Tema 2:
Jerarquía de Memoria

Arquitectura de Computadores

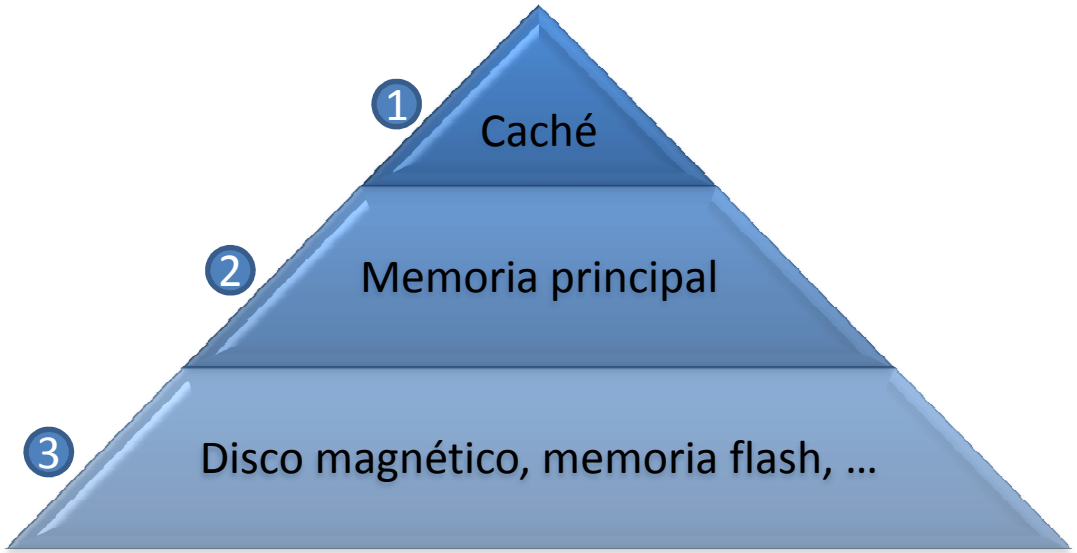
Grado en Ingeniería Informática

Mercedes Rodríguez García

Bibliografía

Estructura y diseño de computadores: Capítulo 5.
Patterson y Hennessy. Editorial Reverte, 2011.

Jerarquía de memoria



Jerarquía de memoria

El procesador solicita los datos y las instrucciones al primer nivel de la jerarquía (memoria caché).

Si los datos que requiere el procesador se encuentran en la memoria caché, se produce un **ACIERTO**.

Si los datos que requiere el procesador **no** se encuentran en la memoria caché, se produce un **FALLO**. En caso de fallo, los datos se deben buscar en el siguiente nivel de la jerarquía.

Los fallos de caché son tratados por la unidad de control del procesador y por el controlador de memoria. El controlador de memoria actualmente reside dentro del procesador y es el encargado de gestionar el acceso a memoria principal para actualizar el contenido de la caché.

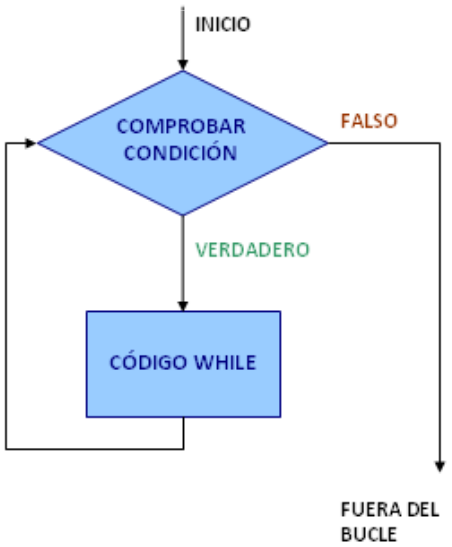
OBJETIVO: Conseguir alta frecuencia de aciertos en la memoria caché.

NOTA: Si los datos están en el la memoria caché significa que también están en la memoria principal.

Principio de localidad temporal

Principio de localidad temporal: Si se accede a una posición de memoria, probablemente esta posición será **utilizada de nuevo** en un corto intervalo de tiempo.

La memoria caché es capaz de aprovechar el principio de localidad temporal.



Ejemplo: La mayoría de los programas contienen **bucles**, por lo que probablemente se accederá repetidamente a las mismas instrucciones y datos. Por tanto, los programas presentan alta localidad temporal.

Principio de localidad espacial

Principio de localidad espacial: Si se accede a una posición de memoria, las posiciones que se encuentran **próximas** a ella probablemente serán utilizadas en un corto intervalo de tiempo.

La memoria caché es capaz de aprovechar el principio de localidad espacial.

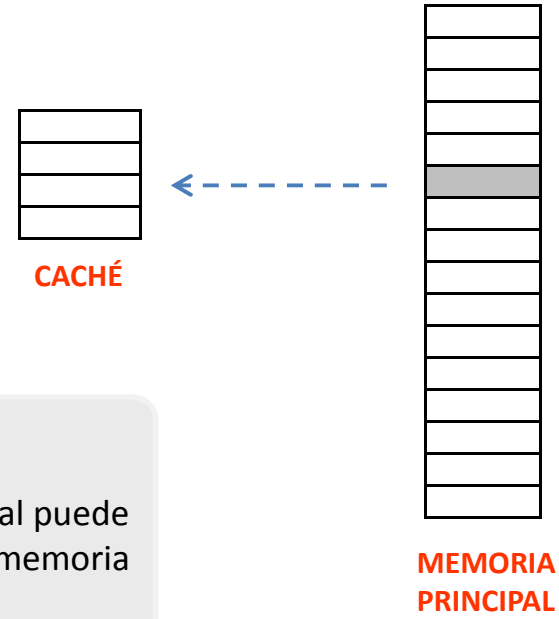
```
…

12:  Add $t0, $t1, $t2
16:  Sw $t0, 04
20:  Sub $t2, $t5, $t3

…
```

Ejemplo: Las **instrucciones** de un programa y los elementos de un **vector** se almacenan de forma contigua en memoria. Por tanto, los programas presentan alta localidad espacial.

Tipos de memoria caché

1 **CACHÉ DIRECTA**

Un bloque procedente de memoria principal sólo puede situarse en **UNA** posición determinada de la memoria caché.



2 **CACHÉ ASOCIATIVA**

Un bloque procedente de memoria principal puede situarse en **CUALQUIER** posición de la memoria caché.

3 **CACHÉ ASOCIATIVA POR CONJUNTOS**

Un bloque procedente de memoria principal sólo puede situarse dentro de un **CONJUNTO** de posiciones determinado de la memoria caché.

Caché Directa

Un bloque de memoria principal sólo puede ser situado en **UNA** posición concreta de la memoria caché directa.

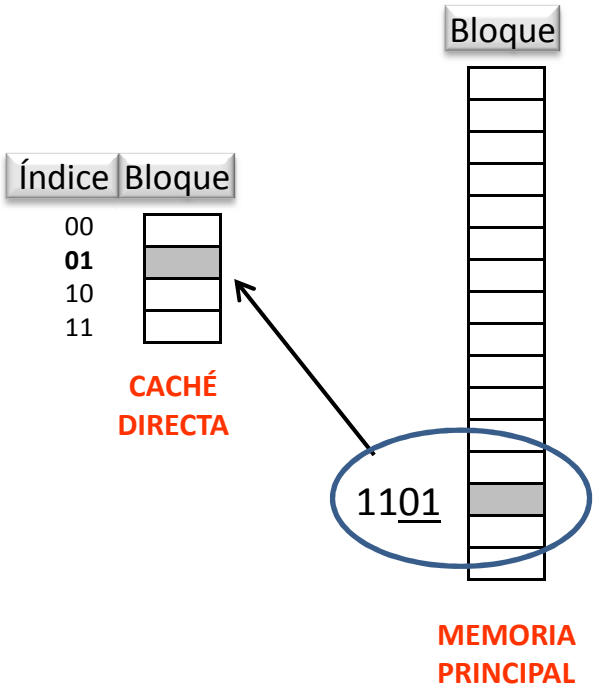
Esta posición está determinada por la dirección que tiene ese bloque en memoria principal.

NOTA: Las posiciones de la memoria caché se identifican con un índice.

$$\text{INDICE}_{\text{caché}} = \text{DIRECCION BLOQUE}_{\text{MP}} \text{ MOD } \text{Nº BLOQUES}_{\text{caché}}$$

Caché Directa

$INDICE_{caché}$ = $DIRECCION\ BLOQUE_{MP}$ MOD $NºBLOQUES_{caché}$



En el ejemplo,

¿En qué posición de la caché directa se situará el bloque de memoria principal que tiene como dirección 1101_2?

$$Índice_{caché} = 13 \ \ MOD \ \ 4 = 1_{10} = 01_2$$

Caché Directa

$\text{INDICE}_{\text{caché}} = \text{DIRECCION BLOQUE}_{\text{MP}} \text{ MOD } \text{NºBLOQUES}_{\text{caché}}$

Índice	Bloque
00	
01	
10	
11	

CACHÉ DIRECTA

Bloque	
0000	
0001	
0010	
0011	
0100	
0101	
0110	
0111	
1000	
1001	
1010	
1011	
1100	
1101	
1110	
1111	

MEMORIA PRINCIPAL

En el ejemplo,

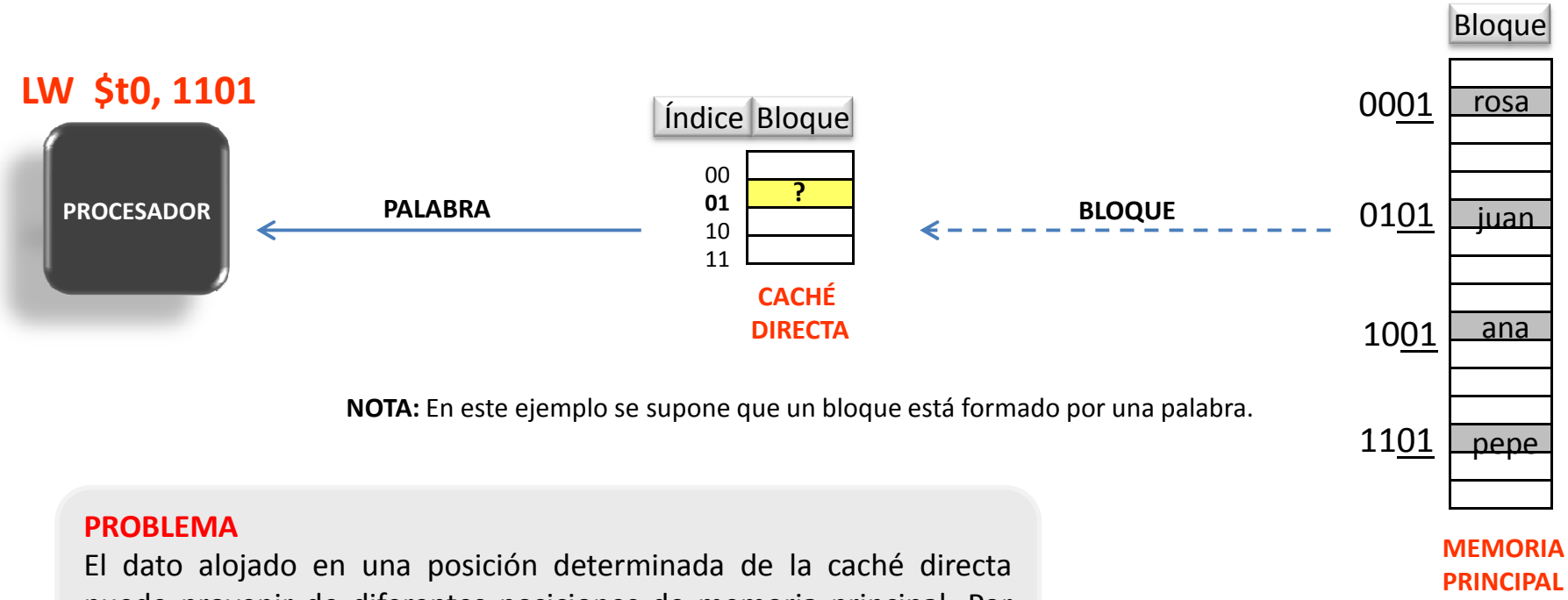
1.- ¿Por qué el índice de la caché sólo tiene dos bits?

2.- ¿En qué posición de la caché se almacenaría el bloque señalado en amarillo?

3.- ¿En qué posición de la caché se almacenaría el bloque señalado en verde?

4.- ¿Qué bloques de la memoria principal sólo podrían almacenarse en la posición 01 de la caché?

Caché Directa



NOTA: En este ejemplo se supone que un bloque está formado por una palabra.

PROBLEMA

El dato alojado en una posición determinada de la caché directa puede provenir de diferentes posiciones de memoria principal. Por ejemplo, el dato alojado en la posición 01 de la caché directa puede provenir de 4 posiciones diferentes de memoria principal.

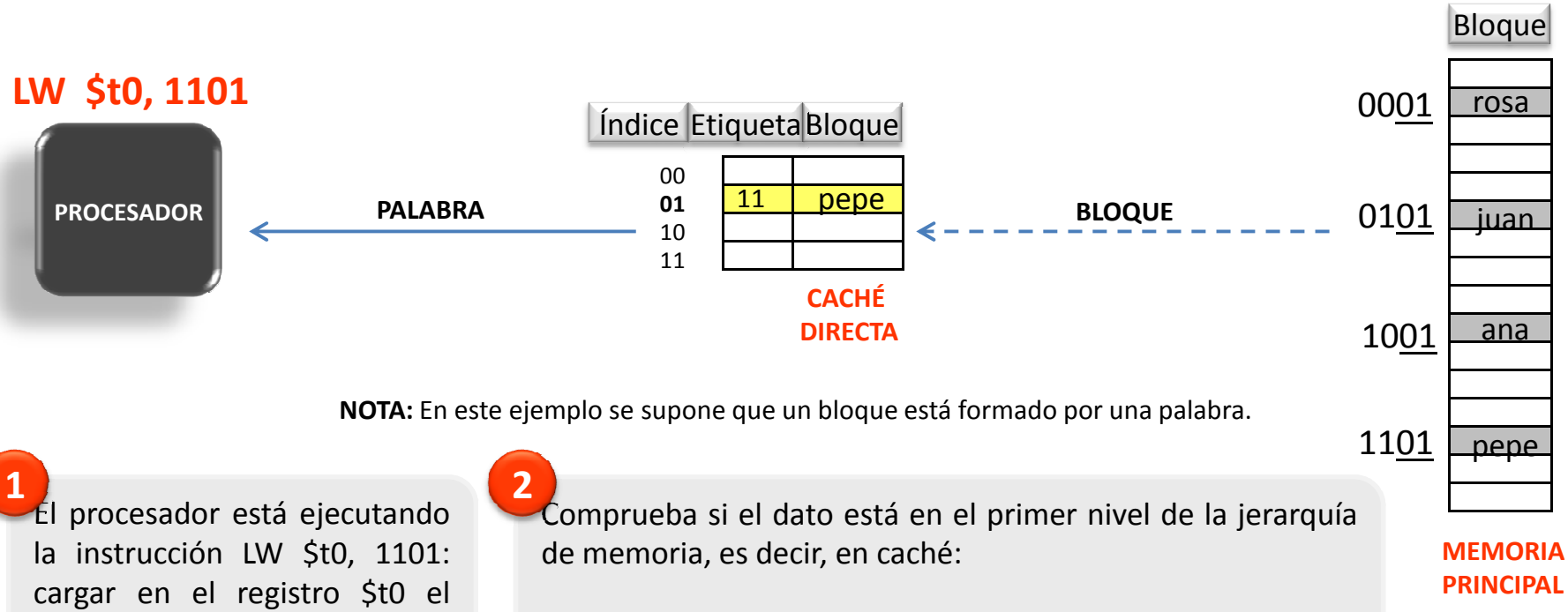
Si el procesador necesita el dato que está en la posición de memoria 1101 ¿cómo puede saber el procesador si ese dato está en caché?

Caché Directa

SOLUCIÓN

Se añade un nuevo campo en cada entrada de la caché. Este campo se denomina **etiqueta** y contendrá la parte de la dirección que no aparece en el índice, es decir, la parte más significativa.

Caché Directa



NOTA: En este ejemplo se supone que un bloque está formado por una palabra.

1 El procesador está ejecutando la instrucción LW $t0, 1101: cargar en el registro $t0 el dato que está en la dirección de memoria 1101.

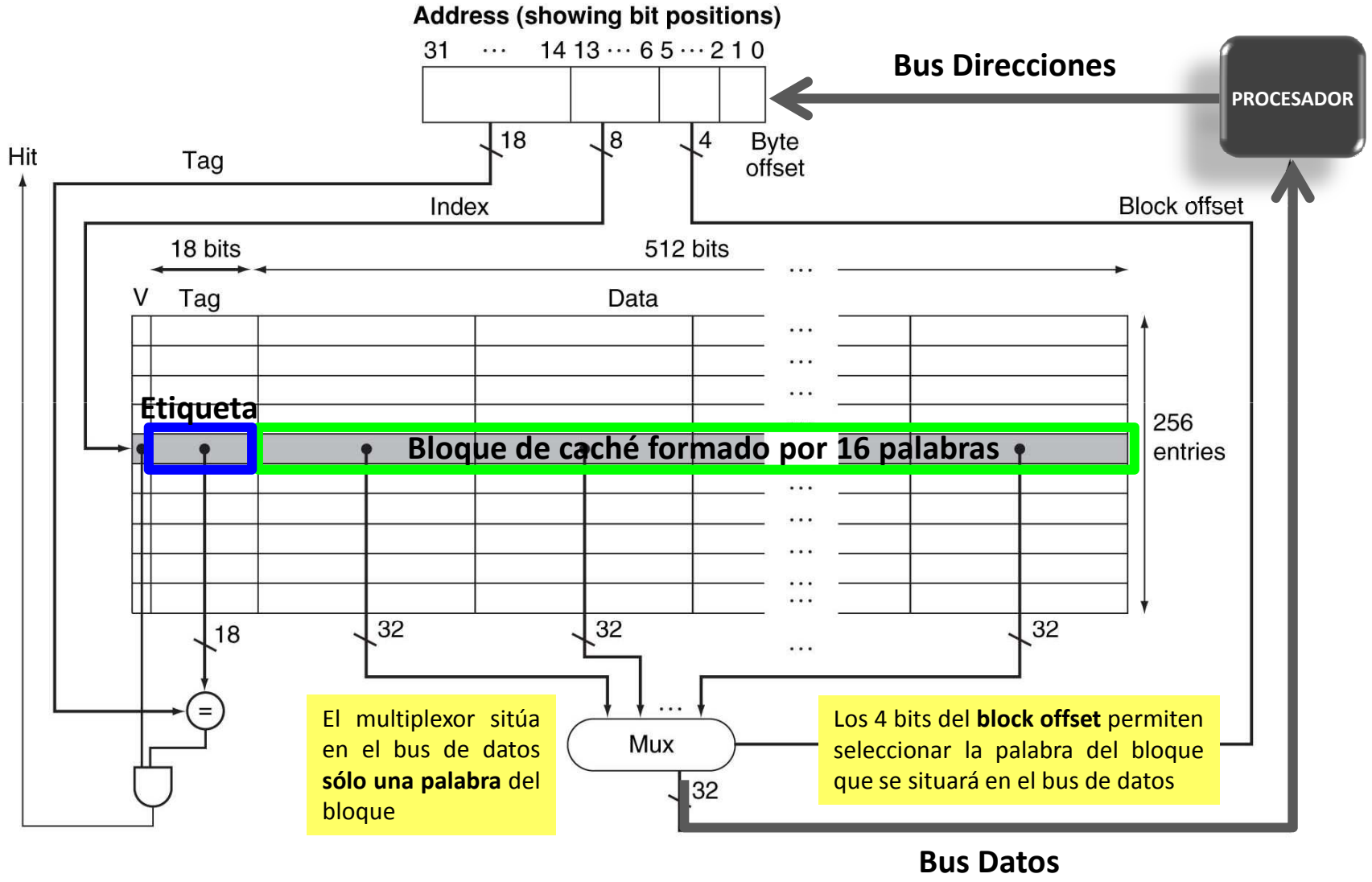
Por tanto, el procesador tiene que acudir al sistema de memoria para obtener el dato.

2 Comprueba si el dato está en el primer nivel de la jerarquía de memoria, es decir, en caché:

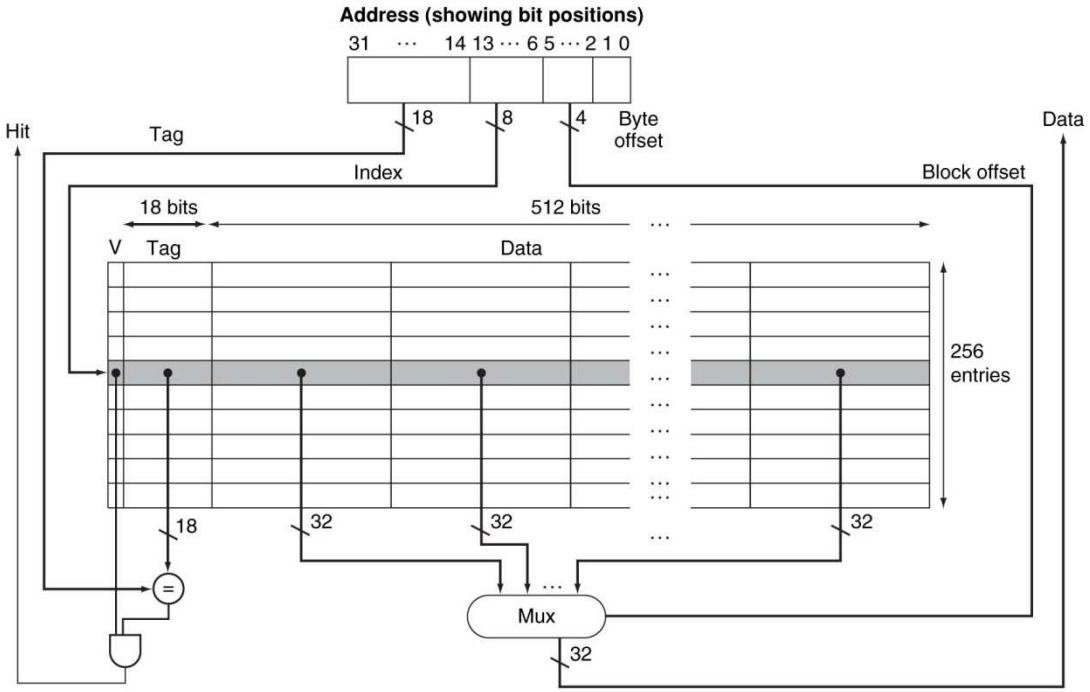
1º) Ir al índice que coincide con la parte menos significativa de la dirección 11<u>01</u> presente en la instrucción LW.

2º) Verificar si la etiqueta de ese índice coincide con la parte más significativa de la dirección <u>11</u>01. Como coincide, se puede afirmar que el dato está en caché (ACIERTO). En caso de no coincidir, se puede afirmar que el dato no está en caché (FALLO) y habría que ir a memoria principal a buscarlo.

Caché Directa



Caché Directa



¿Cuántos bits tiene la etiqueta?

Tamaño etiqueta = **m – n – BLOCKoffset – BYTEoffset**

m: bits del bus de direcciones.

n: bits del índice.

BLOCKoffset: bits necesarios para identificar una palabra dentro del bloque.

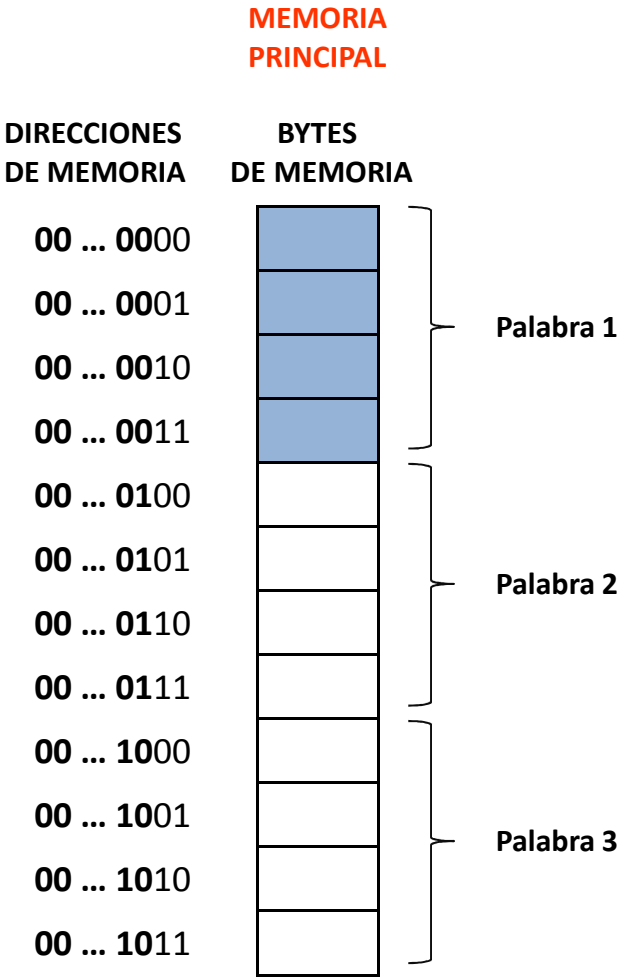
BYTEoffset: bits necesarios para identificar un byte dentro de la palabra.

Bytes, palabras y bloques

Recordando la arquitectura MIPS-32…

- La palabra está formada por 4 bytes (32 bits).
- El direccionamiento es a nivel de byte.
- La dirección de memoria de una palabra siempre es múltiplo de 4.

¿Cuál es la dirección de la palabra 2?



Bytes, palabras y bloques

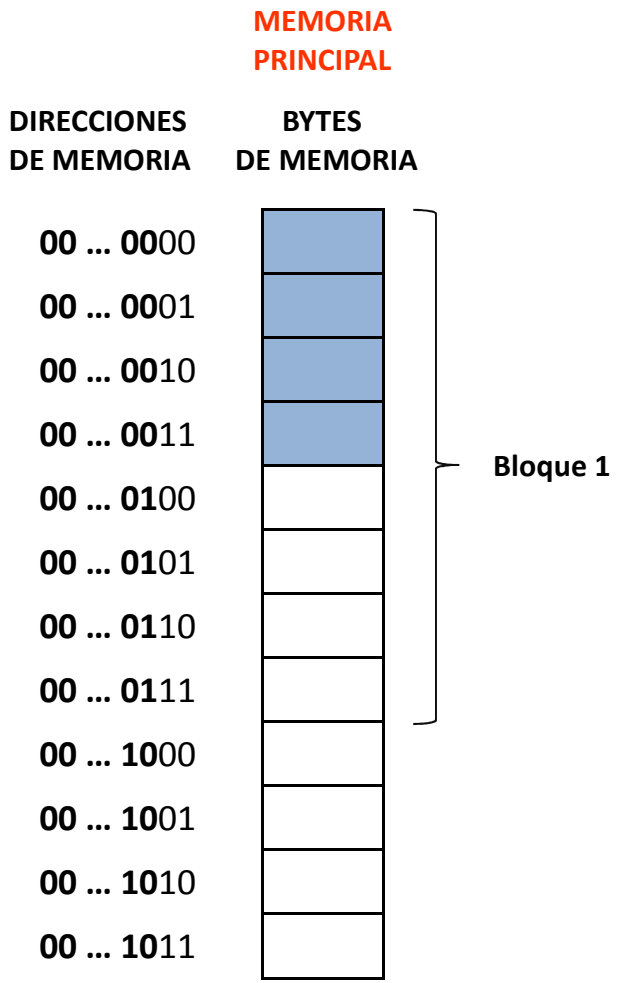
¿Qué es un bloque?

Es la unidad mínima de información que puede transmitirse entre caché y memoria principal.

En las implementaciones reales los bloques tienen **varias palabras**.

Ejemplo :
- Palabra = 4 bytes
- Bloque = 8 bytes

¿Cuál es la dirección del bloque 2?



Bytes, palabras y bloques

1.- ¿Cuál es la dirección del bloque 1?

2.- ¿Qué tienen en común las direcciones de los bytes del bloque 1?

3.- ¿Cuál es la dirección del bloque 2?

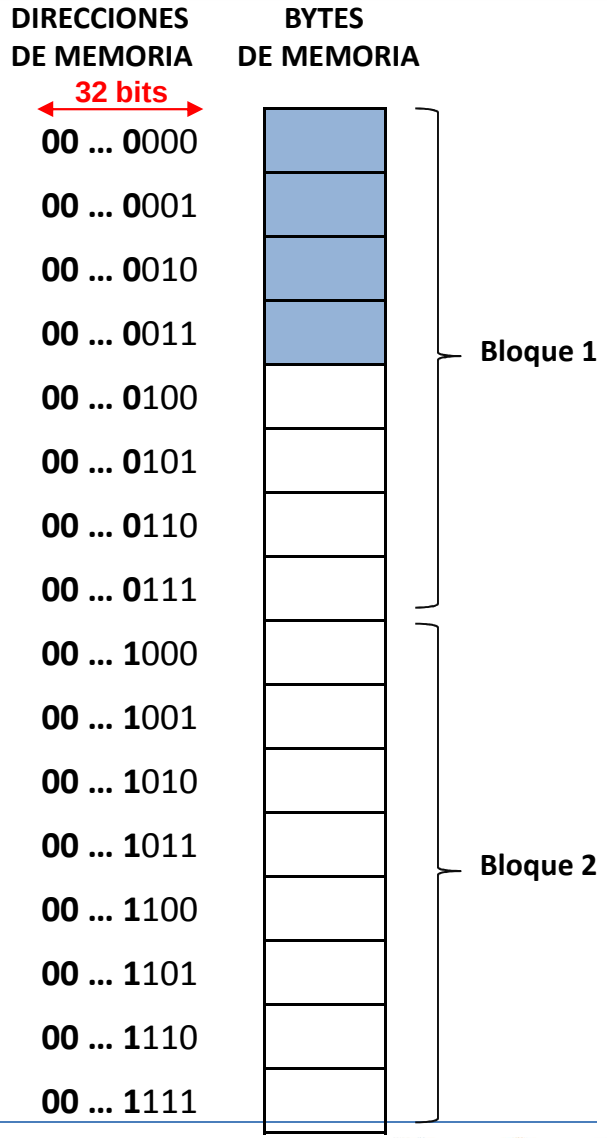
4.- ¿Qué tienen en común las direcciones de los bytes del bloque 2?

5.- Entonces, ¿Qué bits de la dirección se pueden ignorar si sólo queremos identificar bloques?

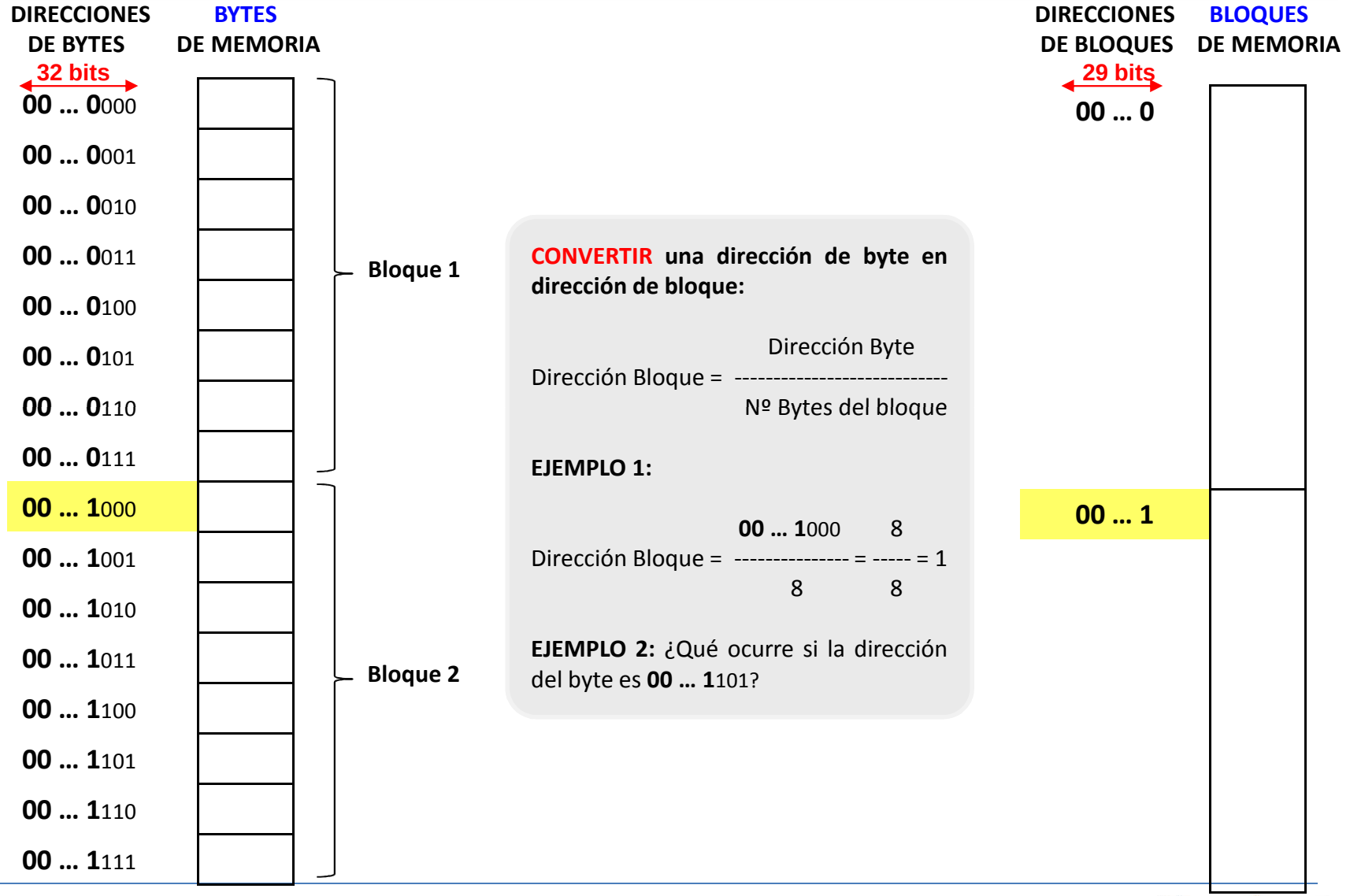
6.- ¿Sería correcto considerar que la dirección del bloque 1 es 00…0 y la dirección del bloque 2 es 00…1?

29 bits más significativos

29 bits más significativos



Bytes, palabras y bloques



CONVERTIR una dirección de byte en dirección de bloque:

$$\text{Dirección Bloque} = \frac{\text{Dirección Byte}}{\text{Nº Bytes del bloque}}$$

EJEMPLO 1:

$$\text{Dirección Bloque} = \frac{\mathbf{00 \ldots 1}000}{8} = \frac{8}{8} = 1$$

EJEMPLO 2: ¿Qué ocurre si la dirección del byte es **00 … 1**101?

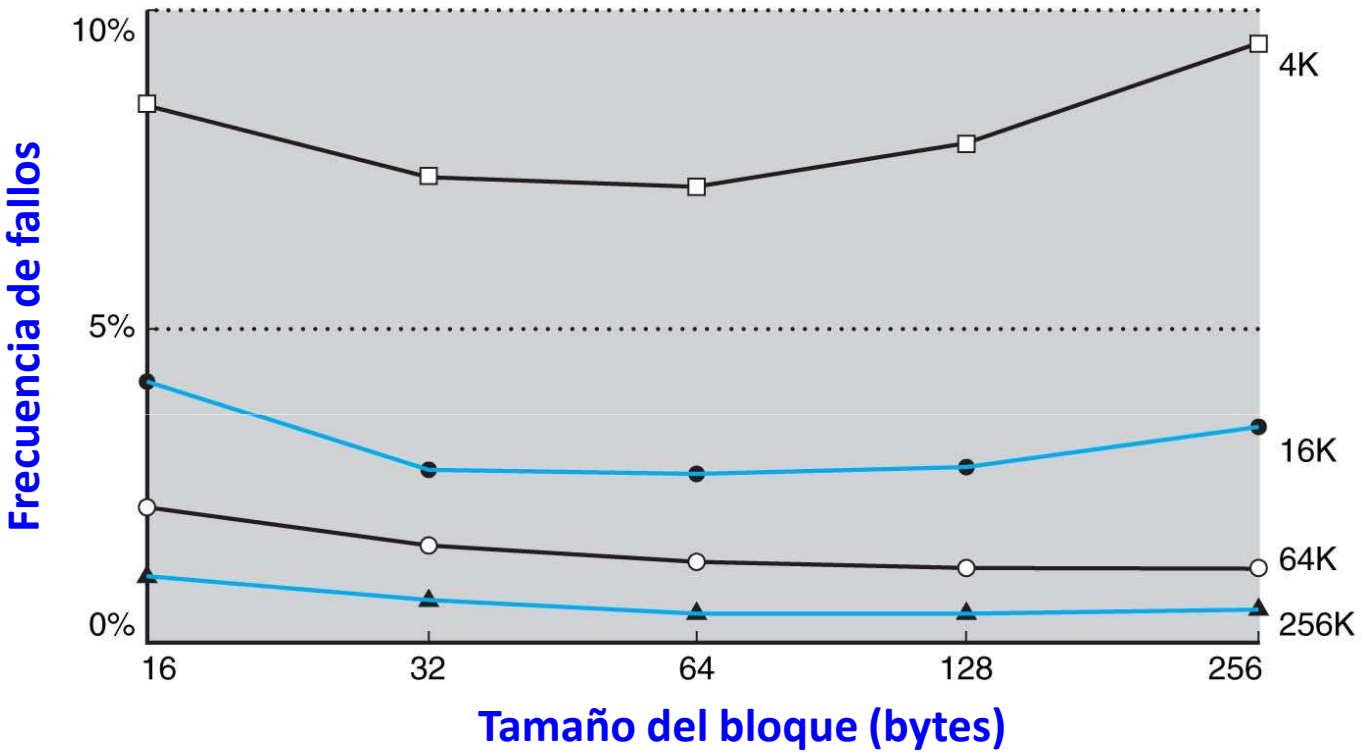
¿Qué tamaño debe tener el bloque?

Para aprovechar la localidad espacial, el tamaño del bloque debe ser **SUPERIOR** a una palabra. Dependiendo de la arquitectura del procesador, la palabra es de 4 bytes -32bits- o de 8 bytes -64 bits-.

Si el bloque está formado por más de una palabra obtenemos las siguientes ventajas:

- La frecuencia de fallos en caché disminuye.
- Se reduce el número de etiquetas y, por tanto, el espacio requerido para ellas en memoria caché.

¿Qué tamaño debe tener el bloque?



-Figura obtenida de [PATT11]-

¡Perfecto! Si incrementamos el tamaño del bloque, disminuye la frecuencia de fallos debido a la localidad espacial.

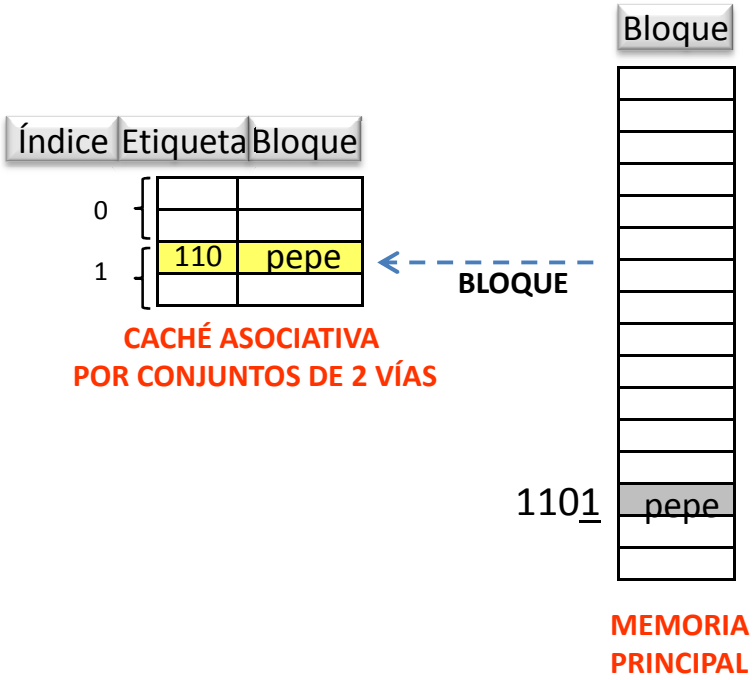
¡Cuidado! no incrementarlo demasiado, puedo conseguir el efecto contrario.

Caché asociativa por conjuntos

Es una caché que está dividida en conjuntos de bloques. Si los **conjuntos están formados por n bloques** se dice que es una caché asociativa por conjuntos de **n vías**. Cada conjunto se identifica por un índice.

Un bloque de memoria principal sólo puede ser situado en **UN CONJUNTO** concreto de la memoria caché asociativa por conjuntos, pero dentro de ese conjunto puede emplazarse en cualquier bloque (método aleatorio) o en el bloque que lleva más tiempo sin utilizarse (método LRU -Least Recently Used-).

¿Cómo sabemos en qué conjunto se puede situar? Para averiguarlo hay que realizar un cálculo con la dirección que tiene ese bloque en memoria principal.



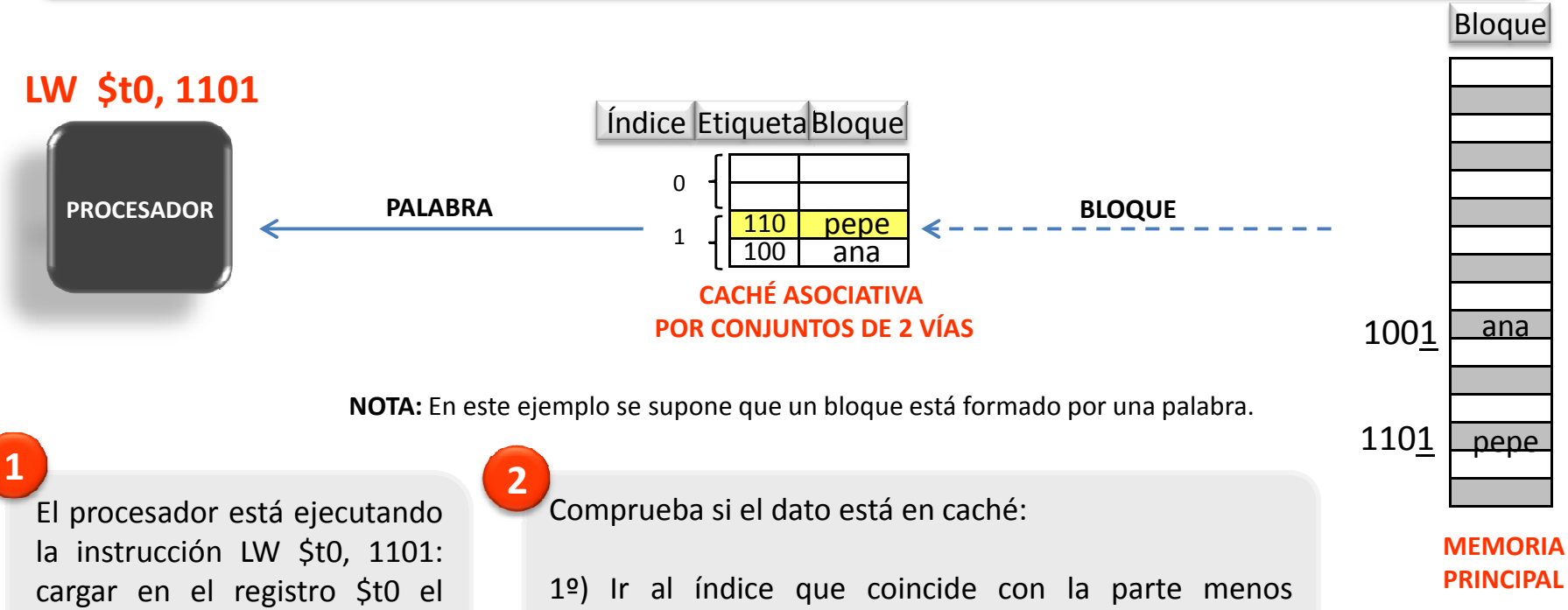
$\text{Índice}_{\text{caché}} = 13 \text{ MOD } 2 = 1 = 1_2$

$\text{INDICE}_{\text{caché}} = \text{DIRECCION BLOQUE}_{\text{MP}} \text{ MOD } \text{NºCONJUNTO}_{\text{caché}}$

El bloque se tiene que situar en el conjunto de índice 1, dentro de ese conjunto puede emplazarse en cualquier bloque (método aleatorio) o en el bloque que lleva más tiempo sin utilizarse (método LRU).

Caché asociativa por conjuntos

LW $t0, 1101



NOTA: En este ejemplo se supone que un bloque está formado por una palabra.

1 El procesador está ejecutando la instrucción LW $t0, 1101: cargar en el registro $t0 el dato que está en la dirección de memoria 1101.

2 Comprueba si el dato está en caché:

1º) Ir al índice que coincide con la parte menos significativa de la dirección 110<u>1</u> presente en la instrucción LW.

2º) Inspeccionar todas las etiquetas del conjunto en paralelo para verificar si hay alguna que coincida con la parte más significativa de la dirección <u>1101</u>. Como hay una etiqueta que coincide, la petición de memoria ACIERTA en caché. En caso contrario, se produce un FALLO y hay que ir a memoria principal a buscar el dato.

Caché asociativa

Un bloque de memoria principal puede situarse en **CUALQUIER** posición de la caché.

Este tipo de caché **no tiene índices**, sólo tiene etiquetas. Si no tiene índices, entonces

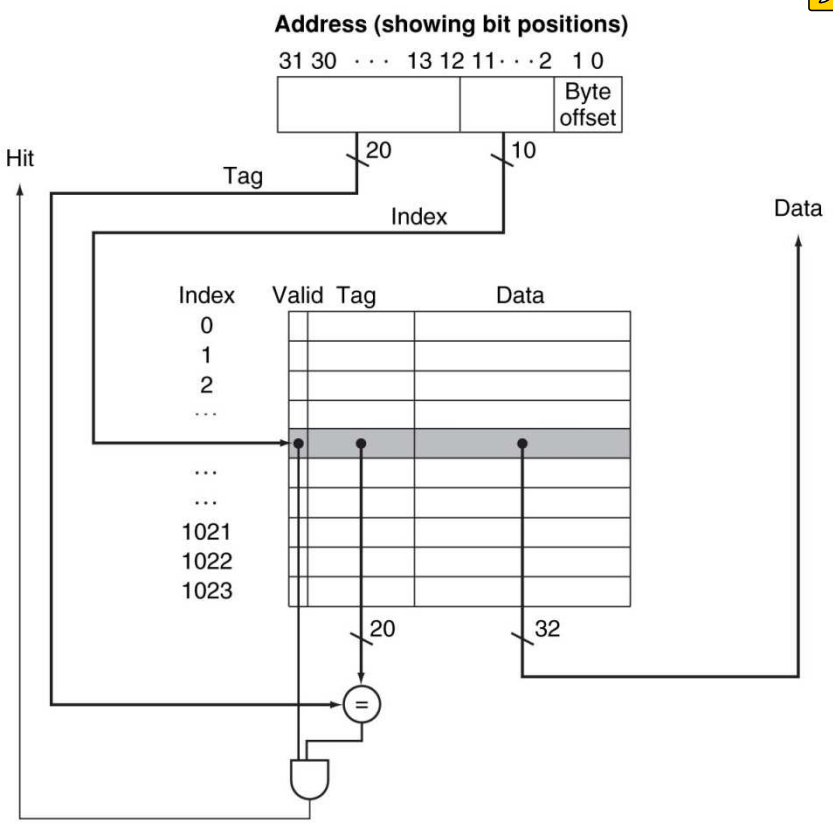
¿Cómo encontramos un dato en la caché?

Para encontrar un dato determinado hay que inspeccionar todas las entradas de la caché.

Para que la búsqueda sea factible, se realiza en paralelo a través de unos comparadores que tienen las entradas.

A diferencia de los comparadores que utiliza la caché directa o la caché asociativa por conjuntos, éstos son de mayor dimensión (tienen más puertas lógicas).

Arquitectura de la memoria caché

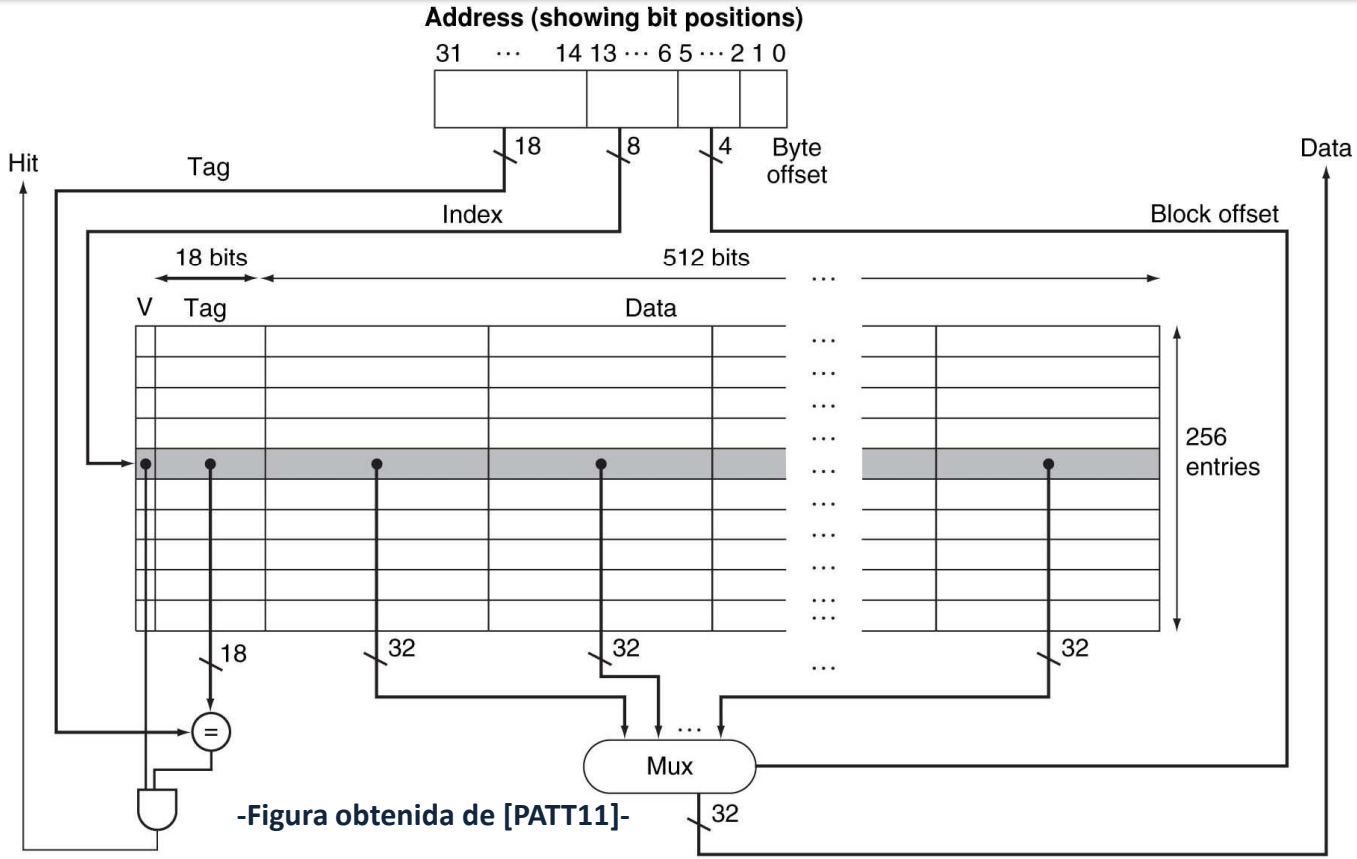


-Figura obtenida de [PATT11]-

1. Tamaño de palabra =
2. Direccionamiento a nivel de byte =
3. Nº de palabras por bloque =
4. Tipo de caché =

NOTA: Los **dos bits menos significativos** de la dirección identifican los bytes que constituyen una palabra. Como lo que se sitúa en el bus de datos hacia el procesador es una palabra completa (no un byte de la palabra) estos bits de la dirección se ignoran.

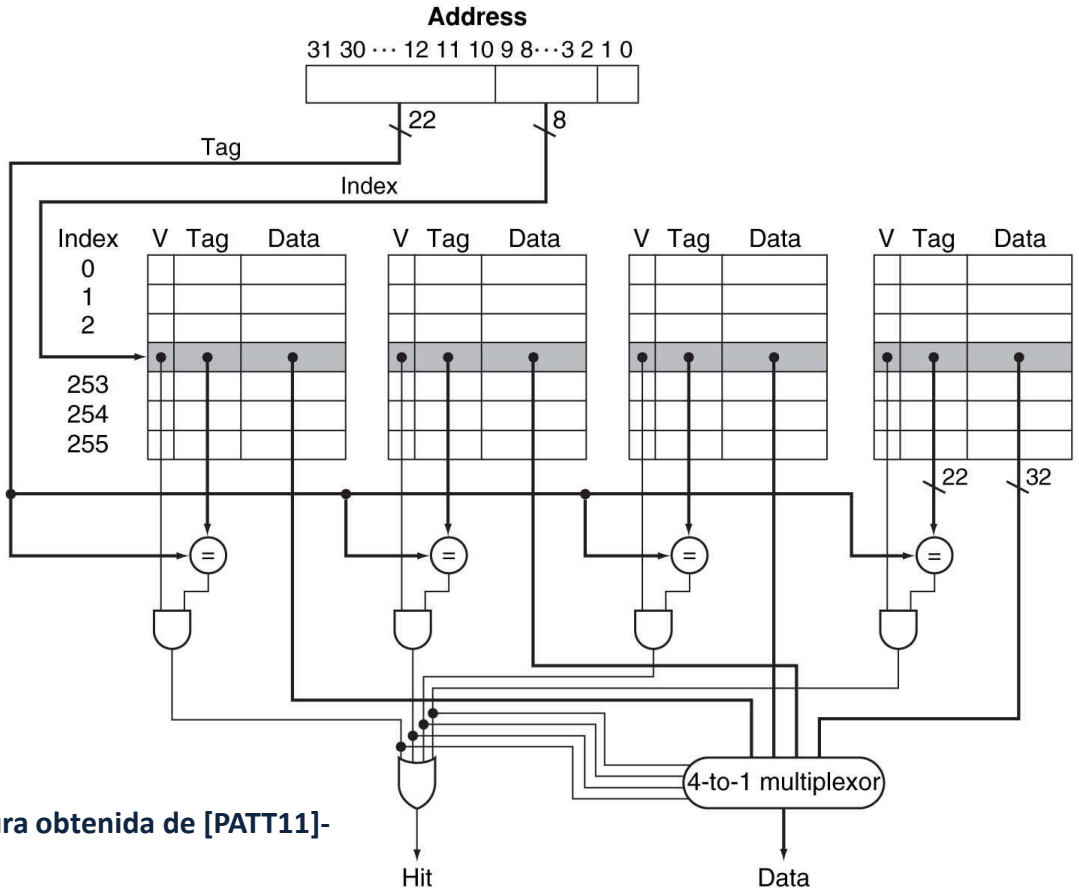
Arquitectura de la memoria caché



-Figura obtenida de [PATT11]-

1. Tamaño de palabra =
2. Direccionamiento a nivel de byte =
3. Nº de palabras por bloque =
4. Tipo de caché =

Arquitectura de la memoria caché



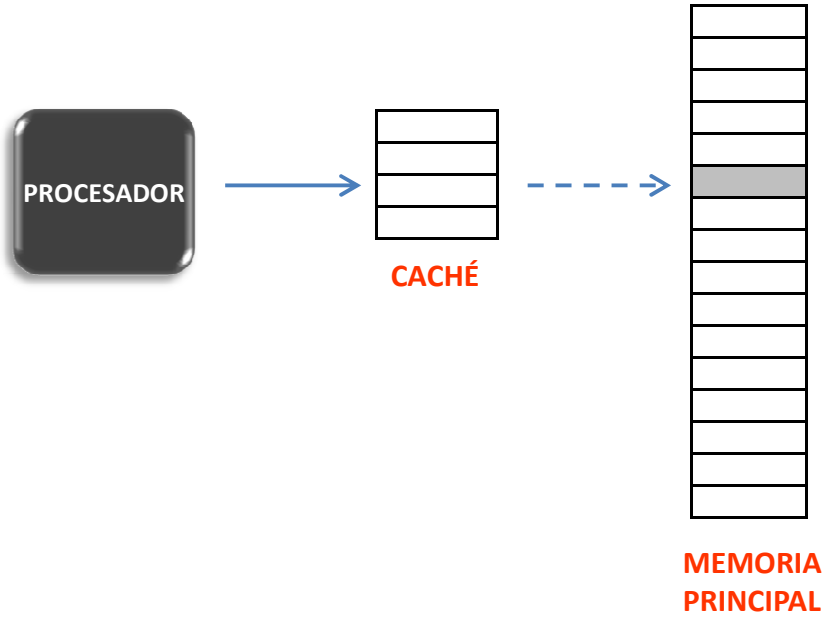
-Figura obtenida de [PATT11]-

1. Tamaño de palabra =
2. Direccionamiento a nivel de byte =
3. Nº de palabras por bloque =
4. Tipo de caché =

Políticas de escritura

POLÍTICA DE ESCRITURA: cuando el procesador actualiza un dato en memoria caché, la política de escritura determina en qué momento se debe actualizar la memoria principal.

OBJETIVO: mantener la **coherencia** de los datos entre caché y memoria principal.



1

Escritura DIRECTA

El dato se escribe a la vez en caché y memoria principal.

Prestaciones:

2

Escritura RETARDADA o POST-ESCRITURA

El dato sólo se escribe en caché. La memoria principal se actualizará cuando se reemplace ese bloque.

Prestaciones:

Sistemas de caché multinivel

La caché L1 está situada en medio de los componentes principales del procesador. Cuando el procesador requiere un dato del sistema de memoria siempre lo lee de caché L1. Es importante **minimizar el tiempo de acceso a la caché L1** para conseguir que el procesador trabaje con un ciclo de reloj lo más corto posible.

Las cachés L2 y L3 también residen en el procesador, pero se sitúan a medio camino entre la L1 y la memoria principal. Cuando se produce un fallo en la caché L1, se busca el dato en la caché L2 para almacenarlo en la L1; finalmente el procesador lo lee de la L1. Si el dato no estuviese en la L2, se buscaría en la caché L3 y, en su defecto, en la memoria principal. Por tanto, es importante **minimizar la frecuencia de fallos en las cachés L2 y L3** para disminuir la penalización por fallo debida a los largos tiempos que se requieren para acceder a memoria principal.

¿Qué características tendrá la caché L1?

¿Qué características tendrá la caché L2?

¿Las decisiones de diseño son diferentes?

Sistemas de caché multinivel

Para minimizar el tiempo de acceso a la caché L1:

1.- **Decrementar** el grado de asociatividad de la caché L1.- Caché con bajo grado de asociatividad → comparadores y multiplexores para gestionar las vías con menos puertas lógicas → menor tiempo de retardo.

2.- **Decrementar** la capacidad de la caché L1.- Caché con baja capacidad → decodificador para gestionar el índice con menos puertas lógicas → menor tiempo de retardo.

Sistemas de caché multinivel

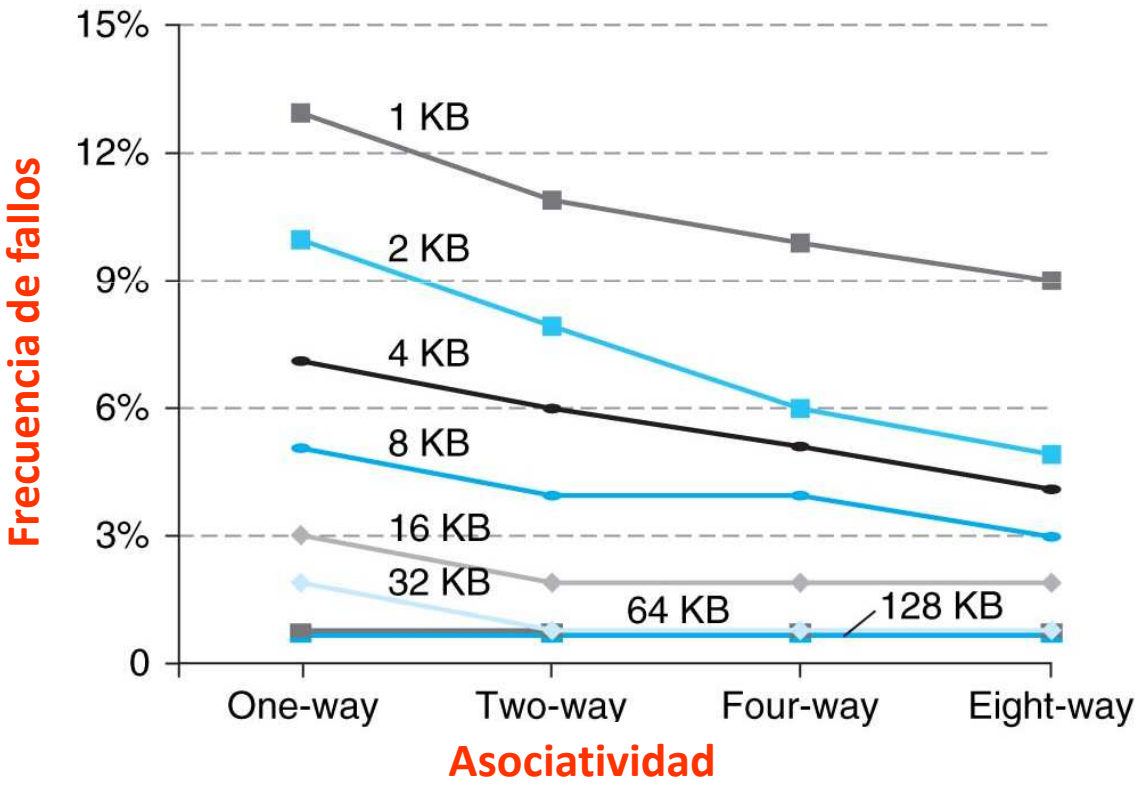
Para minimizar la frecuencia de fallos en las cachés L2 y L3:

1.- **Incrementar** el grado de asociatividad de la cachés.- En las cachés directas y en las cachés asociativas por conjuntos, varios bloques de memoria compiten por situarse en la misma posición (cachés directas) o en el mismo conjunto (cachés asociativas por conjuntos) → los bloques almacenados en estos tipos de caché son más proclives a ser desalojados → puede suceder que se desalojen bloques que iban a ser utilizados en corto plazo.

2.- **Incrementar** la capacidad de las cachés.- Como es lógico, cuanto mayor sea la caché, más datos podrá contener y, por tanto, más probable será obtener un ACIERTO.

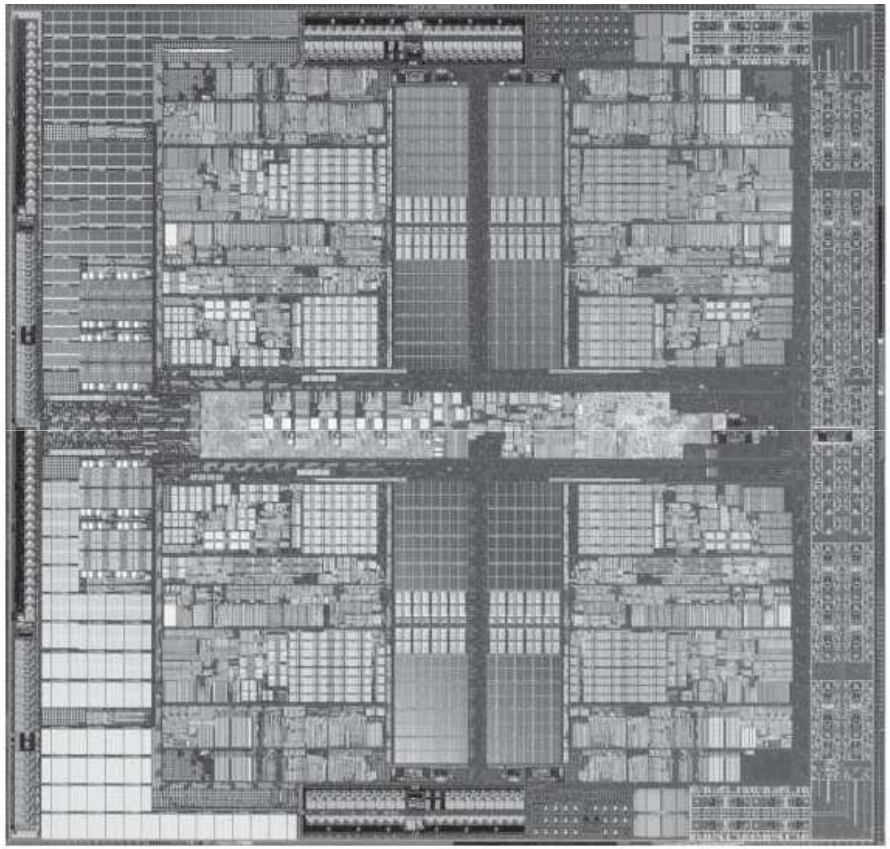
Sistemas de caché multinivel

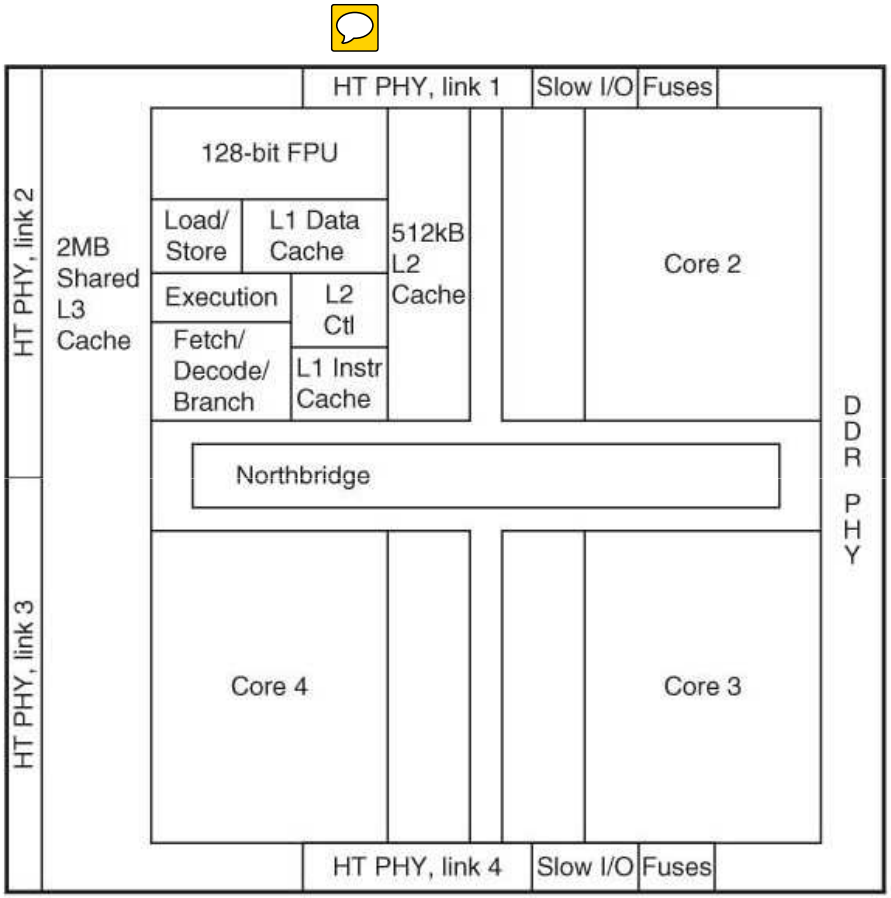
Esta gráfica muestra la frecuencia de fallos en cachés de varios tamaños con la asociatividad variando desde correspondencia directa (1 vía) a asociativa por conjuntos de 8 vías.



-Gráfica obtenida de [PATT11]-

Sistemas de caché multinivel

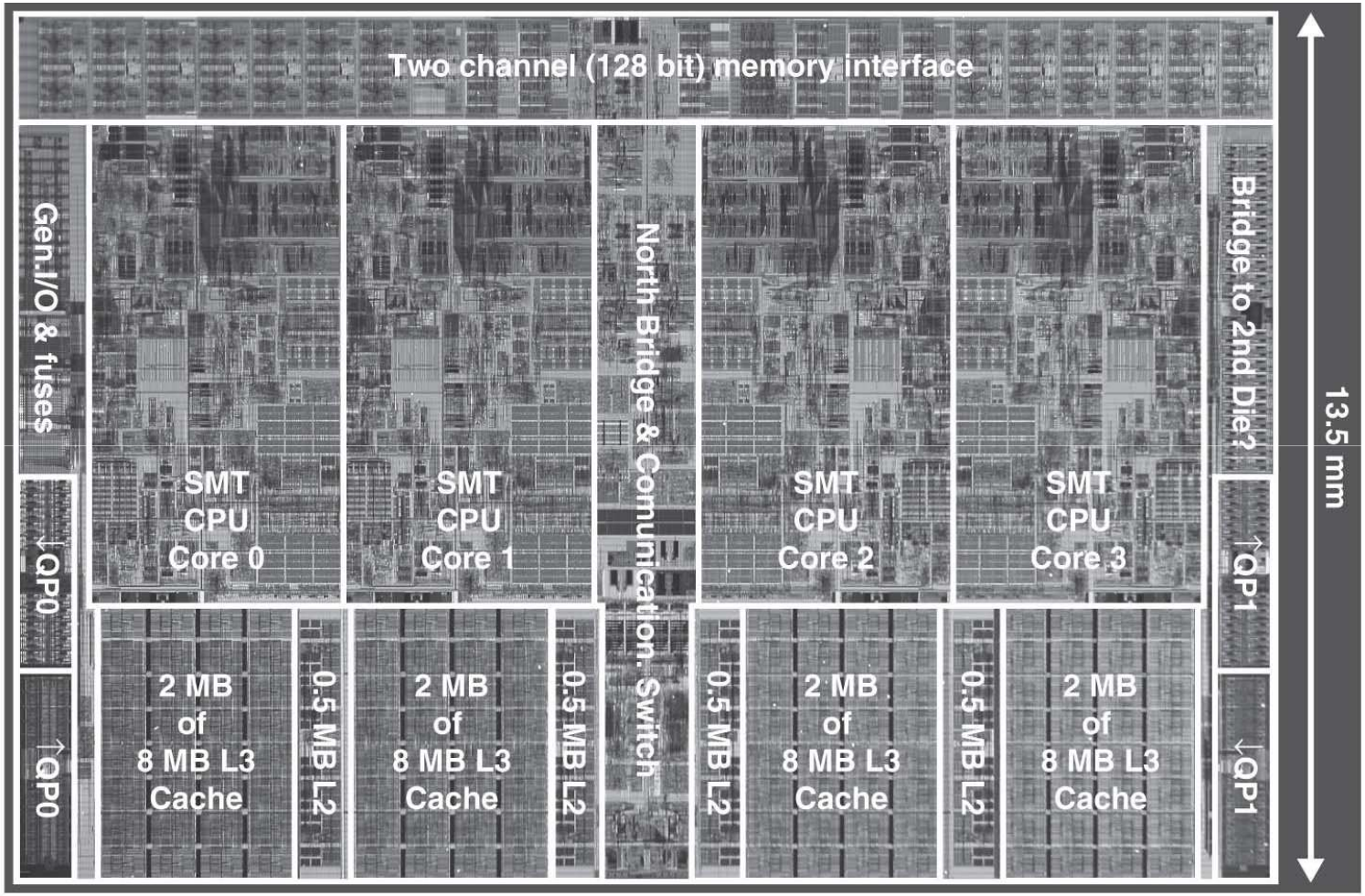




-Figura obtenida de [PATT11]-

Procesador AMD Opteron X4

Sistemas de caché multinivel



-Figura obtenida de [PATT11]-

Microarquitectura Intel Nehalem (procesadores basados en esta microarquitectura: Core i7, Xeon)

Sistemas de caché multinivel

	Características	Intel Nehalem	AMD Opteron
L1	Organización	Cachés I y D separadas	Cachés I y D separadas
	Capacidad	32 KB (I) y 32KB (D) por núcleo	64 KB (I) y 64 KB (D) por núcleo
	Asociatividad	Asociativa por conjuntos 4 vías (I), 8 vías (D)	Asociativa por conjuntos 2 vías
	Tamaño del bloque	64 bytes	64 bytes
	Política de reemplazo	LRU aproximado	LRU
	Política de escritura	Escritura directa	Escritura directa
L2	Organización	Unificada	Unificada
	Capacidad	256 KB por núcleo	512 KB por núcleo
	Asociatividad	Asociativa por conjuntos 8 vías	Asociativa por conjuntos 16 vías
	Tamaño del bloque	64 bytes	64 bytes
	Política de reemplazo	LRU aproximado	LRU aproximado
	Política de escritura	Escritura directa	Escritura directa
L3	Organización	Unificada	Unificada
	Capacidad	8 MB compartida	2 MB compartida
	Asociatividad	Asociativa por conjuntos 16 vías	Asociativa por conjuntos 32 vías
	Tamaño del bloque	64 bytes	64 bytes
	Política de reemplazo	------ (dato no disponible)	------ (dato no disponible)
	Política de escritura	Escritura directa	Escritura directa

-Recopilación de datos obtenidos de [PATT11]-

Coherencia en caché

En un procesador **multinúcleo**, cada núcleo tiene su propia L1 y L2. Sin embargo, la L3 y la memoria principal es compartida por todos los núcleos.
¿Cómo logra el sistema mantener la coherencia en las actualizaciones de datos?