

Introducción

Conceptos básicos de computadores

Katia Leal Algara, URJC

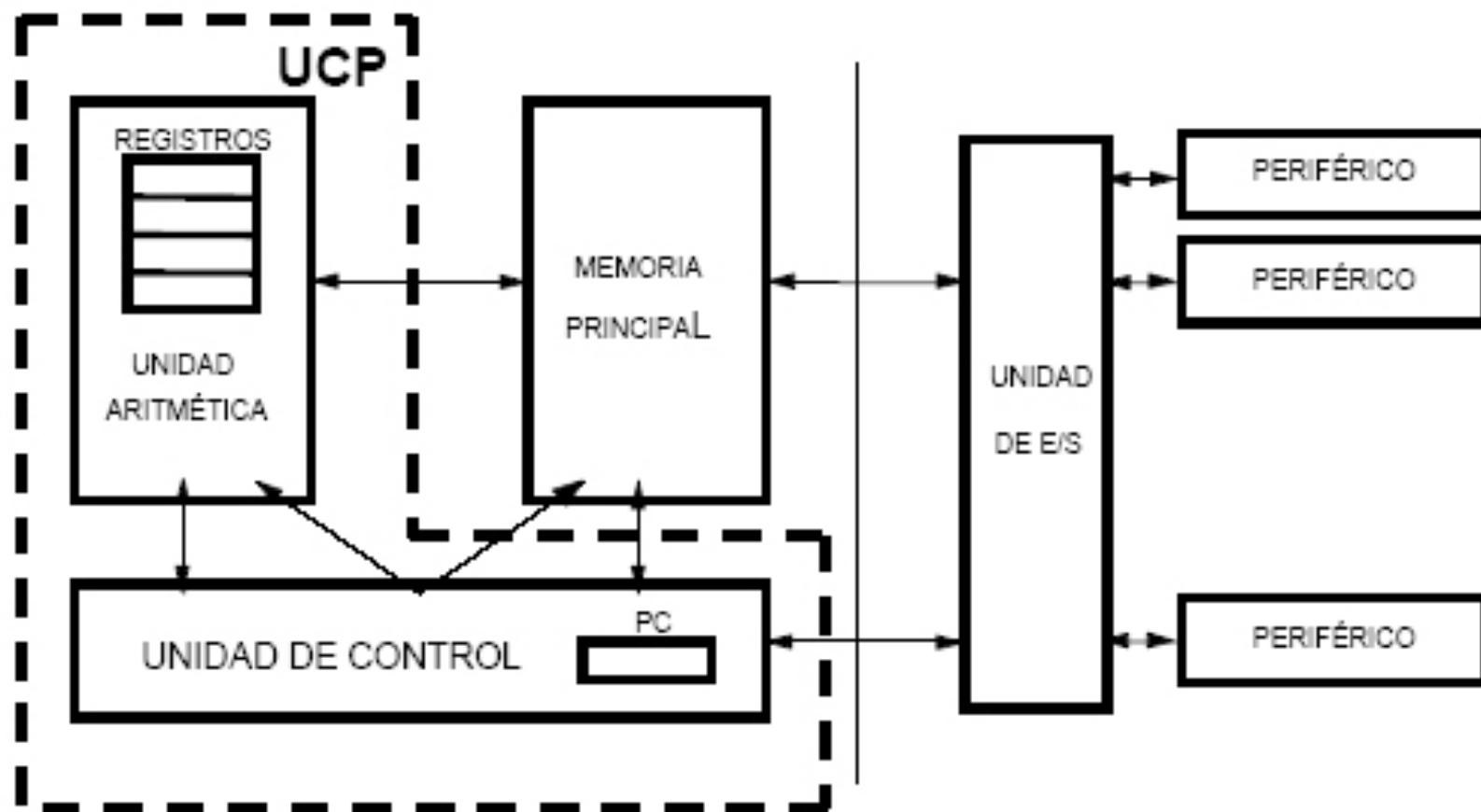
Juan González Gómez, URJC

Arquitectura de Computadores

- La *arquitectura del computador* hace referencia a todos aquellos elementos del sistema visibles al programador que tienen impacto directo en la ejecución lógica de un programa.
- Kai Hwang & Fayé Briggs: “es, en realidad un concepto del sistema que integra hardware, software, algoritmos y lenguajes para realizar grandes cálculos”

Componentes de un ordenador

- Arquitectura Von Neuman & Eckert-Mauchly,
*First Draft of a Report on the EDVAC, 30 Junio
de 1945*



Contextualización

- **Taxonomía de Flynn:** clasificación de arquitecturas de computadores propuesta por Michael J. Flynn en 1972
 - **SISD:** una instrucción, un dato
 - **SIMD:** una instrucción, múltiples datos
 - **MISD:** múltiples instrucciones, un solo dato
 - **MIMD:** múltiples instrucciones, múltiples datos

Clases de computadores

■ Ordenadores personales

- Propósito general
- Sujetos a un compromiso entre coste y rendimiento

■ Servidores

- Basados en la red
- Alta capacidad, rendimiento, fiabilidad
- Desde servidores pequeños hasta del tamaño de un edificio

Clases de computadores

■ Supercomputadores

- Cálculos científicos y de ingeniería de alta gama
- Máxima capacidad pero representa una pequeña fracción del mercado informático general

■ Sistemas empotrados

- Oculto como componentes de sistemas
- Restricciones estrictas de potencia / rendimiento / coste

Clases de computadores

■ Sistemas robóticos

- Sensores / Actuadores
- Los más avanzados están basados en microprocesadores

Paralelismo

■ Paralelismo interno

- **SISD**: arquitectura Von Neumann con un solo procesador, CPU
 - Segmentación (*pipelining*) de funciones: sin replicar hardware!

■ Paralelismo explícito

- **SIMD**: una única unidad de control que gobierna varias ALUs
- **MIMD**: varios procesadores que manejan, cada uno, un flujo de instrucciones sobre un flujo de datos

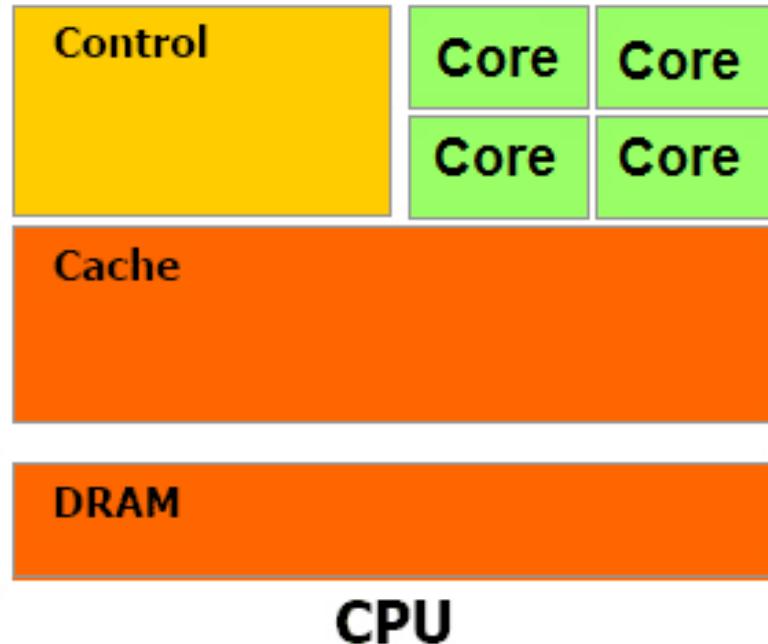
Paralelismo: MIMD

- Las máquinas MIMD tienen un número de procesadores que funcionan de manera asíncrona e independiente para lograr **paralelismo puro**
- En cualquier momento, cualquier procesador puede ejecutar diferentes instrucciones sobre distintos datos
- Pueden tener memoria compartida o distribuida

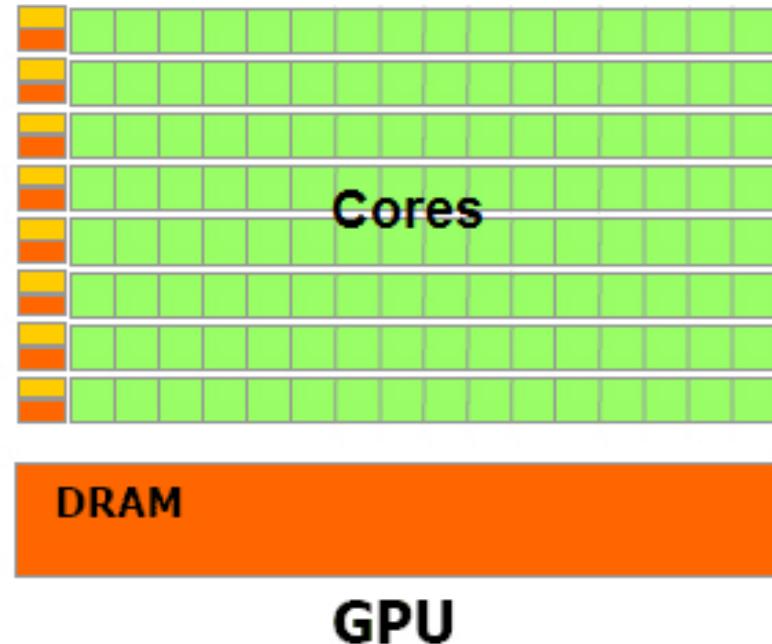
Paralelismo: SIMD

- **CPU + juego extendido de instrucciones**, como el repertorio AVX (Advanced Vector Extensions, 2008) de Intel, que se utiliza para álgebra lineal: suma de vectores, multiplicación escalar, multiplicación de matrices, etc.
- **Basic Linear Algebra Subprograms (BLAS)**: es una especificación (interfaz) que define un conjunto de rutinas de bajo nivel para realizar operaciones comunes de álgebra lineal. Este conjunto de rutinas suponen el *estándar de facto* para las librerías de álgebra lineal.
 - Ejemplos de implementación de la librería BLAS: AMD Core Math Library (**ACML**), **ATLAS**, Intel Math Kernel Library (**MKL**) y **OpenBLAS**.
- **GPU**:
 - 2007, gran salto a la computación de propósito general gracias a la primera distribución de Nvidia CUDA
 - 2009, Big Bang del Deep Learning

CPU Vs GPU



Von Neumann, *multicore*



SIMD, *manycore*

CPUs: diseño orientado a la latencia

- **Arquitectura Von Neumann:** frecuencia de procesador alta
- **Cachés grandes:** reducen el tiempo medio de acceso a la jerarquía de memoria
- **Control sofisticado:**
 - *Predicción de salto* para reducir la latencia por riesgos de control
 - *Adelantamiento de datos* para reducir la latencia por riesgos de datos
- **ALU potente:**
 - Operaciones con poca latencia

GPUs: diseño orientado al *throughput*

- **Arquitectura SIMD:** frecuencia de procesador moderada
- **Cachés pequeñas:** para impulsar el *throughput* de memoria
- **Control sencillo:**
 - No *predicción de salto*
 - No *adelantamiento de datos*
- **ALUs energéticamente eficientes:**
 - Muchas unidades con latencia alta, cauce largo para incrementar el *throughput*
 - Requiere una cantidad masiva de hilos para “ocultar” la latencia

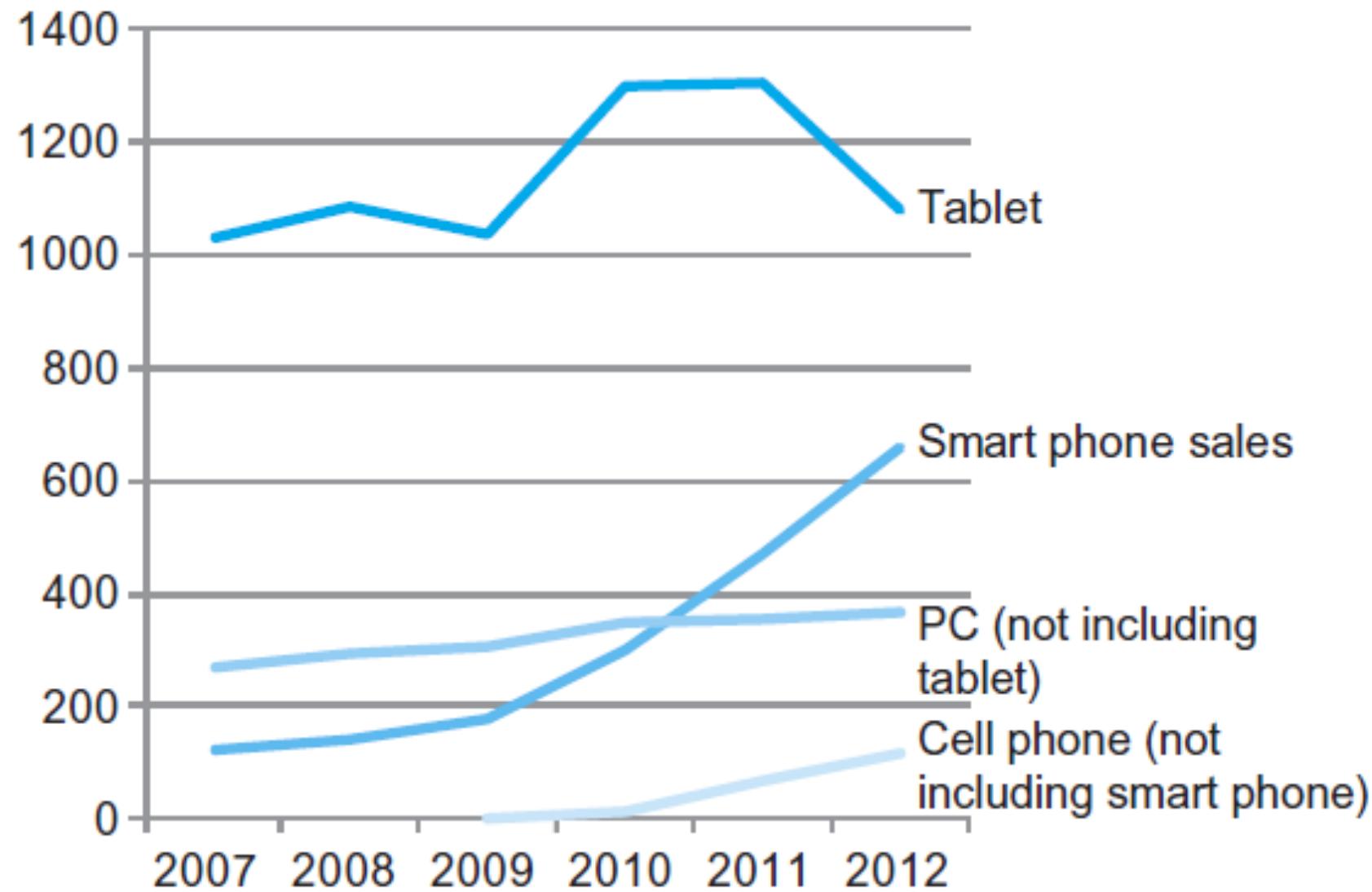
CPUs y GPUs son complementarias

- **CPUs:** para la partes secuenciales en las que prima la latencia
 - Las CPUs pueden ser órdenes de magnitud más rápidas que las GPUs para código secuencial
- **GPUs:** para las partes paralelas en las que gana el throughput
 - Las GPUs pueden ser órdenes de magnitud más rápidas que las CPUs para código paralelo

Problemas adecuados para GPUs

1. Requerir de cálculos complejos sobre grandes volúmenes de datos
2. Presentar alta intensidad aritmética (ratio operaciones ALU Vs memoria)
3. Permitir un paralelismo significativo
4. Ser significativamente más dependiente del *throughput* general que de la latencia de cada operación

La era PostPC



Objetivos

- ¿Cómo un programa escrito en un **lenguaje de alto nivel**, como C o Java, se traduce a **lenguaje máquina** y cómo el hardware ejecuta el programa resultante? Rendimiento de los programas
- ¿Cuál es la interfaz hardware / software y cómo el software indica al hardware la realización de las tareas necesarias?
- ¿Qué determina el rendimiento de un programa y cómo puede un programa mejorar su rendimiento? Esto dependerá de cómo esté escrito el programa, de su traducción a lenguaje máquina y de su ejecución por parte del hardware
- ¿Qué técnicas pueden aplicar los diseñadores de hardware para mejorar el rendimiento? Conceptos básicos de diseño de computadores
- ¿Cuáles son los motivos y las consecuencias del cambio de **procesamiento secuencial** al **procesamiento paralelo**? Mecanismos hardware que soportan el paralelismo

Objetivos

- La respuesta a estas preguntas nos permitirán:
 - Mejorar el rendimiento de un programa
 - Determinar la máquina con las características más adecuadas para ejecutar una determinada aplicación
- En este primer capítulo se presentan:
 - Ideas básicas y definiciones
 - Componentes hardware y software
 - Evaluación del rendimiento

Rendimiento

- Si medimos el rendimiento de un sistema en MIPS (millones de instrucciones por segundo), se tiene:
MIPS = (instrucciones/ciclo) x (ciclos/seg) x 10⁻⁶
- (instrucciones/ciclo), depende de la arquitectura
- (ciclos/segundo), depende de la tecnología utilizada

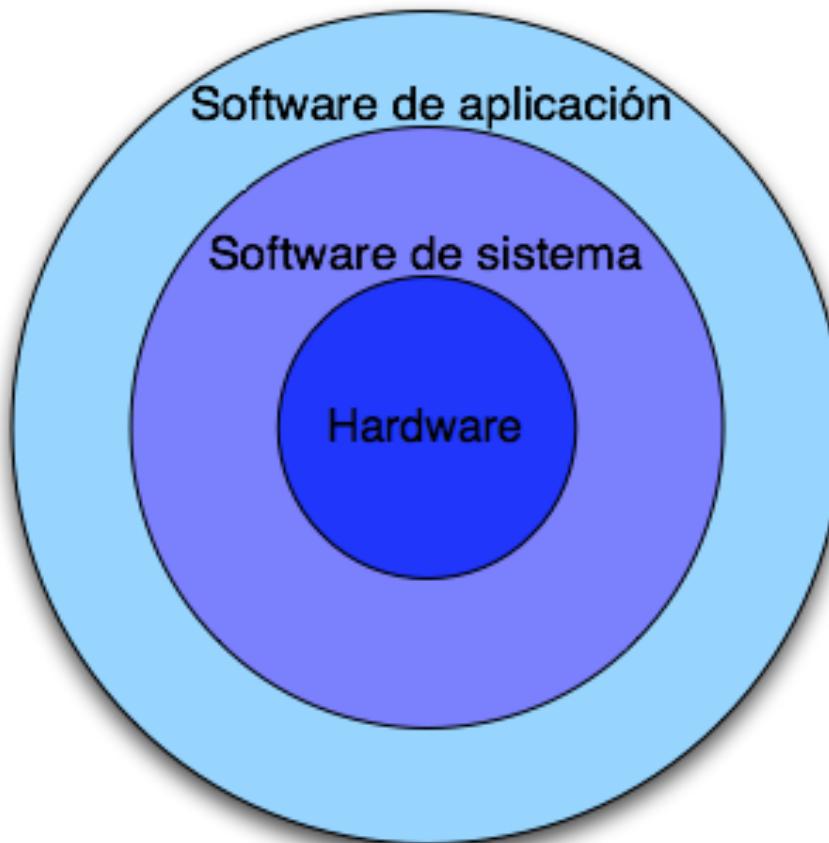
Rendimiento

- **Algoritmo**
 - Determina el número de *operaciones* a ser ejecutadas
- **Lenguaje de programación, compilador, arquitectura**
 - Determinan el número de *instrucciones* máquina ejecutadas por operación
- **Procesador y sistema de memoria**
 - Determinan cómo de rápido se ejecutan las instrucciones
- **Sistema de E / S (incluido el sistema operativo)**
 - Determina cómo de rápido se sirven las operaciones de E / S

Rendimiento

Componente hardware o software	Cómo afecta éste componente al rendimiento
Algoritmo	Determina el número de sentencias y el número de operaciones de E/S
Lenguaje de programación, compilador y arquitectura	Determina el número de instrucciones de computador para cada sentencia
Procesador y sistema de memoria	Determinan cómo de rápido se ejecutan las instrucciones
Sistema de E/S (hardware y SO)	Determina cómo de rápido se sirven las operaciones de E/S

Abstracción: simplifica el diseño



Software de Sistema

■ Sistema operativo

- Gestión básica de las operaciones de E/S
- Asignación de almacenamiento y memoria
- Reparto seguro de los recursos del sistema entre múltiples aplicaciones que se ejecutan de forma simultánea
- Ejemplos SSOO: Linux, MacOS, Windows, Plan9

■ Compiladores

- Un programa que traduce un programa escrito en un lenguaje de alto nivel (C, C++, Java) en instrucciones que el hardware puede ejecutar
- Dada la sofisticación de los actuales lenguajes de programación, la traducción de un programa en lenguaje de alto nivel a lenguaje máquina es compleja

Del lenguaje de alto nivel al lenguaje máquina

Lenguaje de alto nivel

- Nivel de abstracción más cercano al dominio del problema
- Proporciona productividad y portabilidad

High-level
language
program
(in C)

```
swap(int v[], int k)
{int temp;
 temp = v[k];
 v[k] = v[k+1];
 v[k+1] = temp;
}
```

Compiler

Assembly
language
program
(for RISC-V)

```
swap:
slli x6, x11, 3
add x6, x10, x6
ld x5, 0(x6)
ld x7, 8(x6)
sd x7, 0(x6)
sd x5, 8(x6)
jalr x0, 0(x1)
```

Assembler

Binary machine
language
program
(for RISC-V)

```
00000000001101011001001100010011
000000000011001010000001100110011
0000000000000000110011001010000011
0000000001000001100110011001110000011
00000000011100110011000000100011
00000000010100110011010000100011
000000000000000010000000001100111
```

Lenguaje ensamblador

- Representación textual de instrucciones

Lenguaje máquina

- Dígitos binarios (bits)
- Instrucciones y datos codificados

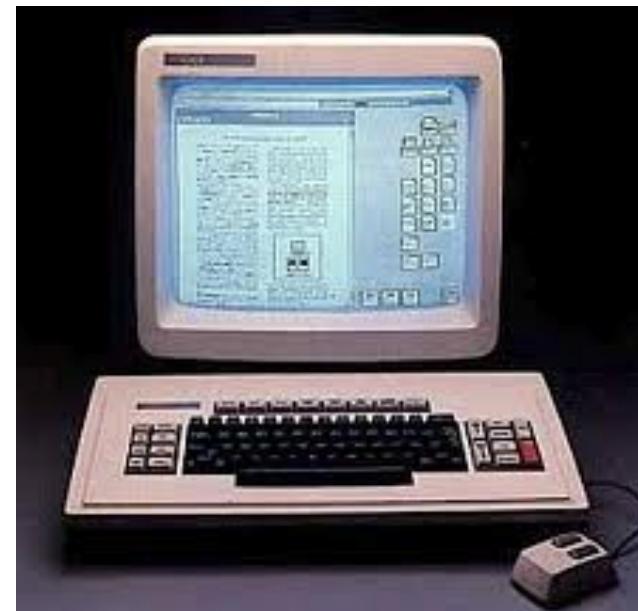
Interfaz Hardware - Software

- ISA (Instruction Set Architecture) o arquitectura, de un computador es la interfaz entre el hardware y el software de más bajo nivel
 - Incluye cualquier cosa que los programadores necesiten saber para hacer que un programa en lenguaje máquina funcione correctamente: instrucciones, dispositivos de E/S, registros, acceso a memoria ...
- El SO encapsula los detalles de la E/S, de la asignación de memoria y del resto de funciones de bajo nivel
- ... así, el principio de **abstracción** es el que permite el desarrollo de sistemas complejos basándose en la ocultación de los detalles de bajo nivel entre niveles sucesivos de una jerarquía

Componentes de un computador

- Todo hardware realiza las mismas funciones básicas
 - **Entrada de datos**
 - **Salida de datos**
 - **Procesamiento de datos**
 - **Almacenamiento de datos**
- Los cinco componentes clásicos de un computador son
 - **Entrada**
 - **Salida**
 - **Ruta de datos y de control o procesador**
 - **Memoria**
- Cualquier componente de cualquier ordenador (pasado o presente) se puede clasificar en una de estas cinco categorías con independencia de la tecnología utilizada

Componentes de un computador



Componentes de un computador

- La **placa base** contiene paquetes de circuitos integrados o **chips**. Tiene tres partes:
 - La que conecta con los dispositivos de E/S
 - La memoria
 - El procesador
- **Memoria**: programas + datos que necesitan los programas. Normalmente construida a partir de pastillas DRAM (Dynamic Random Access Memory)
- **Procesador** o CPU: es la parte activa de la placa base. El procesador contiene:
 - La **ruta de datos**, encargada de realizar las operaciones aritméticas (músculo)
 - La **ruta de control**: que indica a la ruta de datos, a la memoria y a los dispositivos de E/S qué hacer de acuerdo a las instrucciones del programa (cerebro)
- **Memoria caché**: es un tipo de memoria más pequeña y rápida que actúa como un buffer de la memoria DRAM. Tecnología SRAM (Static Random Access Memory)

Almacenamiento de datos

- Memoria principal volátil
 - La información desaparece cuando se apaga
- Memoria secundaria no volátil
 - Discos magnéticos
 - Memoria flash
 - Discos ópticos (CDROM, DVD)



Tiempo de respuesta Vs Throughput

■ Tiempo de respuesta

- Tiempo que tarda en realizarse una tarea

■ *Throughput*

- Cantidad de trabajo completado por unidad de tiempo

■ Ejemplo: indicar si los siguientes cambios en un ordenador incrementan el *throughput*, decrementan el tiempo de respuesta o ambos

1. Reemplazar el procesador por uno más rápido
2. Añadir más procesadores a un sistema multiprocesador para la ejecución de múltiples tareas

Midiendo el rendimiento

- Como usuarios nos interesa el tiempo de ejecución
- Como diseñadores nos interesa lo rápido que es el hardware a la hora de ejecutar funciones básicas
- Todos los computadores se diseñan utilizando un **reloj** que marca la realización de determinados eventos en el hardware
 - Estos intervalos se denominan **ciclos de reloj**
- **Periodo de reloj:** es el tiempo que se tarda en completarse un ciclo de reloj (250 picosegundos o 250 ps)
- **Frecuencia de reloj:** es la inversa del periodo de reloj (4 gigahertzios o 4 GHz)

Fórmula clásica del rendimiento de la CPU

- Estas fórmulas son particularmente útiles puesto que separan los tres factores clave que afectan al rendimiento
- En función de periodo:

$$T^o \text{ de CPU} = N^o \text{ de instrucciones} \times CPI \times \text{Periodo de reloj}$$

- En función de la frecuencia:

$$T^o \text{ de CPU} = \frac{N^o \text{ de instrucciones} \times CPI}{\text{Frecuencia de reloj}}$$

Fórmula clásica del rendimiento de la CPU

- El **CPI** o ciclos por instrucción, es el número medio de ciclos de reloj que se necesitan para ejecutar una instrucción
- El CPI nos permite comparar dos implementaciones distintas del mismo ISA
- **Speedup** se usa para mostrar el efecto en el rendimiento después de cualquier mejora en los recursos

$$S = (I \times CPI \times P)_{\text{sin mejora}} / (I \times CPI \times P)_{\text{con mejora}}$$

- **Ejemplo:** tiempo que tarda en ejecutarse un programa
 - 10s en A, 15s en B
 - Tiempo de ejecución_B / Tiempo de ejecución_A
 $= 15s / 10s = 1.5$
 - Así, A es 1.5 veces más rápido que B

Ejercicio 1

- Un programa tarda **10 segundos** en ejecutarse en el ordenador A, que tiene una frecuencia de reloj de **2 GHz**. Queremos que el computador B tarde **6 segundos** en ejecutar el mismo programa. En este caso, se puede introducir una mejora para incrementar la frecuencia de reloj, pero esto supone que el ordenador B necesitará **1,2 veces más ciclos de reloj** que el A para ejecutar el mismo programa. ¿Cuál es la frecuencia de reloj que cumple con estas condiciones?

Ejercicio 2

- Tenemos dos implementaciones del mismo ISA. La implementación A tiene un periodo de reloj de **250 ps** y un **CPI** de **2** para un programa, mientras que la B tiene un periodo de reloj de **500 ps** y un **CPI** de **1,2** para el mismo programa. ¿Qué ordenador es más rápido ejecutando este programa y por cuánto?

Ejercicio 3

- Un diseñador de compiladores se tienen que decidir entre dos secuencias de código para un computador particular. Se proporcionan los siguientes datos:

Tipo de instrucción	A	B	C
CPI	1	2	3
Secuencia de código	Número de instrucciones por cada clase de instrucción		
	A	B	C
1	2	1	2
2	4	1	1

¿Qué secuencia de código ejecuta más instrucciones?
¿Cuál es el CPI de cada secuencia? ¿Cuál es más rápida?

Resumen rendimiento

Componente hardware o software	¿A qué afecta?
Algoritmo	Recuento de instrucciones y posiblemente al CPI
Lenguaje de programación	Recuento de instrucciones y CPI
Compilador	Recuento de instrucciones y CPI
ISA	Recuento de instrucciones, frecuencia de reloj, CPI

Evolución histórica de los computadores

■ Inicios - válvulas

- En 1941 se inicia la construcción del ENIAC
- En 1945 John Von Neuman publica el artículo “First Draft of a Report on the EDVAC”

■ 2^a generación - transistores:

- 1947, invención del transistor
- CDC 1604, PDP-1 ...

■ 3^a generación - circuitos integrados:

- 1964, aparición del circuito integrado
- CDC 6400, PDP-5, ...

■ 4^a generación – microprocesadores:

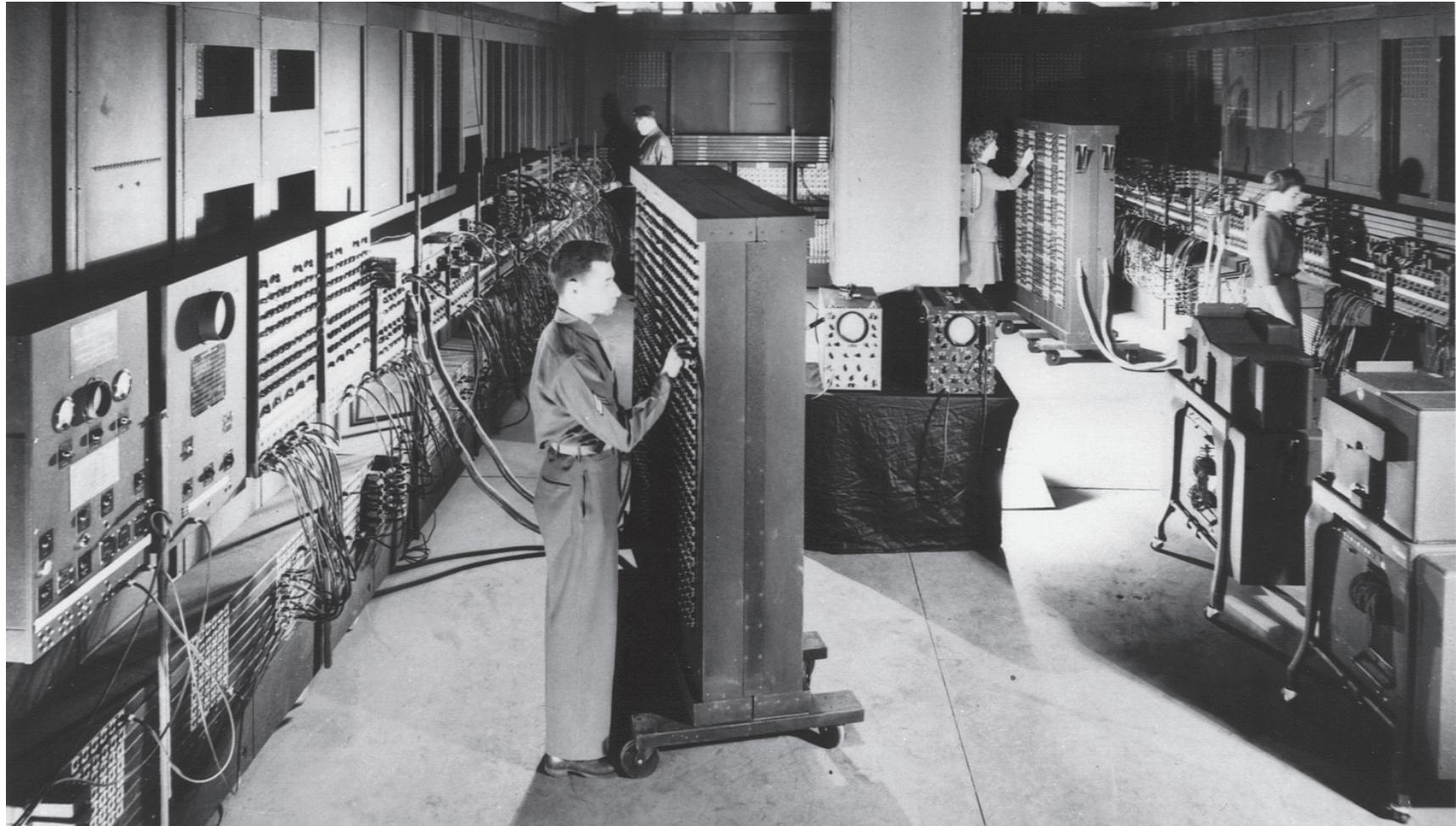
- Década de los 70, primer microprocesador y pastillas de memoria de semiconductor
- Ordenador personal

■ Redes de computadores y más

Tendencias tecnológicas

Año	Tecnología	Rendimiento / coste relativo
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit (IC)	900
1995	Very large scale IC (VLSI)	2,400,000
2013	Ultra large scale IC	250,000,000,000

ENIAC (Electronic Numerical Integrator and Calculator)



UNIVAC I, primer computador comercial de los EEUU



IBM System/360 modelos 40, 50, 60 y 75 introducidos en 1964



Cray-1, primer supercomputador comercial, 1976



El Apple IIc Plus



El Xerox Alto sirvió de inspiración para los modernos desktops

