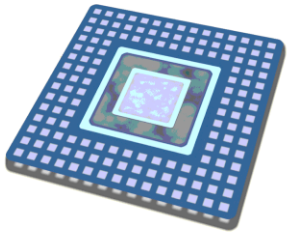


Departamento de
Arquitectura y
Tecnología de Computadores
UNIVERSIDAD DE SEVILLA

Tema 2: Jerarquía de Memoria

2.1. Memoria Caché

Arquitectura de Computadores

Grado en Ingeniería Informática

Índice de contenidos

1 - Jerarquía de memoria
- 1.1 - Conceptos y funcionamiento.
- 1.2 - Rendimiento.

2 - Memoria caché
- 2.1 - Estructura y funcionamiento básico.
- 2.2 - Políticas de ubicación e identificación
- 2.3 - Políticas de reemplazo
- 2.4 - Políticas de escritura
- 2.5 - Tipos de fallos
- 2.6 - Rendimiento

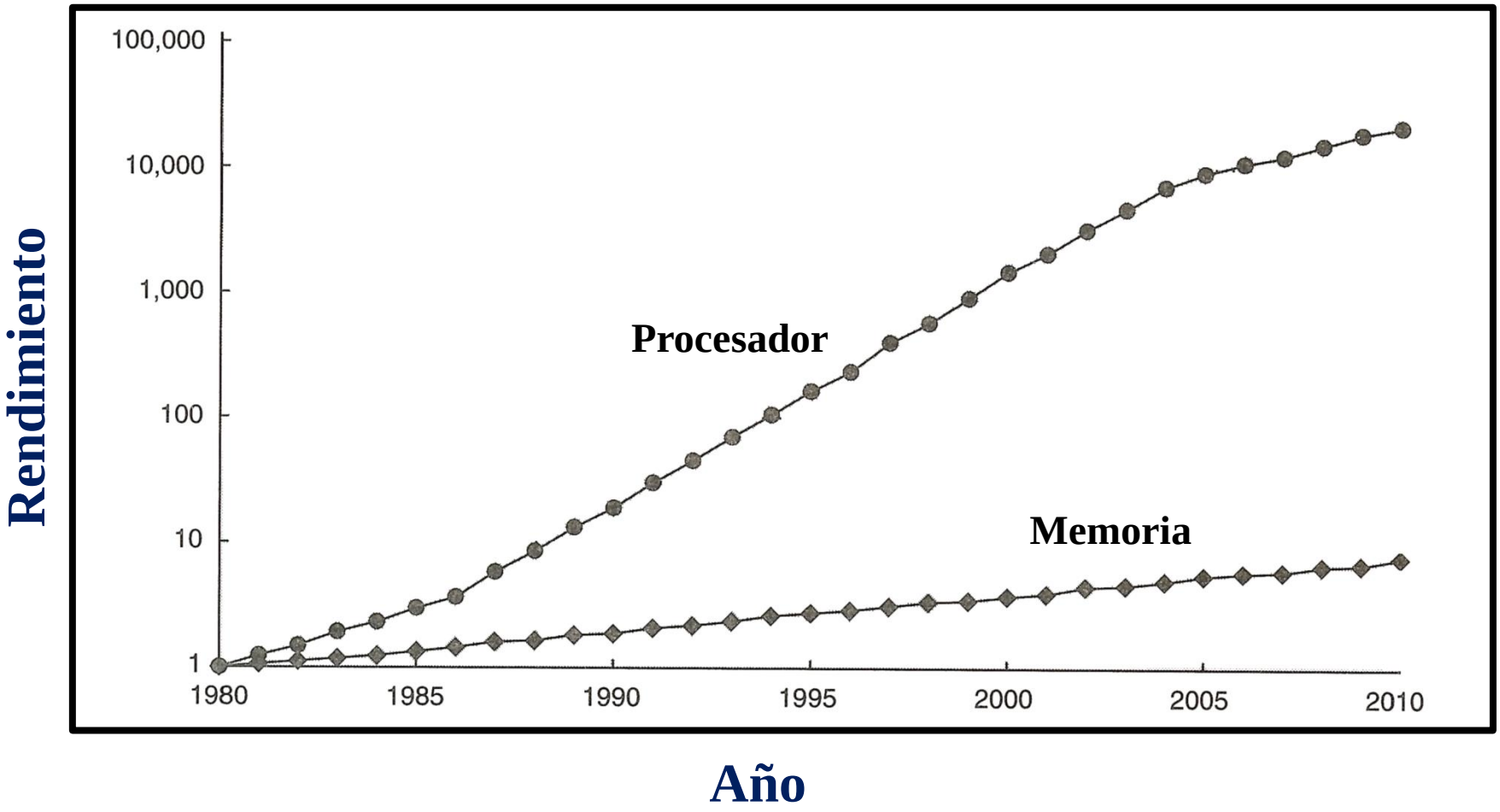
3 - Memoria virtual
- 3.1 - Objetivos, Definiciones y visión general
- 3.2 - Ciclo de vida
- 3.3 - Funcionamiento
- 3.4 - Tabla de páginas
- 3.5 – Buffer de traducción anticipada (TLB)

Lecturas Obligatorias

- W. Stallings, “Organización y Arquitectura de Computadores”:
  - Apartados 4.1, 4.2, 4.3 (Jerarquía de memoria y Memoria Caché)
  - Apartado 8.3 (Memoria Virtual)

- D.A.Patterson, J.L. Hennessy, “Estructura y Diseño de Computadores”:
  - Apartados 5.1, 5.2, 5.3 y 5.5 (Jerarquía de memoria y Memoria Caché)
  - Apartado 5.4 (Memoria Virtual)

Jerarquía de memoria. Introducción (I)



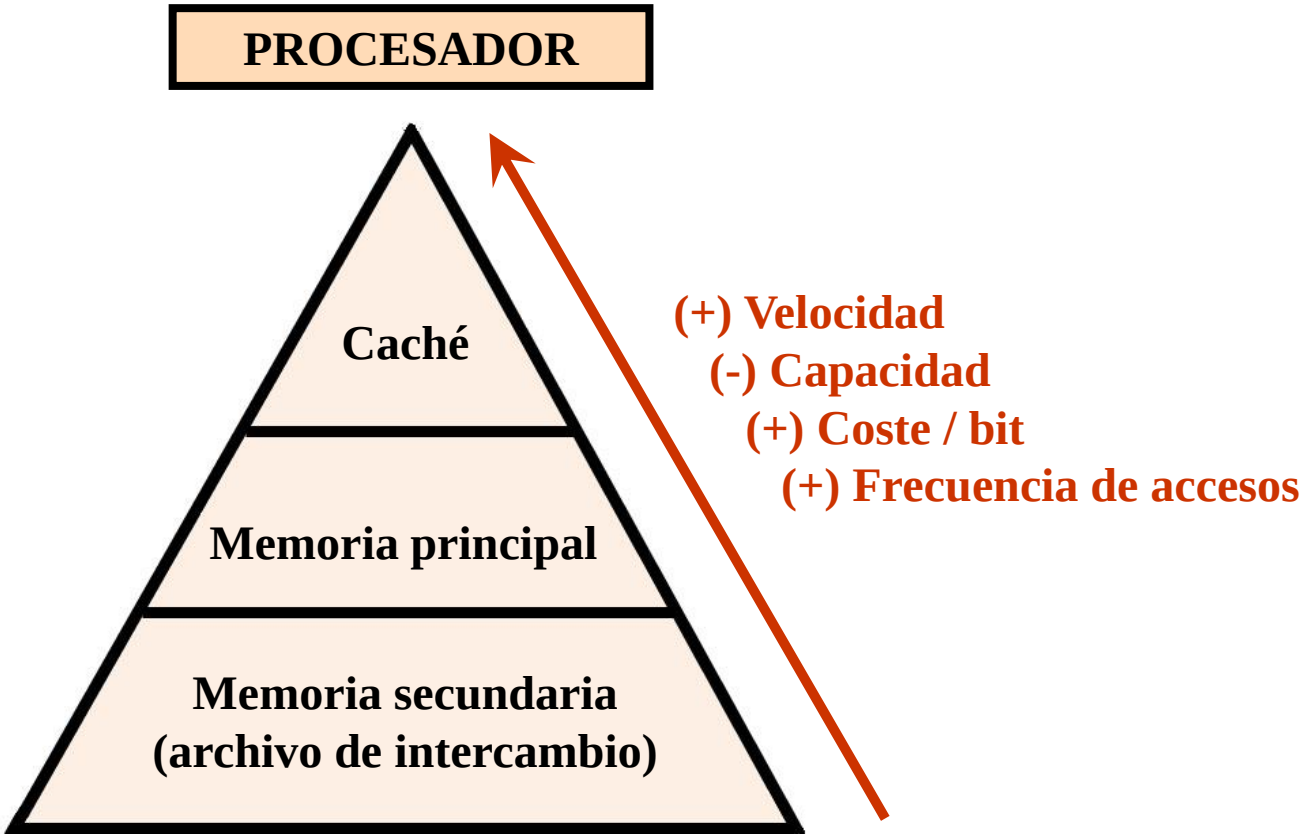
Jerarquía de memoria. Introducción (II)

- Necesidad de grandes cantidades de memoria rápida
- Utilización de un conjunto de memorias con distintas tecnologías según criterios de tamaño, velocidad y coste.
- El éxito de la jerarquía de memoria se debe gracias al principio de localidad.

> *Idealmente sería deseable una capacidad indefinidamente grande de memoria tal que cualquier particular…palabra estuviese inmediatamente disponible… Estamos… forzados a reconocer la posibilidad de construir una jerarquía de memorias, cada una de las cuales tenga mayor capacidad que la precedente pero que sea menos rápidamente accesible*
>
> A.W. Burks, H.H. Goldstine y J. von Neumann,
>
> *Discusión preliminar del diseño lógico de un instrumento de cálculo electrónico (1946)*

Jerarquía de memoria. Introducción (III)

- Una jerarquía de memoria está organizada en varios niveles, cada uno más pequeño, más rápido y más caro por *byte* que el siguiente.



Jerarquía de memoria. Introducción (IV)

Valores Típicos

Jerarquía de Memoria caché

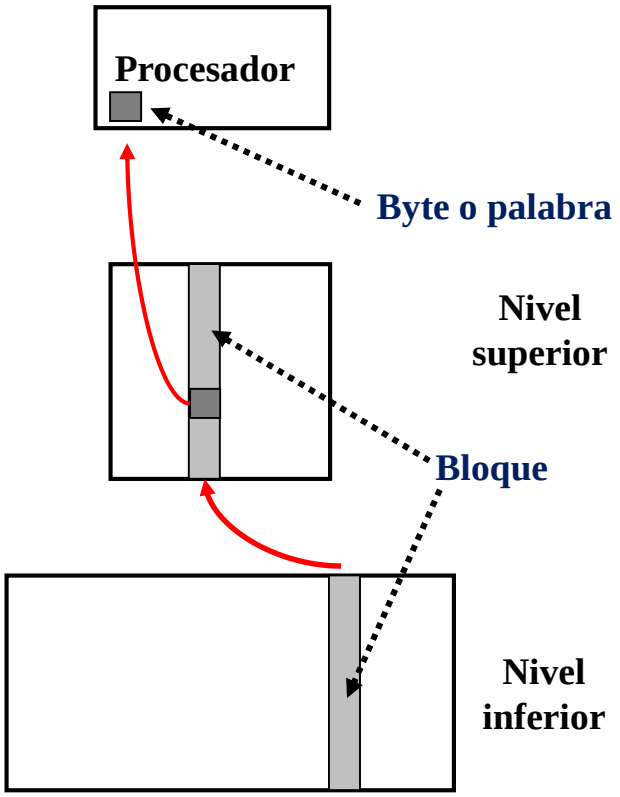
Nivel	1	2	3	4
Nombre	**Registros**	**M. Caché**	**M. Principal**	**M. Secundaria**
Tamaño	< 1KB	< 16MB	< 512GB	> 1 TB
Tecnología	Memoria CMOS de varios puertos	CMOS SRAM	CMOS DRAM	Disco Magnético
Tiempo Acceso* (ns)	0.25 a 0.5	0.5 a 25	50 a 250	5 000 000
Ancho de banda (MB/seg)	50 000 a 500 000	5 000 a 20 000	2 500 a 10 000	50 a 500
Manejado por	Compilador	Hardware	Sistema Operativo	Sistema Operativo

Jerarquía de Memoria Virtual

**** Valores típicos en el año 2006***

Jerarquía de memoria. Funcionamiento (I)

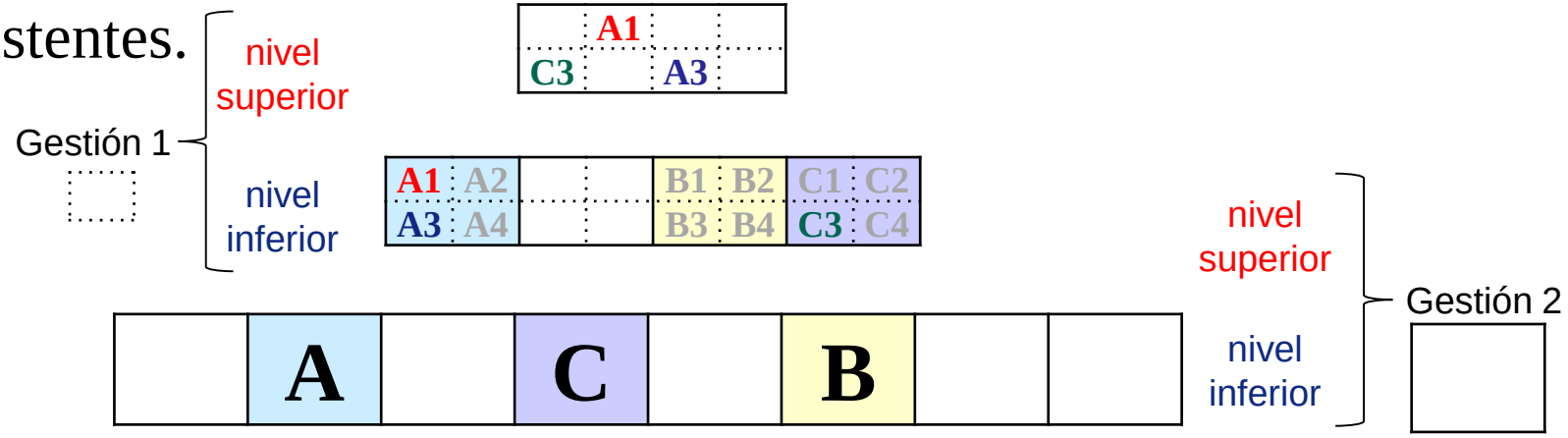
- El procesador sólo accede al nivel más alto de la jerarquía (memorias más rápida y pequeña).
- **Funcionamiento general:**
  - El procesador solicita un dato al nivel más alto de la jerarquía.
  - Si el dato se ya encuentra en el nivel superior (Acierto), simplemente se transfiere el dato al procesador.
  - Si no se encontrase en el nivel superior (Fallo), debe transferirse previamente el bloque que contiene el dato solicitado por el procesador desde el nivel inferior.
  - Si el nivel inferior tampoco posee el dato, deberá solicitar previamente el bloque a un nivel más abajo.



- **Bloque (*block*):** unidad mínima de transferencia entre niveles adyacentes.

Jerarquía de memoria. Funcionamiento (II)

- Según lo anterior:
  - Transferencias entre niveles adyacentes, distinguiéndose entre el nivel superior y el inferior.
  - Transferencias en bloques entre dos niveles adyacentes (nivel divido en bloques).
  - El tamaño de bloque es mayor (o igual) en niveles adyacentes inferiores.
- Cada nivel cumple el principio de inclusión: todo dato de un nivel se encuentran incluido en todos los niveles inferiores a éste.
- Además cumple el principio de coherencia: Las copias de la misma información existentes en los distintos niveles deben ser consistentes.



Jerarquía de memoria. Funcionamiento (III)

- Principio de localidad de las referencias:
  - Los procesadores, al ejecutar un programa, suelen acceder a memoria a partir de ciertos patrones por lo que favorecen una parte de su espacio de direcciones.
  - El principio de localidad referencial posee una doble dimensión:
    - **Localidad temporal:** *Si se referencia un elemento, éste tenderá a ser referenciado pronto*
      - Por ejemplo, en bucles.
    - **Localidad espacial:** *Si se referencia un elemento, los elementos cercanos a él tenderán a ser referenciado pronto*
      - *Por ejemplo, en Tablas y secuenciamiento de instrucciones.*

Jerarquía de memoria. Rendimiento (I)

- **Acierto** (*hit) o* **Fallo***(miss*): se produce acierto cuando el gestor de dos niveles adyacentes encuentra el dato solicitado en el nivel superior y se produce fallo si el dato solicitado **no** se encuentra en el nivel superior (aunque podría no estar tampoco en el nivel inferior).
- Para medir el rendimiento de la jerarquía suele emplearse el **tiempo de acceso medio a memoria**.

Tiempo de acceso medio a memoria = Tiempo de acierto +
+ Frecuencia de fallos * Penalización por fallo

  - **Frecuencia de fallos** (*miss rate)*: relación de fallos respecto al total de accesos realizados.

ff = fallos / accesos

  - **Frecuencia de aciertos** (*hit rate)*: tasa de aciertos respecto al total de accesos.

fa = aciertos / accesos

fa + ff = 1

Jerarquía de memoria. Rendimiento (II)

- **Tiempo de acierto** (*hit time*): tiempo necesario para obtener el dato solicitado cuando se produce un acierto (dependerá de la tecnología del nivel superior).
- **Penalización por fallo** (*miss penalty*): en caso de fallo, tiempo adicional al tiempo de acierto para transferir el bloque del nivel inferior al superior.
  - La penalización por fallo se descompone en:
    - **Tiempo de acceso** (*access time*) al nivel inferior: tiempo para acceder a la primer byte de un bloque en un fallo
    - **Tiempo de transferencia** (*transfer time*): tiempo adicional para transferir las restantes bytes del bloque (la inversa del ancho de banda).
- ¿Cuál de las siguientes alternativas es más rápida?

Opción	Tamaño Bloque	T_{acceso} al Nivel Superior	ff o $Miss_{rate}$ del Nivel Superior	t_{acceso} al Nivel Inferior	t_{tranf} del Nivel Inferior
A	8 bytes	2ns	8%	30 ns	10 ns/byte
B	16 bytes	1ns	12%	25 ns	4 ns/byte

$$tamm_A = 2 + 0,08(30 + 10 * 8) = 10,80ns$$

$$tamm_B = 1 + 0,12(25 + 4 * 16) = 11,68ns$$

El rendimiento de la opción A es mayor.

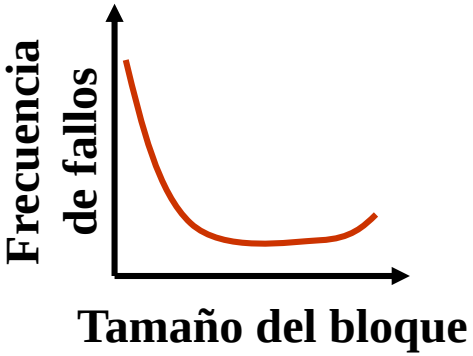
¿Y si hubiese otro nivel más abajo?

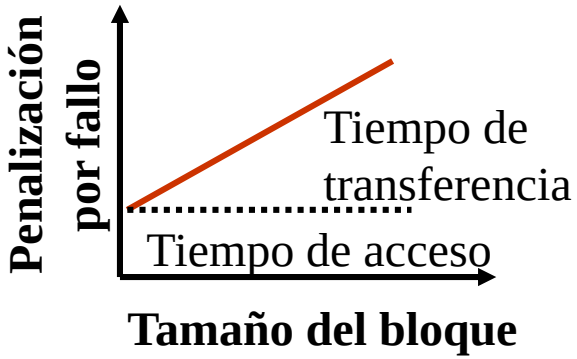
Jerarquía de memoria. Rendimiento (III)

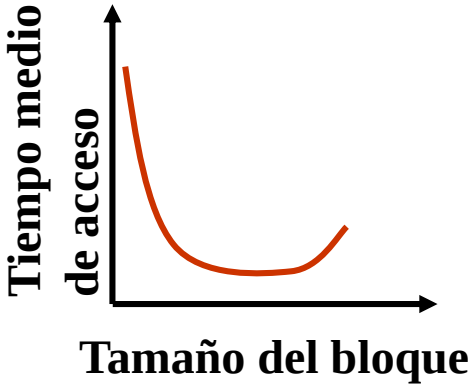
- En el diseño de la jerarquía se tiene en cuenta la influencia de los parámetros sobre el rendimiento.
- **Ejemplo**: ¿Cómo influye el tamaño de bloque sobre rendimiento?

  - Penalización por fallo
  - Frecuencia de fallos

  → Tiempo de acceso medio a memoria







Jerarquía de memoria. Clasificación

- **Ubicación de bloque**

  ¿Dónde puede ubicarse un bloque en el nivel superior?

- **Identificación de bloque**

  ¿Cómo se encuentra un bloque si está en el nivel superior?

- **Sustitución de bloque**

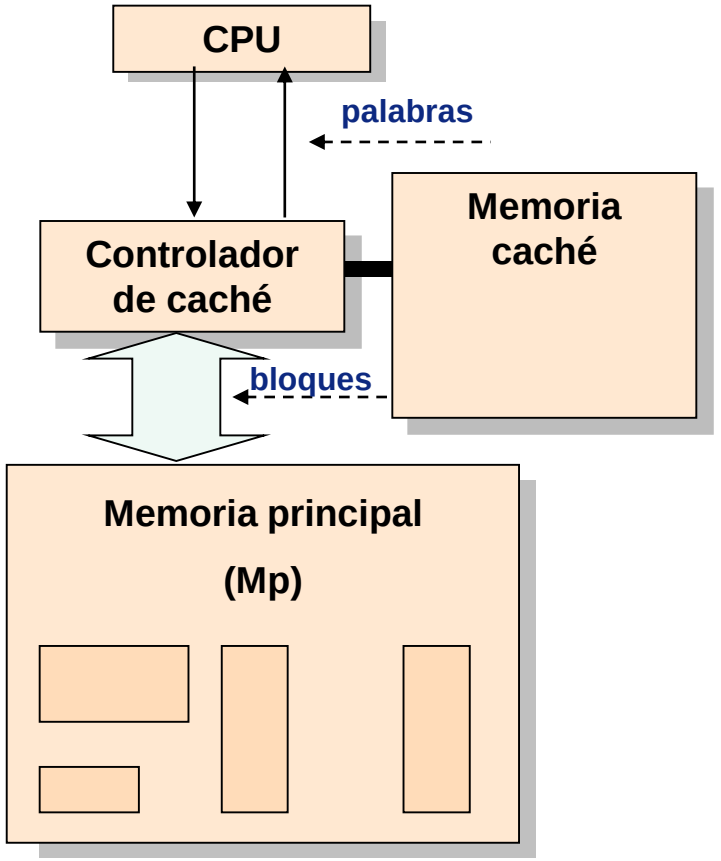
  ¿Qué bloque del nivel superior debe reemplazarse en caso de fallo?

- **Estrategia de escritura**

  ¿Cómo se realiza la escritura para garantizar la coherencia?

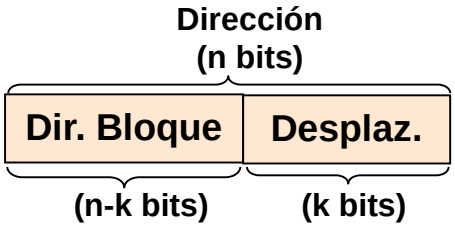
Memorias caché. Introducción (I)

- Representa el nivel de jerarquía de memoria entre la CPU y la memoria principal.
- Se le pueden aplicar todos los aspectos vistos para la jerarquía de memoria (conceptos, funcionamiento, …)
- **Clasificación de las memorias caché:**
  - Según la organización física
    - Internas o externas a la CPU.
    - En serie o en paralelo.
  - Según la información que contiene
    - Unificadas o separadas.
  - Según su estructura/funcionamiento
    - Ubicación/Localización
    - Sustitución
    - Escritura

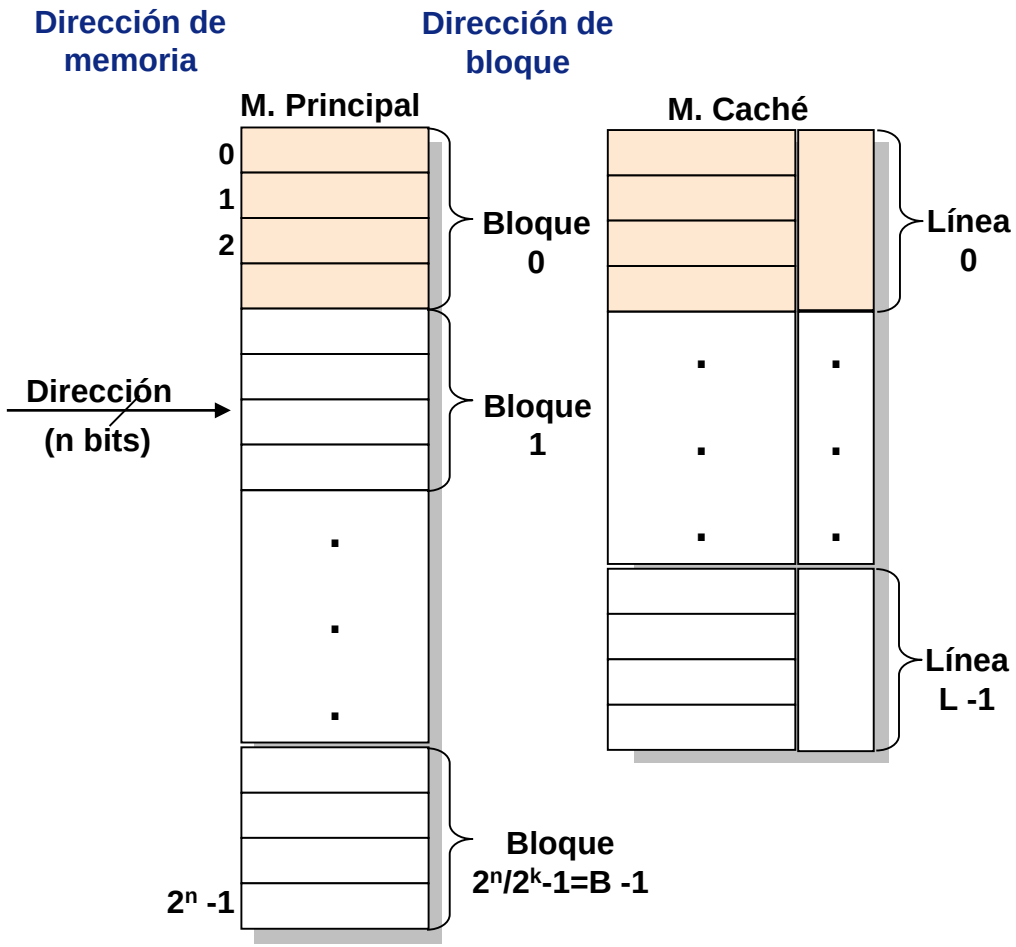


Memorias caché. Estructura básica

- **Mp (memoria principal):**
  - Formada por **2^n palabras** direccionables
  - Organizada en **B bloques** de tamaño fijo de **2^K (bytes por bloque).**
  - Campos de una dirección física:

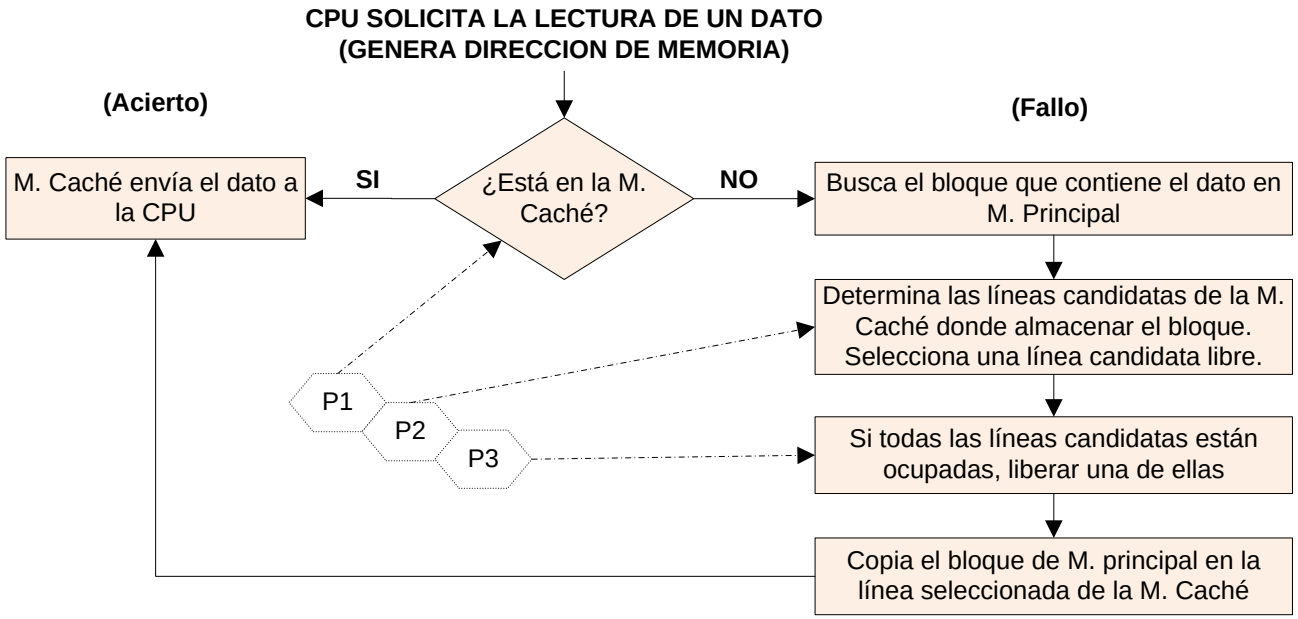


- **Mc (memoria cache):**
  - formada por **L líneas** de **2^K bytes** donde (L << B).
  - **Contenido de cada línea de caché:**
    - Información de bloque
    - Datos adicionales
      - Bit de línea válida.
      - Info. identificación de bloque
      - Otros... (permisos R/W)



Memorias caché. Funcionamiento general

Lectura



- **P1: Identificación de bloque:** ¿Cómo encontramos un bloque que está en el nivel superior?
- **P2: Ubicación de bloque** ¿Dónde puede ubicarse un bloque en el nivel superior?
- **P3: Reemplazo de bloque:** ¿En caso de fallo qué bloque debemos reemplazar?

Política de ubicación de bloque (I)

- *¿Dónde puede ubicarse un bloque en una caché?*
  - **Cachés de mapeado o correspondencia directa:** cada bloque de memoria principal sólo puede alojarse en una línea específica de la caché y que viene determinada por:

(dirección del bloque en memoria) MOD (número de bloques en caché)

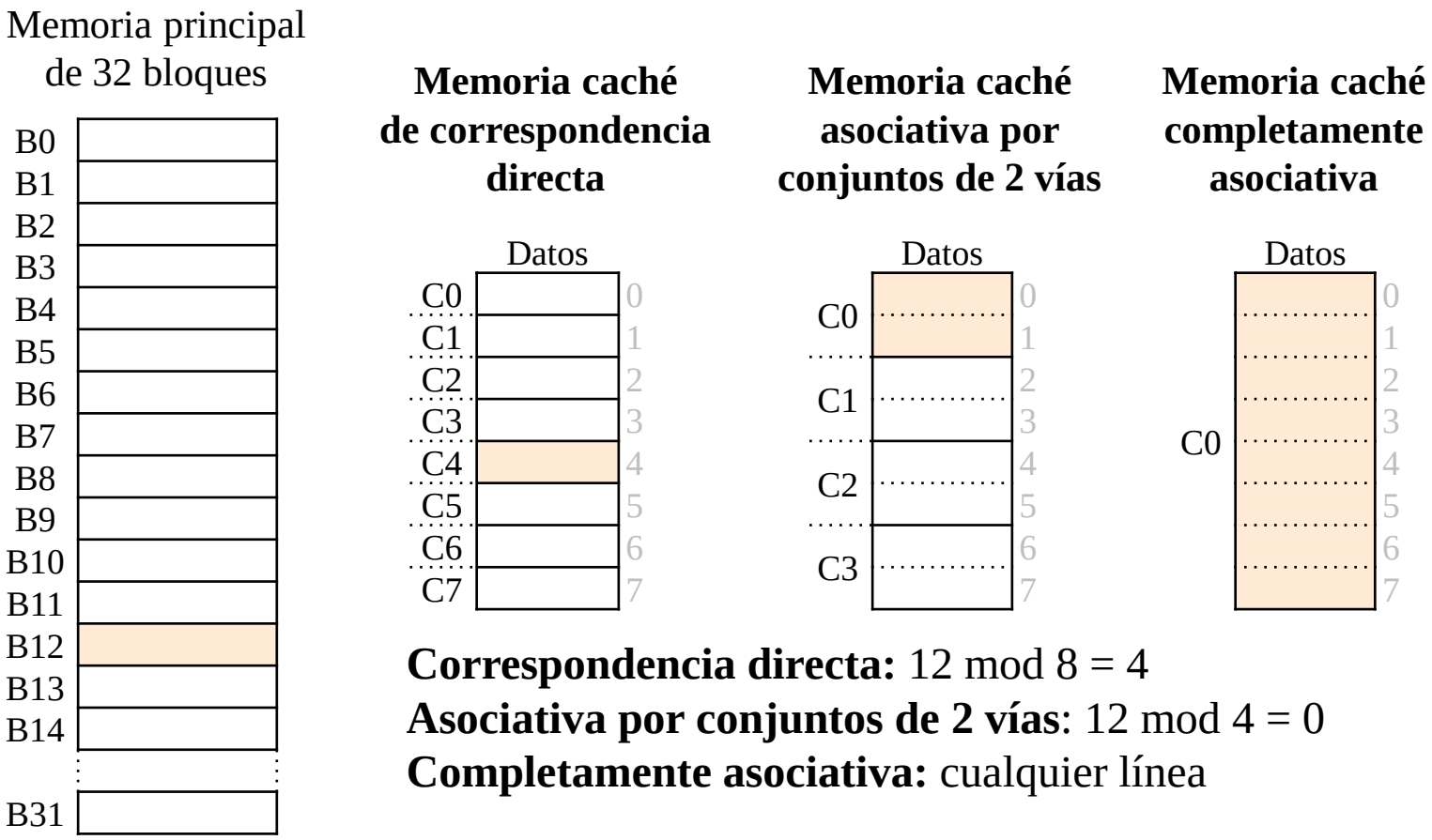
  - **Cachés totalmente o completamente asociativas:** cada bloque se puede ubicarse en cualquier línea de la caché.

  - **Cachés asociativas por conjuntos (asociativas por conjuntos de *n* vías):** La líneas de la caché están agrupadas en conjuntos, de forma que cada bloque de memoria principal puede alojarse en cualquier línea dentro de un conjunto específico de la caché. Dicho conjunto viene indicado por:

(dirección del bloque en memoria) MOD (número de conjuntos en caché)

Política de ubicación de bloque (II)

Ej. ¿En qué líneas de caché puede ubicarse el bloque de memoria 12?



Correspondencia directa: 12 mod 8 = 4
Asociativa por conjuntos de 2 vías: 12 mod 4 = 0
Completamente asociativa: cualquier línea

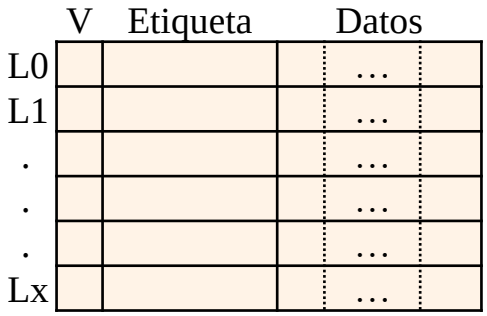
Política de identificación de bloque (I)

- ***¿Cómo se encuentra un bloque si está en el nivel superior?***
  - La dirección de memoria se descompone en varios campos:
    - **Dirección de bloque**:
      - **Índice** (*index*): selecciona el conjunto (en el caso de las asociativas por conjunto) o bloque (en las de mapeado directo). No existe para las completamente asociativas
      - **Etiqueta** (*tag*): se compara con la etiqueta que posee las líneas de caché previamente seleccionadas a fin de determinar la línea que contiene la información solicitada.
    - **Desplazamiento de bloque** (*block offset*): selecciona el dato solicitado dentro de la línea de caché
- ***¿Cómo se sabe que un bloque contiene información válida?***
  - Mediante un bit de válido **(***valid bit***)** por línea que indica si la línea contiene información de algún bloque de memoria.

Política de identificación de bloque (II)

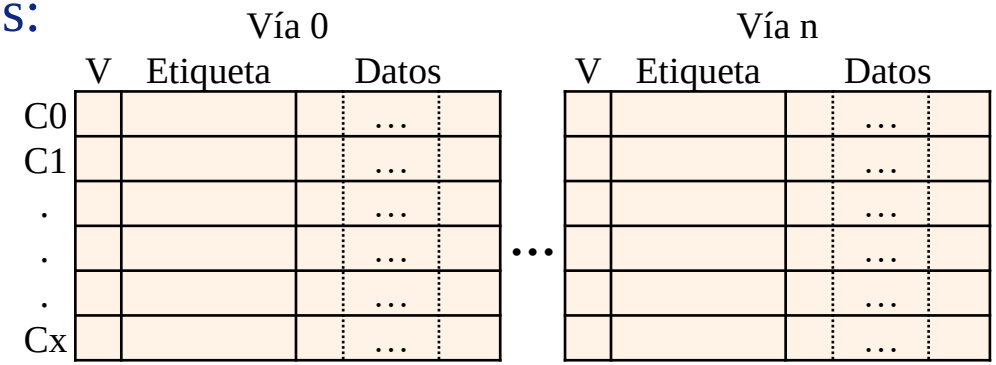
- Descomposición de direcciones de memoria y estructura de la caché
  - Caché de mapeado directo:

Dirección de Bloque		Desplaz.
Etiqueta	Índice de Línea	Desplaz.



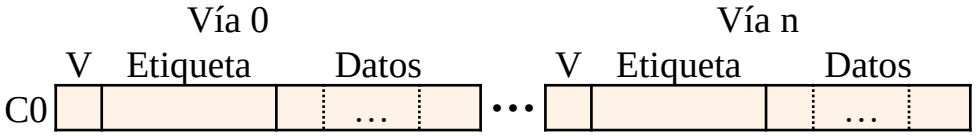
  - Caché asociativa por conjuntos:

Dirección de Bloque		Desplaz.
Etiqueta	Índice de conjuntos	Desplaz.



  - Caché completamente asociativa:

Dirección de Bloque	Desplaz.
Etiqueta	Desplaz.



Política de identificación de bloque (III)

- Ejemplo descomposición de una dirección:
  - Procesador con dirección de 24 bits
  - Tamaño de la caché: 8KB
  - Tamaño de bloque: 8 Bytes

- **Caché de mapeado directo**

Etiqueta: 11bits	Ind. Bloque: 10bits	Desplazamiento: 3bits

- **Caché asociativa por conjuntos de 4 vías**

Etiqueta: 13bits	Ind. Conjunto: 8bits	Desplazamiento: 3bits

- **Caché completamente asociativa**

Etiqueta: 21bits	Desplazamiento: 3bits

Política de identificación de bloque (IV)

- **Ejemplo:** Para acceder a la dirección 50 en una jerarquía de memoria con una memoria principal de 128 bytes, memoria caché de 32 bytes y un tamaño de bloque de 4 bytes.
  - **Dirección en Memoria principal:**

Dirección 50 (0110010) =>

Bloque	Desp
01100	10

Bloque de M. principal: 12
Desplazamiento: 2

  - **Acceso a los distintos tipos de cachés sería:**

Memoria caché de correspondencia directa

Etiq	Ind	Desp
01	100	10

	V	Etiq	Dato	
L0	0			0
L1	1	11	? ? ? ?	1
L2	0			2
L3	0			3
L4	1	01	X	4
L5	0			5
L6	0			6
L7	0			7

Memoria caché asociativa por conjuntos de 2 vías

Etiq	Ind	Desp
011	00	10

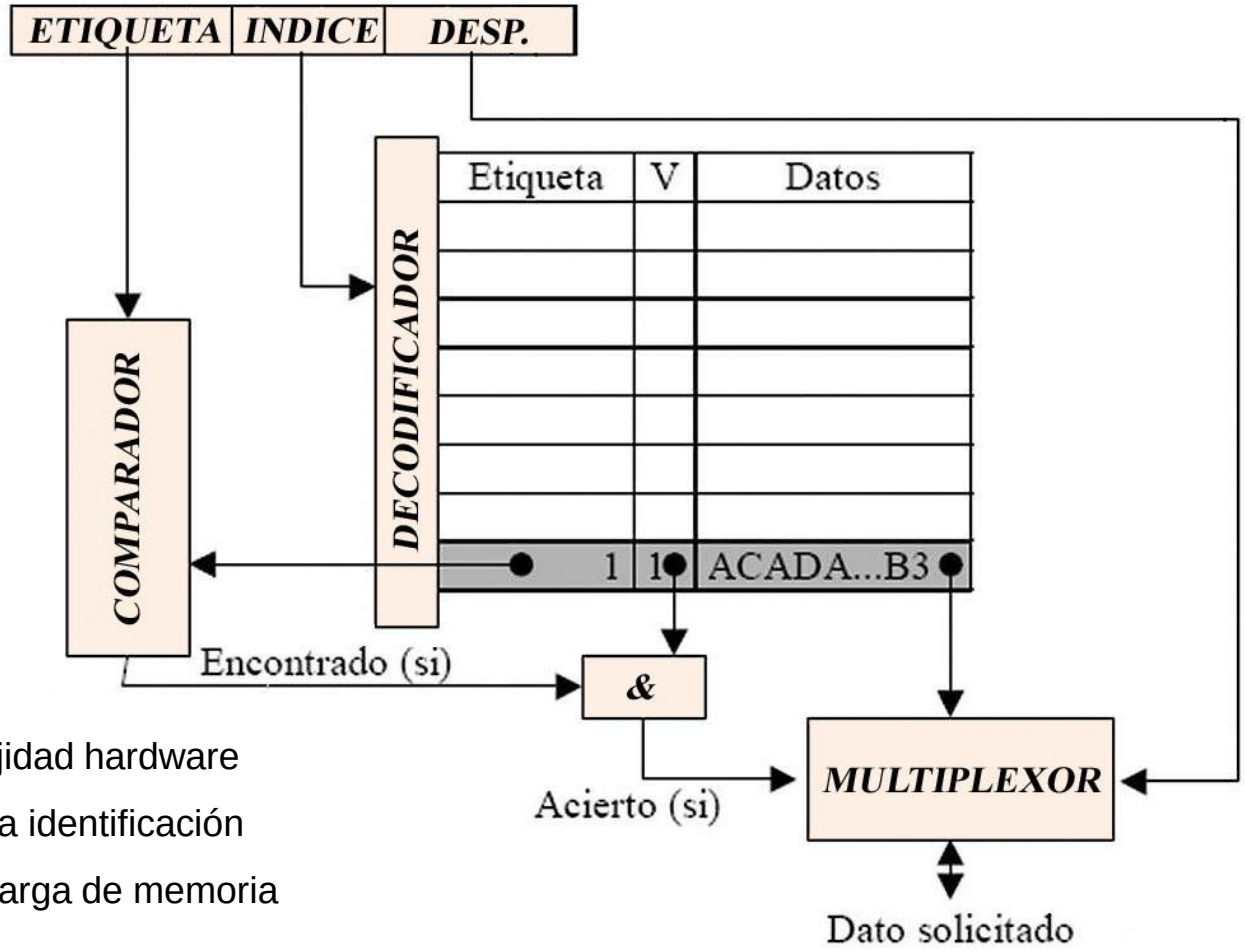
	V	Etiq	Dato	
C0	0			0
	1	011	X	1
C1	0			2
	0			3
C2	0			4
	0			5
C3	0			6
	0			7

Memoria caché completamente asociativa

Etiq	Desp
01100	10

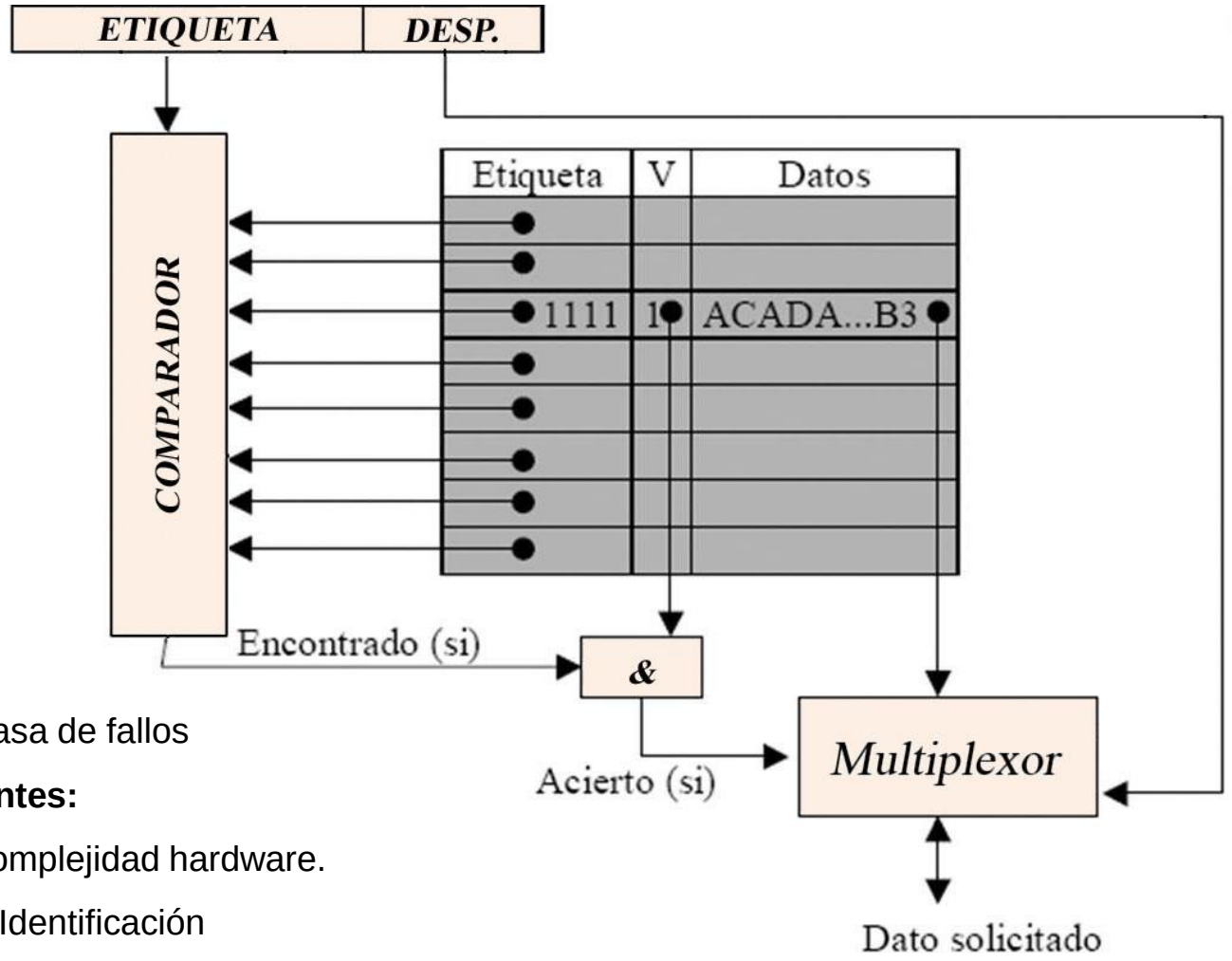
	V	Etiq	Dato	
C0	0			0
	0			1
	1	11011	? ? ? ?	2
	0			3
	0			4
	0			5
	1	01100	X	6
	0			7

Implementación: Mapeado directo



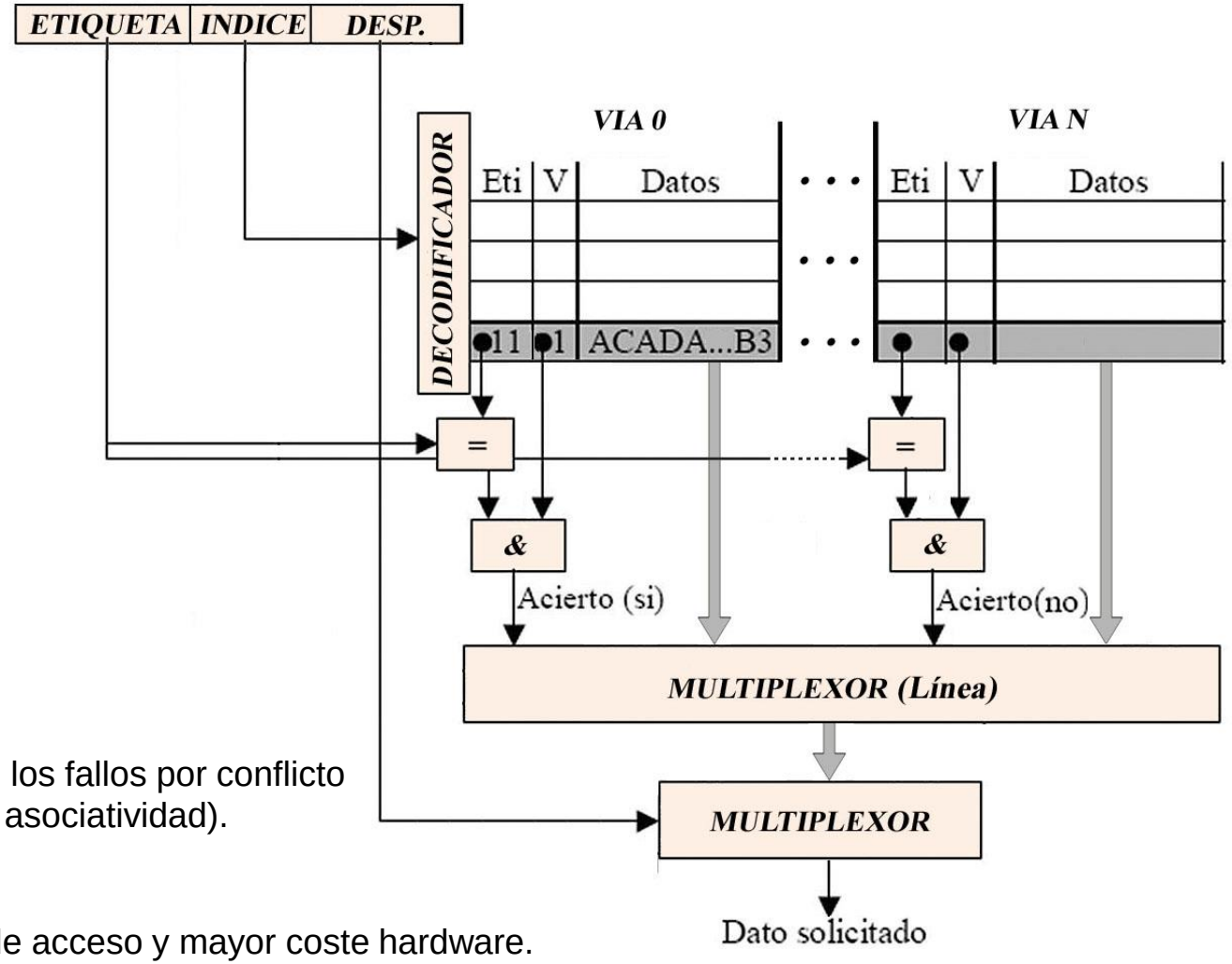
- **Ventajas:**
  - Baja complejidad hardware
  - Rapidez en la identificación
  - Poca sobrecarga de memoria
- **Inconvenientes:**
  - Alta tasa de fallos si varios bloques compiten por la misma línea.

Implementación: Totalmente Asociativas



- **Ventajas:**
  - Baja tasa de fallos
- **Inconvenientes:**
  - Alta complejidad hardware.
  - Lenta Identificación
  - Sobrecarga de memoria debido al tamaño de las etiquetas

Implementación: Asociativas por conjuntos



- **Ventajas:**
  - Disminución de los fallos por conflicto (al aumentar la asociatividad).
- **Inconvenientes:**
  - Mayor tiempo de acceso y mayor coste hardware. (al aumentar la asociatividad)

Políticas de reemplazo de bloque

- ***¿Qué bloque debe reemplazarse en caso de fallo?*** Se pueden seguir diferentes estrategias:
  - **Aleatoria** (*Random*): sustituye un bloque al azar. Muy utilizado.
  - **LRU** (*Least Recently Used*): sustituye el bloque que más tiempo ha estado sin ser referenciado. Es el más sofisticado aunque de mayor coste.
    - Ejemplo: suponiendo una secuencia de accesos a los bloques A,B,C y D

Traza de accesos		D	C	B	A	A	C	D	B	D	A
Bloque LRU	A	A	A	A	D	D	D	B	A	A	C

  - **FIFO** (*First Input First Output*): reemplaza el bloque que más tiempo ha estado en la caché (es una aproximación a la LRU)
  - **LFU** (Least Frecuently Used): sustituye el bloque referenciado menos veces.
- Ejemplo. Tras la traza de accesos:

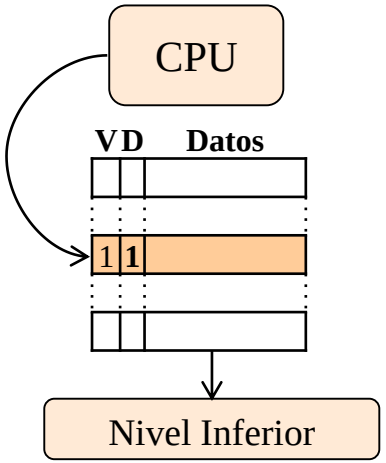
A	B	B	C	C	D	A

Aleatoria	FIFO	LRU	LFU
?	A	B	D

- Se tienen en cuenta criterios de coste y eficiencia
- Suele utilizarse LRU (por su eficacia) y Aleatorio (por su sencillez)

Políticas de escritura: Acierto en escritura (I)

- ***¿Qué ocurre en un acierto en escritura?*** Se distinguen 2 estrategias:
  - Escritura directa o Write through:
  - Postescritura o Copy Back .



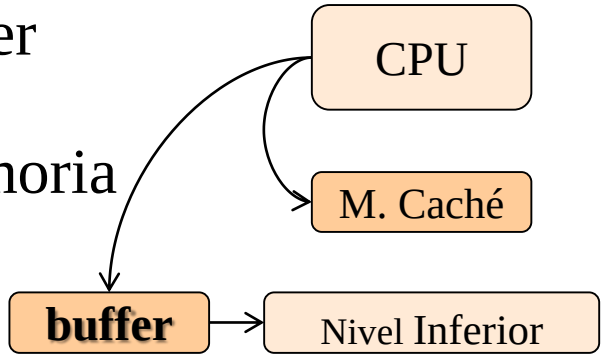
A) Post-escritura (*Copy-Back*)

- En cada acierto de escritura, la información sólo actualiza en la caché. El bloque modificado de la caché se actualiza en la memoria de nivel inferior **sólo** cuando es reemplazado.
- Los bloques de las cachés de post-escritura se denominan sucios o modificados cuando la información de la caché difiere de la memoria de nivel inferior.
- Para reducir la frecuencia de post-escrituras en el reemplazo se usa el bit de sucio (*dirty*). Si el bloque está limpio no se escribe en el nivel inferior.
- **Ventajas:**
  - Las escrituras se realizan a la velocidad de la memoria caché
  - Varias escrituras en un bloque sólo requiere una escritura en el nivel inferior
  - Como no todas las escrituras van a memoria, la post-escritura utiliza menos ancho de banda (multiprocesadores) en las escrituras

Políticas de escritura: Acierto en escritura (II)

B) Escritura Directa (*Write Through*)

- La información se escribe tanto en el bloque de la caché como en el bloque de la memoria de nivel inferior.
- La CPU debe esperar la finalización de cada escritura en memoria antes de proceder con la siguiente operación.
  - Para evitar la espera, se utiliza un buffer de escritura que permite al procesador continuar mientras se actualiza la memoria



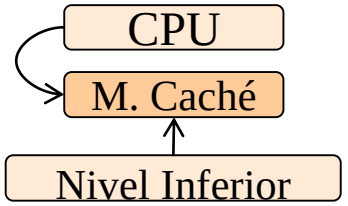
- **Ventajas:**
  - Los fallos de lectura no ocasionan escrituras en el nivel inferior
  - Son más fáciles de implementar
  - La memoria principal tiene la copia más reciente de los datos (coherencia de caché y multiprocesadores)

Políticas de escritura: Fallo en escritura

- ***¿Qué ocurre en un fallo en escritura?*** Hay 2 opciones

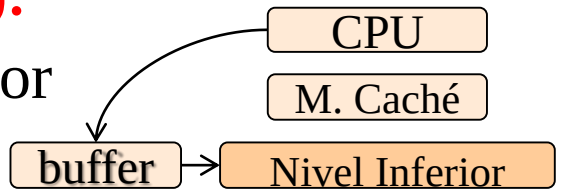
A) Ubicar en escritura (*Write allocation*):

- El bloque se carga en la caché y a continuación se escribe sobre él (similar a un fallo en lectura)



B) No ubicar en escritura (*No write allocation*):

- El bloque se modifica en el nivel inferior y no se carga en la caché



- Aunque cualquier política de fallo de escritura puede utilizarse con la escritura directa o con la post-escritura, sin embargo:
  - Las cachés de post-escritura realizan ubicación en escritura (CB-WA)
  - Las caches de escritura directa no realizan ubicación en escritura (WT-NWA)

Cachés unificadas vs. Cachés separadas (I)

- **Cachés unificadas o mixtas** (*unified or mixed*): Contienen tanto datos como instrucciones
- **Cachés separadas** (*separated*): Existe una caché para datos y otra para instrucciones
  - **Ventajas**
    - No hay competencia entre la unidad búsqueda y decodificación de instrucciones y la unidad de ejecución
    - Duplicación del ancho de banda entre caché y CPU (puertos separados)
    - Mayor rendimiento: los parámetros de diseño (capacidad, tamaños de bloque, asociatividad, etc.) se ajustan a las necesidades de cada caché
  - **Inconvenientes**
    - En general la tasa de fallos global es algo mayor (próxima transparencia)
    - No se equilibra la carga de trabajo de forma automática
      - Las cachés de instrucciones tienen menor frecuencia de fallos que las de datos (mayor localidad); además, la separación de instrucciones y datos elimina fallos debidos a conflictos; sin embargo, al dividir también se fija el espacio dedicado a cada caché

Cachés unificadas vs. Cachés separadas (II)

- **Ejemplo.** Suponiendo un 53% de accesos a instrucción
  a) Caché unificada de 32Kb. Penalización: 50 ciclos; Tiempo de acierto para instrucciones: 1 ciclo; para datos: 2 ciclos (un solo puerto)
  b) Cachés separadas de 16KB. Penalización: 50 ciclos y tiempo de acierto de 1 ciclo

Tamaño	Sólo Instrucciones	Sólo Datos	Unificada
4 KB	8,6 %	8,7 %	11,2 %
8 KB	5,8 %	6,8 %	8,3 %
16 KB	3,6 %	5,3 %	4,9 %
32 KB	2,2 %	2,0 %	4,3 %
64 KB	1,4 %	2,8 %	2,9 %

Frecuencia de fallos:

a) ff = **4.3%**

b) ff = 53%×3.6% + 47%×5.3% = **4.4%**

Tiempo de acceso medio a memoria

a) $T_{\text{acceso medio}}$ =53%*(1+4.3%*50)+47%*(2+4.3%*50) ≈ 3.6 ciclos

b) $T_{\text{acceso medio}}$=53%*(1+3.6%*50)+47%*(1+5.3%*50) ≈ 3.2 ciclos

Cómo mejorar el rendimiento de las cachés

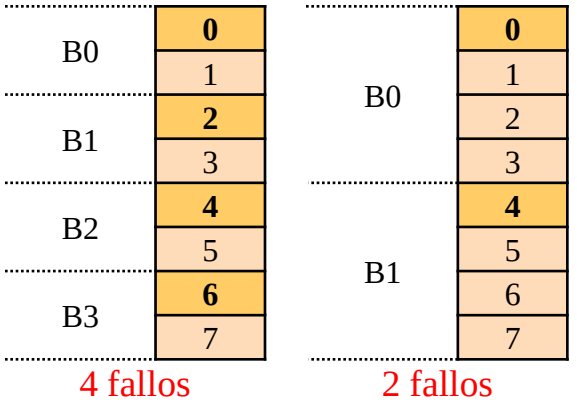
- Para mejorar el rendimiento se debe **reducir el tiempo medio de acceso a memoria**:

> ***↓Tiempo de acceso medio a memoria = ↓Tiempo de acierto + ↓Frecuencia de fallos * ↓Penalización por fallo***

- Existen tres formas de reducir el tiempo medio de acceso a memoria:
  - Reducir los fallos de la caché (*miss rate*)
  - Reducir las penalizaciones por fallo (*miss penalty*)
  - Reducir el tiempo de acceso en caso de acierto (*hit time*)

Reducir la frecuencia de fallos: Tipos de fallos (I)

- **Fallos Forzosos (*Compulsory miss*)**
  - Se produce durante el primer acceso a un bloque de memoria. Este fallo resulta inevitable ya que si es la primera vez que accedemos al bloque, este no puede encontrarse en la caché al no haber sido referenciado anteriormente.
  - Estos fallos son aquellos que ocurren incluso si tuviéramos una caché infinita y totalmente asociativa.
  - El número de fallos forzosos será mayor con bloques pequeños debido al principio de localidad espacial.
  - También se conocen como *fallos de arranque en frío* o *de primera referencia*.
  - Se reducen aumentando el tamaño de bloque.



Fallos generados al recorrer un vector de 8 elementos con distinto tamaño de bloque

Reducir la frecuencia de fallos: Tipos de fallos (II)

- **Fallos por Capacidad (*Capacity miss*)**

– Ocurre cuando la caché no puede contener todos los bloques necesarios durante la ejecución de un programa.

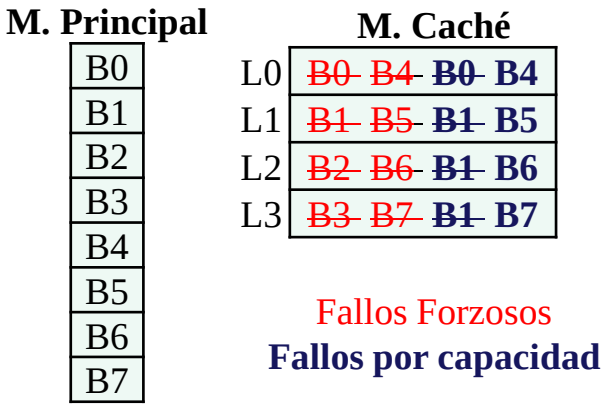
– En una caché totalmente asociativa que esté llena, los bloques en caché son reemplazados por los nuevos bloques que se carguen de memoria. Cuando los bloques reemplazados vuelven a ser referenciados se produce un fallo por capacidad.

– Estos fallos se producen en cualquier tipo de caché y pueden reducirse aumentando el tamaño de la caché

– **Ejemplo:** Al recorrer 2 veces un vector cuyo tamaño abarca 8 bloques disponiendo de una caché de 4 líneas, ocurre:

  - Primer recorrido: La caché se llena 2 veces produciéndose 8 fallos forzosos.
  - Segundo recorrido: La caché vuelve a llenarse 2 veces produciéndose 8 fallos por capacidad.

  Si aumentamos la caché del ejemplo a 8 líneas los fallos por capacidad desaparecerían.



Reducir la frecuencia de fallos: Tipos de fallos (III)

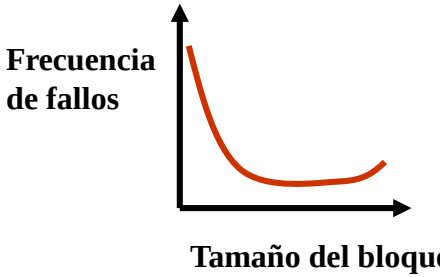
- **Fallos por conflicto (*Compulsory miss*)** o por colisión
  - Sólo ocurren en cachés de correspondencia directa o asociativas por conjuntos
  - Se producen cuando el número de bloques de memoria referenciados a los que les corresponde un mismo conjunto o línea de caché es mayor que la asociatividad (número de líneas por conjunto) de la caché
  - Provoca reemplazos en un conjunto o línea aunque el resto de la caché permanezca vacía
  - Estos fallos podrían reducirse aumentando la asociatividad
  - **Ejemplo:** Acceso a cachés de 4 líneas con diferente asociatividad

Reducir la frecuencia de fallos: Influencia de fallos

Cache size (KB)	Degree associative	Total miss rate	Compulsory		Capacity		Conflict	
4	1-way	0.098	0.0001	0.1%	0.070	72%	0.027	28%
4	2-way	0.076	0.0001	0.1%	0.070	93%	0.005	7%
4	4-way	0.071	0.0001	0.1%	0.070	99%	0.001	1%
4	8-way	0.071	0.0001	0.1%	0.070	100%	0.000	0%
8	1-way	0.068	0.0001	0.1%	0.044	65%	0.024	35%
8	2-way	0.049	0.0001	0.1%	0.044	90%	0.005	10%
8	4-way	0.044	0.0001	0.1%	0.044	99%	0.000	1%
8	8-way	0.044	0.0001	0.1%	0.044	100%	0.000	0%
16	1-way	0.049	0.0001	0.1%	0.040	82%	0.009	17%
16	2-way	0.041	0.0001	0.2%	0.040	98%	0.001	2%
16	4-way	0.041	0.0001	0.2%	0.040	99%	0.000	0%
16	8-way	0.041	0.0001	0.2%	0.040	100%	0.000	0%
32	1-way	0.042	0.0001	0.2%	0.037	89%	0.005	11%
32	2-way	0.038	0.0001	0.2%	0.037	99%	0.000	0%
32	4-way	0.037	0.0001	0.2%	0.037	100%	0.000	0%
32	8-way	0.037	0.0001	0.2%	0.037	100%	0.000	0%
64	1-way	0.037	0.0001	0.2%	0.028	77%	0.008	23%
64	2-way	0.031	0.0001	0.2%	0.028	91%	0.003	9%
64	4-way	0.030	0.0001	0.2%	0.028	95%	0.001	4%
64	8-way	0.029	0.0001	0.2%	0.028	97%	0.001	2%
128	1-way	0.021	0.0001	0.3%	0.019	91%	0.002	8%
128	2-way	0.019	0.0001	0.3%	0.019	100%	0.000	0%
128	4-way	0.019	0.0001	0.3%	0.019	100%	0.000	0%
128	8-way	0.019	0.0001	0.3%	0.019	100%	0.000	0%
256	1-way	0.013	0.0001	0.5%	0.012	94%	0.001	6%
256	2-way	0.012	0.0001	0.5%	0.012	99%	0.000	0%
256	4-way	0.012	0.0001	0.5%	0.012	99%	0.000	0%
256	8-way	0.012	0.0001	0.5%	0.012	99%	0.000	0%
512	1-way	0.008	0.0001	0.8%	0.005	66%	0.003	33%
512	2-way	0.007	0.0001	0.9%	0.005	71%	0.002	28%
512	4-way	0.006	0.0001	1.1%	0.005	91%	0.000	8%
512	8-way	0.006	0.0001	1.1%	0.005	95%	0.000	4%

Frecuencia total de fallos para cada tamaño de caché:

- **Fallo forzoso:** Disminuye al aumentar el tamaño de bloque



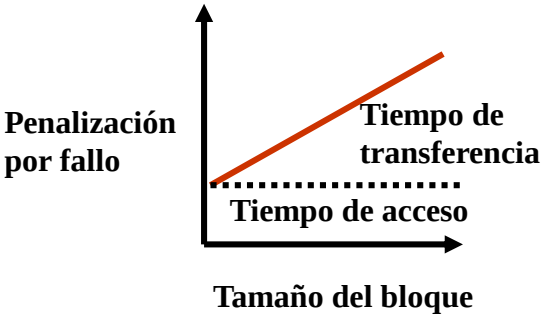
- **Fallo de capacidad:** Decrece al aumentar tamaño de la caché.

- **Fallo de conflicto:** Decrece al aumentar la asociatividad.

Reducir la penalización por fallos

↓Penalización por fallo = ↓Tiempo de acceso + ↓Tamaño del bloque * ↓Tiempo de Transferencia por byte

- El **tiempo de acceso** y **el tiempo de transferencia por byte** dependen de la tecnología y del tamaño de la memoria del nivel inferior.



- El único parámetro que podemos evaluar de la caché es el tamaño de bloque.
  - Al reducir el tamaño de bloque, disminuye el tiempo de transferencia del bloque reduciéndose así la penalización por fallo.
  - **Inconveniente:** Como se ha visto anteriormente, al reducir el tamaño de bloque, aumentan los fallos forzosos.

Reducir el tiempo de acierto

- El tiempo de acierto está condicionado por el tiempo de acceso a la caché, que depende a su vez de:
  - **El tamaño de la caché**: memorias más pequeñas poseen un menor tiempo de acceso pues utilizan un decodificador menor para seleccionar la línea.
  - **La asociatividad**: las memorias con un menor número de vías requieren menor tiempo pues sus comparadores son más simples.
    - **Inconveniente:** Como se indicó anteriormente, reducir el grado de asociatividad provoca el aumento de los fallos por conflicto.
    - **¿Compensa disminuir la asociatividad?**

	Associativity			
Cache size (KB)	One-way	Two-way	Four-way	Eight-way
4	3.44	3.25	3.22	3.28
8	2.69	2.58	2.55	2.62
16	2.23	2.40	2.46	2.53
32	2.06	2.30	2.37	2.45
64	1.92	2.14	2.18	2.25
128	1.52	1.84	1.92	2.00
256	1.32	1.66	1.74	1.82
512	1.20	1.55	1.59	1.66

Tiempo medio de acceso a memoria (ciclos)

$Tamaño_{MC}$: 64KB , Penalización: 25 ciclos

$$ta + (ff_{total} * pf)$$

$tamm_{1via} = 1.00 + (0.037*25) = 1.925$ ciclos

$tamm_{2vias} = 1.36 + (0.031*25) = 2.135$ ciclos

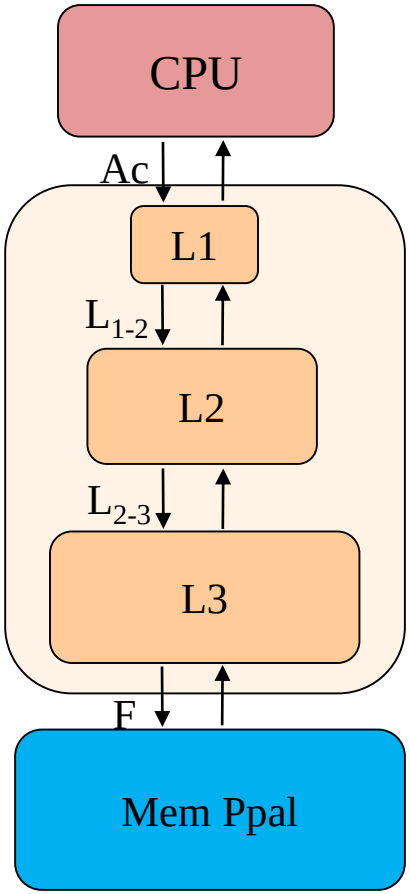
$tamm_{4vias} = 1.44 + (0.030*25) = 2.19$ ciclos

$tamm_{8vias} = 1.53 + (0.029*25) = 2.255$ ciclos

En cachés grandes, aumentar la asociatividad degrada el rendimiento, pues el aumento del ta no compensa la reducción de la ff.

Múltiples niveles de caché (I)

- Consiste en añadir nuevos niveles de caché a la jerarquía de memoria (identificados con L1, L2, etc.) para reducir aún más los accesos a Mem Ppal.



Cada nivel se diseña con características específicas para reducir principalmente cierto tipo de fallo.

La CPU accede al nivel L1, en caso de fallo en L1 accede a L2, si falla en L2 accede a L3.

Los fallos de L1 (L_{1-2}) serán accesos a L2, y los fallos de L2 (L_{2-3}) serán los accesos a L3.

El fallo producido en cada nivel de caché (**Fallo local**) es:

$$Mr_{L1} = \frac{L_{1-2}}{Acc} \qquad Mr_{L2} = \frac{L_{2-3}}{L_{1-2}} \qquad Mr_{L3} = \frac{F}{L_{2-3}}$$

La frec. fallos de todo el sistema de cachés (**Fallo Global**) es:

$$\boldsymbol{Mr_{Global}} = Mr_{L1} * Mr_{L2} * Mr_{L3} = \frac{L_{1-2}}{Acc} * \frac{L_{2-3}}{L_{1-2}} * \frac{F}{L_{2-3}} = \boldsymbol{\frac{F}{Ac}}$$

Múltiples niveles de caché (II)

- **Rendimiento:**
  - Para 2 niveles de caché:

$$\textit{Tiempo de acceso medio a memoria} = \textit{T_acierto}_{L1} + Mr_{L1} * (\textit{T_acierto}_{L2} + Mr_{L2} * \textit{PMiss}_{L2})$$

  - **Frecuencia de fallos local**: número de fallos en la caché dividido por el número total de accesos a esa caché (en L2 es Mr_{L2})
  - **Frecuencia de fallos global**: número de fallos del nivel inferior de la caché dividido entre el número total de accesos a memoria generados por la CPU ($Mr_{L1} * Mr_{L2}$)

- La frecuencia de fallos de los niveles inferiores de la caché suelen ser mayores que la de los niveles superiores, ya que los accesos que cumplen los principio de localidad han sido resueltos por los niveles superiores.
  - Aunque la frecuencia de fallos local de niveles inferiores sea muy superior, permiten reducir de forma importante el Mr global porque actúan sobre los fallos de los niveles inferiores.
- **Ejemplo:** 1000 referencias a memoria, 40 fallos en caché L1 y 20 en caché L2
  - Frecuencia de fallos local para L1 = 40/1000 = **4%**
  - Frecuencia de fallos local para L2 = 20/40 = **50%**
  - Frecuencia de fallos global = 20/1000 = **2%** (4%*50%)

Evolución de las memorias caché

- Familia de procesadores de Intel 80x86, IA32 e IA64.

Procesador	Año	Freq. (MHz)	L1 dato	L1 Instrucción	L2 caché	L3 caché
80386	1985	16-25	-	-	-	-
80486	1989	25-100	8KB unificada		-	-
Pentium	1993	60-300	8KB	8KB	-	-
Pentium Pro	1995	150-200	8KB	8KB	256KB-1MB	-
Pentium II-III	1997-99	233-1400	16KB	16KB	256-512KB	-
Pentium 4	2001	1400-3730	8-16KB	12K- µOps	256KB-2MB	-
Itanium	2001	800	16KB	16KB	96KB	4MB
Itanium 2	2002	1600	32KB		256KB	6MB
Core 2 Duo	2005	1500-2160	32KB/core	32KB/core	2-6MB compartida	-
Core i7 (1ª - 7ª gen)	2008-2017	2660-4000	32KB * 4 cores	32KB * 4 cores	256KB * 4 cores	8MB compartida
Core i7 (8ª gen)	2017	3200-4000	32KB * 6 cores	32KB * 6 cores	256KB * 6 cores	12MB Compartida
Core i7 (9ª gen)	2018	3600	32KB * 8 cores	32KB * 8 cores	256KB * 8 cores	12MB Compartida
Core i9 (7ª-9ª gen)	2018	3600-5000	32KB * 8 cores	32KB * 8 cores	256KB * 8 cores	12-16MB Compartida

MIMD: Taxonomía extendida de Flynn

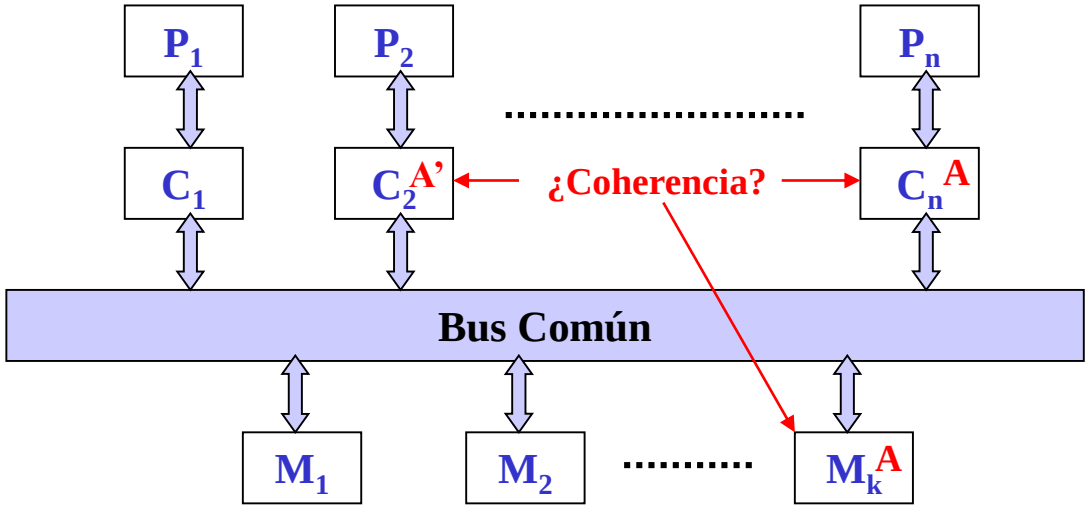
- En base a la distribución e interconexión de la memoria, los procesadores MIMD pueden clasificarse según la taxonomía extendida de Flynn:

		Modelo de comunicación (Disposición lógica)		
		Espacio de direcciones compartido	Espacio de direcciones separado/privado	
Conexión física (Disposición Física)	Memoria Centralizada (Conexión en Bus)	**UMA** (Multiprocesador)	(No tiene sentido)	- Poco escalables
	Memoria Distribuida (Conexión en Red)	**NUMA** (Multiprocesador)	**MPM** (Multicomputador)	- Distinto software. - Casi igual hardware. - Muy escalable.
		- Comunicación por variables compartidas. - Igual software. - Distinto hardware. - Hilos en Paralelo.	- Comunicación por paso de mensajes. - Procesos en Paralelo.	

- **<u>Factores que definen el diseño de un MIMD:</u>**
  - Nº de procesadores.
  - Conexión física de la memoria
    - Bus: Memoria físicamente Centralizada.
    - Red: Memoria físicamente Distribuida.
  - Modelo de comunicación/Compartición de datos en memoria
    - Espacio de direcciones compartido: multiprocesadores escalables con memoria compartida
    - Espacio de direcciones separado/privado: multicomputadores con paso de mensajes.

MIMD: Memoria Centralizada (Conexión por bus)

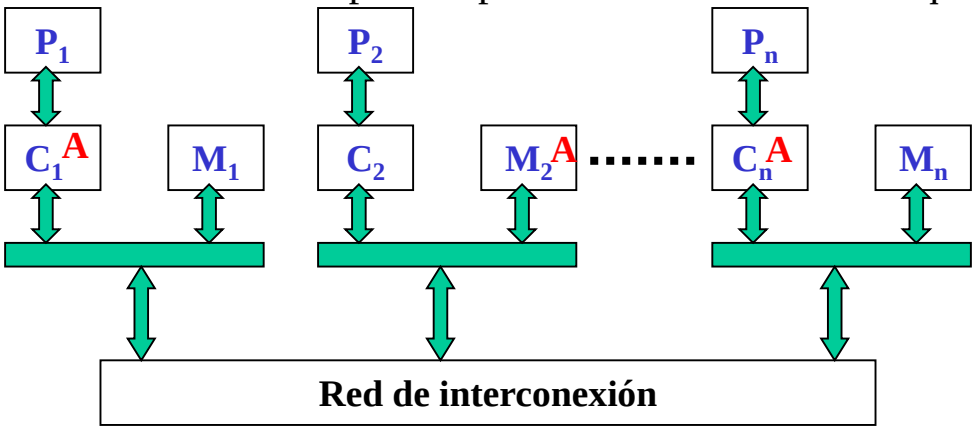
- Procesadores con caché propia que se conectan al bus común para acceder a los recursos compartidos, por lo que la memoria está físicamente centralizada.
- Cada acceso a memoria requiere disponer del bus por lo que genera mucho tráfico.
- Estos sistemas son poco escalables (2-32 procesadores) ya que un mayor número de ellos exigiría un elevado ancho de banda en el bus.
  - Este elevado tráfico se reduce gracias a las caché que cada procesador posee.



- A estos MIMD se les denomina UMA (Acceso uniforme a memoria), pues el tiempo en acceder a memoria es siempre el mismo independientemente del procesador y la dirección, ya que toda la memoria se encuentra en el mismo punto del sistema.
- Puesto que cada uno posee su propia caché, son necesarios mecanismos de coherencia que mantengan la coherencia de los datos.

MIMD: Memoria Distribuida (Conexión por red)

- Cada procesador posee su memoria caché y principal, y accede a la memoria de otros procesadores a través de la red, por lo que la memoria está distribuida.
- Son más escalables (admiten un mayor número de procesadores) ya generan menos tráfico en la red, debido a que cada procesador sólo accede a la red si el dato no está en su propia memoria.
- El tiempo de acceso a memoria es variable pues depende de la memoria a la que se accede.



- Pueden ser de 2 tipos:
  - **Espacio de direcciones separado/privado → Sistemas de paso de mensajes (MPM).**
    - Cada procesador posee un espacio de direcciones propio por lo que la comunicación entre procesadores se realiza por paso de mensajes de forma explícita mediante primitivas de enviar y recibir.
  - Espacio de direcciones compartido → Acceso a memoria no uniforme (NUMA).
    - Todas las memorias forman un espacio de direcciones común a todos los procesadores por lo que la comunicación se realiza compartiendo variables (UMA y NUMA generan códigos semejantes).
    - Al encontrarse la memoria distribuida en los NUMA, el tiempo de acceso a memoria dependerá de la memoria física a la que tenga que acceder.