

Leccion7Tema3.pdf



beatr179



Arquitectura de Computadores



2º Grado en Ingeniería Informática



Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada



Estamos de
Aniversario

De la universidad al
mercado laboral:
especialízate con los posgrados
de EOI y marca la diferencia.



EOI Escuela de
organización
industrial



saber más

DIRÍAMOS QUE ESTE CAFÉ ES MÁS FUERTE QUE TU FUERZA DE VOLUNTAD,

PERO VIENDO QUE LO HAS DEJADO TODO PARA EL ÚLTIMO DÍA, TAMPOCO ERA DIFÍCIL.

Descuentazo para ti

Smilke®

CÓDIGO: WUOLAH10

ARQUITECTURAS CON PARALELISMO A NIVEL DE THREAD (TLP)

LECCIÓN 7: ARQUITECTURAS TLP

Objetivos:

- Distinguir entre cores multithread, multicores y multiprocesadores.
- Comparar entre cores multithread de grano fino, cores multithread de grano grueso y cores con multithread simultánea.

1. CLASIFICACIÓN DE ARQUITECTURAS CON TLP EXPLÍCITO Y UNA INSTANCIA DEL SO

Arq. con DLP (Data Level Parallelism)	Arq. con ILP (Instruction Level Parallelism)	Arq. con TLP (Thread Level Parallelism) explícito y una instancia de SO		Arq. con TLP explícito y múltiples instancias SO
Ejecutan las operaciones de una instrucción concurr. o en paralelo	Ejecutan múltiples instrucciones concurr. o en paralelo	Ejecutan múltiples flujos de instrucciones concurr. o en paralelo		Ejec. múltiples flujos de instr. en paralelo
Unidades funcionales vectoriales o SIMD (90)	Cores escalares (60) segmentados (80), superescalares (90) o VLIW/EPIC (90)	Cores que modifican la archit. ILP para ejecutar threads concurr. o en paralelo (se extendieron en 2000)	Multi-procesadores (60): ejecutan threads en paralelo en un computador con múltiples cores (incluye multicores (2000))	Multi-computadores (85): ejecutan threads en paralelo en un sistema con múltiples computadores

a. Arquitecturas con DLP (Data Level Parallelism)

- **¿Qué hacen?:** ejecutan varias operaciones de una misma instrucción al mismo tiempo.
- **Ejemplo de hardware:** unidades funcionales vectoriales o SIMD (Single Instruction, Multiple Data).
- **Ejemplo típico:** procesadores gráficos (GPUs), que aplican una instrucción sobre muchos datos (como aplicar un filtro a una imagen).

b. Arquitecturas con ILP (Instruction Level Parallelism)

- **¿Qué hacen?:** ejecutan múltiples instrucciones al mismo tiempo, aunque esas instrucciones pertenezcan al mismo flujo de control o thread.



COMPRA AQUÍ

WUOLAH

- **Ejemplo de hardware:** cores escalares(un ciclo, una instrucción), segmentados(pipeline), superescalares(ejecutan varias instrucciones en paralelo), VLIW/EPIC(ejecución de varias instrucciones empaquetadas).
- **Ejemplo típico:** CPUs modernas que pueden ejecutar varias instrucciones al mismo tiempo si son independientes.

c. Arquitecturas con TLP (Thread Level Parallelism)

TLP explícito y una instancia de SO

- **¿Qué hacen?:** ejecutan múltiples flujos de instrucciones (threads) de manera concurrente o en paralelo, pero dentro de un único sistema operativo.
- **Ejemplo de hardware:** cores que modifican la arquitectura ILP(para manejar varios threads), multiprocesadores(varios cores en un mismo chip).

TLP explícito y múltiples instancias de SO

- **¿Qué hacen?:** ejecutan múltiples flujos de instrucciones en paralelo, pero en un sistema distribuido, donde cada nodo puede tener su propio sistema operativo.
- **Ejemplo de hardware:** multi-computadores, como clústeres, donde distintos ordenadores trabajan en paralelo en una red.

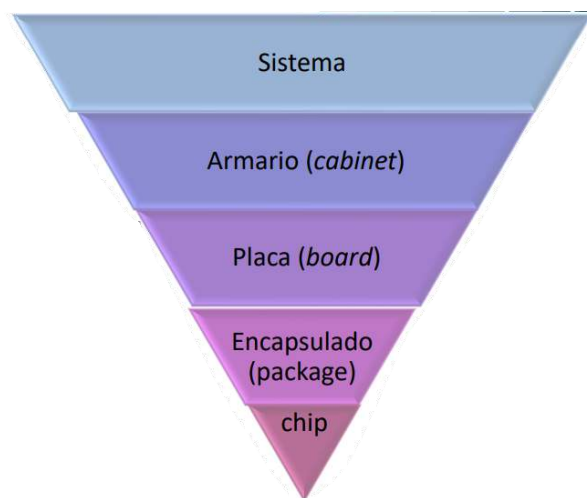
2. MULTIPROCESADORES Y MULTICORES

Ejecutan varios flujos de instrucciones (threads) en paralelo en un computador con varios cores/procesadores (cada thread en un core/procesador distinto).

2.1. Multiprocesadores. Criterio de clasificación: nivel de empaquetado/conexión

Un **multiprocesador** puede estar implementado a distintos niveles:

- Puede ser un **multicore** si los cores están en el mismo chip.
- Puede ser un **MCM** si varios chips están dentro del mismo encapsulado.
- Puede escalarse a **placas, armarios** o incluso **sistemas completos** si interconectamos múltiples encapsulados.



Pirámide de niveles de empaquetado/conexión:

1. Sistema completo:

- Todo el conjunto de hardware y software que trabaja en conjunto.
- Ejemplo: un clúster de servidores completo.

2. Armario (cabinet):

- Un rack que puede contener varios servidores.
- Ejemplo: HPE Superdome Flex, que alberga 8 servidores, cada uno con 4 sockets (lugares donde se conectan procesadores).

3. Placa (board):

- Una placa base donde se montan procesadores, memoria y otros componentes.
- Ejemplo: Placa de servidor con 4 sockets.

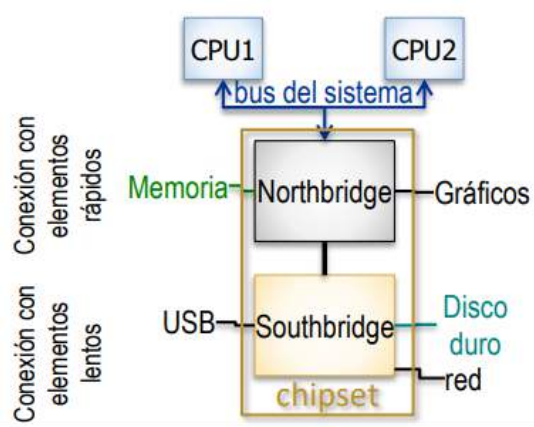
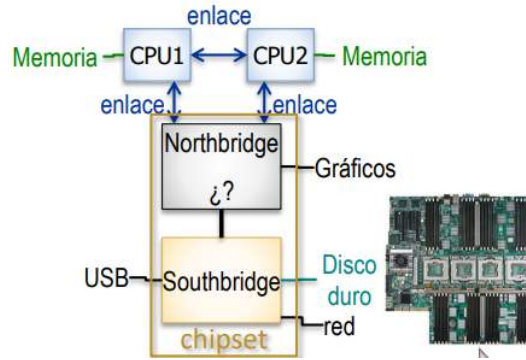
4. Encapsulado (package):

- Es el módulo físico donde se ensamblan los procesadores o núcleos (cores). Puede contener más de un chip (módulo multichip o MCM).
- Ejemplo: AMD EPYC, que integra varios dies (chips) en un mismo encapsulado junto con la red de interconexión.

5. Chip:

- El propio silicio donde están los cores, memorias caché, controladores...
- Ejemplo: Imagen de abajo a la derecha muestra un chip con varios cores y cachés L3 compartidas.

2.2. Multiprocesador en una placa: evolución de UNA a NUMA

UMA	NUMA
 <p>Conexión con elementos rápidos</p> <p>Conexión con elementos lentos</p> <p>¿Qué es?</p> <p>Todos los procesadores (CPUs) acceden a la misma memoria con tiempo uniforme (igual latencia).</p>	 <p>enlace</p> <p>Memoria</p> <p>enlace</p> <p>enlace</p> <p>Northbridge</p> <p>¿?</p> <p>Gráficos</p> <p>USB</p> <p>Southbridge</p> <p>chipset</p> <p>Disco duro</p> <p>red</p> <p>¿Qué es?</p> <p>Cada procesador tiene su propio controlador de memoria y memoria local, pero pueden acceder a la memoria de otros procesadores (aunque con mayor latencia).</p>

DIRÍAMOS QUE ESTE CAFÉ ES MÁS FUERTE QUE TU FUERZA DE VOLUNTAD,

PERO VIENDO QUE LO HAS DEJADO TODO PARA EL ÚLTIMO DÍA, TAMPOCO ERA DIFÍCIL.

Smilke®

Descuentazo para ti

CÓDIGO: WUOLAH10

¿Cómo funciona?

- El **controlador de memoria** está en el **chipset** (Northbridge), fuera del procesador.
- Los procesadores se comunican con la memoria a través de un **bus compartido** (todos los CPUs usan el mismo canal para acceder a la memoria).

Problema

A medida que aumentan los procesadores, el bus se satura (cuello de botella), limitando la escalabilidad.

¿Cómo funciona?

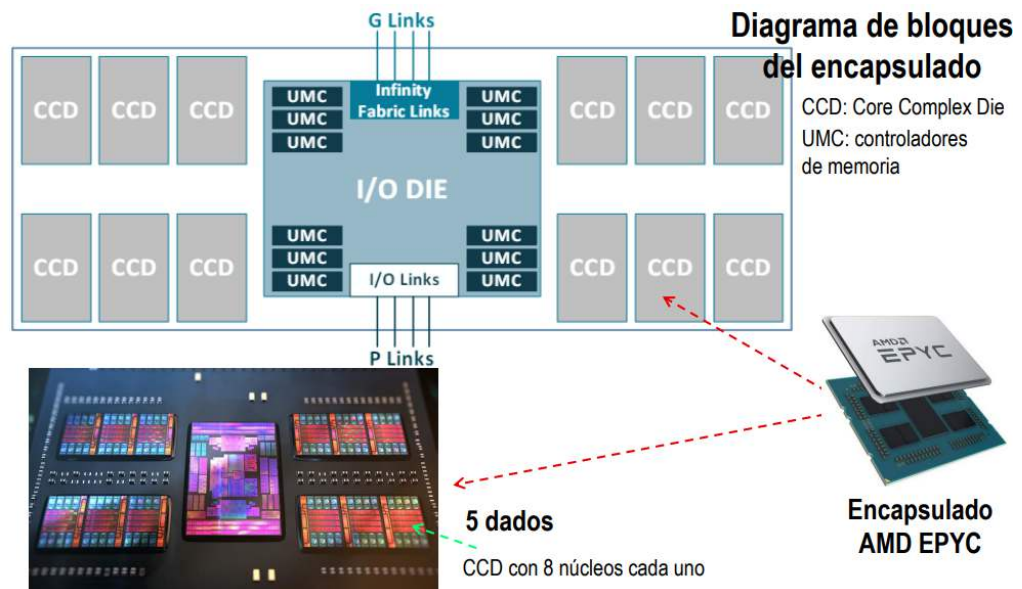
- El controlador de memoria está **dentro del propio procesador**.
- La interconexión se realiza mediante **enlaces punto a punto** (no un bus compartido).
- Los procesadores pueden comunicarse directamente entre sí mediante estos enlaces.

Ventaja

Mejora la escalabilidad y reduce el cuello de botella, ya que los accesos a memoria local son más rápidos y los accesos remotos son posibles, aunque más lentos.

2.3. Multiprocesador en encapsulado

Este tipo de arquitectura **MCM** (Módulo multichip) es un paso más allá del **multicore tradicional** porque en lugar de tener todos los núcleos en un único chip, se distribuyen en varios chips interconectados, pero todo dentro del mismo encapsulado.



COMPRA AQUÍ

WUOLAH

Componentes clave:

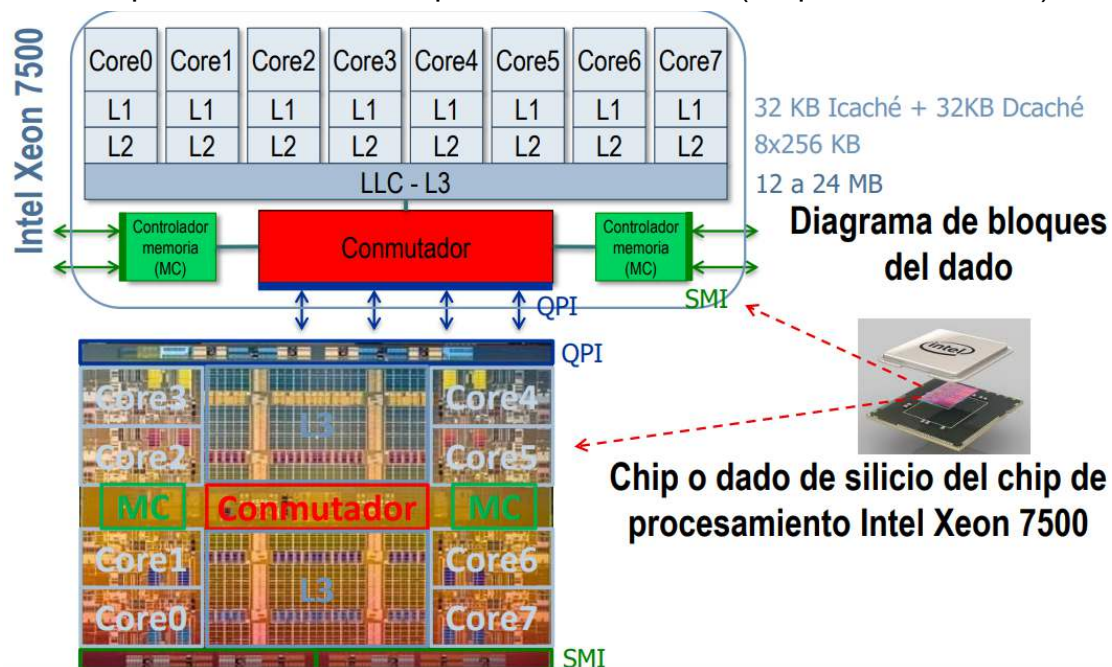
1. **Encapsulado AMD EPYC:** a nivel externo es un solo procesador (un chip grande), pero internamente está formado por varios **dies** (chips más pequeños de silicio).
2. **5 dados** (dies) en total:
 - 4 **CCD** (Core Complex Die): cada uno tiene **8 núcleos (cores)**, por lo que en total puedes tener **32 núcleos** en este encapsulado. Los **CCD** son los bloques donde realmente se ejecutan las instrucciones.
 - 1 **I/O Die** (IOD): no tiene núcleos de procesamiento, pero sí **UMC** (Unified Memory Controllers / Controladores de memoria), **I/O Links** (Interfaces para conectar con dispositivos externos) y **Infinity Fabric Links** (Conexiones internas que permiten la comunicación entre los CCD y el IOD).

Interconexión:

Los **CCD** no se comunican directamente entre sí. Todo pasa a través del **I/O Die**:

- **Infinity Fabric Links** conectan los **CCD** con el **I/O Die**.
- Los **UMC** en el I/O Die se encargan de gestionar el acceso a la memoria.
- **G Links** y **P Links** son conexiones externas (probablemente hacia otros chips o el sistema).

2.4. Multiprocesador en un chip o Multicore o CMP (Chip MultiProcessor)



Smilke®

LLEVARTE 4 HUEVOS A LA
BIBLIOTECA QUEDA RARO.
LLEVARTE UN SMILKE, NO.
22 GR DE PROTEÍNA. CASI NADA



Descuentazo
para ti →

CÓDIGO: WUOLAH10

COMPRA AQUÍ



Un **Chip MultiProcessor (CMP)** es un único chip de silicio que integra **múltiples núcleos de procesamiento (cores)** dentro del mismo dado.

Estructura del chip

8 núcleos (Core0 a Core7):

- **L1 caché:** 32 KB para instrucciones + 32 KB para datos.
- **L2 caché:** 256 KB por núcleo.

L3 caché (LLC - Last Level Cache):

- Compartida por todos los núcleos.
- Tamaño: entre **12 MB y 24 MB**.

Controladores de Memoria (MC):

- Integrados en el propio chip, uno a cada lado.
- Permiten el acceso directo a la memoria (más eficiente que en UMA clásica).

Conmutador (Switch interno):

- Se encarga de **rotar el tráfico** entre núcleos, caché, controladores de memoria y enlaces externos.

Interfaces externas:

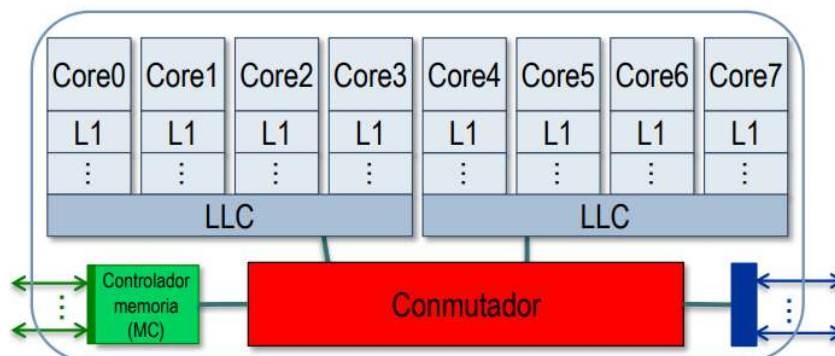
- **QPI** (Quick Path Interconnect): Conecta este chip con otros procesadores.
- **SMI** (Scalable Memory Interconnect): Conecta con los módulos de memoria externos.

Ventajas del CMP

1. Menor latencia.
2. Mejor rendimiento y escalabilidad.
3. Más eficiente.

2.5. Multicore: otras posibles estructuras

1. **Cores con LLC compartida por grupo**



DIRÍAMOS QUE ESTE CAFÉ ES MÁS FUERTE QUE TU FUERZA DE VOLUNTAD,

PERO VIENDO QUE LO HAS DEJADO TODO PARA EL ÚLTIMO DÍA, TAMPOCO ERA DIFÍCIL.

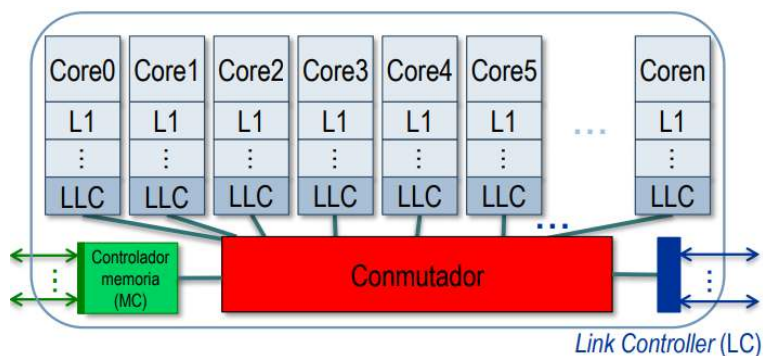
Smilke®

Descuentazo para ti

CÓDIGO: WUOLAH10

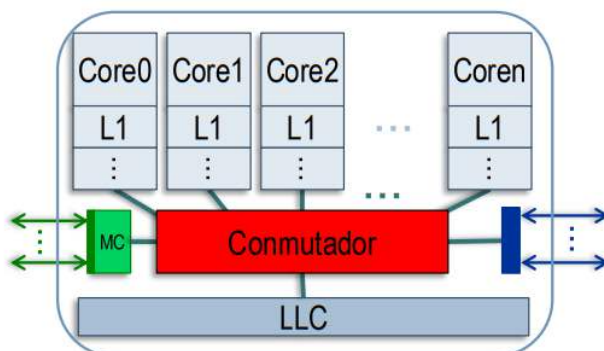
- Núcleos agrupados (Core0–3 y Core4–7).
- Cada grupo **comparte una caché LLC**.
- Todos se conectan a un **conmutador central**, que a su vez se comunica con el **controlador de memoria (MC)**.

2. LLC privada por núcleo



- Cada núcleo tiene su **propia LLC** (además de L1).
- El **conmutador** conecta todos los núcleos y el MC.

3. Topología vertical con LLC común



- Núcleos organizados verticalmente.
- **Una única LLC compartida** en un lateral del chip.
- Conexión a través de un **conmutador central** y un **único MC**.

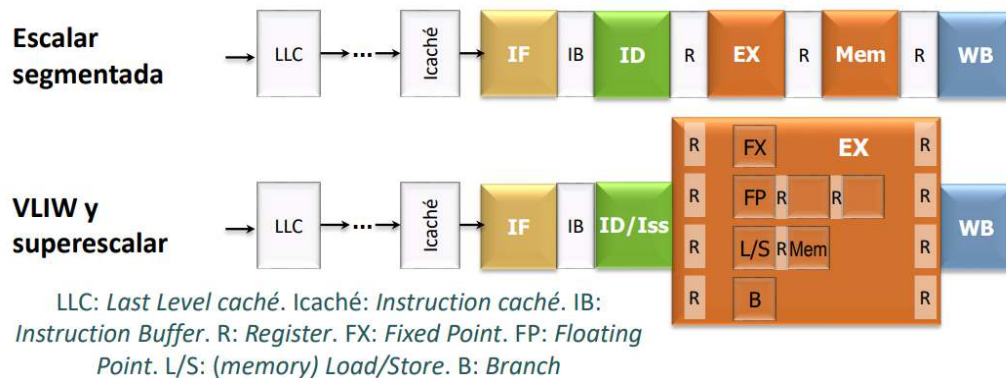


COMPRA AQUÍ

WUOLAH

3. CORES MULTITHREAD

3.1. Arquitecturas ILP



Tipos de arquitectura ILP

Escalada segmentada:

- Cada instrucción pasa por etapas **una por ciclo**:
IF → ID → EX → Mem → WB
- Solo **una instrucción activa por etapa**.
- Ejecución **concurrente** pero no paralela.

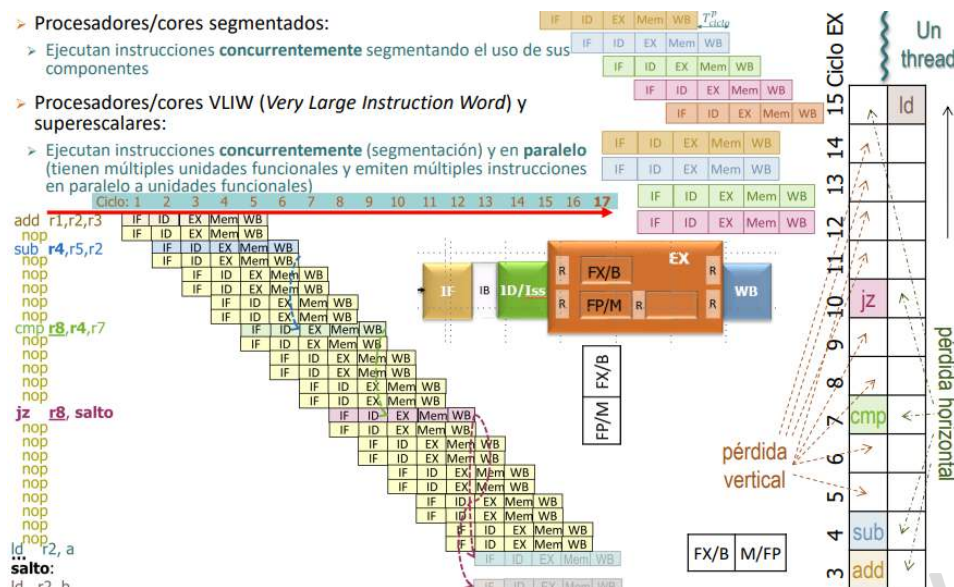
VLIW y superescalar:

- Puede ejecutar **varias instrucciones por ciclo** (paralelismo real).
- Varias **unidades funcionales**: FX (entero), FP (coma flotante), L/S (memoria), B (saltos).

Etapas del cauce (pipeline):

1. **IF (Instruction Fetch)**: Captura de la instrucción.
2. **ID/Iss (Decode / Issue)**: Decodifica y lanza a unidad funcional.
3. **EX (Execute)**: Ejecuta operación.
4. **Mem**: Acceso a memoria (si aplica).
5. **WB (Write-Back)**: Guarda resultado en registros.

3.2. Cores ILP y motivación de TLP



En la parte inferior se muestra un **pipeline segmentado**, donde las instrucciones se ejecutan una a una.

Las instrucciones están separadas por huecos (**nop**) para evitar conflictos o por dependencias.

Se visualiza cómo las instrucciones se "escalonan" en el tiempo.

Problemas del ILP puro

- **Pérdida vertical:**
Ocurre cuando no hay suficientes **instrucciones listas** para emitir en ciclos consecutivos.
- **Pérdida horizontal:**
Ocurre cuando no hay suficientes **instrucciones diferentes** (enteros, flotantes, saltos, etc.) para usar **todas las unidades funcionales** al mismo tiempo.

Motivación para usar TLP (Thread Level Parallelism)

- Cuando **el ILP no basta**, se introducen múltiples threads (TLP) para llenar los huecos del pipeline.
- Así se **aprovechan mejor** los recursos del procesador y se **reduce la ineficiencia** provocada por las pérdidas mencionadas.

Concepto	Explicación breve
ILP	Ejecutar instrucciones independientes en paralelo.
Pérdida vertical	No hay instrucciones listas → pipeline desocupado.
Pérdida horizontal	Unidades funcionales sin uso → instrucciones poco variadas.
TLP	Varios hilos compensan limitaciones del ILP.

3.3. Modificación de la arquitectura ILP en Core Multithread (ej. SMT)

¿Qué es SMT (Simultaneous Multithreading)?

Es una técnica que permite ejecutar instrucciones de múltiples hilos (threads) en el mismo ciclo de reloj dentro de un solo core.

Cambios en la arquitectura ILP para soportar SMT

- Almacenamiento (memoria/cache/registros): se multiplexa, comparte, reparte o replica entre hilos.
- Hardware dentro de las etapas: se multiplexa, comparte o reparte entre hilos.

DIRÍAMOS QUE ESTE CAFÉ ES MÁS FUERTE QUE TU FUERZA DE VOLUNTAD,

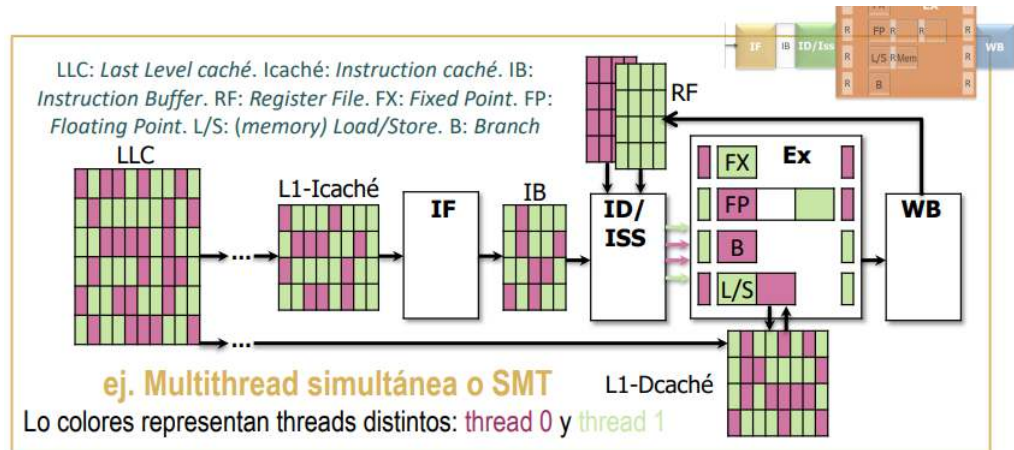
PERO VIENDO QUE LO HAS DEJADO TODO PARA EL ÚLTIMO DÍA, TAMPOCO ERA DIFÍCIL.

Descuentazo para ti

CÓDIGO: WUOLAH10

Smilke®

Ejemplo:

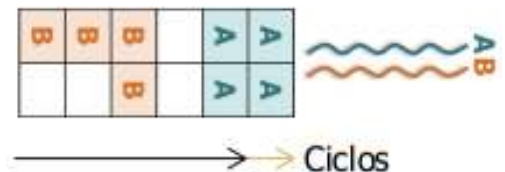


- Instrucciones de thread 0 (■) y thread 1 (■) se ejecutan en paralelo.
- Las **unidades FX, FP, B, L/S** están compartidas entre los dos hilos.
- Se aprovecha mejor el hardware y se reducen huecos del pipeline.

3.4. Clasificación de cores multithread

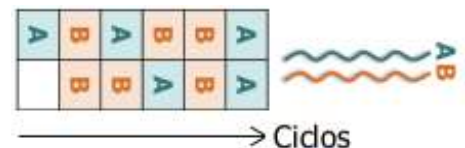
TMT (Temporal Multithreading)

- Ejecución **concurrente** de varios hilos, **uno por ciclo**.
- Conmutación entre hilos: la controla el hardware.
- En **cada ciclo** solo se emite **una instrucción** de un único thread.



SMT (Simultaneous Multithreading)

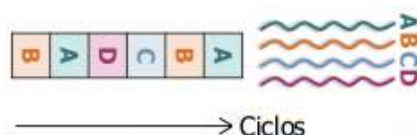
- Ejecución **en paralelo** de varios hilos **en el mismo ciclo** (superescalar).
- Se pueden emitir **instrucciones de distintos hilos a la vez**.
- No hay conmutación entre threads.



3.5. Clasificación de cores con TMT

FGMT (Fine-grain Multithreading)

- El hardware **cambia de thread en cada ciclo** (coste 0).
- Conmutación: **round-robin** (turno rotatorio) / **por eventos de latencia**, con planificación (ej. thread menos recientemente ejecutado).
- Ideal cuando hay muchos hilos y latencias pequeñas.



WUOLAH

CGMT (Coarse-grain Multithreading)

- La conmutación ocurre **cada varios ciclos** o por eventos.
- Modalidades: **timeslice** (conmutación tras tiempo fijo), **switch-on-event** (conmutación al detectar latencia).

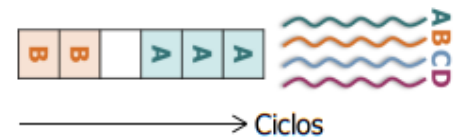


Tabla comparaciones

Tipo	Conmutación	Ciclos	Paralelismo	Ventaja principal
TMT	Hardware decide	1+	✗	Simple, buena para ocultar latencia
FGMT	Cada ciclo	1	✗	Coste 0, útil con muchos hilos
CGMT	Cada N ciclos/evento	0–varios	✗	Bajo coste o más controlado
SMT	Sin conmutación	0	✓	Uso completo de recursos

3.6. Clasificación de cores con CGMT con conmutación por eventos

Estática

- **Explícita:** instrucciones especiales de cambio de contexto.
- **Implícita:** cambio tras instrucciones como carga/salto.
- **Ventaja:** bajo coste (0 o 1 ciclo).
- **Inconveniente:** conmutaciones innecesarias.

Dinámica

- **Eventos como:** fallo de caché, interrupción.
- **Ventaja:** menos conmutaciones innecesarias.
- **Inconveniente:** mayor sobrecarga al cambiar de contexto.

3.7. Alternativas

3.7.1. Alternativas en un core escalar segmentado

Un thread (sin TLP)

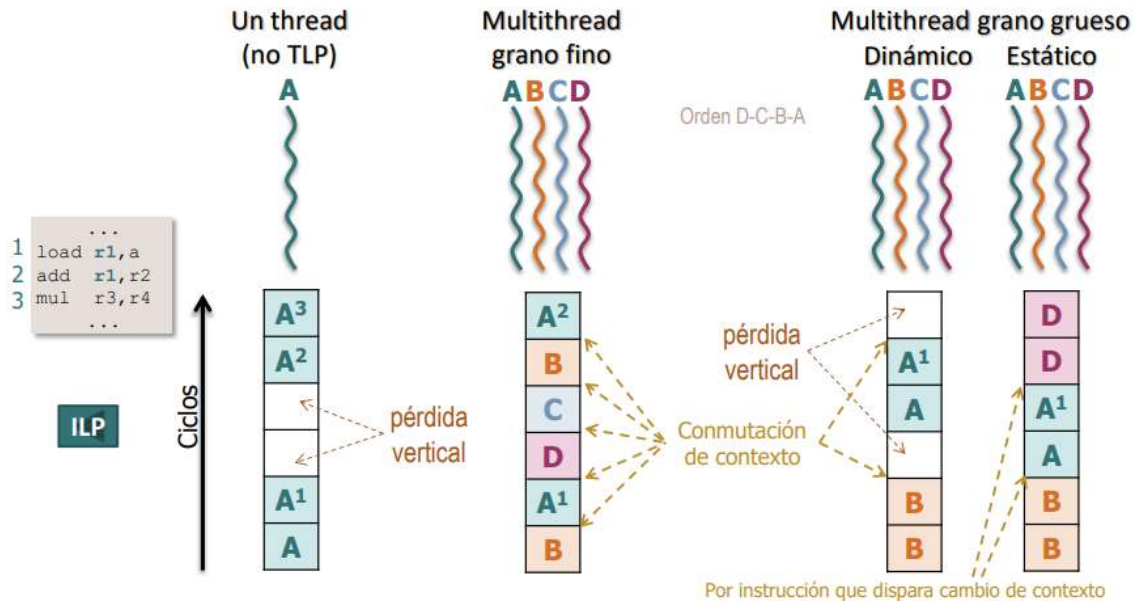
- Se ejecuta una instrucción por ciclo.
- Problema: pérdida vertical (ciclos vacíos por dependencias o latencia).

Multithread grano fino (FGMT)

- Se alternan threads cada ciclo ($A \rightarrow B \rightarrow C \rightarrow D \dots$).
- **Ventaja:** oculta latencias.
- **Pérdida vertical** reducida.

Multithread grano grueso (CGMT)

- Se mantiene el mismo thread **durante varios ciclos**.
- Conmutación ocurre por eventos (dinámico) o tras intervalos (estático).
- **Ventaja:** reduce coste de cambio de contexto.
- **Problema:** si un thread bloquea (cache miss), puede quedar el core sin uso.



3.7.2. Alternativas en un core con emisión múltiple de instrucciones de un thread

Un thread (sin TLP)

- Se emiten **varias instrucciones por ciclo**.
- **Problemas:**
 - **Pérdida horizontal:** no hay suficientes instrucciones de tipos diferentes.
 - **Pérdida vertical:** por dependencias.

Multithread grano fino

- Mezcla de instrucciones de **diferentes threads** cada ciclo.
- Mejora el uso de unidades funcionales.
- **Menos huecos** (mejor aprovechamiento del core).

Multithread grano grueso

- Cambios de thread menos frecuentes.
- Mejora algunas pérdidas, pero **no tan eficiente** como FGMT para ocultar latencias cortas.

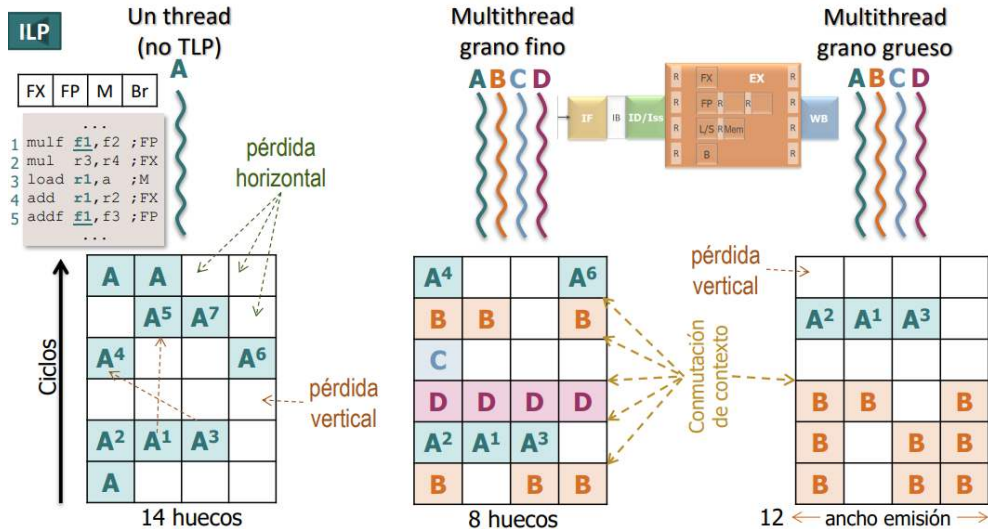
DIRÍAMOS QUE ESTE CAFÉ ES MÁS FUERTE QUE TU FUERZA DE VOLUNTAD,

PERO VIENDO QUE LO HAS DEJADO TODO PARA EL ÚLTIMO DÍA, TAMPOCO ERA DIFÍCIL.

Descuentazo para ti

CÓDIGO: WUOLAH10

Smilke®



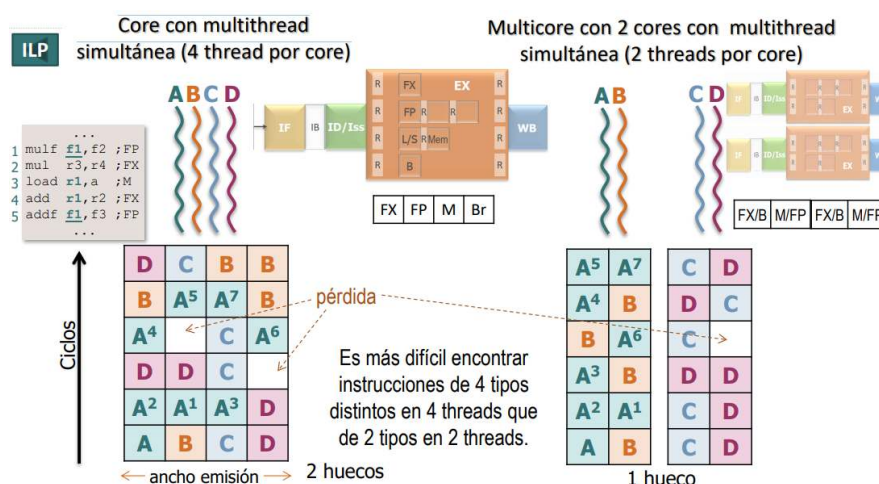
3.7.2. Alternativas en un core multithread simultánea y multicore

Core con multithread simultánea (4 thread por core)

- Puede emitir instrucciones **de hilos distintos** en el mismo ciclo.
- **Mayor aprovechamiento** si las instrucciones son variadas (FX, FP, M, Br).

Multicore con 2 cores con multithread simultánea (2 threads por core)

- Cada core es SMT (emite instrucciones de 2 threads).
- Es **más fácil** encontrar instrucciones complementarias en 2 hilos.




- **Más threads ≠ siempre mejor**: hay más dificultad para **llenar todas las unidades** por falta de variedad.



COMPRA AQUÍ

WUOLAH

4. HARDWARE Y ARQUITECTURAS TLP EN UN CHIP



Hardware	CGMT	FGMT	SMT	CMP
Registros de la arquitectura	replicado (al menos PC)	replicado	replicado	replicado
Almacenamiento	multiplexado	multiplexado, repartido, compartido o replicado	repartido, compartido o replicado	replicado
Otro hardware de las etapas del cauce	multiplexado	Captación: repartida o compartida; Resto: multiplexadas	UF: compartidas; Resto: repartidas o compartidas	replicado
Etiquetas para distinguir el thread de una instr.	Sí	Sí	Sí	No
Hardware para conmutar entre threads	Sí	Sí	No	No

Multiplexar: usar el mismo recurso de forma secuencial entre threads.

Repartir: asignar una parte del recurso a cada thread.

Compartir: todos los threads acceden al mismo recurso.

Replicar: cada thread tiene su copia del recurso.

- **CGMT(Coarse-Grain Multithreading)** es multithreading de grano grueso. Ejecuta un solo hilo por vez y cambia a otro solo tras varios ciclos o eventos como un fallo de caché.

- **FGMT(Fine-Grain Multithreading)** es de grano fino. Cambia de hilo en cada ciclo de reloj, lo que permite ocultar latencias, aunque solo se ejecuta una instrucción por ciclo.

- **SMT(Simultaneous Multithreading)** permite ejecutar instrucciones de varios hilos al mismo tiempo dentro del mismo core, aprovechando mejor los recursos del procesador.

- **CMP(Chip MultiProcessor)** es una arquitectura con varios cores físicos en un chip. Cada core ejecuta su propio hilo en paralelo, como en un procesador multicore típico.