ARCOM 1 – FUND DE DISEÑO DE COMP: SISTEMAS MULTINÚCLEO

PARÁMETROS DE DISEÑO DE UN SISTEMA

- **Rendimiento** (velocidad).
- **Coste**.
- **Potencia** (estática + dinámica)
  - Potencia pico (máxima)
  - Potencia media.
- **Robustez** { tolerancia al ruido. / resistencia a la radiación.
  - Tolerancia al ruido.
  - Resistencia a la radiación.
- **Testeabilidad**.
- **Reconfigurabilidad**.
- **Tiempo de salida al mercado**.
- Etc.

CLASIFICACIÓN DE LOS COMPUTADORES

- ***Personal Mobile Device*** (PMD) ⟶ énfasis en eficiencia energética y tiempo real.
- **Computación de escritorio** ⟶ énfasis en coste-rendimiento.
- **Servidores** ⟶ énfasis en disponibilidad, escalabilidad y productividad.
- ***Clusters / Warehouse Scale Computers*** ⟶ énfasis en disponibilidad y coste-rendimiento.
  - ↳ Usados para "Software como servicio" (*SaaS*).
  - **Supercomputadores** ⟶ énfasis en rendimiento en punto flotante y redes internas rápidas.
- **Internet de las cosas** / *Embedded Computers* ⟶ énfasis en precio.

Feature	Personal mobile device (PMD)	Desktop	Server	Clusters/warehouse-scale computer	Embedded
Price of system	$100–$1000	$300–$2500	$5000–$10,000,000	$100,000–$200,000,000	$10–$100,000
Price of micro-processor	$10–$100	$50–$500	$200–$2000	$50–$250	$0.01–$100
Critical system design issues	Cost, energy, media performance, responsiveness	Price-performance, energy, graphics performance	Throughput, availability, scalability, energy	Price-performance, throughput, energy proportionality	Price, energy, application-specific performance

ARQUITECTURA DE UN COMPUTADOR

Visión **simplificada** del diseño de la arquitectura de un computador:

- Se centra en el diseño del **ISA** (*Instruction Set Architecture*).
  - ↳ Toma de decisiones: registros, direccionamiento de memoria, operandos de las instrucciones, tipos de instrucciones, etc.

Visión **realista** del diseño de la arquitectura de un computador:

- Se diseña para satisfacer los **requisitos específicos de la máquina objetivo**, maximizando el rendimiento con ciertas restricciones (coste, potencia y disponibilidad).
  - ↳ Toma de decisiones: ISA pero también microarquitectura y hardware.

CONCEPTO Y TIPOS DE PARALELISMO

- La **explotación del paralelismo** está relacionada con la **mejora del rendimiento** del sistema.
- Para poder explotar el paralelismo es necesario modificar las aplicaciones: **el código debe exponer el paralelismo de manera explícita**.

TIPOS DE PARALELISMO

Paralelismo en **aplicaciones**:

- Paralelismo a nivel de **datos** (DLP).
- Paralelismo a nivel de **tareas** (ALP).

Paralelismo en **arquitecturas**:

- Paralelismo a nivel de **instrucciones** (ILP) ⟶ se ejecutan a la vez varias instrucciones en distintas fases.
  - ↳ Se dice que está **agotado**, a nivel tecnológico ya no hay muchas mejoras posibles y se deben buscar otros métodos para mejorar el rendimiento.
- Paralelismo a nivel de **hilos** (TLP).
- Paralelismo a nivel de **petición** (RLP).
- **Arquitectura de vectores / Unidades de procesamiento gráfico** (GPUs).
  - ↳ Instrucciones vectoriales ⟶ agrupan las variables en vectores, permitiendo realizar muchos cálculos en pocas instrucciones.

CLASIFICACIÓN DE ARQUITECTURAS PARALELAS

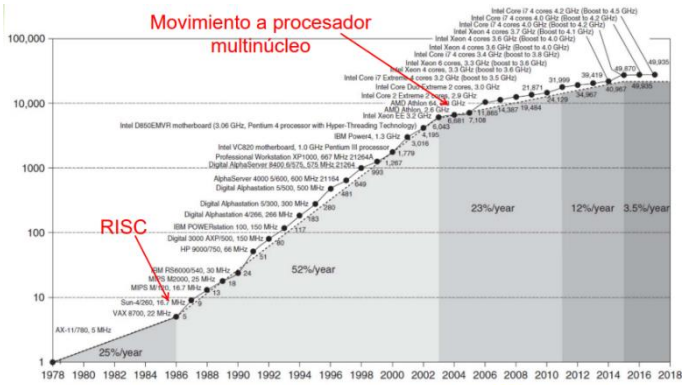
- Existen diferentes criterios que permiten clasificar las arquitecturas paralelas: **taxonomía de Flynn**, según la organización del sistema de memoria, según la escalabilidad, disponibilidad de sus componentes, cociente rendimiento/coste, etc.

TAXONOMÍA DE FLYNN

- La **taxonomía de Flynn** es una clasificación de arquitecturas paralelas según sus flujos de datos e instrucciones.
  - **SISD** (*Single Instruction stream, Single Data stream*) → computador secuencial que puede explotar paralelismo a nivel de instrucción.
  - **SIMD** (*Single Instruction stream, Multiple Data stream*) → arquitecturas vectoriales, GPUs o extensiones multimedia (como las instrucciones SSE o AVX).
  - **MISD** (*Multiple Instruction streams, Single Data stream*) → no hay implementaciones comerciales (no tiene mucho sentido).
  - **MIMD** (*Multiple Instruction streams, Multiple Data streams*) → cada procesador tiene sus propias instrucciones y opera sobre sus propios datos.
- Muchos computadores son **híbridos** entre diferentes tipos.

ARQUITECTURAS MIMD - MULTIPROCESADORES

- Las arquitecturas MIMD son las de **uso más extendido**:
  - Procesadores de memoria compartida (**UMA**, *Uniform Memory Access*).
    - Todos los procesadores comparten el **mismo espacio de direcciones** (como si tuvieran una única memoria compartida entre todos).
    - El programador **no necesita** conocer la ubicación de los datos.
  - Procesadores de memoria distribuida o multicomputadores (**NUMA**, *Non-Uniform Memory Access*).
    - Cada procesador tiene su **propio espacio de direcciones**.
    - El programador **necesita** conocer la ubicación de los datos.
  - **Multicores** → multiprocesadores en un único chip.
    - Tienen **más de un procesador por chip**.
    - Requieren **programación paralela explícita** → el hardware tiene la capacidad de ejecutar varias instrucciones a la vez, pero esta característica suele estar oculta al programados.
    - En ellos es **difícil**:
      - programar para conseguir un buen rendimiento.
      - conseguir balanceo de carga entre los núcleos.
      - optimizar la comunicación y sincronización.
  - Otros tipos como ***clusters***, **MPPs**, etc.



Rendimiento de los procesadores a lo largo del tiempo ejecutando un SPECint comenzando por sistemas de un solo procesador (eje Y rendimiento frente a un procesador estándar de 1978)

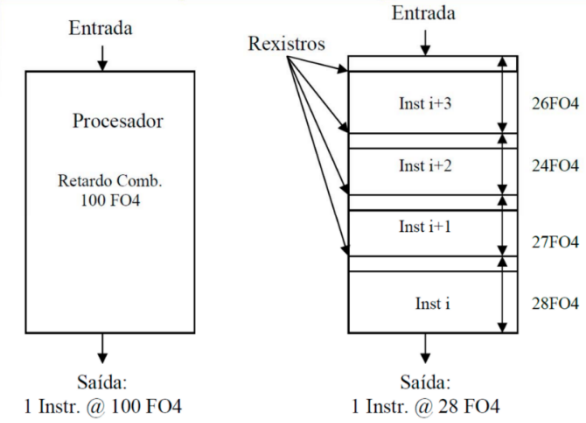
Intel Pentium4 (2003)

- Application: desktop/server
- Technology: 90nm (1/100th of 4004)
- 55M transistors (20,000x)
- 101 mm² (10x)
- 3.4 GHz (10,000x)
- 1.2 Volts (1/10th)
- 32/64-bit data (16x)
- 22-stage pipelined datapath
- 3 instructions per cycle (superscalar)
- Two levels of on-chip cache
- data-parallel vector (SIMD) instructions, hyperthreading

CONCEPTOS DE PROCESADORES MULTINÚCLEO

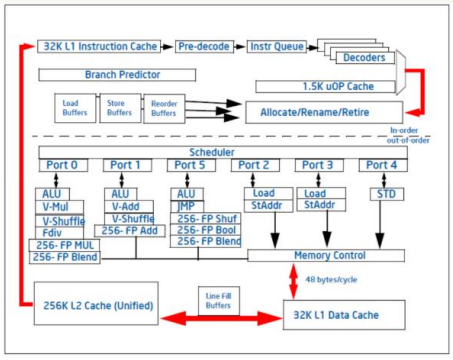
SEGMENTACIÓN (pipeline)

- La SEGMENTACIÓN es la ejecución de las instrucciones en **etapas**, de manera que cada etapa tiene un **retardo adicional** (para almacenar los resultados de la etapa en registros).
- La PROFUNDIDAD DE SEGMENTACIÓN es el número de etapas de segmentación.
- FO4 (*Fan-out of* 4) es una unidad de tiempo equivalente al retardo que tiene una señal al atravesar un único inversor con 4 inversores conectados a la salida.
  - Su valor exacto depende de la tecnología de fabricación, es decir, del nodo tecnológico.



- El **retardo de cada instrucción** en un procesador con segmentación es más alto que sin ella debido a la lógica adicional añadida, pero pasa menos tiempo **entre instrucciones**.
- En cada ciclo de reloj se emite una instrucción para ser ejecutada, pero alguna puede quedarse **bloqueada** en alguna etapa (por ejemplo, para buscar un dato en memoria), provocando que las instrucciones posteriores no puedan avanzar, lo cual se conoce como BURBUJA (parada de emisión), lo que reduce la productividad.
- Con segmentación se usa una **frecuencia de reloj más alta** pues el periodo será de la longitud aproximada de una fase (se intenta que todas tarden aproximadamente lo mismo).

ARQUITECTURA DE UN NÚCLEO TÍPICO



Arquitectura de un núcleo. Cachés L1 y L2 privadas para el núcleo.

- Nivel L1 con 2 cachés separadas para Datos e Instrucciones de 32KB cada una con tamaño de línea de 64 bytes.
- Cache L1 de datos permite 2 lecturas y 1 escritura por ciclo de reloj.
- Cache L2 unificada con latencia de acceso de 12 ciclos y ancho de banda de 32 bytes/ciclo.
- Cache L3 de 4-32MB, ancho de banda de 32 bytes/ciclo, y tiempo de acceso de 25-35 ciclos.
- Hardware para ejecución fuera de orden y unidades de ejecución que operan en paralelo.

ARQUITECTURA DE UN MICROPROCESADOR MULTINÚCLEO

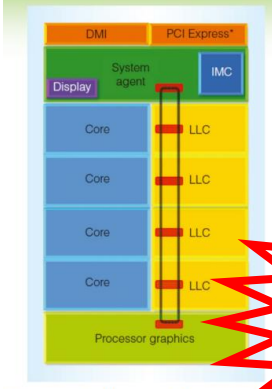


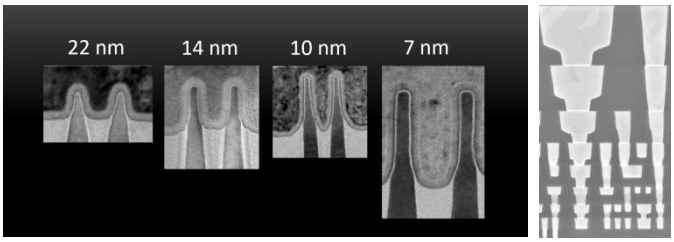
Diagrama de bloques del procesador Haswell 2013 (nodo tecnológico de 22 nm). Core = núcleo

- Se muestra el último nivel de caché (LLC) que es L3 compartida entre todos los núcleos mediante conexión en anillo.
- Tamaño típico de L3 4-32 Mbytes (hasta 96 MBytes)
- Tiempo de acceso de L3 25-35 ciclos.
- Ancho de banda de L3 de 32 bytes/ciclo para cada núcleo.
- Las diferentes unidades (núcleos, cachés, procesador gráfico y el agente del Sistema) son dominios de voltaje-frecuencia independientes.
  - Una unidad de control **dinámicamente decide cómo distribuir el consumo de potencia** entre ellos para mejorar el rendimiento.

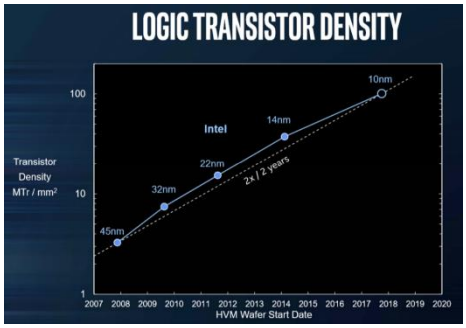
ESCALAMIENTO DE PRESTACIONES EN MICROPROCESADORES

TECNOLOGÍAS DE FABRICACIÓN CMOS

- Los chips tienen **varias capas de transistores** apiladas para ahorrar superficie.
- El valor numérico que identifica al NODO TECNOLÓGICO es el ancho **mínimo** de una **conexión de** cobre de **nivel 1**.
  - ↳ Está relacionado directamente con la superficie que ocupa el transistor.

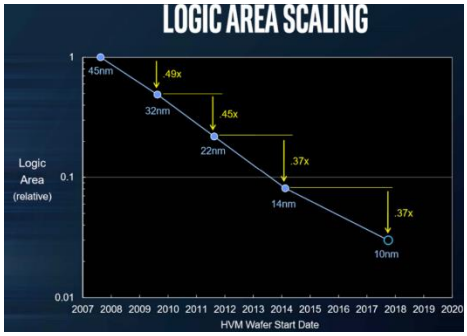


EVOLUCIÓN DEL ÁREA LÓGICA DE UN CIRCUITO EN GENERAL CON LA TECNOLOGÍA



- La **Ley de Moore** predice que la densidad de transistores por unidad de área se duplica cada 2 años aproximadamente.
- Cada nodo tecnológico supone una **reducción del 0.7x de las conexiones** entre transistores y del **0.5x del área** de un chip.
  - En el caso **ideal**, esto supondría que el procesador tiene el **doble de superficie** disponible, pero en **realidad no todas sus partes escalan igual** (especialmente, la memoria DRAM escala especialmente mal).

EVOLUCIÓN DEL ÁREA LÓGICA DE MICROPROCESADORES CON LA TECNOLOGÍA



- **Al principio** el área ocupada por la misma cantidad de transistores se **reducía a la mitad** entre nodos tecnológicos, pero **cada vez se reduce menos** (en torno a 0.4x).

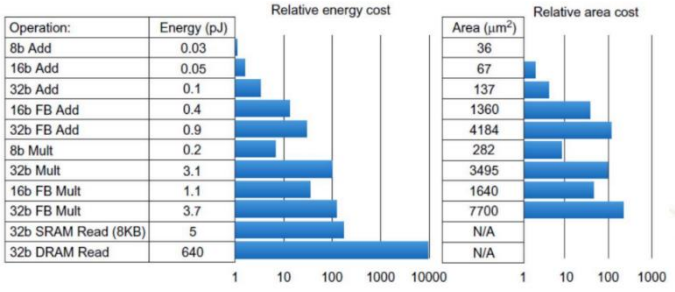
TENDENCIAS EN POTENCIA, ANCHO DE BANDA Y LATENCIA

POTENCIA DINÁMICA

- La POTENCIA DINÁMICA es la energía consumida por unidad de tiempo por un circuito CMOS cuando se produce un paso de un transistor $0 \rightarrow 1$ o $1 \rightarrow 0$.
  Es **proporcional** a:

$$^1/_2 * Carga\ capacitiva * Voltaje^2 * Frecuencia\ reloj$$

- La **carga capacitiva** depende del **número de transistores activos** y de la **tecnología** (que determina la capacitancia de cables y transistores).
  - ↳ En CMOS escala ineficientemente, 0.8x de una generación a otra (mientras que los transistores x2).



POTENCIA ESTÁTICA

- La POTENCIA ESTÁTICA es la energía consumida por unidad de tiempo por un circuito CMOS sólo por estar enchufado.
  Es **proporcional** a:

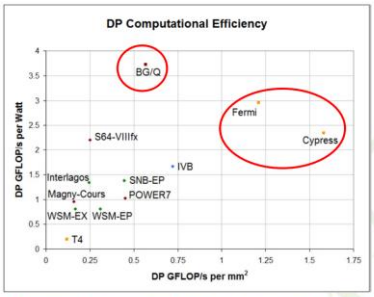
$$Corriente_{estática\ de\ fugas} * Voltaje$$

Para reducirla:

- *Power gating* → desconectar las partes que no se estén usando.
- Diseños *domain-specific* → diseños optimizados para una operación muy frecuente.

- Escala con el **número de transistores**, aunque estén sin funcionar.
  - ↳ Si se usan cachés SRAM grandes, puede llegar a 50%.

EFICIENCIA ENERGÉTICA EN MICROPROCESADORES DE ALTAS PRESTACIONES

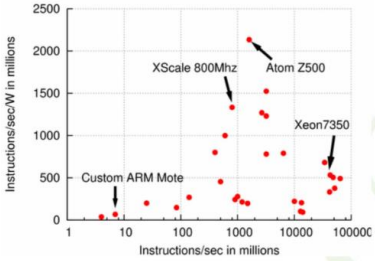


- Es una medida muy adecuada para cargas de trabajo con fuerte componente numérica.

Figura: Eficiencia de microprocesadores de altas prestaciones en GFLOPS/Watt (10^9 operaciones en punto flotante por segundo y por Watt para medir **eficiencia energética**) y en GFLOPS/mm^2 de área de chip (mide **densidad de computación**).

Eficiencia en consumo de potencia para carga numérica
BG/Q-> Blue Gene Q de IBM
Fermi y Cypress -> chips de computación orientados a gráficos

EFICIENCIA ENERGÉTICA EN MICROPROCESADORES DE PROPÓSITO GENERAL

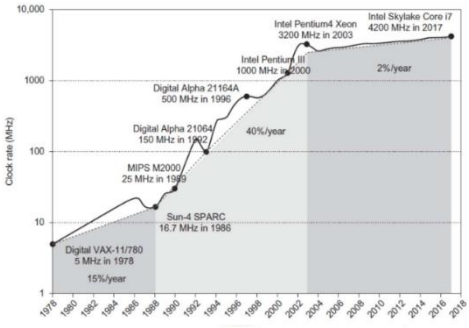


- La mejor posición es la esquina superior derecha.
- La posición en un cuadrante depende del propósito con el que hayan sido fabricados.
  - ✓ Xeon →servidores de altas prestaciones.

Eficiencia de microprocesadores de propósito general en instrucciones ejecutadas por segundo (aplicable a cargas de trabajo no dominadas por computación numérica en punto flotante).

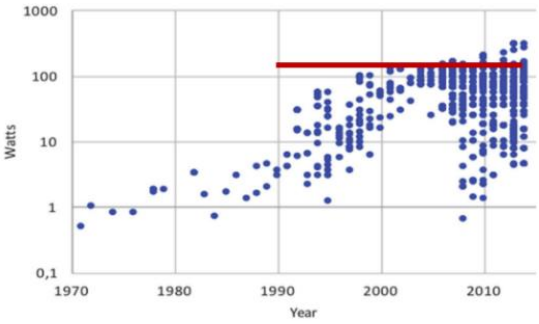
POTENCIA CONSUMIDA: VARIACIÓN DE LA FRECUENCIA DE RELOJ

- Intel 80386 consumía ~ 2 W
- 3.3 GHz Intel Core i7 consume 130 W
- El calor debe ser disipado desde un chip de 1.5cmx1.5 cm
- Atención desde 2003 en la gráfica: la frecuencia crece muy poco para no subir la potencia disipada.



THERMAL DESIGN POWER (TDP)

- El TDP (*Thermal Design Power*) caracteriza el consumo de potencia sostenido.
  - Se usa como objetivo en cuanto a **potencia suministrada** y **sistema de refrigeración**.
  - Su valor está entre la potencia pico y la media.
- La **frecuencia de reloj** se puede **reducir dinámicamente**, es decir, tomar valores distintos en distintas partes del sistema, para limitar el consumo de energía.
- Hay un **límite** en las condiciones de **temperatura** en las que puede funcionar un circuito.
- Se ha llegado a un **límite de disipación** de potencia (100-200W) debido a cuestiones técnicas y económicas.
  - ↳ Mantener el límite en cada nueva generación de microprocesadores requiere **optimizar el consumo de potencia**.

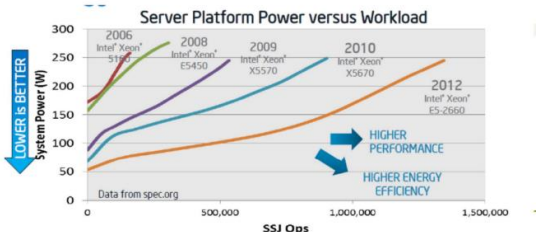


TÉCNICAS PARA AUMENTAR LA EFICIENCIA EN EL USO DE ENERGÍA

- **No hacer nada** ⟶ desconectar los módulos inactivos para que no consuman energía (ni estática ni dinámica).
- **Escalado dinámico de V y F** ⟶ aprovechar periodos de inactividad en portátiles o tabletas para operar a frecuencias y voltajes menores.
- **Estado de baja potencia en memorias DRAM y discos** ⟶ se trata de que las partes inactivas funcionen con una frecuencia menor.
- ***Overclocking*** ⟶ trabajar a frecuencia más alta durante breves periodos de tiempo.
  - Puede implicar apagar los núcleos para los que no se sube la frecuencia.
  - Limitado por la subida de la temperatura.
  - Transparente al usuario.

EFECTO DE LA EVOLUCIÓN TECNOLÓGICA EN PRESTACIONES Y POTENCIA

- Eje X: prestaciones del sistema.
- Eje Y: potencia consumida.
- Datos para servidores en base a Intel Xeon (varios núcleos) ejecutando SSJ
- Cada servidor corresponde a un nodo tecnológico y a una arquitectura diferente y es escalable en prestaciones variando número de núcleos/procesadores activos, frecuencia…



ESCALAMIENTO DEL RETARDO EN CIRCUITOS

- Se ha determinado **experimentalmente** que el **retardo típico de una tecnología**, el F04, es:

$$FO4 = 0.2 * T \text{ (en } ps)$$

donde T es el caracterizador del nodo tecnológico (en nm).

↳ El F04 puede aumentar hasta un 1.4x en condiciones de operación extremas.

Ejemplo:
Supongamos un microprocesador con ciclo de reloj de 150 F04.
- Para tecnología 10 nm.
  - ✓ Retardo de 1 F04 = 0,2 x 10 = 2ps
  - ✓ Frecuencia de reloj = 1/(150 x 2 ps) = 3,33GHz
- Para tecnología 65nm -> Frecuencia de reloj=1/(150 x 13 ps) = 513MHz

EFECTOS DE ESCALAR VOLTAJE Y FRECUENCIA SOBRE LA POTENCIA DINÁMICA

- Para las tecnologías actuales hay una relación obtenida **experimentalmente** entre el **escalado del voltaje**, en un factor Kv, y el escalado que como consecuencia tendrá la **frecuencia**, en un factor Kf:

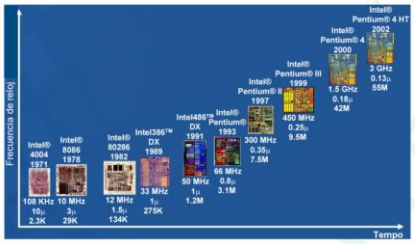
$$Kf = 1.45 * Kv - 0.4$$

- Sobre el **escalamiento del consumo de potencia dinámica** en un factor Kd:

$$Kd = Kv^2 * Kf$$

- Los procesadores actuales explotan esta propiedad masivamente:
  - Para rebajar el consumo de potencia en los circuitos que no están en los caminos críticos.
  - Mediante el escalado dinámico de voltaje en cada núcleo de procesamiento cuando no se demanda computación intensiva.

EVOLUCIÓN DE LA FRECUENCIA DE RELOJ EN LOS MICROPROCESADORES DE INTEL



Cuanto más disminuye el periodo de la señal de reloj más profunda será la segmentación (más etapas de segmentación)

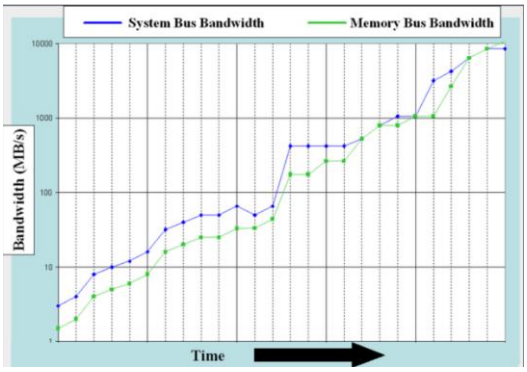
- Intel Pentium II de 1997:
  - Frecuencia = 300 MHz
  - Tecnología 350 nm
  - Ciclo de reloj =
  - 3,3 (ns/ciclo) / 70 (ps/F04) = 47 F04.
- Pentium 4 con HyperThreading de 2002:
  - Frecuencia = 3 GHz
  - Tecnología 130 nm
  - Ciclo de reloj = 0,33 (ns/ciclo)/26 (ps/F04) = 12,5 F04.

ANCHO DE BANDA Y LATENCIA

- ANCHO DE BANDA ⟶ transferencia de información por unidad de tiempo.
- El ANCHO DE BANDA MÁXIMO SOSTENIDO (BWM) se diseña para satisfacer cargas de trabajo representativas para el procesador.

$$BWM = \frac{N * F}{IP * CPI_{core_i}}$$

donde N es el número de núcleos, F es la frecuencia, CPI es el número de instrucciones por ciclo para cada núcleo y IP la intensidad operacional para la cual fue dimensionado el procesador (medida en instrucciones/Byte).

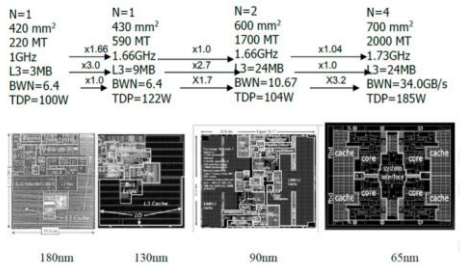


- En la práctica, el BMW **no siempre escala entre generaciones**: se suele **duplicar entre nodos tecnológicos** y mantener el mismo diseño de memoria dentro del mismo nodo.
- El **ancho de banda** y la **latencia** evolucionan de manera muy distinta en diferentes unidades:

Intervalo de mejora	En procesadores	En memoria y discos
Ancho de banda	$32000x - 40000x$	$300x - 1200x$
Latencia	$50x - 90x$	$6x - 8x$

ESCALAMIENTO DE PRESTACIONES

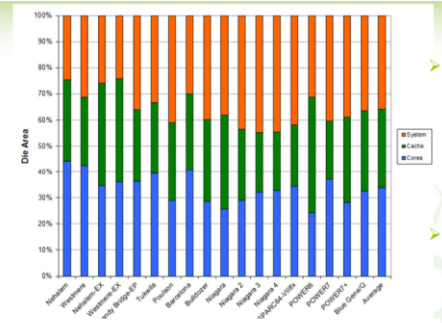
ESCALAMIENTO HARDWARE



Evolución de la familia de microprocesadores de altas prestaciones Intel Itanium 2 en 4 nodos tecnológicos (180nm en 2001, 130nm en 2003, 90nm en 2005 y 65nm en 2008).

- **N** indica el número de núcleos
- **BWN:** Tasa de transferencia nominal con la memoria (ancho de banda nominal)
- **MT** indica el número de transistores en millones
- Para 180nm la distribución de área es
  - 61% caché L3
  - 25% procesador
  - 9% circuitos de I/O
  - 5% lógica de interfaz con el bus

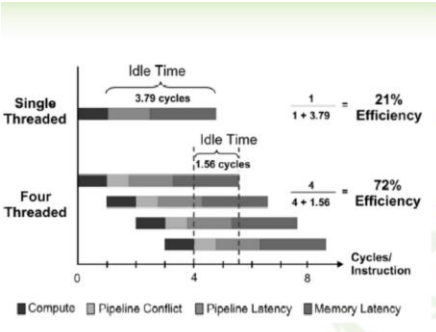
DISTRIBUCIÓN DEL ÁREA DE CHIP EN MICROPROCESADORES



Distribución de área en microprocesadores para servidores de grandes sistemas.
system indica área dedicada al controlador de memoria integrado, a la interfaz con la memoria, y al hardware adicional para interconectar sistemas multiprocesador.

- Valores medios de distribución de área:
  - 34% para núcleos
  - 30% para caché
  - 36% para hardware del sistema
- Ejemplo de densidad de transistores por unidad d área (Itanium 2003):
  - Caché: 2.1 Millones trans/mm2
  - Procesador: 250 mil trans/mm2

EFECTO DE LA EJECUCIÓN MULTIHILO HARDWARE

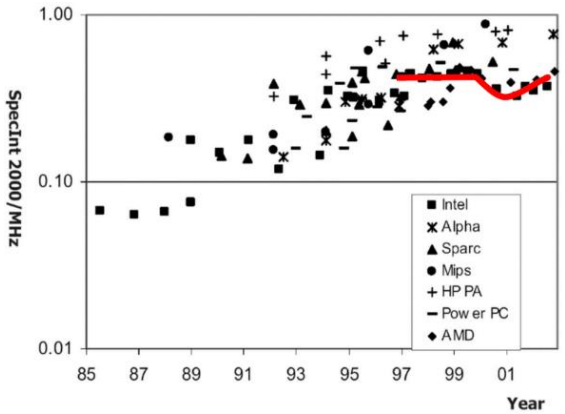


Reducción del efecto de la latencia de memoria mediante ejecución multihilo en hardware. La ganancia depende del tipo de carga de trabajo y del programa que se ejecute.

- Los núcleos actuales soportan más de un hilo hardware para enmascarar parte de la latencia de memoria.
- Con un solo hilo 3.79 ciclos de paradas por instrucción (CPI=4.79) y 0.21 instrucciones por ciclo.
- Con ejecución de 4 hilos cuando uno para, el siguiente entra a ejecutarse
- La presencia de varios hilos además de coste hardware provoca conflictos en el pipeline.
- Con 4 hilos el tiempo de espera pasa a 1,56 ciclos y da CPI=1,39, o sea, 0,72 instrucciones /ciclo.

- Los núcleos actuales soportan **más de un hilo** hardware para enmascarar parte de la latencia de memoria, de manera que cuando alguno se detiene se comienza a ejecutar otro.
  - ↳ La reducción de latencia depende del tipo de carga de trabajo y programa.
- × Implica **coste de hardware** y **conflictos en el pipeline**.

EVOLUCIÓN DE PROCESADORES DE UN SOLO NÚCLEO EN PRESTACIONES

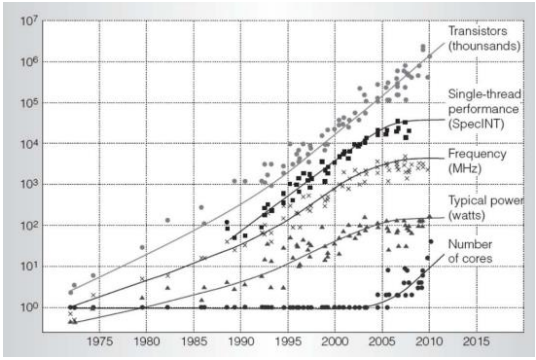


↳ *SpecInt* es un benchmark.

- La tendencia para los procesadores de Intel fue mantener **constante el CPI**, aunque en las últimas generaciones **aumentó** debido a la **profundidad de segmentación**.

FINAL DE LOS PROCESADORES DE UN SOLO NÚCLEO

- Hasta 2003 se usó un modelo **mononúcleo** de procesamiento **superescalar**, es decir, ejecución de varias instrucciones a la vez en cada ciclo.
  - La **latencia** escalaba **0.5x** de una generación de microprocesadores a la siguiente.
  - La **frecuencia** escalaba **2x** de una generación de microprocesadores a la siguiente, manteniendo el CPI constante.
    - ↳ Para ello se aumentaba la **profundidad del pipeline** y el **tamaño de la caché** (y, por tanto, el área del procesador).
- El modelo basado en un solo núcleo se ha **agotado** debido a que no se puede reducir más el CPI sin disparar el consumo:
  - De una generación a otra, el **área** de los componentes escala **0.5x**, lo que en principio reduciría el CPI, pero, debido a usar (por ejemplo) cachés más grandes, aparecen **conexiones que no reducen su longitud**, lo que provoca problemas:
    - Se necesitan más ciclos de reloj para transmitir una señal por esas conexiones largas.
    - Las conexiones largas se segmentan mediante registros, aumentando la profundidad del pipeline y por tanto el CPI.
    - ▸ Para disminuir el CPI, hace falta mejorar la predicción de saltos y conseguir mejores estructuras, más complejidad hardware.
      - ↳ Esto, a su vez, implica tener conexiones más largas, creando una paradoja.
  - Además, el **modelo de programación** limita el **CPI mínimo** que se puede alcanzar.
    - ↳ Por tanto, llega un momento en el que la precarga de datos o instrucciones, la ejecución multihilo, … no son suficiente.



MÉTRICAS DE RENDIMIENTO DE UN SISTEMA

Ejemplo:
```
>time ./programa datos
real  0m11.588s  -> tiempo total o tiempo de respuesta
user  0m3.735s   -> tiempo de CPU del usuario
Sys   0m3.286s   -> tiempo de CPU del S.O.
```

- Las MÉTRICAS DE RENDIMIENTO de un sistema se basan en medir tiempos de ejecución:
  - **TIEMPO DE RESPUESTA**, latencia o tiempo real → tiempo entre el inicio y el final de una tarea.
    - ↳ Incluye todos los sobrecostes del sistema (E/S, acceso a memoria RAM, etc.).
  - **TIEMPO DE CPU** → tiempo de computación en CPU para una tarea concreta $T_{CPU} = T_{usuario} + T_{SO}$
  - **TROUGHPUT** o productividad → cantidad de trabajo realizado por unidad de tiempo.
  - **SPEEDUP** → aceleración de una versión X relativa a una versión Y para un programa en particular $SPEEDUP = T_{ejecucion_X}/T_{ejecucion_Y}$
  - **CPI** → número medio de ciclos de reloj que una instrucción necesita para ejecutarse.
    $CPI = (\sum_{i=1}^{m} CPI_i * NI_i)/NI$ donde NI_i es el número de instrucciones del tipo i, CPI_i es el número de ciclos de reloj para las instrucciones de tipo i y m es el total de tipos de instrucciones existentes.
    - ↳ Instrucciones diferentes pueden necesitar diferente cantidad de ciclos de reloj.
  - **MIPS** (millones de instrucciones por segundo) → $MIPS = NI/(T_{CPU} * 10^6)$
    - × No se puede usar para comparar computadores con diferentes ISAs.
    - × Varía entre programas del mismo computador.
  - **GFLOPS** (miles de millones de operaciones en punto flotante por segundo) → $GFLOPS = FLOPS/(T_{CPU} * 10^9)$
    - × Varía entre programas del mismo computador.
  - **ECUACIÓN BÁSICA DE RENDIMIENTO** → $T_{CPU} = NI * CPI/F.$
    - ↳ Se puede usar para { comparar dos realizaciones diferentes (dos microarquitecturas diferentes para el mismo ISA).
      evaluar un diseño alternativo si se conoce el impacto de los tres parámetros.
  - **BENCHMARKS** → medida de prestaciones en comparación con un sistema de referencia y para un conjunto de programas de prueba.
    - *Kernels* (*matrix multiply*) → programas típicos.
    - Programas "de juguete" (*sorting*) → tienen un código muy simple.
    - *Benchmarks* sintéticos (*Dhrystone*) → no ejecutan código con funcionalidad real.
    - Conjuntos de programas reales que simulan cargas de computación estándar (SPEC06fp, TP-C).

LEY DE AMDAHL

- **La posible mejora de rendimiento está limitada por la proporción en que se use la prestación mejorada**.

$$t_{mejor} = \frac{t_{parte\ mejor}}{\%_{de\ mejora}} + t_{parte\ no\ mejor} \rightarrow t_{mejor} \geq t_{parte\ no\ mejor}$$

- Versión para tontos: a un código que se ejecutaba en t se le aplica una mejora a un porcentaje de su ejecución F que lo reduce en un factor M.

$$t' = \frac{F * t}{M} + (1 - F) * t$$

TÉCNICAS DE MEDIDA DE RENDIMIENTO

UTILIZACIÓN DE INFORMACIÓN MANTENIDA POR EL SO

- Los SOs actuales mantienen información sobre las diferentes tareas que se están ejecutando (uso de memoria y de la CPU, número de tareas del sistema, tiempo de inicio y finalización de cada tarea, etc.).

✓ No hace falta desarrollar **código específico** para realizar las medidas.
× Se añade **sobrecarga** al acceder a esta información.
× **Formato poco amigable** que impone bastantes restricciones.
× No mantiene información de **todos los eventos**.

MONITORES HARDWARE

- Los **fabricantes incluyen en el hardware** monitores que permiten contar con gran precisión una multitud de eventos (utilización de memoria, utilización de CPU, alertas, etc.).

✓ No se consumen **recursos** (ni tiempo de CPU ni almacenamiento).
✓ Se pueden monitorizar **eventos de muy bajo nivel** que son transparentes para el software y el SO.
✓ Muy **precisos**.
× Se miden **magnitudes físicas** (como la temperatura) y no lógicas.
× Requieren **librerías específicas**.

MONITORES SOFTWARE

- Se ejecuta una **aplicación específicamente programada** para recoger y tratar la información necesaria.

Tipos de monitores software:

- Detección de eventos.
- Muestreo { cuenta de eventos. / traza.

✓ Se adecúan a las necesidades de los usuarios recogiendo la **información necesaria**.
× Se añade **sobrecarga** al acceder a esta información si se hace a menudo.

ANÁLISIS DE PROGRAMAS - PROFILING

- Se utilizan medidas de tiempo y recursos **incluidas por el programador en su código**.
- Son útiles cuando la evaluación del rendimiento es a **alto nivel** y en términos de **optimización de código o de un sistema**.

✓ No está sujeta a **errores aleatorios** como el muestreo.
× **Sobrecarga** la aplicación.

BENCHMARKS

- Como se dijo antes, para evaluar el rendimiento de un sistema se puede usar un conjunto específico de programas de prueba conocidos como BENCHMARKS
  - La práctica estándar es usar conjuntos de programas de **aplicación real**.
  - Los programas de prueba forman una carga con la que el usuario espera **predecir el rendimiento de la carga real del sistema**.
- Para indicar las medidas se debe hacer un **informe** de forma que otra persona pueda reproducir los resultados (versión del SO, compilador, entradas, …).
- El rendimiento de la máquina medido con un benchmark se suele dar como un **único número**.
- El grupo de programas de prueba más popular y completo es el SPEC (*Standard Performance Evaluation Corporation*).
  - Se usan para medir **tiempo de CPU** y **productividad**.
  - Los 43 programas que se incluyen en la última versión de SPEC para procesador se agrupan en:
    - Carga computacional intensa en **punto flotante** (SPECfp2017).
    - Carga computacional intensa en **enteros** (SPECint2017).
  - El **procesador**, la **memoria** y **la E/S** del sistema influyen en el valor medido, junto con el **programa utilizado**.
  - Los tiempos de ejecución del sistema deben ser normalizados, para lo que se usa otro sistema como **referencia** (normalmente uno muy antiguo):
    - Ratio SPEC para un programa → $SPEC = T_{CPU\ Referencia}/T_{CPU\ Test}$

    El test se repite para todos los programas del conjunto SPEC y se computa la media geométrica de los resultados:
    - Ratio SPEC para un conjunto de n programas → $velocidad\ SPEC = \sqrt[n]{\prod_{i=1}^{n}(SPEC_i)}$

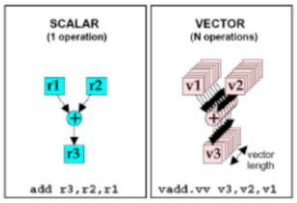
CONCLUSIONES DE EVALUACIÓN DE RENDIMIENTO

- La **medida más importante** del rendimiento de un computador es el **tiempo de ejecución**. La **segunda** más importante es la **productividad**.
- El **tiempo de ejecución** en un procesador depende de:
  - La organización de su hardware (CPI).
  - Su ciclo de reloj (T_{ciclo}).
  - El número de instrucciones máquina ejecutadas (NI, CPI).
  - El compilador usado (NI).
- Existen otras métricas del rendimiento como los **MIPS** o **GFLOPS**.
- Los **benchmarks** nos permiten comparar tiempos de ejecución entre máquinas diferentes y obtener una **media normalizada del rendimiento**.
- La **Ley de Amdahl** permite acotar la aceleración que se obtendrá al mejorar alguno de los subsistemas del conjunto.

PARALELISMO DE DATOS

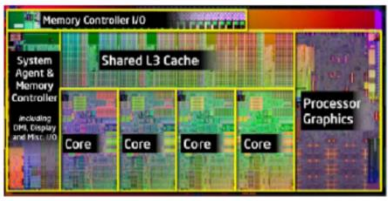
- Existen aplicaciones estructuradas en tareas simples denominadas KERNELS (conjuntos de instrucciones) que operan sobre muchos datos, con potencial de ejecución paralela:
  - ▷ Por ejemplo, aplicaciones gráficas, multimedia y de realidad virtual.
  - ▪ En ellas es **fácil** disponer de hilos que puedan **operar en paralelo**, lo **difícil** es **proporcionar datos a la velocidad necesaria**.

Ejemplo de operaciones vectoriales para realizar una suma. Una sola instrucción da lugar a un elevado número de computaciones. Se incrementa, por tanto, el paralelismo.



PROCESADORES VECTORIALES

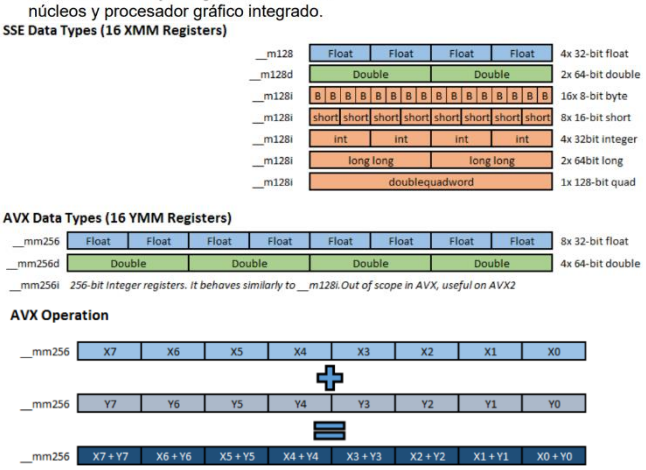
- ○ Una posible solución es incluir en los procesadores **núcleos especializados** en el **procesamiento gráfico** para aprovechar el paralelismo de datos.
  - ↳ AMD llama a este tipo de arquitectura APU, Intel no le da un nombre específico.



Procesador Sandy Bridge de Intel, con 4 núcleos y procesador gráfico integrado.

EXTENSIONES VECTORIALES

- ○ Otra posible solución es incluir en los **núcleos de propósito general** la posibilidad de **procesamiento vectorial**.
- • Intel fue incorporando **instrucciones para procesamiento vectorial** en todos sus procesadores mediante las **extensiones MMX, SSE y AVX**.
- • Actualmente se pueden realizar operaciones en paralelo sobre **vectores de 256 bits** (es decir, 8 operandos en *float* o 4 en *double*).
- ✗ Las instrucciones vectoriales aumentan la complejidad de la programación.
- ✓ El aumento en velocidad suele compensar.
  - ↳ Para tamaños de problema grandes se puede llegar a ganar hasta 8x en tiempo de ejecución si se usan operaciones AVX.

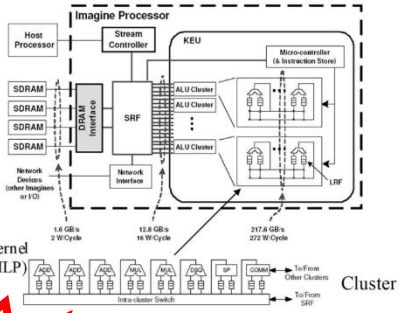


MICROPROCESADORES BASADOS EN STREAMING

- ○ Los microprocesadores basados en streaming extienden el concepto de procesador vectorial.
  - ↳ Son los usados en las tarjetas gráficas de Nvidia.
  - ▪ En lugar de organizar la información en vectores, usan **streams**, que son vectores de estructuras.
  - ▪ En lugar de instrucciones vectoriales simples, usan **kernels**, que actúan sobre cada uno de los elementos del stream.
- ✓ El **modelo de programación** basado en streams ofrece **más oportunidades de paralelismo** que el de programación paralela.
- ✗ La **programación** para estas unidades, (se suele realizar con CUDA o OpenCL) es en general **más compleja** que en entornos habituales de programación.
  - ▪ La **carga y escritura** de streams de datos a memoria son **responsabilidad del programador**.
  - ↳ Ha habido un desarrollo reciente de muchas librerías que facilitan la programación para cualquier tipo de tarea.

Ejemplo de arquitectura de un procesador con *streaming*:

- La unidad de ejecución consta de varios clústers de ALUs con sus registros.
- En esta arquitectura de la derecha todas las ALU ejecutar el mismo *kernel*.
- Los *streams* se reparten entre los clústers

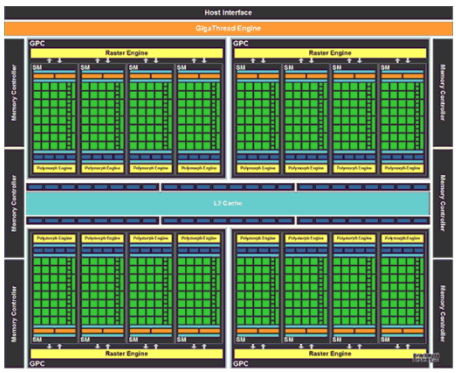


Mapeo do kernel nos clúster (ILP)

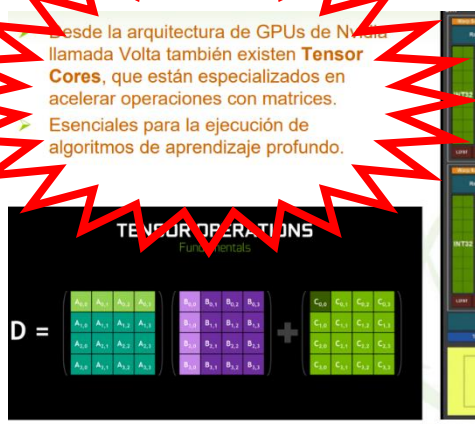
Cluster

Arquitectura del procesador gráfico para la arquitectura Fermi de Nvidia (2009):

- 16 *streaming processors* que ocupan la mayor parte del chip.
- Están dedicados a tareas más allá del procesamiento gráfico
- Se usan como coprocesadores de la CPU del sistema.
- Se programan mediante CUDA o OpenCL.



- Desde la arquitectura de GPUs de Nvidia llamada Volta también existen **Tensor Cores**, que están especializados en acelerar operaciones con matrices.
- Esenciales para la ejecución de algoritmos de aprendizaje profundo.



Detalle de un SM (Streaming multiprocessor) con tensor cores de la arquitectura Turing de Nvidia para GPUs, e particular, para GTX 2080 Ti.