**Tema 10: Segmentación y Paralelismo en el diseño de Computadores**

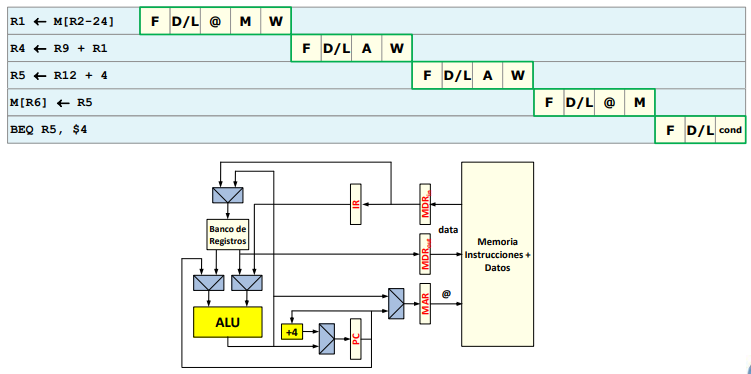
**Introducción**

Técnicas Básicas en el Diseño de Procesadores (¡ya vistas!):

* Memorización:
  + Memoria Cache.
  + TLB.
  + Predicción vía.
* Concurrencia:
  + *Segmentación*:
    - Cache segmentada.
    - Escrituras segmentadas.
    - SDRAM.
  + *Paralelismo*:
    - Bancos DDR.
    - Cache multibanco.
    - RAIDs.

Procesador Secuencial:

Las instrucciones se ejecutan de forma secuencial. Las instrucciones pueden tener tiempos de ejecución diferentes, dependiendo de su complejidad:



**Paralelismo a Nivel de Instrucciones (ILP)**

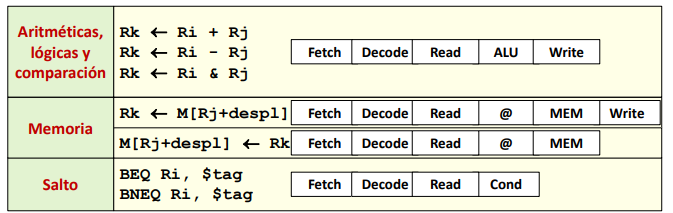
Procesadores segmentados:

Objetivo: ejecutar 1 instrucción por ciclo, CPI = 1.

Dificultades para alcanzar este objetivo:

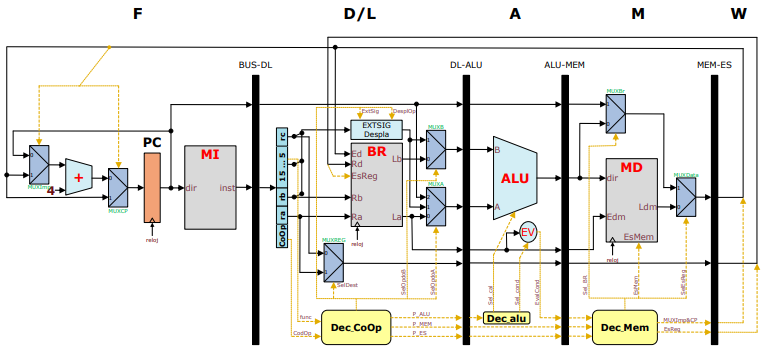
* Los recursos hardware disponibles.
* Respetar la semántica del Lenguaje Máquina.

Para simplificar la segmentación se utiliza un LM tipo RISC:



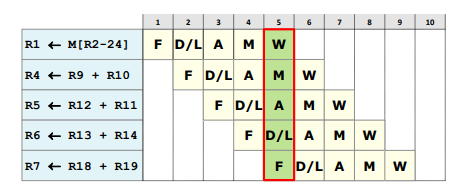
Para simplificar el hardware, se busca que todas las instrucciones usen la misma segmentación (segmentado en 5 etapas):





Todos los registros y elementos de almacenamiento se actualizan en el flanco ascendente de reloj. Para ver cuando se usa un elemento hay que observar sus conexiones, no su ubicación.

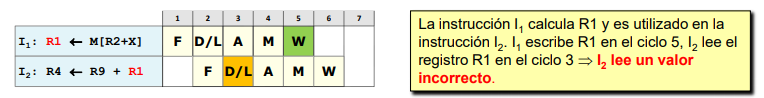
Ejecución Segmentada:



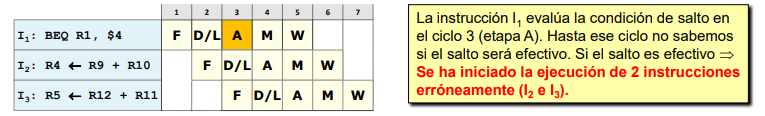
* Las instrucciones (datos + control) se mueven por el pipeline.
* Al inicio de ciclo, todas las etapas empiezan a funcionar con los datos que hay en los registros de desacoplo en la entrada de la etapa.
* Al final de ciclo, las señales de salida de cada etapa se almacenan en los registros de desacoplo.
* En un ciclo determinado se está ejecutando una instrucción diferente en cada una de las etapas.
* Prácticamente, con el hardware necesario para ejecutar 1 instrucción, estamos ejecutando concurrentemente 5 instrucciones independientes.

Límites a la Segmentación:

* *Riesgos de Datos*:



* *Riesgos de Control*:



* *Riesgos Estructurales*:

Se producen cuando un único recurso se intenta utilizar por 2 instrucciones diferentes. El procesador segmentado que hemos presentado está libre de riesgos estructurales:

* + El Banco de Registros permite 2 lecturas y 1 escritura en el mismo ciclo.
  + Dispone de Memorias independientes para instrucciones (MI) y datos (MD).

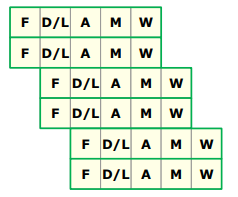
Los Procesadores Segmentados son complicados:

* No todas las instrucciones han de tener las mismas etapas:
  + Operaciones complejas: multiplicación.
  + No siempre es fácil segmentar una operación: división.
  + Aritmética en Coma Flotante.
* Instrucciones con tiempo de ejecución variable:
  + Accesos a Memoria: con fallo o acierto en cache (diferentes niveles).
* Gestión de Interrupciones y Excepciones.

Procesadores Superescalares:

*Objetivo CPI < 1*:

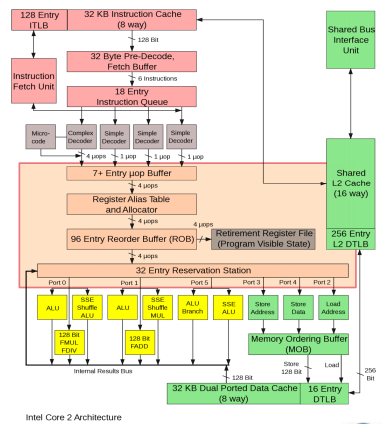
* Dispone de múltiples Unidades Funcionales.
* Permite iniciar más de una instrucción (u operación) por ciclo.
* Sigue siendo un procesador segmentado.
* Las instrucciones pueden tener tiempos de ejecución diferentes.
* La mayoría de los procesadores de propósito general actuales son superescalares.



Procesadores Superescalares OoO:

*Objetivo CPI < 1*:

* Muchos procesadores superescalares permiten la ejecución fuera de orden (OoO Processors, Out of Order Processors).
* Las instrucciones se leen en orden, pero pueden ejecutarse en desorden.
* Las instrucciones se bloquean si sus operandos no están disponibles.
* Las instrucciones inician su ejecución cuando tiene sus operandos disponibles y la correspondiente U.F. está libre.
* La mayoría de los procesadores de propósito general actuales son superescalares con ejecución fuera de orden.

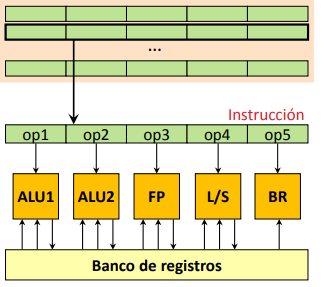


VLIW:

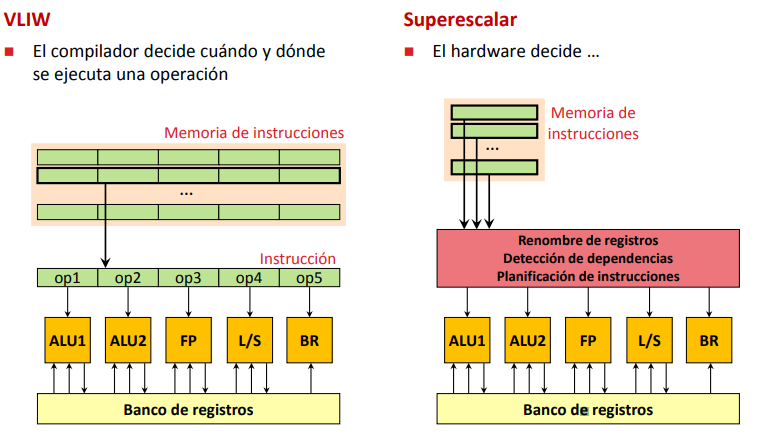
VLIW: Very Long Instruction Word (Arquitecturas con tamaño de instrucción muy grande):

* Objetivo: explotar ILP (Instruction Level Parallelism), CPI < 1.
* Una instrucción especifica múltiples operaciones independientes.
* Cada operación se ejecuta en una unidad funcional predeterminada.
* La planificación de las instrucciones es estática: realizada por el compilador.
* La planificación estática permite usar menos hardware de control y una arquitectura más simple:
  + Mayor frecuencia.
  + Menor consumo.
  + Menor coste.
  + Más espacio para recursos (registros, UFs, caches, …).
  + Mayor facilidad de verificación.
* El procesador sigue estando segmentado.
* Un porcentaje muy grande de los procesadores que se fabrican son embedded y de éstos muchos son VLIW.

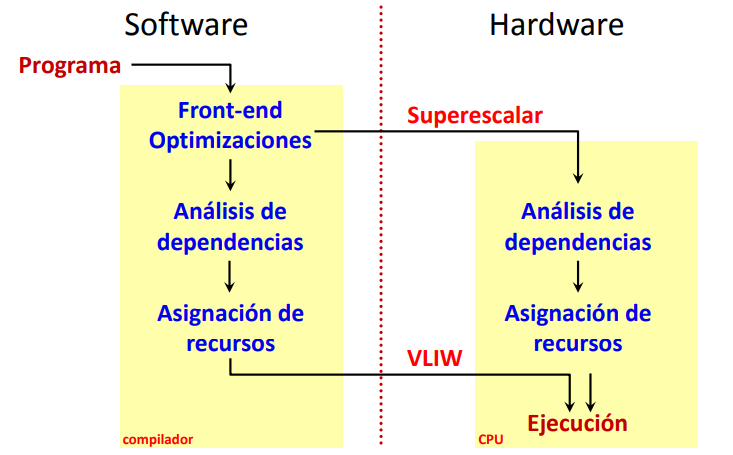




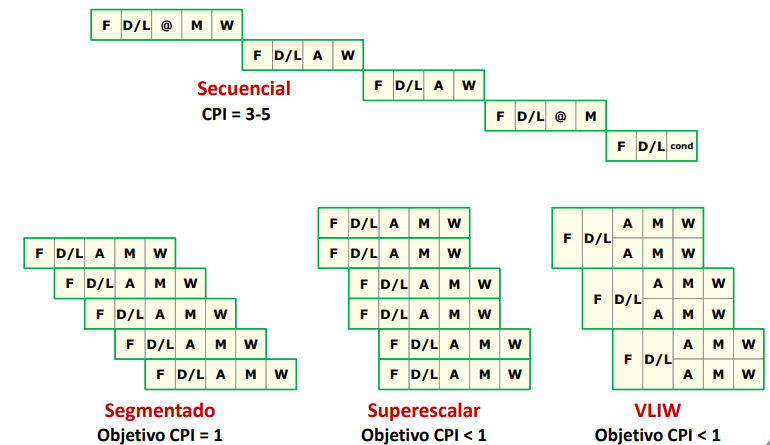
VLIW vs Superescalar:



Compilación para VLIW vs para Superescalar:



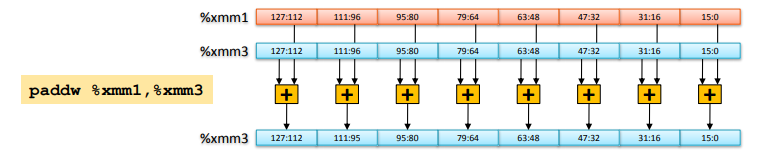
Resumiendo ILP:



**Paralelismo a Nivel de Datos (DLP)**

SIMD (Single Instruction Multiple Data):

1 única instrucción que permite operar con múltiples datos del mismo tipo.



Especialmente pensadas para aplicaciones multimedia: procesado de imagen, sonido, …

La mayoría de los procesadores actuales tienen extensiones SIMD:

* Procesadores x86: **MMX**; MultiMedia eXtension, Multiple Math eXtension, or Matrix Math eXtension (64 bits), **SSE**; Streaming SIMD Extensions, (128 bits), **AVX**; Advanced Vector eXtensions, (256 bits), **AVX-512**; in Xeon Phi (512 bits).
* Procesadores ARM: **NEON** (128 bits).
* Procesadores Power (IBM): **AltiVec** (128 bits).

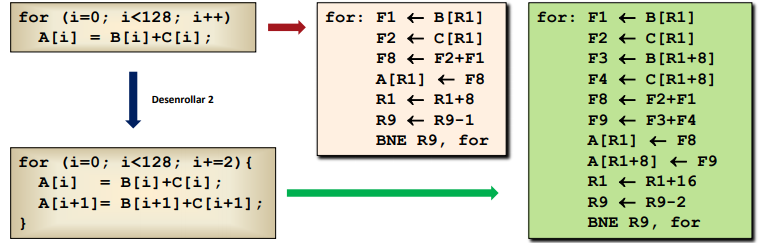
Procesadores Vectoriales:

Los primeros supercomputadores (desde los años 70 hasta mediados de los 90) fueron procesadores vectoriales. Ahora obsoletos.

Orientados al cálculo científico en coma flotante:

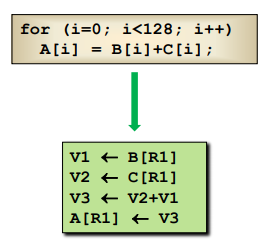
* Misma operación sobre distintos datos (vectores o matrices).
* Muchas operaciones independientes.

En un procesador de propósito general las aplicaciones científicas tienen un rendimiento bastante limitado:

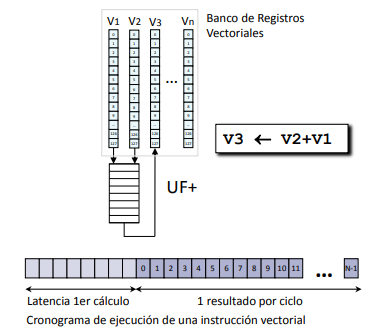


Los procesadores vectoriales disponen de instrucciones que operan con vectores de números en coma flotante:

* Usan Registros Vectoriales de tamaño fijo (MVL, p.e. 128 elementos):

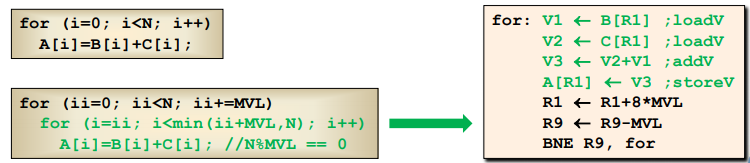


Operaciones independientes: Permite un elevado grado de segmentación de las Unidades Funcionales y tener tiempos de ciclo muy pequeños.



Características:

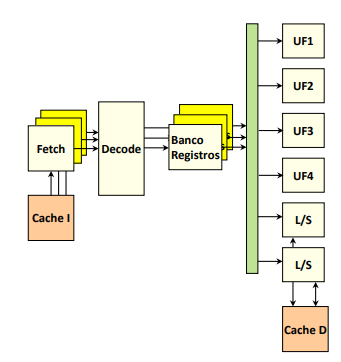
* Acceso a datos muy eficiente. Acceso a memoria con patrones conocidos (localidad espacial). Acceso a registros vectoriales (localidad temporal).
* La memoria principal está especialmente organizada para soportar accesos a memoria con patrones regulares: acceso a elementos consecutivos en memoria (stride 1); acceso a elementos con distancia constante entre ellos (stride P).
* Los datos que utilizan las U.F. Vectoriales no pasan por Memoria Cache.
* Se reduce la sobrecarga debida al control de los bucles y los saltos condicionales.
* El rendimiento de estos procesadores está limitado por la fracción de código que se puede vectorizar (Ley de Amdahl).
* El compilador (vectorizador) es un elemento fundamental del sistema.

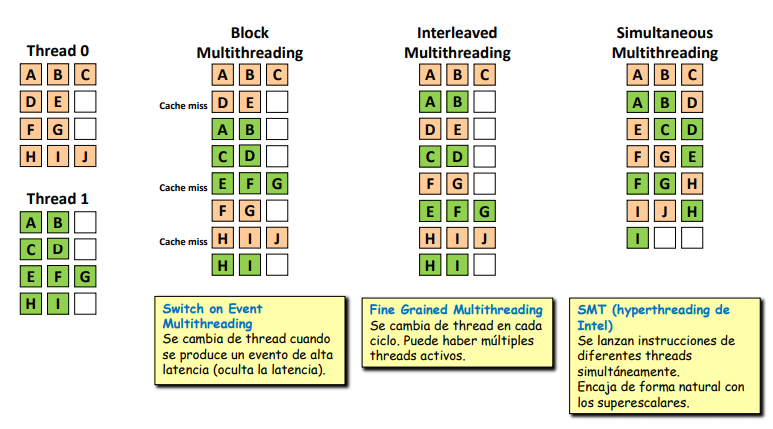


**Paralelismo a Nivel de Thread (TLP)**

Multithreading:

* Un procesador actual dispone de múltiples unidades funcionales.
* Se intenta iniciar P instrucciones/operaciones por ciclo.
* Los programas no siempre disponen de suficiente ILP.
* Una forma de aprovechar el hardware disponible es intentar ejecutar instrucciones de threads diferentes.
* Sólo es necesario multiplicar alguno de los elementos hardware: Fetch y Banco de Registros. Hay que mantener el estado de cada thread en ejecución.
* El SO ve tantas CPUs lógicas como estados puede mantener la CPU.

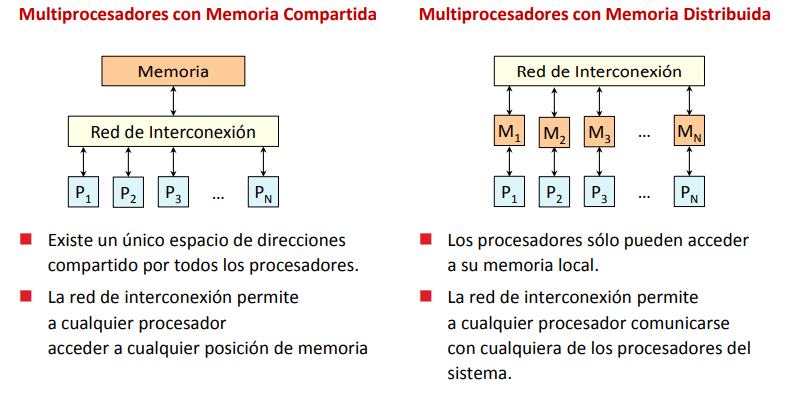




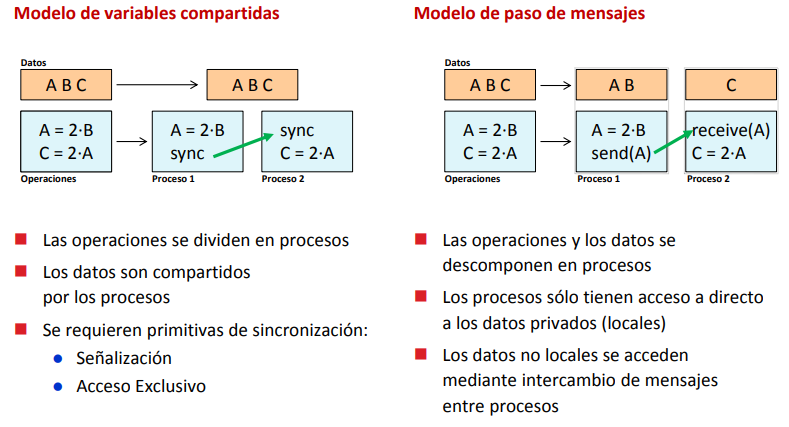
Multiprocesadores:

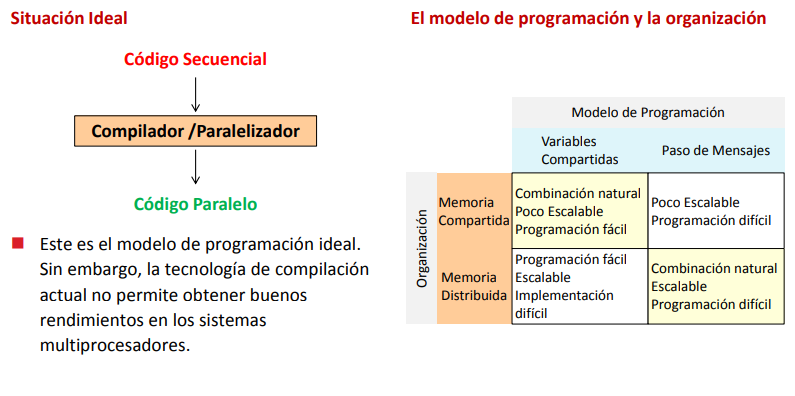
* Un multiprocesador es un computador que tiene N procesadores.
* En un mismo computador pueden ejecutarse varios threads pertenecientes a una misma aplicación o a aplicaciones independientes.
* Los sistemas multiprocesador pueden utilizarse para:
  + Ejecutar una aplicación paralela entre todos los elementos de proceso del computador **El objetivo es la velocidad**: Supercomputación.
  + Ejecutar más aplicaciones por unidad de tiempo **El objetivo es el throughput**: Servidores de aplicaciones.
  + O una mezcla de ambos.

Organización:



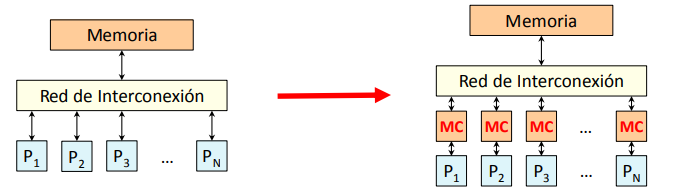
Programación:





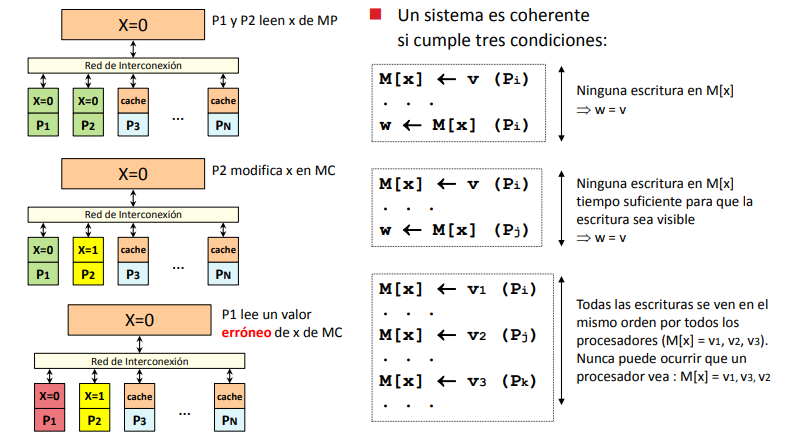
Mejora de la Organización:

* La red de interconexión aumenta la latencia con memoria.
* El uso de memorias cache locales de gran capacidad es imprescindible.



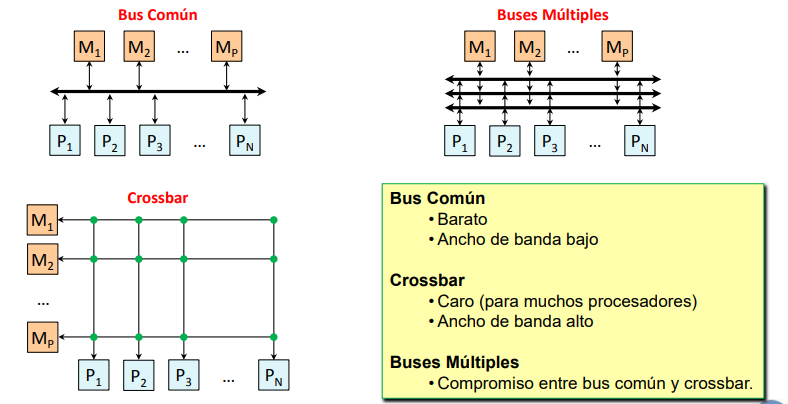
* El uso de memorias cache locales en un entorno de memoria compartida provoca la aparición de un problema que el hardware ha de resolver: COHERENCIA de MEMORIA.
* Un sistema de memoria es coherente si cualquier lectura de un dato devuelve el último valor escrito sobre esa posición de memoria. Existen 2 temas a tener en cuenta:
  + Coherencia.
  + Consistencia.

El Problema de la Coherencia de Memoria:

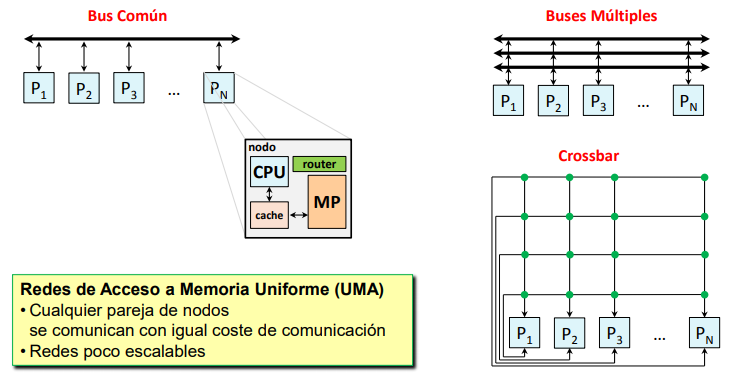


Redes de Interconexión:

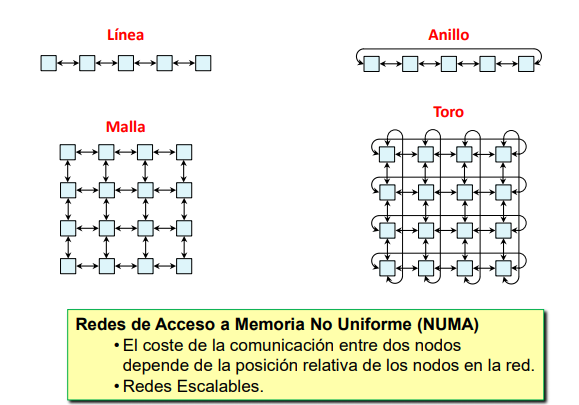
Elemento fundamental en el rendimiento de un multiprocesador.



La red de interconexión también es fundamental en los multiprocesadores con memoria distribuida.



Los multiprocesadores con memoria distribuida pueden utilizar conexiones punto a punto.



**Ejemplos Reales**: Un procesador de propósito general, un TV 3D, una GPU, una consola, un SoC para móvil, Supercomputador MareNostrum…