

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación



IIC2343 – Arquitectura de Computadores

Paralelismo avanzado

Profesor: Jurgen Heysen

Ejecución secuencial vs ejecución paralela

- Vimos anteriormente que aumentar la frecuencia del clock no es la única manera de acelerar el procesamiento.
- En esta clase revisaremos **arquitecturas paralelas** desde el punto de vista de instrucciones y datos.
- Estas arquitecturas requieren hardware más complejo, pero sus ventajas son potencialmente mucho mayores que sólo aumentar la frecuencia del clock o dividir una instrucción en varias etapas.

Taxonomía de Flynn nos permite categorizar arquitecturas en base al paralelismo

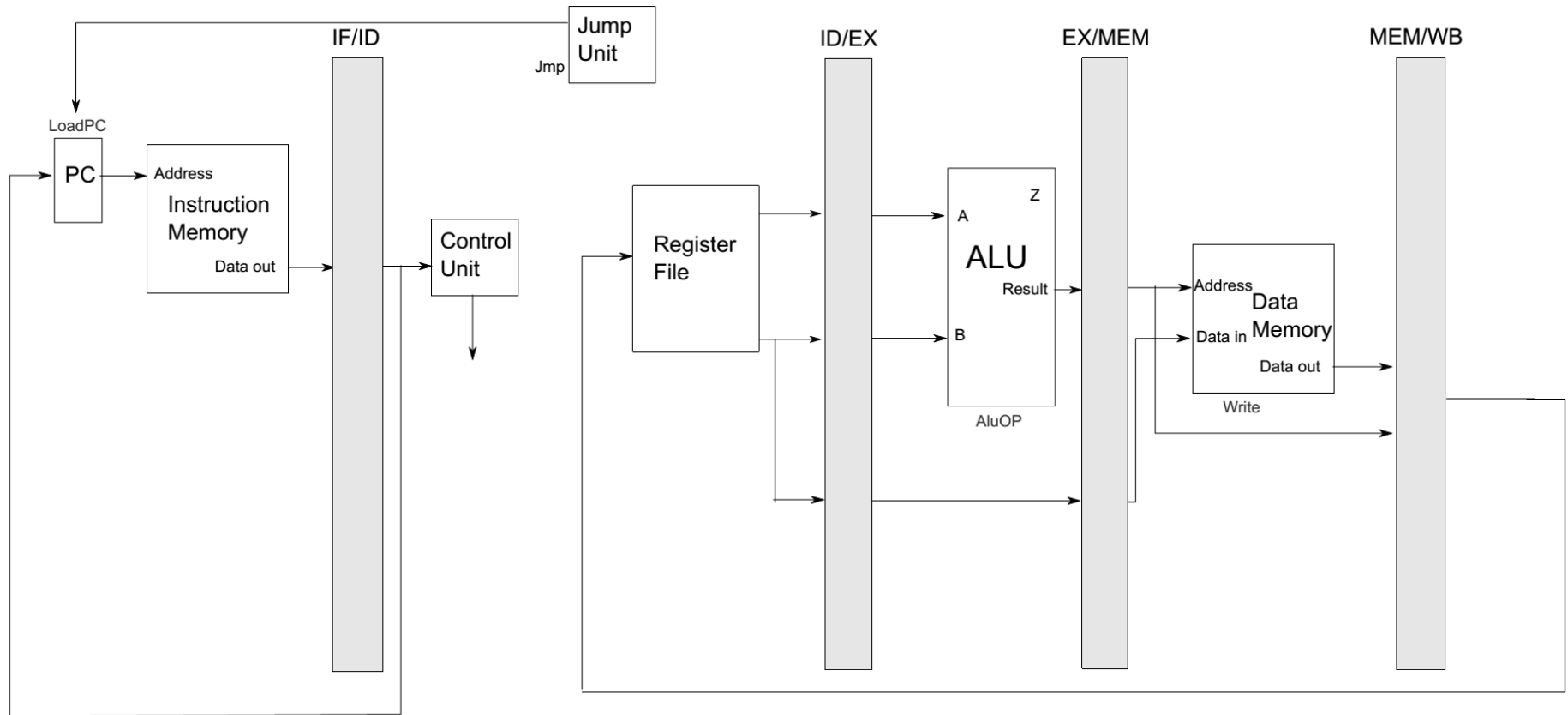
Dependiendo de si utilizamos múltiples programas y/o múltiples fuentes de datos, la taxonomía de Flynn nos entrega 4 posibles tipos de arquitectura.

	Single Instruction	Multiple Instruction
Single Data	SISD	MISD
Multiple Data	SIMD	MIMD

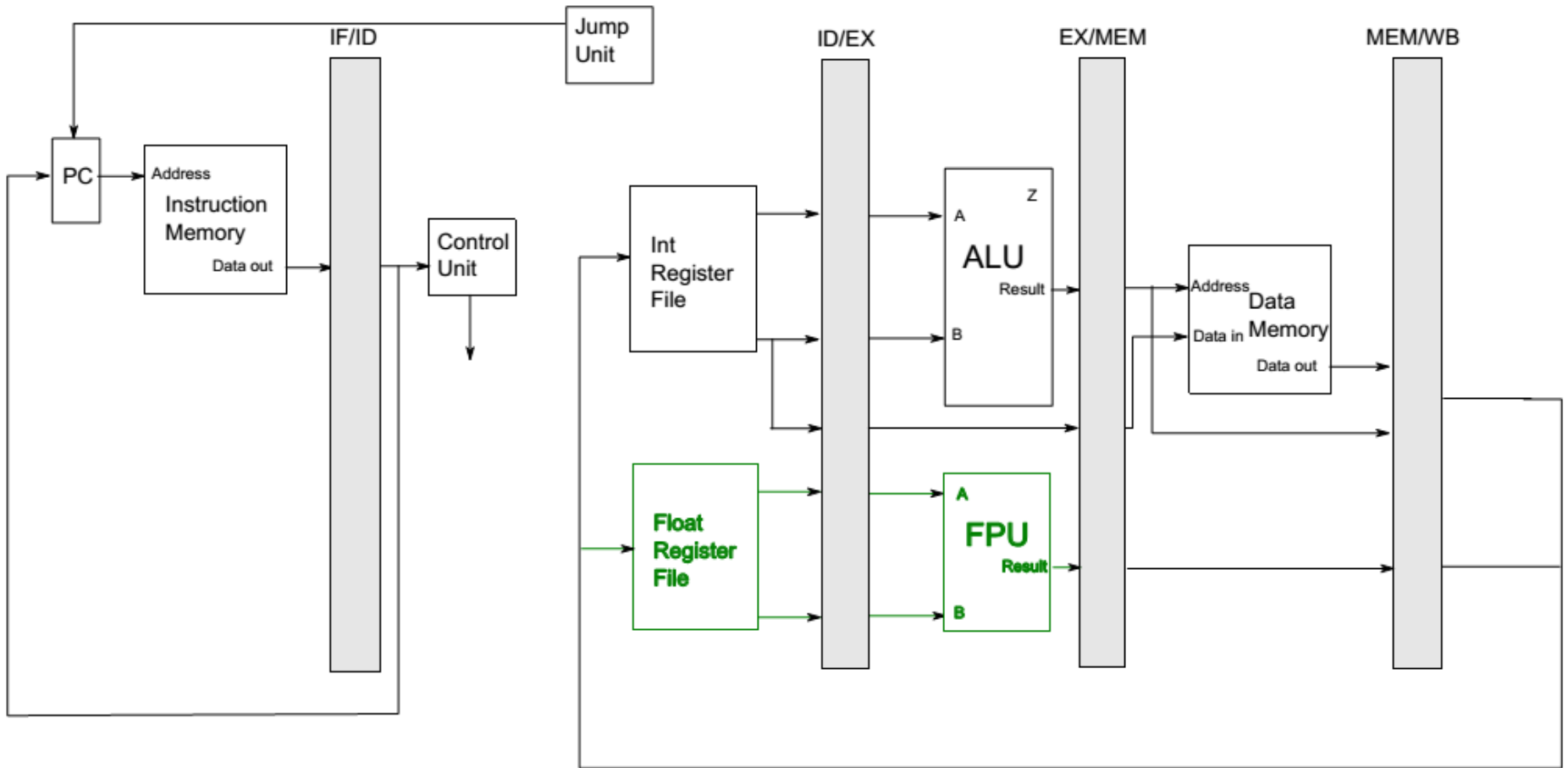
Paralelismo SISD es una generalización del paralelismo a nivel de instrucción (ILP)

- Nace de una idea simple: que pasa si agregamos una unidad de ejecución secundaria al procesador, y así permitir que este ejecute más de una instrucción al mismo tiempo
- Un ejemplo canónico de esto es agregar una unidad de procesamiento de números de punto flotante, FPU.

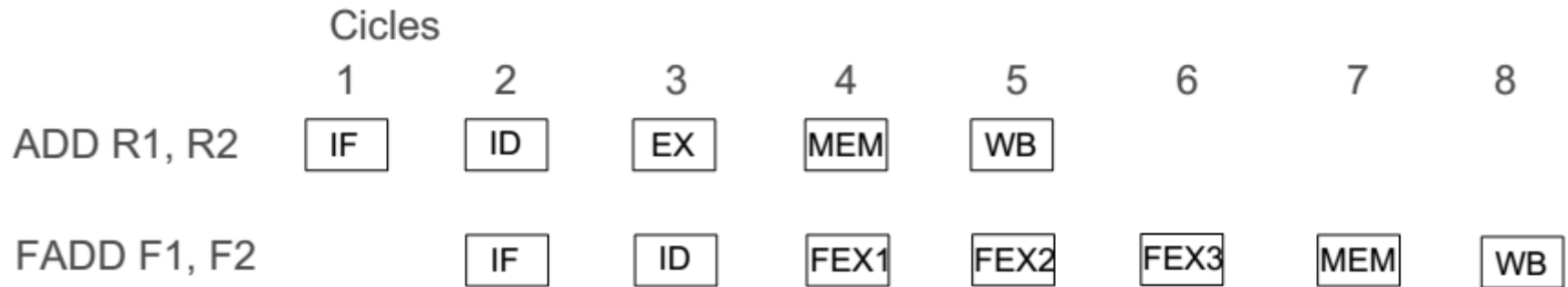
Revisemos el computador básico con pipeline



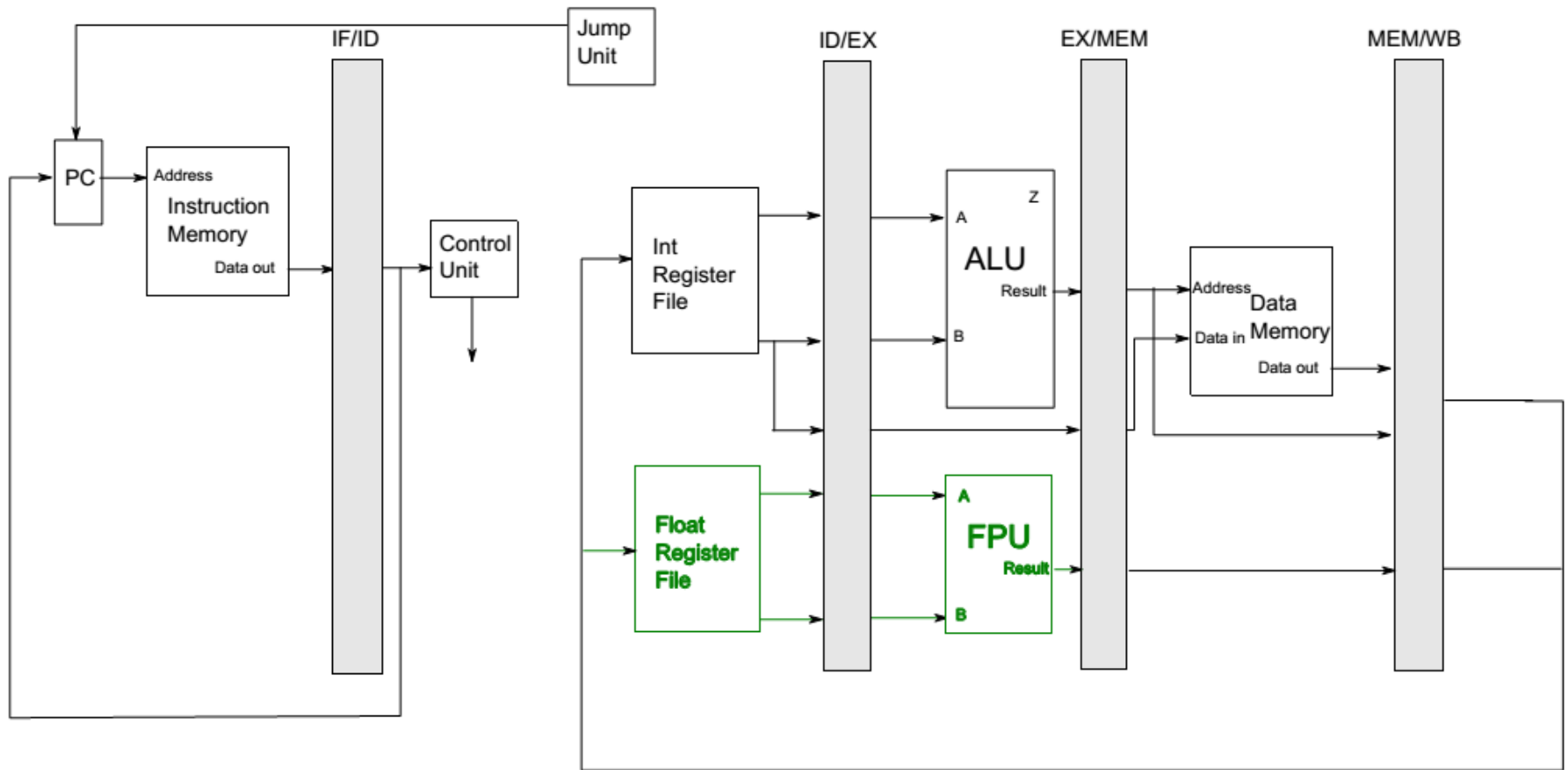
Agregamos un **register file de floats** y una **FPU**



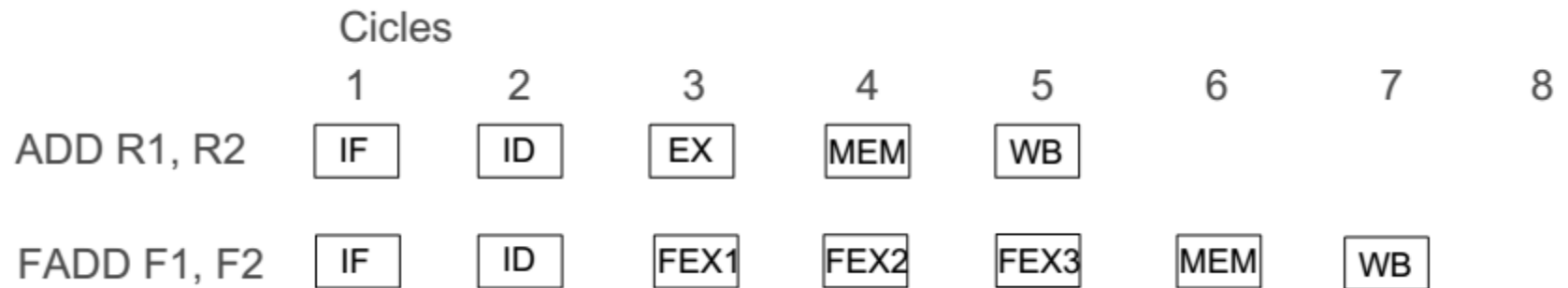
Supongamos que la etapa **EX** de la ejecución de la **FPU** se puede dividir en **3 subetapas**



EX y WB de ambas instrucciones son independientes, ¿cómo podemos aprovechar esto?



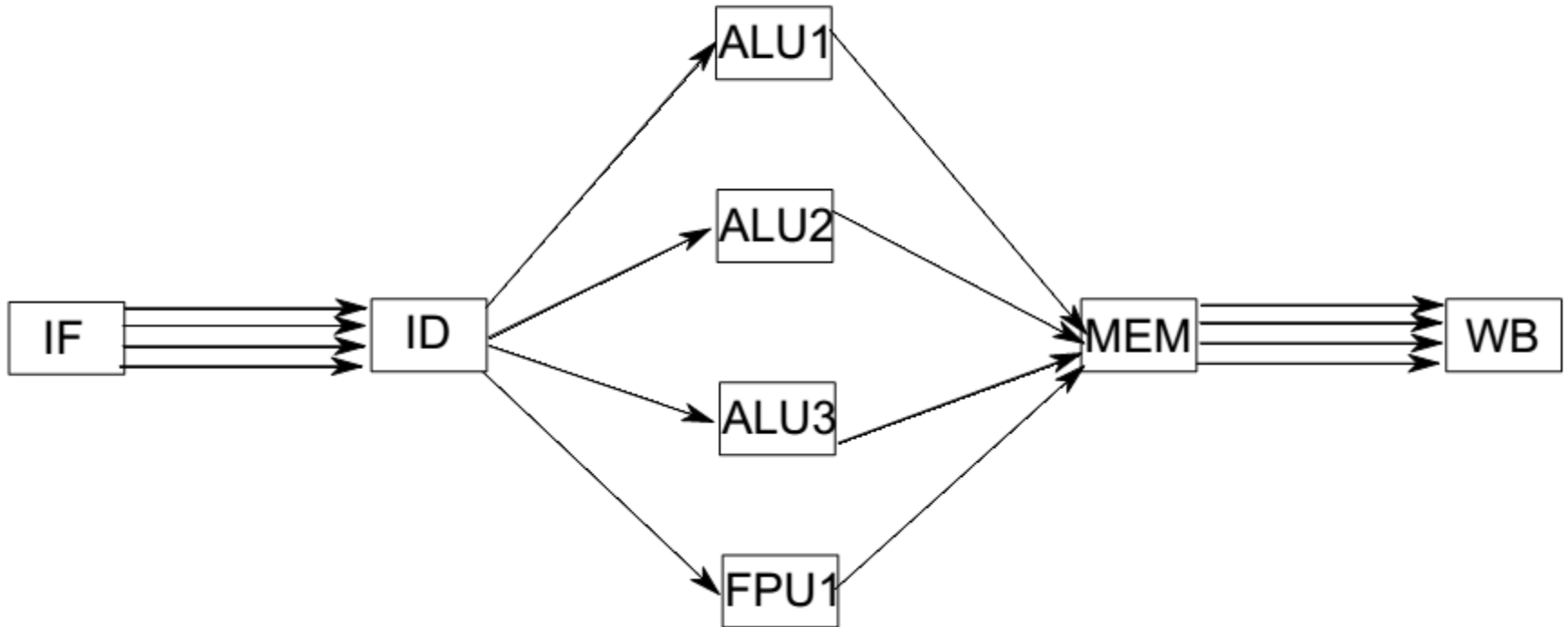
Aumentamos capacidad de IF e ID para lograr paralelismo



Concepto SISD se puede extender a múltiples ALUs, registros, FPUs, etc

- Un procesador que permite obtener, decodificar y ejecutar múltiples instrucciones al mismo tiempo se conoce como **multiple-issue**.
- Si es capaz de procesar 2 instrucciones al mismo tiempo, será un procesador 2-issue, si procesa 4, un 4-issue, etc.

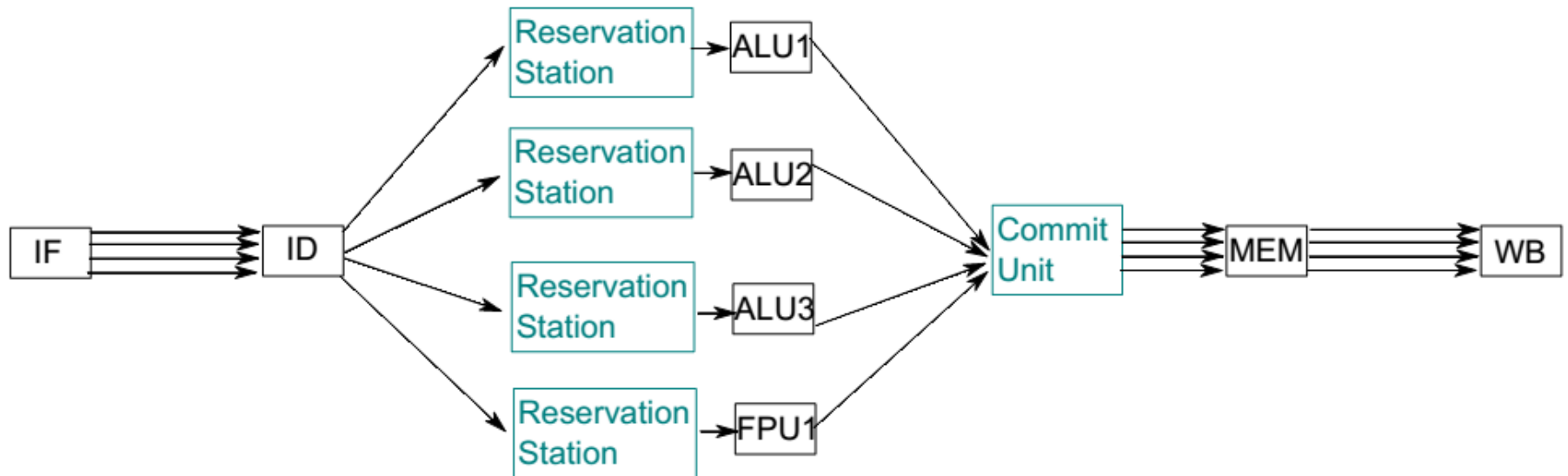
Concepto SISD se puede extender a multiples ALUs, registros, FPU's, etc



Procesadores **multiple-issue** requieren elementos para decidir sobre paralelismo

- Existen dos tipos de técnicas para realizar esto: **estáticas y dinámicas**.
- Técnicas estáticas dependen del compilador para agrupar instrucciones paralelizables.
- Técnicas dinámicas permiten a la CPU determinar en tiempo de ejecución las instrucciones a paralelizar, despachándolas a unidades de ejecución distintas.

Técnica dinámica más usada es
la **Arquitectura Superescalar**



Técnica estática más utilizada es Very Large Instruction Word (VLIW)

- Compilador genera un paquete (**bundle**) de instrucciones que pueden ejecutarse en paralelo.
- **Bundle** es enviado al procesador como una instrucción muy larga
- CPU reordena las instrucciones del grupo y lo envía en paralelo a las distintas unidades de ejecución.

Dirección	Instrucción
0x00	Instrucción 1
0x01	Instrucción 2
0x02	Instrucción 3
0x03	Instrucción 4
0x04	Instrucción 5
0x05	Instrucción 6
0x06	Instrucción 7
0x07	Instrucción 8
0x08	Instrucción 9

Tabla 2: Secuencia de instrucciones sin VLIW.

Dirección	Bundle			
0x00	Instrucción 1	Instrucción 6	Instrucción 7	NOP
0x01	NOP	NOP	Instrucción 3	Instrucción 4
0x02	NOP	Instrucción 2	NOP	NOP
0x03	NOP	Instrucción 5	Instrucción 9	NOP
0x04	NOP	NOP	NOP	Instrucción 8

Tabla 3: Secuencia de bundles con VLIW.

SISD tiene problemas de complejidad

- Para entregar buen rendimiento y manejar todas las posibles situaciones, los procesadores aumentan enormemente su complejidad.
- Esto implica un aumento en el costo y en el uso de energía.
- Una alternativa a esto es utilizar múltiples procesadores simples.

Taxonomía de Flynn nos permite categorizar arquitecturas en base al paralelismo

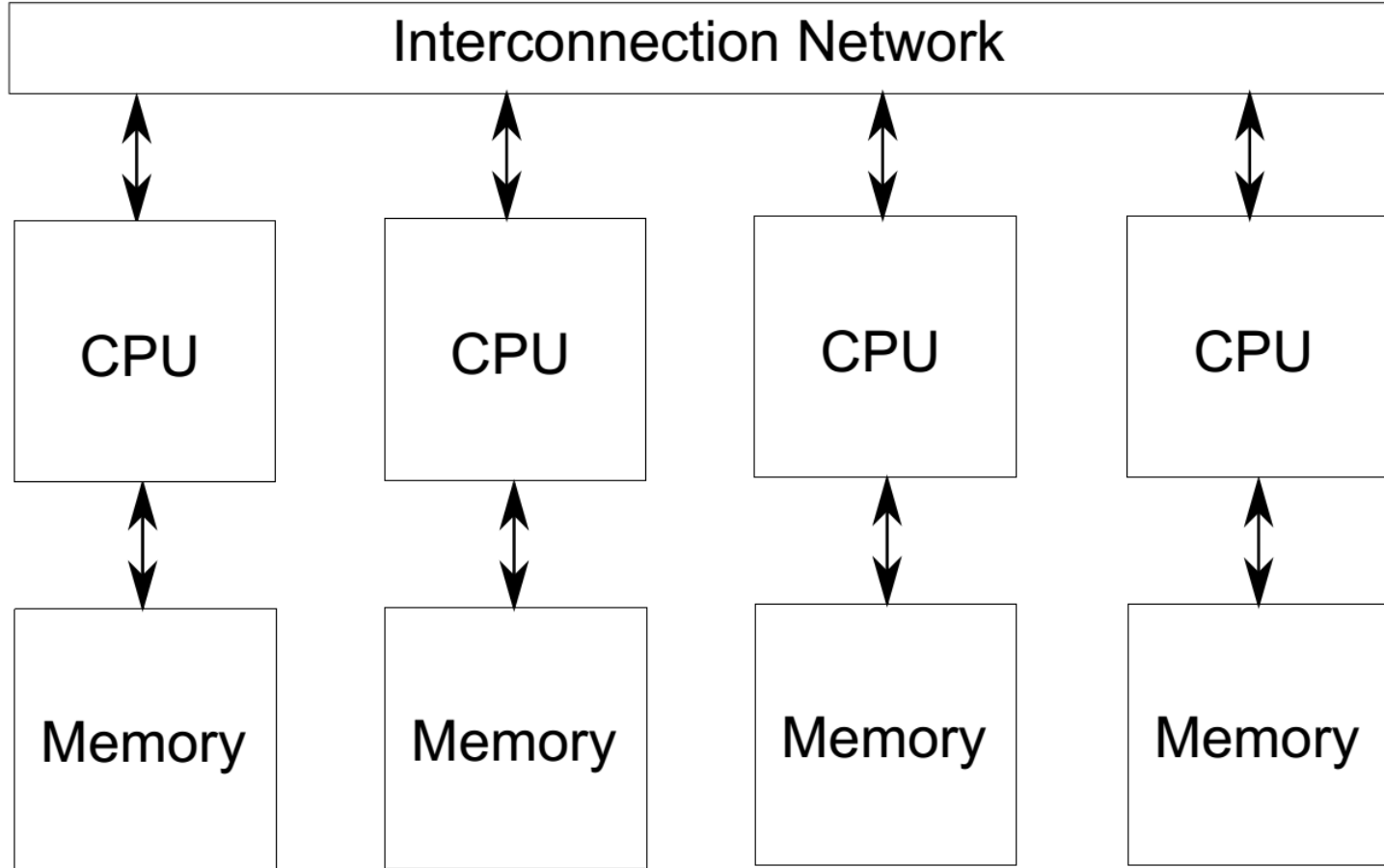
Dependiendo de si utilizamos múltiples programas y/o múltiples fuentes de datos, la taxonomía de Flynn nos entrega 4 posibles tipos de arquitectura.

	Single Instruction	Multiple Instruction
Single Data	SISD	MISD
Multiple Data	SIMD	MIMD

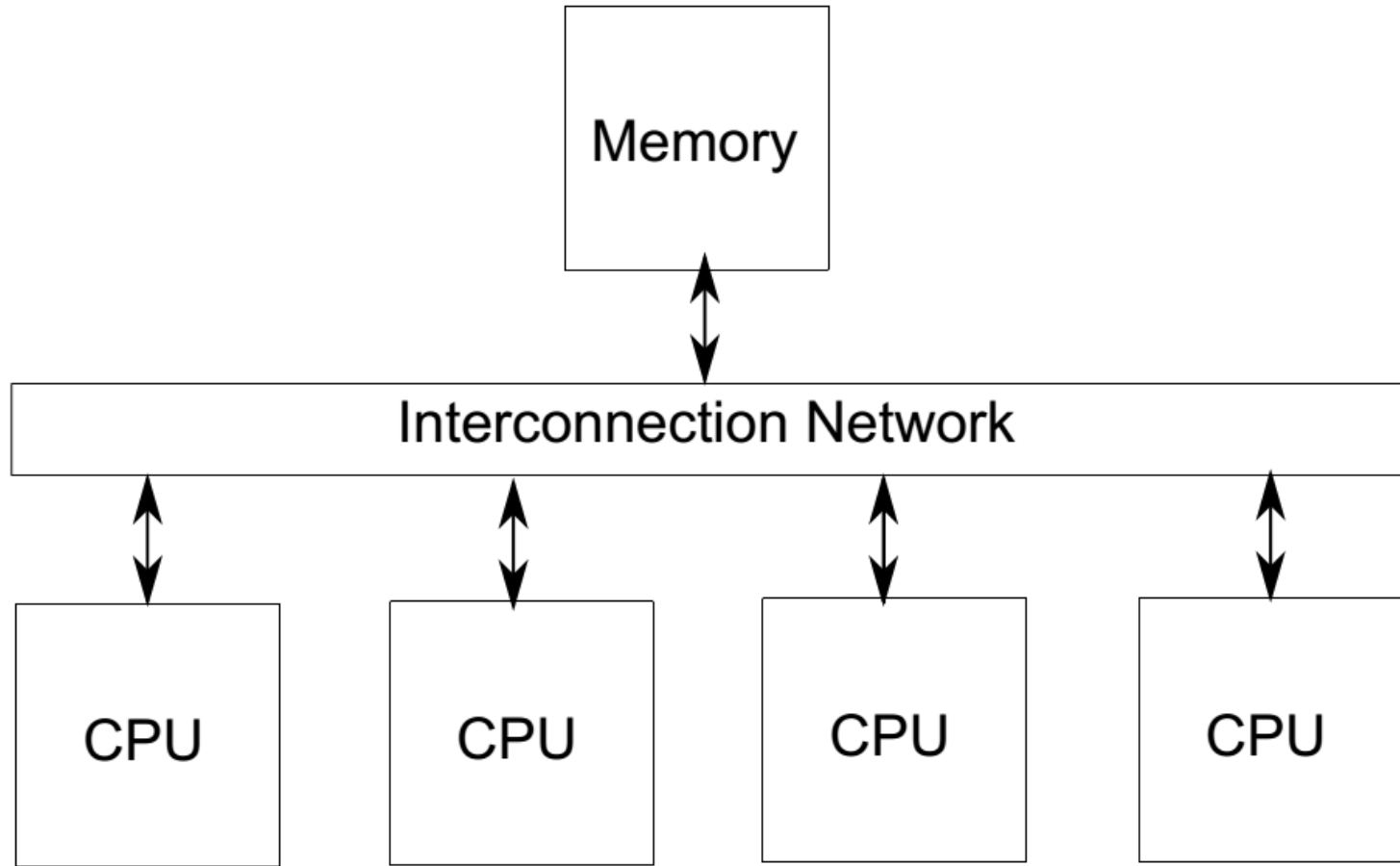
MIMD permite disminuir complejidad del procesador

- Ideal para tareas independientes.
- A los sistemas que lo implementan se les conoce como sistema multiprocesador.
- Los más usados son multiprocesador por **paso de mensajes** y multiprocesador de **memoria compartida**.

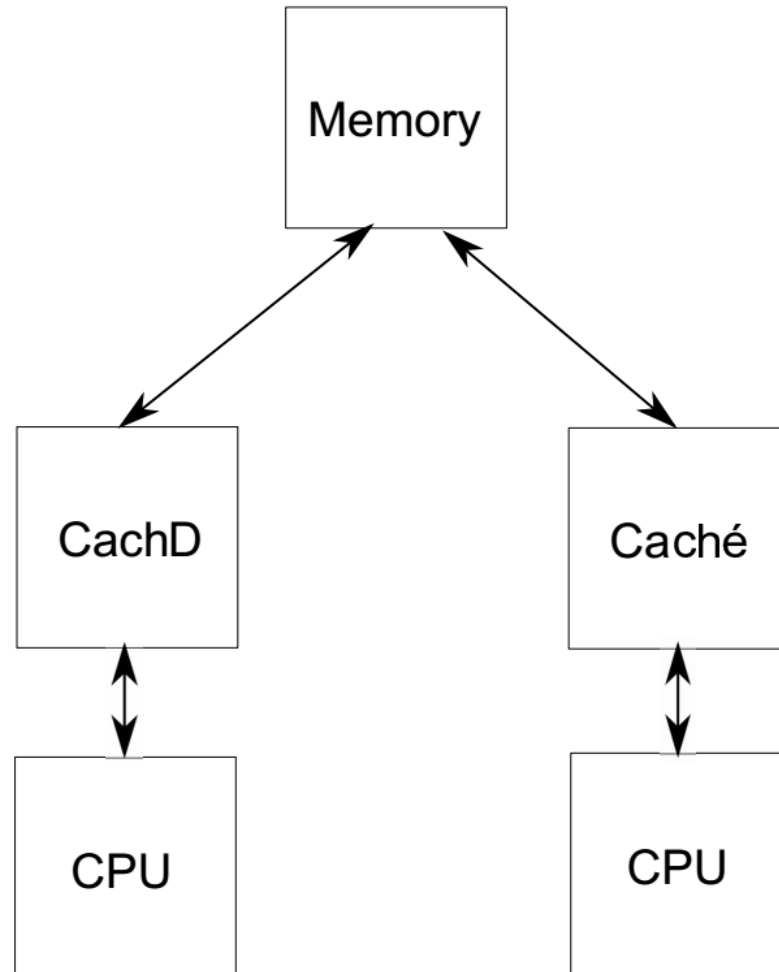
Multiprocesador por paso de mensajes puede ser un **clúster** o un **sistema distribuido**



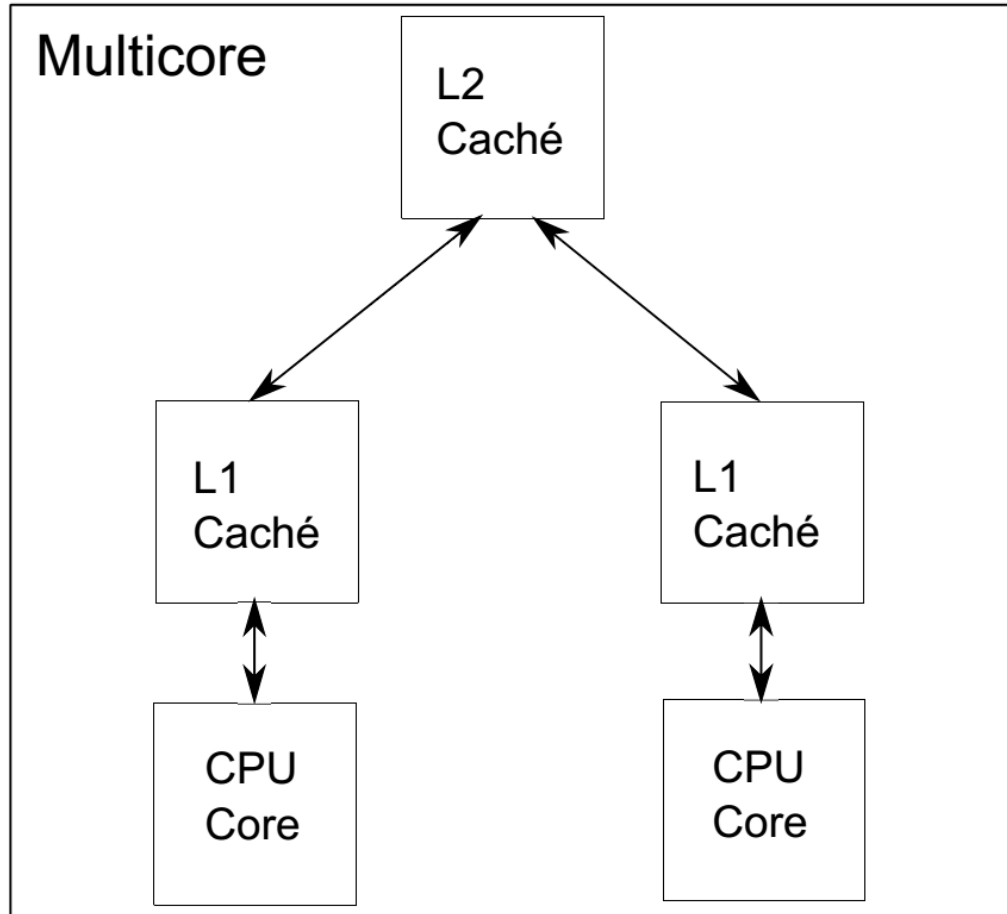
Multiprocesador con **memoria compartida** permite una comunicación más rápida...



..., pero necesita mecanismos para mantener coherencia en memoria, principalmente en la caché

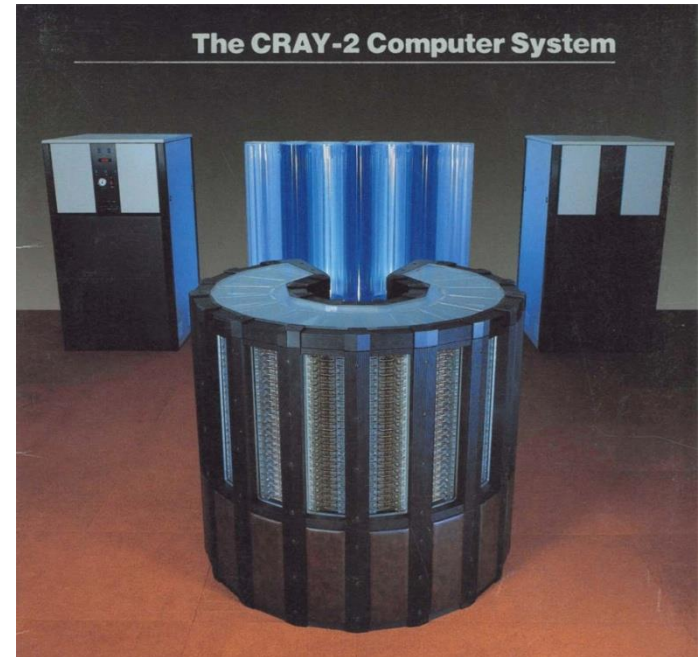


Procesadores *multicore* implementan el multiprocesamiento en un solo chip



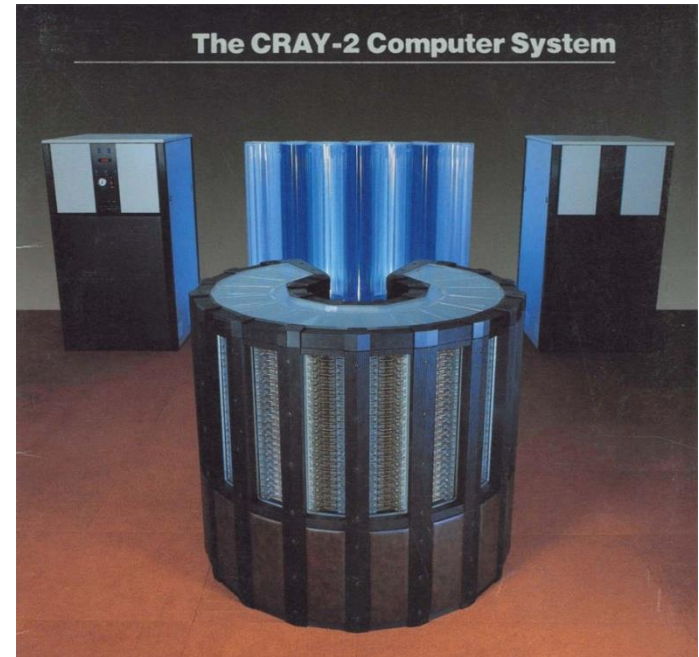
¿Cómo era un **supercomputador** hace 30 años?

- Problemas requerían pocos datos: almacenamiento y transferencia no eran problema.
- Procesamiento **centralizado**.
- Pocas CPUs.
- Procesamiento paralelo a costa de hardware especializado y caro.
- Foco en alto **throughput**.



¿Cómo era un **supercomputador** hace 30 años?

- Cray-2: lanzado en 1985.
- Fue el supercomputador más rápido del mundo hasta 1990.
- 8 CPUs.
- Máx. throughput: 1.9 GigaFlops.



¿Cómo son los **supercomputadores** en la **actualidad**?

- Procesamiento **distribuido** (clústers) basado en hardware de menor costo (**commodity hardware**).
- Miles de nodos, CPUS y núcleos.
- Procesamiento altamente paralelo.
- Foco en alto **throughput**, **escalabilidad** y **robustez**.
- Problemas requieren **muchos** datos: almacenamiento y transferencia **son un problema**.



¿Cómo son los **supercomputadores** en la **actualidad**?

- Tianhe-2: lanzado en 2013.
- Más rápido del mundo hasta mediados de 2016.
- Máx. throughput: 33.86 PetaFlops.
- 16K nodos, 88 GB RAM cada uno.
- Más de 3M de núcleos y 1375 TB de RAM.



Taxonomía de Flynn nos permite categorizar arquitecturas en base al paralelismo

Dependiendo de si utilizamos múltiples programas y/o múltiples fuentes de datos, la taxonomía de Flynn nos entrega 4 posibles tipos de arquitectura.

	Single Instruction	Multiple Instruction
Single Data	SISD	MISD
Multiple Data	SIMD	MIMD

Paralelismo **SIMD** se presenta de manera natural en múltiples problemas

- Consiste en la ejecución de un mismo programa/instrucción sobre múltiples datos distintos.
- Este tipo de operación se da de manera natural en los cálculos matriciales y vectoriales.
- Hardware especializado permite sacar provecho de las características del problema.

Instrucciones multimedia (SIMD) son parte central de nuevas arquitecturas de CPU

- Introducidas por Intel en 1997 (MMX)
- Cada nueva generación de procesadores agrega nuevas instrucciones (MMX, SSE, AVX).
- Agregan también nuevos registros de gran tamaño (512 bits en AVX2).
- Instrucciones especiales realizan operaciones en paralelo sobre muchos números de menor tamaño
- Por ejemplo, se puede multiplicar en paralelo 16 floats de 32 bits, usando registros de 512 bits.

GPUs también son dispositivos SIMD (SIMT)

- Dispositivo de I/O, especializado en cálculos masivamente paralelos.
- Origen en aceleración de contenidos gráficos en 3D.
- CPU traspasa el cálculo de ciertos elementos a la GPU.
- Mientras la GPU trabaja, la CPU está libre.

GPUs se especializan en problemas “ridículamente” paralelos

- Se centraban originalmente sólo en el procesamiento de cada uno de los píxeles.
- Al ser independientes, era posible aplicar unas cuantas funciones fijas a cada píxel en paralelo.
- Además, al ser en una sola dirección, desde vértices a píxeles, se puede usar un pipeline.
- GPUs modernas tienen cientos de pequeños procesadores, que pueden usarse para cómputo general.

GPUs son en realidad procesadores **SIMT**

MIMD/SPMD



Múltiples threads independientes

SIMD/Vector



Un thread con gran “amplitud” de datos

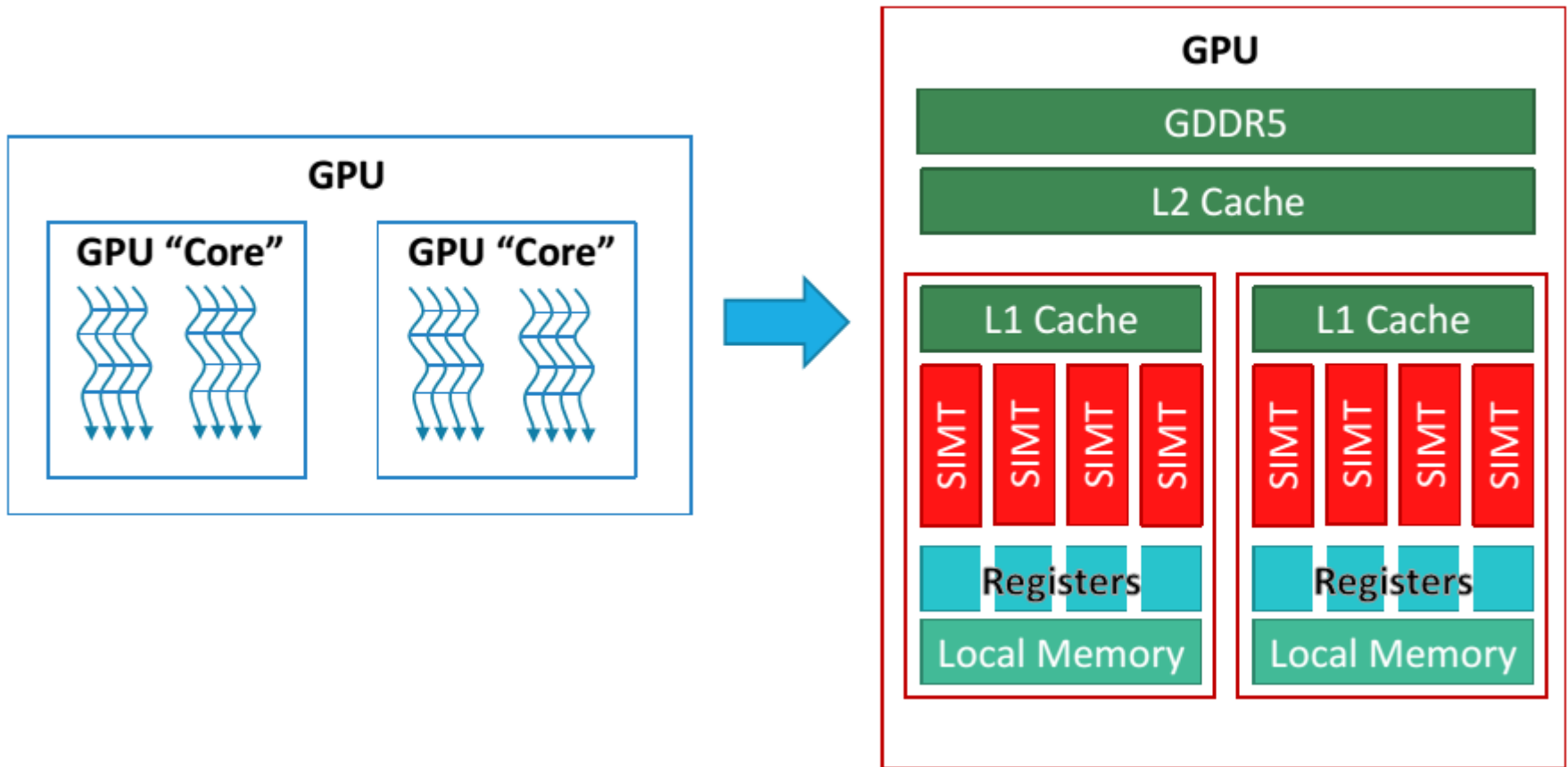
SIMT



Múltiples threads sincronizados

Arquitectura	CPU Multicore	X86 SSE/AVX	GPU
Pro	Propósito general	Puede mezclar código secuencial y paralelo	Escritura y lectura más eficiente y rápida
Contra	Malo para paralelismo de datos	Escritura y lecturas de memoria son complejas	Latencia de memoria y divergencia de código.

GPUs son en realidad **múltiples** procesadores **SIMT**



Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación



IIC2343 – Arquitectura de Computadores

Paralelismo avanzado

Profesor: Jurgen Heysen