UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO

FACULTAD
DE INGENIERÍA

Licenciatura en Ciencias de la Computación

Arquitectura de Computadoras II

Unidad 2

Arquitecturas Paralelas

Dr. Ing. Pablo Daniel Godoy (pablo.godoy@ingenieria.uncuyo.edu.ar)

Temas

- **Tipos de sistemas paralelos (Clasificación de Flynn)**
- Paralelismo a nivel de instrucción
- Paralelismo a nivel de hilos (Procesadores multihilo simultáneo)
- Paralelismo a nivel de núcleos
  - Arquitecturas multi núcleo simétricas
  - Arquitecturas multi núcleo heterogéneas con igual ISA o con ISA diferentes
  - Sistemas CPU-GPU, CPU-DSP. GPGPU. Introducción a CUDA y OpenCL
  - Sistemas operativos para multiprocesadores.
- Paralelismo a nivel de computadoras (Cluster)
- Paralelismo a nivel de datos (SIMD)
- Sistemas RAID (arreglos de discos duros)
- Performance y escalabilidad
  - Speedup y eficiencia
  - Modelos de problemas paralelizables
  - Paralelismo en el software

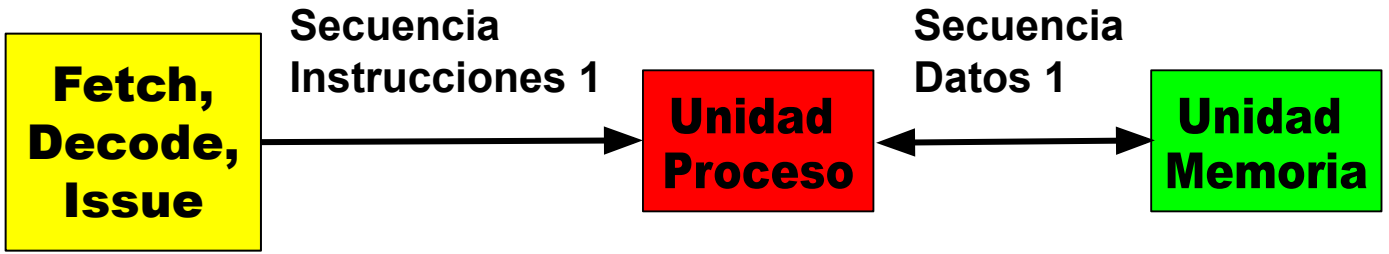
Tipos de sistemas paralelos
Clasificación de Flynn

Propuesta en 1972. Cuatro clasificaciones:

- **SISD: Single Instruction, Single Data**
- **SIMD: Single Instruction, Multiple Data**
- **MISD: Multiple Instruction, Single Data**
- **MIMD: Multiple Instruction, Multiple Data**

Tipos de sistemas paralelos
Clasificación de Flynn

- Single Instruction, Single Data (**SISD**): Una única secuencia de instrucciones y una única fuente de datos.
  - Implementación: Una única unidad de ejecución operando sobre datos en una única memoria.
  - Ejemplos: Arquitectura de procesador único (**primeras computadoras**, **sistemas embebidos simples**).



Clasificación de Flynn

- Single Instruction, Multiple Data (**SIMD**): Una única secuencia de instrucciones y múltiples fuentes de datos.

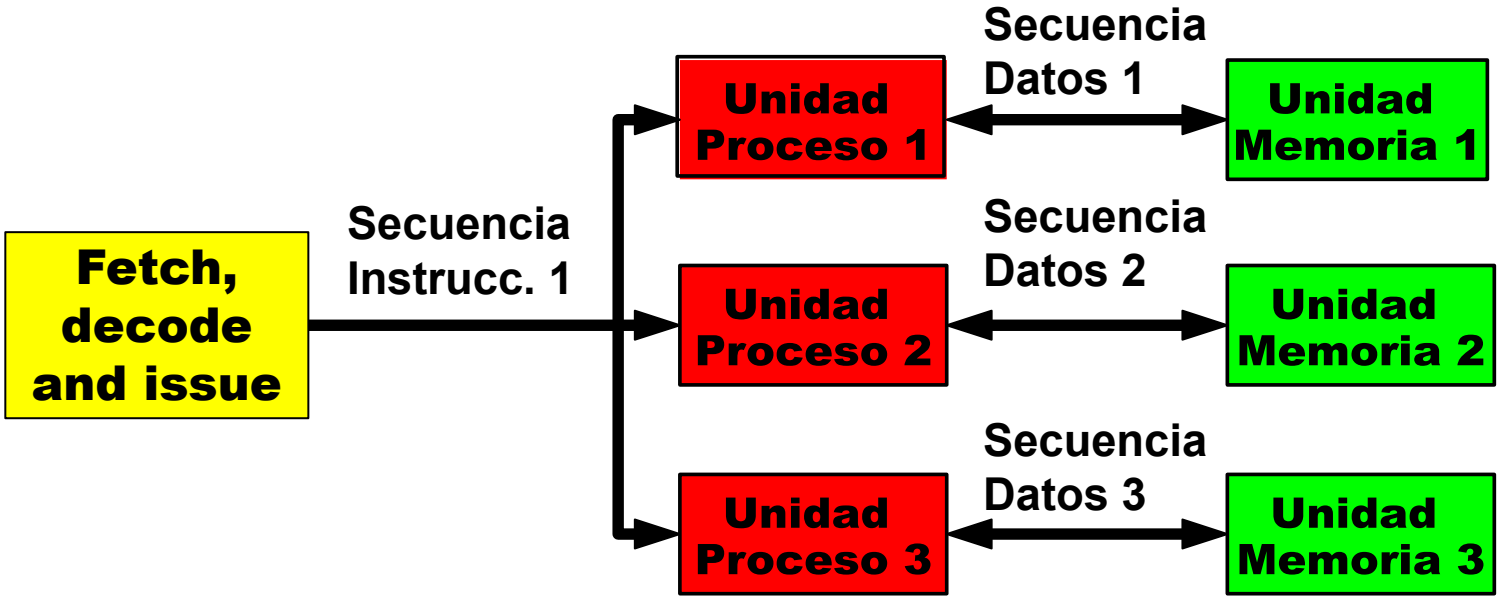


Figura basada en Computer Organization and Architecture 9th Edition William Stallings

Clasificación de Flynn

- Single Instruction, Multiple Data (**SIMD**): Una única secuencia de instrucciones y múltiples fuentes de datos.

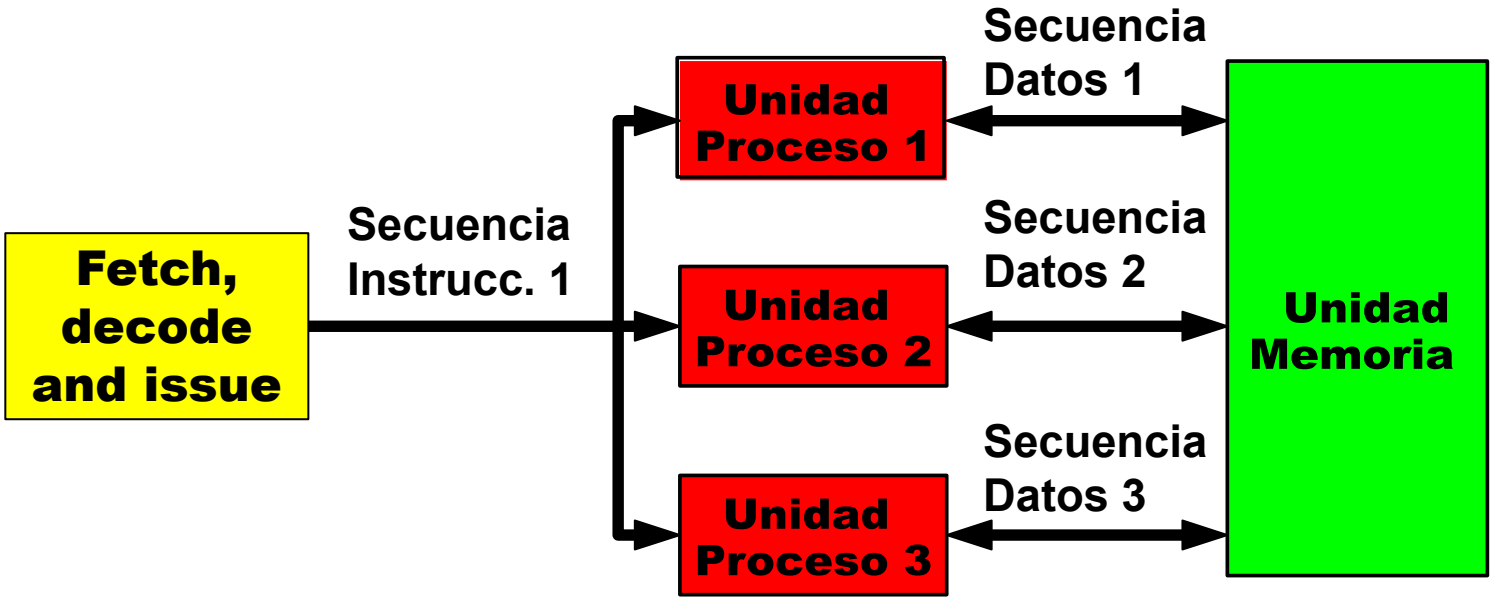


Figura basada en Computer Organization and Architecture 9th Edition William Stallings

Clasificación de Flynn

Single Instruction, Multiple Data (SIMD):

- Distintas implementaciones:
  - Una única etapa de búsqueda (fetch), decodificación y envío de instrucciones (issue) enviando esas instrucciones a **varias unidades de ejecución** (UE). Cada UE ejecuta la instrucción pero sobre distintos datos.
  - **Varias ALUs** que ejecutan la misma instrucción sobre distintos datos (ALUs superescalar).
    - Pipeline sobre la ALU: Una ALU ejecuta la instrucción sobre diferentes datos “casi” al mismo tiempo (pipeline de datos).
- Ejemplos:
  - Procesadores matriciales o vectoriales.
  - Extensiones MMX o SSE de x86
  - GPUs, GPGPUs (también consideradas SIMT)

Clasificación de Flynn

- Multiple instruction, single data (**MISD)**: múltiples secuencias de instrucciones y una única fuente de datos.

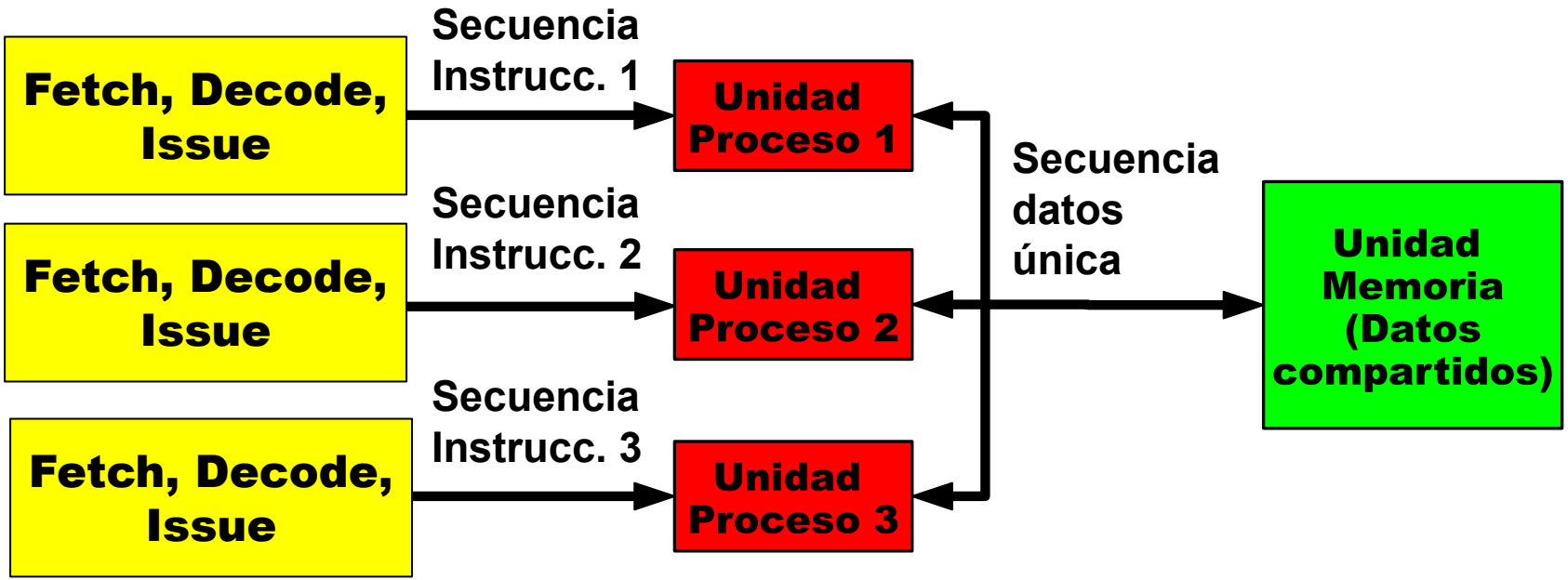


Figura basada en Computer Organization and Architecture 9th Edition William Stallings

Clasificación de Flynn

- Multiple instruction, single data (MISD): múltiples secuencias de instrucciones y una única fuente de datos.
- Implementación: múltiples unidad de ejecución operando sobre datos en una única memoria
- Ejemplos Aplicación:
  - Computación altamente redundante (sistemas que requieren alta tolerancia a fallas) [1].
  - Búsqueda de patrones [2, 3].

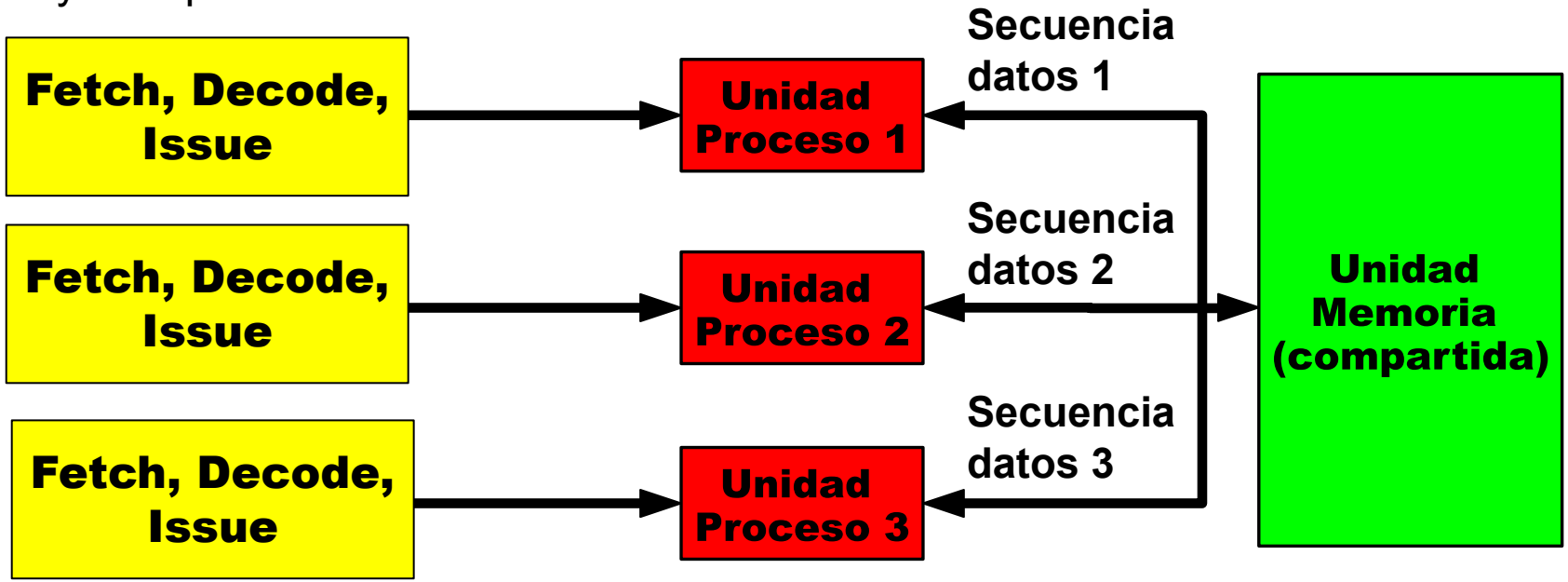
[1] Spector, A.; Gifford, D. (September 1984). "The space shuttle primary computer system". Communications of the ACM. 27 (9): 872–900. doi:10.1145/358234.358246

[2] Arne Halaas, Børge Svingen, Magnar Nedland, Pål Sætrom, Ola Snøve, Jr., and Olaf René Birkeland. 2004. A recursive MISD architecture for pattern matching. IEEE Trans. Very Large Scale Integr. Syst. 12, 7 (July 2004), 727-734.

[3] Algunos autores afirman que éstas arquitecturas no se ajustan a ninguna de los tipos de Flynn.

Clasificación de Flynn

- Multiple instruction, multiple data (**MIMD**): múltiples secuencias de instrucciones y múltiples fuentes de datos.

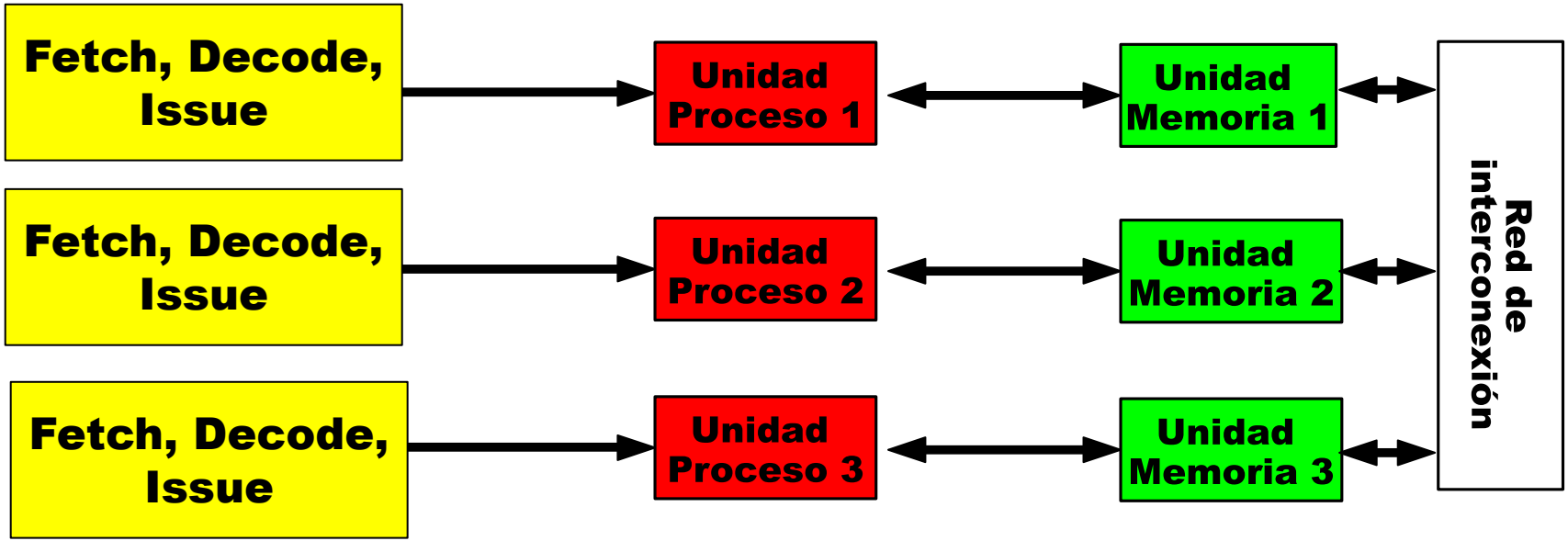


Ejemplo: SMP, NUMA

Figura basada en Computer Organization and Architecture 9th Edition William Stallings

Clasificación de Flynn

- **Multiple instruction, multiple data (MIMD):**



Ejemplo: Cluster

Figura basada en Computer Organization and Architecture 9th Edition William Stallings

Tipos de sistemas paralelos
Clasificación de Flynn

- Multiple instruction, multiple data (MIMD):
  - Implementaciones:
    - Memoria compartida: SMP, NUMA.
    - Cada procesador tiene su memoria (Memoria Distribuida): Cluster.

Nota 1: Un procesador superescalar NO es un ejemplo de MIMD, ya que, si bien puede ejecutar varias instrucciones al mismo tiempo, estas instrucciones forman parte de la misma secuencia de instrucciones. Puede ser SISD o SIMD (SIMD si soporta instrucciones vectoriales como MMX o SSE)
Nota 2: En los procesadores multithreading, todos los hilos que se ejecutan al mismo tiempo deben ser parte del mismo proceso, por lo que deben compartir recursos (como memoria caché, espacio de memoria, etc.). Por lo que puede suponerse que los hilos provienen del mismo "programa". Por lo tanto, estos procesadores son del tipo SISD o SIMD.

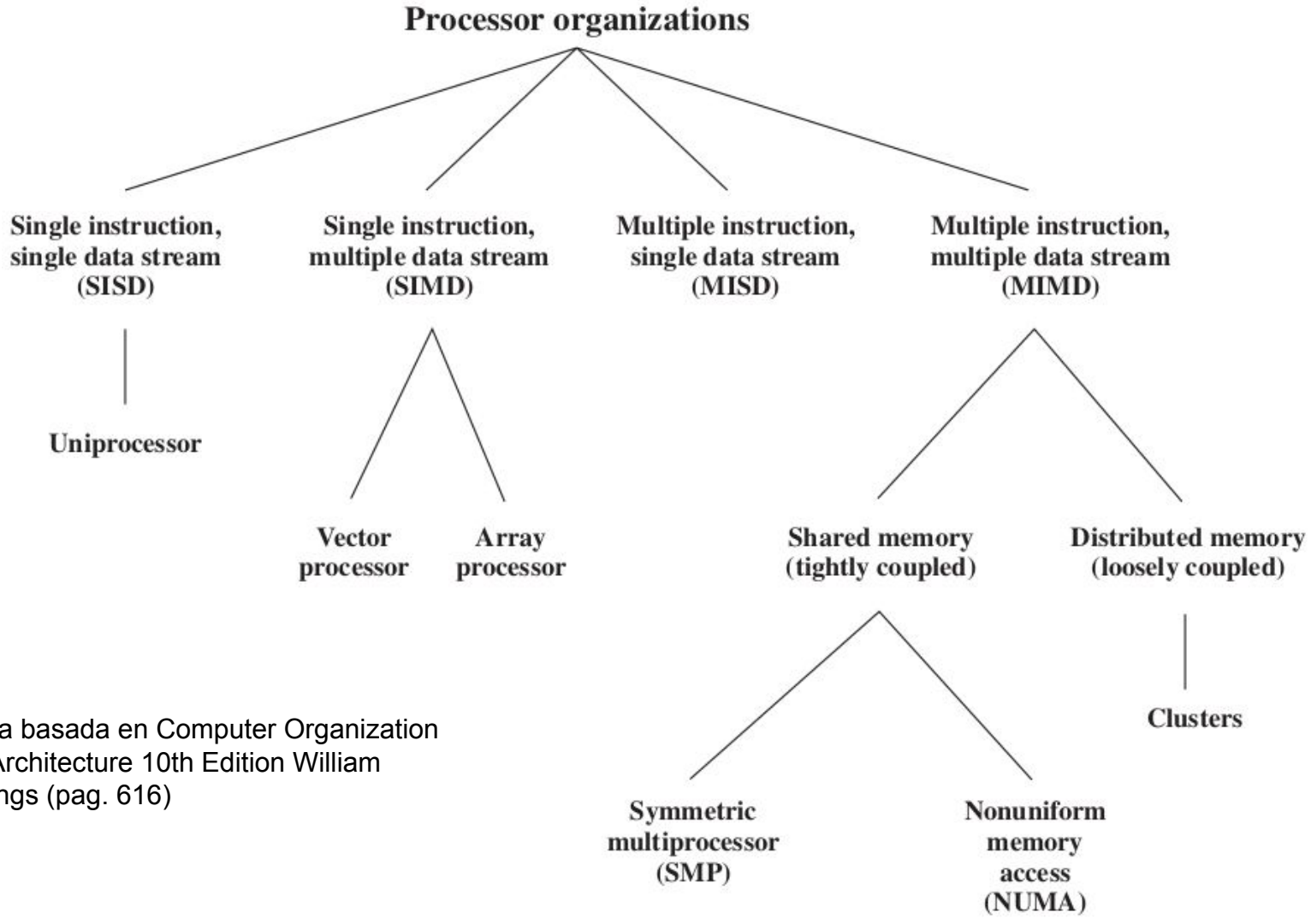


Figura basada en Computer Organization and Architecture 10th Edition William Stallings (pag. 616)

Clasificación de Flynn

- **SIMT** (Single Instruction, Multiple Threads): Muchos hilos ejecutando la misma tarea de modo sincronizado.
  - Si trabaja sobre diferentes datos, mucha similitud con SIMD.
- Procesadores **vectoriales** o **matriciales**: Trabajan con vectores y matrices.
  - Muchos autores sugieren que son SIMD.

Multithreading (Multi hilo)

Ventaja del uso de hilos

- Facilidad para escribir un programa, ya que:
  - El problema puede dividirse en sub-problemas, resolviendo cada sub-problema con diferentes hilos.
  - Pueden usarse llamadas al sistema bloqueantes.
  - Ejemplos de aplicaciones que son más eficientes gracias a los hilos: aplicaciones cliente-servidor, sistemas distribuidos, interfaces de usuario, aplicaciones productor-consumidor, etc.
  - **Permiten crear aplicaciones paralelas con memoria compartida.**

- **Hacer uso más eficiente de arquitecturas que permiten la ejecución de instrucciones en paralelo** (superescalar o multinúcleo)**.**

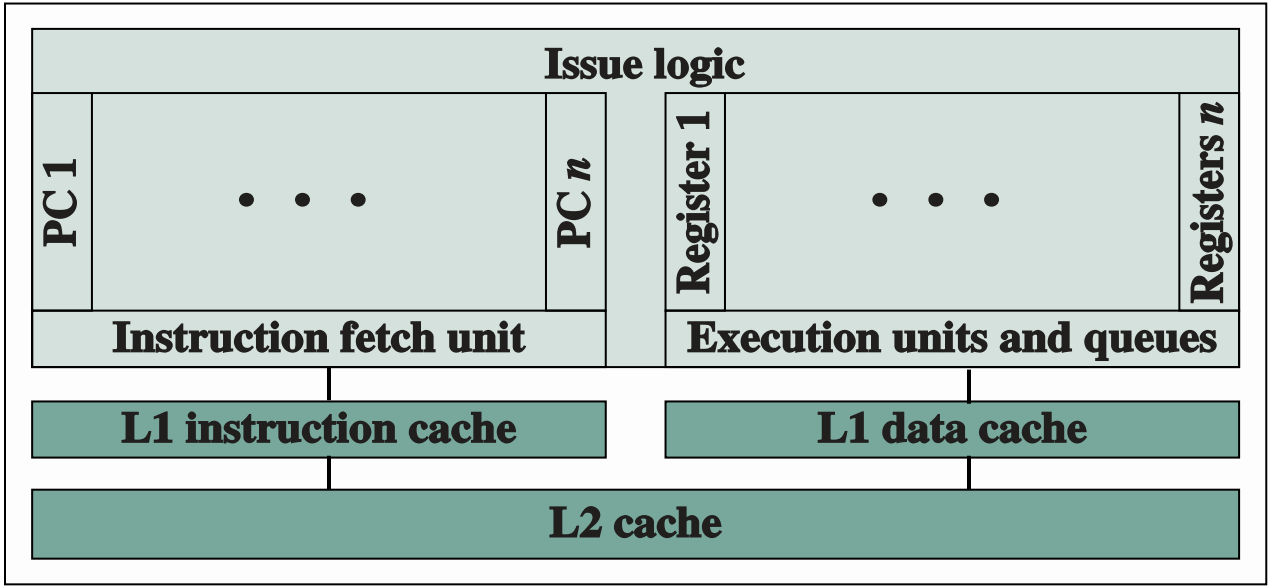
Esta es la ventaja que nos interesa en esta materia

Multithreading simultaneo sobre procesador de un núcleo

- Procesador **superescalar** que toma varias instrucciones de **diferentes hilos**, para incrementar la independencia entre instrucciones a ejecutar al mismo tiempo.
- **Incrementa más el speedup**. Mejor performance para tratar con dependencias de datos y accesos a memoria.

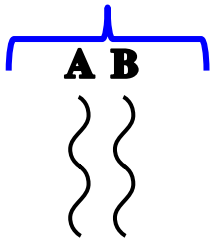
F1	D1	ADD1	ADD2					
F2	D2	LD (Acceso a memoria)						
	F1	D1	MUL1	MUL2	MUL3			
	F2	D2	ADD1	ADD2				
		F1	D1	ADD1	ADD2			
		F1	D1	MUL1	MUL2	MUL3		

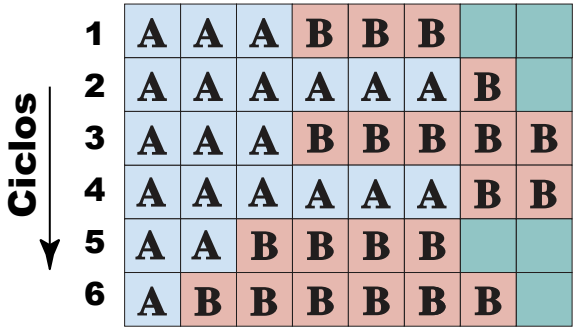
Multithreading simultaneo sobre procesador de un núcleo



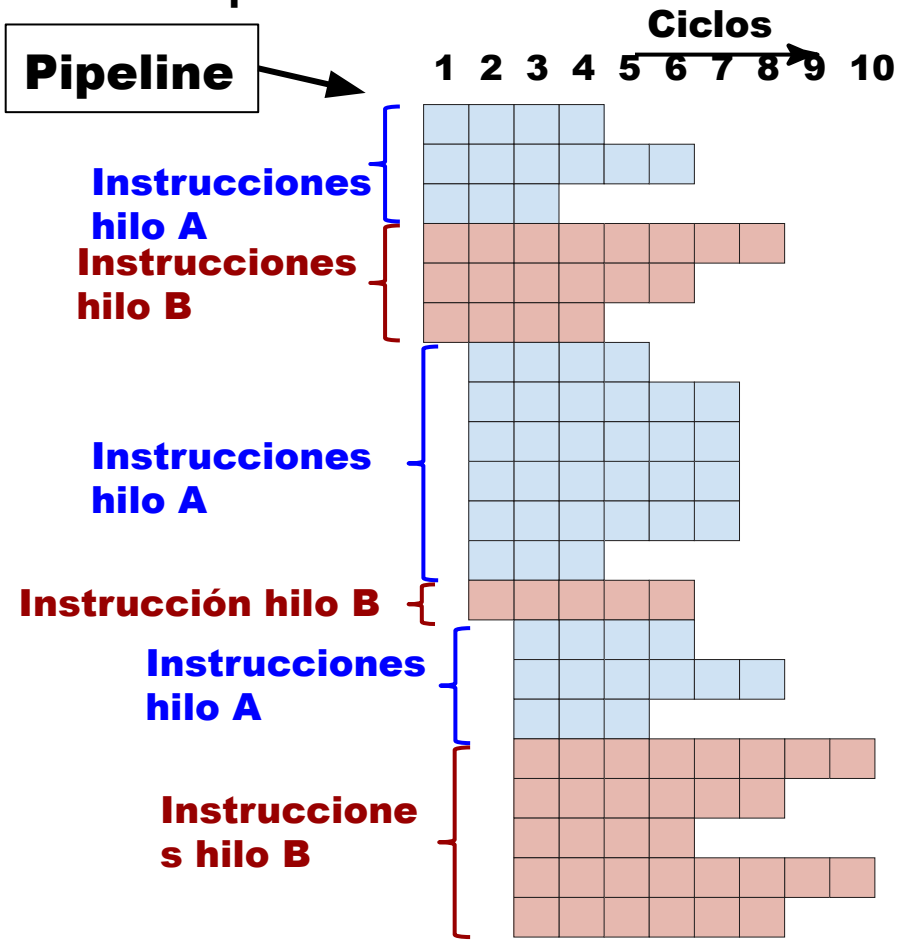
La microarquitectura Kaby Lake (i3, i5 e i7 año 2017) y posteriores de Intel responde a este diseño (Hyperthreading)

El SO ve 2 núcleos lógicos





Secuencia de entrada a unidades de ejecución



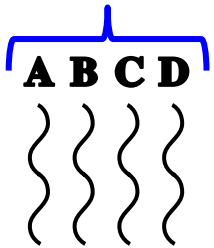
UNCUYO
UNIVERSIDAD NACIONAL DE CUYO

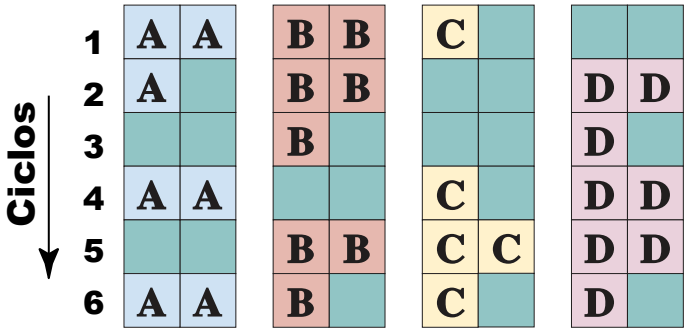
FACULTAD DE INGENIERÍA

Licenciatura en Ciencias de la Computación

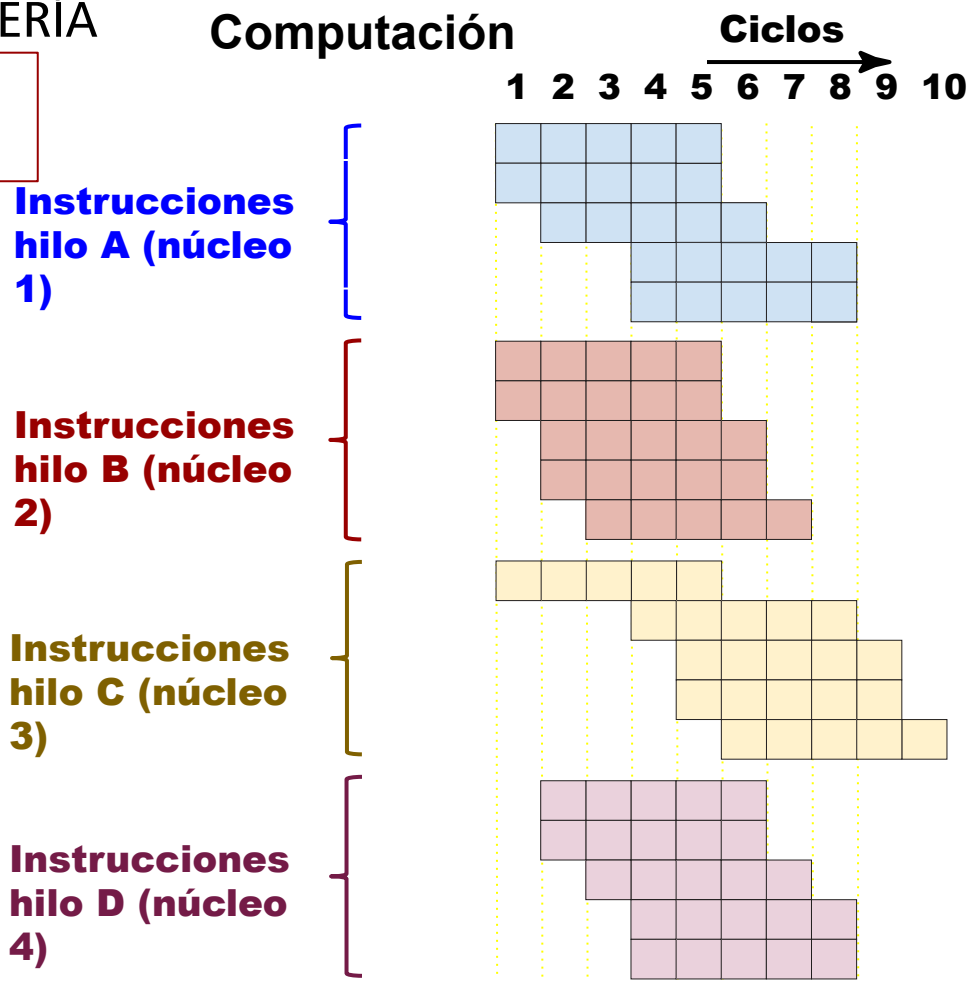
La microarquitectura del ARM Cortex A75 (año 2017) responde a este diseño

El SO ve 4 núcleos lógicos



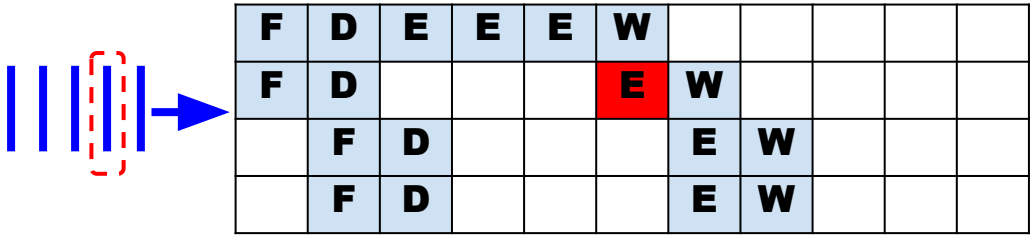


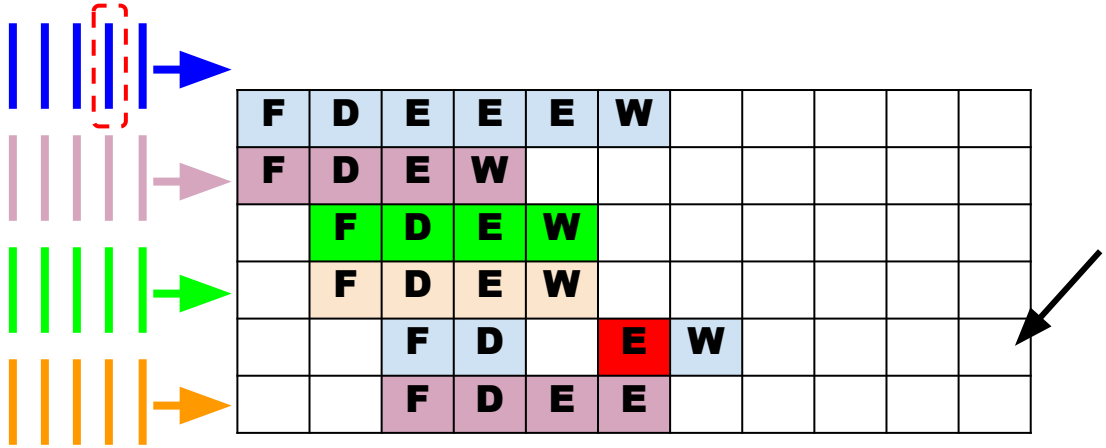
Secuencia de entrada a unidades de ejecución



Multithreading (Multi hilo)

Ejemplo de funcionamiento con procesador superescalar de dos vías, con problema de dependencia de datos RAW



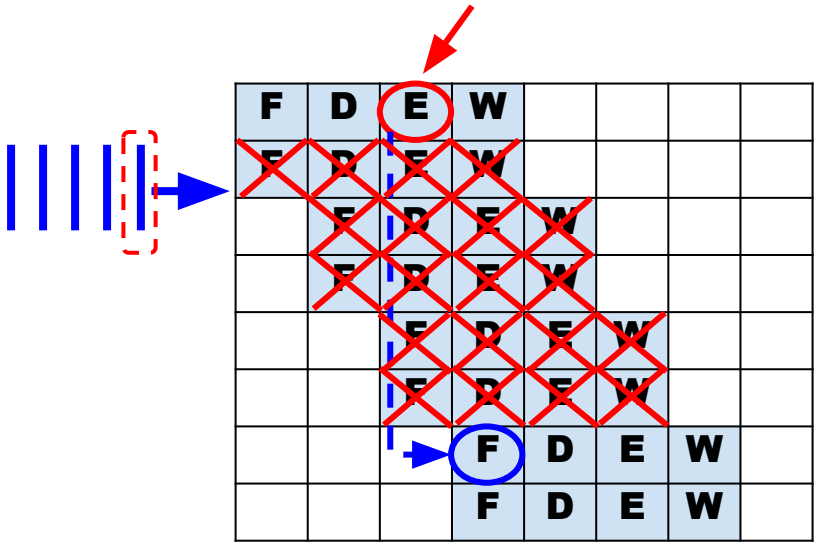


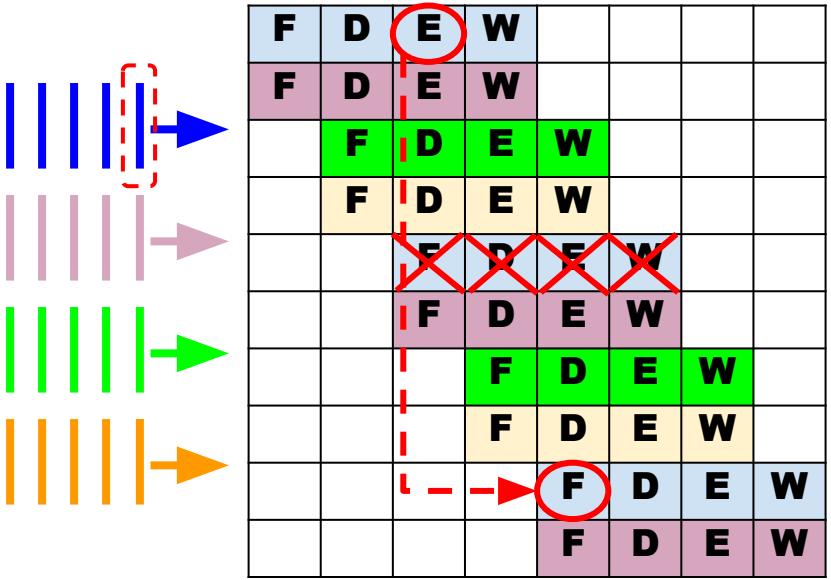
Si se combina con ejecución fuera de orden, podría no perderse ningún ciclo, ya que en lugar de ejecutar la segunda instrucción del primer hilo (que tiene problemas de dependencia), podría ejecutarse la tercera antes de la segunda.

Multithreading (Multi hilo)

Ejemplo de funcionamiento con procesador superescalar de 2 vías, con problema de conflicto estructural (salto condicional)

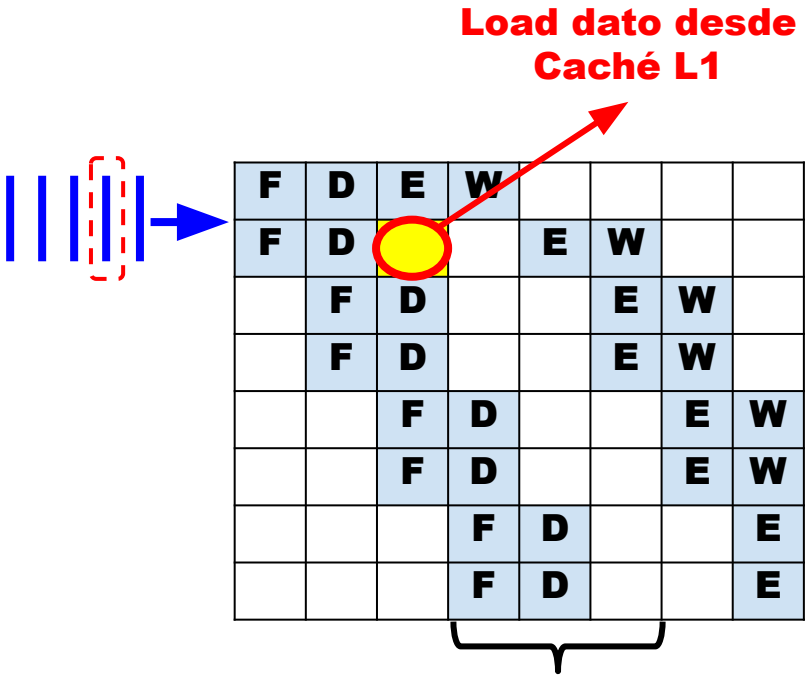
Instrucción de salto condicional

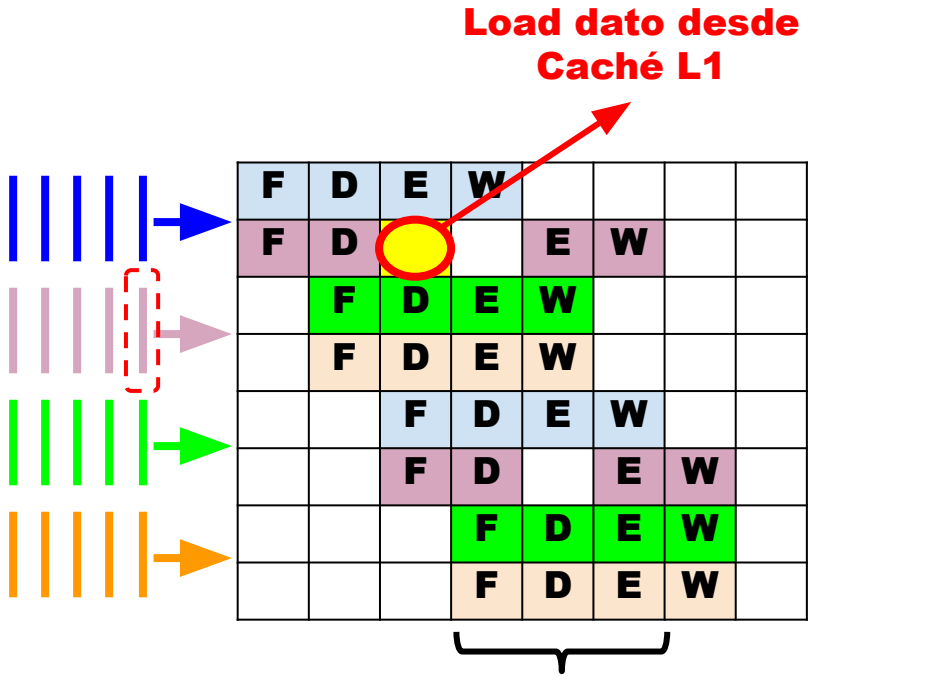




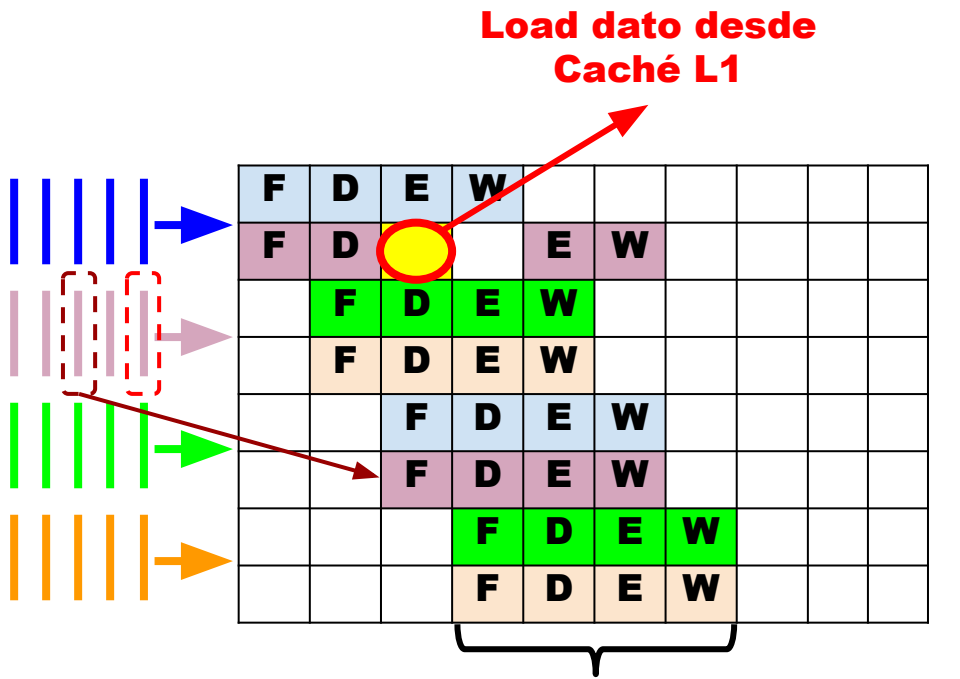
Multithreading (Multi hilo)

Ejemplo de funcionamiento con procesador superescalar de dos vías, acceso a memoria Caché L1 y 4 hilos





Ejemplo de funcionamiento con procesador superescalar de dos vías, acceso a memoria Caché L1, 4 hilos y ejecución fuera de orden.



Notas:

1) En el ciclo 5 tenemos 3 instrucciones en ejecución. Esto es posible si las instrucciones utilizan unidades de ejecución diferentes.

2) En el ciclo 6 hay tres instrucciones escribiendo resultados en memoria, esto es posible si al menos una no escribe en memoria (es decir, uno de los resultados queda en registros).

3) En un procesador fuera de orden, si la instrucción 2 del hilo 2 depende de la instrucción 1, el procesador podría ejecutar la instrucción 3 antes de la 2.

Colección de productos	Intel® Core™ i7 processors (14th gen)
Nombre de código	Products formerly Raptor Lake
Segmento vertical	Embedded
Número de procesador	i7-14701TE
Litografía	Intel 7

Especificaciones de la CPU

Cantidad de núcleos	8
Cantidad de Performance-cores	8
Cantidad de Efficient-cores	0
Total de subprocesos	16

https://www.intel.la/content/www/xl/es/products/sku/238774/intel-core-i7-processor-14701te-33m-cache-up-to-5-20-ghz/specifications.html

Paralelismo a nivel de hilos: Librería thread de C++

- Permite crear y administrar hilos. Muy similar a threading de Python
  - Es necesario agregar: #include <thread>
  - Java: clase **java.lang.Thread**
- Los hilos deben escribirse como funciones.
- Crear hilo:

```
void codigo_hilo1(){
    Declaración de la función
}
```

Función que implementa el hilo

- Crear y comenzar ejecución del hilo:

```
std::thread hilo1(codigo_hilo1, argumento1, argumento2);
```

Paralelismo a nivel de hilos: Librería thread de C++

- Implementación típica en programas paralelos: lista o arreglo de hilos.
- Esperar a que un hilo finalice para continuar con el programa:
  hilo_hijo.join()
  - El programa principal o hilo padre se bloqueará en dicha instrucción hasta que el hilo *hilo_hijo* termine su ejecución.

Temas

- Tipos de sistemas paralelos (Clasificación de Flynn)
- Paralelismo a nivel de instrucción
- Paralelismo a nivel de hilos (Procesadores multihilo simultáneo)
- Paralelismo a nivel de núcleos
  - Arquitecturas multi núcleo simétricas
  - Arquitecturas multi núcleo heterogéneas con igual ISA o con ISA diferentes
  - Sistemas CPU-GPU, CPU-DSP. GPGPU. Introducción a CUDA y OpenCL
  - Sistemas operativos para multiprocesadores.
- Paralelismo a nivel de computadoras (Cluster)
- → **Paralelismo a nivel de datos (SIMD)**
- Sistemas RAID
- Performance y escalabilidad
  - Speedup y eficiencia
  - Modelos de problemas paralelizables
  - Paralelismo en el software

Procesadores Vectoriales y SIMD

Procesadores vectoriales actuales (SIMD)

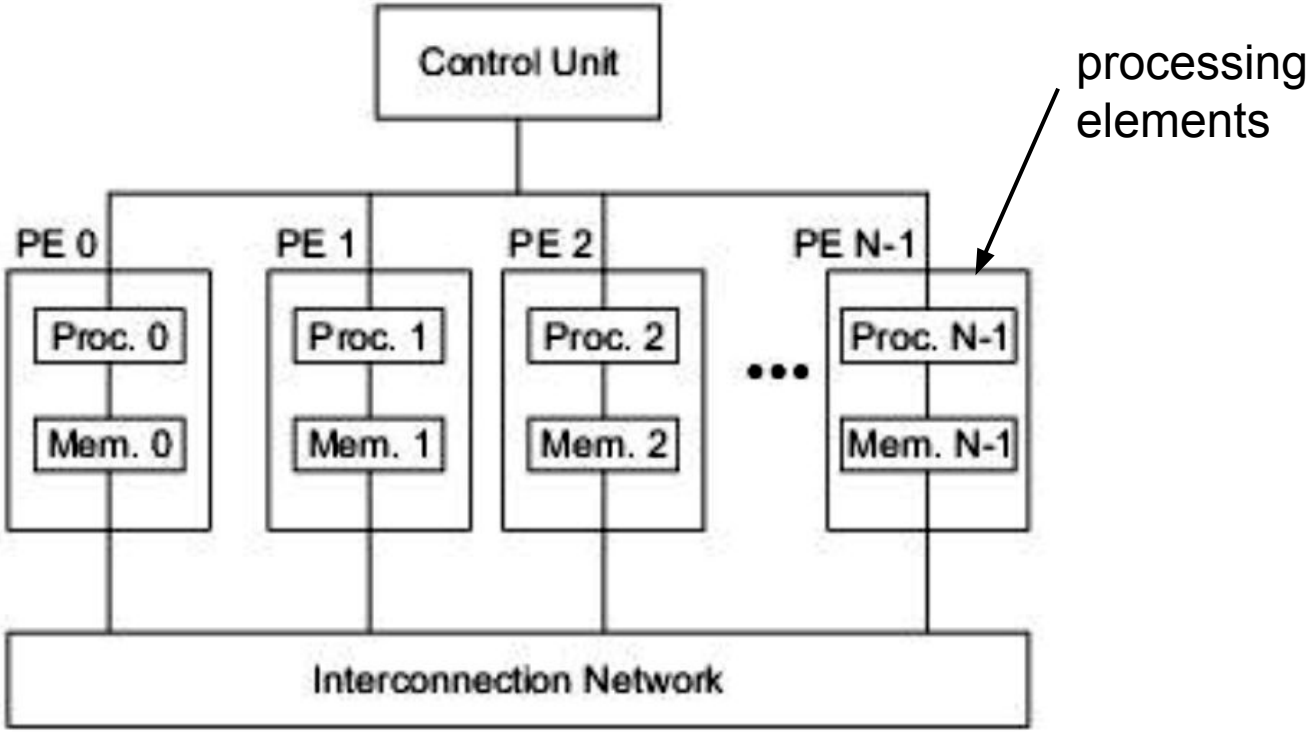
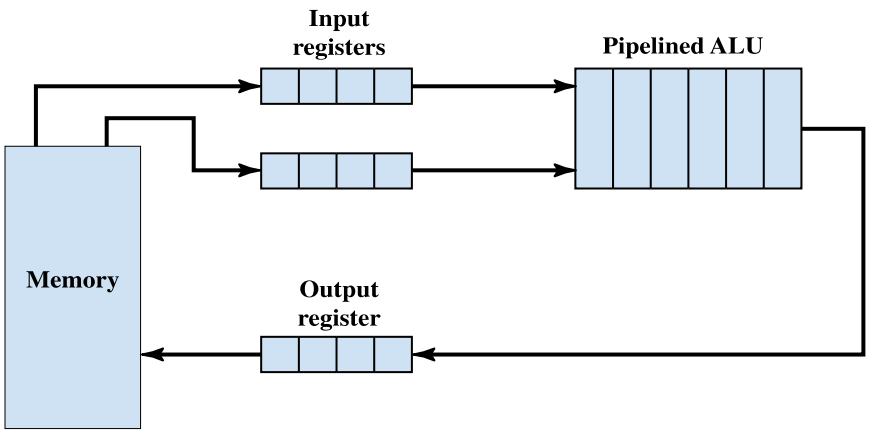
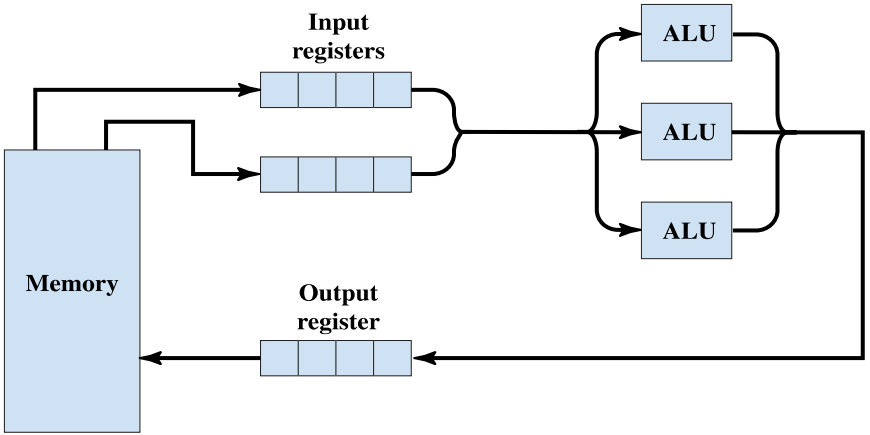


Figura obtenida de Kai Hwang, “Advanced Computer Architecture Parallelism Scalability Programmability”, edición 2001, página 27

Procesadores Vectoriales y SIMD: Implementaciones



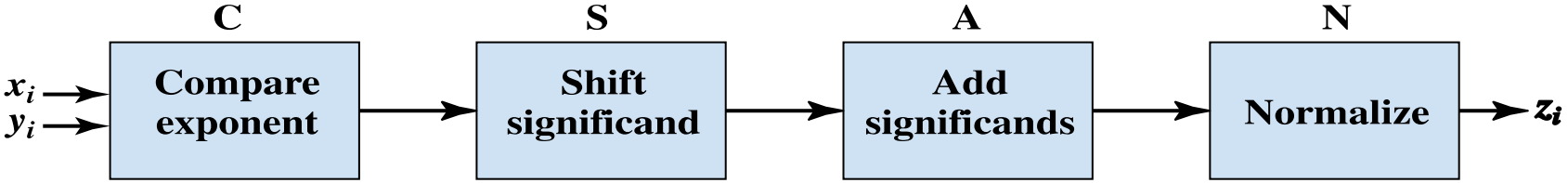
Pipelined ALU

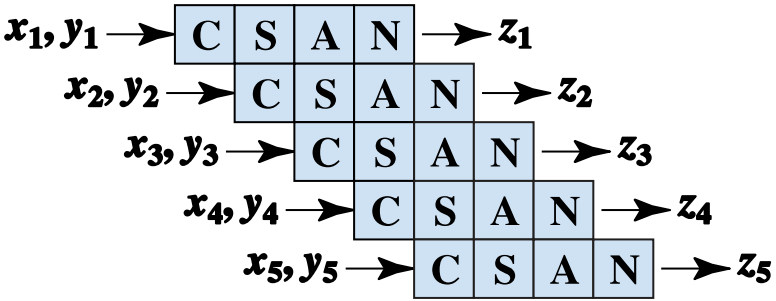


Parallel ALUs

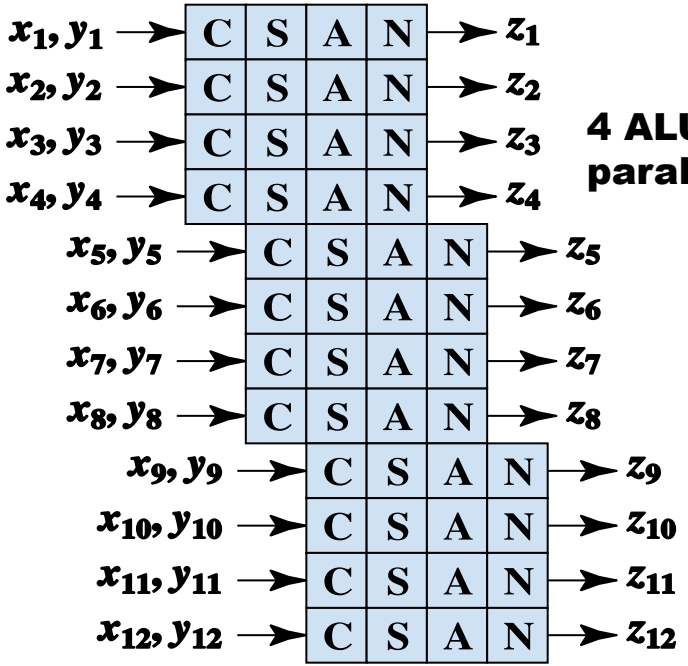
Figura basada en Computer Organization and Architecture 9th Edition William Stallings

Ejemplo implementación suma punto flotante





Pipelined ALU



4 ALUs en paralelo

Nota: los vectores pueden tener cientos de componentes. Se operan en paralelo grupos de componentes según la ALU disponible.

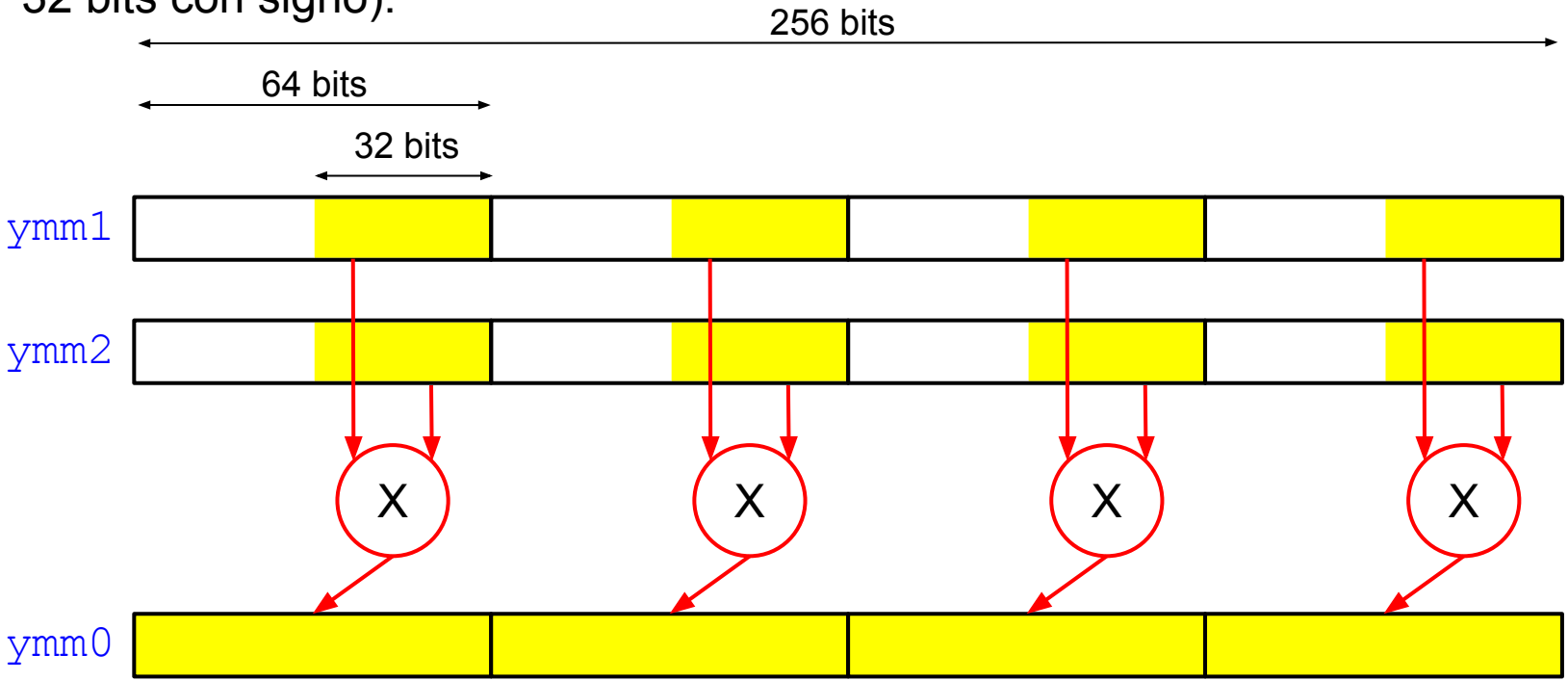
Figura basada en Computer Organization and Architecture 9th Edition William Stallings

Computación vectorial: Ejemplos: extensiones SIMD de x86

- **SSE** (Streaming SIMD Extensions). 1999, mejora a la extensión MMX.
  - Ultima versión: SSE 4 (2006).
  - Agrega 70 instrucciones (la mayoría de punto flotante) y nuevos registros de 128 bits (conjunto de registros XMM).
- **AVX** (Advanced Vector Extensions). Propuesto en 2008, implementado en 2011.
  - Agrega registros de 256 bits (denominados XMM0-XMM15 a YMM0-YMM15).
  - Sistemas Operativos compatibles: Linux 2.6.30 (2009), Windows 7.
- **AVX512F** (2016).
  - Agrega registros de 512 bits.
- **AMX** (Advanced Matrix Extensions).
  - Propuesto en 2020. Primera implementación: Microarquitectura Sapphire Rapids (Procesadores Xeon para servidores), Enero de 2023.

Ejemplos: extensiones SIMD de x86

`vpmuldq ymm0, ymm2, ymm1` (extensión AVX2. Multiplica enteros de 32 bits con signo).



Fuente: https://www.intel.com/content/www/us/en/docs/intrinsics-guide/index.html

```
//g++ -mavx ejemplo.cpp -o ejemplo.o
#include <stdio.h>
#include <iostream>
#include <immintrin.h>
using namespace std;
int main() {
    // Crear dos vectores
    __m128 a = _mm_set_ps(4.1, 3.0, 2.0, 1.0);
    __m128 b = _mm_set_ps(8.1, 7.0, 6.0, 5.5);
    // multiplicar los vectores
    __m128 c = _mm_mul_ps(a, b);
    // Imprimir los resultados
    cout <<"vector a [ "<< a[3] << "," << a[2] << "," << a[1] << "," << a[0] <<" ]"<< endl;
    cout <<"vector b [ "<< b[3] << "," << b[2] << "," << b[1] << "," << b[0] <<" ]"<< endl;
    cout <<"vector c [ "<< c[3] << "," << c[2] << "," << c[1] << "," << c[0] <<" ]"<< endl;
    return 0;
}
```

Ejemplo con la instrucción **mulps**, extensión SSE.

De la página de Intel:

Synopsis

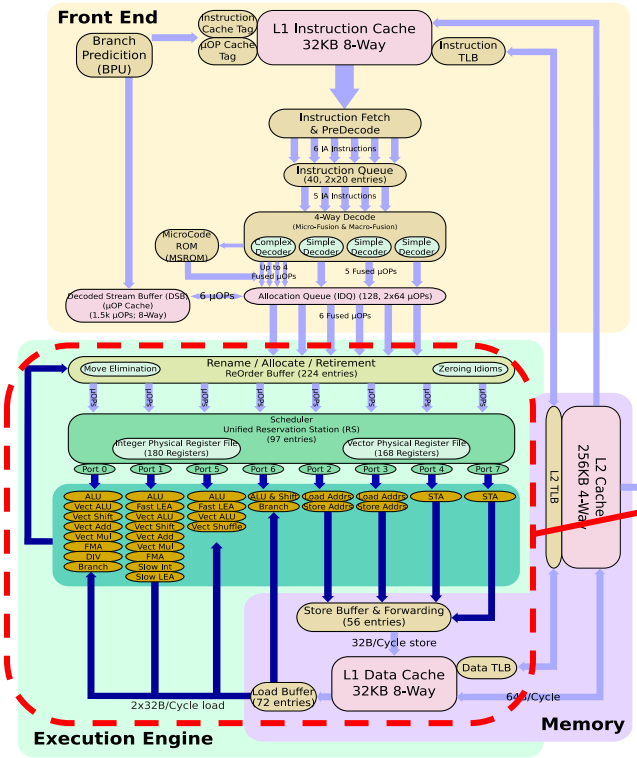
```
__m128 _mm_mul_ps (__m128 a, __m128 b)
#include <xmmintrin.h>
Instruction: mulps xmm, xmm
CPUID Flags: SSE
```

Description

Multiply packed single-precision (32-bit) floating-point elements in `a` and `b`, and store the results in `dst`.

```
pablo@pablo-HP-1000-Notebook-PC:~$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 58
model name      : Intel(R) Core(TM) i3-3110M CPU @ 2.40GHz
stepping        : 9
microcode       : 0x21
cpu MHz         : 1197.366
cache size      : 3072 KB
physical id     : 0
siblings        : 4
core id         : 0
cpu cores       : 2
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtr
h dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx rdtscp lm co
s rep_good nopl xtopology nonstop_tsc cpuid aperfmperf pni pclmul
st tm2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic popcnt tsc_
```

```
pablo@pablo-HP-1000-Notebook-PC:~$ g++ -mavx ejemplo.cpp
 -o ejemplo.o
pablo@pablo-HP-1000-Notebook-PC:~$ ./ejemplo.o
vector a [ 4.1 , 3 , 2 , 1 ]
vector b [ 8.1 , 7 , 6 , 5.5 ]
vector c [ 33.21 , 21 , 12 , 5.5 ]
pablo@pablo-HP-1000-Notebook-PC:~$
```


Front End
Branch Predicition (BPU)
Instruction Cache Tag
µOP Cache Tag
L1 Instruction Cache 32KB 8-Way
Instruction TLB
Instruction Fetch & PreDecode
6 IA Instructions
Instruction Queue (40, 2x20 entries)
5 IA Instructions
4-Way Decode (Micro-Fusion & Macro-Fusion)
MicroCode ROM (MSROM)
Complex Decoder
Simple Decoder
Simple Decoder
Simple Decoder
Up to 4 Fused µOPs
5 Fused µOPs
Decoded Stream Buffer (DSB) (µOP Cache) (1.5k µOPs; 8-Way)
6 µOPs
Allocation Queue (IDQ) (128, 2x64 µOPs)
6 Fused µOPs
Move Elimination
Rename / Allocate / Retirement
ReOrder Buffer (224 entries)
Zeroing Idioms
Scheduler
Unified Reservation Station (RS)
(97 entries)
Integer Physical Register File (180 Registers)
Vector Physical Register File (168 Registers)
Port 0
Port 1
Port 5
Port 6
Port 2
Port 3
Port 4
Port 7
ALU
Vect ALU
Vect Shift
Vect Add
Vect Mul
FMA
DIV
Branch
ALU
Fast LEA
Vect ALU
Vect Shift
Vect Add
Vect Mul
FMA
Slow Int
Slow LEA
ALU
Fast LEA
Vect ALU
Vect Shuffle
ALU & Shift
Branch
Load Addrs
Store Addrs
Load Addrs
Store Addrs
STA
STA
Store Buffer & Forwarding (56 entries)
32B/Cycle store
L1 Data Cache 32KB 8-Way
Data TLB
64B/Cycle
Load Buffer (72 entries)
2x32B/Cycle load
L2 TLB
L2 Cache 256KB 4-Way
L3
Execution Engine
Memory

Ejemplo procesadores SIMD: Extensión NEON de ARM

- Extensión NEON en ARM-64 bits: 32 registros de 128 bits (V0 a V31) (utilizados también por instrucciones escalares con números de punto flotante)
- Extensión NEON en ARM-32 bits: Se combinan registros de 64 bits para formar registros de 128 bits que se comportan como la extensión NEON para ARM-64 bits.

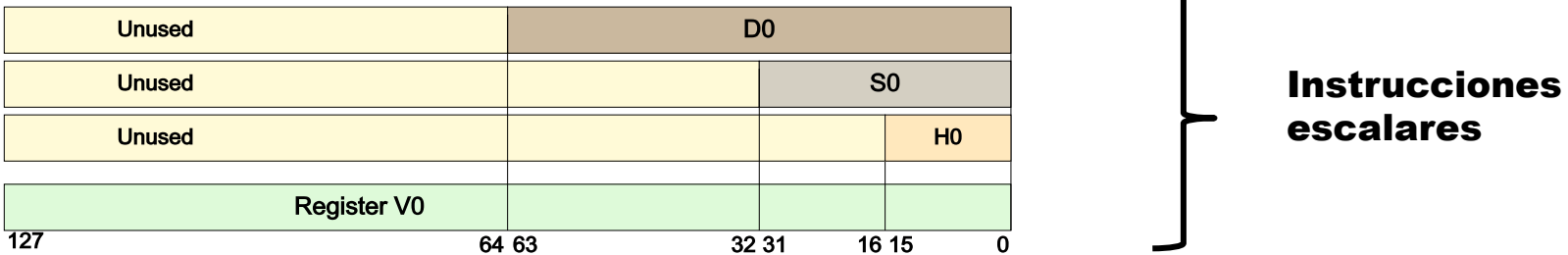


Figura obtenida de ARM Cortex -A Series, Programmer's Guide for ARMv8-A, Version: 1.0, pag 4-17 a 4-20

Ejemplo procesadores SIMD: Extensión NEON de ARM

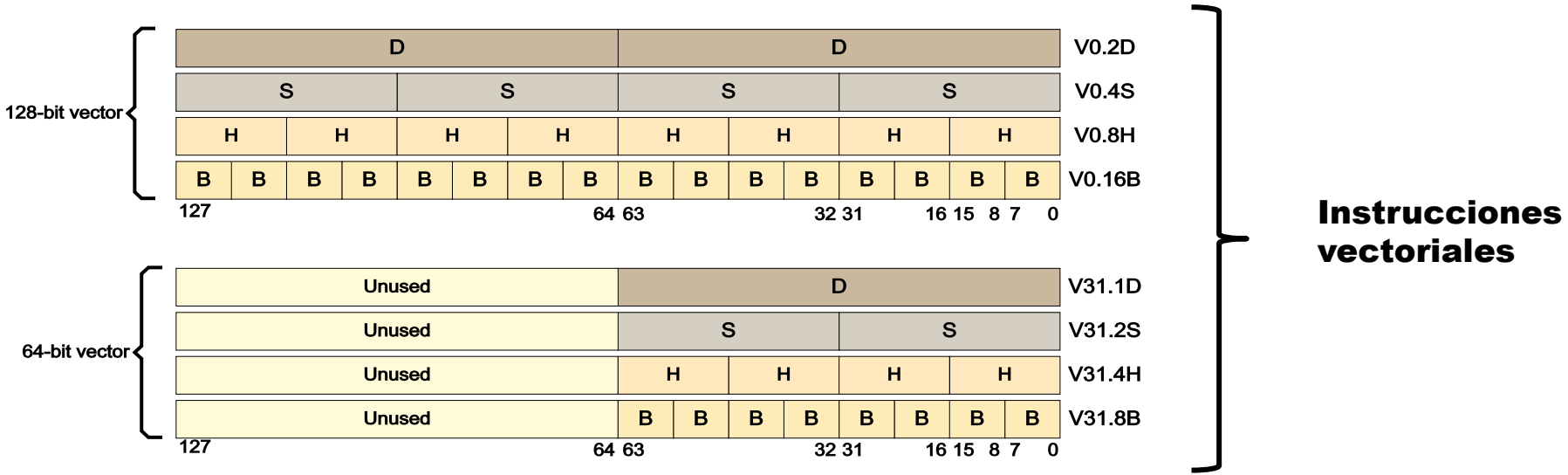
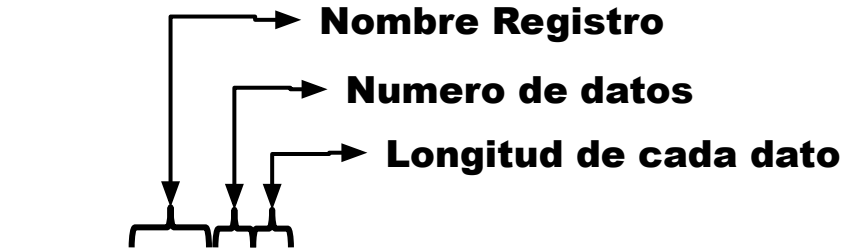


Figura obtenida de ARM Cortex -A Series, Programmer's Guide for ARMv8-A, Version: 1.0, pag 4-17 a 4-20

Ejemplo procesadores SIMD: Extensión NEON de ARM



ADD V0.4H, V1.4H, V2.4H

(V0=V1+V2)

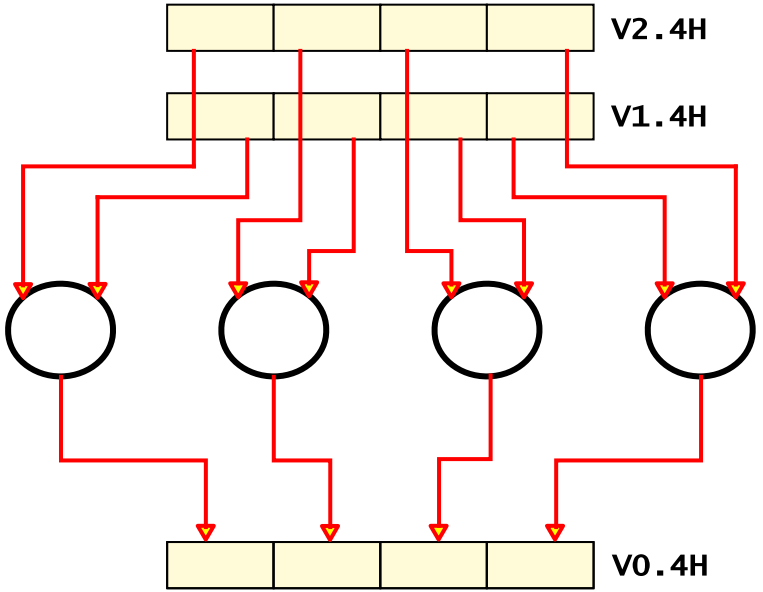
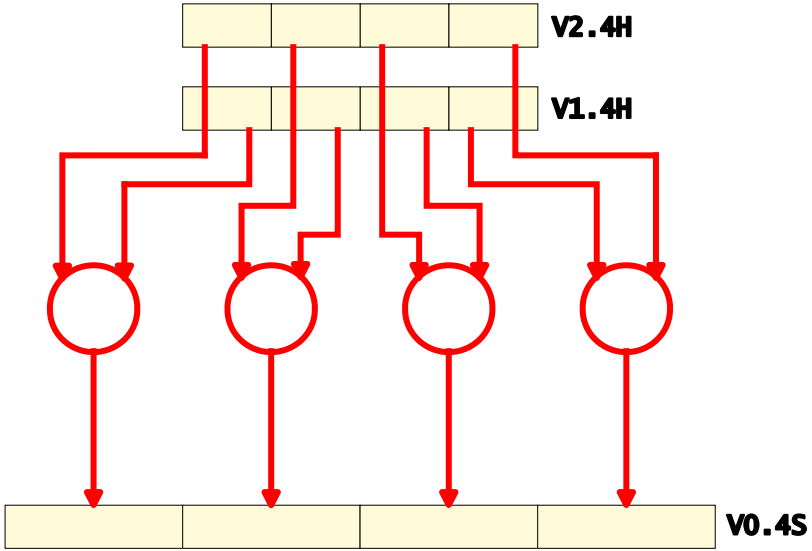
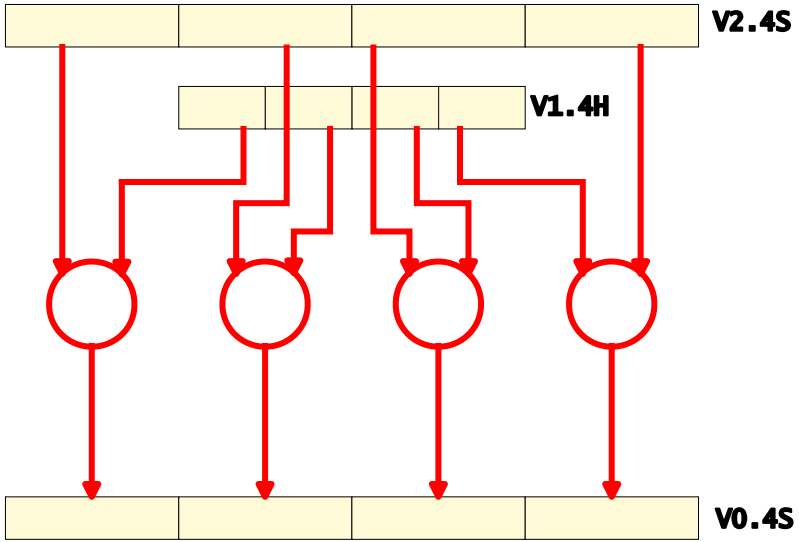


Figura obtenida de ARM Cortex -A Series, Programmer's Guide for ARMv8-A, Version: 1.0, pag 4-17 a 4-20
Ejemplos obtenidos de ARM Cortex -A Series, Programmer's Guide for ARMv8-A, Version: 1.0, pag 7-9

Ejemplo procesadores SIMD: Extensión NEON de ARM



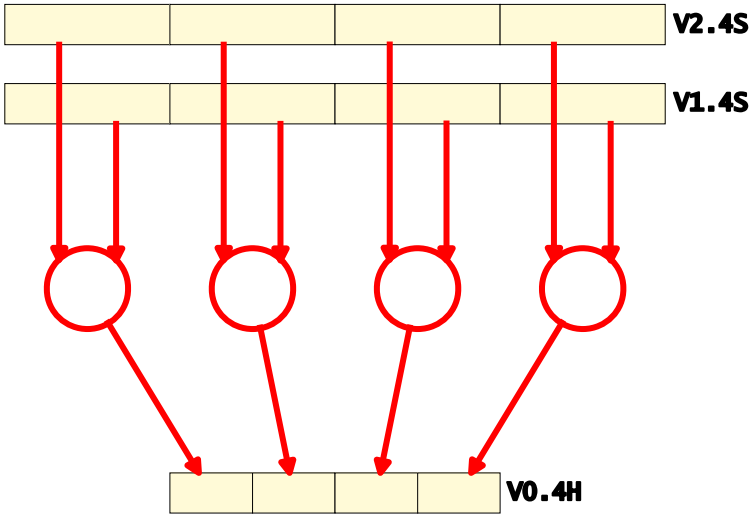
SADDL V0.4S, V1.4H, V2.4H



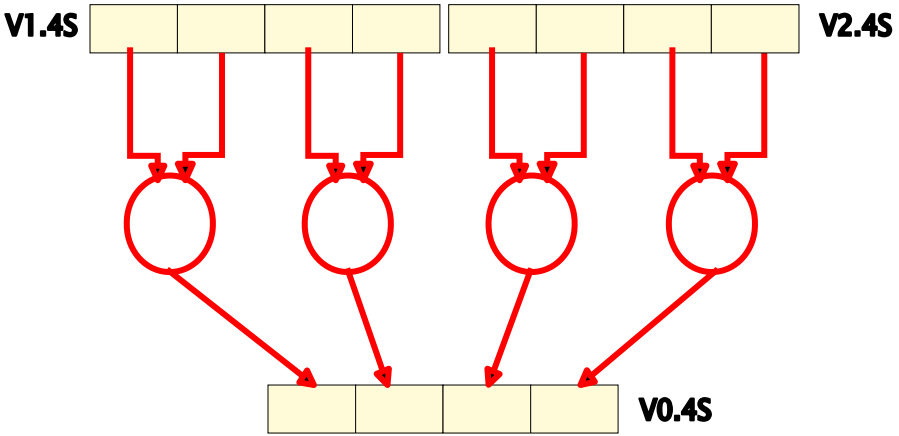
SADDW V0.4S, V1.4H, V2.4S

Figura obtenida de ARM Cortex -A Series, Programmer's Guide for ARMv8-A, Version: 1.0, pag 4-17 a 4-20
Ejemplos obtenidos de ARM Cortex -A Series, Programmer's Guide for ARMv8-A, Version: 1.0, pag 7-9

Ejemplo procesadores SIMD: Extensión NEON de ARM



SUBHN V0.4H, V1.4S, V2.4S



ADDP V0.4S, V1.4S, V2.4S
(pairwise operations)

Figura obtenida de ARM Cortex -A Series, Programmer's Guide for ARMv8-A, Version: 1.0, pag 4-17 a 4-20
Ejemplos obtenidos de ARM Cortex -A Series, Programmer's Guide for ARMv8-A, Version: 1.0, pag 7-9

Bibliografía:

- William Stallings, "Computer Organization and Architecture”, 10º edición, editorial Pearson, año 2016.
- Tanenbaum and Bos, “Modern Operating Systems”, 4° edición, editorial Pearson, año 2015.
- Kai Hwang, “Advanced Computer Architecture, Parallelism, Scalability, Programmability”, 2º edición, editorial Mc Graw-Hill, año 2011.
- Hesham and Mostafa, “Advanced Computer Architecture and Parallel Processing”, 1° edición, editorial Wiley, año 2005.
- ARM, "ARM Cortex-A Series Programmer’s Guide for ARMv8-A", Version 1.0, año 2015