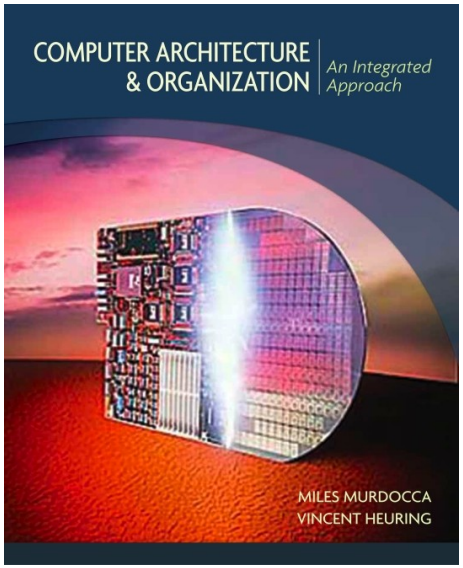
Organización de las Computadoras

Leonardo Giovanini



Microarquitecturas multihilo y multinúcleo

Contenidos

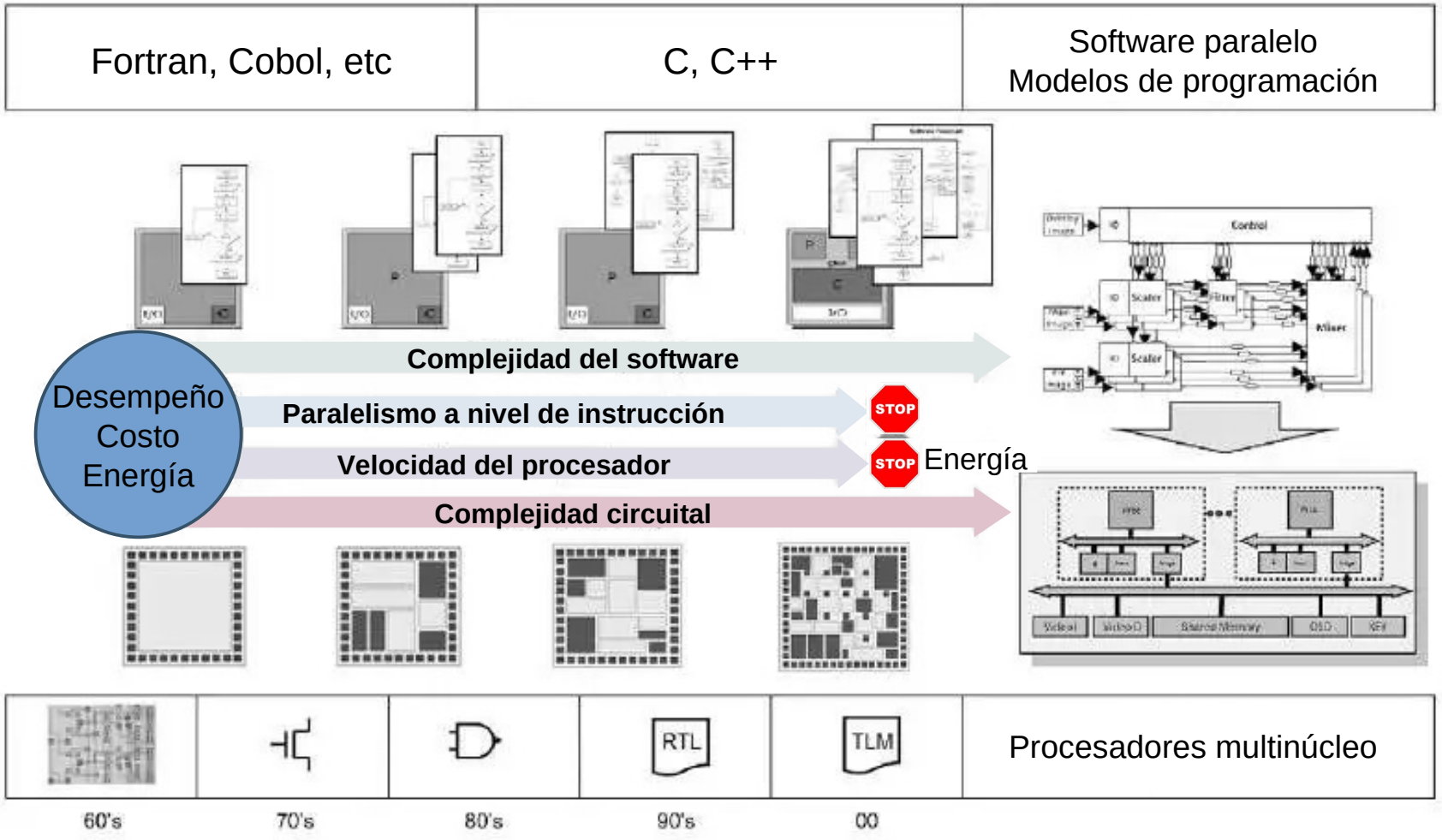
6.1 Evolución de la arquitectura de los procesadores

6.2 Microarquitectura multihilo

6.3 Microarquitectura multinúcleo

Evolución de la arquitectura de los procesadores

Evolución de las computadoras y las tecnologías asociadas



Evolución de los procesadores

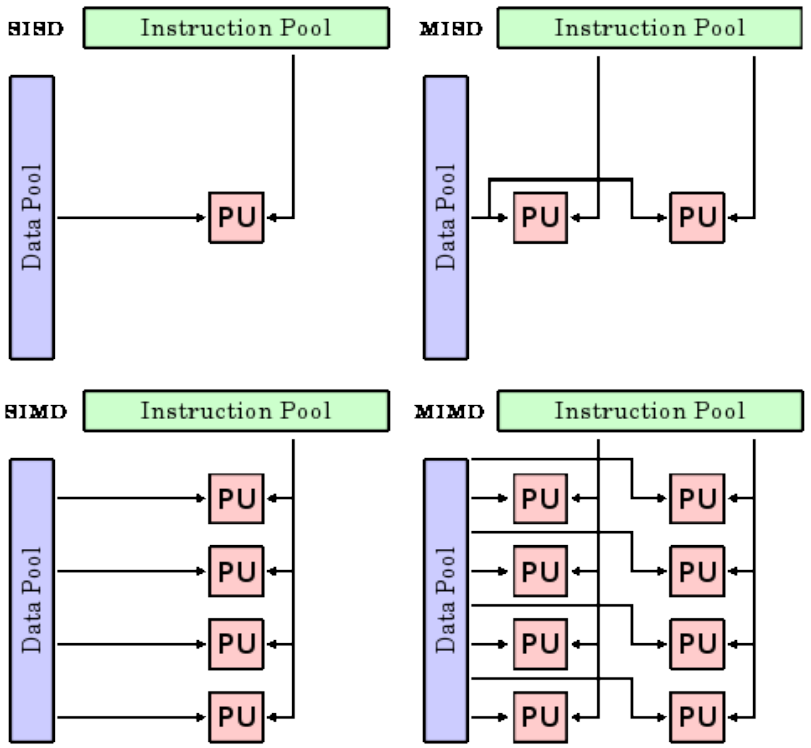
Las arquitecturas se clasifican de acuerdo con el número de instrucciones concurrentes (control) y en los flujos de datos disponibles en:

Una instrucción – un dato (SISD): Computador secuencial que no explota el paralelismo en las instrucciones ni en flujos de datos.

Múltiples instrucción – un dato (MISD): Computador secuencial que explota el paralelismo en las instrucciones.

Una instrucción – múltiples datos (SIMD): Un computador que explota varios flujos de datos dentro de un único flujo de instrucciones para realizar operaciones que pueden ser paralelizadas.

Múltiples instrucciones – múltiples datos (MIMD): Varios procesadores que ejecutan simultáneamente instrucciones diferentes sobre datos diferentes.



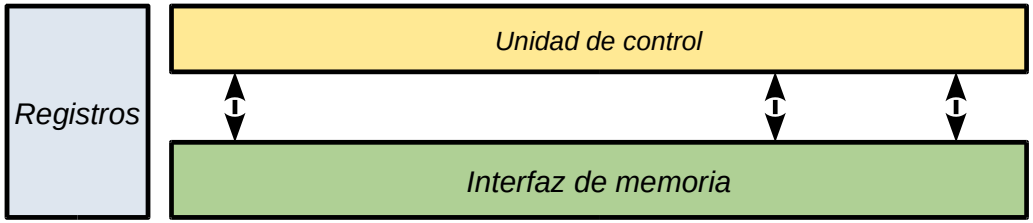
Un programa - múltiples datos (SPMD): Múltiples procesadores autónomos que trabajan simultáneamente sobre el mismo conjunto de instrucciones sobre datos diferentes.

Múltiples programas - múltiples datos (MPMD) - Múltiples procesadores autónomos que trabajan simultáneamente sobre múltiples programas independientes.

Evolución de los procesadores

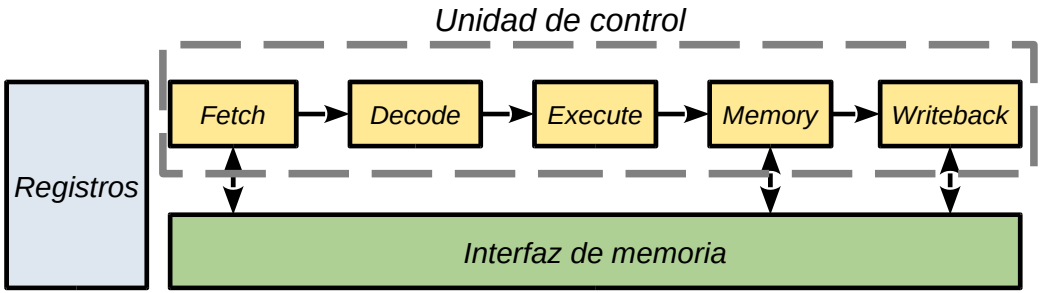
CISC

Instrucciones complejas
Optimizadas para memoria



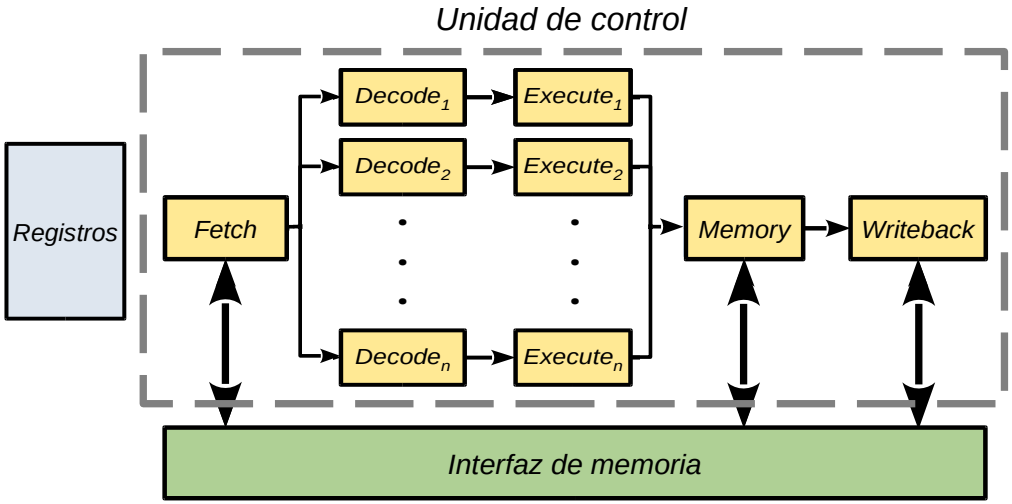
RISC (*microcódigo*)

Instrucciones simples
Optimizadas para velocidad



RISC (*segmentados*)

Instrucciones simples
Optimizadas para ejecución paralela



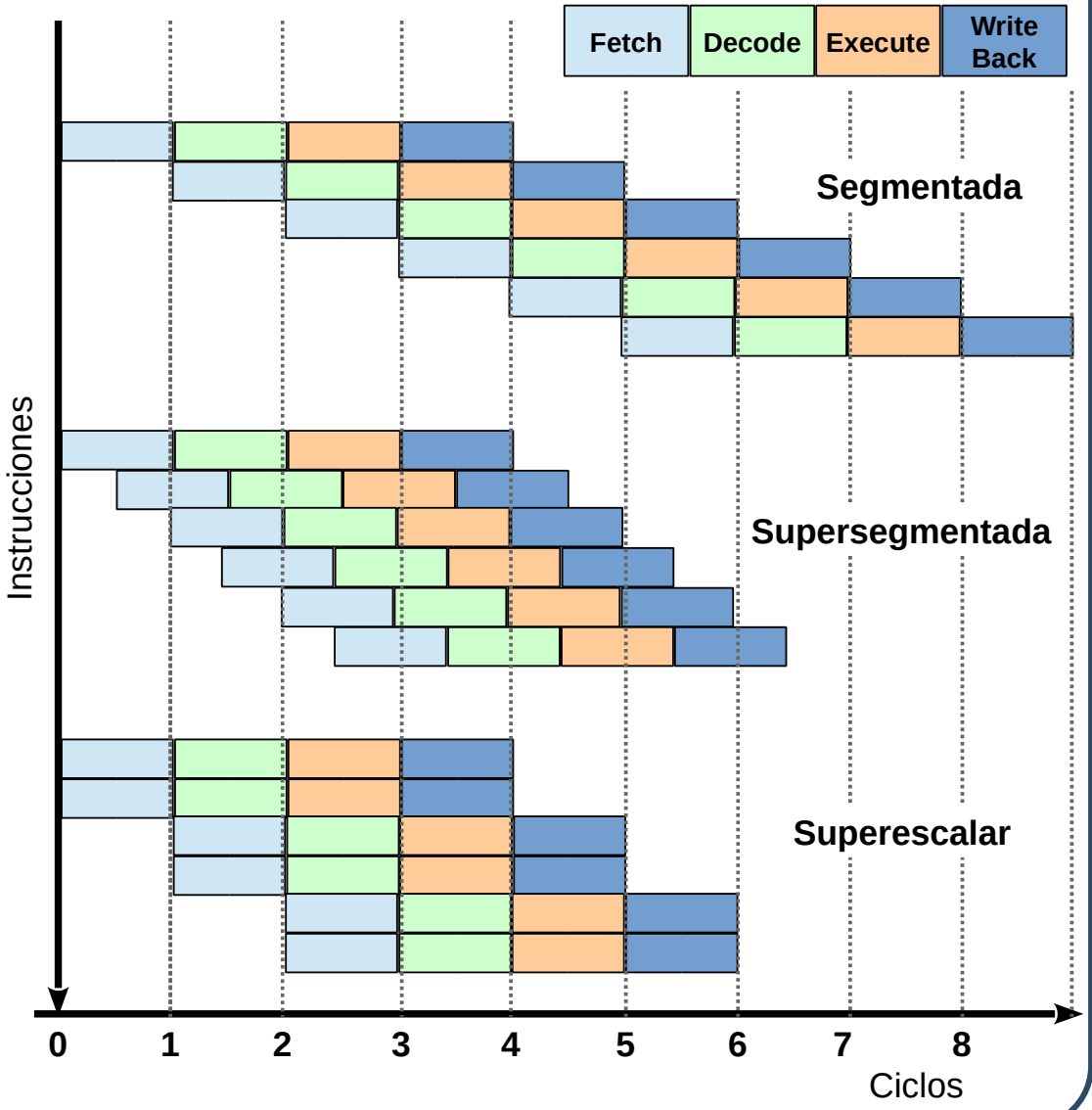
Superescalar

Instrucciones simples
Optimizadas para ejecución simultánea

Evolución de los procesadores

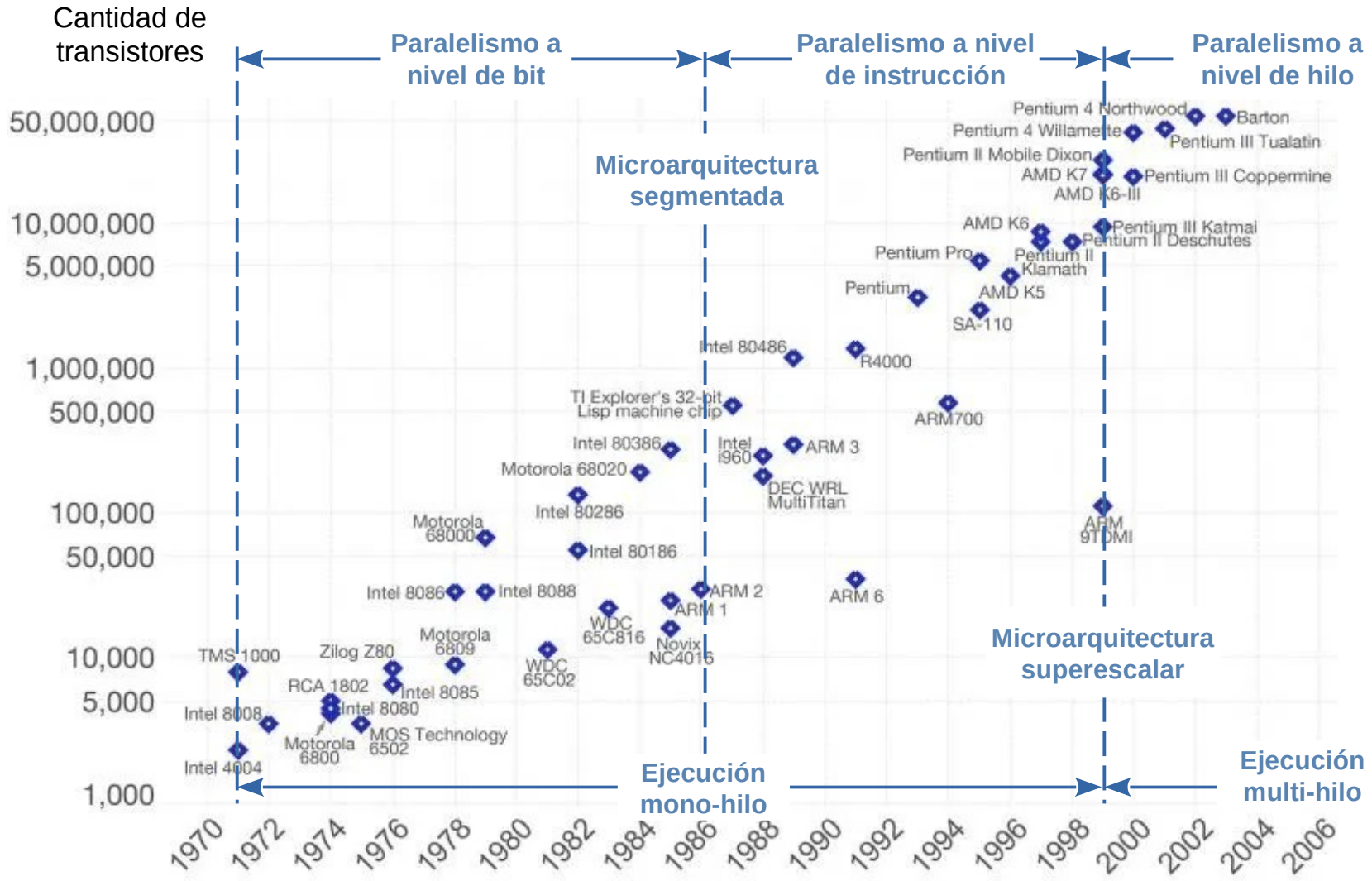
Un procesador superescalar lee múltiples instrucciones al mismo tiempo. Intentando encontrar instrucciones que puedan ejecutarse de manera independiente.

La esencia del enfoque superescalar es la posibilidad de ejecutar instrucciones en paralelo en diferentes datapath.



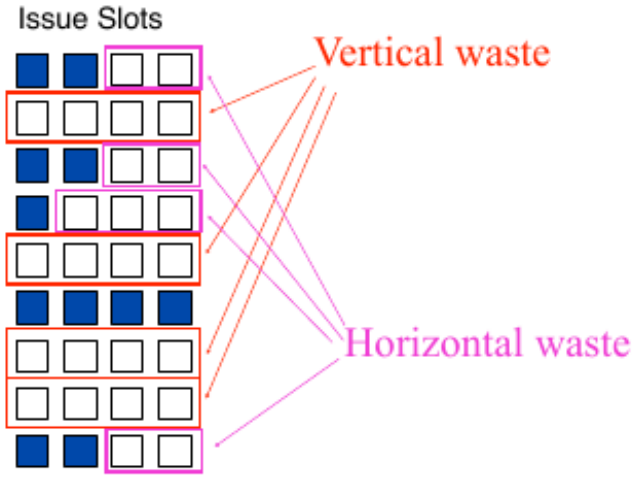
Evolución de los procesadores

Ley de Moore - el número de transistores en un microprocesador se duplica cada dos años.



Evolución de los procesadores

Limitaciones de la microarquitectura superescalar



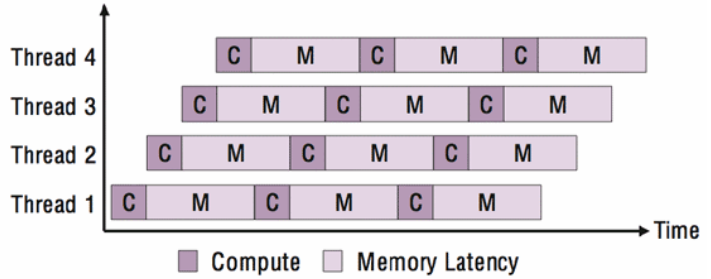
El desempeño de la microarquitectura superescalar está limitada por:

Problema	Soluciones
Fallos en tablas de traducción de instrucciones y datos	Aumentar tamaño de tablas; Aumentar velocidad de las tablas; Incorporar prefetching de instrucciones y datos
Fallos en las cache de instrucciones	Aumentar tamaño y velocidad de la memoria; Incorporar prefetching de instrucciones
Fallos en las cache de datos	Aumentar tamaño y velocidad de la memoria; Incorporar prefetching de datos; Mejorar planificación de ejecución; Mejorar ejecución dinámica
Errores de predicción de saltos	Mejorar esquema de predicción de saltos
Riegos de control	Incorporar ejecución especulativa;
Retardo en accesos a datos e instrucciones desde memoria	Mejorar planificación de ejecución; Mejorar ejecución dinámica
Desbalances de tiempos de ejecución de instrucciones	Mejorar la latencia de las operaciones; Mejorar la planificación de ejecución;
Conflictos al acceder a memoria	Mejorar planificación de ejecución; Mejorar ejecución dinámica

Microarquitectura multihilo

El principal problema de la microarquitectura superescalar es que el uso de ***un único hilo de ejecución*** (thread) lo cual ***limita el paralelismo a nivel de instrucción*** (ILP) por que no se pueden aprovechar todos los recursos disponibles

Generalmente las computadoras ejecutan ***varios programas al mismo tiempo*** (también conocidos como procesos ó hilos) utilizando ***multiplexación de tiempo***.



Las instrucciones de los diferentes hilos no tienen dependencias por lo que se puede aprovechar el **paralelismo de nivel de hilo** para mejorar el rendimiento del procesador a través de la ejecución concurrente de los hilos

En esta implica que el procesador ejecuta instrucciones de varios hilos diferentes de modo que el **procesador físico** actuá como **varios procesadores lógicos**.

Para ello es necesario proveer al procesador físico de

- Múltiples contextos de hardware;
- Capacidad de programar hilos de hardware;
- Capacidad de cambio de contexto; y
- Ocultar eficazmente las latencia largas.

Microarquitectura multihilo

La CPU del CDC 6600 tenía 10 unidades funcionales paralelas (PP), que proveian de datos a la ALU, permitiendo la ejecución de múltiples instrucciones (arquitectura superescalar).

Las unidades funcionales fueron diseñados para acceder a la memoria durante los momentos en que la CPU estaba ocupada.

Esto permite realizar entrada/salida sin asignar tiempo de procesamiento, manteniendo la CPU ocupada tanto como fuera posible.

Los PPs se implementaron virtualmente; había un único PP implementado en hardware. Este hardware se compartía y operaba con 10 conjuntos de registros que representaban cada uno de los estados de cada PP.

CDC 6600 (Cray, 1964)

La unidad de control conectaba un conjunto de registros de un PP (estados) a la unidad funcional física, de modo que ejecute una instrucción (o parte de ella), hasta que la unidad de control conmutara a otro conjunto de registros (estado) del PP.

Microarquitectura multihilo

Implementación

Modificaciones necesarias

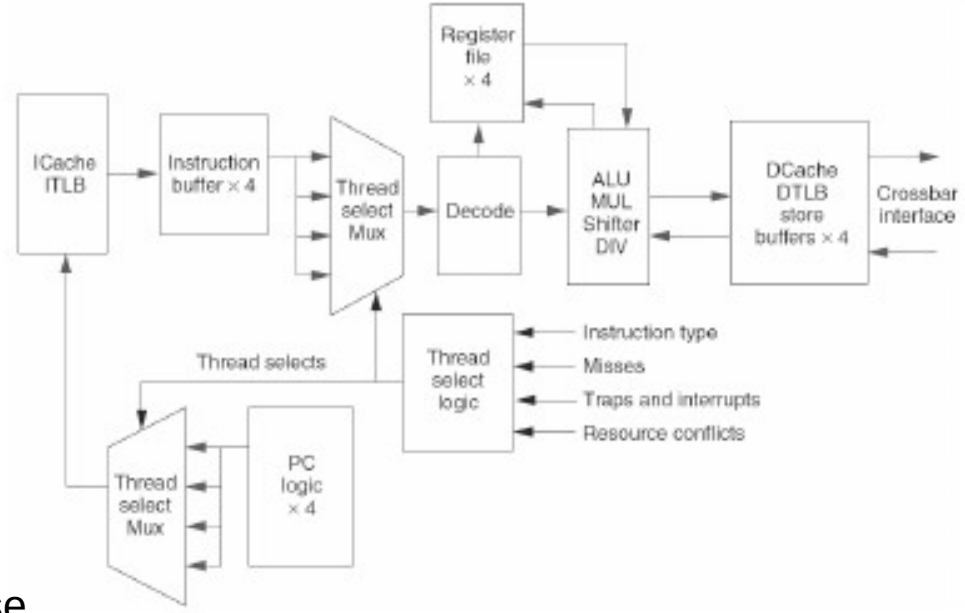
- Múltiples contadores de programas, registros y un mecanismo mediante el cual una unidad de búsqueda selecciona uno hilo en cada ciclo (política de fetch/issue);
- Una pila de retorno separada para cada hilo para predecir las direcciones de retorno de cada subrutina;
- Mecanismos de fetch/issue de instrucciones, vaciado de la cola de instrucciones y mecanismos de captura para cada hilo; y
- Identificador de hilo por cada entrada del buffer de destino de salto (BTB) para evitar la predicción de saltos fantasmas.

Modificaciones para mejorar el desempeño

- Un registro de renombrado más grande, para admitir registros lógicos para todos los hilos y etapas adicionales en el datapath;
- Mayor ancho de banda en las transacciones con la memoria principal;
- Un buffer de traducción de datos (TLB) más grande para compensar el aumento de las traducciones de direcciones; y
- Una caché mejorada para compensar la degradación de su rendimiento debido al uso compartido entre los hilos, y la localidad reducida resultante.

Microarquitectura multihilo

Implementación

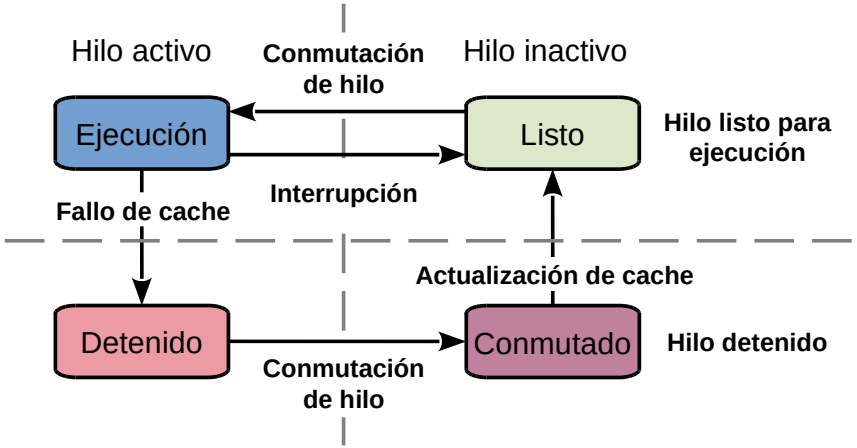


Estas ideas conducen a la siguiente microarquitectura

La pregunta que se necesita responder es ***cuando y como se asignan*** los recursos del procesador a los hilos disponibles.

Hay dos opciones para

- **Multihilo temporal** – Los recursos se asignan a cada hilo durante un periodo de tiempo (fijo o variable) dependiendo de los conflictos de recursos y dependencias de datos (***multihilo grueso*** y ***multihilo entrelazado*** *ó* ***fino***); y



- **Multihilo simultáneo** – Los recursos se asignan, en cada ciclo de operación, a los hilos que pueden ejecutarse en función de los recursos disponibles y las dependencias de datos.

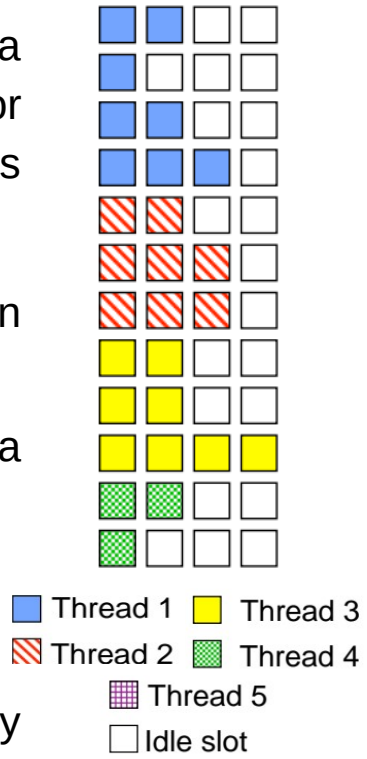
Microarquitectura multihilo

Multihilo grueso

La implementación más simple del multihilo temporal es el ***multihilo grueso***, se basa en:

Ejecutar un hilo hasta que es bloqueado por un evento de latencia prolongada y conmutar a un hilo que esté listo para ejecutarse

Por ejemplo, un fallo de caché necesita cientos de ciclos para continuar la ejecución. En lugar de esperar a que se resuelva el bloqueo, el procesador conmuta la ejecución a un hilo listo para ejecutarse. Cuando los datos están disponibles, se vuelve a colocar en la lista de ejecución.



Hay muchas variaciones posibles del multihilo grueso, las cuales están relacionadas con el algoritmo de conmutación de hilo.

Este algoritmo se basa en múltiples factores, incluidos el tiempo de latencia y fallos de cache, entre otros.

Su principal desventaja son las pérdidas de rendimiento debido conmutaciones debidas a bloqueos cortos:

- Cuando se produce un bloqueo, el datapath debe vaciarse o congelarse; y
- El nuevo hilo debe llenar el datapath antes de que se completen las instrucciones.

Por ello, el multihilo grueso es mejor para reducir la penalización de las paradas cuando

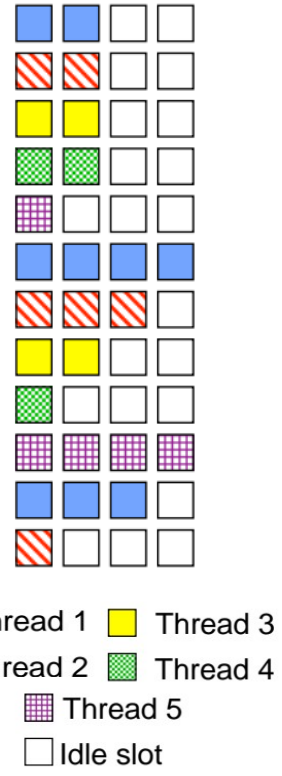
Inicialización de datapath << tiempo de parada

Microarquitectura multihilo

Multihilo entrelazado

El ***múltiple-hilo entrelazado*** elimina los efectos de las dependencias de dato ya que los hilos son independientes. Se basa en

Intercalar la ejecución de diferentes hilos de manera rotatoria, omitiendo los hilos bloqueados



El datapath del procesador contiene múltiples hilos y cambios de contexto que ocurren entre las etapas del datapath.

Este tipo de procesamiento se conoce como ***barrel processing***, pero actualmente se lo denomina ***multi-hilo entrelazado*** o ***fino***.

Esta implementación es más costosa porque ***los recursos tienen que gestionarse simultáneamente*** y cada etapa del datapath tiene que ***rastrear el ID de hilo*** de la instrucción que está ejecutando.

Además, dado que los hilos que se ejecutan simultáneamente, los recursos compartidos (caches y TLB, etc.) deben ser más abundantes para evitar que se produzcan bloqueos y errores.

Su principal desventaja es que ***ralentiza la ejecución de hilos individuales***, ya que un hilo listo para ejecutarse sin paradas se retrasará por instrucciones de los otros hilos.

Microarquitectura multihilo

Multihilo simultáneo

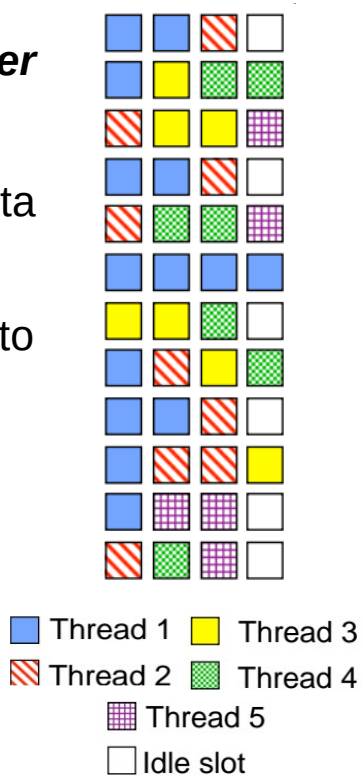
Como los hilos tienen una cantidad limitada de paralelismo a nivel de instrucción, el ***multihilo simultáneo*** explota el paralelismo disponible en varios hilos. Se basa en

Instrucciones de diferentes hilos se pueden ejecutar en cualquier etapa del datapath simultáneamente

A la arquitectura del procesador se le agrega: la ***capacidad de obtener instrucciones de varios hilos*** y un ***conjunto de registros más grande***.

Debido al ***aumento de los conflictos*** (en recursos compartidos) resulta difícil medir su eficacia, sin embargo su ***eficiencia energética es muy alta***.

Además, permite ocultar la latencia de la memoria y aumentar el rendimiento de los cálculos.



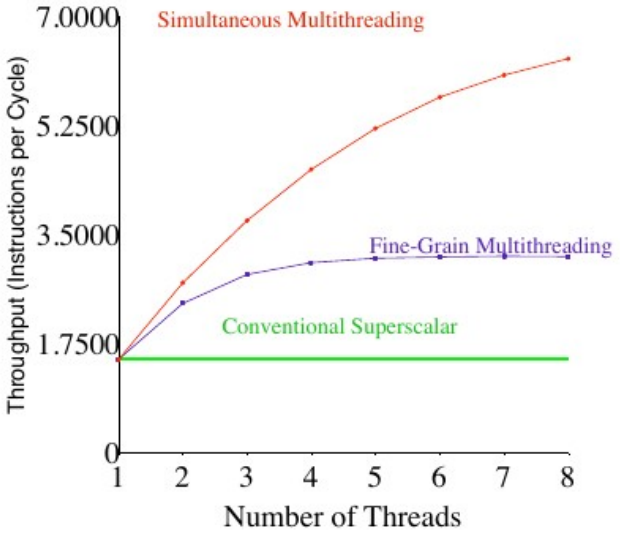
Las principales ventajas de este esquema son:

- Supera las limitaciones producidas por el bajo ILP de un solo hilo; y
- Oculta los riesgos de control y otras latencias prolongadas.

Sus principales desventajas son:

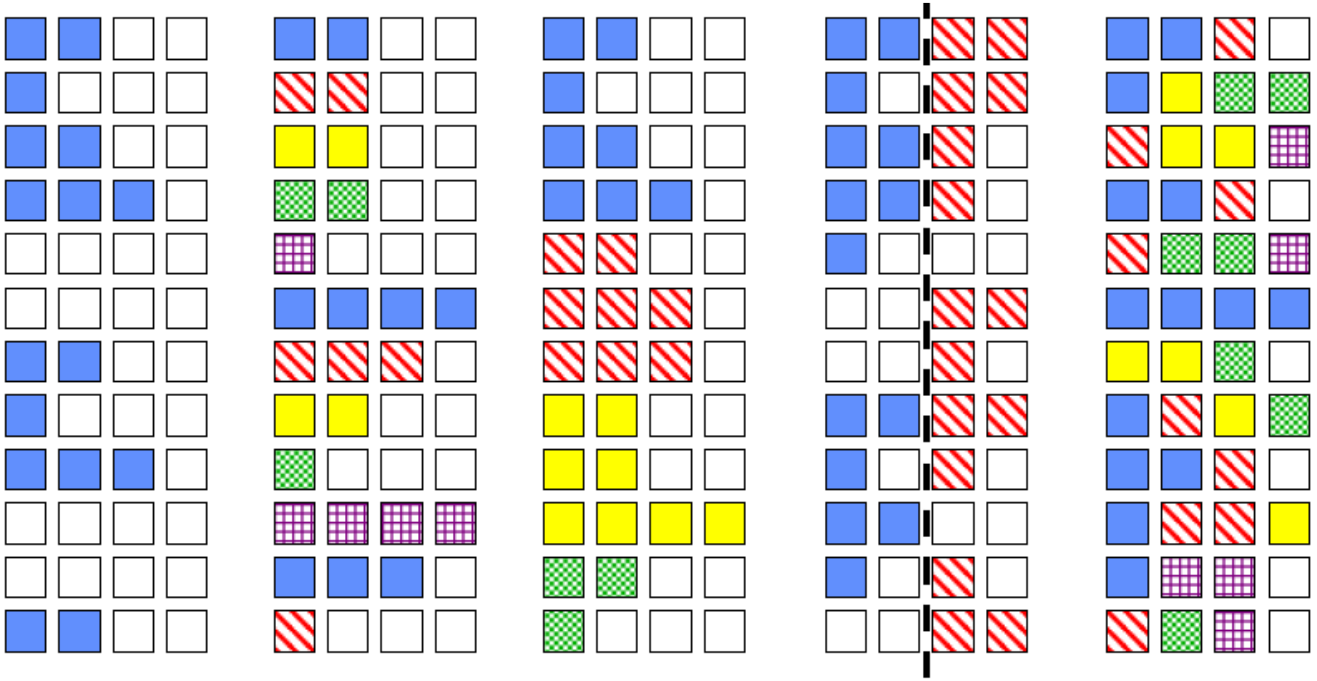
- Sobrecarga de trabajo a la jerarquía de la memoria;
- Aumento de la complejidad de la unidad de control; y
- Aumento de la cantidad de los recursos (caché, predicción de rama, TLB, etc.)

Microarquitectura multihilo



Thread 1 Thread 3 Thread 5
Thread 2 Thread 4 Idle slot

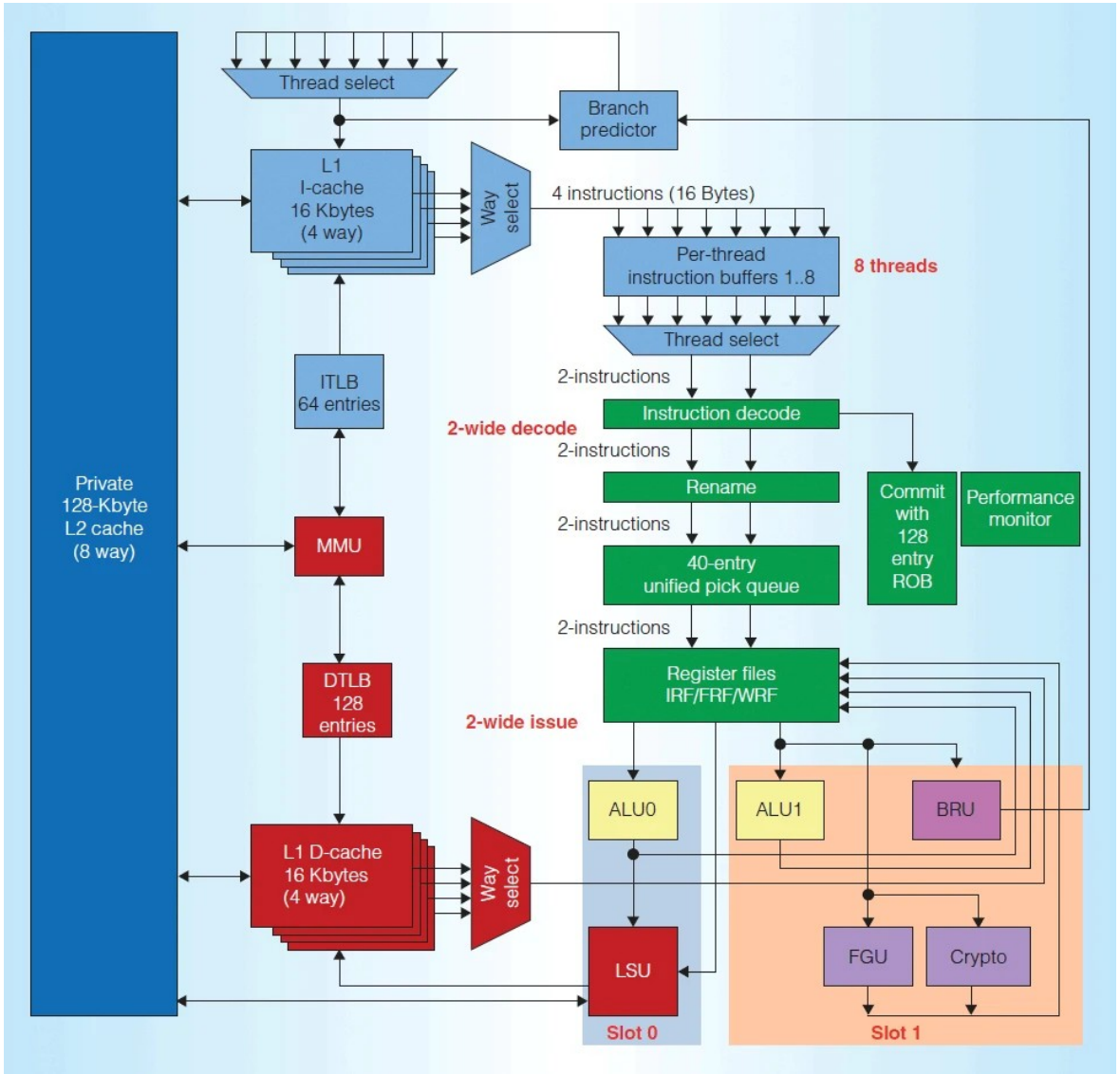
Superescalar | Multihilo entrelazado | Multihilo grueso | Multiproceso | Multihilo simultáneo



Microarquitectura multihilo

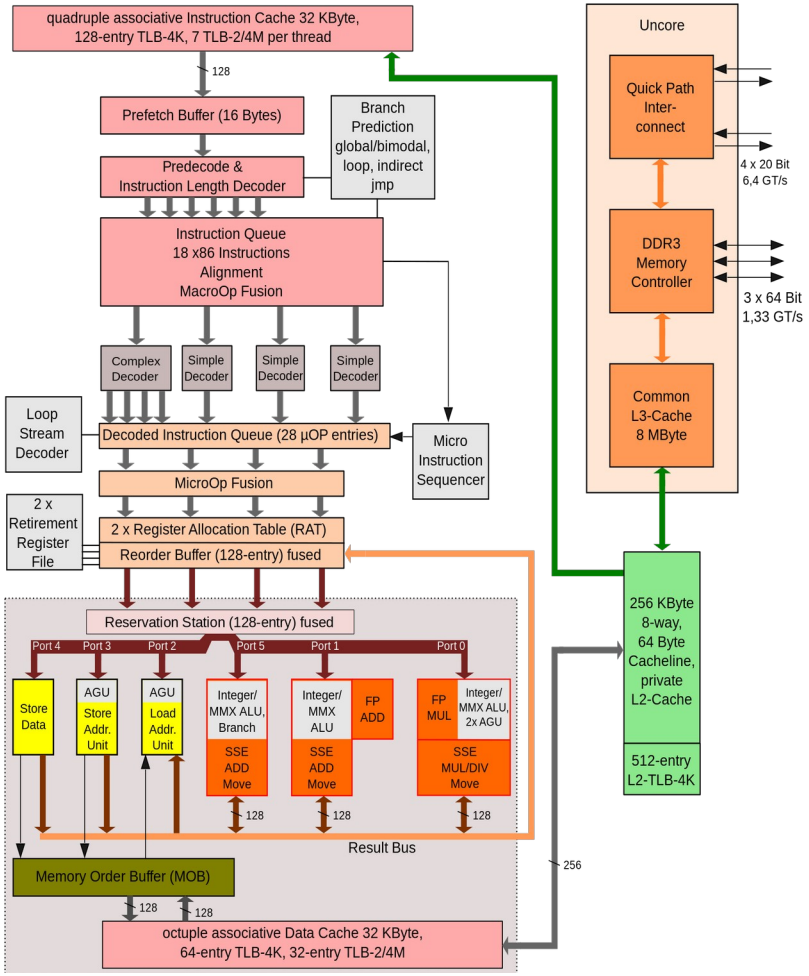
Multihilo grueso

Oracle Sparc S3

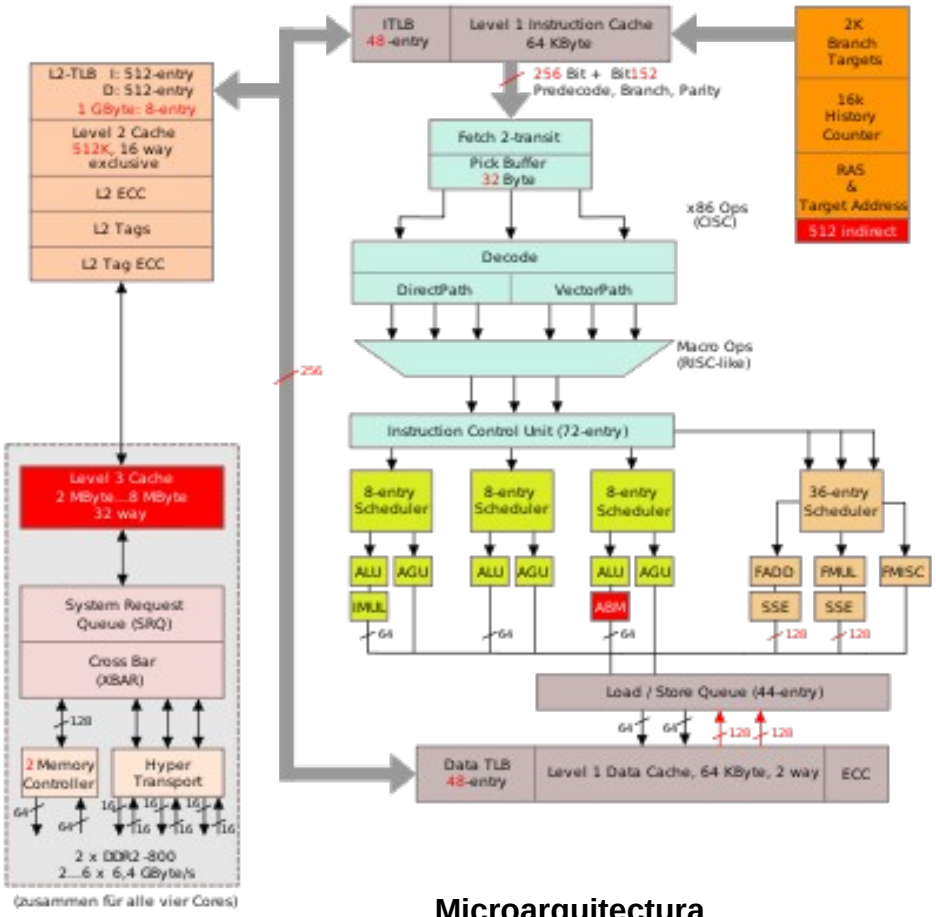


Microarquitectura multihilo

Procesadores comerciales



Microarquitectura
Intel - Nehalen

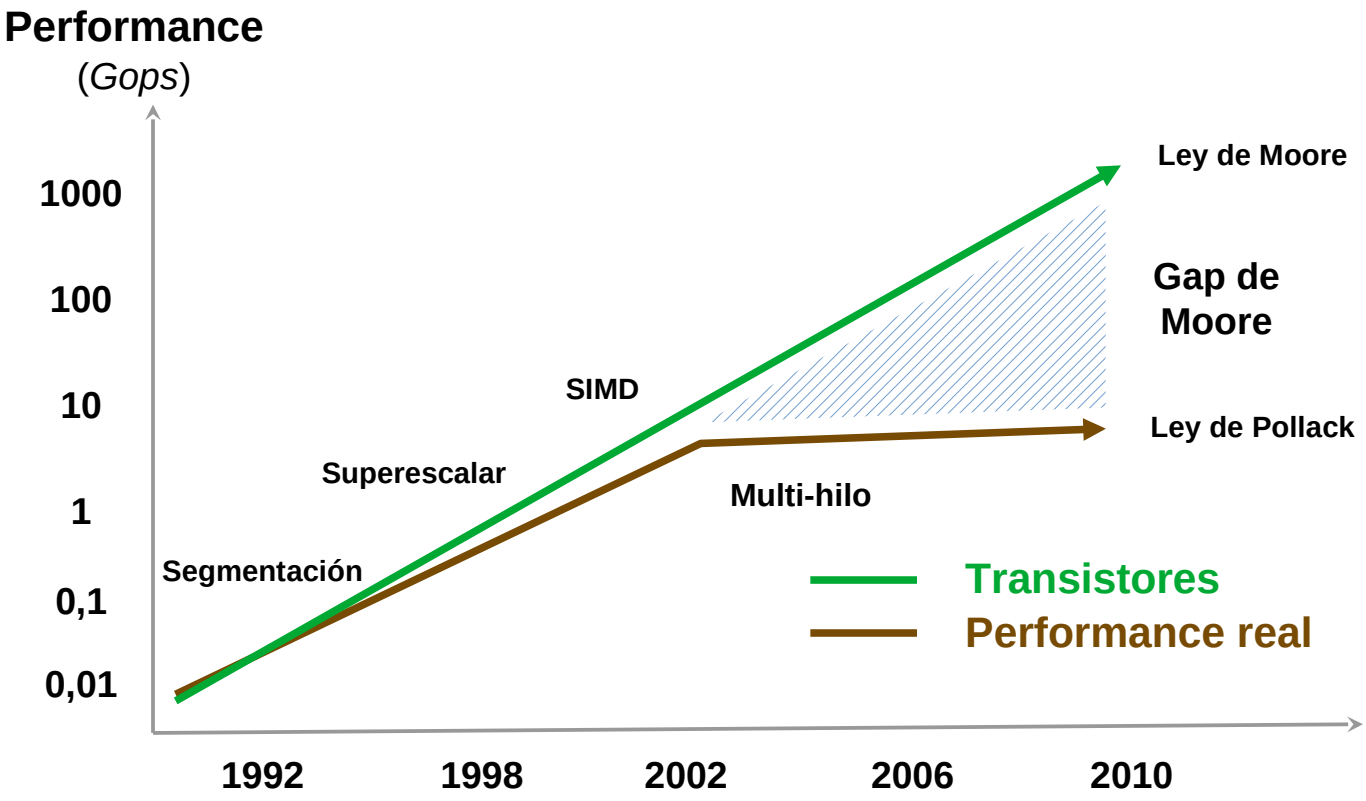


Microarquitectura
AMD - K10

Evolución de los procesadores

Ley de Moore - el número de transistores en un microprocesador se duplica cada dos años.

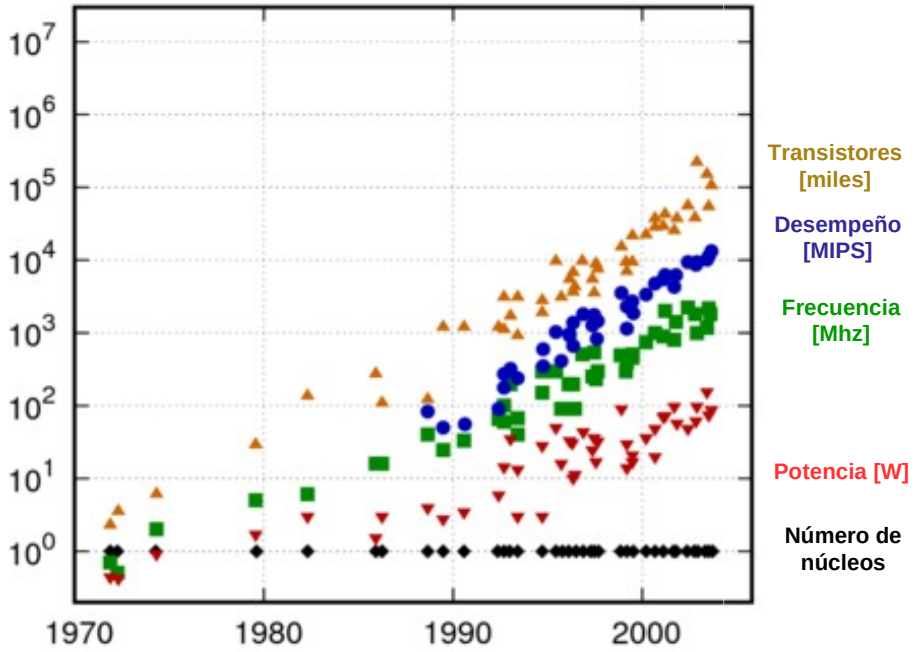
Ley de Pollack - el aumento del rendimiento debido a los avances de la microarquitectura proporcional a la raíz cuadrada del aumento de la complejidad (o cantidad de transistores).



Evolución de los procesadores

El ***gap de Moore*** se produce por que la mayor parte del rendimiento que se puede obtener en una microarquitectura mono-hilo ya ha sido aprovechada (paralelismo a nivel de instrucción; accesos a memoria; etc.).

Para aumentar el rendimiento de una microarquitectura mono-hilo hay que implementar técnicas más complejas (para obtener mejoras cada vez menores).



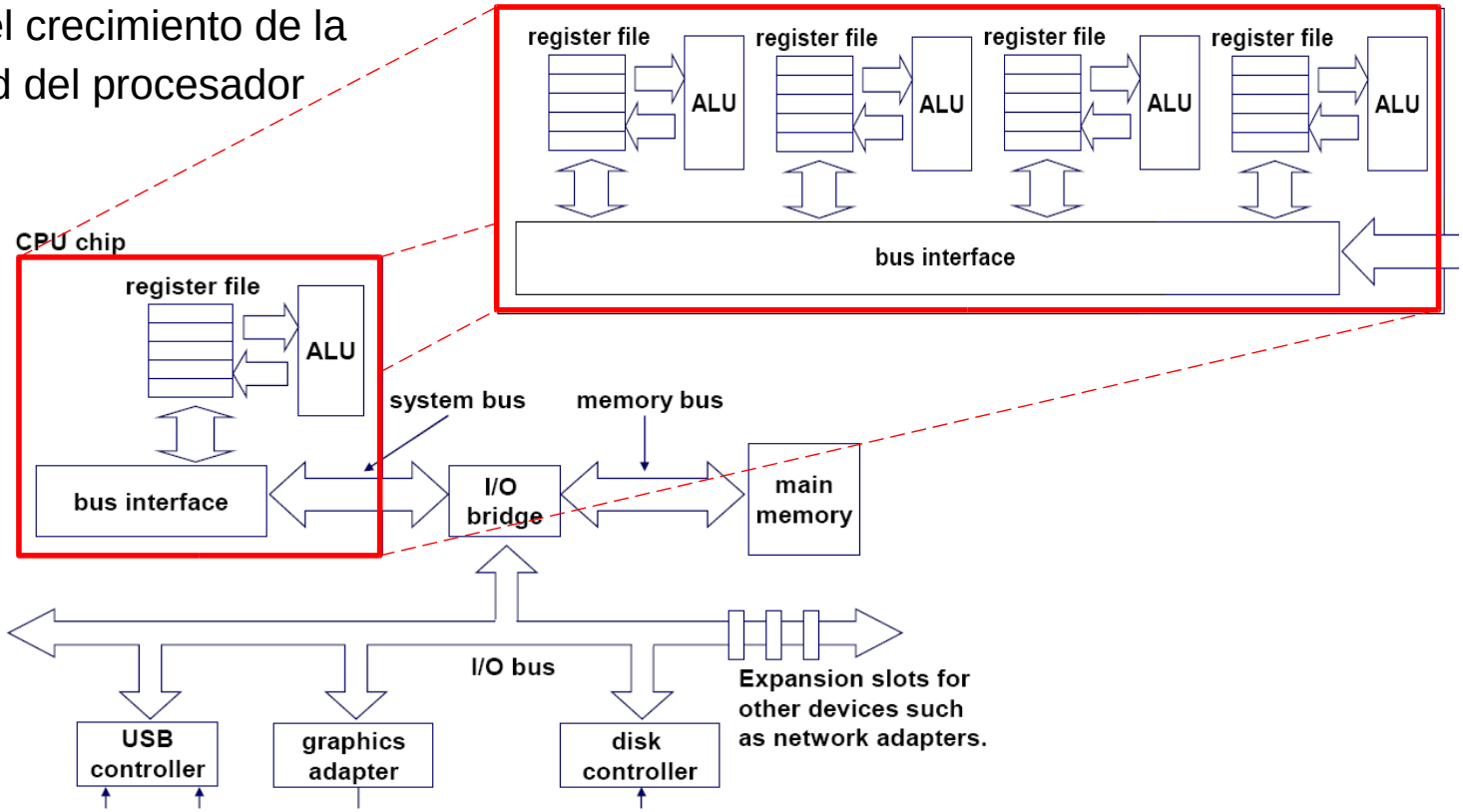
Ello conduce a

- Aumento de la frecuencia de operación;
- Aumento de la ***cantidad de transistores*** utilizados;
- Aumento de la ***potencia consumida***, y a disipar, del procesador; y
- Aumento de la ***temperatura de operación***.

Microarquitectura multinúcleo

Esto condujo al desarrollo de la arquitectura multinúcleo, la cual

- Aprovecha el aumento de la densidad y el tamaño de las funciones;
- Aumenta la cantidad de unidades funcionales por chip (eficiencia espacial);
- Limita el consumo de energía del procesador;
- Restringe el crecimiento de la complejidad del procesador



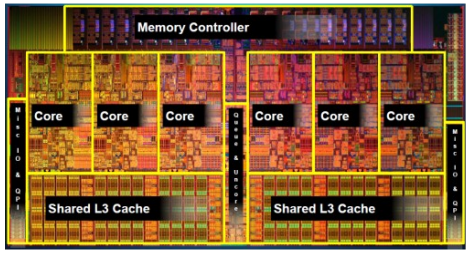
Microarquitectura multinúcleo

Un ***procesador multinúcleo*** es un sistema de procesamiento ***compuesto por varios núcleos*** (o CPUs) ***independientes***.

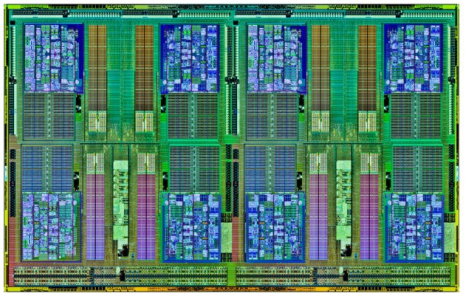
Los núcleos se integran en un solo circuito integrado, o múltiples circuitos integrados (dies) empaquetados en un solo chip.

Los núcleos se pueden acoplar entre sí de forma fuerte o débil, compartiendo cache, implementando métodos de comunicación basados en paso de mensajes o memoria compartida.

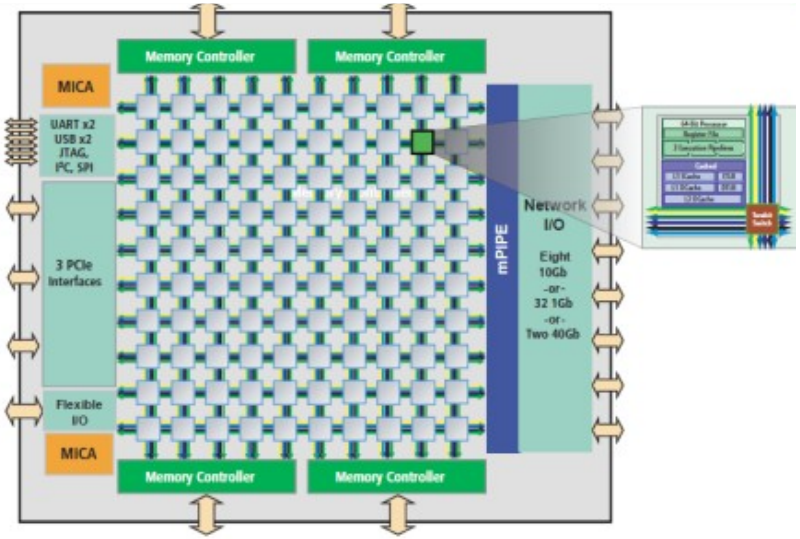
Las topologías de red de interconexión incluyen: bus, anillo, malla bidimensional y barra transversal.



Intel Core i7 - 8 Núcleos



Rizen - 32 Núcleos

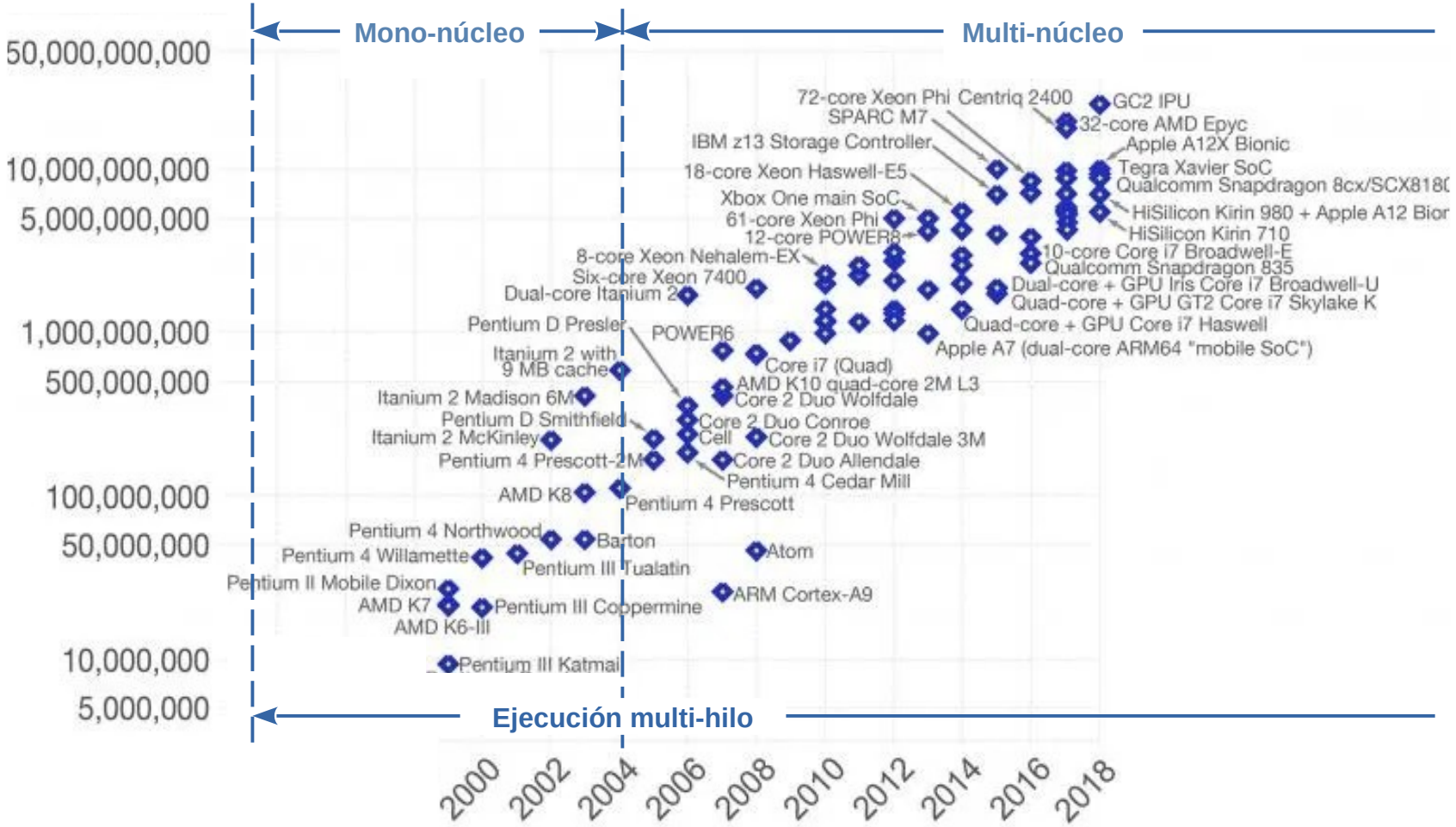


Tile Gx - 100 Núcleos

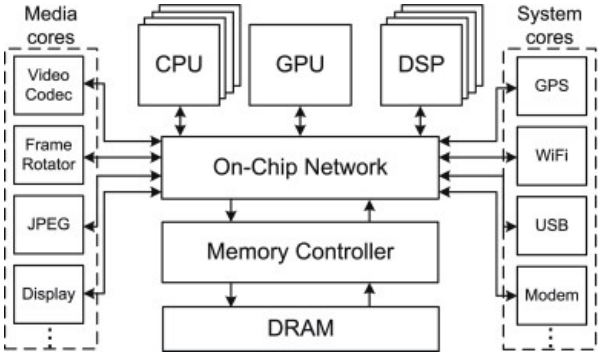
Los núcleos de los sistemas multinúcleo pueden implementar arquitecturas superescalar, procesamiento vectorial o multiproceso.

Un ***procesador de muchos núcleos*** es aquel en el que la cantidad de núcleos es tan grande que las técnicas de procesamiento paralelo ya no son eficientes y se ***requiera de una red de comunicaciones*** en el chip para operar eficientemente.

Microarquitectura multinúcleo



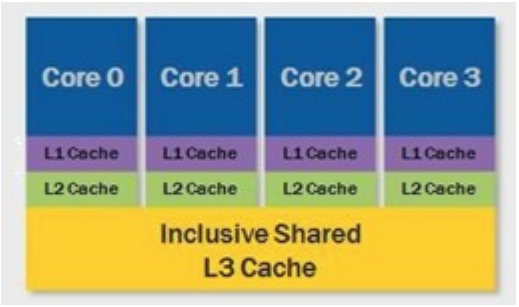
Microarquitectura multinúcleo



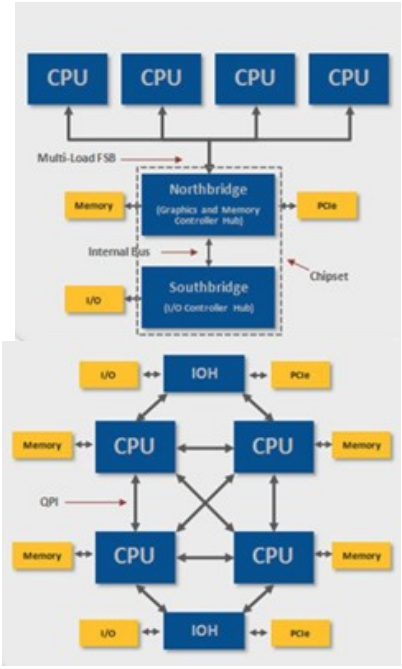
Al intentar clasificar los procesadores multinúcleo, hay que considerar la aplicación en sí misma y los siguientes elementos:

El tipo de elementos de procesamiento implementados en el procesador, los cuales determinan la elección de las herramientas específicas para programar la arquitectura específica.

Los **sistema de comunicación** entre los núcleos, que determina cuánto tiempo deben esperar los núcleos antes de poder acceder a los datos.



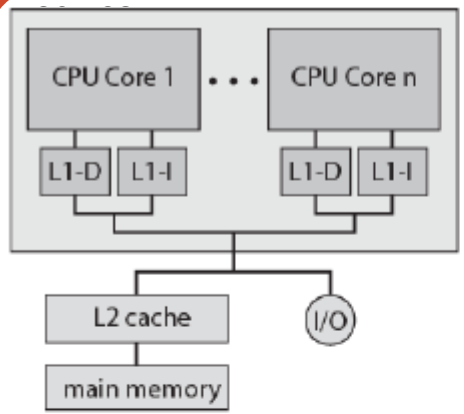
La **arquitectura de memoria** que se utiliza el chip, la cual tiene un impacto profundo en las latencias en los procesos que acceden a los datos.



La **optimización de la arquitectura** de hardware para la aplicación que se ejecuta. Esto tiene un gran impacto en la complejidad de la programación y el desempeño del sistema.

Microarquitectura multinúcleo

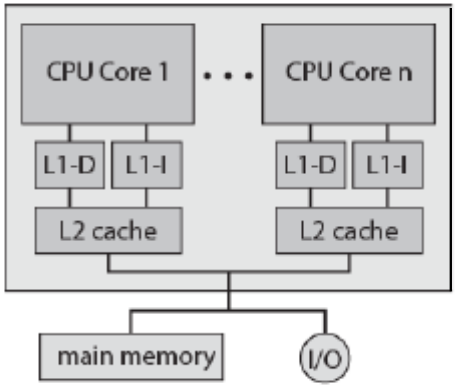
Arquitectura de memoria



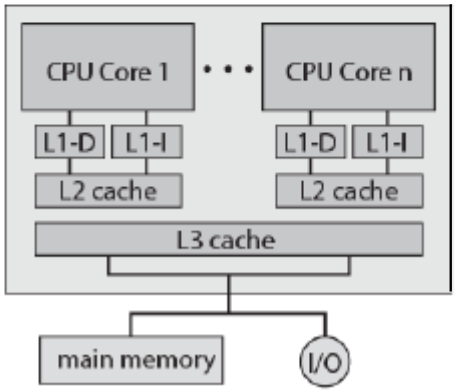
Las arquitecturas multinúcleo organizan la memoria de dos maneras:

Memoria compartida – la memoria es única y está compartida por todos los procesadores;

Memoria distribuida - cada procesador tiene su propia memoria local y sus contenidos no están replicados.



La cache de **primer nivel** siempre es **dedicada a cada núcleo**, porque se debe acceder rápido a los datos, las caches de **nivel intermedio pueden ser compartidas** entre todos los núcleos, mientras que las de **último nivel siempre son compartidas**.



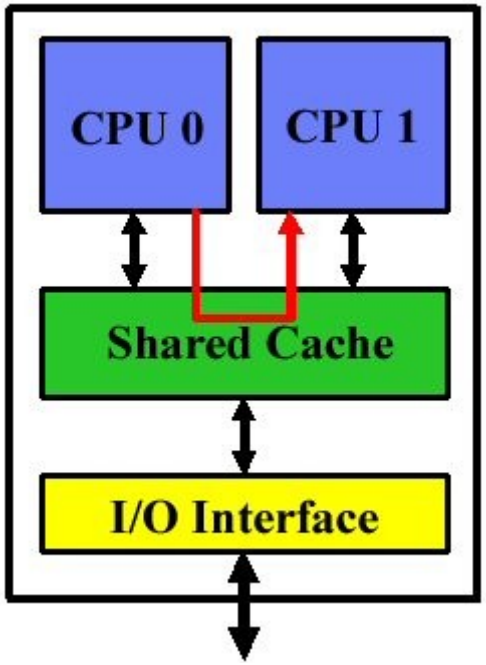
Las ventajas de compartir memoria son:

- La interferencia constructiva reduce la tasa de errores;
- Los datos compartidos no se repiten;
- La memoria se asigna de forma dinámica a cada núcleo;
- Facilita la comunicación entre procesos;
- La coherencia es confinada a niveles inferiores;
- Facilita el acceso a los datos;
- Mejoran el desempeño general del sistema.

Microarquitectura multinúcleo

Arquitectura de memoria

Memoria compartida
Interfaz compartido

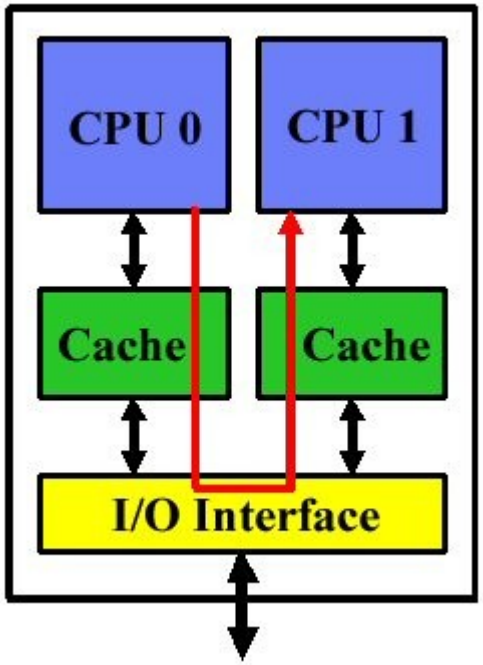


Los núcleso se comunican a través de la memoria compartida

Latencia de comunicación baja

Intel Pentium Core Duo, Quad Core AMD K10

Memoria local
Interfaz compartida

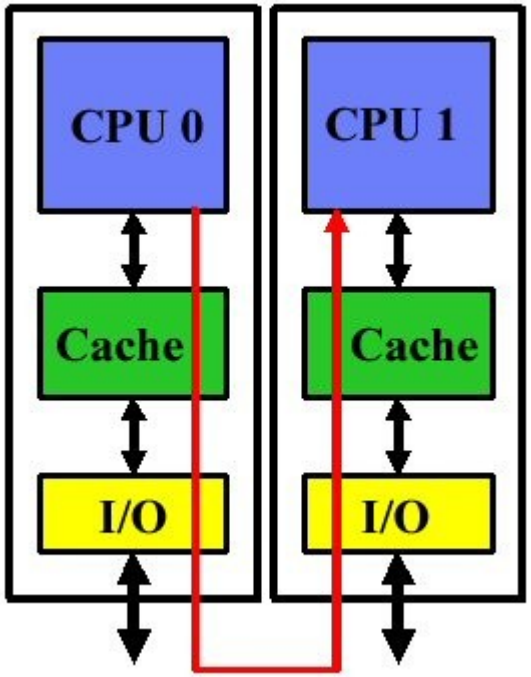


Los núcleso se comunican a través de la interfaz compartida

Latencia de comunicación media

Intel Itanium 2, Dual Core Opteron, Athlon 64

Memoria local
Interfaz local



Los núcleso se comunican a través de la interfaz externa

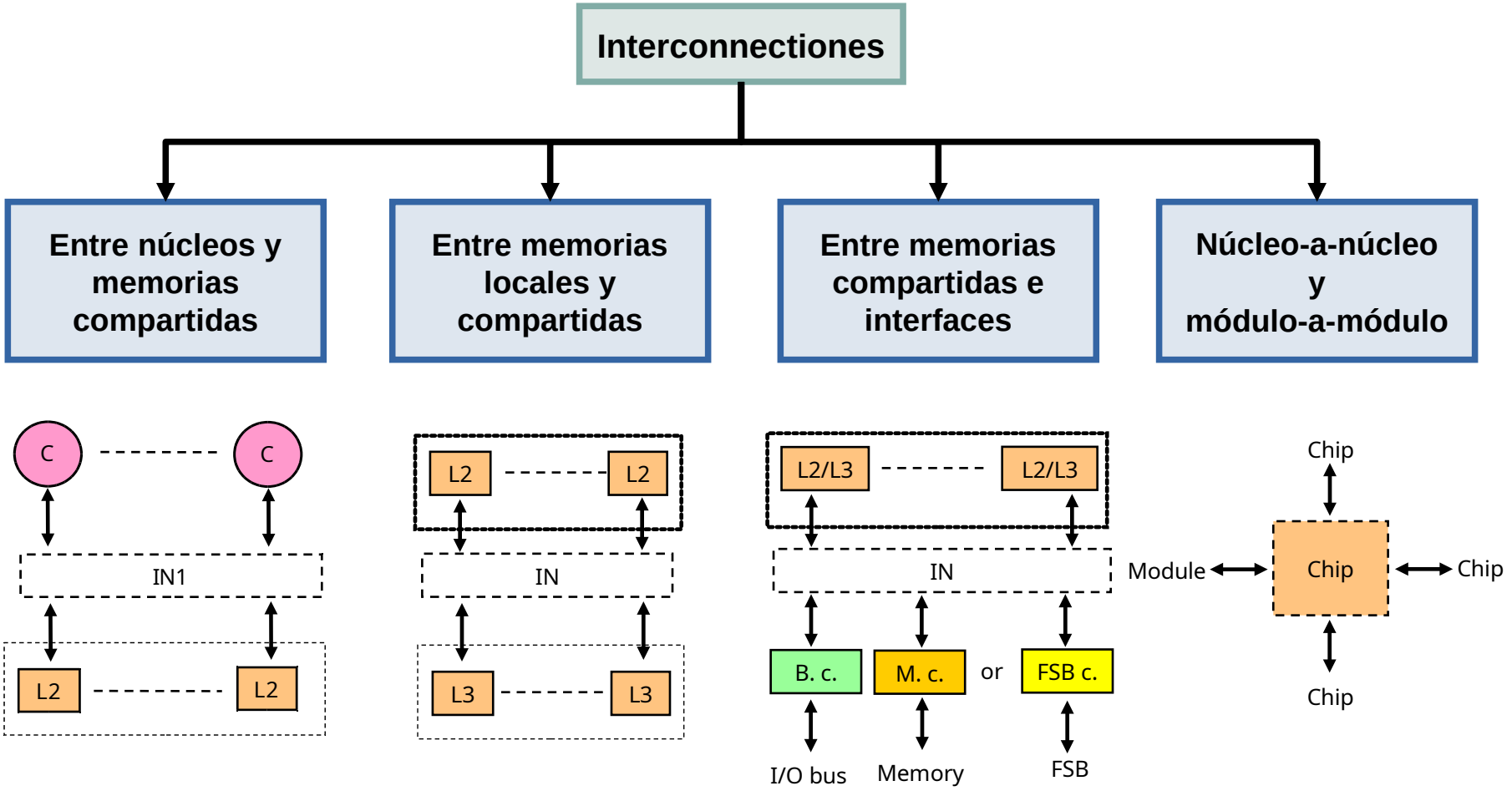
Latencia de comunicación alta

Intel Pentium D

Microarquitectura multinúcleo

Sistema de comunicación

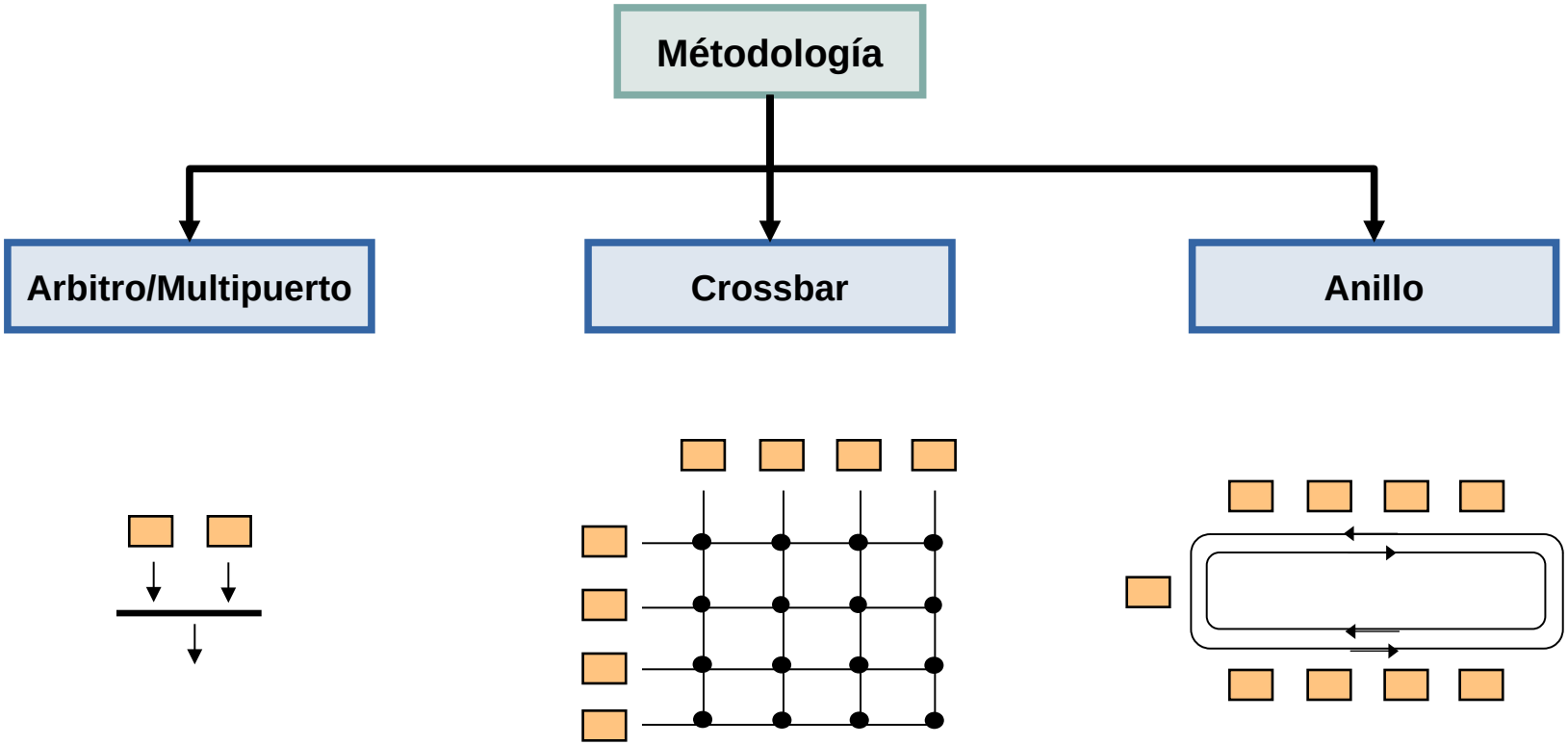
Todos los núcleos son idénticos en los sistemas simétricos de varios núcleos y no son idénticos en los sistemas asimétricos de varios núcleos.



Microarquitectura multinúcleo

Sistema de comunicación

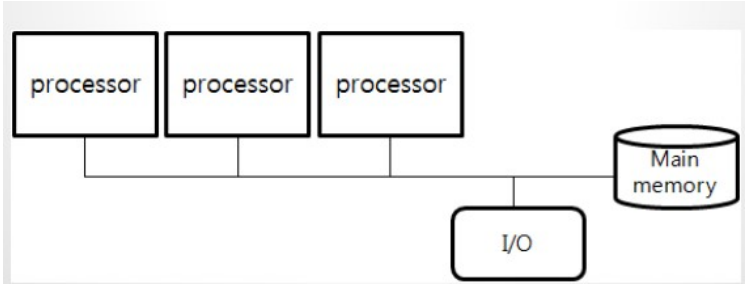
Las topologías de red comunes para interconectar núcleos incluyen: bus, anillo, malla bidimensional y barra transversal.



Microarquitectura multinúcleo

Elementos de procesamiento (tipos de núcleos)

Según los elementos de procesamiento utilizados, una microarquitectura multinúcleo se puede clasificar en

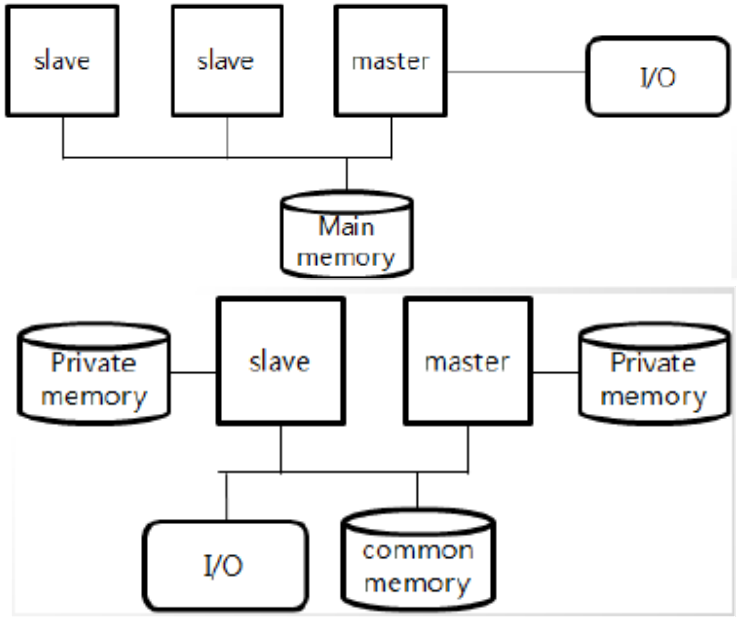


Arquitectura homogénea - todos los elementos de procesamiento son iguales.

- Los procesadores so idénticos y de propósitos generales;
- Cualquier tarea puede ser asignada a cualquier procesador.

Arquitectura heterogénea - los elementos de procesamiento son diferentes.

- Cada procesador está diseñado para tareas específicas;
- Consiste en un procesador maestro, que controla la operación, y procesadores dedicados (esclavos);
- Los procesadores dedicados pueden tener memorias locales y estar especializados en funciones específicas;
- Los programadores deben entender la distribución de tareas en los procesadores,y crear comunicaciones eficientes.

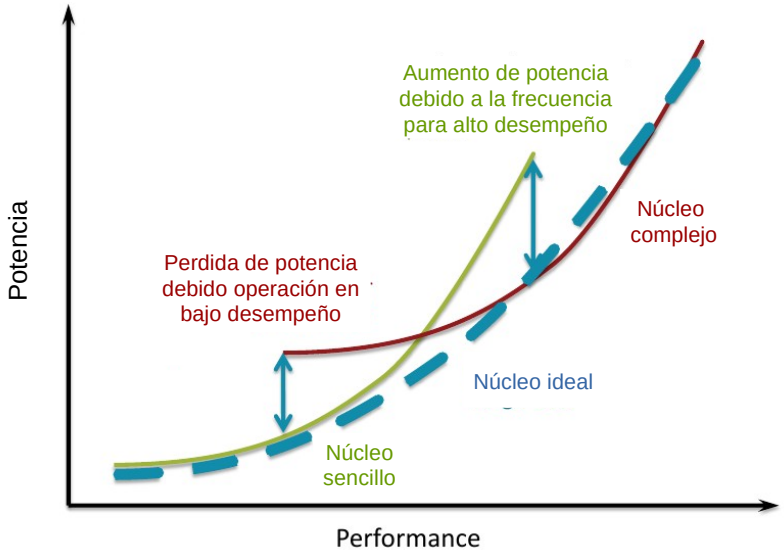


Microarquitectura multinúcleo

Elementos de procesamiento (tipos de núcleos)

Advantages of Asymmetric

+ Can provide better performance when thread parallelism is limited
+ Can be more energy efficient

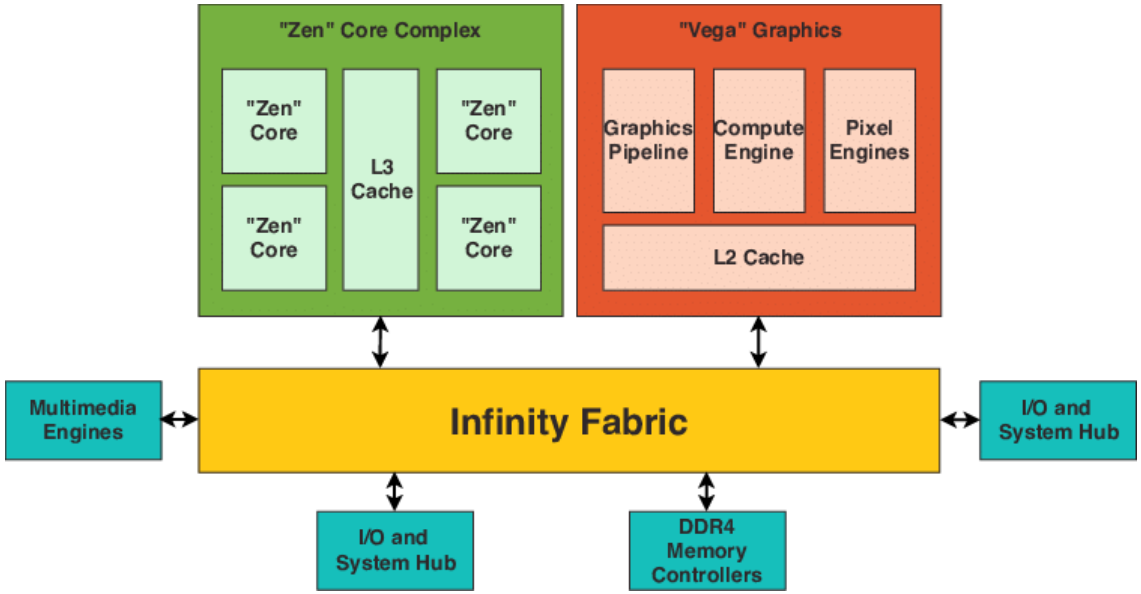


Disadvantages

- Need to design more than one type of core. Always?
- Scheduling becomes more complicated
- Managing locality and load balancing can become difficult if threads move between cores (transparently to software)
- Cores have different demands from shared resources

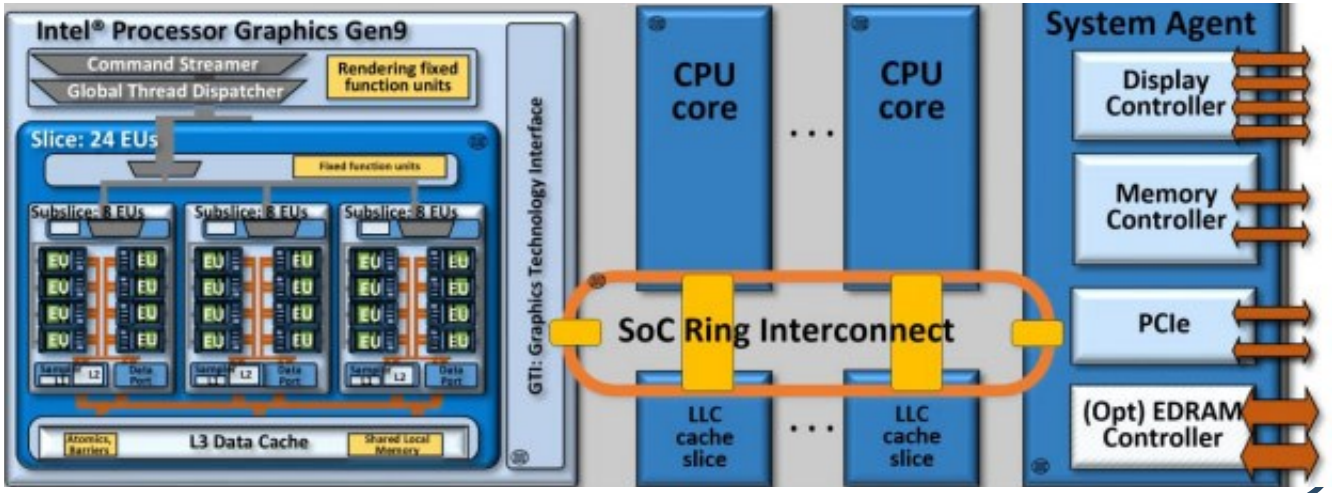
Microarquitectura multinúcleo

Ejemplos



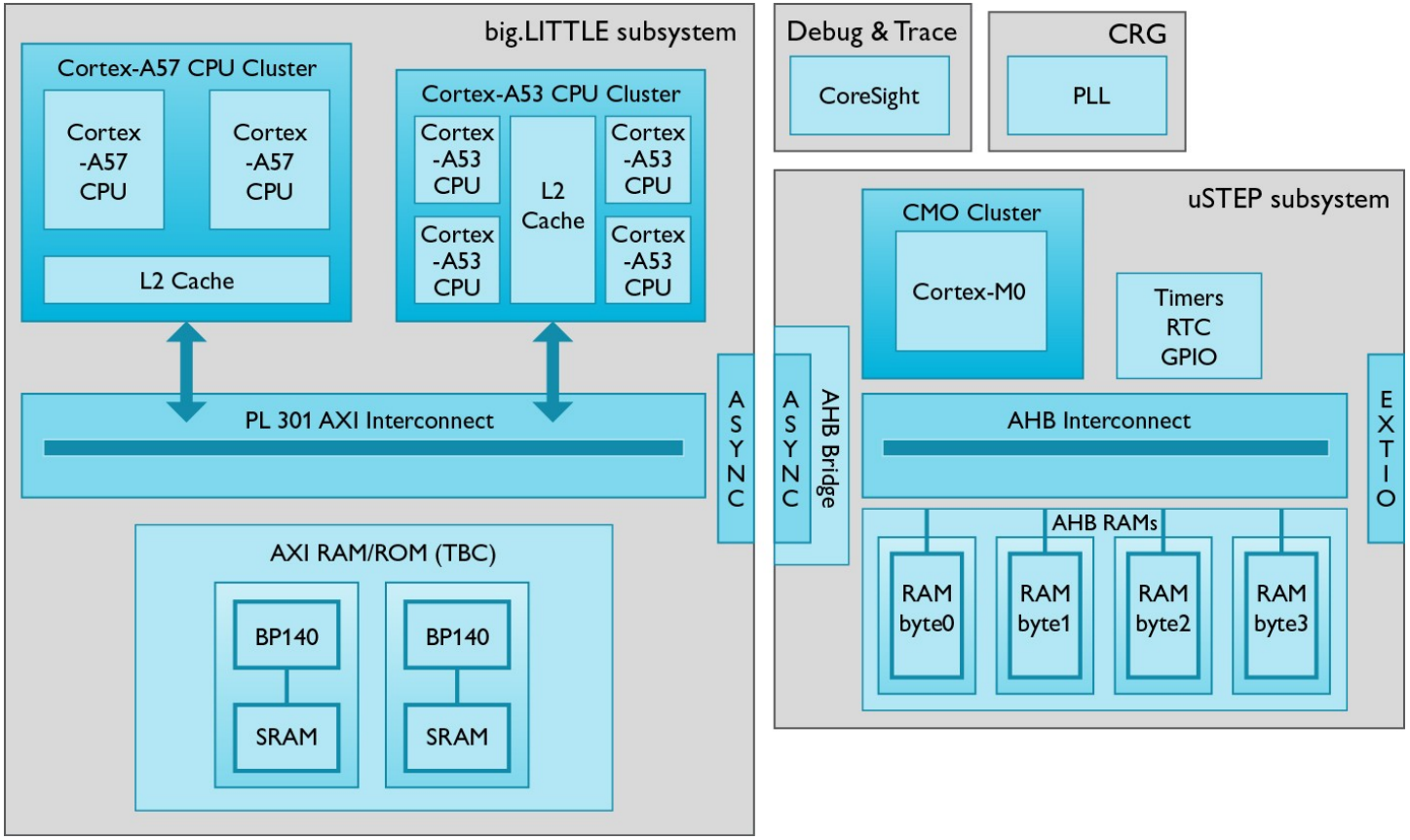
Procesadores Rizen
AMD

Procesadores I3 a I9
Intel



Microarquitectura multinúcleo

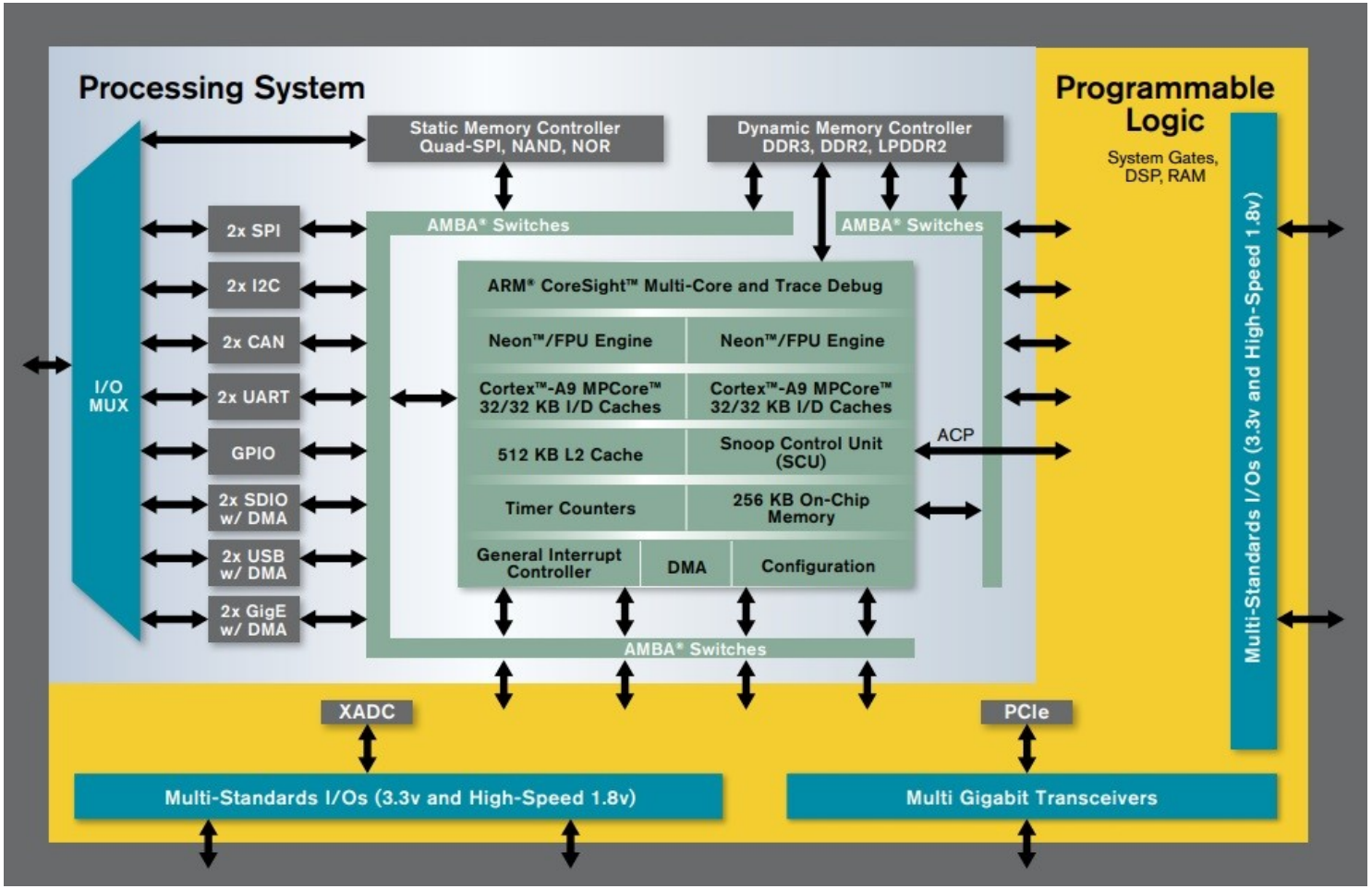
Ejemplos



Procesadores big.LITTLE
ARM

Microarquitectura multinúcleo

Ejemplos



Soc Zynq
Xilinx

Microarquitectura multi-núcleo

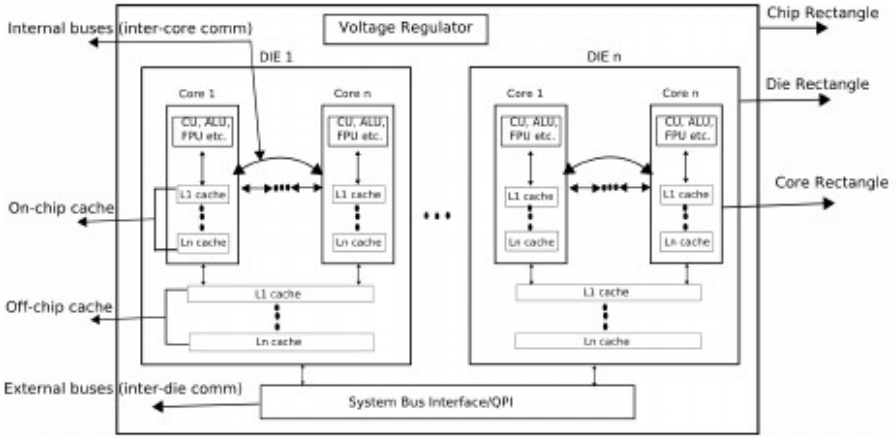
Ventajas

Rendimiento - Un procesador de varios núcleos puede realizar más trabajo que un procesador de un solo núcleo por dos motivos: el paralelismo natural (***MLP explicito***) y velocidades superiores. Como los núcleos están próximos, se logran velocidades de reloj más altas debido a que las señales viajar distancias cortas y son persistentes.

Fiabilidad y mutitarea - los programas se asignan en diferentes núcleos por lo que siempre que hay un defecto se limita a un solo núcleo y pueden tolerar, en mayor medida, fallas.

Interacciones de software - incluso si hay software ejecutado en diferentes núcleos, seguirá interactuando con uno y otro. Un procesador de múltiples núcleos se somete a un proceso conocido como aislamiento espacial y temporal. Estos procesos aseguran que los subprocesos centrales nunca se retrasen.

Consumo de energía - solo la parte del procesador que opera consume la energía. Además, los elementos comunes tienen un consumo independiente de los núcleos que están operando. Por ello, son más eficientes desde el punto de vista energético.



Microarquitectura multi-núcleo

Desventajas

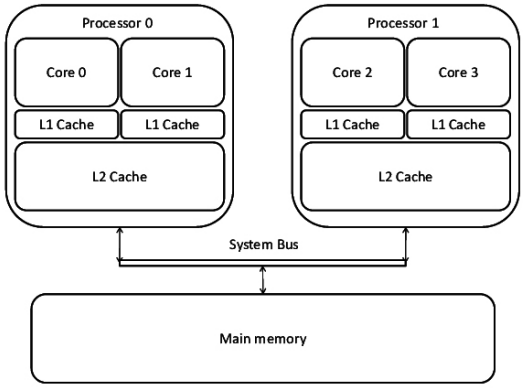
Velocidad de la aplicación - aunque está diseñado para realizar múltiples tareas, su velocidad no es lo suficientemente significativa. Cada vez que se ejecuta una aplicación, hay que conmutar de un núcleo a otro y es necesario actualizar la caché, lo que compensando su velocidad del núcleo.

Jitter - el aumento del número de núcleos produce interferencias en la ejecución que provoca una pérdida de sincronización, lo que lleva a una degradación del rendimiento de los programas debido a conflicto de recursos.

Análisis - realizar varias tareas requiere gestionar los accesos a la memoria, por lo que es difícil determinar y coordinar los tiempos de acceso y las interferencias que se producen.

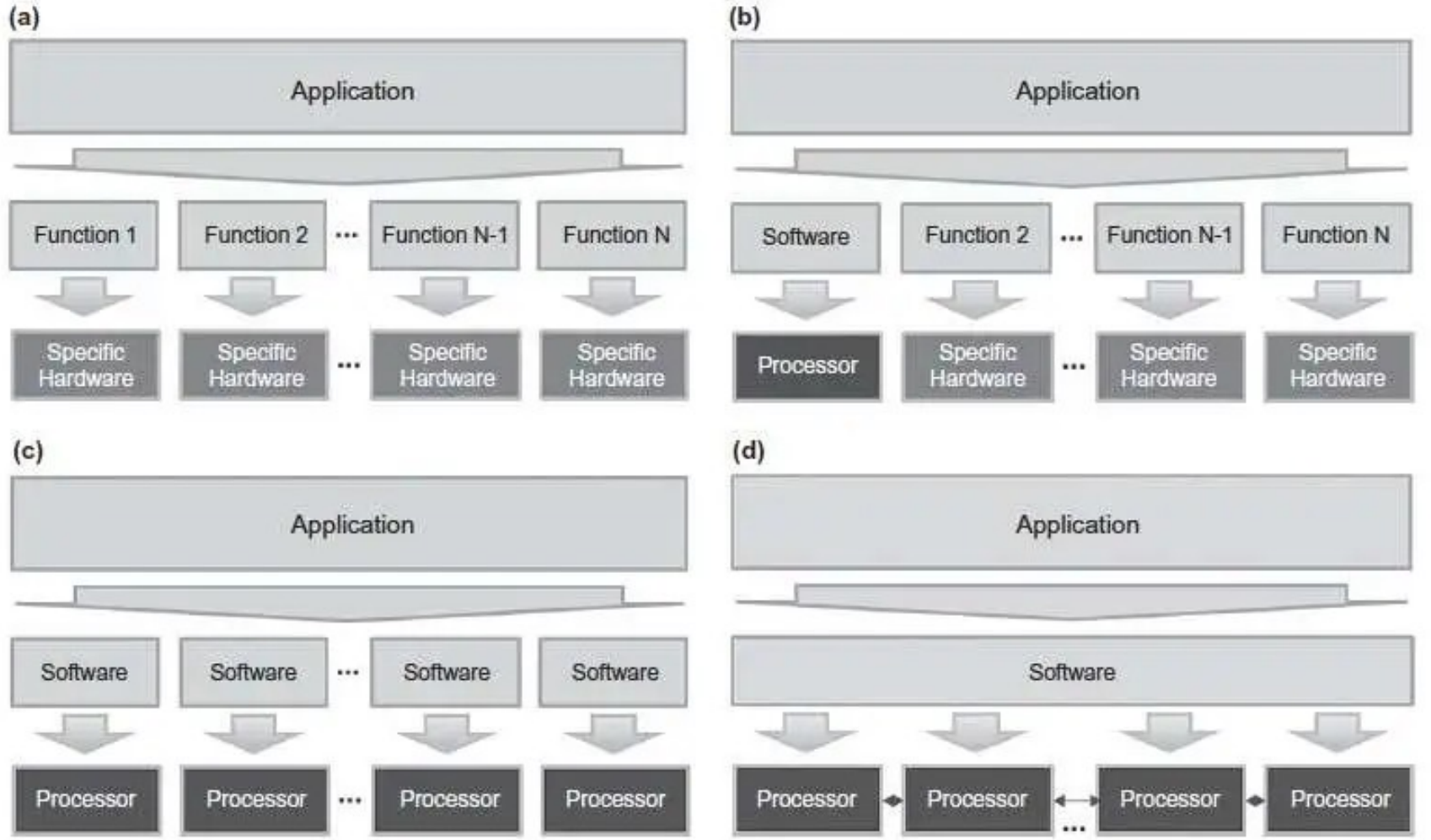
Recursos compartidos - los recursos (memoria principal, buses del sistema y periféricos) son compartidos por todos los núcleos por lo que cualquier aplicación tenderá a interferir la ejecución de los programas en los diferentes núcleos.

Interferencias - una interferencia es causada por el uso de recursos compartidos, planteando problemas de aislamiento espacial y temporal de cada programa. Esta posibilidad es mayor si hay varios núcleos operando simultáneamente, por lo que es imposible analizar todas y cada una de las posibles interferencias.



Microarquitectura multinúcleo

Paradigmas de programación



Evolución de los sistemas tradicionales SoC a los MPSoC - (*a*) SoC tradicional; (*b*) SoC de un solo procesador; (*c*) MPSoC usando “silos” y (*d*) MPSoC con software distribuido.

Conclusiones

Los procesadores multinúcleo representan una nueva tendencia en la arquitectura de computadoras que:

- Disminuye el consumo de energía y la generación de calor;
- Minimiza la longitudes de las conexiones y las latencias de interconexión;
- Permite un verdadero paralelismo a nivel de subprocesos con escalabilidad;

Para utilizar todo su potencial, las aplicaciones deberán pasar de un modelo único a uno de subprocesos múltiples, lo que implica que

- Las técnicas de programación paralela y concurrente ganarán importancia.
- Programar los sistemas multinúcleo de manera que permita que las aplicaciones se beneficien del continuo crecimiento en el rendimiento de los procesadores;

La industria del software necesita volver al estado en el que las aplicaciones existentes se ejecutan más rápido en hardware nuevo.