Ciencia y Tecnología
Secretaría de Ciencia, Humanidades, Tecnología e Innovación

CIMAT
UNIDAD MÉRIDA

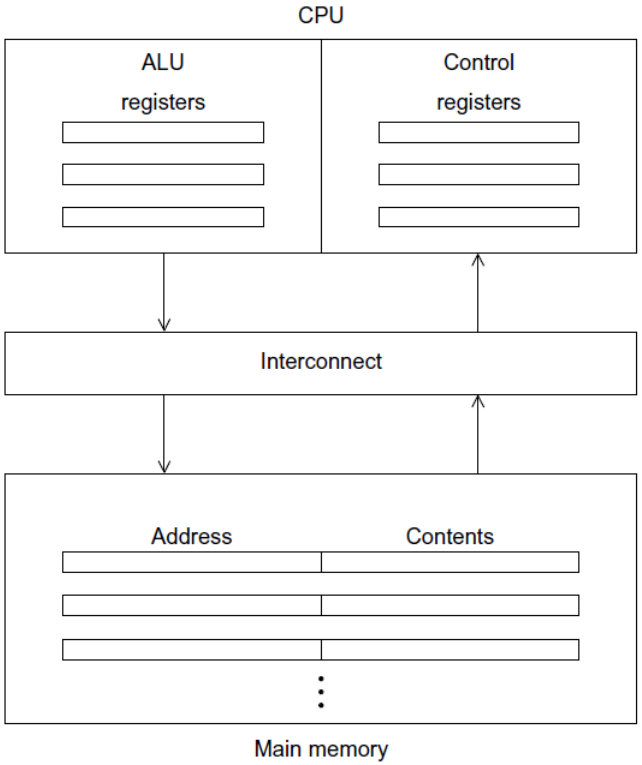
# CÓMPUTO PARALELO (INTRODUCCIÓN)

Dr. Francisco J. Hernández López
SECIHTI – CIMAT-Mérida
fcoj23@cimat.mx, www.cimat.mx/~fcoj23

# ARQUITECTURA VON NEUMANN (1945)

- Básicamente, contiene:
  - Memoria principal
  - Unidad de Procesamiento Central (CPU)
  - Interconexión entre la memoria y la CPU

John Von Neumann




*Pacheco, P. (2011). An introduction to parallel programming. Elsevier.*

# MEMORIA RAM

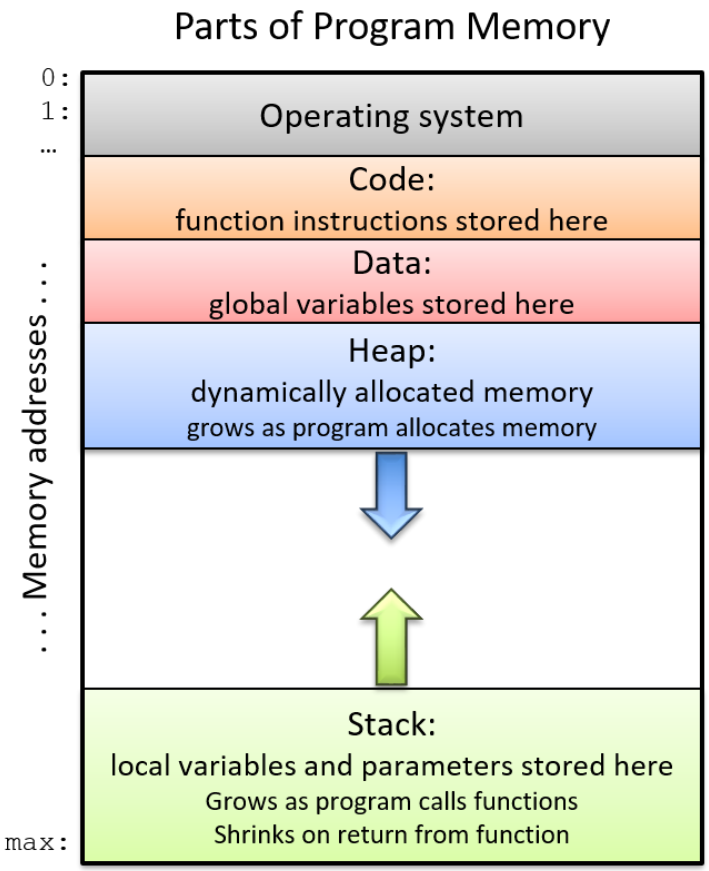
https://www.muycomputer.com/2018/11/04/memoria-ram-que-es-recomendaciones/

- Memoria principal o Random Access Memory (RAM)
  - DRAM: Dynamic RAM, son dispositivos basados en carga, en donde cada bit está representado por una carga eléctrica almacenada en un condensador (capacitor) muy pequeño. Dicha carga puede perderse en poco tiempo, por lo que es necesario que el sistema se actualice continuamente para evitar perder los datos. Ofrece: menor costo, menos espacio en la placa madre, menos energía y menos calor
  - SRAM: Static RAM, están basados en compuertas, cada bit se almacena en cuatro a seis transistores conectados. Retienen los datos mientras tengan energía, sin necesidad de ningún tipo de actualización de datos. Mayor costo, espacio en la placa madre, energía y calor

Charles Severance and Kevin Dowd. High Performance Computing. Rice University, Houston, Texas. 2012. *Available online:* http://cnx.org/content/col11136/1.5/

# PARTES DE LA MEMORIA

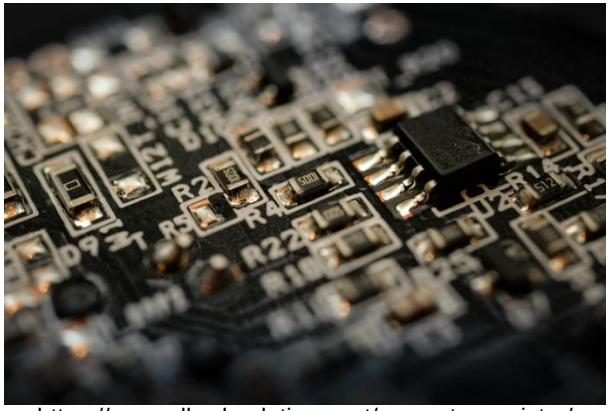
- **Pila (stack):** Para variables locales. La pila aumenta en la llamada al procedimiento y disminuye en el retorno. Son escalares en lugar de arreglos o matrices
- **Datos globales:** Para objetos declarados estáticamente como varibles globales o constantes. Pueden ser arreglos u otras estructuras de datos
- **Motículo (heap):** Para objetos dinámicos. Se acceden mediante apuntadores y típicamente no son escalares




https://diveintosystems.org/book/C2-C_depth/scope_memory.html

John L. Hennessy and David A. Patterson. 2017. Computer Architecture, Sixth Edition: A Quantitative Approach (6th. ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

# REGISTROS



https://www.allwebsolutions.net/computer-register/

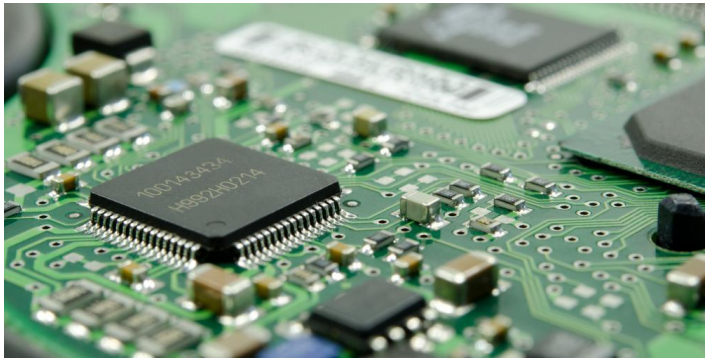
- Son un conjunto de espacios de almacenamiento temporal que se encuentran en el procesador, ocupando la capa superior de la jerarquía de la memoria
- Son lo que el procesador opera, por ej., en una operación como:

$$y = m * x + b$$

  - $m, x$ y $b$ se almacenan en registros
  - se realiza la multiplicación $m * x$ y el resultado se almacena en otro registro
  - luego, se realiza la suma $m * x + b$ y el resultado se almacena en otro registro
  - finalmente, el resultado va del registro a la posición de memoria $y$

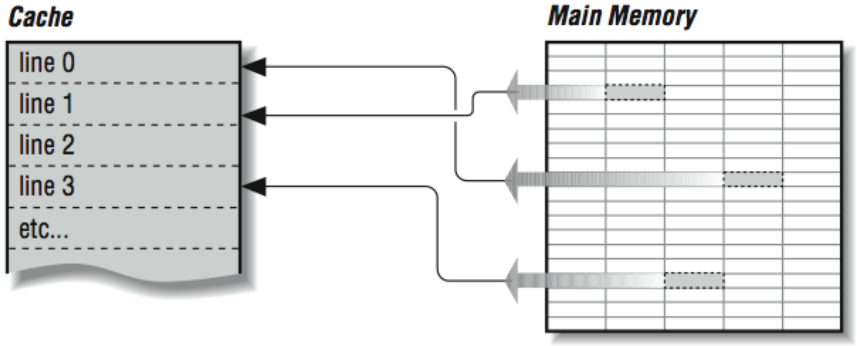
Charles Severance and Kevin Dowd. High Performance Computing. Rice University, Houston, Texas. 2012. *Available online:* http://cnx.org/content/col11136/1.5/

# CACHÉ



https://www.techyv.com/article/cache-memory-types-and-importance/

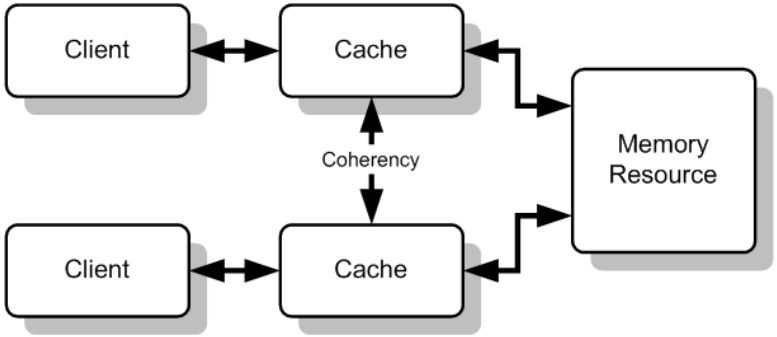
- Entre los registros y la memoria principal hay varios niveles de memoria caché que tienen menor latencia y mayor ancho de banda que la memoria principal y en donde los datos se conservan durante un periodo de tiempo intermedio
- Si un elemento de los datos se reutiliza poco después de que se necesitó por primera vez, este seguirá estando en la caché y por lo tanto se podrá acceder a él mucho más rápido que si se tuviera que traer desde la memoria principal




Charles Severance and Kevin Dowd. High Performance Computing. Rice University, Houston, Texas. 2012. *Available online:* http://cnx.org/content/col11136/1.5/

# COHERENCIA ENTRE LOS CACHÉS

- En multiprocesadores, las variables que se modifican deben devolverse a la memoria principal para que el resto de los procesadores puedan obtener el valor correcto de cada una de ellas
- Los procesadores deben ser conscientes de la actividad de la caché local. Quizá deban invalidar las líneas antiguas que contienen el valor anterior de la variable modificada para no utilizar datos obsoletos




Protocolos de coherencia:

- MSI (Modified-Shared-Invalid)
- MESI (M-Exclusive-S-I)
- MOSI (M-Owned-S-I)
- MOESI

https://es.wikipedia.org/wiki/Coherencia_de_caché

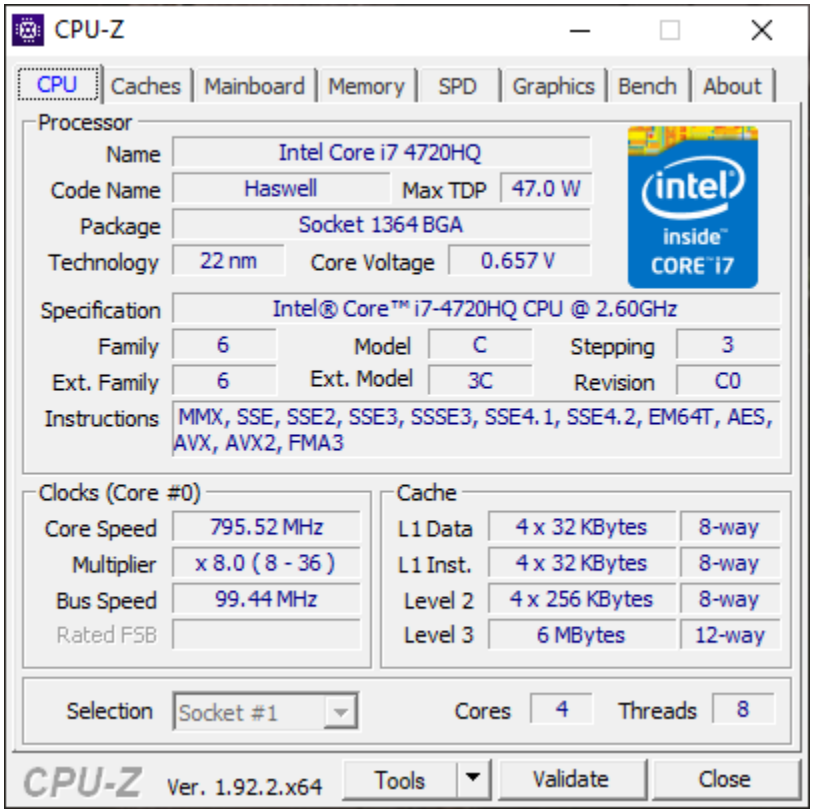
Charles Severance and Kevin Dowd. High Performance Computing. Rice University, Houston, Texas. 2012. *Available online:* http://cnx.org/content/col11136/1.5/

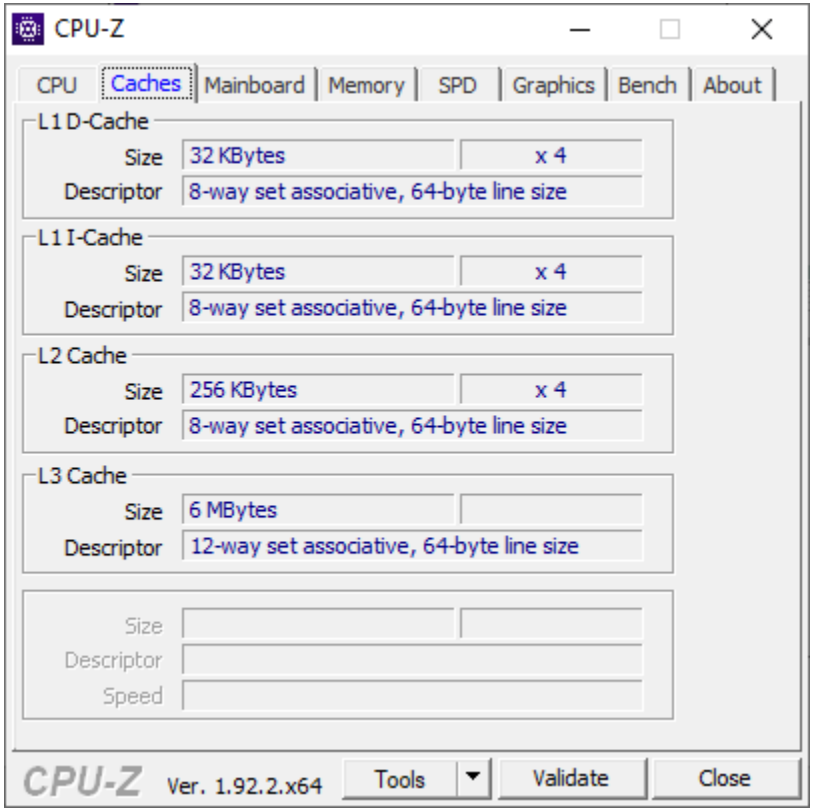
| Memory | Size | Latency | Bandwidth |
|---|---|---|---|
| L1 cache | 32 KB | 1 nanosecond | 1 TB/second |
| L2 cache | 256 KB | 4 nanoseconds | 1 TB/second<br>Sometimes shared by two cores |
| L3 cache | 8 MB or more | 10x slower than L2 | >400 GB/second |
| MCDRAM | | 2x slower than L3 | 400 GB/second |
| Main memory on DDR DIMMs | 4 GB-1 TB | Similar to MCDRAM | 100 GB/second |
| Main memory on Cornelis* Omni-Path Fabric | Limited only by cost | Depends on distance | Depends on distance and hardware |
| I/O devices on memory bus | 6 TB | 100x-1000x slower than memory | 25 GB/second |
| I/O devices on PCIe bus | Limited only by cost | From less than milliseconds to minutes | GB-TB/hour Depends on distance and hardware |

Datos aprox. para un sistema con un procesador Intel de 16 núcleos en el 2016.

https://www.intel.com/content/www/us/en/developer/articles/technical/memory-performance-in-a-nutshell.html

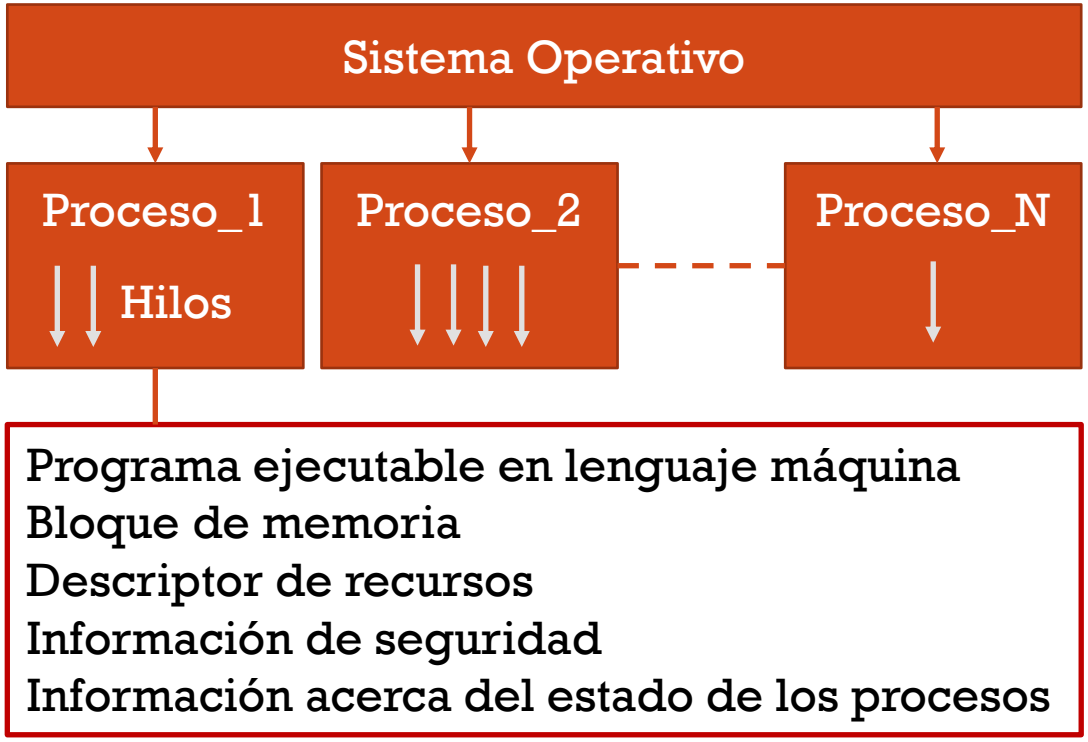
# CPU-Z


CPU-Z
CPU
Caches
Mainboard
Memory
SPD
Graphics
Bench
About
Processor
Name
Intel Core i7 4720HQ
Code Name
Haswell
Max TDP
47.0 W
Package
Socket 1364 BGA
Technology
22 nm
Core Voltage
0.657 V
Specification
Intel® Core™ i7-4720HQ CPU @ 2.60GHz
Family
6
Model
C
Stepping
3
Ext. Family
6
Ext. Model
3C
Revision
C0
Instructions
MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, EM64T, AES, AVX, AVX2, FMA3
Clocks (Core #0)
Core Speed
795.52 MHz
Multiplier
x 8.0 ( 8 - 36 )
Bus Speed
99.44 MHz
Rated FSB
Cache
L1 Data
4 x 32 KBytes
8-way
L1 Inst.
4 x 32 KBytes
8-way
Level 2
4 x 256 KBytes
8-way
Level 3
6 MBytes
12-way
Selection
Socket #1
Cores
4
Threads
8
CPU-Z
Ver. 1.92.2.x64
Tools
Validate
Close


CPU-Z
CPU
Caches
Mainboard
Memory
SPD
Graphics
Bench
About
L1 D-Cache
Size
32 KBytes
x 4
Descriptor
8-way set associative, 64-byte line size
L1 I-Cache
Size
32 KBytes
x 4
Descriptor
8-way set associative, 64-byte line size
L2 Cache
Size
256 KBytes
x 4
Descriptor
8-way set associative, 64-byte line size
L3 Cache
Size
6 MBytes
Descriptor
12-way set associative, 64-byte line size
Size
Descriptor
Speed
CPU-Z
Ver. 1.92.2.x64
Tools
Validate
Close

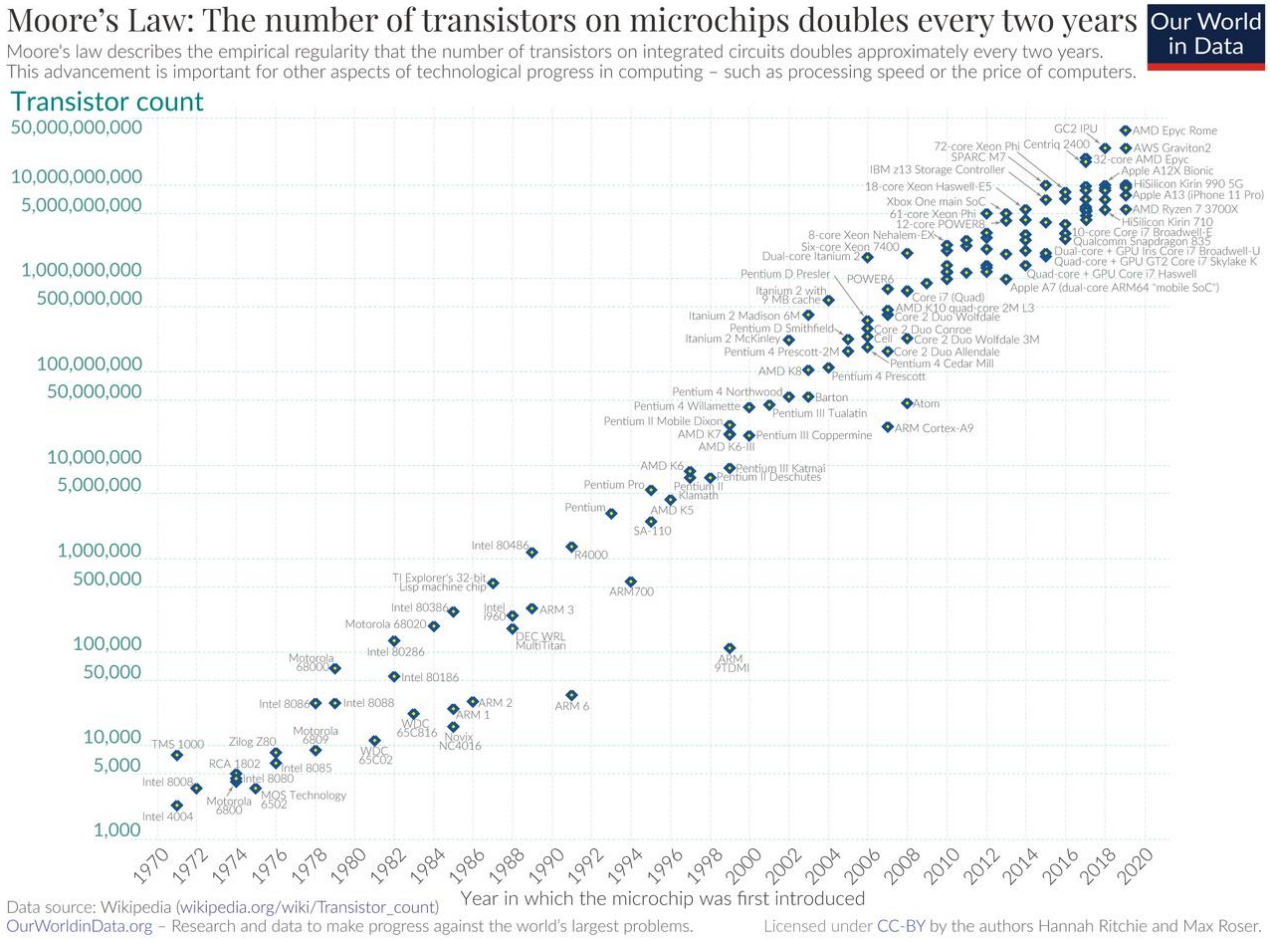
# HILOS Y PROCESOS

- Proceso: Es un programa en ejecución (Proceso pesado)
- Hilo: Es una unidad básica que es parte de un proceso (Proceso ligero)



# LEY DE MOORE (GORDON E. MOORE, 1965)

- La cantidad de transistores en un chip se duplica aprox. cada dos años



MOSFET scaling (process nodes)

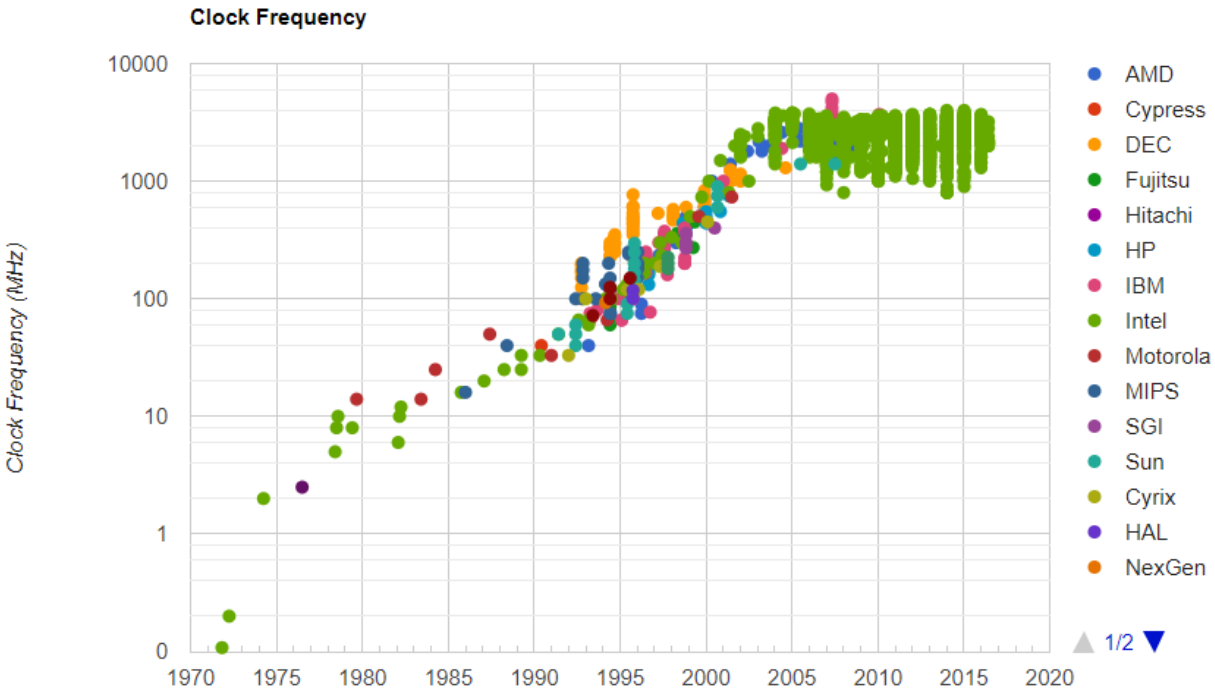
10 µm – 1971
6 µm – 1974
3 µm – 1977
1.5 µm – 1981
1 µm – 1984
800 nm – 1987
600 nm – 1990
350 nm – 1993
250 nm – 1996
180 nm – 1999
130 nm – 2001
90 nm – 2003
65 nm – 2005
45 nm – 2007
32 nm – 2009
22 nm – 2012
14 nm – 2014
10 nm – 2016
7 nm – 2018
5 nm – 2020

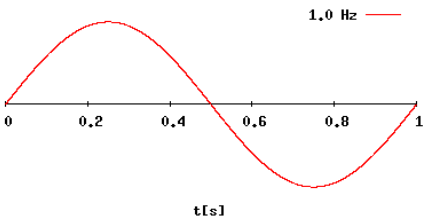
Future
3 nm – ~2022
2 nm – >2023

https://www.intel.com/content/www/us/en/history/history-moores-law-fun-facts-factsheet.html

# VELOCIDAD DE RELOJ (CLOCK RATE)

- Cantidad de ciclos que ejecuta la CPU por segundo, medida en MHz o GHz




3.2 GHz significa 3.2 billones de ciclos por segundo

http://cpudb.stanford.edu/visualize/clock_frequency

# ¿QUÉ ES EL CÓMPUTO PARALELO (CP)?

- Ejecución de más de un cómputo (cálculo) al mismo tiempo o "en paralelo", utilizando más de un procesador.

Sistema de Cómputo Paralelo

Hardware

Software

*Parallel programming: For multicore and cluster systems.* Rauber, Thomas, and Gudula Rünger. Springer Science & Business Media, 2013.

# MODELOS DE PROGRAMACIÓN EN PARALELO

- Modelo máquina
  - Es el nivel más bajo de abstracción.
  - Consiste en una descripción de HW y el SO. Ejemplo: Los registros y la entrada/salida de los buffers.
  - El lenguaje ensamblador esta basado en este nivel de modelos.

- Modelo de arquitectura
  - Interconexión de red de plataformas paralelas.
  - Organización de la memoria
  - Procesamiento síncrono o asíncrono.
  - Modelo de ejecución SIMD (Simple Intruction Multiple Data) o MIMD (Multiple Instruction Multiple Data)

*Parallel programming: For multicore and cluster systems.* Rauber, Thomas, and Gudula Rünger. Springer Science & Business Media, 2013.

# MODELOS DE PROGRAMACIÓN EN PARALELO (C1)

- Modelo computacional
  - Provee un método analítico para diseñar y evaluar algoritmos
  - La complejidad de un algoritmo se debe ver reflejado en el rendimiento en una computadora

Random Access Machine (RAM)

Parallel Random Access Machine (PRAM)

*Parallel programming: For multicore and cluster systems.* Rauber, Thomas, and Gudula Rünger. Springer Science & Business Media, 2013.

# MODELOS DE PROGRAMACIÓN EN PARALELO (C2)

- Modelo de programación
  - Describe un sistema de cómputo paralelo en términos de la semántica del lenguaje de programación o el ambiente de programación.
  - Especifica al programador como codificar un algoritmo paralelo.
  - Influenciado por: La arquitectura, el compilador, librerías, etc.
  - Criterios por los cuales pueden ser diferentes:
    - El nivel de paralelismo.
    - Especificaciones explicitas definidas por el usuario.
    - El modo de ejecución: SIMD, MIMD, Síncrono o Asíncrono.
    - Modos y patrones de comunicación para el intercambio de información.
    - Mecanismos de sincronización.

*Parallel programming: For multicore and cluster systems.* Rauber, Thomas, and Gudula Rünger. Springer Science & Business Media, 2013.

# PARALELIZACIÓN DE PROGRAMAS

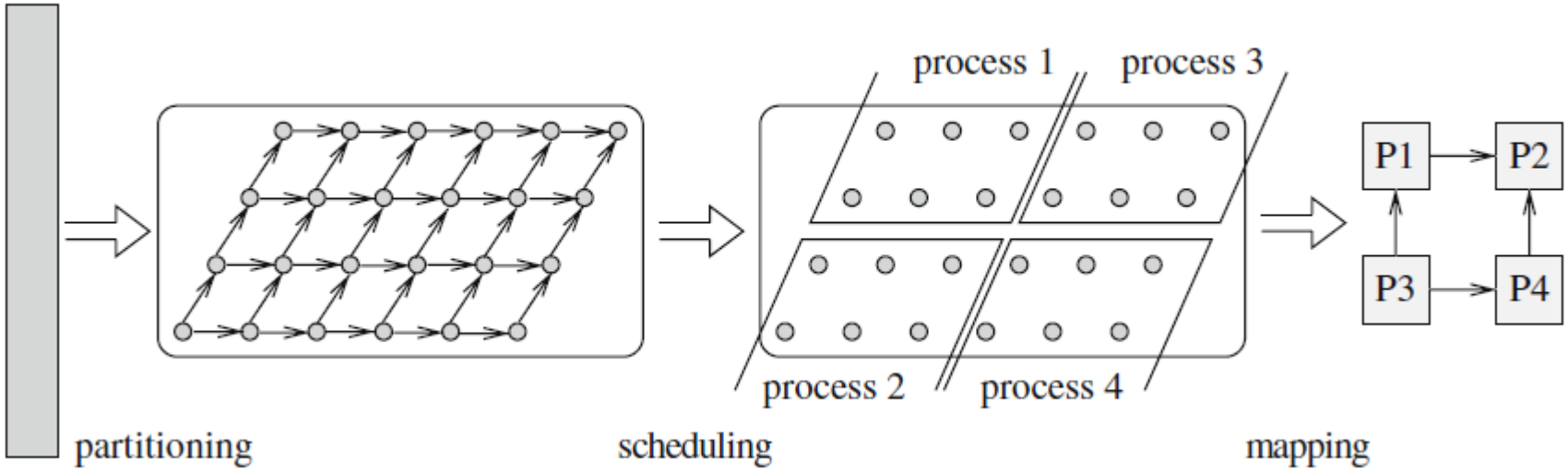
- Se asume que el cómputo que se pretende paralelizar esta dado a partir de un algoritmo o programa secuencial.
- Se debe considerar el control y dependencia de los datos.
- Asegurar que el programa paralelo produce los mismos resultados que el programa secuencial.

Objetivo: Reducir el tiempo de ejecución de un programa tanto como sea posible, usando múltiples procesadores o cores.

*Parallel programming: For multicore and cluster systems.* Rauber, Thomas, and Gudula Rünger. Springer Science & Business Media, 2013.

# POSIBLES PASOS PARA LA PARALELIZACIÓN

1. Descomposición de los cálculos computacionales
2. Asignación de tareas a procesar o hilos
3. Mapeo de procesos o hilos a procesadores o cores físicos




*Parallel programming: For multicore and cluster systems.* Rauber, Thomas, and Gudula Rünger. Springer Science & Business Media, 2013.

# DESCOMPOSICIÓN DE LOS CÁLCULOS COMPUTACIONALES

- Aquí, los cálculos del algoritmo secuencial se descomponen en tareas y se determinan las dependencias entre la tareas.
- Dependiendo del modelo de la memoria, una tarea puede involucrar accesos a espacios de *direcciones compartidas* o puede ejecutar operaciones de *paso de mensajes*.
- Dependiendo de la aplicación, la descomposición de tareas se puede llevar a cabo:
  - En la fase de Inicialización → Descomposición Estática
  - Durante la ejecución del Programa → Descomposición Dinámica

Objetivo: Generar suficientes tareas para mantener a todos los cores ocupados todo el tiempo.

*Parallel programming: For multicore and cluster systems.* Rauber, Thomas, and Gudula Rünger. Springer Science & Business Media, 2013.

# ASIGNACIÓN DE TAREAS A PROCESAR O HILOS

- Un procesador o hilo:
  - Representa un flujo de control ejecutado por un procesador físico.
  - Puede ejecutar diferentes tareas, una por una.
- El número de hilos no necesariamente va ser el mismo que el número de cores físicos en un problema, aunque se suele utilizar de esta manera.
- La asignación de tareas a hilos también se le conoce como "Scheduling", puede haber planificación estática o dinámica.

Objetivo: Asignar las tareas tal que se obtenga un buen "Balance de Carga".

*Parallel programming: For multicore and cluster systems.* Rauber, Thomas, and Gudula Rünger. Springer Science & Business Media, 2013.

# MAPEO DE HILOS A CORES FÍSICOS

- El caso más simple es cuando cada proceso o hilo es mapeado a un solo core.
- Si ∃ menos cores que hilos, entonces múltiples hilos deben ser mapeados a un solo core. Este mapeo puede ser hecho por:
  - El Sistema Operativo (SO).
  - Soportado por sentencias del programa.

Objetivo: Tener a todos los cores trabajando, mientras que la comunicación entre ellos sea mínima.

*Parallel programming: For multicore and cluster systems.* Rauber, Thomas, and Gudula Rünger. Springer Science & Business Media, 2013.

# NIVELES DEL PARALELISMO

- **A nivel de bits**
- **A nivel de instrucción**
- **A nivel de datos**
- **A nivel de tareas**

# A NIVEL DE BITS

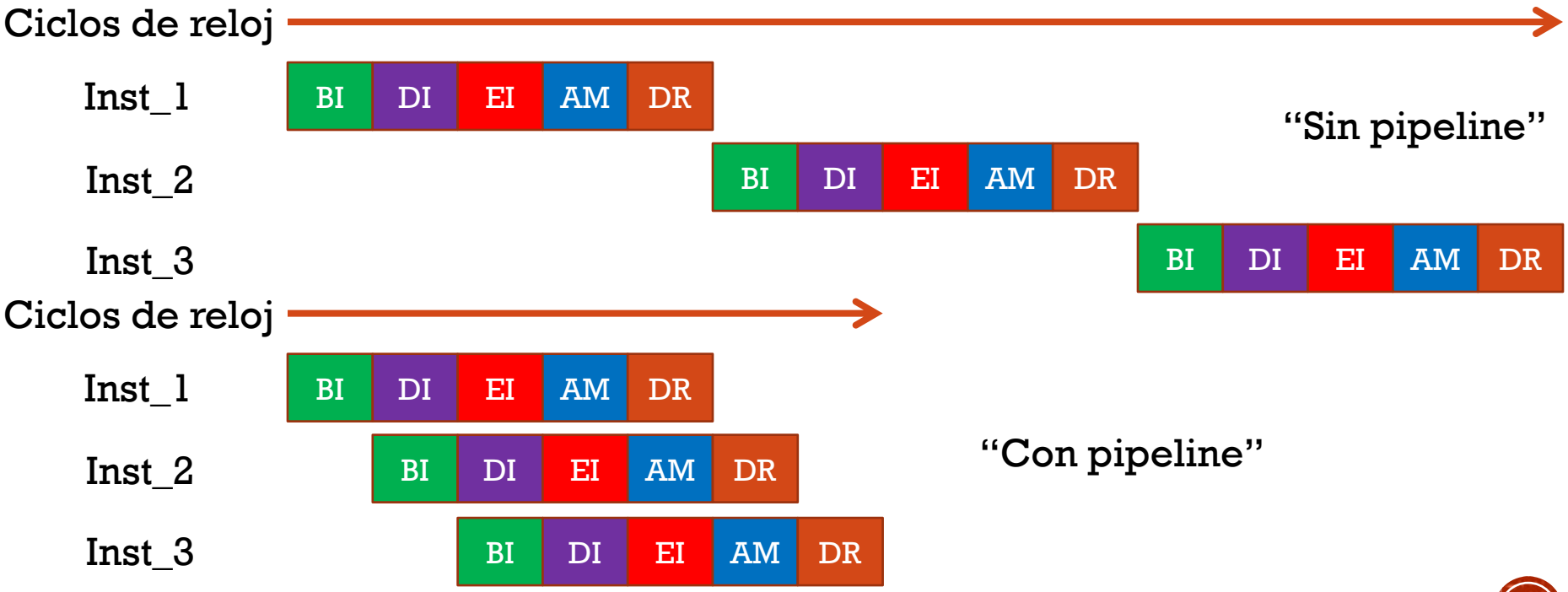
- Basado en incrementar el número de bits de una palabra (*word length, word size* o *word width*)
- Importante característica de los procesadores (8,16, 32 o 64 bits)
- Ejemplo:
  - Tenemos un procesador de 16 bits y queremos sumar dos enteros (**int**) de 32 bits
  - Primero el procesador realiza la suma de los primeros 16 bits y después de los últimos 16 bits, completando la suma de los **int** en dos instrucciones
  - Un procesador mayor o igual a 32 bits realizaría la operación en una sola instrucción.

*https://en.wikipedia.org/wiki/Parallel_computing*

# A NIVEL DE INSTRUCCIÓN

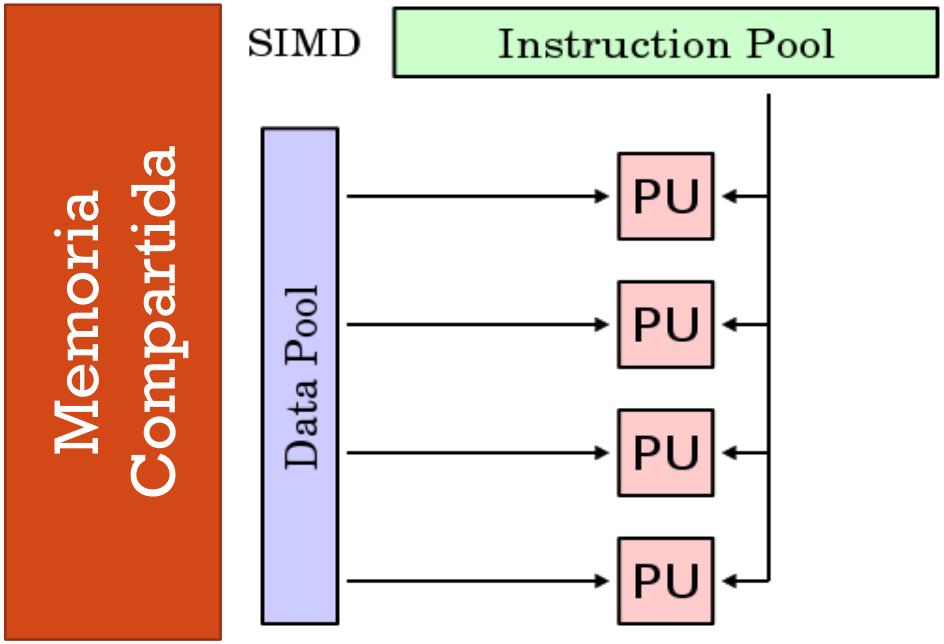
BI → Buscar Instrucción
DI → Descifrar Instrucción
EI → Ejecutar Instrucción
AM → Acceso a Memoria
DR → Dar Respuesta

- Capacidad de traslapar instrucciones
- Depende del pipeline del procesador



# A NIVEL DE DATOS

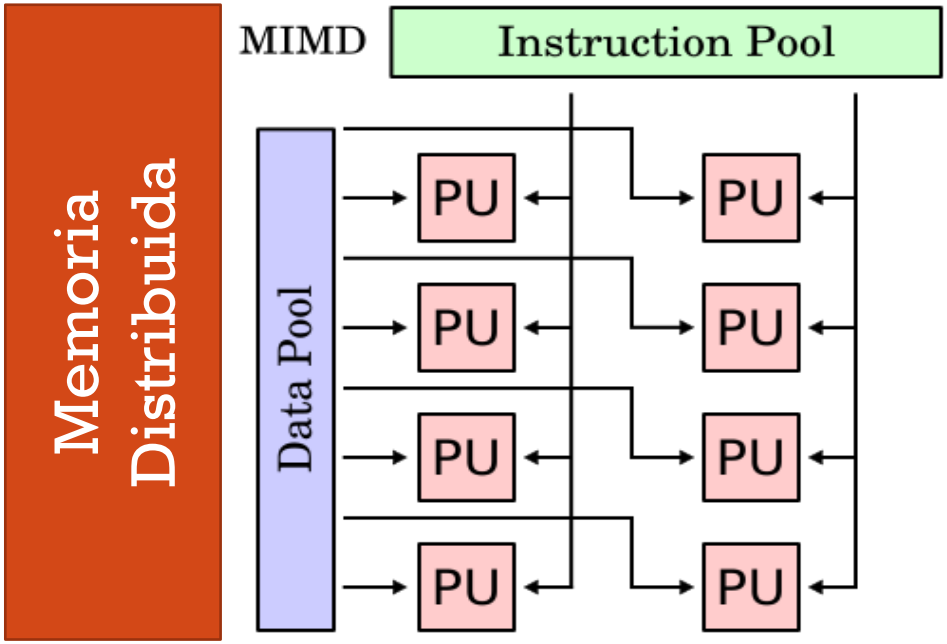
- Cada procesador realiza la misma tarea en diferentes partes de los datos (SIMD → Single Instruction Multiple Data)




SIMD [wikipedia], Taxonomía de Flynn

# A NIVEL DE TAREAS

- Diferentes procesadores pueden estar ejecutando diferentes instrucciones en diferentes partes de los datos (MIMD)




MIMD [wikipedia], Taxonomía de Flynn

# ARQUITECTURAS PARA EL CP

Computadora multi-core

Cluster

GPUs

FPGAs

Procesadores vectoriales

Xeon Phi Coprocessor

# LIBRERÍAS DE PROGRAMACIÓN PARA EL CP

**OpenMP** (Memoria Compartida)

**OpenMPI** (Memoria Distribuida)

**CUDA** (***C****ompute* ***U****nified* ***D****evice* ***A****rchitecture*) para GPUs.

**OpenCL (Open C**omputing **L**anguage**)** para GPUs y CPUs.

## Mezcla de Arquitecturas:

Computadora (o múltiples cores) y un GPU (o un arreglo de GPUs) .

Cluster con computadoras que contienen múltiples cores y a su vez cada maquina tiene un GPU (o un arreglo de GPUs).

Así como mezclamos las arquitecturas también mezclamos las librerías:

- CUDA con (OpenMP, OpenMPI).
- OpenCL con (OpenMP, OpenMPI).

# GRACIAS POR SU ATENCIÓN

Francisco J. Hernández-López

fcoj23@cimat.mx

WebPage:

www.cimat.mx/~fcoj23