la **DDR SDRAM (Double Data Rate)**, que **duplica** ese rendimiento (dos operaciones por ciclo). Las sucesivas generaciones mejoran en velocidad, capacidad y consumo, y **no son compatibles entre sí** en la misma placa base (la muesca del módulo cambia para impedir montarlas mal).

Generación	Año aprox.	Pines (DIMM)	Notas
DDR (DDR1)	~2000	184	Primera generación DDR.
DDR2	~2003	240	Más rápida y menor consumo que DDR.
DDR3	~2007	240	Mayor eficiencia energética.
DDR4	~2014	288	Muy extendida; menor voltaje.
DDR5	~2020	288	Doble velocidad que DDR4; dos canales por módulo.

Aunque **DDR4 y DDR5 comparten 288 pines** en formato de sobremesa, **no son compatibles** porque la posición de la muesca es distinta.

⚠️ **CLAVE DE EXAMEN · 2016 · P22**

PREGUNTA ORIGINAL *"Señale cuál de las siguientes es un tipo de memoria estática de acceso aleatorio (RAM): XDR RAM · FPM-RAM · RDRAM · NVRAM."*

Distinción esencial: **SRAM = estática** (transistores, sin refresco, usada como caché) frente a **DRAM = dinámica** (condensadores, con refresco). Variantes como **XDR RAM, FPM-RAM y RDRAM son dinámicas (DRAM)**. La **NVRAM** es una RAM estática que además retiene la información sin alimentación (batería de respaldo).

7.3 La memoria RAM-CMOS de configuración

La **configuración básica del equipo** (parámetros de la BIOS, fecha y hora, orden de arranque...) se guarda en una pequeña memoria **RAM de tecnología CMOS (RAM-CMOS)**, de muy bajo consumo, **alimentada por una pila** de la placa base para que no se pierda al apagar. Es importante no confundirla con la ROM, que almacena el **programa** de la BIOS (el firmware), no su **configuración**.

En los equipos actuales la BIOS clásica ha sido sustituida por **UEFI** (Unified Extensible Firmware Interface, interfaz de firmware extensible unificada), un firmware más moderno que cumple la misma función de inicializar el hardware y arrancar el sistema operativo, pero con mejoras: interfaz gráfica,

arranque más rápido, soporte de discos de más de 2 TB mediante el particionado **GPT** (GUID Partition Table) y **arranque seguro** (Secure Boot), que impide cargar durante el inicio software no firmado digitalmente. La pila y la memoria de configuración de la placa base cumplen el mismo papel con BIOS y con UEFI.

⚠ **CLAVE DE EXAMEN · 2016 · P25**

PREGUNTA ORIGINAL *"¿Qué nombre recibe la memoria encargada de almacenar la configuración básica del equipo? (RAM-CMOS)."*

La memoria que guarda la **configuración básica** del equipo es la **RAM-CMOS** (alimentada por pila). La **ROM** guarda el firmware de la BIOS, no sus ajustes; la RAM convencional es volátil y los perdería.

8. La placa base y el bus del sistema

La **placa base** (motherboard) es el circuito impreso principal donde se conectan e intercomunican todos los componentes del ordenador. Entre sus elementos están el **zócalo del procesador** (conector donde se instala la CPU), el **chipset** (conjunto de circuitos que comunican procesador, memoria y periféricos) y el **reloj** del sistema.

⚠ **CLAVE DE EXAMEN · 2018 · P24**

PREGUNTA ORIGINAL *"¿Cuál NO es un componente de la placa base de un ordenador? (Chip EAC)."*

Elementos reales de la placa base: **zócalo del procesador, chipset y reloj**. El «**chip EAC**» **no existe** como componente de la placa base: es el distractor de la pregunta.

8.1 Factores de forma: ATX y MicroATX

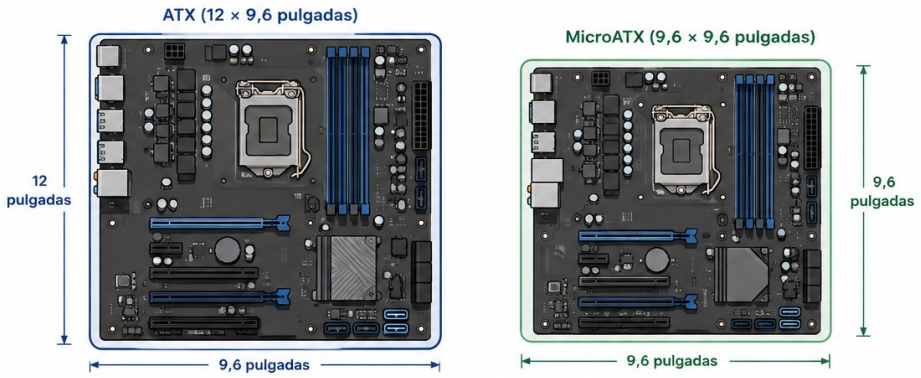
El **factor de forma** define el tamaño y la disposición física de la placa base (dónde van el zócalo, las ranuras de memoria y expansión o los conectores de alimentación). El estándar **ATX** tiene un tamaño máximo de **12 × 9,6 pulgadas**. El **MicroATX** es una **evolución reducida del ATX** —deriva de él y mantiene la compatibilidad de anclajes y conectores de alimentación— con un tamaño máximo de **9,6 × 9,6 pulgadas**.

Existen más factores de forma dentro de dos familias. Los formatos **ATX** (E-ATX, ATX, MicroATX, FlexATX) comparten anclajes y alimentación; la familia **ITX** (creada por VIA) agrupa los formatos más pequeños para mini-PC y sistemas embebidos.

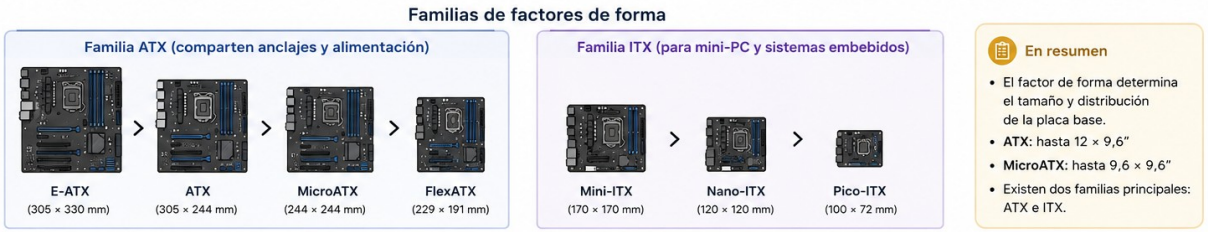
8.1 Factores de forma: ATX y MicroATX

El **factor de forma** define el tamaño y la disposición física de la placa base (dónde van el zócalo, las ranuras de memoria y expansión o los conectores de alimentación).

El estándar **ATX** tiene un tamaño máximo de 12 × 9,6 pulgadas.
El **MicroATX** es una evolución reducida del ATX —deriva de él y mantiene la compatibilidad de anclajes y conectores de alimentación— con un tamaño máximo de 9,6 × 9,6 pulgadas.



Compatibilidad
MicroATX deriva de ATX: comparte anclajes (agujeros de montaje) y conectores de alimentación, por lo que es compatible con muchas cajas y fuentes ATX.



En resumen

- El factor de forma determina el tamaño y distribución de la placa base.
- **ATX:** hasta 12 × 9,6"
- **MicroATX:** hasta 9,6 × 9,6"
- Existen dos familias principales: ATX e ITX.

Factor de forma	Tamaño máx. aprox.	Uso típico
E-ATX (Extended ATX)	305 × 330 mm (12 × 13″)	Gama alta, servidores, doble CPU.
ATX	305 × 244 mm (12 × 9,6″)	Sobremesa estándar.
MicroATX (mATX)	244 × 244 mm (9,6 × 9,6″)	Equipos compactos; deriva del ATX.
FlexATX	229 × 191 mm (9 × 7,5″)	Mini-torres económicas.
Mini-ITX	170 × 170 mm (6,7 × 6,7″)	HTPC, mini-PC, NAS.
Nano-ITX	120 × 120 mm	Sistemas embebidos.
Pico-ITX	100 × 72 mm	IoT y dispositivos muy pequeños.

⚠ **CLAVE DE EXAMEN · 2017 · P101 (R)**

PREGUNTA ORIGINAL *"En las placas base, el factor de forma MicroATX... (deriva del factor de forma ATX)."*

El **MicroATX deriva del factor de forma ATX**. Su tamaño máximo es **9,6 × 9,6 pulgadas**; el del ATX estándar, **12 × 9,6 pulgadas**.

8.2 El bus del sistema

Un **bus** es un conjunto de líneas por las que circula la información entre los componentes. El **bus del sistema** es el que **transfiere datos, direcciones y señales de control entre la CPU, la memoria y los periféricos**. Suele describirse como formado por tres partes: el **bus de datos** (transporta la información), el **bus de direcciones** (indica la posición de memoria) y el **bus de control** (coordina las operaciones).

⚠️ **CLAVE DE EXAMEN · 2026 · P24**

PREGUNTA ORIGINAL *"¿Cuál es la función principal del bus del sistema? (transferir datos, direcciones y señales de control entre la CPU, la memoria y los periféricos)."*

La función del **bus del sistema** es **transferir datos, direcciones y señales de control** entre la CPU, la memoria y los periféricos. No ejecuta instrucciones (eso es la CPU), ni suministra energía (la fuente de alimentación), ni almacena datos de forma permanente (el disco/SSD).

9. Ejercicios de conversión

En la siguiente página tenéis las soluciones, sin embargo, es muy recomendable tratar de resolverlos primero por vuestra cuenta para que se os quede.

Pasa los siguientes números a decimal:

Binario → decimal

1. 1011_{2}
2. 110010_{2}
3. 1101011010_{2}

Octal → decimal

4. 27_{8}
5. 145_{8}
6. 3721_{8}

Hexadecimal → decimal

7. $2F_{16}$
8. $A3_{16}$
9. $1B7E_{16}$

Pasa los siguientes números decimales a binario, octal o hexadecimal según lo indicado:

10. 45 → binario
11. 726 → binario
12. 100 → octal
13. 1250 → octal
14. 196 → hexadecimal
15. 1001 → hexadecimal

Solución:

Ejercicio 1 — 1011_2

$1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 =$

$8 + 0 + 2 + 1 =$

11

Ejercicio 2 — 110010_2

$1\times2^5 + 1\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 =$

$32 + 16 + 0 + 0 + 2 + 0 =$

50

Ejercicio 3 — 1101011010_2

$1\times2^9 + 1\times2^8 + 0\times2^7 + 1\times2^6 + 0\times2^5 + 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 =$

$512 + 256 + 0 + 64 + 0 + 16 + 8 + 0 + 2 + 0 =$

858

Ejercicio 4 — 27_8

$2\times8^1 + 7\times8^0 =$

$16 + 7 =$

23

Ejercicio 5 — 145_8

$1\times8^2 + 4\times8^1 + 5\times8^0 =$

$64 + 32 + 5 =$

101

Ejercicio 6 — 3721_8

$3\times8^3 + 7\times8^2 + 2\times8^1 + 1\times8^0 =$

$1536 + 448 + 16 + 1 =$

2001

Ejercicio 7 — $2F_{16}$

$2\times16^1 + 15\times16^0 =$

$32 + 15 =$

47

Ejercicio 8 — $A3_{16}$

$10\times16^1 + 3\times16^0 =$

$160 + 3 =$

163

Ejercicio 9 — $1B7E_{16}$

$1\times16^3 + 11\times16^2 + 7\times16^1 + 14\times16^0 =$

$4096 + 2816 + 112 + 14 =$

7038

Ejercicio 10 — 45 a binario

45 ÷ 2 = 22 resto 1

22 ÷ 2 = 11 resto 0

11 ÷ 2 = 5 resto 1

5 ÷ 2 = 2 resto 1

2 ÷ 2 = 1 resto 0

Cogemos el último cociente, y leemos los restos de abajo arriba =

101101_2

Ejercicio 11 — 726 a binario

726 ÷ 2 = 363 resto 0

363 ÷ 2 = 181 resto 1

181 ÷ 2 = 90 resto 1

90 ÷ 2 = 45 resto 0

45 ÷ 2 = 22 resto 1

22 ÷ 2 = 11 resto 0

11 ÷ 2 = 5 resto 1

5 ÷ 2 = 2 resto 1

2 ÷ 2 = 1 resto 0

Cogemos el último cociente, y leemos los restos de abajo arriba =

1011010110_2

Ejercicio 12 — 100 a octal

100 ÷ 8 = 12 resto 4

12 ÷ 8 = 1 resto 4

Cogemos el último cociente, y leemos los restos de abajo arriba =

144_8

Ejercicio 13 — 1250 a octal

1250 ÷ 8 = 156 resto 2

156 ÷ 8 = 19 resto 4

19 ÷ 8 = 2 resto 3

Cogemos el último cociente, y leemos los restos de abajo arriba =

2342_8

Ejercicio 14 — 196 a hexadecimal

196 ÷ 16 = 12 resto 4

Cogemos el último cociente, y leemos los restos de abajo arriba =

$C4_{16}$ (12 se convierte en C en hex)

Ejercicio 15— 1001 a hexadecimal

1001 ÷ 16 = 62 resto 9

62 ÷ 16 = 3 resto 14

Cogemos el último cociente, y leemos los restos de abajo arriba =

$3E9_{16}$ (14 se convierte en E)