Modelos y Clasificación de los Sistemas Paralelos

CC3069 – Computación Paralela y Distribuida

Ciclo 1 - 2022 – Semana 1 (ver 2.0)

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

Agenda

- Arquitectura Von Neumann y sus modificaciones
- Niveles de Paralelismo
- Clasificación de sistemas
- Modelos de Máquinas paralelas

Computadoras Modernas

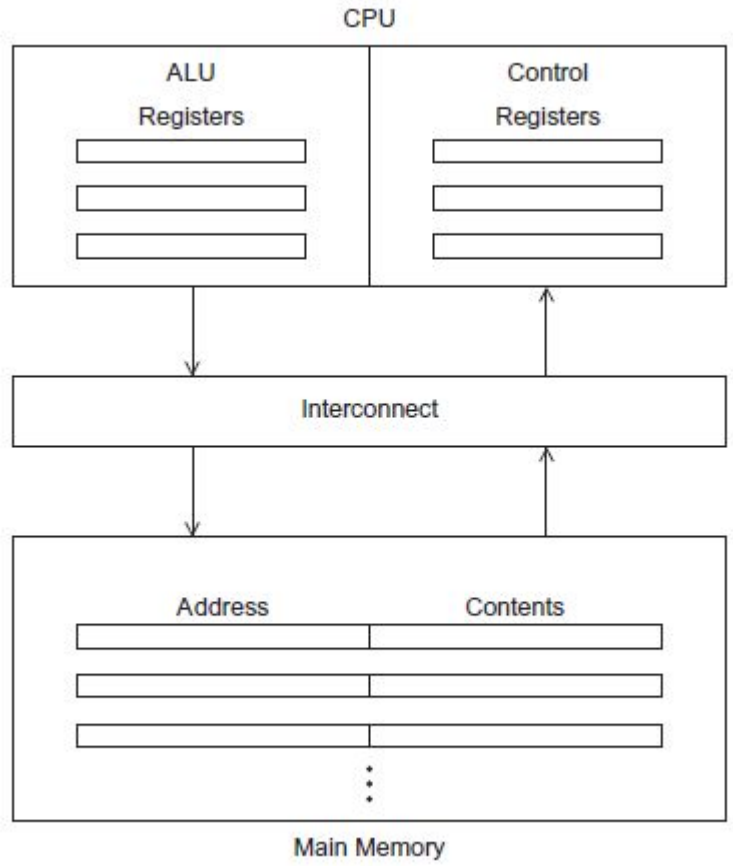
- La *Ley de Moore* predecía el crecimiento de los equipos uniprocesador. *Ya no es válida*.
- A partir del 2002 debemos adoptar *estrategias paralelas para seguir mejorando* el desempeño (**performance**) y rendimiento (**throughput**)
- *Toda computadora moderna es paralela* (escritorio, server, móvil, embebida)

Arquitectura Von Neumann

Arquitectura de Von Neumann

Fundamental para todos los sistemas:

- Unidad de Procesamiento (UP): CPU / Core
- Sistema de interconexión (bus)
- Memoria Primaria



Arquitectura de Von Neumann

(1) *En equipos, investigue sobre dos ejemplos de ventajas y desventajas de la Arquitectura de Von Neumann*

Pacheco
Ch 2
Pp 15-17

PROS

CONS

Arquitectura de Von Neumann

Cuello de botella de Von Neumann

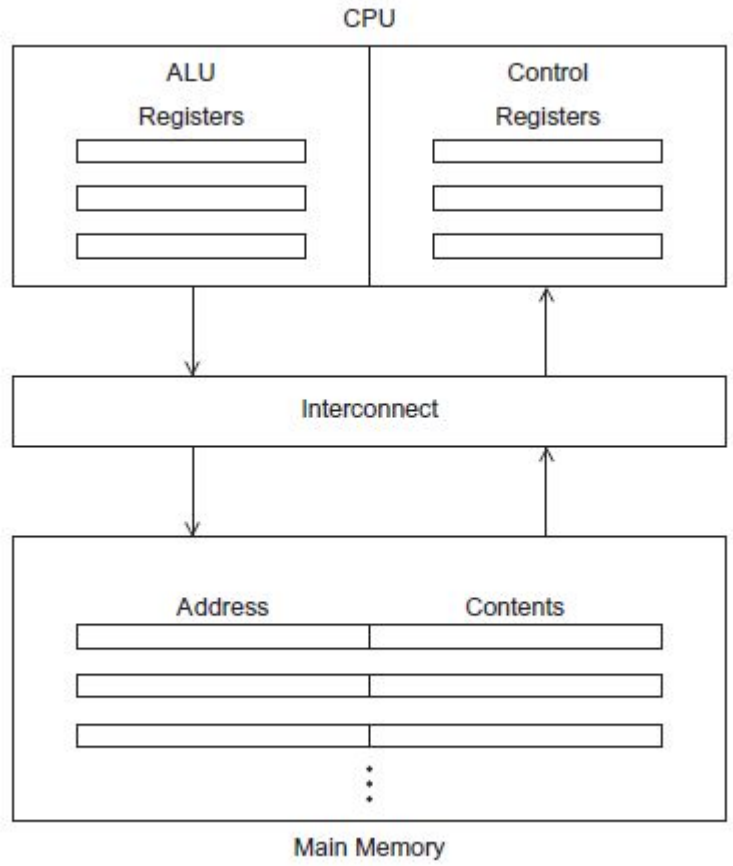
- Igual forma de acceder a datos en I/O y memoria
- Tamaño y características del bus de interconexión
- Simplicidad el diseño de las unidades de control
- Reescritura de instrucciones
- Igual forma de recuperar datos e instrucciones
- Procesamiento secuencial (PC – program counter)

PROS

CONS

Cuello de botella de Von Neumann

- Instrucciones ejecutadas una a la vez
- El bus puede limitar el intercambio entre CPU y memoria



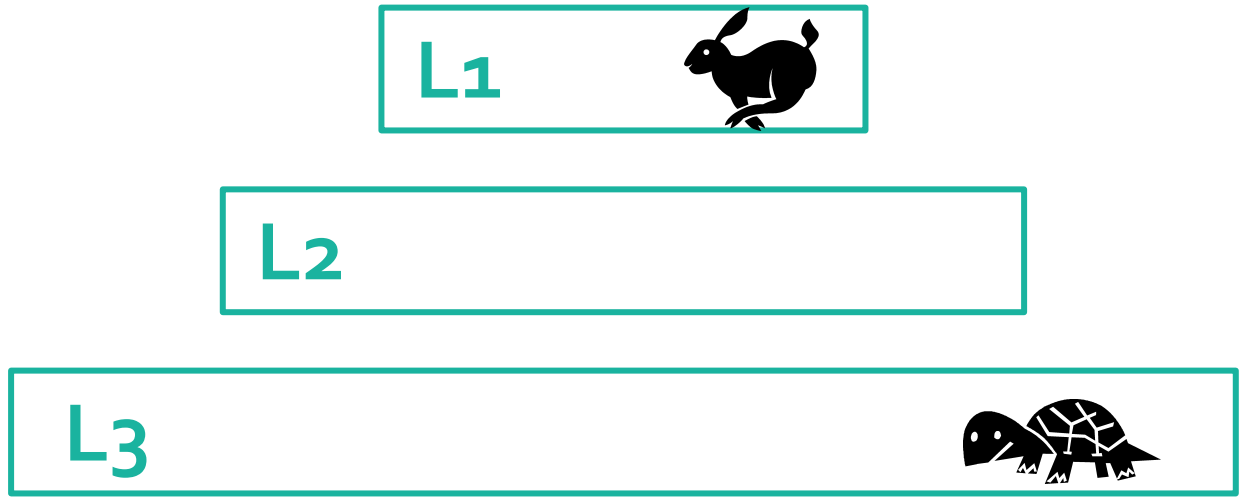
Modificaciones a Von Neumann

JAMBOARD

Modificaciones a la Arquitectura de Von Neumann

- **Arquitectura Harvard** – Canales separados de acceso a memoria de datos y de instrucciones.
- **Caché** – Jerarquía de memorias para almacenaje temporal.
- **Memoria virtual** – mapeo de memoria primaria a medios de almacenaje.
- **Paralelismo de HW** – múltiples recursos y estrategias para procesamiento paralelo a nivel de HW (pipeline, multithreading, multiple issue)

Memoria Caché



- Memoria en el mismo chip (o fuera pero con una interconexión de alta velocidad)
- Aprovecha la localidad
  - Espacial – datos contiguos
  - Temporal – datos con mayor probabilidad inmediata de ser requeridos

Localidad

```
float z[1000];
…
sum = 0.0;
for (i = 0; i < 1000; i++)
    sum += z[i];
```

- Espacial – La lectura de z[i], ,luego z[i+1] …
- Temporal – La dirección de memoria donde se encuentra la dirección ***for*** será usada en varias iteraciones

Consideraciones:

Inconsistencia y competencia

Con varios niveles de caché o un caché compartido por varias UPs:

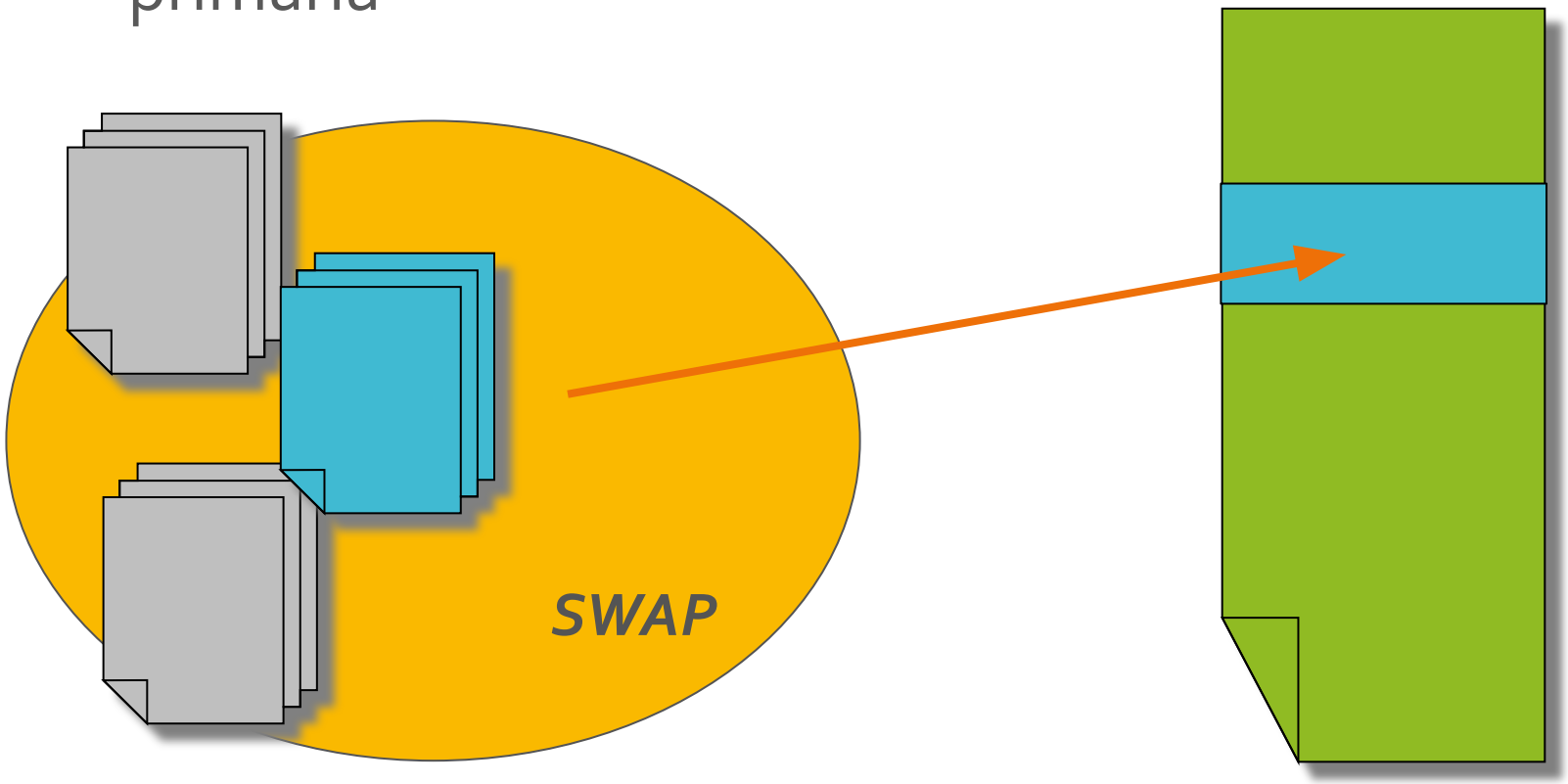
- Inconsistencia – copias de un valor compartido tiene diferentes valores al haber cachés privadas
- Competencia – un valor compartido en una caché compartida puede ser accedido al mismo tiempo

Memoria Virtual

- Programas que requieren gran cantidad de datos
- Equivalente a una caché pero usando el disco duro
- Localidad (espacial & temporal)
- Carga a memoria páginas activas

Swap y page

- Swap – espacio en HD para las páginas inactivas
- Page – bloques de datos que se cargan a la vez a la memoria primaria



Paralelismo a nivel de HW

Estrategias a nivel de unidades funcionales en los CPUs que ayudan a mejorar el desempeño y buscan paralelismo. No tenemos mucho control sobre estas:

- **Pipelining** – división en segmentos de tareas específicas para el proceso de una instrucción
- **Multiple emisión** – Multiplicidad de unidades funcionales del pipeline para trabajar de forma concurrente
- **Vectorización** – posibilidad de ejecutar las mismas instrucciones sobre muchos datos (*)
- **Especulación** – predice si una decisión (branch) será tomada o no en base al comportamiento anterior

Niveles de Paralelismo

Paralelismo de aplicaciones (más control / responsabilidad del usuario)

Data-level (DLP) - elementos de datos que pueden procesarse al mismo tiempo

Task-level (TLP) - tareas que pueden operar independientemente y al mismo tiempo o con traslape

Paralelismo de dispositivos

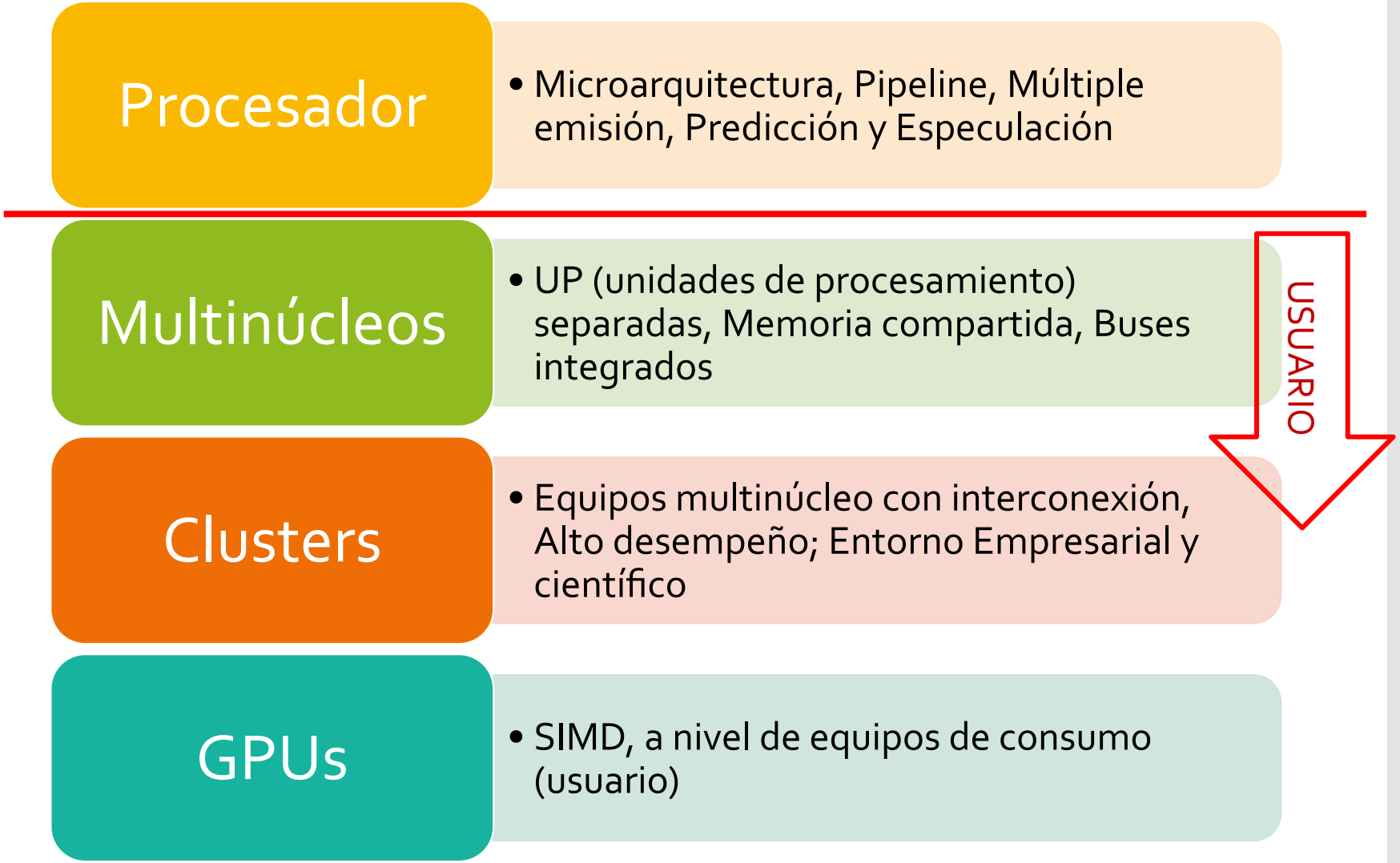
Instruction-level (ILP) - aprovecha DLP a través del compilador, aprovechando pipelines y ejecución especulativa (branch prediction).

Arquitecturas vectoriales, GPUs y extensión al repertorio de instrucciones - aprovecha DLP aplicando una misma instrucción a un conjunto de datos en paralelo.

Thread-level - aprovecha DLP o TLP mediante modelos de hardware que permiten interacción entre hilos paralelos.

Request-level - aprovecha TLP mediante modelos de hardware de tareas desacopladas y especificadas por el programador o el OS.

Paralelismo en múltiples niveles



Tipos de Sistemas Paralelos

Memoria Compartida

- Múltiples UP
- Una memoria

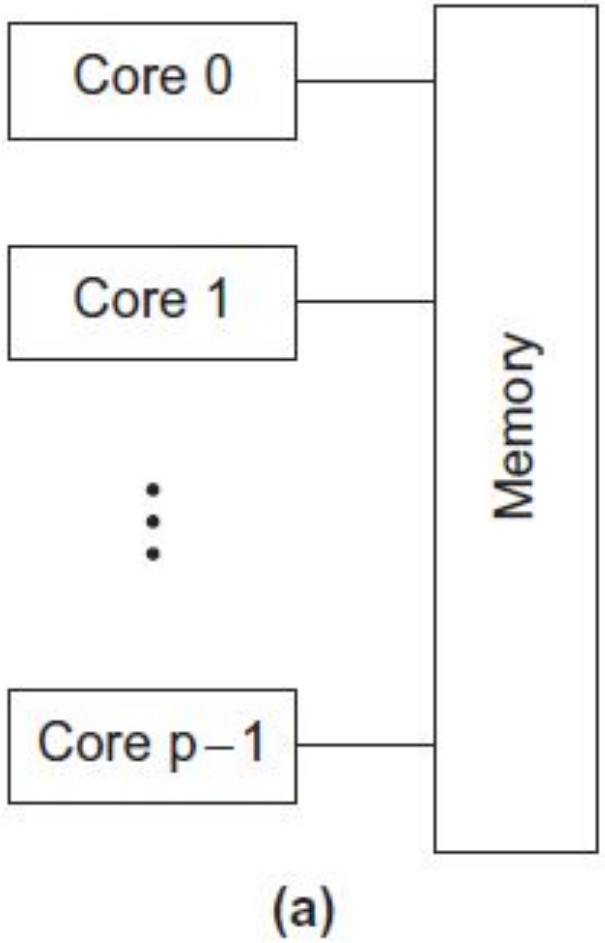
Sistemas Distribuidos

- Múltiples nodos
- Redes de interconexión rápidas

GPUs

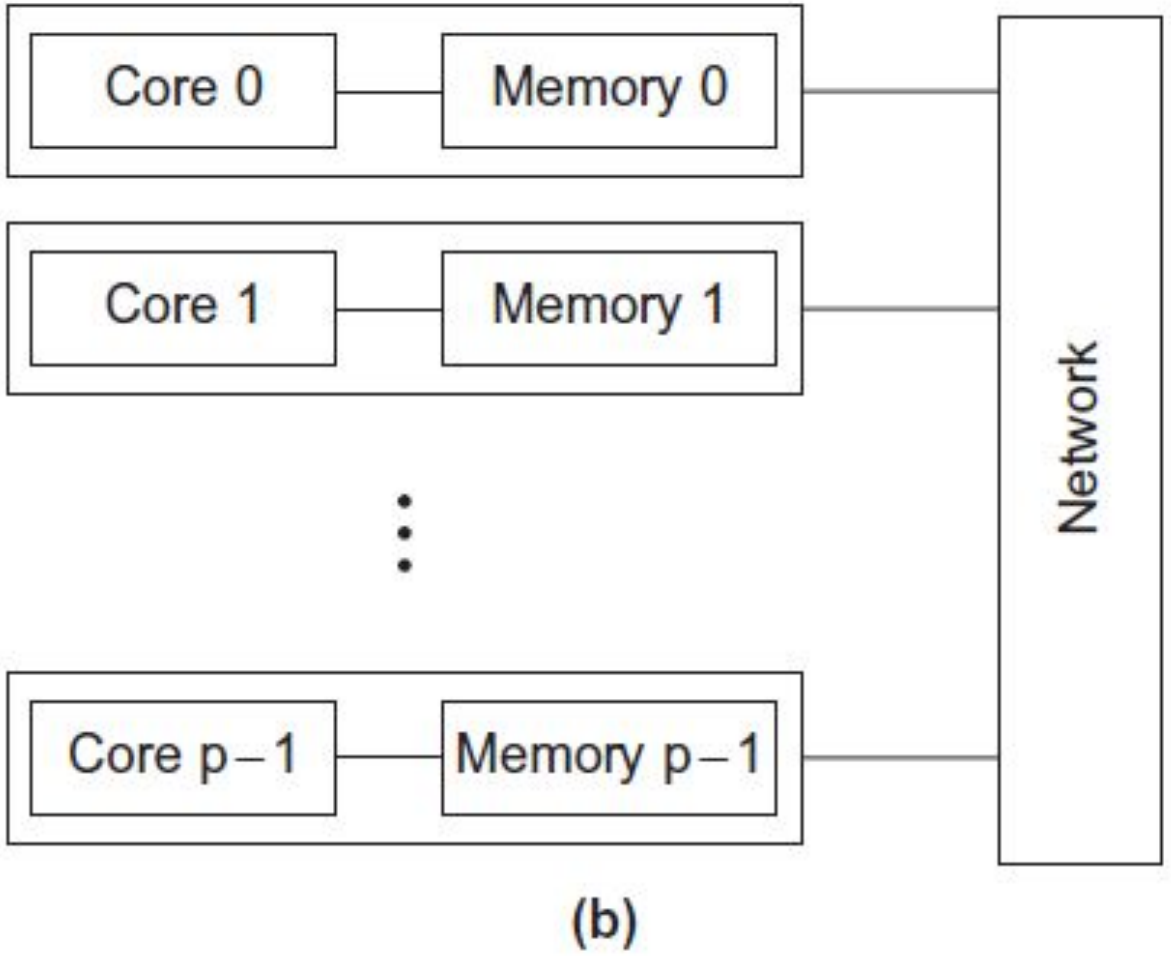
- Co-procesadores o aceleradores
- Problemas numéricos intensivos

Sistemas de Memoria Compartida



(a)

Sistemas de Memoria Distribuida



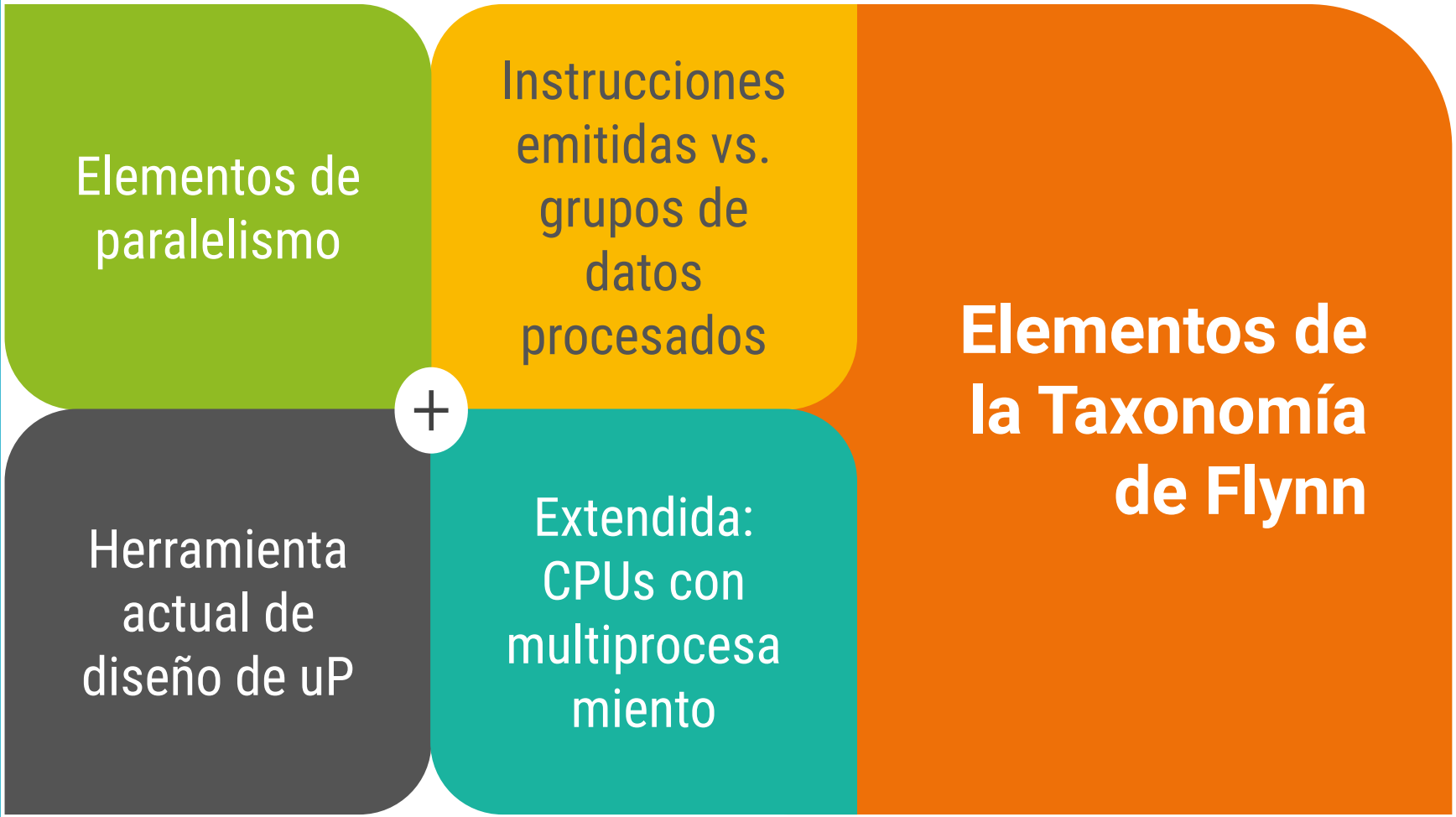
Clasificación de Sistemas

Taxonomía

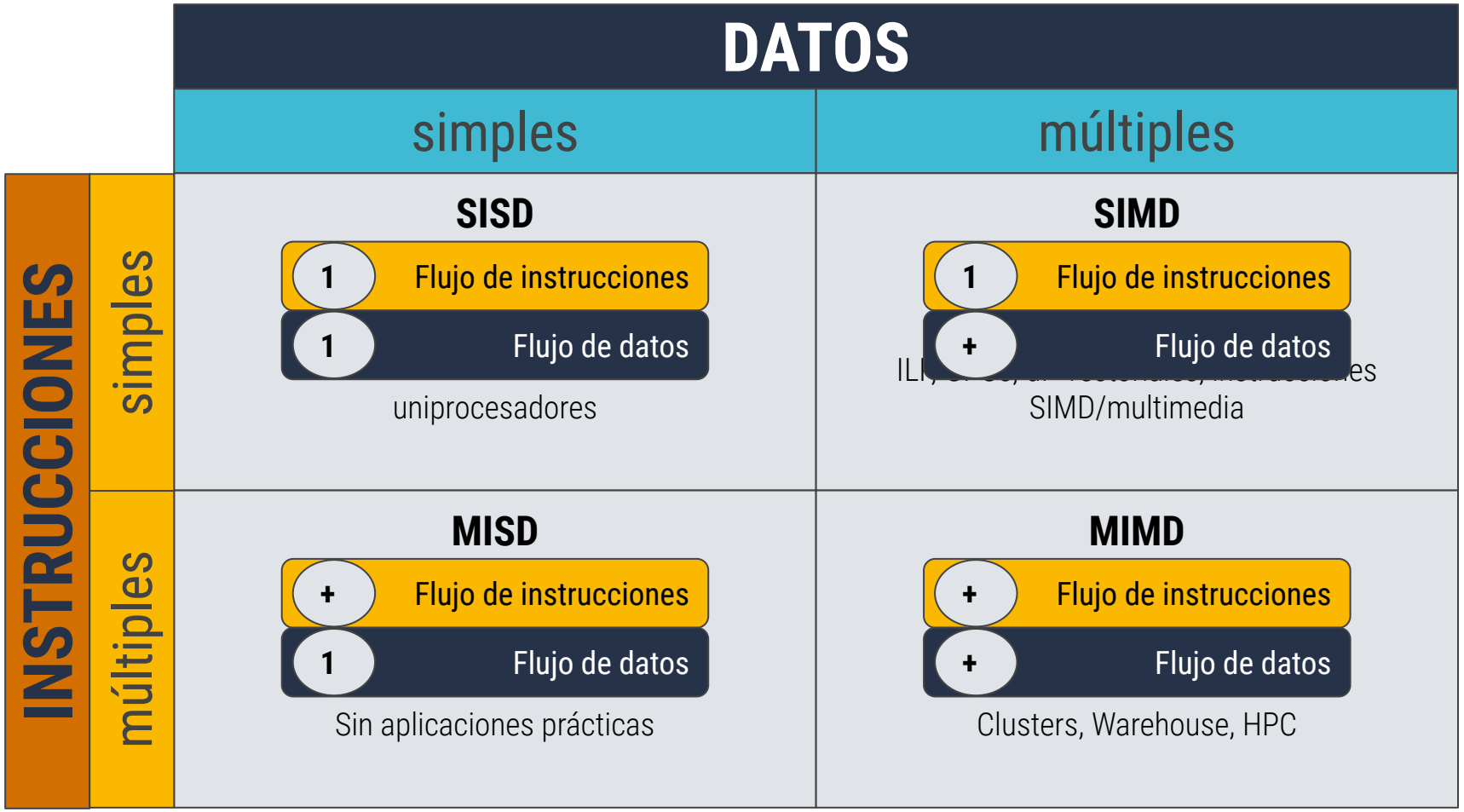
*(1) En equipos, defina qué es **taxonomía** y sus **características**. Piense en un ejemplo de taxonomía.*

(2) Reflexione individualmente las ventajas o desventajas de usar una taxonomía. Escriba una o dos ideas para compartir en clase

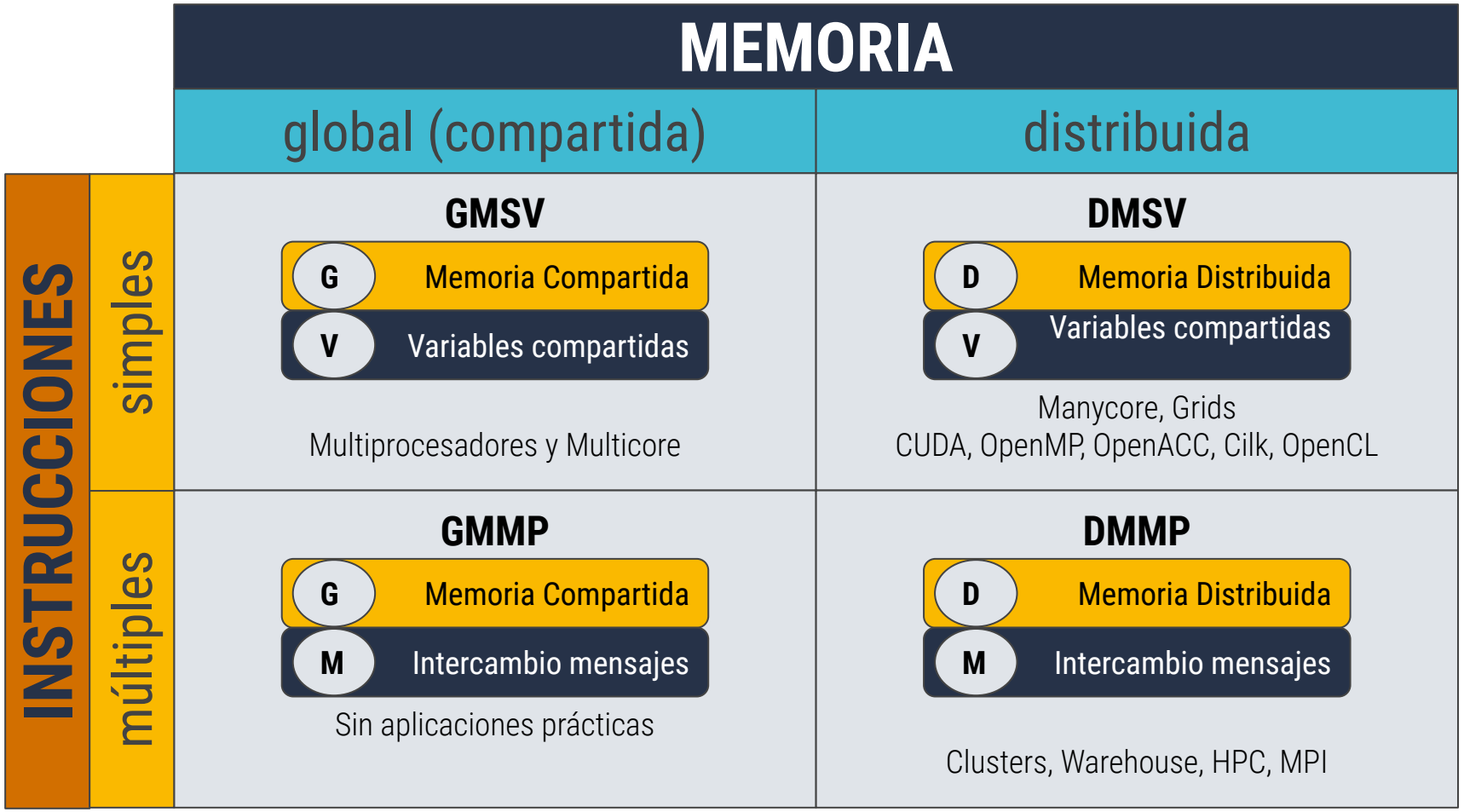
Elementos


Elementos de paralelismo
Instrucciones emitidas vs. grupos de datos procesados
+
Herramienta actual de diseño de uP
Extendida: CPUs con multiprocesa miento
Elementos de la Taxonomía de Flynn

Taxonomía de Flynn



Extensión de Johnson a la Taxonomía de Flynn



Modelos de Máquinas Paralelas

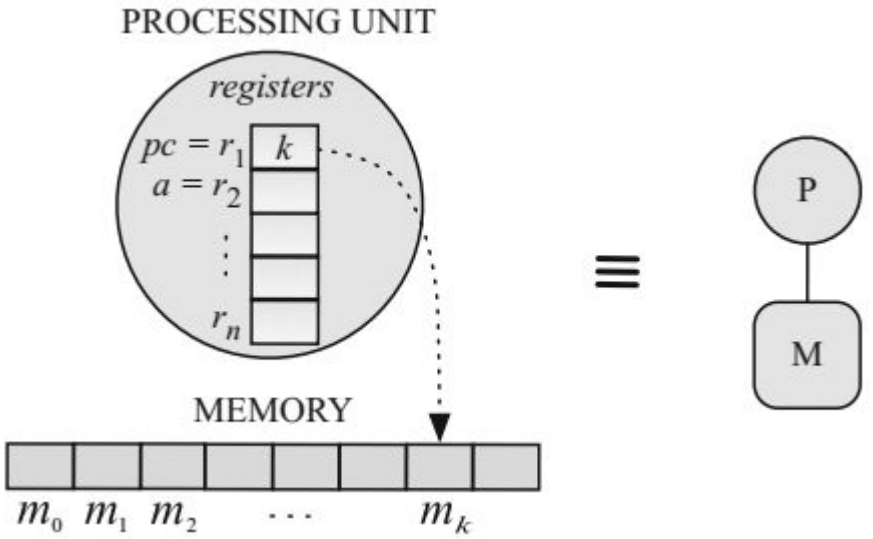
Máquinas Seriales de Acceso Aleatorio RAM

Random Access Machine

- Modelo secuencial
- Define la forma de acceso a la memoria
- Base para otros modelos paralelos

Máquinas Seriales de Acceso Aleatorio

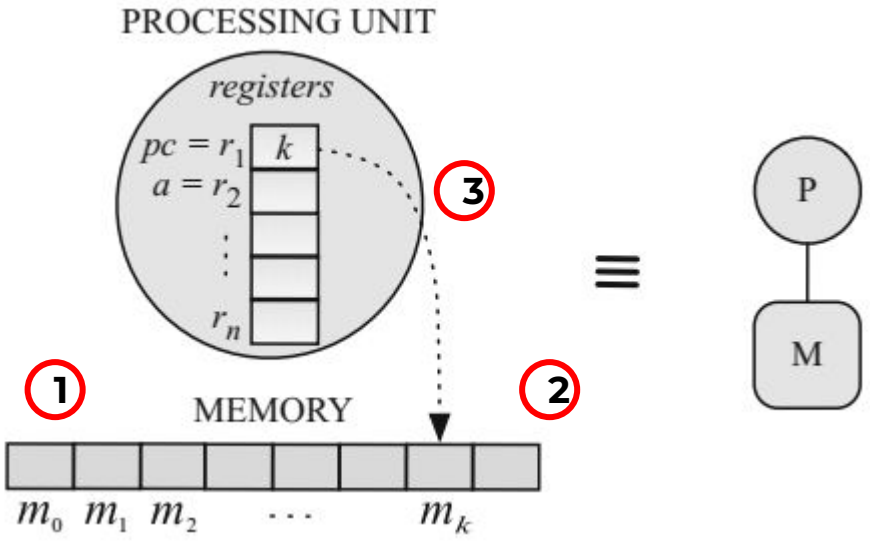
RAM



- CPU
- Memoria finita, en secuencia, acceso directo
  - Posición arbitraria m_i
  - R / W hecho en tiempo constante
- Registros acceso directo
  - PC, Acumulador A
- Programa finito (secuencia)

Máquinas Seriales de Acceso Aleatorio

RAM

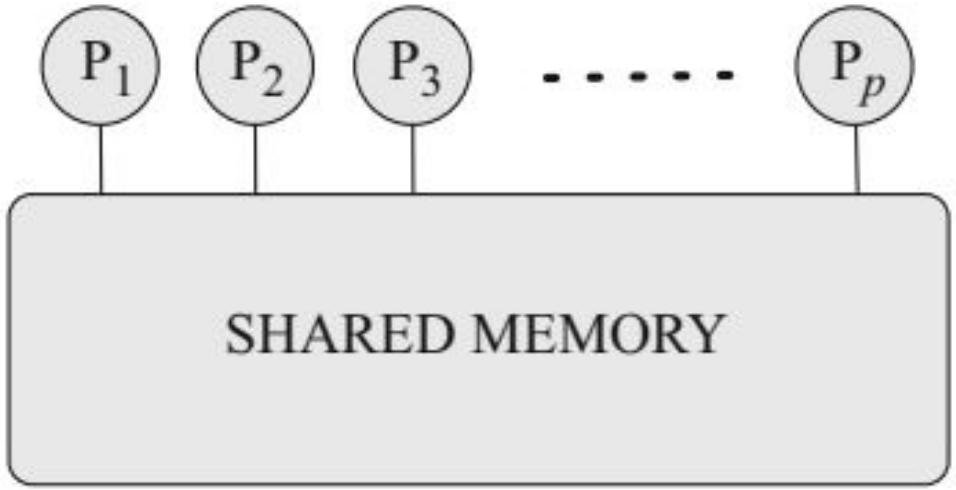


1. Programa en ubicaciones consecutivas desde m_0 ... m_i
2. Input (data) en ubicaciones disponibles
3. PC apunta a la ubicación de la instrucción en ejecución mk
4. PC se incrementa a la siguiente dirección mk+1 (o a una definida por un jump)

Máquinas Paralelas de Acceso Aleatorio

PRAM

Parallel Random Access Machine



Máquinas Paralelas de Acceso Aleatorio

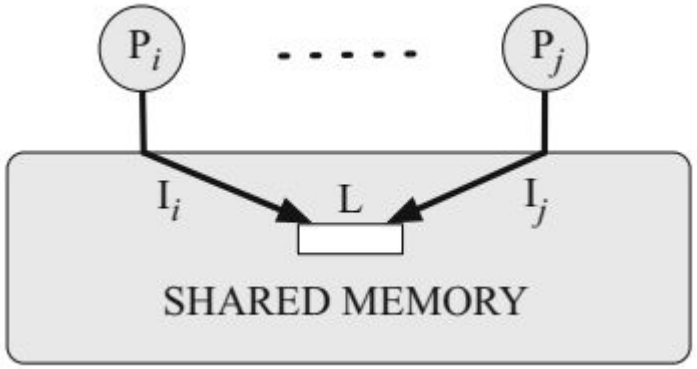
PRAM

Parallel Random Access Machine

- P unidades de procesamiento conectadas a memoria compartida (sin límite, finitas)
- La memoria no está vinculada (bounded) a ninguna UP
- Cada UP tiene acceso a cualquier ubicación de la memoria (word) en 1 solo paso.
- Cada UP puede solicitar acceso a memoria de forma directa.
- No hay red de interconexión.

Riesgos de las PRAM

Competencia
Secuenciación
Indeterminismo

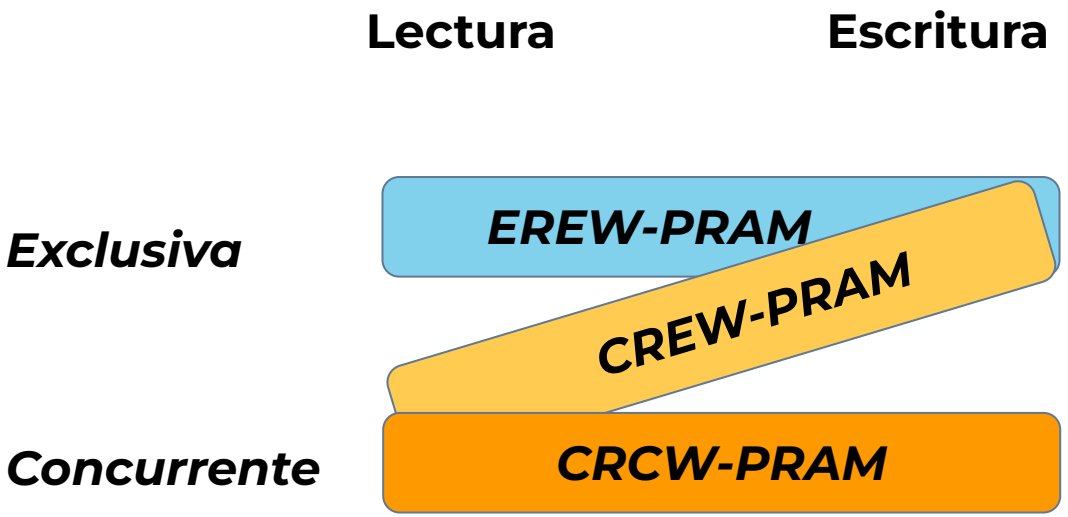


- Race condition – acceso simultáneo a (L)
- Indeterminismo – acceso simultáneo a (L)
- Secuenciación – HW serializa acceso a (L)
- No hay garantía en el orden
- No hay problema en lectura pero si en escritura

Variantes PRAM

Ajustar PRAM para:

- Permitir accesos simultáneos
- Evitar indeterminismo



EREW vs CREW

PRAM

EREW – PRAM

- Más realista de los modelos.
- No permite acceso simultáneo a memoria. Debe detener la ejecución si hay acceso simultáneo.
- Acceso exclusivo (mutex, locks)
- Responsabilidad del diseñador del algoritmo

CREW – PRAM

- Lecturas simultáneas de memoria
- Escrituras exclusivas
- Responsabilidad del diseñador del algoritmo.

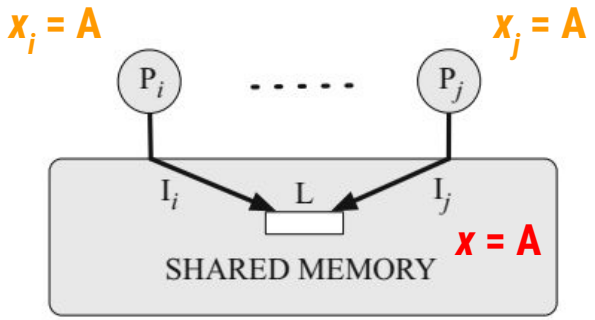
CRCW

PRAM

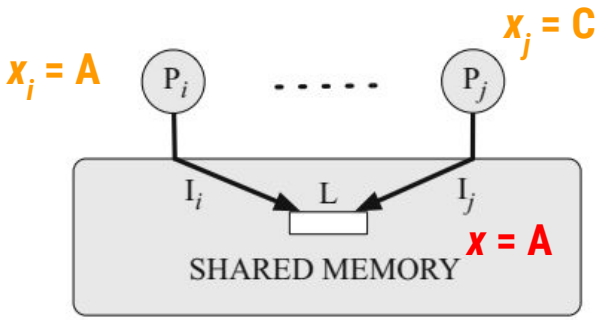
- Menos realista de los modelos.
- Lecturas y escrituras simultáneas.
- Restricciones para evitar indeterminismo.

CRCW PRAM

CONSISTENTE – UPs con escritura simultánea, pero el mismo valor.

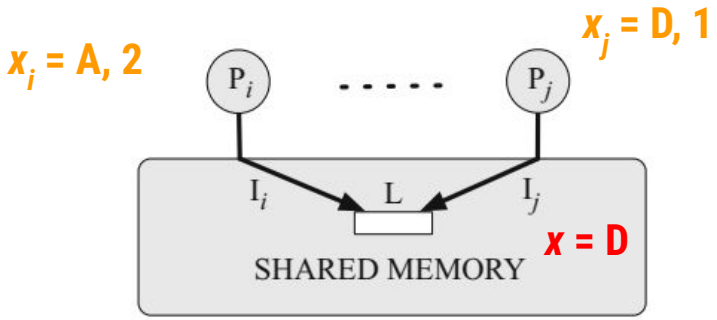


ARBITRARIA – UPs con escritura simultánea, diferente valor, se asume que solo una será exitosa.



CRCW PRAM

PRIORITARIA – Prioridad en el orden de acceso. Acceso simultáneo, solo el > prioridad será exitoso.

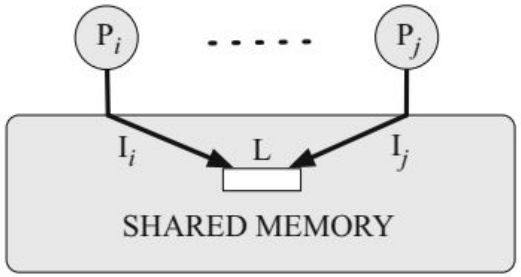


POR FUSIÓN (REDUX) – UPs con escritura simultánea, pero asumimos una operación de estas características:

- La misma operación aplicada en cada UP.
- La operación es asociativa y conmutativa.
- Solo se copia el resultado parcial a L.

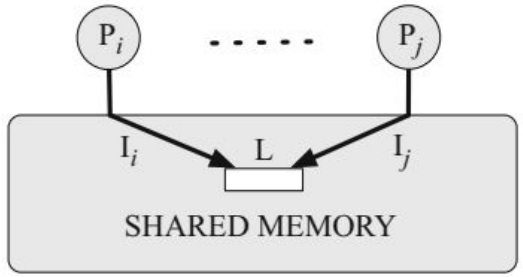
CRCW PRAM por fusión

P_i = max(a,b,c,d, …) P_j = max(j,k,l,m, …)



Suma, producto, max, min, OR, AND…

P_i = max$_i$ P_j = max$_j$



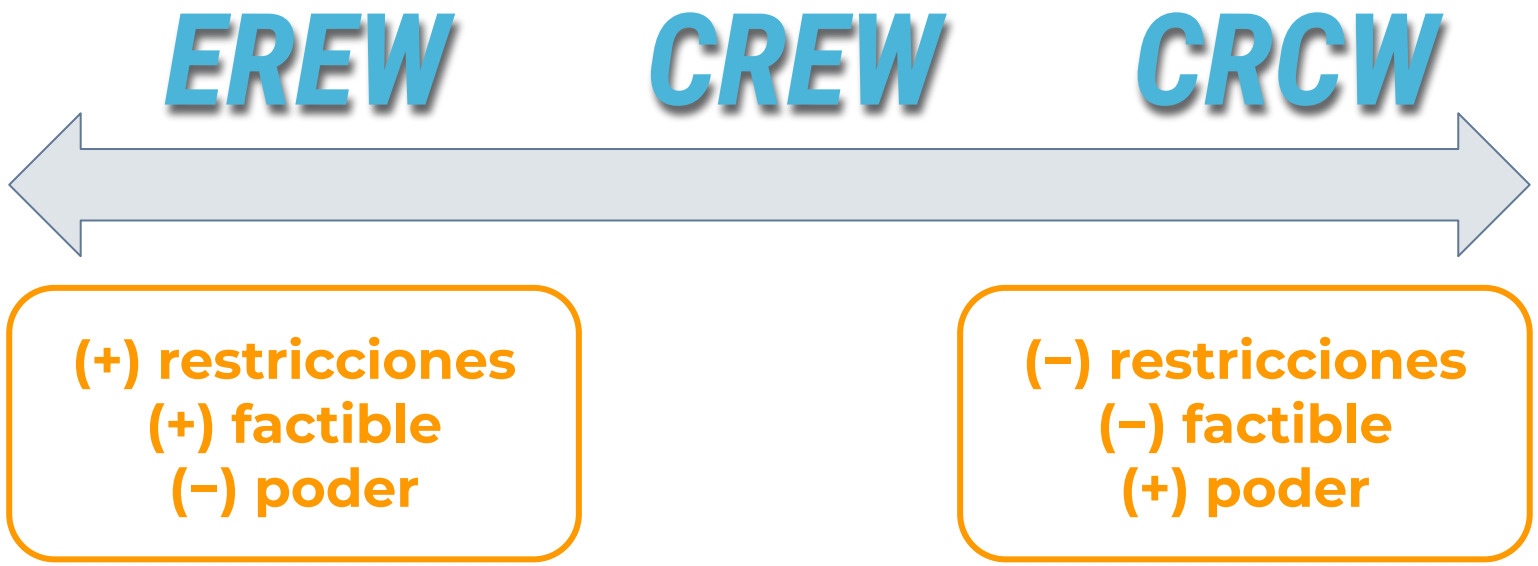
max(P_i ,P_j)

Compare los modelos PRAM

Clasifíquelos en orden según cada característica

EREW **CREW** **CRCW**

Comparación de modelos PRAM



Relevancia de modelos PRAM

El modelo PRAM no es apropiado para memoria vinculada de manera práctica, pero sirve para identificar los aspectos paralelizables de un problema.

1. Diseño de un programa que resuelva un problema con el modelo CRCW (+ potente, – realista, + sencillo de diseñar)
2. Ejecutar el programa bajo el modelo EREW
3. Realizar ajustes necesarios para garantizar paralelismo. El problema será resuelto en tiempo O(log p) que el ideal CRCW.

REFERENCIAS

- **PACHECO, Peter**. (2011). "Chapter 2 – Parallel Hardware and Parallel Software". An Introduction to Parallel Programming. [PDF]. Morgan Kaufmann.
- **TROBEC, Roman. SLIVNIK, Bostjan**. **BULIC, Patricio. ROBIC, Borut.** (2018). "Chapter 2 – Parallel Computer Architecture". Introduction to Parallel Computing. [PDF] Springer.
- **RAUBER, Thomas. RÜNGER, Gudula**. (2010). "2 – Overview of Parallel Systems". Parallel Programming – For Multicore and Cluster Systems. [PDF] Springer.