TEMA 1

TECNOLOGÍA INFORMÁTICA Y NIVELES DE ABSTRACCIÓN

Escuela Ingeniería Informática Valladolid
UVa

Departamento de Informática
Universidad de Valladolid

La Revolución de los Computadores

- **Espectacular avance en la tecnología empleada en la fabricación de los computadores.**
  - Sustentado por la ley de Moore.
- **Ha permitido el desarrollo de nuevas aplicaciones**
  - Sector de Automoción
  - Tablets, Smartphones, Smartwatches
  - Ingeniería, Cálculo Científico
  - Inteligencia Artificial
- **Los computadores están en todas partes**

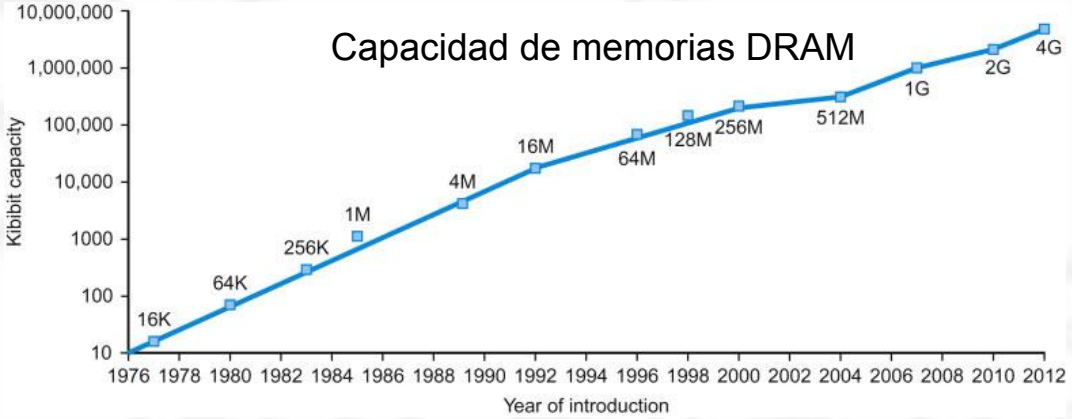
Tipos de Computadores

- **Sobremesa/Portátiles**
  - Propósito general, gran variedad de software.
  - Sometidos al compromiso entre coste y prestaciones.
- **Empotrados**
  - Propósito específico.
  - Ocultos como componentes del sistema.
  - Estrictas restricciones consumo/rendimiento/coste.
- **Servidores/Clusters**
  - Acceso a través de la red.
  - Alto rendimiento, capacidad y seguridad.
  - Alojados en CPDs con refrigeración adecuada.

Tendencias en la Tecnología

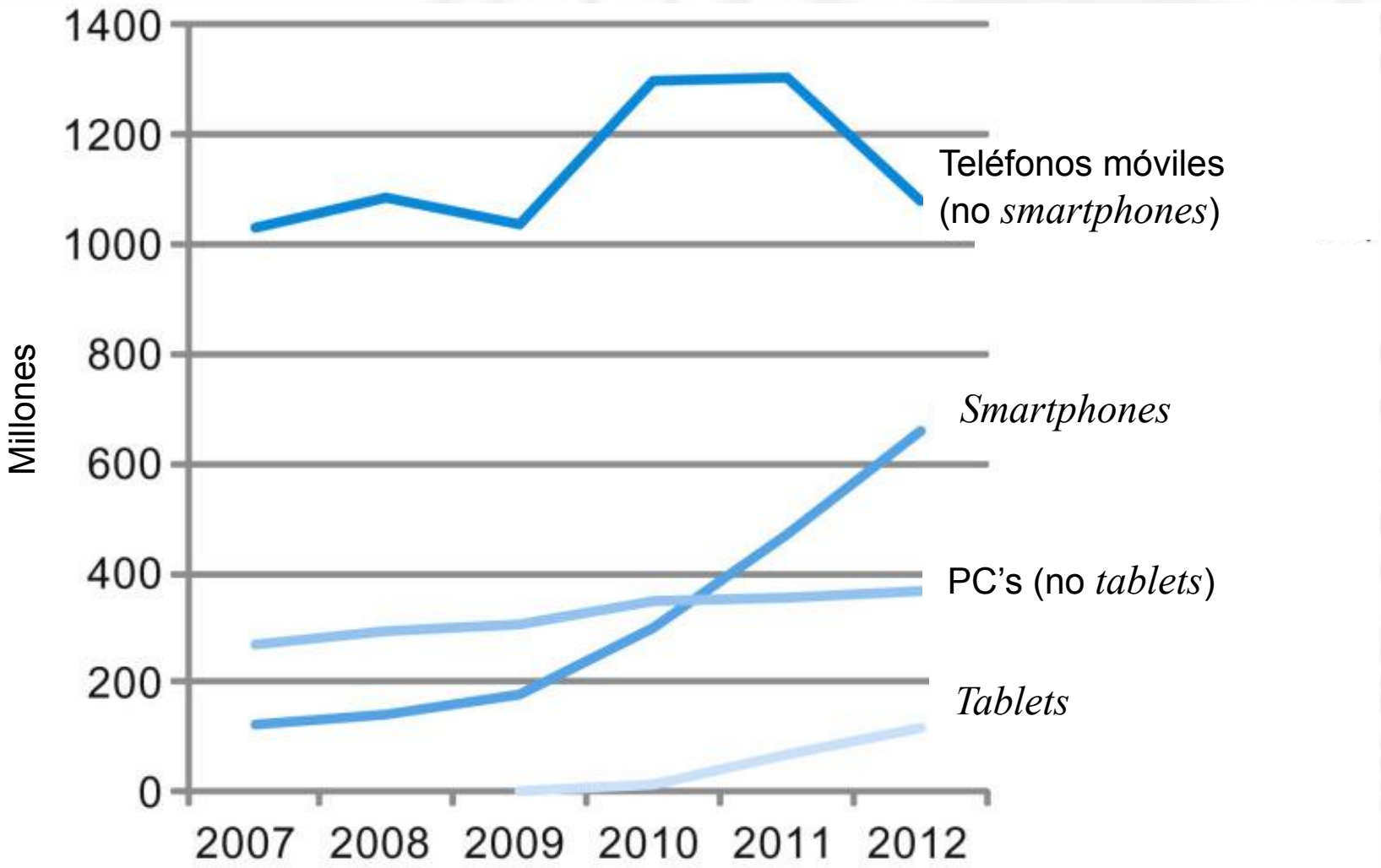
La tecnología electrónica continúa avanzando

- Se incrementa la capacidad y el rendimiento
- Se reduce el coste



Año	Tecnología	Relación rendimiento/coste
1951	Tubo de vacío	1
1965	Transistor	35
1975	Circuito integrado (IC)	900
1995	“Very large scale IC” (VLSI)	2.400.000
2013	“Ultra large scale IC”	250.000.000.000

Mercado de Procesadores



La era *post-PC*

- **Gran difusión de dispositivos móviles y "wearables"**
  - Utilidad: uso de aplicaciones conectadas a la red.
  - Funcionan con batería → Limitaciones.
  - *Tablets*, *Smartphones*, *Smartwatches*, etc.
- **Desarrollo en la "Nube"**
  - Gran capacidad de computación y de almacenamiento.
  - Cierta proporción de cómputo se realiza de forma local, pero la mayor parte se hace en un computador remoto.

¿Qué vamos a estudiar?

- **Cómo se traducen los programas al lenguaje máquina.**
  - Y cómo los ejecuta el hardware.
- **Interfaz entre hardware y software.**
- **Factores que afectan al rendimiento de un programa:**
  - Y cómo puede mejorarse.
- **Función de los diseñadores de hardware en la mejora del rendimiento.**
- **Qué es el procesamiento paralelo.**

Unidades de Medida de la Información

- **La unidad elemental de información es el *bit* (dos estados)**
  - 8 bits = 1 byte.
- **Tradicionalmente, los múltiplos del byte se han medido en potencias de 2**
  - Kilobyte (KB) = 2^{10} bytes; Megabyte (MB) = 2^{20} bytes, etc.
  - Inconveniente: se genera confusión al compararlo con el resto de magnitudes del Sistema Internacional de Unidades, que se basan en potencias de 10.
  - Se trató de resolver la confusión en 1998 definiendo el KB, MB, GB, etc. mediante potencias de 10 y utilizando nuevos prefijos para las potencias de 2.

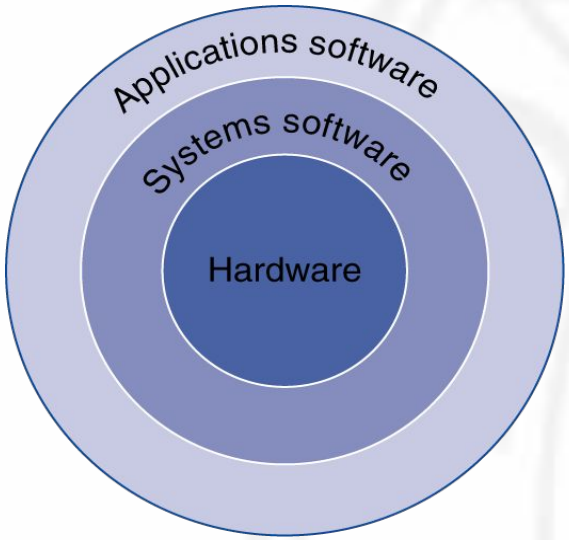
Múltiplos del byte

Notación decimal	Abreviatura	Valor	Notación binaria	Abreviatura	Valor	Diferencia
Kilobyte	KB	10^3	Kibibyte	KiB	2^{10}	2 %
Megabyte	MB	10^6	Mebibyte	MiB	2^{20}	5 %
Gigabyte	GB	10^9	Gibibyte	GiB	2^{30}	7 %
Terabyte	TB	10^{12}	Tebibyte	TiB	2^{40}	10 %
Petabyte	PB	10^{15}	Pebibyte	PiB	2^{50}	13 %
Exabyte	EB	10^{18}	Exbibyte	EiB	2^{60}	15 %

- En la comunidad informática no está plenamente aceptada la notación decimal (base 10)
- Los fabricantes de discos y los proveedores de red utilizan la notación decimal, pues salen beneficiados ($10^9 < 2^{30}$)

Niveles de Abstracción

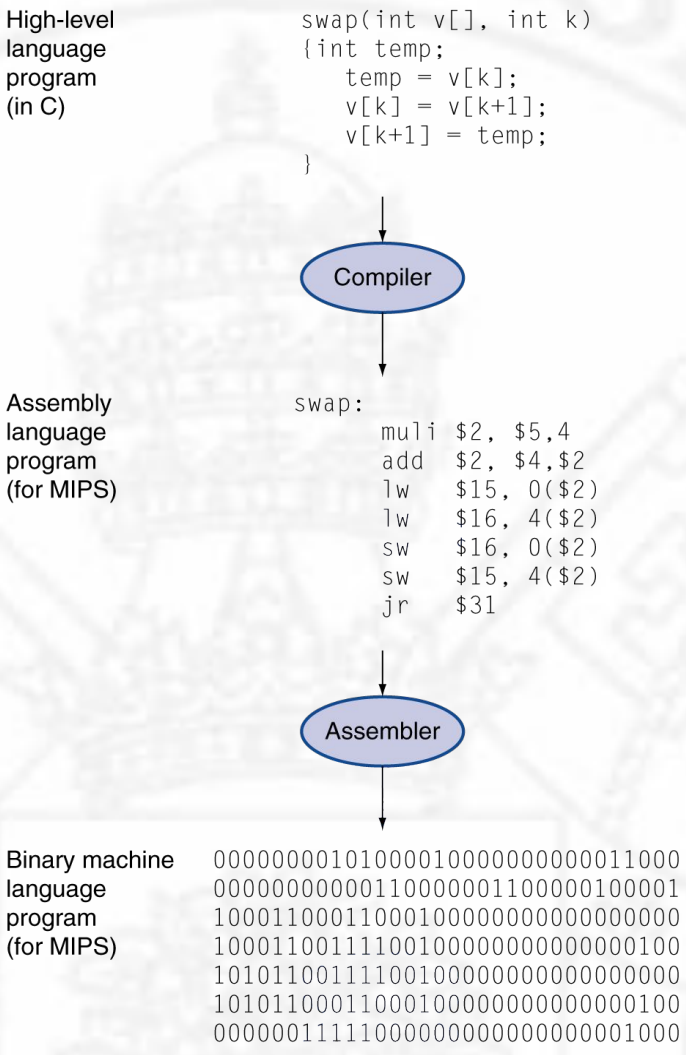
- **Software de Aplicación**
  - Escrito en lenguaje de alto nivel (HLL): C, C++, Java, Python.



- **Software de Sistema**
  - Compilador: traduce HLL a código máquina.
  - Sistema Operativo:
    - Gestión del Procesador: planificación de tareas.
    - Gestión de Memoria y Disco: paginación, particiones, espacio de almacenamiento.
    - Gestión de Seguridad y de E/S.
    - Compartición de recursos.
- **Hardware**
  - Procesador, Memoria y Controladores de E/S

Niveles de Código en Programas

- **Lenguaje de Alto Nivel**
  - Nivel de abstracción próximo al campo del problema.
  - Pensado para ser portable y fácilmente usado.
- **Lenguaje Ensamblador**
  - Representación textual de las instrucciones máquina.
- **Código Máquina:**
  - Dígitos binarios (bits)
  - Instrucciones y datos codificados
  - Representación a nivel hardware



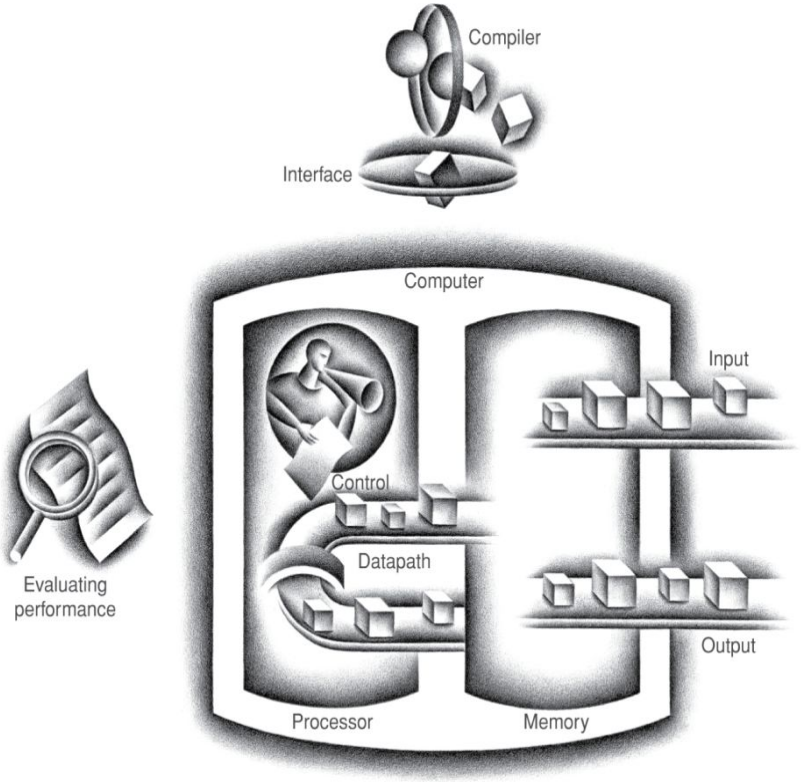
Las abstracciones

- **Facilitan la comprensión y uso de sistemas complejos:**
  - Ocultando detalles de bajo nivel.
- **Proporcionan interfaz entre el hardware y el software**
  - Repertorio de Instrucciones (ISA: *Instruction Set Architecture*)
- **Interfaz binaria de las aplicaciones**
  - ISA + Interfaz de Software del Sistema (funciones del S.O)
- **Ocultan los detalles de implementación**
  - Lo que está por debajo y su interfaz.

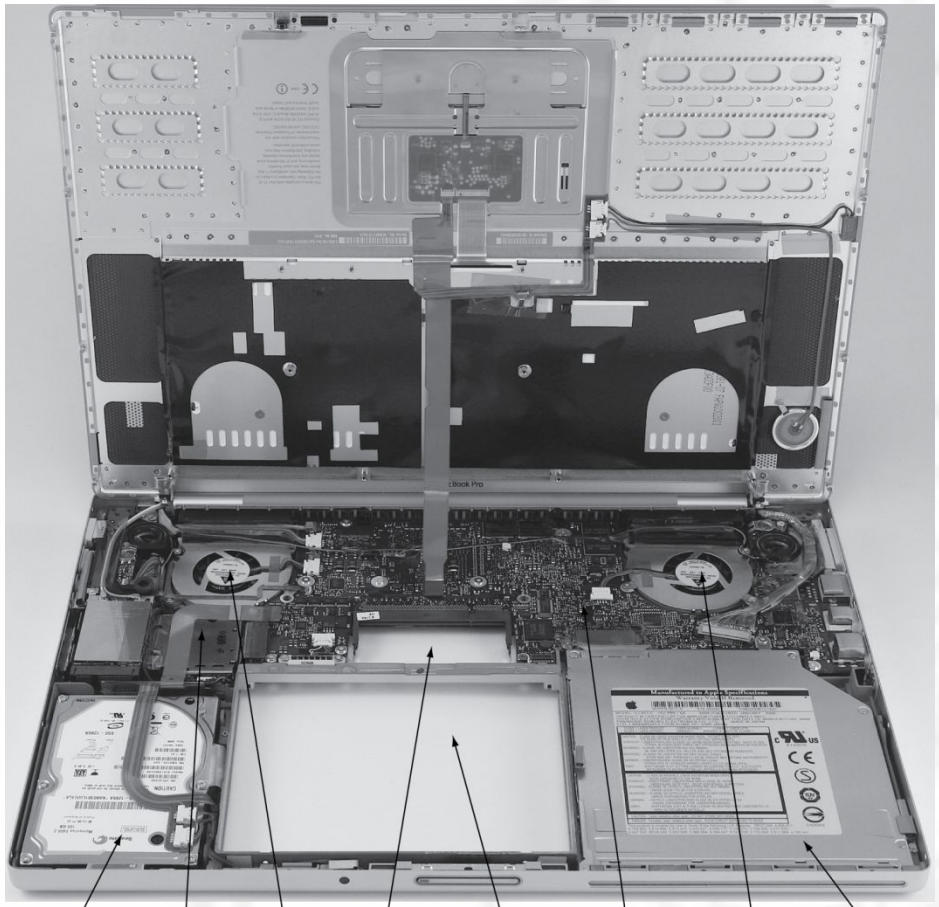
Componentes de un Computador

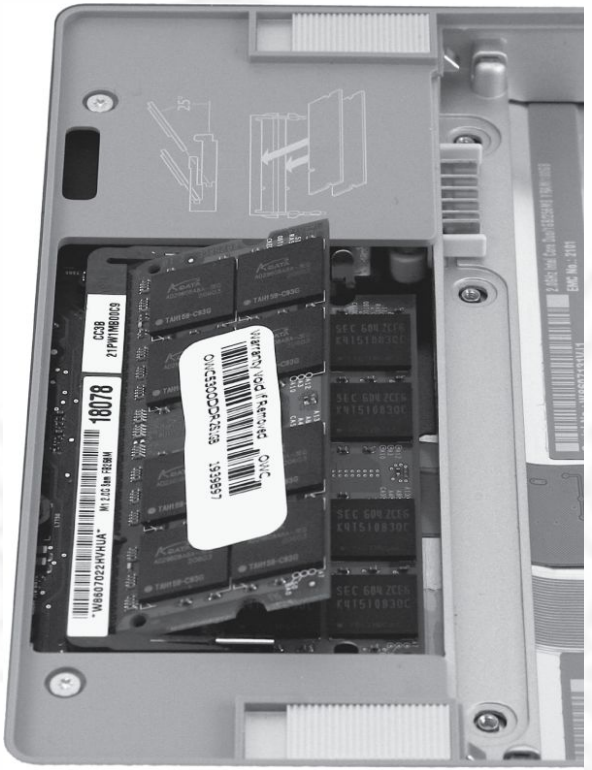
- **Mismos componentes para todos los tipos de computadores:**
  - Procesador: control y ejecución.
  - Memoria (principal, secundaria)
  - Entrada/Salida.
- **Entrada/Salida:**
  - Interfaz de usuario:
    - Pantalla, Teclado, Ratón.
  - Almacenamiento:
    - Disco Duro, CD/DVD, USB, SD.
  - Adaptadores de Red:
    - Ethernet, Wi-Fi.

Arquitectura Von Neumann

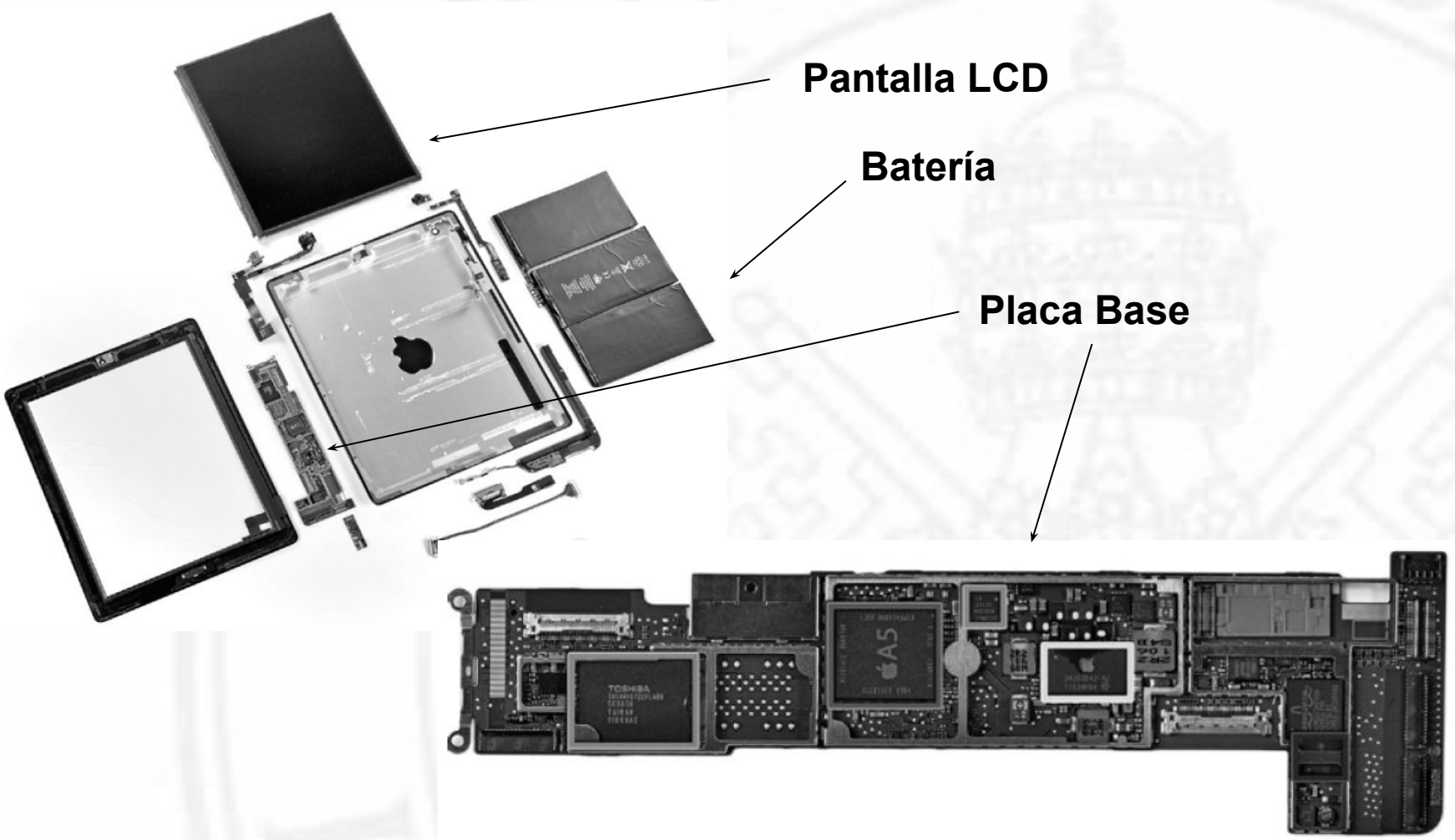


Ordenador Portátil



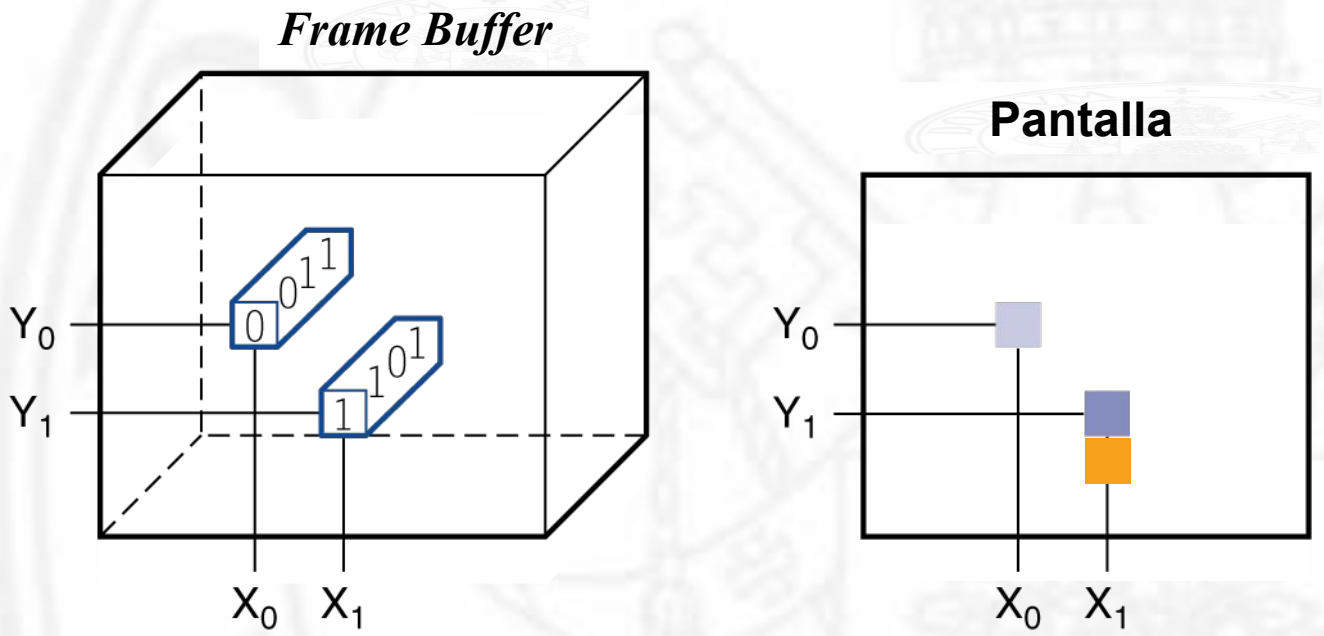


Tablet



Pantalla

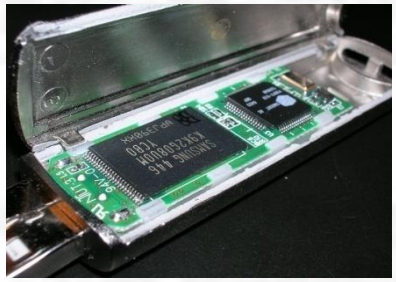
- **Se compone de puntos (**píxeles**), cada uno con su color.**
  - La imagen representada está determinada por el contenido del almacenamiento de la imagen en memoria (*frame buffer*)



Almacenamiento

- **Memoria Principal (*volátil*)**
  - Pierde los datos e instrucciones que contenga al cortar el suministro de energía.
- **Memoria Secundaria (*no volátil*)**
  - No pierde su contenido al cortar el suministro de energía.

  - Disco Duro (magnético o SSD), Disco Óptico (CDROM, DVD)



Memoria Principal

Analogía con una cajonera:

- Memoria → Cajonera (n cajones)
- Posición de Memoria → Cada cajón
- Dato → Contenido del cajón
- Dirección → Número del cajón

Memoria DDR4 8 GiB

No figura en el libro

Acceso a Memoria: Lectura

Leer, Cargar:

- Coger bola de cajón 0 y dejarla en la mano:
  - Mano ← Cajonera[Cajón 0]
- Coger dato de memoria de la posición 0 y depositarlo en un registro:
  - Registro ← Memoria[0]

Registro CPU

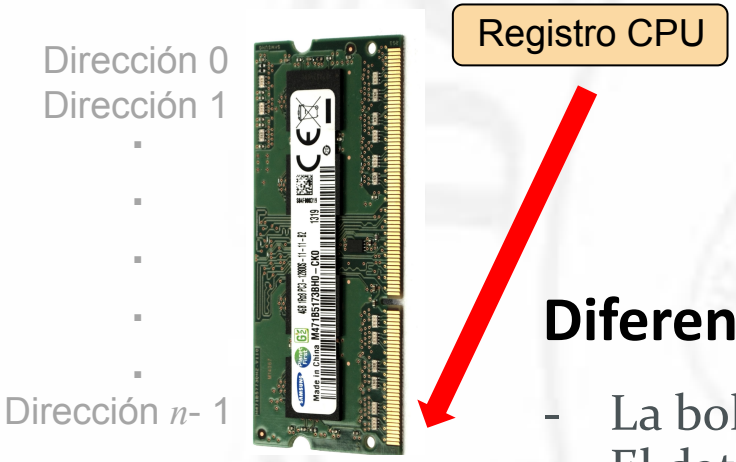
Diferencia:

- La bola está solo en la mano.
- El dato está en el registro y en la memoria.

Acceso a Memoria: Escritura

Almacenar/Guardar:

- Dejar la bola en el último cajón:
  - Cajonera[Cajón n-1] ← Mano
- Guardar el dato del registro en la última posición de memoria:
  - Memoria[n-1] ← Registro



Diferencia:

- La bola está solo en el cajón
- El dato está en el registro y en la memoria

Elementos de la Memoria

Bus de direcciones

Se deposita la dirección de memoria sobre la que se va a realizar la operación de L/E.

Celdas de memoria

Almacenan la información.

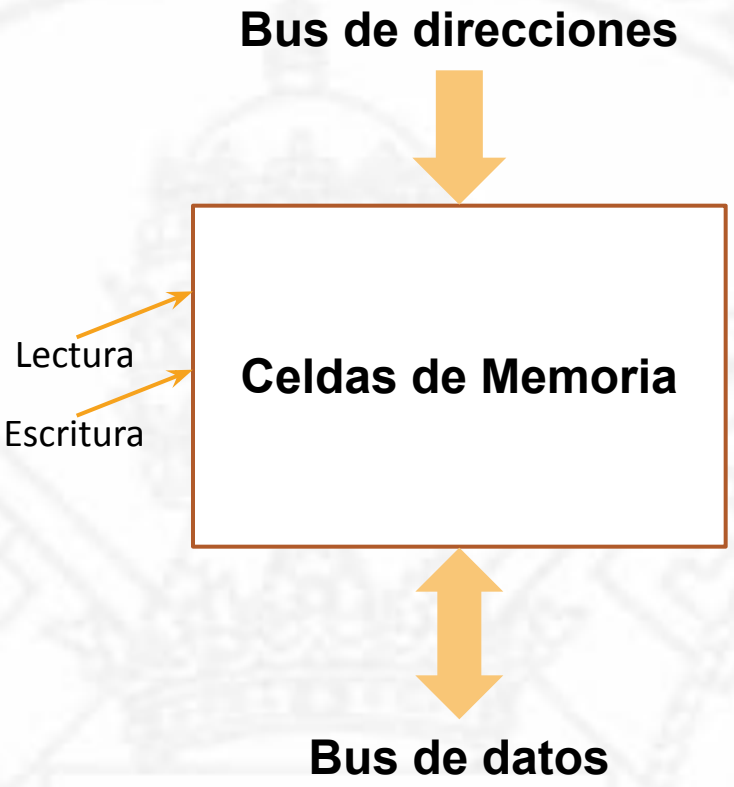
Bus de datos

- En lectura:

  La memoria deposita la información contenida en la dirección solicitada.

- En escritura:

  Se deposita la información a escribir en la dirección solicitada.

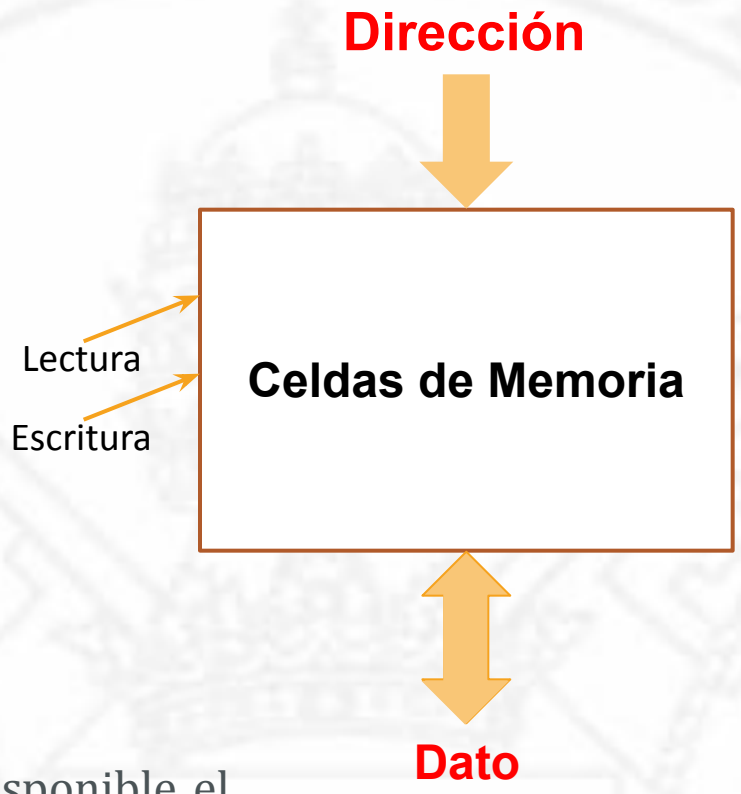


Líneas lógicas de control

Indican tipo de operación a realizar (L/E, entre otras)

Funcionamiento de la Memoria (lectura)

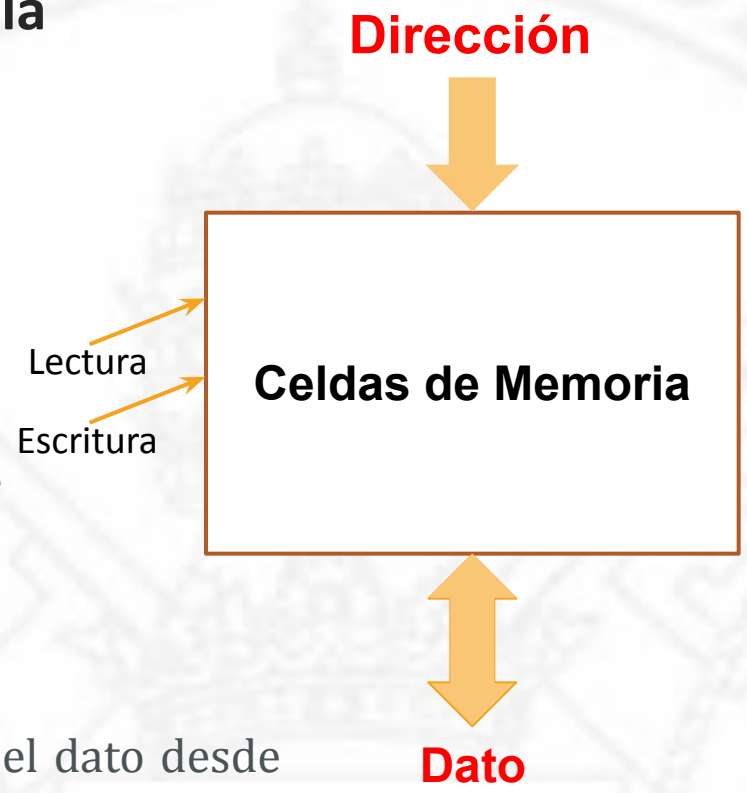
1. **Se deposita en el bus de direcciones la dirección de memoria.**
2. **Se activa la línea de lectura.**
3. **Se deposita en el bus de datos el contenido de la dirección de memoria.**



El tiempo que tarda la memoria en tener disponible el dato desde que se activa la línea de lectura se denomina latencia de la memoria o tiempo de acceso.

Funcionamiento de la Memoria (escritura)

1. **Se deposita en el bus de direcciones la dirección de memoria y en el bus de datos el dato a escribir.**
2. **Se activa la línea de escritura.**
3. **Se escribe en dirección depositada en el bus de direcciones el dato que hay en el bus de datos.**



El tiempo que tarda la memoria en escribir el dato desde que se activa la línea de escritura se denomina latencia de memoria o tiempo de acceso (en escritura)

Funcionamiento del Computador

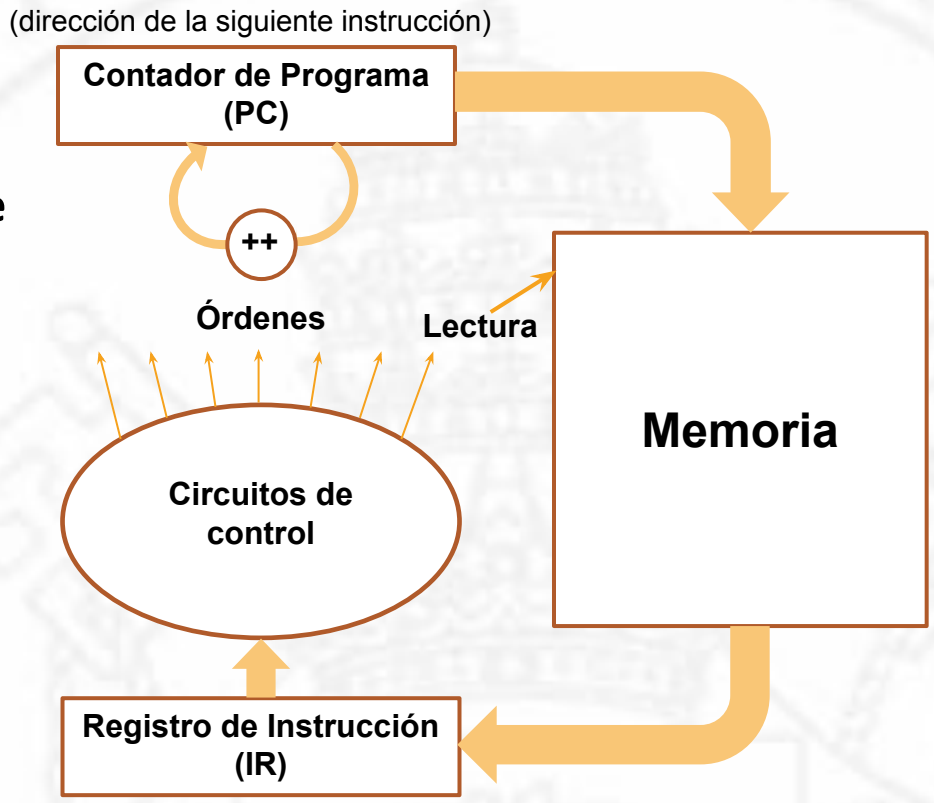
- **Programa en Código Máquina:** secuencia de dígitos binarios que representan a las instrucciones y sus operandos.
- **Ejecución de una Instrucción**
  - Fase de búsqueda:
    - Localización de la instrucción y sus operandos, si los tiene.
  - Fase de ejecución:
    - Ejecución de la instrucción y escritura del resultado (en memoria o registros)
    - Volver a la fase de búsqueda para ejecutar la instrucción siguiente.
- **Registros Importantes**
  - Contador de Programa (PC)
    - Contiene la dirección de la siguiente instrucción a ejecutar.
  - Registro de Instrucción (IR)
    - Contiene la instrucción (no la dirección) en curso.

No figura en el libro

Vídeo (Richard Stallman)

Ejecución de una Instrucción

1. **Se deposita el PC en el bus de direcciones y se activa la orden de búsqueda de la instrucción.**
2. **Se incrementa el PC para que apunte a la instrucción siguiente.**
3. **Se lee la instrucción de Memoria, se deposita en el bus de datos y se almacena en el IR.**
4. **Se decodifica la instrucción almacenada en el IR y se activan las órdenes para su ejecución.**
5. **Se repite el proceso con la instrucción siguiente.**

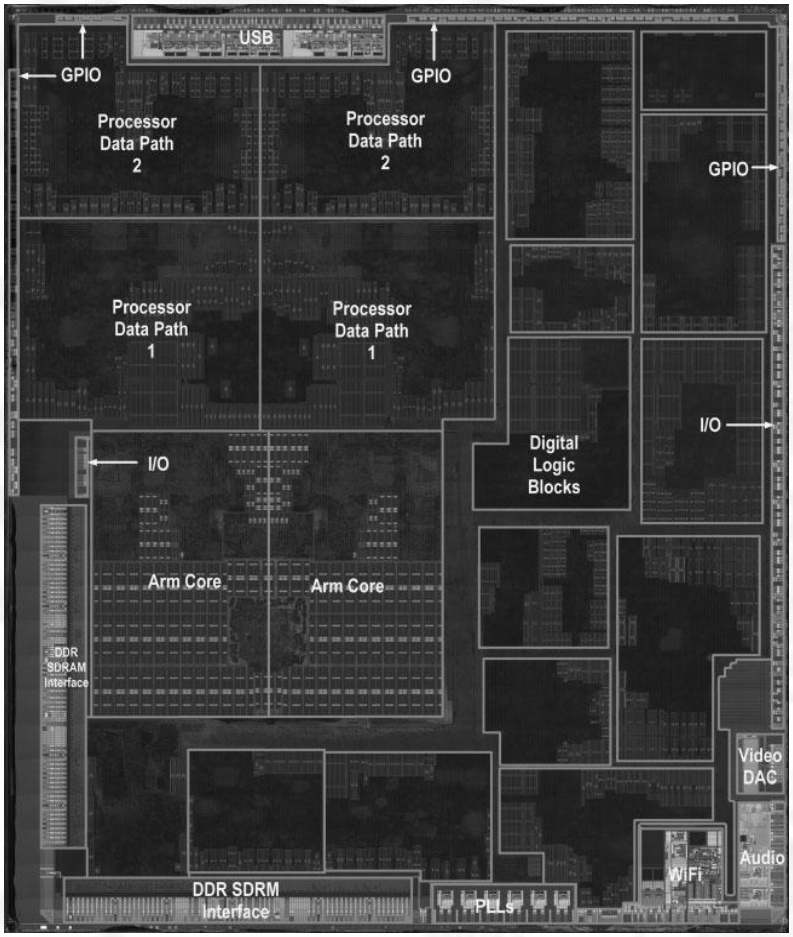


El Procesador (CPU)

- **Cauce (Ruta de Datos)**
  - Ejecuta las operaciones sobre los datos.
- **Control**
  - Establece la secuencia de órdenes necesarias para ejecutar la instrucción en el cauce, acceder a memoria, etc.
- **Memoria Caché**
  - Memoria SRAM pequeña y rápida que permite el acceso de forma inmediata a datos e instrucciones.
  - Varios niveles de caché:
    - El nivel 1 (más cercano a CPU) consta de una caché para datos y otra para instrucciones → Arquitectura Harvard.

Procesador

Apple A5 (tamaño original: 10.1 x 12.1 mm)

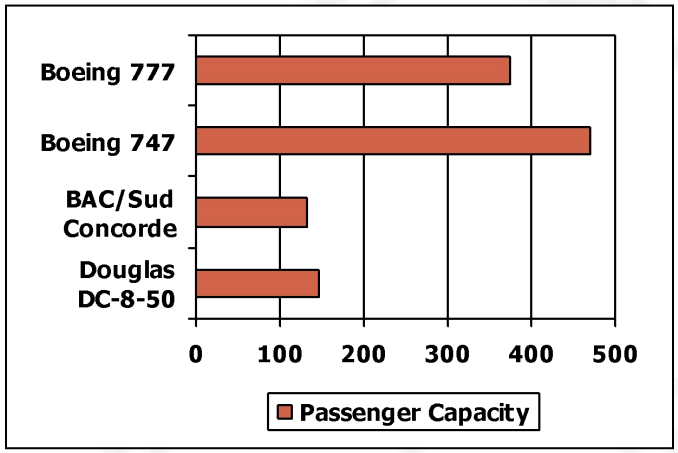


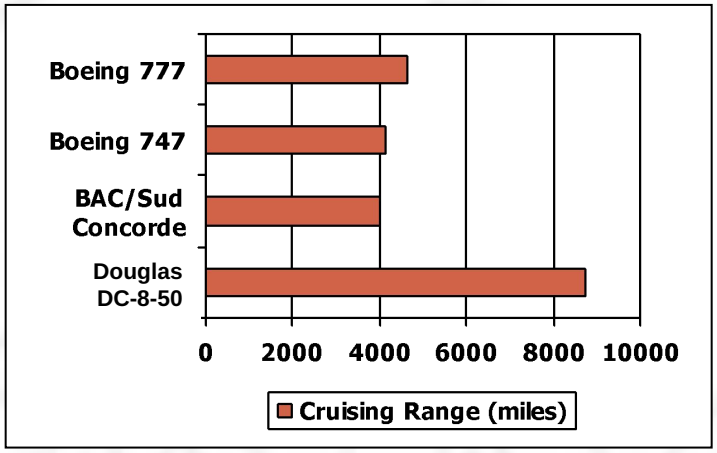
Factores en Rendimiento

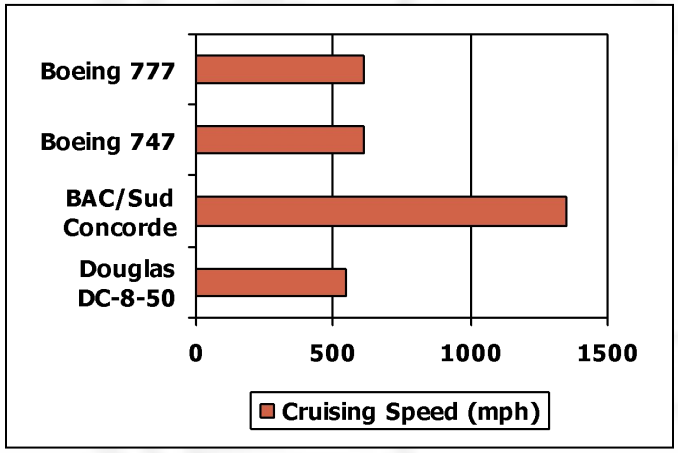
- **Algoritmo**
  - Determina el número de operaciones ejecutadas.
- **Lenguaje de Programación, Compilador y Arquitectura**
  - Determinan el número de instrucciones máquina ejecutadas en cada operación.
- **Procesador y Sistema de Memoria**
  - Determinan lo rápido que se ejecutan las instrucciones.
- **Sistema de E/S, incluyendo el S.O.**
  - Determinan lo rápido que se ejecutan las operaciones de E/S

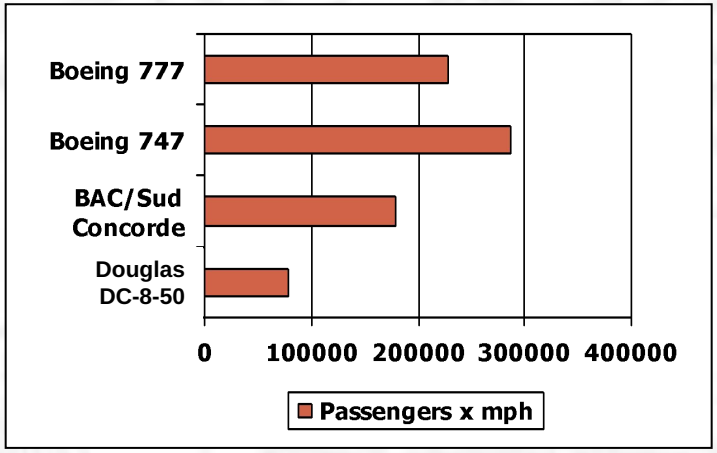
Ejemplos de Rendimiento

¿Qué avión tiene el mejor rendimiento?









Tiempo de Respuesta vs. Productividad

- **Tiempo de Respuesta**
  - Cuánto tiempo lleva terminar una tarea.
- **Productividad**
  - Trabajo total realizado por unidad de tiempo.
    - Por ejemplo: tareas/transacciones/… por hora/por segundo.
- **¿Cómo se ven afectados el tiempo de respuesta y la productividad por...**
  - Reemplazar el procesador por una versión más rápida?
  - Añadir más procesadores?
- **Nos centraremos en el tiempo de respuesta.**

Rendimiento Relativo

$$\text{Rendimiento} = 1/\text{Tiempo de ejecución}$$

- **"X es n veces más rápido que Y"**

$$\text{Rendimiento}_X/\text{Rendimiento}_Y =$$
$$= \text{Tiempo de ejecución}_Y/\text{Tiempo de ejecución}_X =$$
$$= t_Y/t_X = n \quad (\text{X es } n \text{ veces más rápido que Y})$$

Ejemplo: tiempo invertido en ejecutar un programa

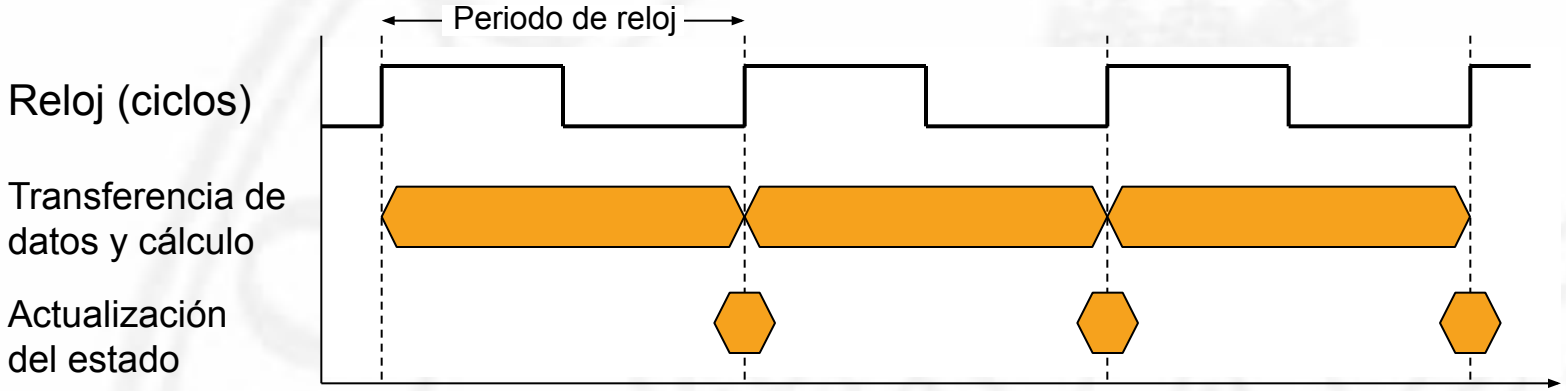
- 10 s en A, 15 s en B
- $t_B / t_A = 15\text{ s} / 10\text{ s} = 1.5$
- A es 1.5 veces más rápido que B.

Medida del Tiempo de Ejecución

- **Tiempo Invertido (*elapsed time*)**
  - Tiempo de respuesta total, incluyendo el de procesamiento de CPU, operaciones de E/S y tareas del Sistema Operativo
  - Determina el rendimiento del sistema.
- **Tiempo de CPU (*runtime*)**
  - Tiempo consumido procesando una determinada tarea.
    - Se descuenta E/S y otras tareas con recursos compartidos.
  - Incluye el tiempo de CPU de usuario y el empleado por el S.O
  - Distintos programas se ven afectados de forma diferente por los rendimientos de la CPU y del sistema.

Temporización de la CPU

El funcionamiento del hardware está gobernado por un reloj de frecuencia constante



- Periodo de reloj (T): duración de un ciclo de reloj
  - Por ejemplo, $250\ \text{ps} = 0.25\ \text{ns} = 250\times10^{-12}\ \text{s}$
- Frecuencia de reloj ($f=1/T$): ciclos por segundo
  - Por ejemplo, $4.0\ \text{GHz} = 4000\ \text{MHz} = 4.0\times10^{9}\ \text{Hz}$

Tiempo de CPU

$$t = \text{Tiempo de CPU} =$$
$$= \text{Número de ciclos} \times \text{Duración del ciclo} = N \times T =$$
$$= \frac{\text{Número de ciclos}}{\text{Frecuencia de reloj}} = \frac{N}{f}$$

- El rendimiento mejora cuando:
  - Disminuye el número de ciclos de reloj.
  - Aumenta la frecuencia de reloj.
- El diseñador de hardware suele establecer un compromiso entre la frecuencia de reloj y el número de ciclos.

Ejemplo de Tiempo de CPU

- **Computador *A* :** reloj de 2 GHz , tiempo de CPU = 10 s
- **Computador *B* :**
  - Se pretende conseguir un tiempo de CPU = 6 s
  - Se puede acelerar el reloj, pero causa 1.2 × número de ciclos
- **¿Cuál debe ser la frecuencia de reloj de B?**

$$t_B = N_B \times T_B = \frac{N_B}{f_B} \Rightarrow f_B = \frac{N_B}{t_B} = \frac{1.2 \times N_A}{6\ \text{s}}$$

$$t_A = N_A \times T_A = \frac{N_A}{f_A} \Rightarrow N_A = t_A \times f_A = 10\ \text{s} \times 2\ \text{GHz} = 20 \times 10^9\ \text{ciclos}$$

$$f_B = \frac{1.2 \times N_A\ \text{ciclos}}{6\ \text{s}} = \frac{1.2 \times 20 \times 10^9\ \text{ciclos}}{6\ \text{s}} = \frac{24 \times 10^9\ \text{ciclos}}{6\ \text{s}} = 4\ \text{GHz}$$

Número de Instrucciones y CPI

$$N = \text{Número de ciclos} =$$
$$= \text{Número de instrucciones} \times \text{Ciclos por instrucción} = Ic \times CPI$$
$$t = N \times T = Ic \times CPI \times T = \frac{Ic \times CPI}{f}$$

- ***Ic*: Número de Instrucciones del Programa**
  - Determinado por el programa, el ISA y el compilador.
- **CPI: Ciclos por Instrucción**
  - Determinado por el hardware de la CPU.
  - Si distintas instrucciones tienen CPI distintos, el CPI total estará afectado por la combinación de instrucciones

Ejemplo de CPI

- Computador A: Tiempo de ciclo = 250 ps, CPI = 2.0
- Computador B: Tiempo de ciclo = 500 ps, CPI = 1.2
- Misma ISA (mismo número de instrucciones)

- **¿Cuál es más rápido y cuánto más?**

$$t_A = Ic \times CPI_A \times T_A =$$
$$= Ic \times 2.0 \times 250 \text{ ps} = Ic \times 500 \text{ ps}$$
$$t_B = Ic \times CPI_B \times T_B =$$
$$= Ic \times 1.2 \times 500 \text{ ps} = Ic \times 600 \text{ ps}$$

A es más rápido…

…un 20% más:

$$\frac{\text{Rendimiento A}}{\text{Rendimiento B}} = \frac{t_B}{t_A} = \frac{Ic \times 600 \text{ ps}}{Ic \times 500 \text{ ps}} = 1.2$$

Ganancia de velocidad, aceleración o *speed up*

CPI (con más detalle)

- **Si se tienen diferentes tipos de instrucciones (n) con distinto número de ciclos:**

$$N = \sum_{i=1}^{n} (CPI_i \times Ic_i)$$

CPI: media ponderada.

$$CPI = \frac{N}{Ic} = \frac{1}{Ic} \sum_{i=1}^{n} (CPI_i \times Ic_i) = \sum_{i=1}^{n} \left(CPI_i \times \frac{Ic_i}{Ic} \right)$$

Frecuencia relativa de instrucciones de tipo i

Ejemplo de CPI

- **Diferentes secuencias de código compilado utilizan instrucciones de tipos A, B y C**

Tipo	A	B	C
CPI para ese tipo	1	2	3
Ic (nº) en la secuencia 1	2	1	2
Ic (nº) en la secuencia 2	4	1	1

- Secuencia 1: $Ic = 5$

  $\mathrm{N} = 2\times1 + 1\times2 + 2\times3 = 10$

  $\mathrm{CPI} = 10/5 = 2.0$

- Secuencia 2: $Ic = 6$

  $\mathrm{N} = 4\times1 + 1\times2 + 1\times3 = 9$

  $\mathrm{CPI} = 9/6 = 1.5$

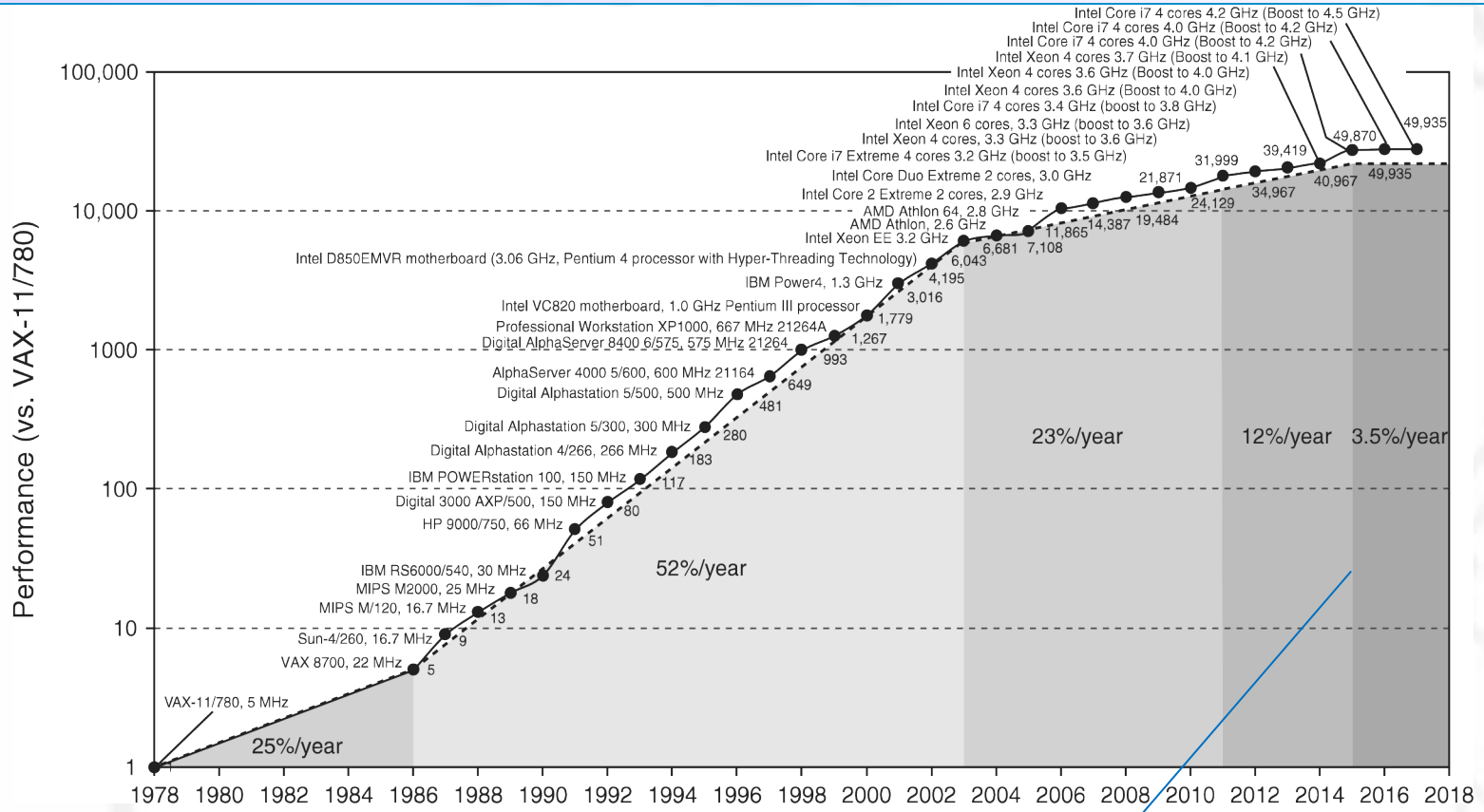
Resumen sobre Rendimiento

- **El rendimiento depende de ...**
  - Algoritmo: afecta a Ic y, posiblemente, a CPI.
  - Lenguaje de Programación: afecta a Ic y CPI.
  - Compilador: afecta a Ic y CPI.
  - ISA: afecta a Ic, CPI y T.
  - Implementación de la ISA: CPI y T.

$$t = \frac{\text{Instrucciones}}{\text{Programa}} \times \frac{\text{Ciclos}}{\text{Instrucción}} \times \frac{\text{Tiempo}}{\text{Ciclo de reloj}}$$

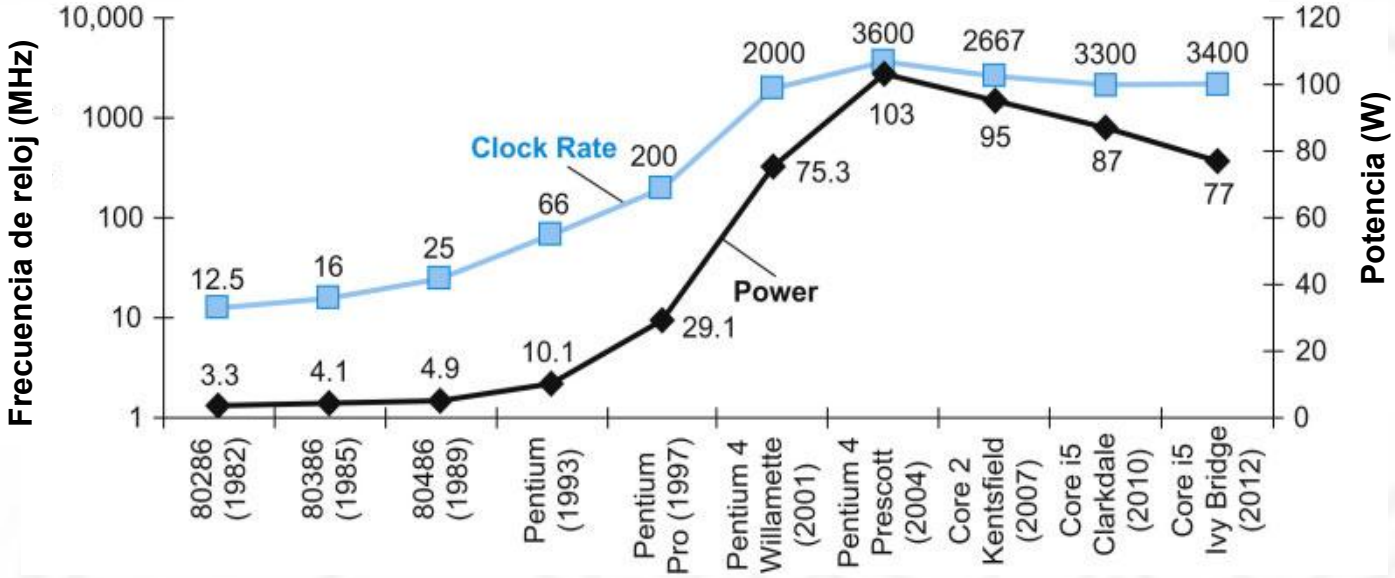
$$t = Ic \times CPI \times T = \frac{Ic \times CPI}{f}$$

Rendimiento Monoprocesador



Limitado por la potencia, el paralelismo a nivel de instrucciones y la latencia de memoria.

Tendencia en la Potencia



Factor de actividad Carga capacitiva Voltaje Frecuencia

Potencia Dinámica

$$W_{din} = \alpha\, C V^2 f$$

Corrientes de fuga

Potencia Estática

$$W_{st} = V\, I_{fuga}$$

Reducción de Potencia

- **Supongamos que una nueva CPU tiene:**
  - 85 % de la carga capacitiva de la antigua CPU.
  - Reducción de 20 % en voltaje y 10 % en frecuencia.

$$\frac{W_{nueva}}{W_{anterior}} = \frac{\alpha\, C_{ant} \times 0.85 \times (V_{ant} \times 0.80)^2 \times f_{ant} \times 0.90}{\alpha\, C_{ant} V_{ant}^2 f_{ant}} =$$

$$= 0.85 \times 0.8^2 \times 0.9 = 0{,}49$$

- Muro de la potencia:
  - No se puede reducir más el voltaje.
  - No se puede eliminar más calor.
- ¿Cómo podemos así mejorar el rendimiento?

Multiprocesadores

- **Procesadores Multinúcleo**
  - Más de un núcleo procesador (*core*) en cada chip.
- **Requieren programación paralela:**
  - Paralelismo de Instrucciones
    - El hardware ejecuta varias instrucciones a la vez.
  - No es fácil
    - Programar obteniendo buen rendimiento.
    - Equilibrar la carga entre los cores.
    - Optimizar comunicación y sincronización.

Aceleración y Eficiencia

- **La mejora del rendimiento en los multiprocesadores se mide mediante la aceleración (o *speed up*):**

$$S(N) = \frac{\text{Rendimiento para } N \text{ procesadores}}{\text{Rendimiento para 1 procesador}} = \frac{t_1}{t_N}$$

- **La eficiencia mide la mejora respecto al aumento de recursos, en este caso procesadores, es decir:**

$$E(N) = \frac{S(N)}{N} = \frac{t_1/t_N}{N} = \frac{t_1}{Nt_N}$$

Ejemplo

- Un programa que en un solo procesador se ejecuta en 320 ms, en 4 procesadores se ejecuta en 100 ms ¿Cuál ha sido la aceleración y la eficiencia?
  - Aceleración: $S(N) = \frac{t_1}{t_N} = \frac{320}{100} = 3{,}2$
  - Eficiencia: $E(N) = \frac{S(N)}{N} = \frac{3{,}2}{4} = 0{,}8 \quad (80\%)$

En general, la eficiencia disminuye con el número de procesadores porque aumentan las necesidades de comunicación y sincronización.

Enteros sin Signo (Binario Natural)

- **Un número x se representa mediante la secuencia de bits $x_{n-1}x_{n-2}\ldots x_1x_0$ para la que:**

$$x = x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \ldots + x_1 2^1 + x_0 2^0 = \sum_{i=0}^{n-1} x_i 2^i$$

  - Ejemplo:

$$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1011_2 =$$

$$= 0 + \ldots + 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 1\times 2^0 =$$

$$= 0 + \ldots + 8 + 0 + 2 + 1 = 11_{(10}$$

- **Rango representable:** de 0 a $+(2^n - 1)$
  - Con 32 bits: de 0 a +4.294.967.295
  - Inconveniente: solo sirve para representar no negativos.

Enteros con Signo en Binario Signado o Signo-Magnitud

- **El signo se representa mediante un bit. El resto de bits representan el módulo. Un número x se representa mediante:**

$$x = (-1)^{x_{n-1}} \times (x_{n-2}2^{n-2} + \ldots + x_1 2^1 + x_0 2^0)$$

- Ejemplo: $1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1011_{(2} =$

$$= (-1)^1 \times (0 + \ldots + 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 1\times 2^0) =$$

$$= -(0 + \ldots + 8 + 0 + 2 + 1) = -11_{(10}$$

- **Rango representable**: desde $-(2^{n-1}-1)$ a $+(2^{n-1}-1)$
- **Inconvenientes:**
  - Hay dos representaciones para el 0 (+0 y -0)
  - Hay que procesar el signo antes y después de cada operación.

No se emplea para representar números enteros.

Enteros con Signo en Complemento a 2 (I)

- **Un número x se representa mediante:**

$$x = -x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \ldots + x_1 2^1 + x_0 2^0$$

- Ejemplo:

$$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1100_2 =$$
$$= -1\times2^{31} + 1\times2^{30} + \ldots + 1\times2^2 + 0\times2^1 + 0\times2^0 =$$
$$= -2.147.483.648 + 2.147.483.644 = -4_{(10}$$

- **Rango representable:** de -2^{n-1} a $+(2^{n-1}-1)$
  - Con 32 bits: de –2.147.483.648 a 2.147.483.647

Enteros con Signo en Complemento a 2 (II)

- **El bit de más peso es el bit de signo:**
  - 1 para números negativos.
  - 0 para números no negativos.
  - $-(-2^{n-1})$ no se puede representar.
- **Los números no negativos se representan de la misma forma que en binario natural**
- **Algunos números concretos:**
  - 0: **0000 0000…0000**
  - –1: **1111 1111…1111**
  - Número más negativo: **1000 0000…0000**
  - Número más positivo: **0111 1111…1111**

Cálculo del Opuesto en Complemento a 2

- **Complementar y Sumar 1**
  - Complementar significa $1 \rightarrow 0,\ 0 \rightarrow 1$
- **Demostración:**

$$x + \bar{x} = 1111\ldots111_{(2} = -1 \Rightarrow \bar{x} + 1 = -x$$

  - Ejemplo: calcular el opuesto de +2
    - $+2 = 0000\ 0000\ \ldots\ 0010_{(2}$
    - $-2 = 1111\ 1111\ \ldots\ 1101_{(2} + 1 =$
      $= 1111\ 1111\ \ldots\ 1110_{(2}$

Extensión de Signo

- **Representación de un número con más bits**
  - Debe preservar el valor numérico con su signo
- **Se rellena por la izquierda con el bit de signo**
  - De la misma forma que en binario natural se rellena con 0's
- **Ejemplo:** extensión de 8 a 16 bits:
  - +2: 0000 0010 → 0000 0000 0000 0010
  - – 2: 1111 1110 → 1111 1111 1111 1110

Hexadecimal

- **Base 16**
  - Es una representación compacta de cadenas de bits.
  - 4 bits por cada dígito hexadecimal:

0	**0000**	4	**0100**	8	**1000**	c	**1100**
1	**0001**	5	**0101**	9	**1001**	d	**1101**
2	**0010**	6	**0110**	a	**1010**	e	**1110**
3	**0011**	7	**0111**	b	**1011**	f	**1111**

  - Ejemplo: eca8 6420

e	c	a	8	6	4	2	0
1110	1100	1010	1000	0110	0100	0010	0000

Representación de Caracteres

- **Juegos de caracteres codificados en bytes.**
  - ASCII: 128 caracteres
    - 95 visibles, 33 de control
  - Latin-1: 256 caracteres
    - ASCII y 96 caracteres visibles más
- **Unicode:** juego de caracteres de 32 bits
  - Se usa en Java, C++, etc.
  - Incluye la mayoría de los alfabetos del mundo y símbolos
  - UTF-8 y UTF-16: son implementaciones del Unicode con codificaciones de longitud variable.

Código ASCII

- Surgió en los años 60 para el intercambio de información entre computadores diferentes.
- Originalmente tenía solo 7 bits, ya que se empleaba un bit para el control de errores en la transmisión.
- Posteriormente se extendió a 8 bits con variantes para diferentes idiomas.
- El código de 7 bits está estandarizado y es compatible con codificaciones de caracteres más completas, como el UTF-8, que tiene longitud variable entre 1 y 4 bytes para cada carácter.

Código ASCII de 7 bits

- **0 a 31 (`0x00` a `0x1f`): caracteres de control**
- **32 (`0x20`): espacio**
- **33 a 47 (`0x21` a `0x2f`): caracteres especiales**
- **48 a 57 (`0x30` a `0x39`): cifras**
- **58 a 63 (`0x3a` a `0x3f`): especiales**
- **64 a 90 (`0x40` a `0x5a`): mayúsculas**
- **91 a 96 (`0x5b` a `0x60`): especiales**
- **97 a 122 (`0x61` a `0x7a`): minúsculas**
- **123 a 126 (`0x7b` a `0x7e`): especiales**
- **127 (`0x7f`): *delete* (control)**

Dígito hexadecimal menos ignificativo \ Dígito hexadecimal más ignificativo	0	1	2	3	4	5	6	7
0	NUL	DEL	Space	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
a	LF	SUB	*	:	J	Z	j	z
b	VT	ESC	+	;	K	[	k	{
c	FF	FS	,	<	L	\	l	|
d	CR	GS	-	=	M	]	m	}
e	SO	RS	.	>	N	^	n	~
f	SI	US	/	?	O	_	o	DEL

Conclusiones

- **Se establecen niveles jerárquicos de abstracción**
  - Tanto para el hardware como el software.
- **La arquitectura del repertorio de instrucciones (ISA)**
  - Es la interfaz entre hardware y software.
- **Está mejorando la relación coste/rendimiento**
  - Debido al desarrollo de la tecnología subyacente.
- **La potencia es un factor limitador**
  - Uso de paralelismo para mejorar el rendimiento.
- **Las secuencias de bits no tienen significado por sí mismas**
  - Su interpretación depende del contexto: la misma secuencia de bits puede representar un número, un carácter, etc.