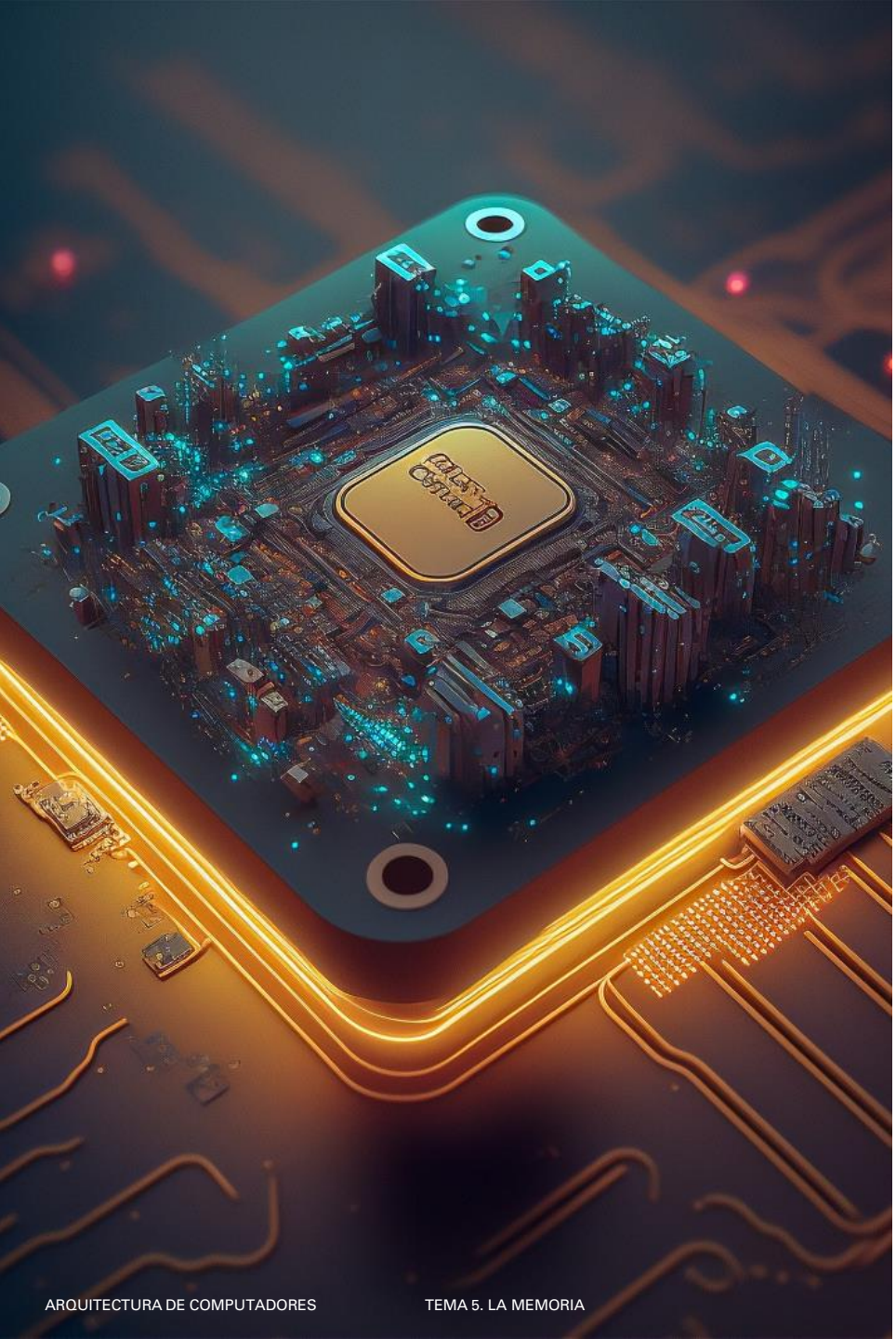


Tema 5

LA MEMORIA

Índice

1. Introducción

- Los usuarios desean memorias grandes, rápidas y baratas. Sin embargo:
  - a menor tiempo de acceso, mayor coste por bit
  - a mayor capacidad, menor coste por bit
  - a mayor capacidad, mayor tiempo de acceso
- Es necesario equilibrar el gasto, el tamaño y la velocidad de las memorias utilizadas
- Se utilizan diferentes tipos de memoria

Principio de localidad

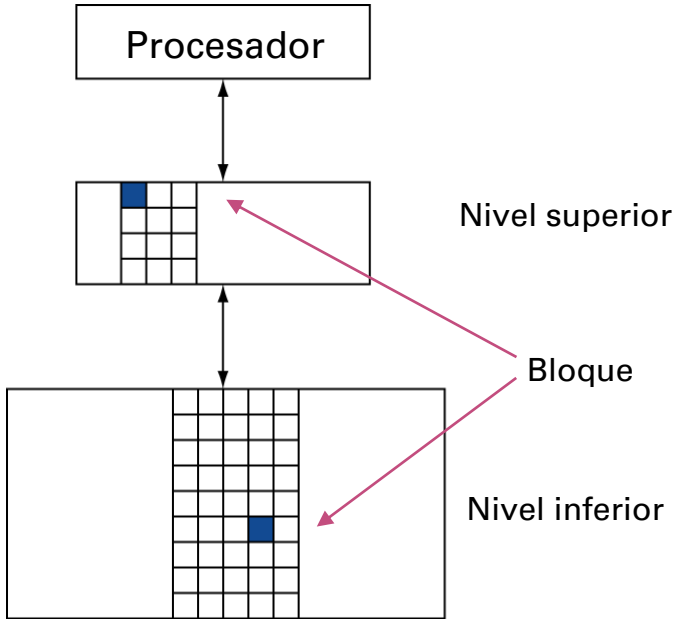
- Los programas acceden a una porción relativamente pequeña del espacio de direcciones en cualquier instante de tiempo. Dos tipos:
- Localidad temporal:
  - Si se referencia un elemento, tenderá a ser referenciado pronto
  - Por ejemplo: instrucciones dentro de un bucle
- Localidad espacial:
  - Los elementos cercanos al elemento referenciado tenderán a ser referenciados pronto
  - Por ejemplo, instrucciones de acceso secuencial, array de datos.

2. Jerarquía de memoria

- Según se desciende en la jerarquía:
  - Disminuye el coste por bit
  - Aumenta la capacidad
  - Aumenta el tiempo de acceso
  - Disminuye la frecuencia de accesos por parte de la CPU
- La información se ubica en un nivel dependiendo de su probabilidad de uso
- Como ésta varía durante la ejecución de un programa, se produce un tráfico continuo de información entre los niveles

Niveles de la jerarquía de memoria

- Una jerarquía de memoria consta de varios niveles, pero en cada momento se gestiona entre dos niveles.
  - **Nivel superior:** el más cercano a la CPU, corresponde a la memoria más rápida, pequeña y cara
  - **Nivel inferior:** el más alejado a la CPU, corresponde a la memoria más lenta, grande y barata.
- Todos los datos del nivel superior se encuentran también el nivel inferior.
- **Bloque:** la mínima unidad de información que se puede referenciar y copiar en la jerarquía de dos niveles.
  - Está compuesto de palabras de memoria

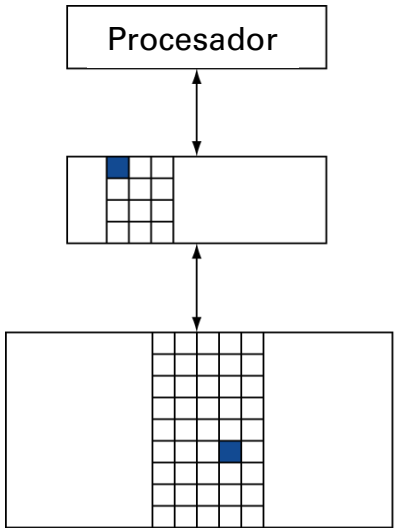


Acierto y Fallo

- Si el dato pedido por el procesador está presente en el nivel superior
  - **Acierto** (hit): el nivel superior proporciona el dato
  - Frecuencia o **tasa de aciertos** (hit ratio): porcentaje de aciertos en accesos a la memoria del nivel superior
- Si no está el dato al que se quiere acceder: **Fallo** (miss)
  - Hay que copiar el bloque desde el nivel inferior
  - Después el nivel superior proporciona el dato
  - **La tasa de fallos** (miss ratio) se define como

    1-tasa de aciertos

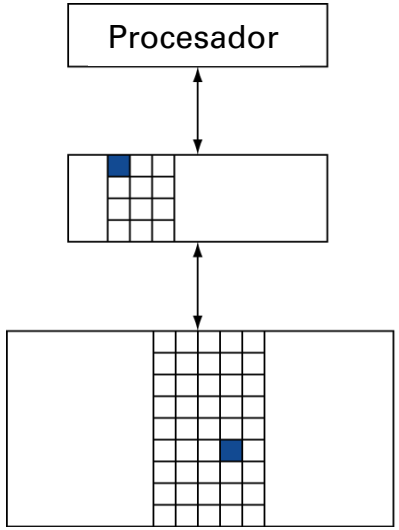
Principal razón de la jerarquía de memoria: rendimiento



La velocidad de aciertos y fallos es importante

Tiempo de acierto y Penalización de fallo

- **Tiempo de acierto** (**TA**) (Hit time): tiempo necesario para acceder al nivel superior cuando se produce un acierto. Incluye:
  - El tiempo necesario para determinar si ha sido acierto o fallo
  - El tiempo necesario para acceder a la información
- **Penalización de fallo** (**PF**) es el tiempo consumido en obtener la información cuando se produce un fallo. Incluye:
  - Tiempo empleado en sustituir un bloque del nivel superior por el bloque correspondiente del nivel inferior
  - Tiempo en proporcionar el bloque al dispositivo que lo ha pedido



3. Tecnologías de memoria

- **RAM Estática (SRAM).** Usadas en las Cachés.
- **RAM Dinámica (DRAM).** Usadas por la memoria principal.
- **Discos magnéticos**. Usados para implementar el nivel más grande y lento de la jerarquía.
- **Memoria Flash**. Utilizada en los dispositivos móviles personales.

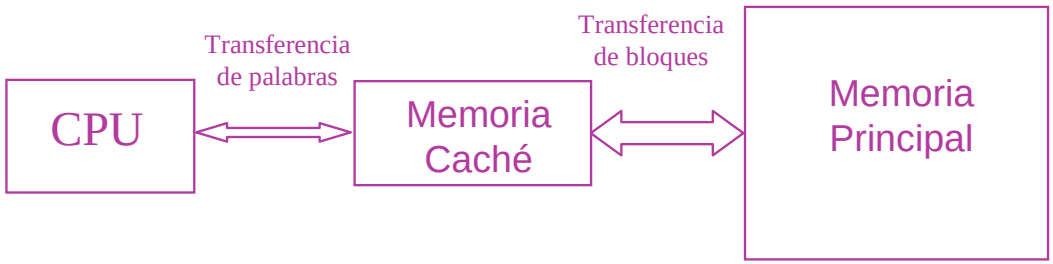
Tecnologías de memoria	Tiempo de acceso típico	$ por GB en 2020
SRAM	0.5 – 2.5ns	500-1000
DRAM	50-70ns	3-6
Memoria Flash	5000-50000ns	0.06-0.12
Discos magnéticos	5000000-20000000ns	0.01-0.02

- Memoria ideal**.**
  - Tiempo de acceso a la SRAM
  - Capacidad y coste por GB del disco

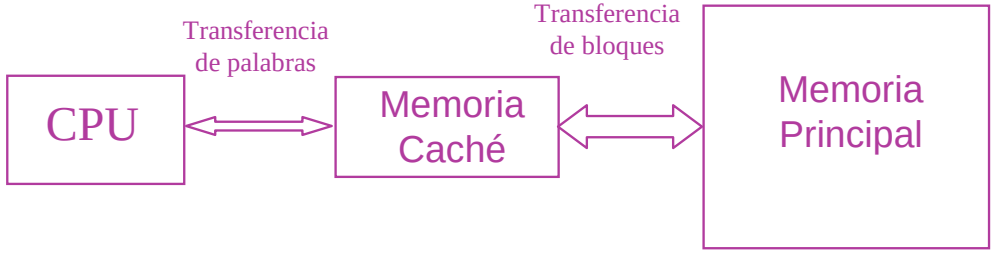
4. La memoria caché

- La memoria caché aprovecha el principio de localidad:
  - Es una memoria muy rápida colocada entre la CPU y la memoria principal
  - Contiene una copia de las posiciones de memoria principal que están siendo utilizadas por la CPU
  - Al espacio de la caché donde se copia el bloque se le denomina **Línea**

- **Funcionamiento**:



Operación de la caché



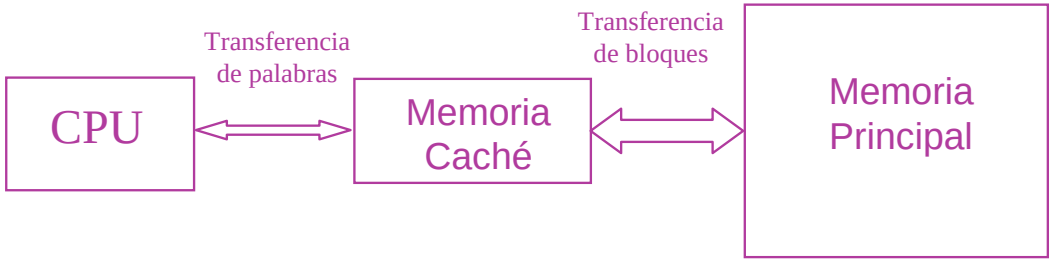
- Petición de lectura

- Éxito (hit) → se envía al procesador
- Fallo (miss) → traer de MP
  - Si caché llena → Política de reemplazo
  - Una copia de la palabra se almacena en la caché

- Petición de escritura

- Éxito → Realizar escritura
- Fallo →
  1. Traer copia palabra desde MP
  2. Realizar operación escritura
- Operación de escritura
  - Write-Through (Escritura Directa)
  - Write-Back (Postescritura)

Cuestiones que resolver



- Ubicación de un bloque. ¿Dónde se ubica un bloque en la caché?
- Identificación de un bloque. ¿Cómo se encuentra un bloque en la caché?
- Reemplazamiento. ¿Qué bloque de la caché se elimina ante un fallo?
- Política ante escrituras. ¿Qué se hace ante una escritura?

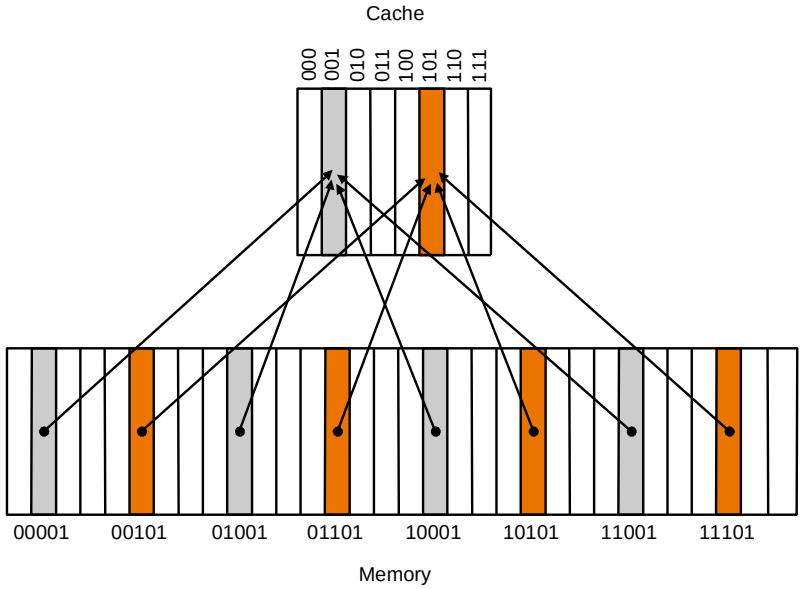
¿Dónde se ubica un bloque en la caché?

- **Función de Correspondencia**: Traducir direcciones de memoria generadas por la CPU en posiciones de palabras en la caché.
- Basado en este proceso, la organización de la caché se clasifica en:
  - Correspondencia directa
  - Correspondencia completamente asociativa
  - Correspondencia asociativa por conjuntos

Correspondencia directa

- **A un bloque de la MP le corresponde un único bloque en la caché**
- Correspondencia:

  dirección del bloque o línea = (número del bloque) modulo (bloques en la caché)



- Dirección de memoria:
  - Etiqueta: Dirección de la estructura del bloque
  - Índice: bits para la selección del bloque o línea en la caché
  - Desplazamiento: palabra dentro del bloque

Dirección de la estructura de bloque		Dirección del desplazamiento de bloque
0111	01110010	1001
Etiqueta	Índice	Desplazamiento

Etiqueta y bit de validez

- ¿Cómo saber si un bloque en particular se encuentra almacenado en una ubicación de caché?
  - Al almacenarlo en una línea de la caché lo etiquetamos para diferenciarlo del resto de bloques que podrían estar en la misma línea
  - Sólo se necesitan los bits de orden superior de la dirección
- ¿Cómo saber si un bloque de la caché contiene información válida?
  - Añadir Bit de validez: 1= hay información, 0= no hay información (inicialmente a 0)

Ejemplo de acceso a la Caché

- 8 bloques, 1palabra por bloque, correspondencia directa
- Estado inicial de la caché:

Índice	Bit validez	Etiqueta	Dato
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

Ejemplo de acceso a la Caché

Dirección palabra en decimal	Dirección en binario	Éxito/Fallo en Caché	Bloque caché
22	10 110	Fallo	(10110 mod 8)=110

Índice	Bit validez	Etiqueta	Dato
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	**S**	**10**	**Mem[10110]**
111	N		

Ejemplo de acceso a la Caché

Dirección palabra en decimal	Dirección en binario	Éxito/Fallo en Caché	Bloque caché
26	11 010	Fallo	(11010 mod 8)=010

Índice	Bit validez	Etiqueta	Dato
000	N		
001	N		
010	**S**	**11**	**Mem[11010]**
011	N		
100	N		
101	N		
110	S	10	Mem[10110]
111	N		

Ejemplo de acceso a la Caché

Dirección palabra en decimal	Dirección en binario	Éxito/Fallo en Caché	Bloque caché
22	10 110	Éxito	110
26	11 010	Éxito	010

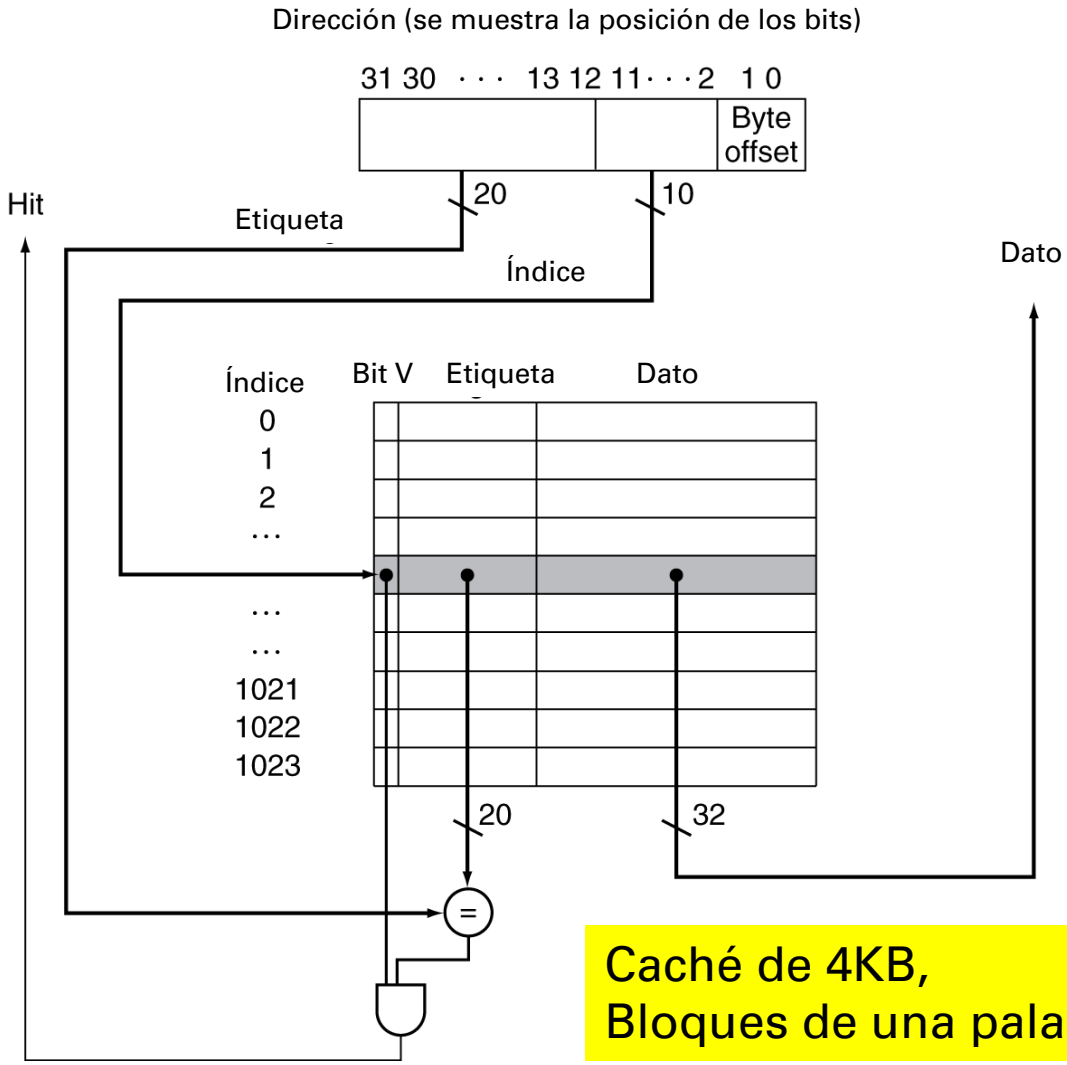
Índice	Bit validez	Etiqueta	Dato
000	N		
001	N		
010	S	11	Mem[11010]
011	N		
100	N		
101	N		
110	S	10	Mem[10110]
111	N		

Ejemplo de acceso a la Caché

Dirección palabra en decimal	Dirección en binario	Éxito/Fallo en Caché	Bloque caché
16	10 000	Fallo	000
3	00 011	Fallo	011
16	10 000	Éxito	000

Índice	Bit validez	Etiqueta	Dato
000	**Y**	**10**	**Mem[10000]**
001	N		
010	S	11	Mem[11010]
011	**Y**	**00**	**Mem[00011]**
100	N		
101	N		
110	S	10	Mem[10110]
111	N		

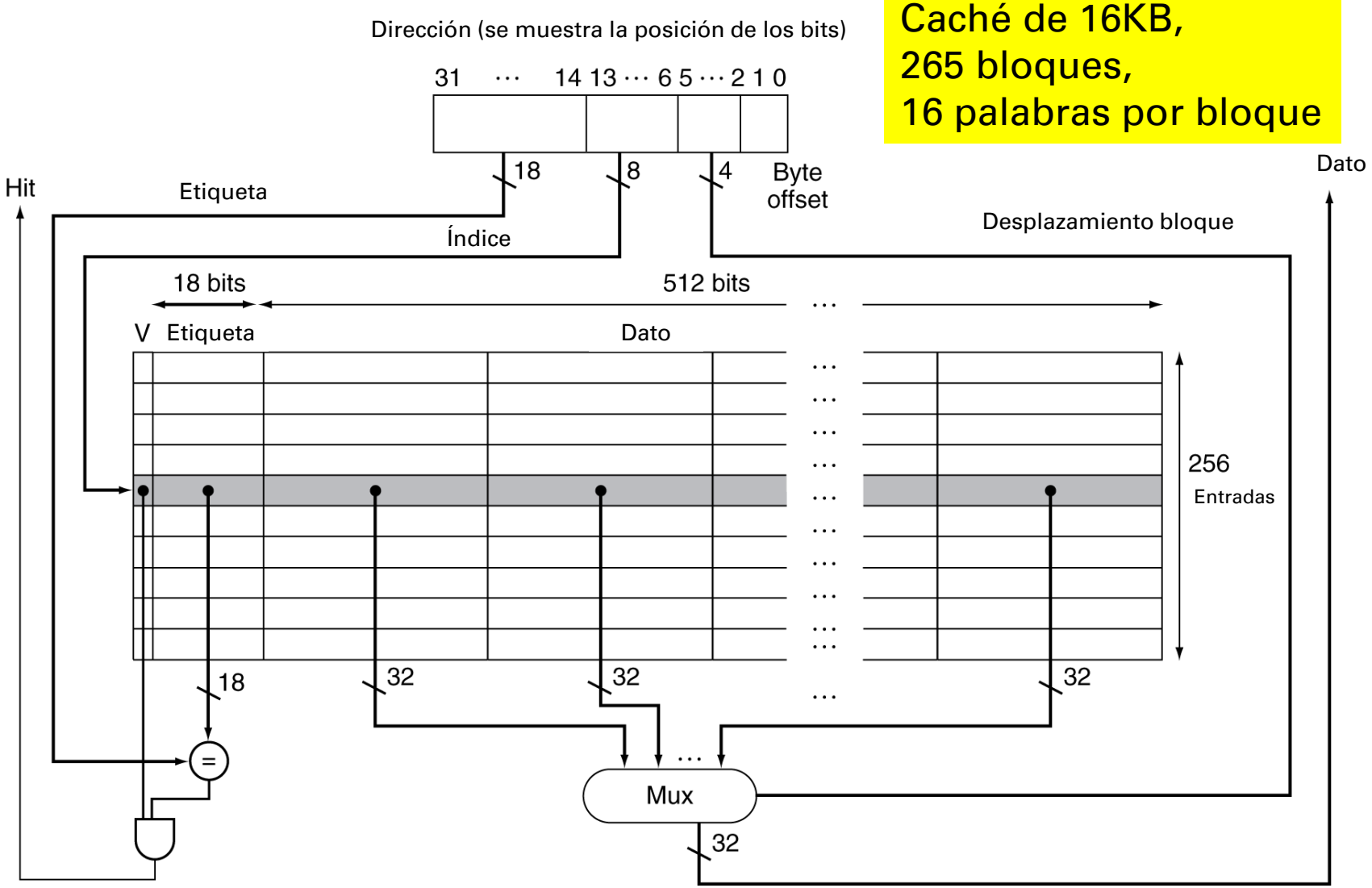
Ejemplo organización interna



Caché de 4KB,
Bloques de una palabra (4bytes)

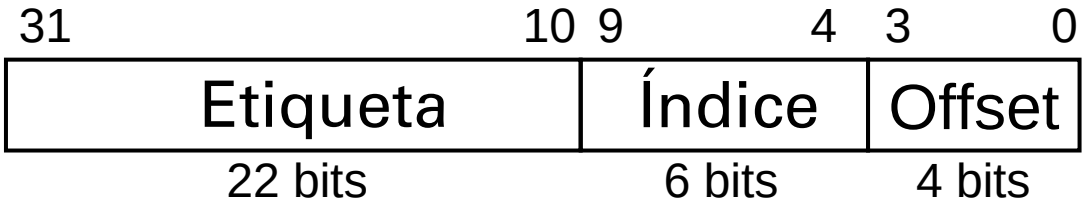
Ejemplo organización interna(FastMATH)

Caché de 16KB,
265 bloques,
16 palabras por bloque



Ejemplo

- Suponer una caché con 64 bloques y un tamaño de bloque de 16 bytes. ¿A qué número de bloque le corresponde la dirección del byte 1200?
  - $Dirección\ del\ bloque = \frac{Dirección\ del\ byte}{Bytes\ por\ bloque} = \frac{1200}{16} = 75$
  - $Número\ del\ bloque = 75\ mod\ 64 = 11$
  - En realidad, este bloque contiene todas direcciones entre 1200 y 1215



Consideraciones sobre el tamaño del bloque

- Bloques más grandes deberían reducir la tasa de fallos
  - Debido a la localidad espacial
- Pero en cachés de tamaño fijo
  - Bloques más grandes $\Rightarrow$ menor cantidad de ellos
    - Más competición para ocuparlos $\Rightarrow$ incrementa la tasa de fallos
  - Bloques más grandes $\Rightarrow$ Disminuye principio de localidad espacial
- La penalización de fallo es más grande
  - Puede invalidar el beneficio de reducir la tasa de fallos
  - Hay técnicas para para suavizarlo (*early restart* y *critial-word-first*)

Lectura en la caché

- Si acierto en caché, la CPU opera normalmente
- Si fallo en la caché
  - Parada en la segmentación de la CPU
  - Se busca el bloque en el siguiente nivel de la jerarquía
  - Si fallo en la caché de instrucciones
    - Reanudar la búsqueda de la instrucción (PC - 4)
  - Si fallo en la caché de datos
    - Completar el acceso al dato

Escritura en la caché

- Si acierto en caché, podría simplemente actualizar el bloque en la caché
  - Pero entonces la caché y la memoria serían incoherentes
- **Escritura directa (*write-through*):** también se actualiza la memoria principal
- Pero las escrituras serían más lentas
  - Ejemplo: si el CPI base=1, el 10% de las instrucciones son almacenamientos y la escritura en memoria tarda 100 ciclos entonces:
    - CPI efectivo = 1+0,1x100 = 11
- Solución: buffer escritura
  - Mantiene los datos a la espera de escribirse en la memoria
  - La CPU continua inmediatamente
    - Solo para en escrituras si el buffer ya está lleno

Escritura en la caché

- Alternativa: **Postescritura (write-back)** si acierto en escritura, simplemente actualizar el bloque en la caché
  - Se requiere un bit adicional: bit MODIFICADO (*dirty bit*)
- Cuando un bloque *modificado* se reemplaza
  - Escribirlo en la memoria
  - Se puede usar un búfer de escritura para permitir que el bloque de reemplazo se lea primero

Fallo en escritura en la caché

- Ubicar en escritura (write-allocate)
  - Gestión del fallo similar a un fallo de lectura
  - Se trae el bloque y se escribe el dato (escritura directa o postescritura)
- No ubicar en escritura (write-non-allocate)
  - Se escribe el dato en el nivel inferior no en la caché
  - Solo se utiliza en escritura directa

Rendimiento en la caché

- El tiempo de ejecución CPU puede dividirse en:
  - Ciclos de reloj que tarda en ejecutar el programa (Incluye el tiempo de acierto a la caché)
  - Ciclos de reloj de parada esperando a la memoria (Principalmente debidos a los fallos de la caché)

$$Tiempo\ CPU = (Ciclos\ CPU + Ciclos\ Parada) \times Tiempo\ ciclo$$

$$Ciclos\ Parada = \frac{Accesos\ a\ memoria}{Programa} \times Tasa\ de\ Fallos \times Penalización\ Fallos$$

- Dos formas de mejorar el rendimiento:
  - Reducir la tasa de fallos: organizaciones más eficientes de la caché
  - Reducir la penalización de fallos:
    - Cachés multinivel (2 y hasta 3 niveles: L1, L2, L3)
    - Buses más anchos entre niveles de memoria caché

Ejemplo

- Suponer:
  - Tasa fallos de la caché de instrucciones = 2%
  - Tasa de fallos de la caché de datos = 4%
  - El procesador tiene un CPI de 2 sin paradas en caché (caché ideal)
  - Penalización de fallos = 100 ciclos
  - La frecuencia de las Instrucciones Load y Store es de 36%
  - Obtén cuánto más rápido es el procesador ejecutándose en una cache sin fallos

- Número de ciclos por instrucción debidos a fallos en la memoria:
  - Caché de instrucciones: 2% x 100 = 2
  - Caché de Datos: 36% x 4% x 100 = 1.44
  - Número total de ciclos de parada= 2 + 1.44 = 3.44
- CPI real = 2 + 3.44 = 5.44
- La CPU ideal es 5.44/2=2.72 veces más rápida

Tiempo medio de acceso a memoria

- Como el Tiempo de Acierto (TA) también es importante para el rendimiento:
- Tiempo medio de acceso a la memoria (TMA) se calcula como:
  - TMA = Tiempo Acierto + Tasa fallos x Penalización de fallos

    TMA = TA + TF x PF

- **Ejemplo**: Calcular el TMA para un procesador con un tiempo de ciclo de 1ns, una penalización de fallos de 20 ciclos, una tasa de fallos de 0.05 fallos por instrucción y una caché con tiempo de acceso (incluyendo la detección de acierto) de 1ciclo. Suponer que la penalización de fallos de lectura y escritura son la misma e ignorar también cualquier otra parada de escritura.
  - TMA = 1+ 0.05 x 20 = 2ns
    - 2 ciclos por instrucción

Conclusiones sobre rendimiento

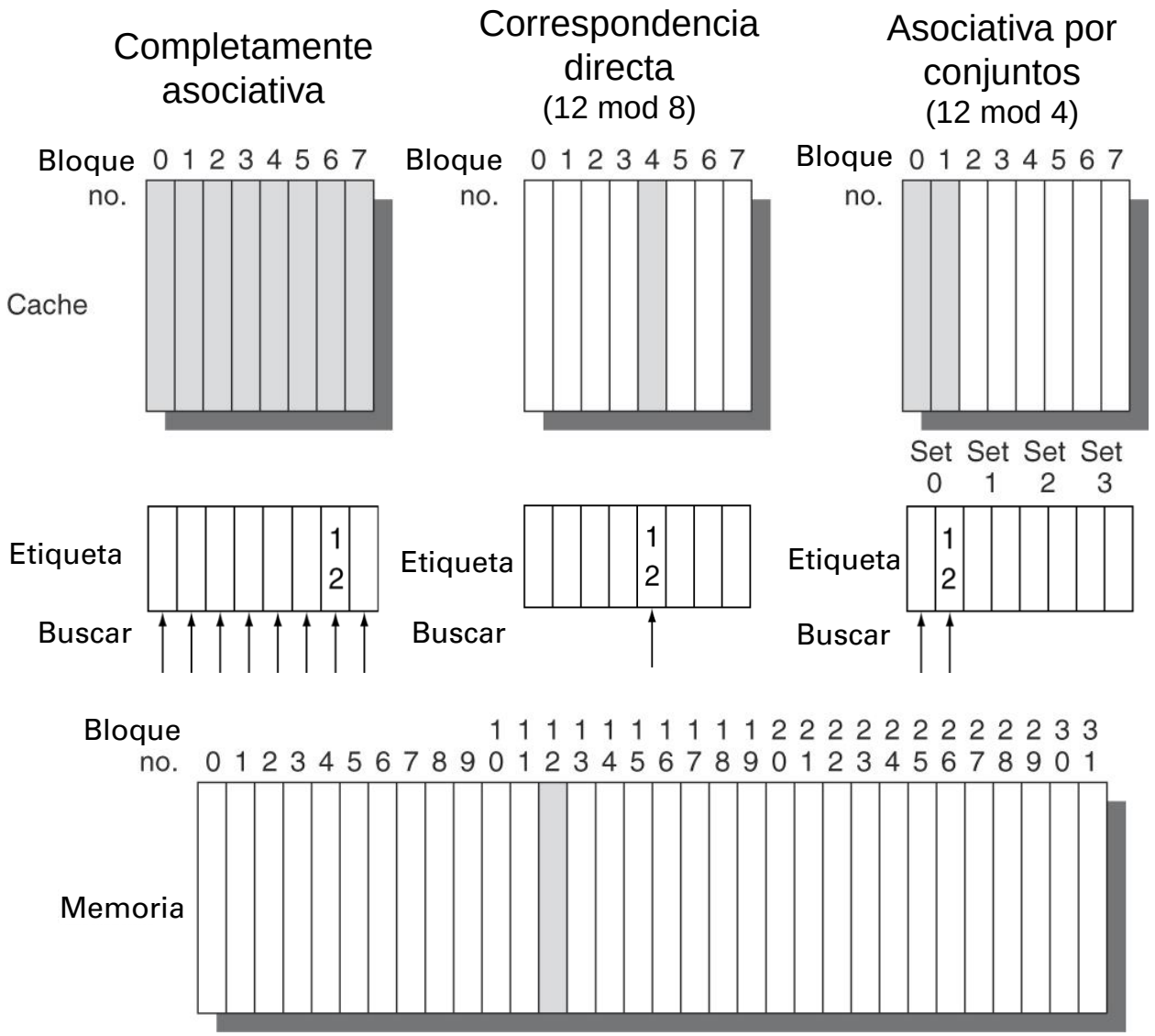
- Cuando el rendimiento de la CPU aumenta
  - La penalización por fallos se vuelve más significativo
- A medida que el CPI disminuye:
  - Más grande es la proporción de tiempo que se dedica a las paradas de memoria
- Al aumentar la frecuencia de reloj
  - Las paradas de memoria representan más ciclos de CPU
- No se puede descuidar el comportamiento de la caché al evaluar el rendimiento del sistema

Cachés asociativas

- **Correspondencia completamente asociativa:**
  - Un bloque se puede ubicar en cualquier línea de la caché
  - Para encontrar un bloque en la caché es necesario una búsqueda de todas las líneas a la vez
  - Búsqueda en paralelo con un comparador por cada línea (caro)
  - Dirección memoria

- **Correspondencia asociativa por conjuntos de N vías:**
  - Cada conjunto contiene N líneas
  - A un bloque de MP le corresponde un conjunto de la caché, pudiéndose ubicar en cualquier línea del conjunto:
    - Dirección conjunto= (número del bloque) módulo (Número Conjuntos en la caché)
  - Dirección memoria: como en la correspondencia directa. El campo índice selecciona el conjunto
  - Búsqueda de todas las líneas en un conjunto al mismo tiempo
  - N comparadores (menos caro)

Ejemplo de Caché asociativa



Ejemplo para una caché de 8 bloques

Asociativa por conjuntos de una vía (Correspondencia directa)

Bloque	Tag	Data
0		
1		
2		
3		
4		
5		
6		
7		

Asociativa por conjuntos de 2 vías

Conjunto	Tag	Data	Tag	Data
0				
1				
2				
3				

Asociativa por conjuntos de 4 vías

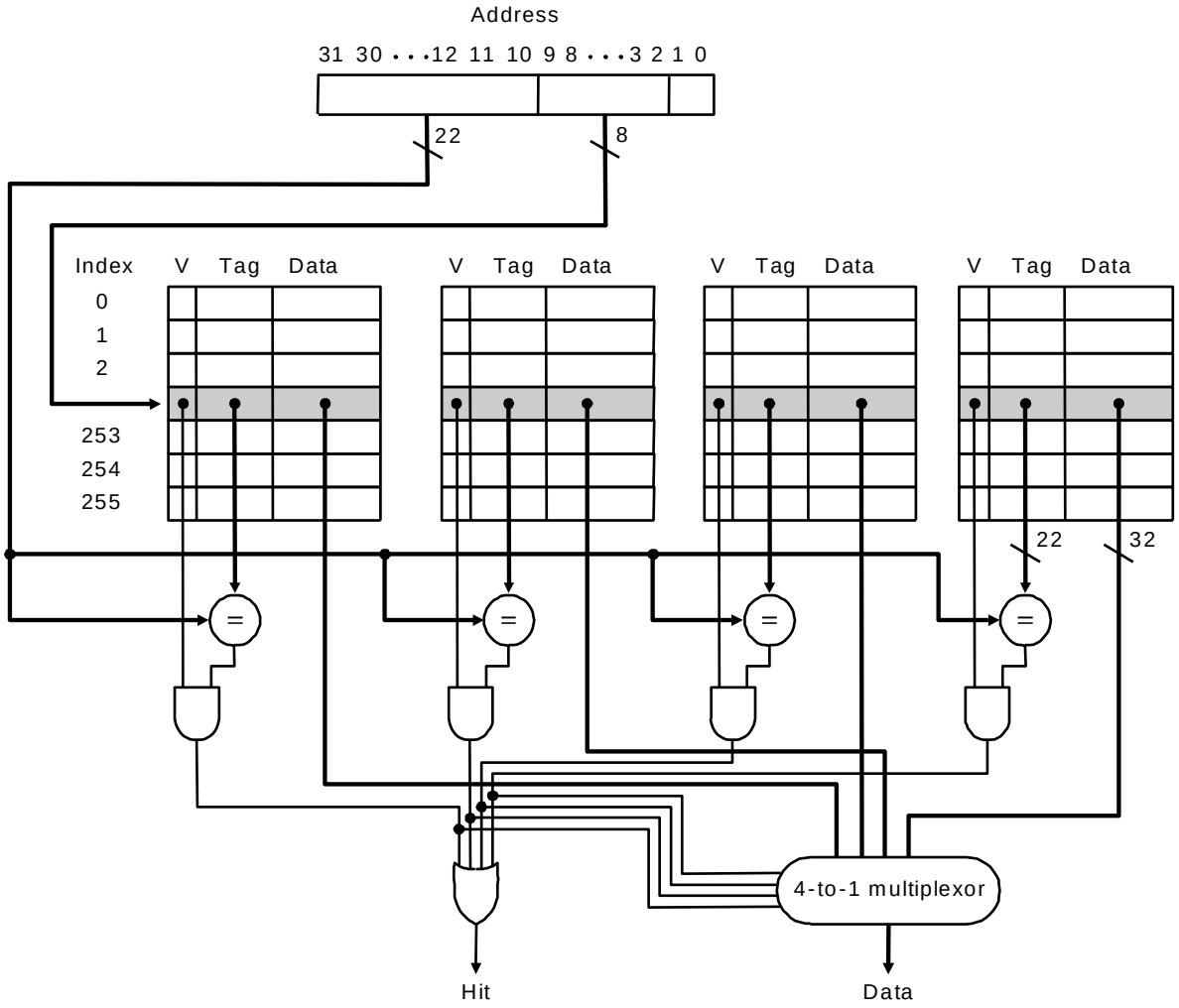
Conjunto	Tag	Data	Tag	Data	Tag	Data	Tag	Data
0								
1								

Asociativa por conjuntos de 8 vías (Completamente asociativa)

Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data

Ejemplo de implementación

- Caché asociativa por conjuntos de 4 vías



Ejemplo acceso según tipo de asociatividad

- Comparar el acceso en una caché de 4 bloques en:
  - Correspondencia directa, Asociativa por conjunto de 2 vías y completamente asociativa
  - Suponer el acceso a la secuencia de bloques: 0, 8, 0 , 6, 8
- **Correspondencia directa**

Dirección bloque	Índice caché	Acierto/ Fallo	Contenido de la caché después del acceso			
			0	1	2	3
0	0	fallo	**Mem[0]**			
8	0	fallo	**Mem[8]**			
0	0	fallo	**Mem[0]**			
6	2	fallo	Mem[0]		**Mem[6]**	
8	0	fallo	**Mem[8]**		Mem[6]	

Ejemplo acceso según tipo de asociatividad

- **Asociativa por conjuntos de 2 vías**

Dirección bloque	Índice caché	Acierto/ Fallo	Contenido de la caché después del acceso			
			Conjunto 0		Conjunto 1	
0	0	fallo	**Mem[0]**			
8	0	fallo	Mem[0]	**Mem[8]**		
0	0	Acierto	**Mem[0]**	Mem[8]		
6	2	fallo	Mem[0]	**Mem[6]**		
8	0	fallo	**Mem[8]**	Mem[6]		

- **Completamente asociativa**

Dirección bloque	Índice caché	Acierto/ Fallo	Contenido de la caché después del acceso			
			0	1	2	3
0	0	fallo	**Mem[0]**			
8	0	fallo	Mem[0]	**Mem[8]**		
0	0	Acierto	**Mem[0]**	Mem[8]		
6	2	fallo	Mem[0]	Mem[8]	**Mem[6]**	
8	0	Acierto	Mem[0]	**Mem[8]**	Mem[6]	

¿Qué correspondencia elegir?

- Al incrementar la asociatividad disminuye la tasa de fallos
  - Pero con rendimientos decrecientes

- Ejemplo: Simulación de un sistema con 64KB de caché de datos, bloques de 16 palabras. Resultados con SPEC2000:
  - 1-vía (Correspondencia directa): 10.3%
  - 2-vías: 8.6%
  - 4-vías: 8.3%
  - 8-vías: 8.1%
  - Esta mejora es mayor al pasar de correspondencia directa a Cache asociativa de 2 vías, con mayor asociatividad ya no se obtiene tanta mejora.
  - Las cachés más pequeñas obtienen mayor mejora ya que la tasa de fallos es mayor

Algoritmos de reemplazo

- Correspondencia directa: sólo hay una posible línea para cada bloque
- Para las técnicas asociativas (implementados por hardware)
  - Método LRU (Menos recientemente utilizado). Se sustituye el bloque que hace más tiempo que no fue referenciado
    - Fácil de implementar para asociativas por conjuntos de dos vías
  - Método aleatorio. Se selecciona un bloque al azar para ser reemplazado
    - Implementación hardware sencilla, mismo rendimiento que LRU para gran asociatividad
  - Método LFU (Menos frecuentemente utilizado) Reemplaza el bloque menos usado, asumiendo que apenas se necesita.
  - Método FIFO (primero en entrar primero en salir) Se reemplaza el bloque que hace más tiempo que está en caché

Múltiples niveles de la caché

- La caché de primer nivel junto a la CPU
  - Pequeña y muy rápida
- Caché de nivel 2 (L2) solventa los fallos de la caché de primer nivel
  - Más grande y más lenta que la L1 pero más rápida que la memoria principal
- La memoria principal solventa los fallos de la caché L2
- Algunos sistemas incluyen caché L3

Ejemplo de caché multinivel

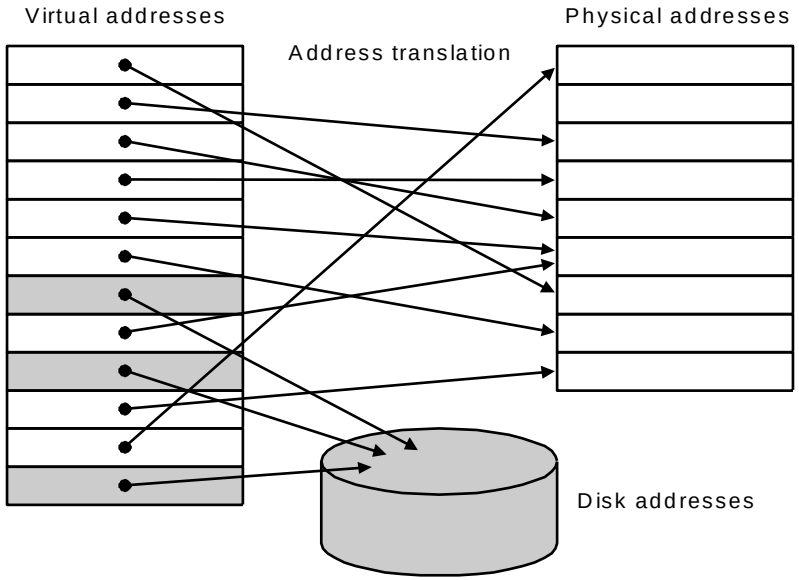
- Supongamos:
  - CPI de 1.0 en una máquina de 4Ghz con una tasa de fallos del 2%, el tiempo de acceso a la DRAM es de 100ns
- Con solo una caché de primer nivel:
  - La penalización de fallos = 100ns/0,25 ns =400ciclos
  - CPI efectivo = 1 + 0,02 x400 = 9
- Al añadir una caché de 2º nivel con un tiempo de acceso de 5 ns, y una tasa de fallos a la memoria principal del 0,5%
- Fallos primarios con acierto a L2
  - Penalización = 5ns/0,25ns =20 ciclos
- Fallos primarios con fallos a la L2
  - Penalización extra =500 ciclos
- CPI = 1+ 0,02x20 +0.005 x 400 = 3.4
- Ganancia en rendimiento = 9/3.4=2.6

Consideraciones de las cachés multinivel

- Caché de primer nivel
  - Se intenta optimizar el tiempo de éxito
  - caché pequeña normalmente con un tamaño de bloque menor
- Caché de 2º nivel
  - Se intenta optimizar la tasa de fallos para evitar los accesos a la memoria principal
  - El tiempo de éxito tiene menos impacto global
  - Caché más grande con un tamaño de bloque mayor
- Resumen
  - Caché L-1 normalmente más pequeña que una única caché
  - El tamaño del bloque en la L-1 más pequeño que el tamaño del bloque de la L-2

5. La memoria virtual

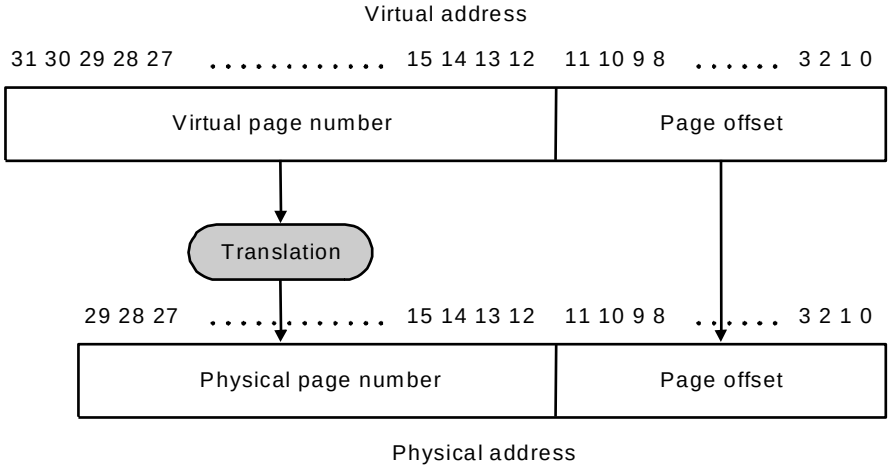
- La memoria principal “actúa” como una caché para el almacenamiento secundario (disco)



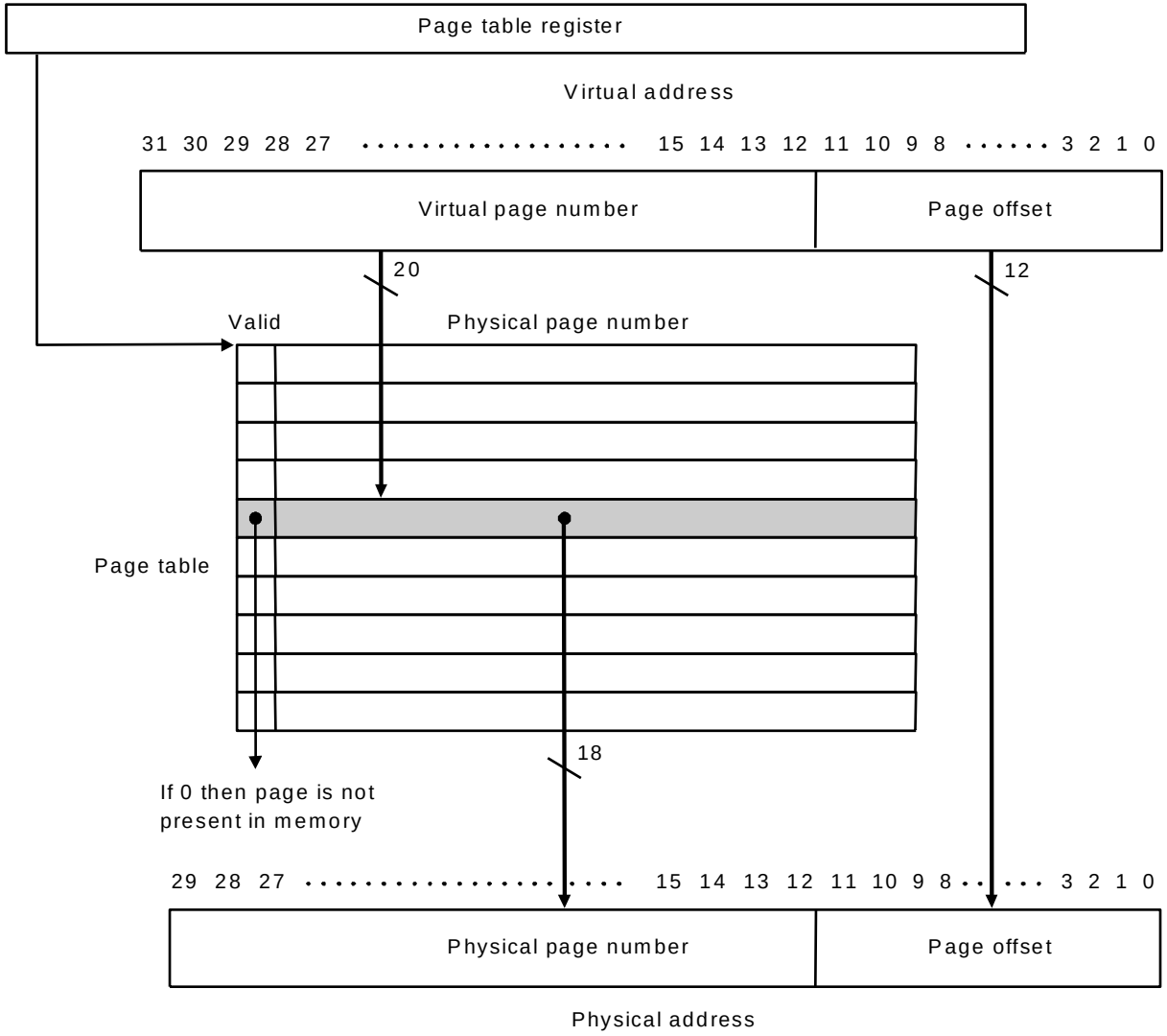
- Ventajas
  - Ilusión de tener más memoria física
  - Reubicación de programas
  - protección

Páginas: bloques de memoria virtual

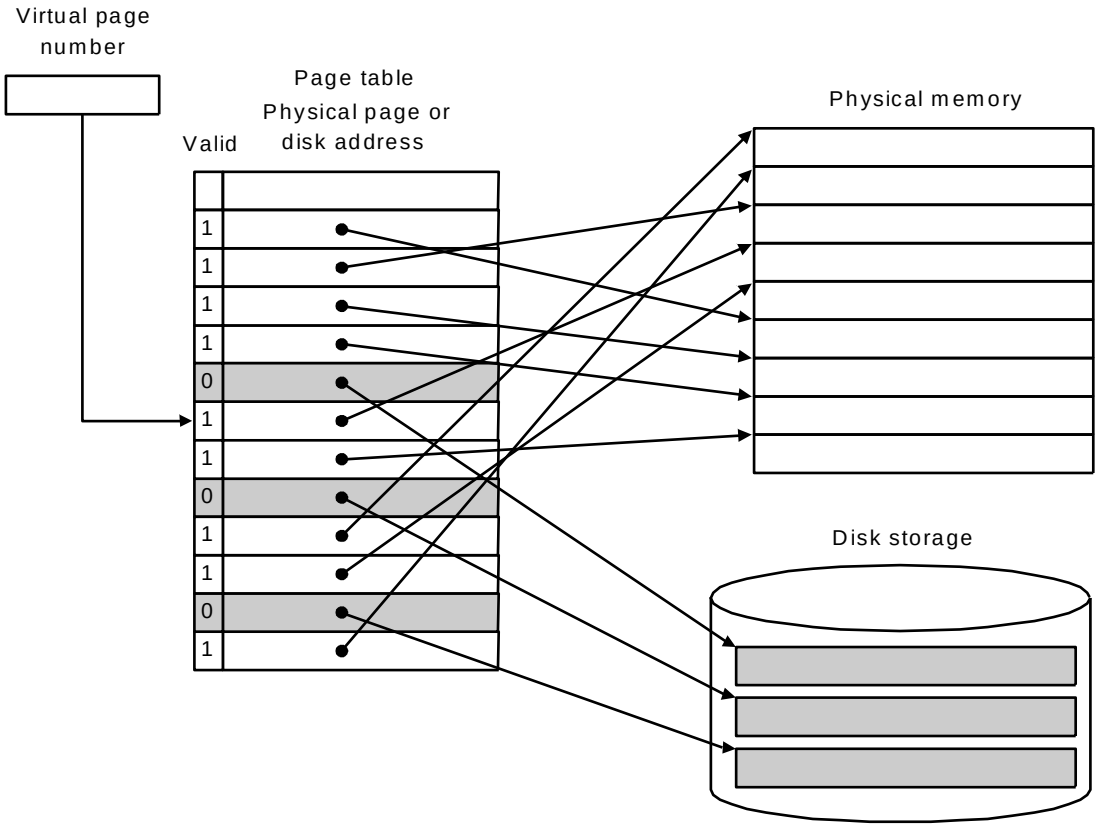
- Fallos de página: El dato no está en la memoria, hay que traerlo del disco
  - Penalización de fallo muy grande, por tanto, las páginas deberían ser bastante grandes (ej., 4KB a 16KB, nuevos sistemas: 32KB y 64KB)
  - Reducir los fallos de página es importante (se permite la asignación completamente asociativa)
  - Se pueden manejar los fallos por software en lugar de por hardware
  - Usar escritura directa (write-through) es demasiado caro por tanto se usará postescritura (writeback)



Tablas de página



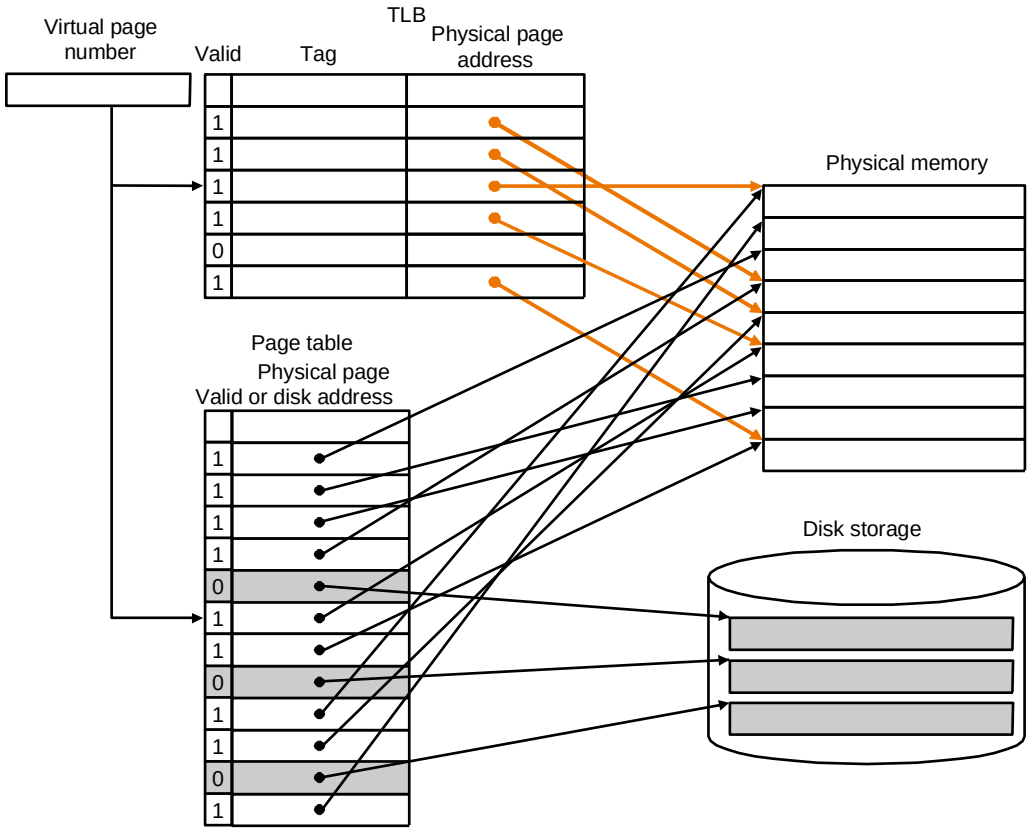
Asignación de páginas al almacenamiento



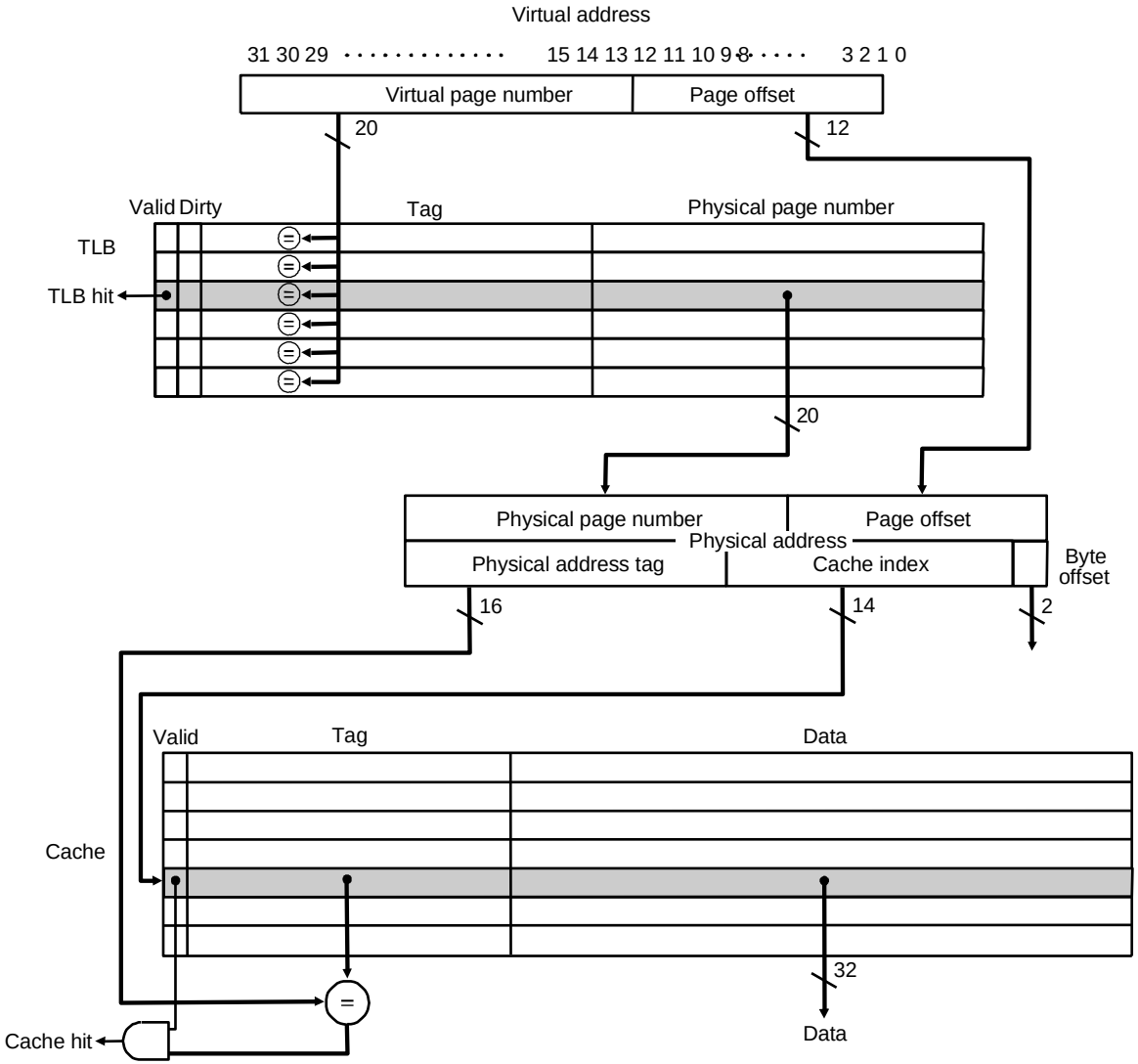
Traducción rápida de direcciones

- Disponer de una caché para la traducción de direcciones:

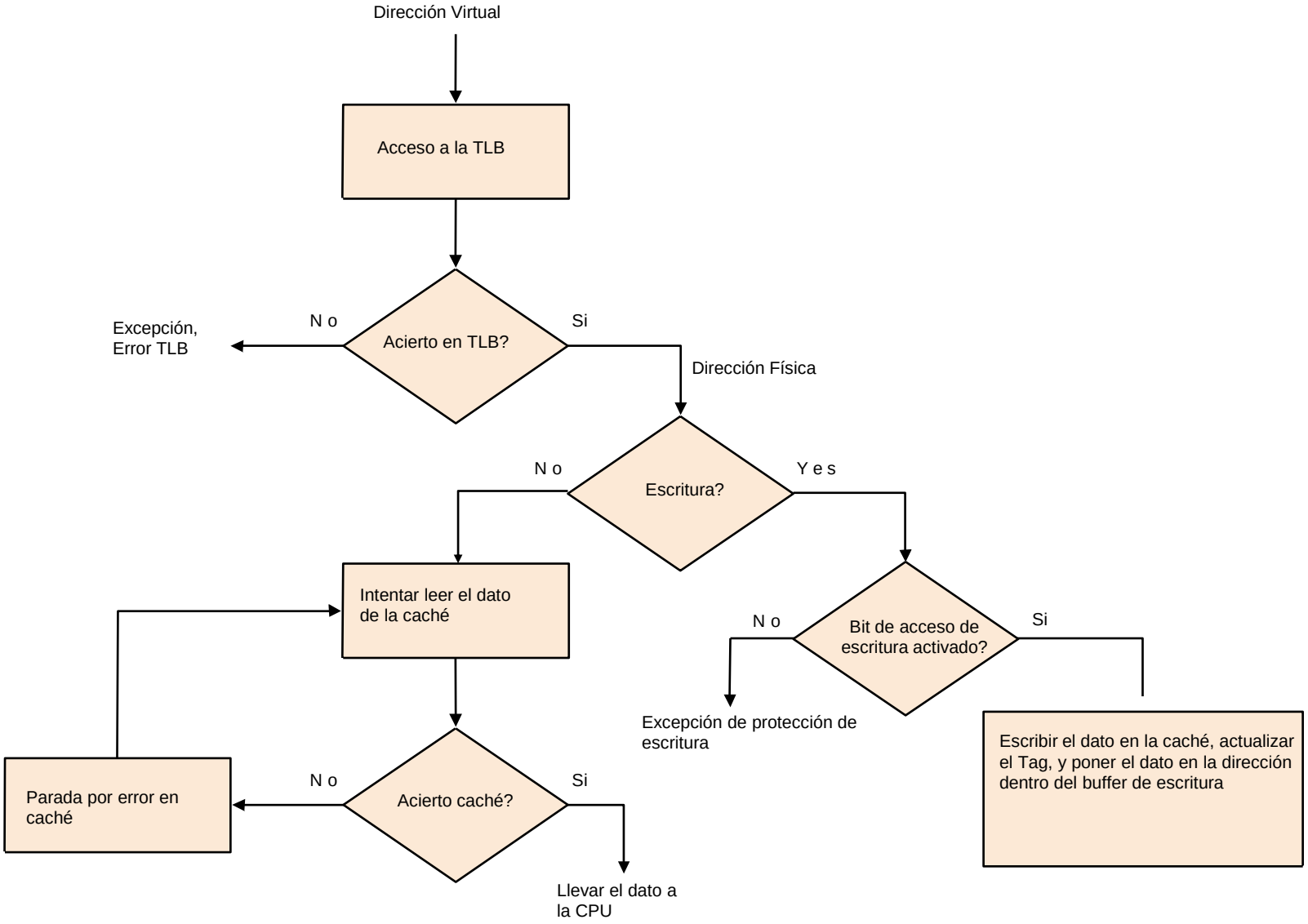
  translation lookaside buffer (TLB)



Memoria virtual, TLB y caché



TLBs y cachés



Cachés multinivel on-chip

Characteristic	ARM Cortex-A53	Intel Core i7
L1 cache organization	Split instruction and data caches	Split instruction and data caches
L1 cache size	Configurable 16 to 64 KiB each for instructions/data	32 KiB each for instructions/data per core
L1 cache associativity	Two-way (I), four-way (D) set associative	Four-way (I), eight-way (D) set associative
L1 replacement	Random	Approximated LRU
L1 block size	64 bytes	64 bytes
L1 write policy	Write-back, variable allocation policies (default is Write-allocate)	Write-back, No-write-allocate
L1 hit time (load-use)	Two clock cycles	Four clock cycles, pipelined
L2 cache organization	Unified (instruction and data)	Unified (instruction and data) per core
L2 cache size	128 KiB to 2 MiB	256 KiB (0.25 MiB)
L2 cache associativity	16-way set associative	8-way set associative
L2 replacement	Approximated LRU	Approximated LRU
L2 block size	64 bytes	64 bytes
L2 write policy	Write-back, Write-allocate	Write-back, Write-allocate
L2 hit time	12 clock cycles	10 clock cycles
L3 cache organization	–	Unified (instruction and data)
L3 cache size	–	8 MiB, shared
L3 cache associativity	–	16-way set associative
L3 replacement	–	Approximated LRU
L3 block size	–	64 bytes
L3 write policy	–	Write-back, Write-allocate
L3 hit time	–	35 clock cycles