**Introducción a Fundamentos de computadores**

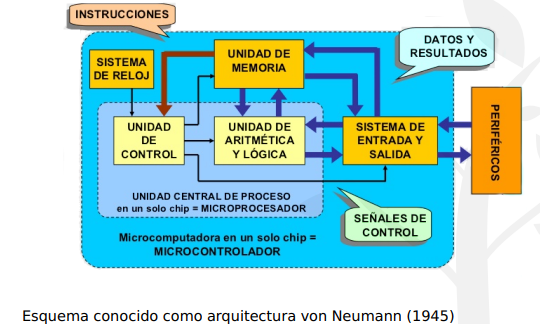
**Tipos de ordenadores**

* **PCs:** propósito general, compromiso coste/rendimiento
* **Servidores:** Elevada capacidad y rendimiento, uso intensivo de redes
* **Supercomputadores:** Aplicaciones científicas y en ingeniería, gran memoria, elevado coste y rendimiento
* **Ordenadores empotrados (embedded):** Son un componente dentro de otro sistema (ie un coche), limitaciones de consumo de potencia, coste y rendimiento

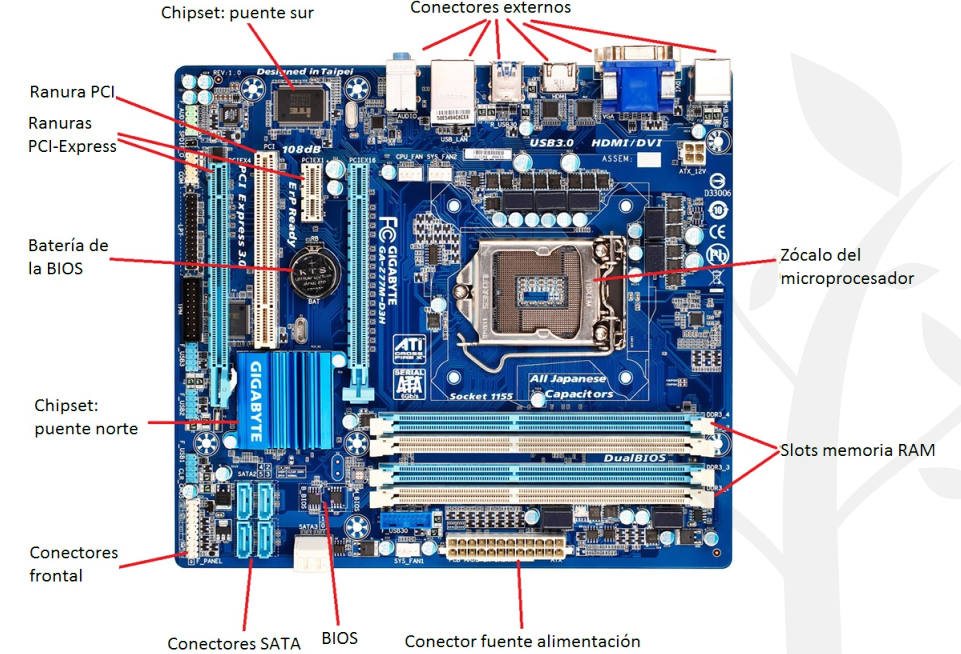
**Era post-PC**

* **Dispositivos móviles personales (PMD):** Pequeños, rápidos, permanentemente conectados, coste bajo
* **Computación en la nube:** Propiedad de grandes compañías con gigantescos datacenters
  + Infrastructure as a Service: Se alquila parte de la infraestructura
  + Software as a Service: Aplicaciones en la nube, la mayoría del software se ejecuta en la nube

**Componentes del computador**

****

**Placa base PC**

****

**CPU**

* Encargado del control y ejecución de operaciones
* Partes:
  + Unidad de control (CU): Dirige el resto de unidades
  + Unidad de procesamiento (ALU): Realiza operaciones aritmético/lógicas
  + Banco de registros: Elementos de almacenamiento, guardan datos a ser procesados por la ALU
  + Memoria cache: memoria SRAM rápida para un acceso eficiente a datos e instrucciones
* Sistema multinúcleo → varias CPUs en un chip

**RAM**

* Mantiene las instrucciones y datos de programas mientras se ejecutan
* Se borra al apagar el ordenador
* Se conecta con la CPU mediante el bus de memoria
* Suele ser Dynamic RAM (DRAM): Asynchronous DRAM o Synchronous DRAM

**ROM**

* No volatil, read-only, mantiene información para iniciar el ordenador

**Disco duro**

* Magnéticos (HDD) o sólidos (SDD)

**Redes**

* De área local (LAN), de área amplia (WAN) e inalámbricas (wifi, bluetooth)

**GPU**

* Aceleran procesamiento gráfico
* Existen tarjetas dedicadas, integradas o híbridas

**Hardware y software**

* **Software de aplicaciones:** Lenguaje de alto nivel
* **Software de sistema:** Compiladores (de alto nivel a máquina o a ensamblador) y SO
* El hardware solo entiende lenguaje máquina (binario), que es creado por un ensamblador (de lenguaje ensambnlador a máquina)

**ISA (Instructions set architecture)**

* Conjunto de instrucciones máquina que entiende una CPU
* Distintos CPU pueden tener el mismo ISA pero diferenciarse en su microarquitectura
* **ABI:** ISA + interfaz de sistema operativo

**RISC: Reduced Instruction Set Computer**

* ISA pequeño y optimizado, de instrucciones uniformes.
* Accede a memoria con carga y almacenamiento

**CISC: ComplexInstruction Set Computer**

* ISA con muchas instrucciones de tamaño y formato variable
* Accede a memoria con cualquier instrucción

**32/64 bits**

* Con n bits se tienen direcciones de memoria de 0 a 2n-1.
* 32 bits: 4GiB, 4 x 230 bytes
* 64 bits: 4EiB, 4 x 260 bytes
* El software de 32 bits funciona en hardware de 64 bits, al revés no



**Arquitectura MIPS**

* Instrucciones sencillas de longitud fija: 32 bits
* Operandos en registro excepto en operaciones de carga de memoria
* Acceso a memoria mediante instrucciones de carga y almacenamiento
* 32 registros de 32 bits, el 1 está reservado por el ensamblador

**Rendimiento**

* **Tiempo de respuesta:** tiempo entre comienzo y final de un evento, en segundos.
* **Productividad:** Cantidad de trabajo en un tiempo determinado, medida en transacciones.
* Afectado por algoritmo, lenguaje de programación, compilado e ISA, procesador y sistema de memoria, entrada/salida y sistema operativo.

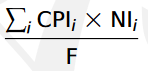
**Rendimiento relativo**

* El rendimiento se puede medir como 1 / tiempo de ejecución de un programa determinado.
* Para comparar el rendimiento relativo de B frente a A, se calcula
* Si un programa tarda 10s en A y 15s en B, A es 1.5 veces más rápido que B.
* Se puede medir tiempo transcurrido (incluyendo todos los aspectos) o sólo tiempo de CPU (no incluye tiempo de E/S ni el debido a la ejecución simultánea de otros prgoramas)
  + El tiempo de CPU se divide en tiempo de usuario y tiempo de sistema.

**Reloj de CPU**

* Período: Duración del ciclo del reloj, medido en segundos
* Frecuencia: Ciclos en un segundo, medido en Herzios. La frecuencia es el inverso del período.
* El **tiempo de CPU** de una aplicación se calcula realizando el cociente entre el número de ciclos que requiere y la frecuencia del procesador.

**CPI**

* **CPI:** Número de ciclos de reloj que necesita una instrucción máquina para ejecutarse. Depende de la instrucción y del hardware de la CPU.
* Dada una aplicación, podemos determinar el CPI medio de sus instrucciones si conocemos su número de instrucciones total y su número de ciclos.
* CPI de una aplicación puede ser menor que 1 si se ejecutan varias instrucciones en un ciclo.
* TCPU = . Esta fórmula se utiliza para calcular rendimiento relativo.

**Benchmarks**

* Método fiable para evaluar el rendimiento. Consiste en la ejecución de programas.
* Benchmarks sintéticos: Estructuras de programas reales, no son aplicaciones
* Benchmarks naturales: Utilizan programas reales.

**SPEC CPU Benchmark**

* Benchmark natural más utilizado, agrupando unos 20 programas.
* Crea un índice de rendimiento ponderando los resultados con cada programa.
* Existen distintas versiones, para calcular la eficiencia de distintos cálculos.

**Consumo de potencia**

* **Potencia dinámica:** Debido a la carga de los condensadores. Se puede estimar como ½ \* C \* V2 \* f
* **Potencia estática:** Debida a las corrientes de fugas. Se puede estimar como V \* Ifugas.

**Sistemas multicore**

* Más de un procesador por chip
* Dificil de programar, requiere equilibrar la carga de trabajo entre cores y optimizar su sincronización.
* **Ley de Amdahl:** la posible mejora del rendimiento está limitada por la proporción en la que se utilice la prestación mejorada.
* Tmejorado =
* La hipotética eficiencia máxima está limitada por la parte no mejorable.
* Otra formulación: Si a un código que se ejecuta en tiempo T le aplicamos una mejora de un F % en un factor M, el nuevo tiempo de ejecución es:

**Organización de memoria**

* Se organiza como un array de **palabras:** cadena finita de bits manejada como un conjunto por el procesador. Suele tener 1,2,4 o 8 bytes.
* Cada palabra se identifica por su posición en la memoria
* Ejemplo: si las palabras son de 4 bytes, el tercer byte de la palabra **d** estará en B(4\*d+2)
* Dada una dirección, se puede identificar el byte. Ejemplo: 0x3F28AC2E
  + Palabra: 3F28AC2E/4 = Palabra 0x0FCA2B0B
  + Byte: F28AC2E %4 = Byte 2
* **Big-Endian:** Byte más significativo en el final de la palabra
* **Little-Endian:** Byte más significativo en el principio de la palabra