Tema 4

Jerarquías de memoria

Tecnología de memoria

- **RAM estática (SRAM)**
  - 0.5ns – 2.5ns, 2000 – 5000$ por GB
- **RAM dinámica (DRAM)**
  - 50ns – 70ns, 20 – 75$ por GB
- **Memoria “flash”**
  - 90ns – 150ns, 2 – 3$ por GB
- **Disco magnético**
  - 5ms – 20ms, 0.20 – 2$ por GB
- **Memoria ideal**
  - Tiempo de acceso de una SRAM
  - Capacidad y coste/GB del disco magnético

El principio de localidad

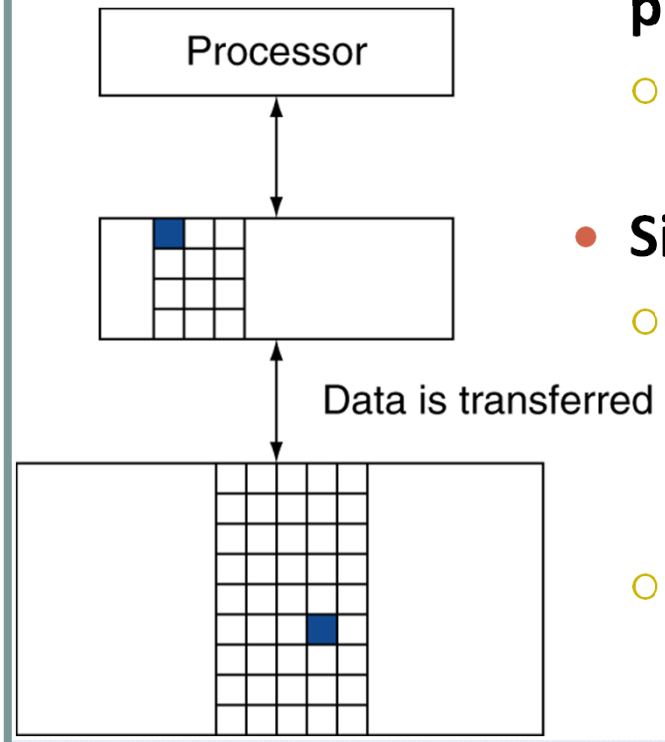
- **Los programas acceden a una pequeña proporción de su espacio de direcciones en cada momento.**
- **Localidad temporal**
  - Es más probable que en el futuro se vuelva a referenciar los mismos items que han sido recientemente referenciados.
  - Por ejemplo, es lo que pasa en las instrucciones de un lazo o en las variables índice de los mismos.
- **Localidad espacial**
  - Es más probable que en el futuro sean referenciados los items que están cerca de aquéllos que hayan sido recientemente referenciados.
  - Por ejemplo, el acceso secuencial a las instrucciones, el acceso a los arrays de datos.

Aprovechar la localidad

Jerarquía de memoria:

- Todo está en el disco
- Se copian los items a los que se ha accedido hace poco (y sus alrededores) desde el disco a una memoria DRAM más pequeña
  - Memoria principal
- Se copian los items a los que más recientemente se ha accedido (y sus alrededores) desde la DRAM hacia una memoria SRAM más pequeña
  - Memora caché enganchada a la CPU

Niveles de la jerarquía de memoria



- **Bloque (o línea): es la unidad de copia**
  - Puede contener varias palabras
- **Si el dato al que se quiere acceder está presente en el nivel superior:**
  - Acierto: el acceso es resuelto en el nivel superior
    - Tasa de acierto: aciertos/accesos
- **Si el dato al que se quiere acceder no está:**
  - Fallo: un bloque será copiado desde el nivel que está debajo
    - El tiempo invertido en ello: penalización de fallo
    - Tasa de fallo: fallos/accesos = 1 – tasa de aciertos
  - Después el nivel superior accede al dato proporcionado

La memoria caché

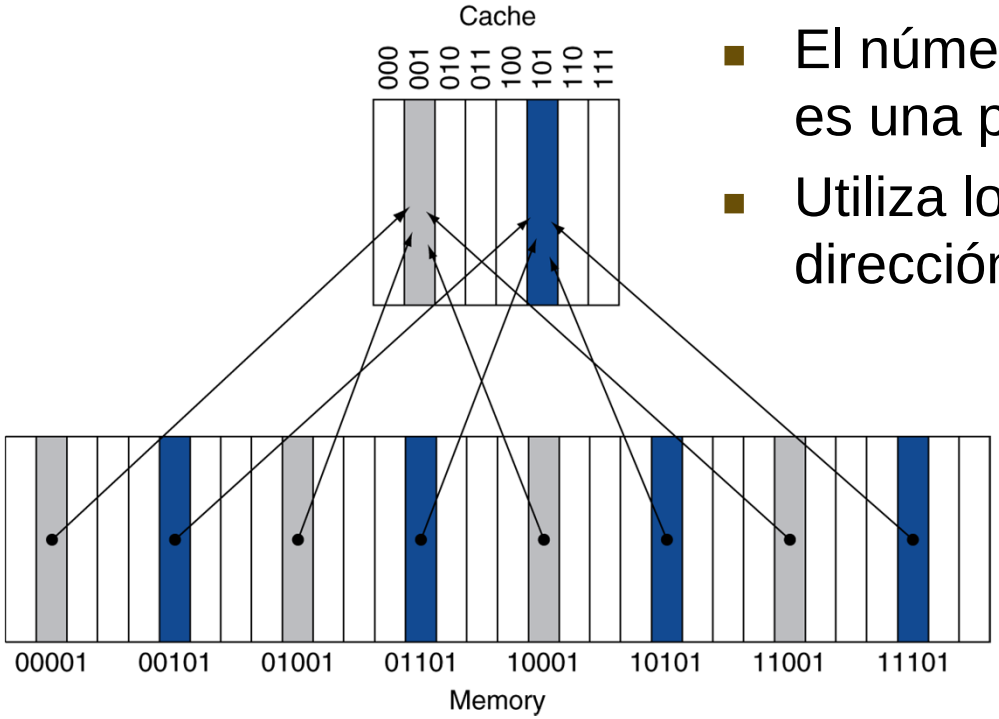
- **Memoria caché**
  - Es el nivel de la jerarquía de memoria que está más próximo a la CPU
- **Dados los accesos** $X_1, X_2, X_3, X_4, \ldots, X_{n-2}, X_{n-1}, X_n$

X_4
X_1
X_{n-2}
X_{n-1}
X_2
X_n
X_3

- ¿Cómo podemos saber si un dato está presente en la caché?
- ¿Dónde lo buscamos?

Caché de correspondencia directa

- **Ubicación determinada por la dirección**
- **Correspondencia directa: una sola posibilidad**
  - (Dirección de bloque) módulo (número de bloques en cache)



- El número de bloques de la caché es una potencia de 2
- Utiliza los bits más bajos de la dirección para determinar el bloque

Etiquetas y bits de validez

- **¿Cómo sabemos que un bloque concreto está almacenado en una ubicación de la caché?**
  - Se almacena la dirección y los datos
  - Realmente, sólo se necesita almacenar los bits más altos de la dirección
  - A eso se denomina “etiqueta”
- **¿Qué pasa si no hay datos en una ubicación?**
  - Bit de validez: 1=presente, 0=no presente
  - Iniciamente, todos los bits de validez están a 0

Ejemplo de funcionamiento de una memoria caché (1)

- **8 bloques, 1 palabra/bloque, correspondencia directa**

Peticiones

Dirección de palabra	Dirección binaria	Bloque de caché	Acierto o fallo

Memoria caché

Índice	V	Etiqueta	Datos
000	0		
001	0		
010	0		
011	0		
100	0		
101	0		
110	0		
111	0		

Estado inicial

Ejemplo de funcionamiento de una memoria caché (2)

- **8 bloques, 1 palabra/bloque, correspondencia directa**

Peticiones

Dirección de palabra	Dirección binaria	Bloque de caché	Acierto o fallo
22	10 110	110	Fallo
26	11 010	010	Fallo
22	10 110	110	Acierto
26	11 010	010	Acierto
16	10 000	000	Fallo
3	00 011	011	Fallo
16	10 000	000	Acierto

Memoria caché

Índice	V	Etiqueta	Datos
000	1	10	Mem[10000]
001	0		
010	1	11	Mem[11010]
011	1	00	Mem[00011]
100	0		
101	0		
110	1	10	Mem[10110]
111	0		

Ejemplo de funcionamiento de una memoria caché (3)

- **8 bloques, 1 palabra/bloque, correspondencia directa**

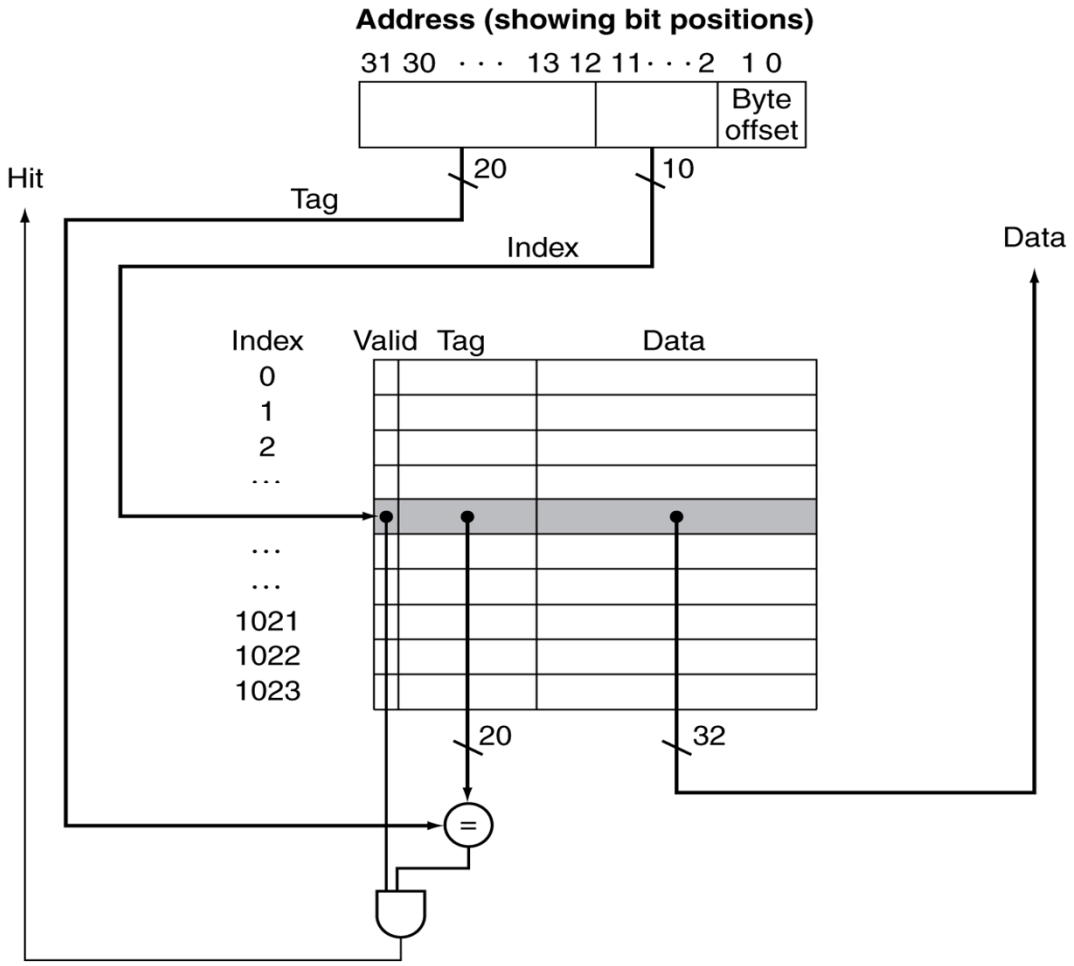
Peticiones

Dirección de palabra	Dirección binaria	Bloque de caché	Acierto o fallo
22	10 110	110	Fallo
26	11 010	010	Fallo
22	10 110	110	Acierto
26	11 010	010	Acierto
16	10 000	000	Fallo
3	00 011	011	Fallo
16	10 000	000	Acierto
18	10 010	010	Fallo

Memoria caché

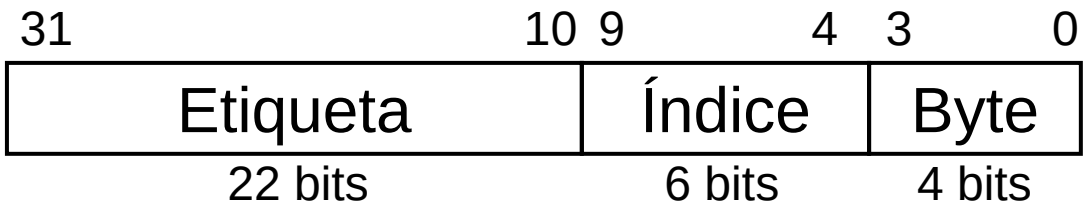
Índice	V	Etiqueta	Datos
000	1	10	Mem[10000]
001	0		
010	1	10	Mem[10010]
011	1	00	Mem[00011]
100	0		
101	0		
110	1	10	Mem[10110]
111	0		

Subdivisión de las direcciones



Ejemplo: bloques de mayor tamaño

- **64 bloques, 16 bytes/bloque**
  - ¿A qué bloque corresponde la dirección 1200?
- **Dirección de bloque = $\lfloor 1200/16 \rfloor$ = 75**
- **Número de bloque (caché)= 75 módulo 64 = 11**



Consideraciones sobre el tamaño de los bloques

- **Tener bloques más grandes reduciría la tasa de fallos**
  - Debido a la localidad espacial
- **Pero en una caché de tamaño fijo …**
  - Bloques más grandes ⇒ menos bloques
    - Más competencia ⇒ aumenta la tasa de fallos
- **Tener bloques más grandes aumenta la penalización por fallo**
  - Y eso puede anular el beneficio de haber reducido la tasa de fallos, e incluso superarlo
  - Pueden ayudar: comienzo inmediato y traer primero la palabra crítica

Los fallos de caché

- **En un acierto de caché, la CPU procede sin más**
- **Pero en un fallo de caché:**
  - Se detiene el cauce de datos de la CPU
  - Se trae un bloque desde el siguiente nivel de la jerarquía
  - En un fallo de caché para una instrucción ...
    - Reemprender la búsqueda de instrucciones
  - En un fallo de caché para traer un dato ...
    - Terminar el acceso al dato ...

Políticas de actualización: Escritura directa

- **Cuando se produce un acierto en escritura, se actualiza inmediatamente el bloque de la cache**
  - Pero entonces podría haber una inconsistencia entre la caché y la memoria
- **Escritura directa o "a través": también se actualiza la memoria principal**
- **Pero eso hace que las escrituras tarden más**
  - Por ejemplo, si CPI base = 1 y 10% de las instrucciones son de escritura en memoria, y una escritura en memoria tarda 100 ciclos,
    - CPI efectivo = 1 + 0.1×100 = 11
- **Solución: un *buffer* de escrituras**
  - Mantiene los datos que están esperando para ser escritos en memoria principal
  - La CPU prosigue inmediatamente
    - Sólo se detendrá ante una escritura si el *buffer* de escrituras está lleno

Políticas de actualización: Post-escritura o escritura retardada

- **Alternativa: en un acierto de escritura, sólo se actualiza el bloque de la caché**
  - Hay que seguir la pista de si cada bloque ha sido escrito ("bloque sucio")
- **Cuando un bloque sucio es reemplazado …**
  - Se copia el bloque en memoria
  - Se puede utilizar un *buffer* de escritura para hacer posible que el bloque reemplazante sea leído en primer lugar

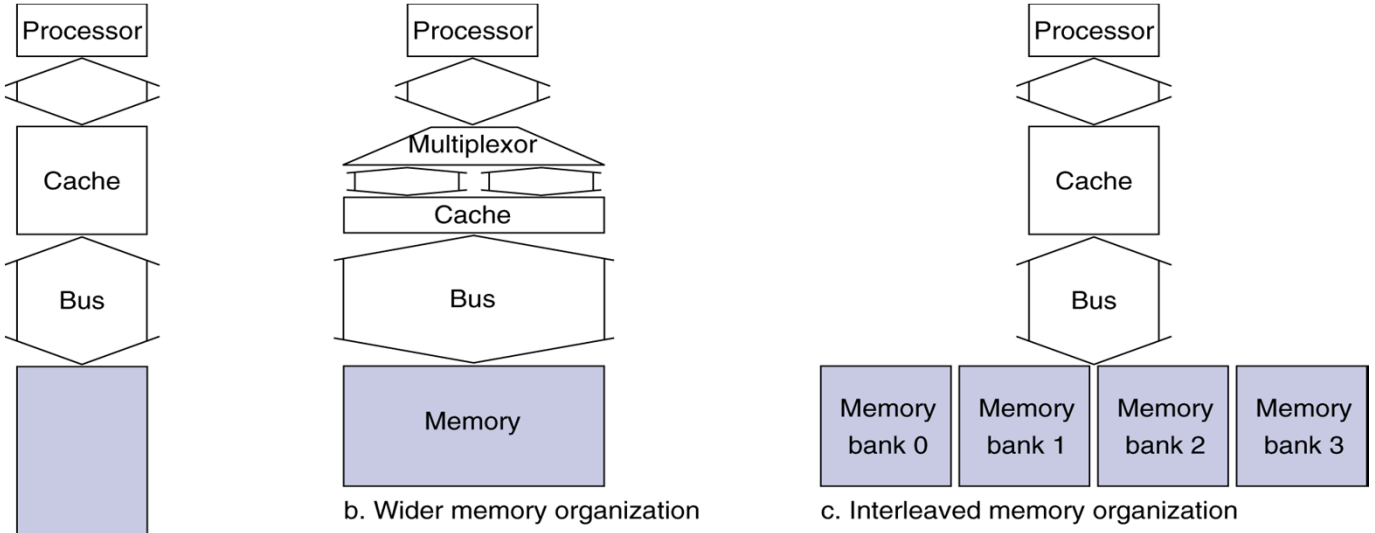
Reserva en escritura

- **¿Qué ocurre cuando se produce un fallo en escritura?**
- **Alternativas con escritura directa:**
  - Reserva en escritura cuando se falla: se trae el bloque a caché
  - Sin reserva en escritura: no se trae el bloque
    - Porque los programas generalmente escriben sobre cada bloque completo antes de leer de él (por ejemplo, en la inicialización)
- **Y con escritura retardada:**
  - Habitualmente se trae el bloque a la caché, es decir, se utiliza "Reserva en escritura"

La memoria principal que soporta a las cachés

- **Se utiliza DRAMs para la memoria principal**
  - Anchura fija (por ejemplo, 1 palabra)
  - Conectada a través de un bus sincronizado por un reloj y de anchura fija
    - El reloj del bus suele ser de frecuencia más baja que el de la CPU
- **Ejemplo de lectura de un bloque de caché**
  - 1 ciclo de bus para transferir la dirección
  - 15 ciclos de bus por acceso a DRAM
  - 1 ciclo de bus por cada transferencia de datos
- **Para una DRAM de 1 palabra de ancho y bloques de 4 palabras**
  - Penalización por fallo = 1 + 4×15 + 4×1 = 65 ciclos de bus
  - Ancho de banda = 16 bytes / 65 ciclos = 0.25 bytes/ciclo

Aumentar el ancho de banda de memoria



a. One-word-wide memory organization

b. Wider memory organization

c. Interleaved memory organization

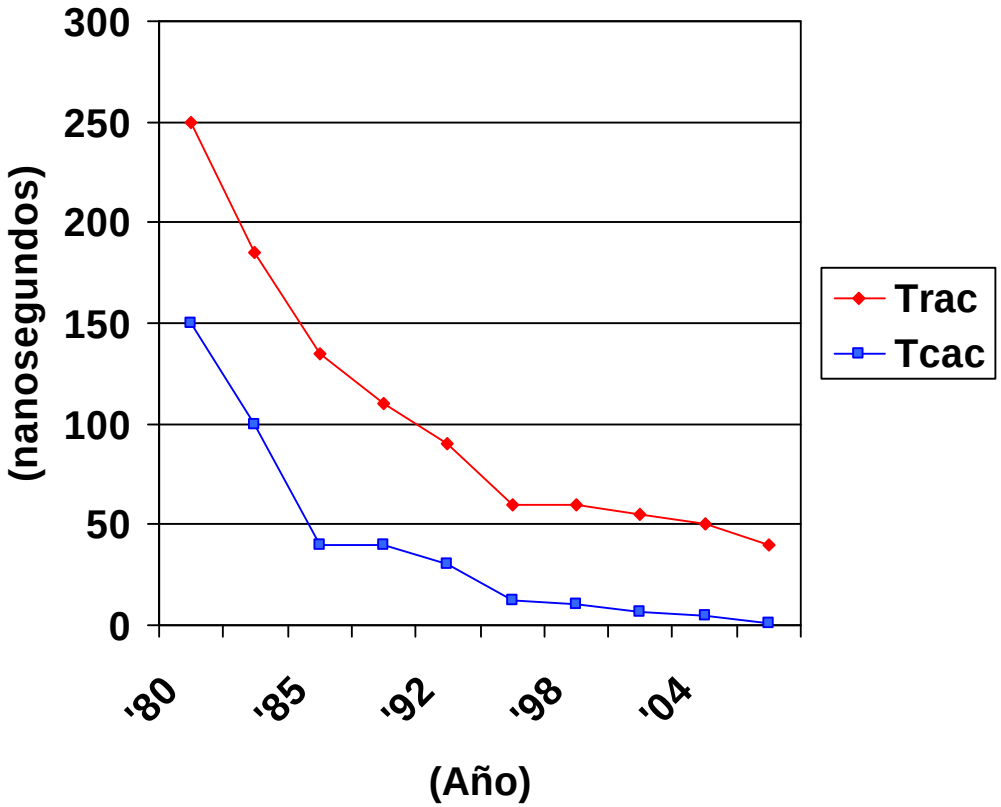
- Memoria de 2 palabras de ancho
  - Penalización por fallo = 1 + 2x15 + 2x1 = 33 ciclos de bus
  - Ancho de banda = 16 bytes / 33 ciclos = 0.48 bytes/ciclo
- Memoria entrelazada de 4 bancos
  - Penalización por fallo = 1 + 15 + 4×1 = 20 ciclos de bus
  - Ancho de banda = 16 bytes / 20 ciclos = 0.8 bytes/ciclo

Organización de la DRAM

- **Los bits están organizados dentro de la DRAM como un array rectangular**
  - Cuando se accede a DRAM se hace a una fila completa
  - Modo ráfaga: se proporcionan palabras sucesivas de una fila con menor latencia
- **"Double data rate" (DDR) DRAM**
  - Transfiere datos en la subida y bajada de la señal de reloj
- **"Quad data rate" (QDR) DRAM**
  - Varias entradas y salidas DDR diferentes

Generaciones de DRAM

Año	Capacidad	$/GB
1980	64Kbit	$1500000
1983	256Kbit	$500000
1985	1Mbit	$200000
1989	4Mbit	$50000
1992	16Mbit	$15000
1996	64Mbit	$10000
1998	128Mbit	$4000
2000	256Mbit	$1000
2004	512Mbit	$250
2007	1Gbit	$50



Medida del rendimiento de las cachés

- **Componentes del tiempo de CPU**
  - Los ciclos de ejecución del programa
    - Incluye el tiempo invertido en aciertos de caché
  - Los ciclos de detención a causa de la memoria
    - Principalmente debidos a los fallos de caché
- **Haciendo algunas suposiciones simplificadoras resulta:**

$$\text{Ciclos de detención por memoria} = \frac{\text{Accesos a memoria}}{\text{Programa}} \times \text{Tasa de fallos} \times \text{Penalización por fallo}$$

$$= \frac{\text{Instrucciones}}{\text{Programa}} \times \frac{\text{Fallos}}{\text{Instrucción}} \times \text{Penalización por fallo}$$

Ejemplo de rendimiento de una caché

- **Datos**
  - Tasa de fallos de la caché de instrucciones = 2%
  - Tasa de fallos de la caché de datos = 4%
  - Penalización por fallo = 100 ciclos
  - CPI base (con caché ideal) = 2
  - Load & stores son el 36% de las instrucciones
- **Ciclos de fallo por instrucción**
  - Debido a caché de instrucciones: $0.02 \times 100 = 2$
  - Debido a caché de datos: $0.36 \times 0.04 \times 100 = 1.44$
- **CPI Real = 2 + 2 + 1.44 = 5.44**
  - La CPU ideal es 5.44/2 =2.72 veces más rápida

Tiempo medio de acceso

- **También el tiempo de acierto es importante en relación al rendimiento**
- **Tiempo medio de acceso a memoria (*Average memory access time*, AMAT)**
  - AMAT = Tiempo de acierto + Tasa de fallo × Penalización por fallo
- **Ejemplo**
  - CPU con reloj de 1 ns, tiempo de acierto = 1 ciclo, penalización por fallo = 20 ciclos, tasa de fallos en la caché de instrucciones = 5%
  - AMAT = 1 + 0.05 × 20 ciclos = 2 ciclos = 2 ns. (ya que periodo=1 ns.)
    - 2 ciclos por acceso a memoria en promedio

Resumen sobre el rendimiento

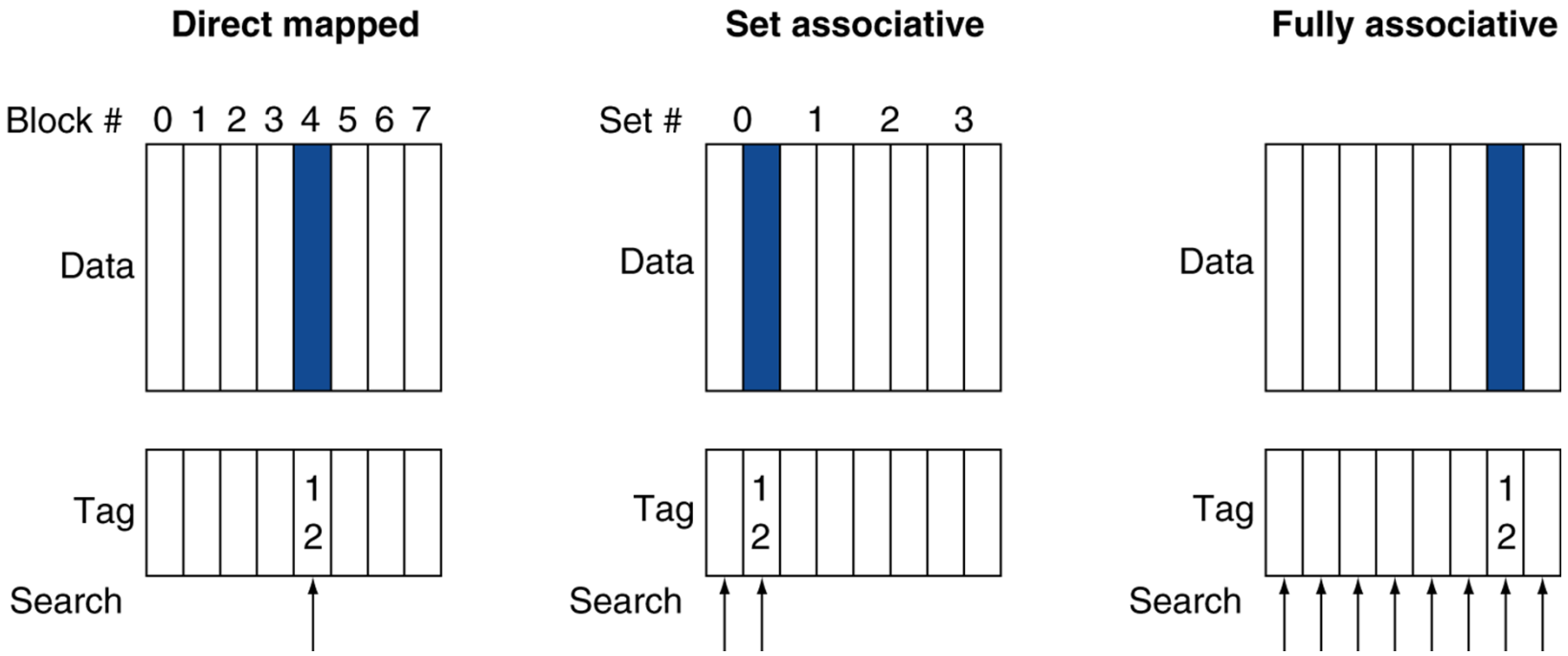
- **Cuando aumenta el rendimiento (velocidad) de la CPU**
  - La penalización por fallo se hace más significativa
- **Cuando disminuye el CPI base**
  - Una mayor proporción de tiempo se gasta en las detenciones debidas a la memoria
- **Cuando aumenta la frecuencia de reloj**
  - Las detenciones por memoria suponen más ciclos
- **No se puede despreciar el comportamiento de la caché cuando se evalúa el rendimiento de un sistema**

Cachés asociativas

- **Totalmente asociativas**
  - Permiten que un bloque se aloje en cualquier entrada de caché
  - Requiere buscar a la vez en todas las entradas (bloques)
  - Hace falta un comparador por cada entrada (costoso)
- **Asociativa por conjuntos de *n*-vías**
  - Cada conjunto contiene *n* entradas
  - El número de bloque determina a qué conjunto va:

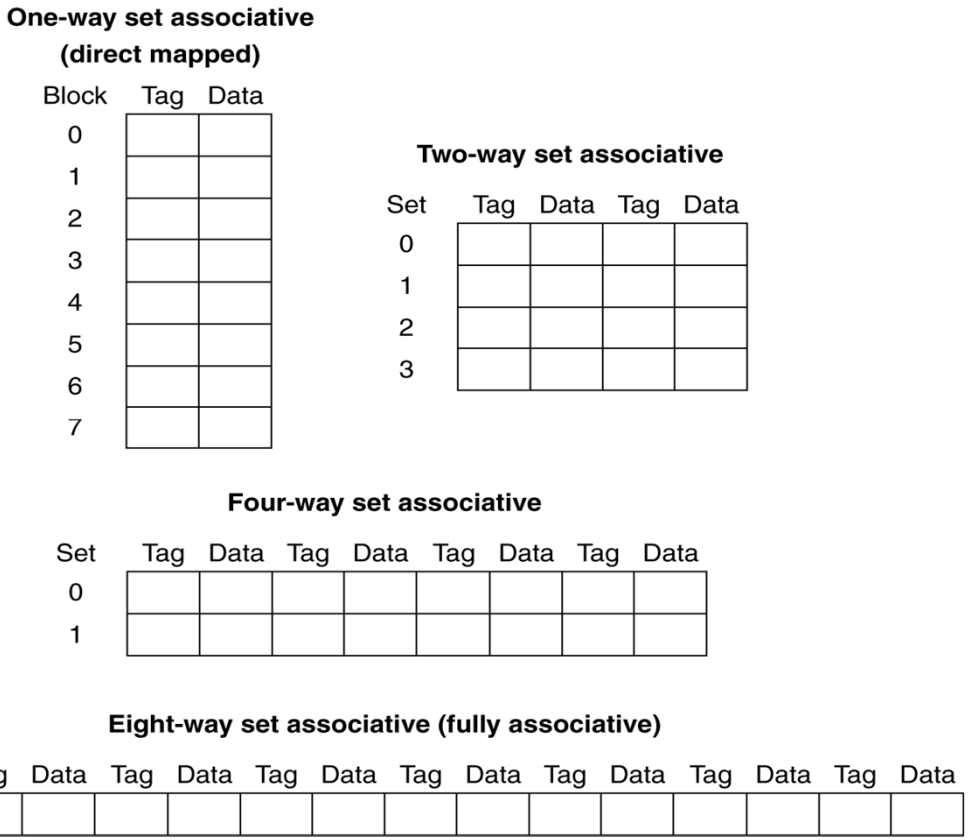
    (Número de bloque) módulo (Número de conjuntos en la caché)
  - Buscar a la vez en todas las entradas de un conjunto dado
  - *n* comparadores (menos costoso)

Ejemplo de caché asociativa



Posibilidades de asociatividad

- **Para una caché con 8 entradas**



Ejemplo de asociatividad (1)

- **Comparación de cachés de 4 bloques**
  - Correspondencia directa, asociativa por conjuntos de 2 vías, totalmente asociativa
  - Secuencia de accesos a los bloques: 0, 8, 0, 6, 8
- **Caché de correspondencia directa**

Dirección de bloque	Índice de 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 de asociatividad (2)

- **2-Asociativa**

Dirección de bloque	Índice de 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	0	fallo	Mem[0]	**Mem[6]**		
8	0	fallo	**Mem[8]**	Mem[6]		

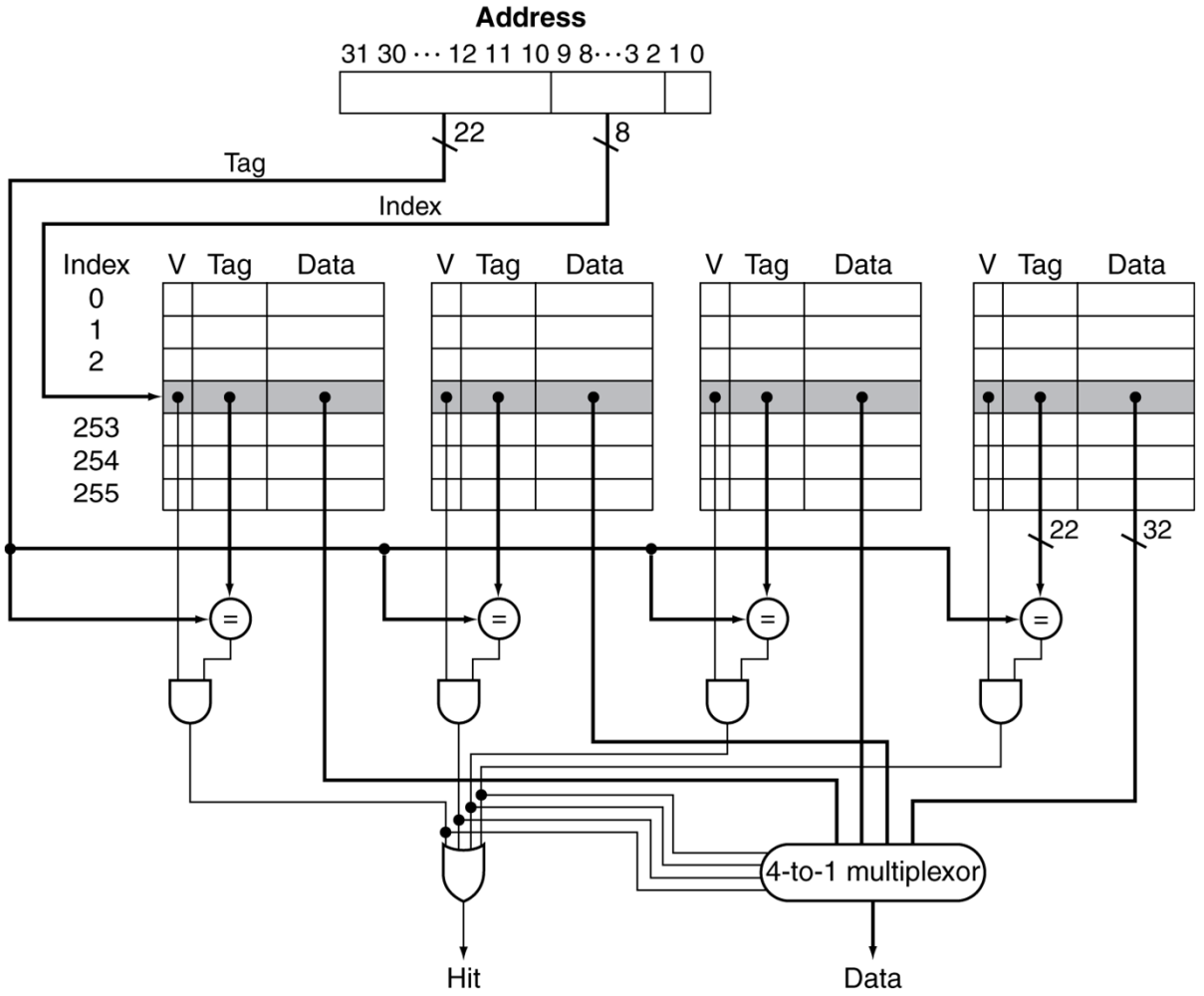
- **Totalmente asociativa**

Dirección de bloque		Acierto /fallo	Contenido de la caché después del acceso			
0		fallo	**Mem[0]**			
8		fallo	Mem[0]	**Mem[8]**		
0		acierto	**Mem[0]**	Mem[8]		
6		fallo	Mem[0]	Mem[8]	**Mem[6]**	
8		acierto	Mem[0]	**Mem[8]**	Mem[6]	

¿Y cuánta asociatividad?

- **Aumentar la asociatividad disminuye la tasa de fallos**
  - Pero el beneficio es cada vez menor
- **Simulación de un sistema con una caché de datos de 64 KB y bloques de 16 palabras con SPEC2000 (tasa de fallos)**
  - 1 vía: 10.3%
  - 2 vías: 8.6%
  - 4 vías: 8.3%
  - 8 vías: 8.1%

Organización de caché asociativa por conjuntos



Política de reemplazo

- **Correspondencia directa: no hay que escoger**
- **Asociativa por conjuntos:**
  - Se prefiere una entrada no válida, si la hay
  - De lo contrario, escoger entre las entradas del conjunto
- **Menos recientemente utilizado**

  (*least recently used*, LRU)
  - Escoge para expulsar el bloque que lleva más tiempo sin usarse
    - Sencillo para 2 vías, manejable para 4 vías, demasiado pesado para asociatividad mayor
- **Aleatorio**
  - Da aproximadamente el mismo rendimiento que LRU para asociatividades elevadas

Las fuentes de los fallos

- **Fallos inevitables, forzosos o de arranque en frío**
  - El primer acceso a un bloque
- **Fallos de capacidad**
  - Debidos al tamaño limitado de la caché
  - Se accede nuevamente a un bloque ya reemplazado
- **Fallos de conflicto (o fallos de colisión)**
  - Ocurren en una caché no totalmente asociativa
  - Son debidos a la competencia por entradas dentro de un conjunto
  - No ocurrirían en una caché totalmente asociativa del mismo tamaño total

Cachés multinivel

- **La caché primaria está enganchada a la CPU**
  - Pequeña, pero rápida
- **La caché de nivel 2 (L2) da servicio a los fallos de la caché primaria (o de nivel 1, L1)**
  - Mayor, más lenta, pero sigue siendo más rápida que la memoria principal
- **La memoria principal da servicio a los fallos de la caché L2**
- **Algunos sistemas superiores incluyen una caché de tercer nivel (L3)**

Ejemplo de caché multinivel

- **Datos**
  - CPI base de la CPU = 1, frecuencia de reloj = 4GHz
  - Tasa de fallos / instrucción = 2%
  - Tiempo de acceso a memoria principal = 100ns
- **Sólo con caché L1**
  - Penalización por fallo = 100ns/0.25ns = 400 ciclos
  - CPI efectivo= 1 + 0.02 × 400 = 9

Ejemplo (continuación)

- **Ahora añadimos una caché L2**
  - Tiempo de acceso = 5 ns
  - Tasa de fallos global a memoria principal = 0.5%
- **Fallo en L1 con acierto en L2**
  - Penalización = 5 ns/0.25 ns = 20 ciclos
- **Fallo en L1 con fallo en L2**
  - Penalización extra = 100 ns/0.25 ns = 400 ciclos
- **CPI = 1 + 0.02 × 20 + 0.005 × 400 = 3.4**
- **Razón de rendimientos = 9/3.4 = 2.6**

Consideraciones sobre las cachés multinivel

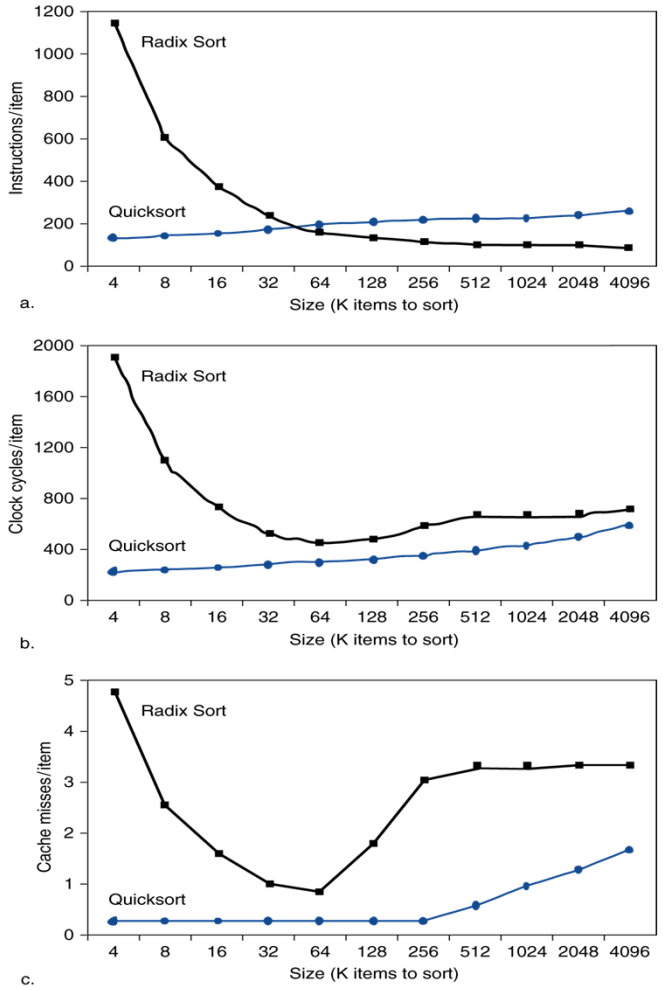
- **Caché primaria o L1**
  - Diseñada tratando de minimizar el tiempo de acierto
- **Caché L2**
  - Su objetivo es conseguir baja tasa de fallos para evitar accesos a memoria principal
  - Su tiempo de acierto tiene poca repercusión en el conjunto
- **Resultados**
  - Habitualmente el tamaño de la caché L1 es menor que una sola caché
  - El tamaño de bloque de L1 es menor que el de L2

Interacciones con diseños avanzados de CPU

- **Las CPU "en desorden" pueden ejecutar instrucciones durante un fallo de caché**
  - Los *store* pendientes permanecen en una unidad de *load/store*
  - Las instrucciones dependientes esperan en *estaciones de reserva*
    - Las instrucciones independientes continúan su ejecución
- **El efecto de los fallos depende de cómo esté organizado el flujo de datos del programa**
  - Es difícil analizarlo
  - Para evaluarlo hay que utilizar simulación a nivel de sistema

Las interacciones con el software

- **El número de fallos depende de los patrones de acceso a memoria**
  - Comportamiento del algoritmo
  - Optimizaciones del compilador relativas a los accesos a memoria

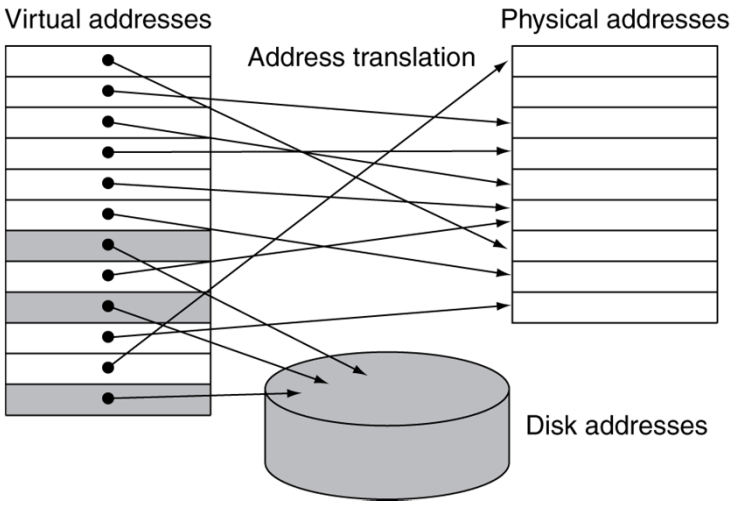


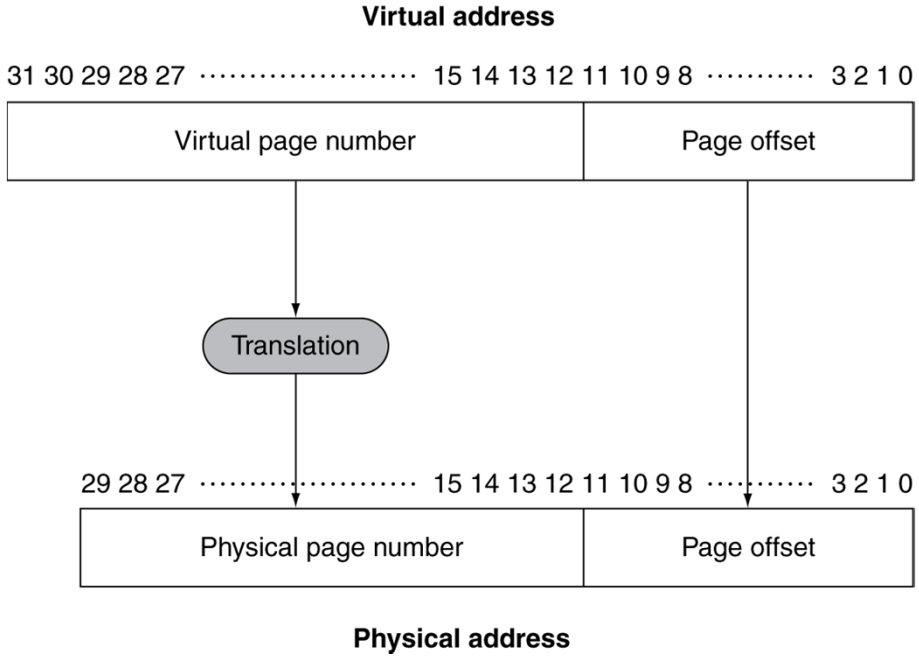
Memoria virtual

- **Utiliza la memoria principal como una "caché" para el nivel de almacenamiento secundario (los discos)**
  - Es gestionada conjuntamente por el hardware de la CPU y el software del sistema operativo (SO)
- **Los programas comparten la memoria principal**
  - Cada uno maneja su espacio privado de direcciones virtuales alojando en él sus datos y código de uso frecuente
  - Ese espacio está protegido respecto del acceso de otros programas
- **La CPU y el SO traducen las direcciones virtuales en direcciones físicas**
  - El "bloque" de memoria virtual (MV) se denomina "página"
  - Un fallo en traducción de MV se denomina "fallo de página"

Traducción de direcciones

- **Páginas de tamaño fijo (por ejemplo, 4 KB)**





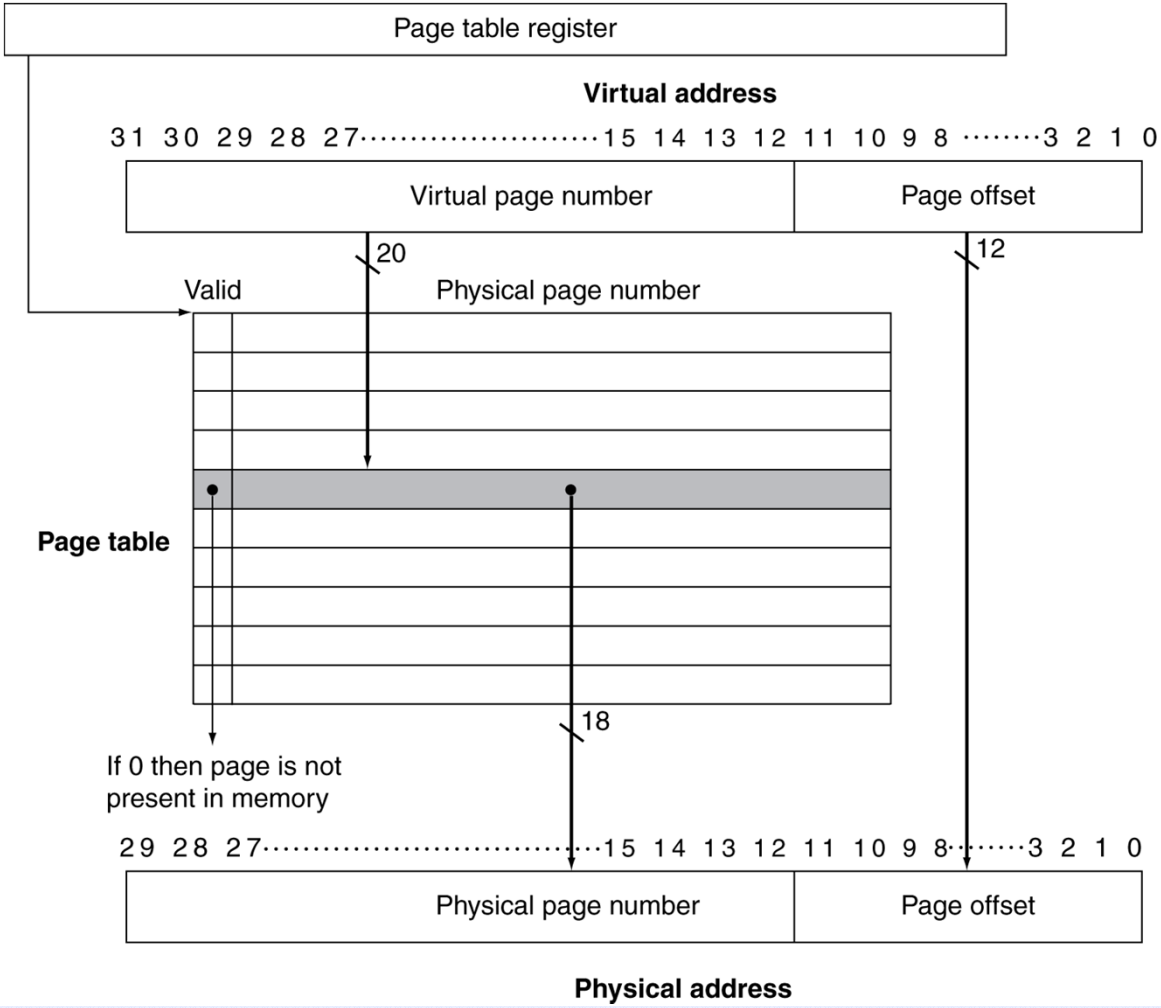
Penalización por fallo de página

- **Cuando se produce un fallo de página, la página debe ser traída desde el disco**
  - Tarda millones de ciclos de reloj
  - Es gestionado por el código del SO
- **Se intenta minimizar la tasa de fallos de página**
  - Ubicación totalmente asociativa
  - Algoritmos de reemplazo bien diseñados

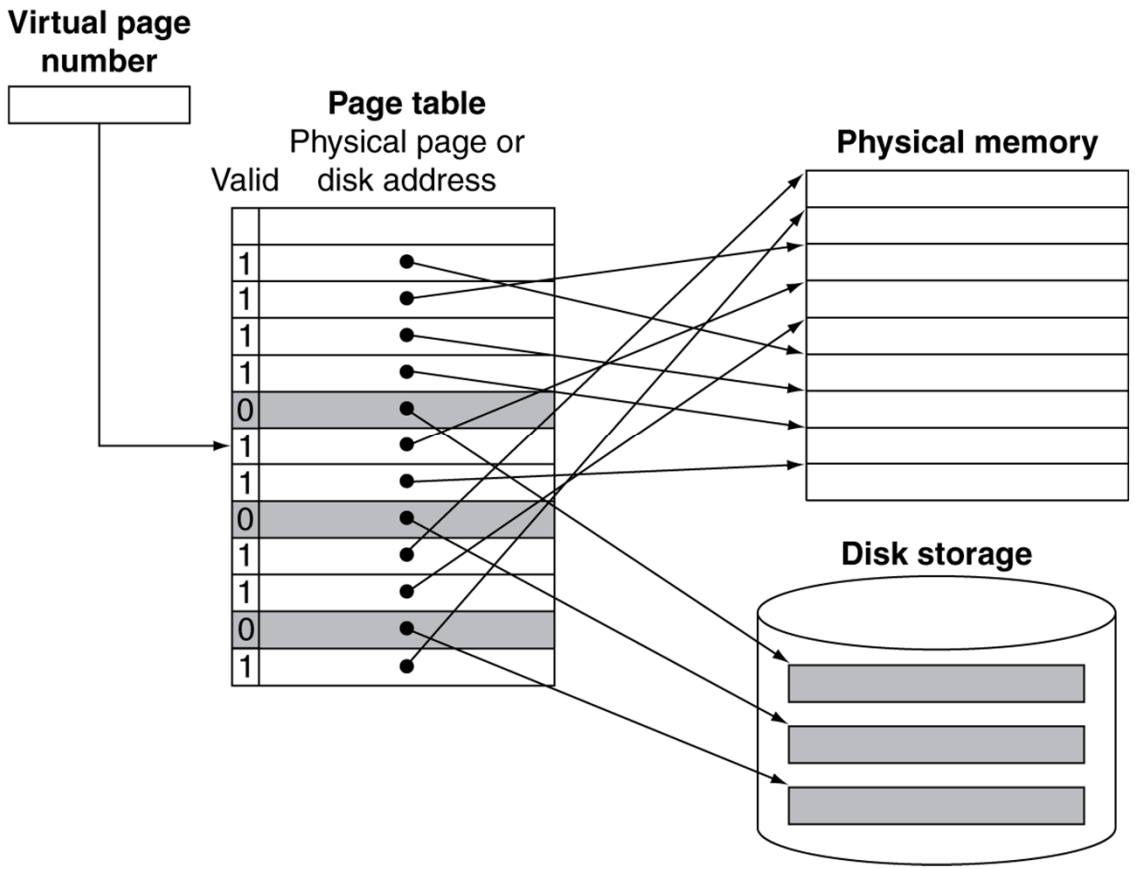
Tablas de páginas

- **Almacenan información sobre la ubicación**
  - Conjunto de entradas de tabla de páginas, indexadas por el número de página virtual
  - El registro de tabla de páginas de la CPU apunta hacia la tabla de páginas que está en memoria física
- **Si la página está presente en memoria**
  - Su PTE (*page table entry*, entrada de la tabla de páginas) almacena el número de página física
  - Y además otros bits de estado (referenciado, sucio, válido,...)
- **Si la página no está presente**
  - PTE puede contener una referencia para encontrar la página en el área de intercambio de páginas del disco (espacio de *swap*)

La traducción utilizando una tabla de páginas



Relación entre páginas y espacio de almacenamiento



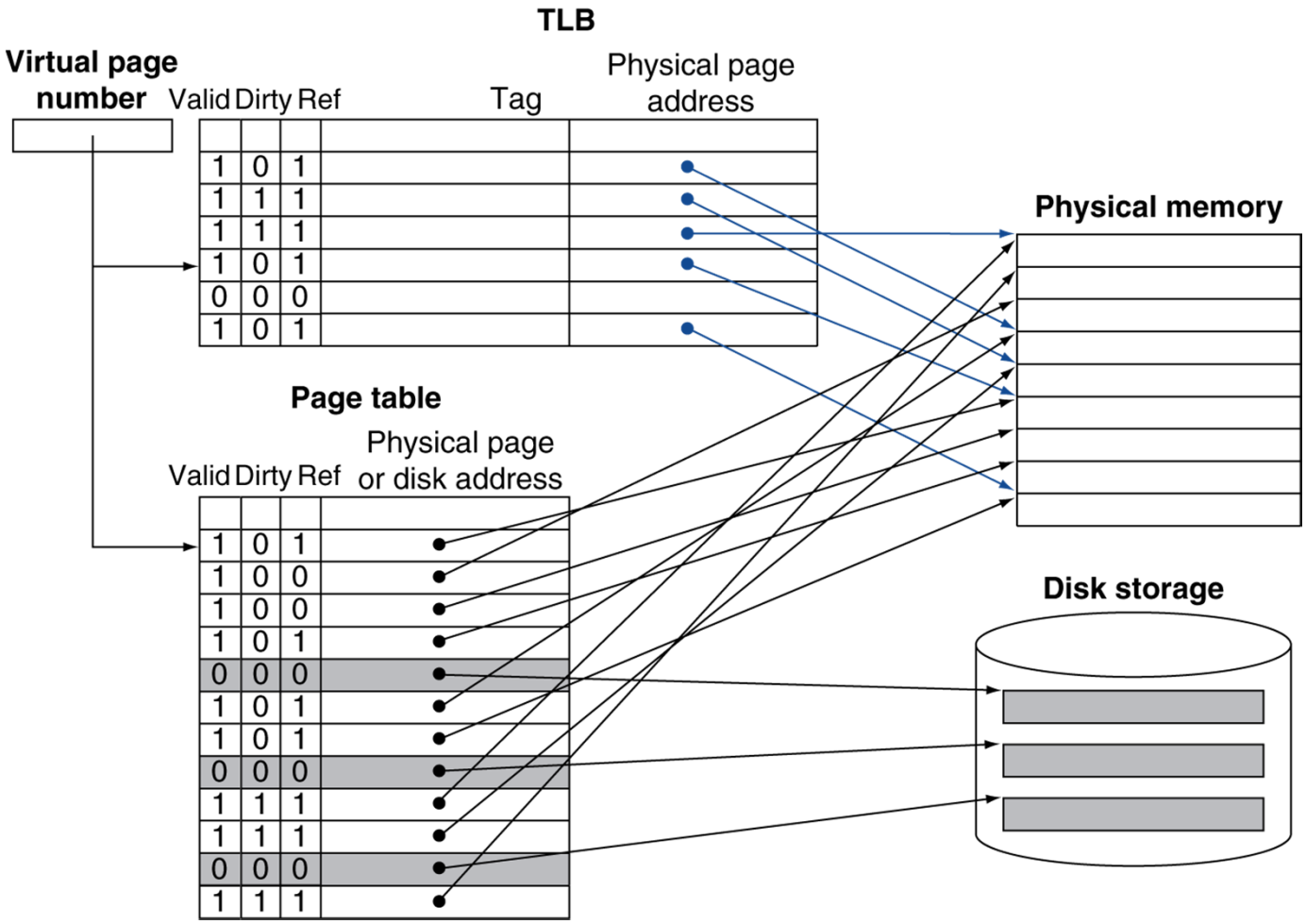
El reemplazo y las escrituras

- **Para reducir la tasa de fallos de páginas, se prefiere un reemplazo LRU (reemplazar la página "menos recientemente utilizada")**
  - Un bit de referencia (o bit de uso) de la PTE se pone a 1 cuando se accede a esa página
  - Periódicamente el SO pone a 0 esos bits
  - Una página con el bit de referencia en cero no ha sido utilizada recientemente
- **La escritura en disco tarda millones de ciclos**
  - Se escribe un bloque cada vez, no posiciones individuales
  - La escritura directa no se puede hacer porque sería muy lenta
  - Siempre hay que utilizar escritura retardada
  - Hay un bit de "sucio" en la PTE que se pone a 1 cuando se modifica la página, por lo que cuando se reemplace hay que escribirla

Traducción rápida utilizando un TLB (*translation-lookaside buffer*)

- **La traducción de direcciones requeriría referencias a memoria adicionales**
  - Una para acceder a la PTE
  - Y después el acceso a memoria en sí
- **Pero el acceso a la tabla de páginas tiene buena localidad**
  - Por ello se utiliza una caché rápida de PTEs dentro de la CPU, llamada ***Translation Look-aside Buffer* (TLB)**
  - Típicamente: 16–512 PTEs, 0.5–1 ciclos por acierto, 10–100 ciclos por fallo, tasa de error de 0.01%–1%
  - Los fallos podrían ser gestionados por hardware o software

Traducción rápida utilizando un TLB (2)



Los fallos en TLB

- **Si la página está en memoria**
  - Se carga la PTE desde memoria y se reintenta
  - Podría ser gestionado por hardware
    - Puede resultar complejo para tablas de páginas con estructuras complicadas
  - O por software
    - Se dispara una excepción especial que se resuelve con un manejador optimizado
- **Si la página no está en memoria ("fallo de página")**
  - El SO gestiona cómo traer la página y actualizar la tabla de páginas
  - Y después se vuelve a lanzar la instrucción que dio lugar al fallo de página

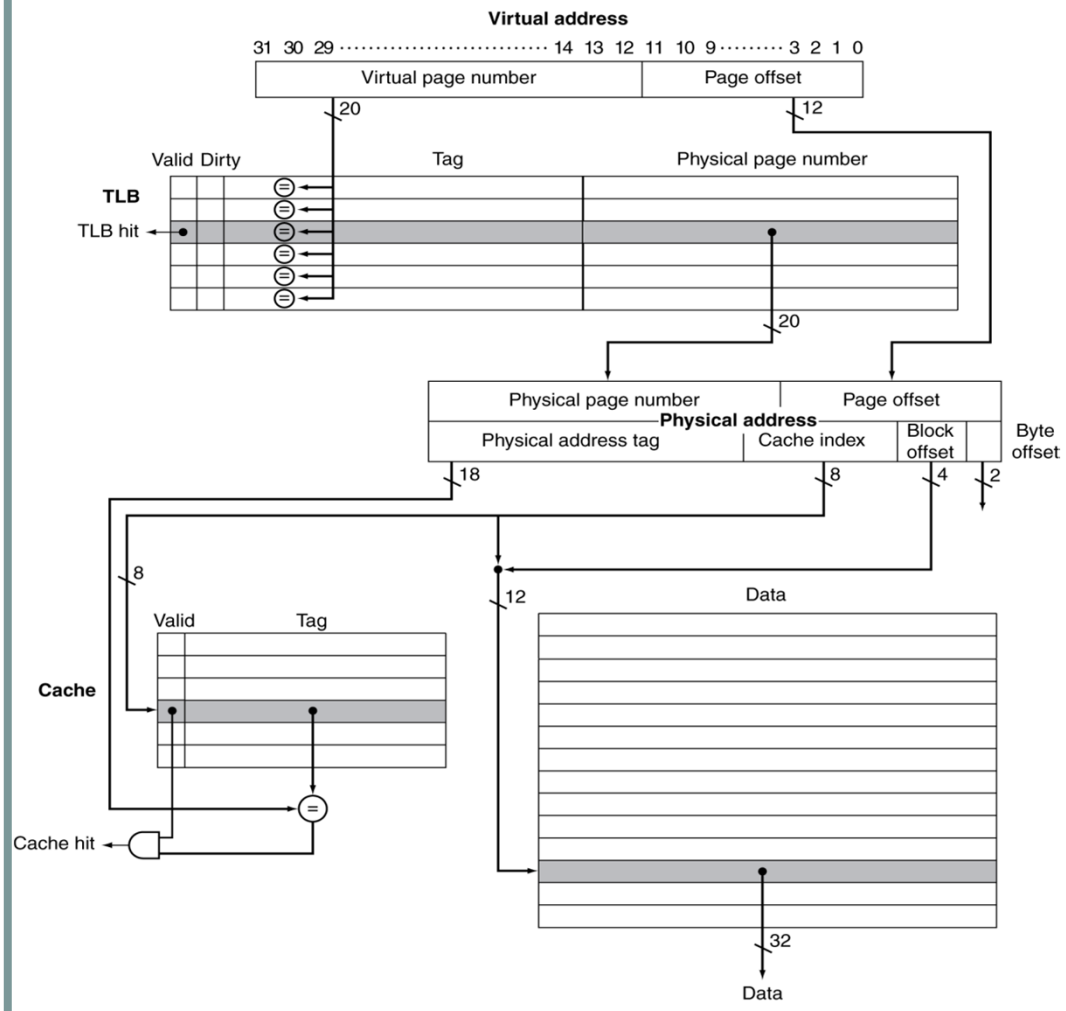
El manejador de fallo de TLB

- **Un fallo de TLB indica**
  - Que la página está presente, pero no su PTE en el TLB
  - O bien que la página no está presente
- **Es necesario reconocer el fallo en TLB antes de que pueda ser escrito el registro destino**
  - Eso implica disparar una excepción
- **El manejador copia una PTE desde la memoria hasta el TLB**
  - Después reinicia la ejecución de la instrucción
  - Si la página no está en memoria, se produce fallo de página

El manejador de fallo de página

- **Utiliza la dirección virtual que ha dado lugar al fallo para buscar la PTE**
- **Localiza la página sobre el disco**
- **Escoge una página para reemplazarla**
  - Si está sucia, primero la escribe en disco
- **Trae la página a memoria y actualiza la tabla de páginas**
- **Hace que el proceso pueda continuar ejecutándose**
  - Continúa desde la instrucción que produjo el fallo

Interacción entre caché y TLB



- **Si la caché utiliza etiquetas obtenidas de las direcciones físicas**
  - Se necesita traducir antes de acceder a la caché
- **Alternativa: utilizar etiquetas obtenidas de la dirección virtual**
  - Surgen complicaciones debidas al *aliasing*
    - Puede haber direcciones virtuales diferentes que son compartidas por la misma dirección física

Protección de memoria

- **Diferentes tareas pueden compartir partes de sus espacios de direcciones virtuales**
  - Pero necesitan ser protegidas frente a accesos erróneos
  - Eso requiere asistencia del SO
- **Hay hardware para dar soporte a la protección que gestiona el SO**
  - Modo con privilegios de supervisor (modo kernel)
  - Instrucciones "privilegiadas"
  - La tabla de páginas y otras informaciones de estado sólo son accesibles en modo supervisor
  - Excepción de llamada al sistema (por ejemplo, `syscall` en MIPS)

La jerarquía de memoria (en resumen)

Recuadro importante

- **Se aplican los mismos principios en todos los niveles de la jerarquía de memoria**
  - Basados en las nociones de gestión de caché
- **En cada nivel de la jerarquía**
  - Se ubican bloques
  - Se busca un bloque
  - Se reemplaza cuando hay un fallo
  - Hay una política para realizar las escrituras

El alojamiento de los bloques (continúa el resumen) (2)

- **Está determinado por la asociatividad**
  - Correspondencia directa (asociativo de una vía)
    - Una sola posibilidad de alojamiento
  - Asociativo por conjuntos de *n*-vías
    - *n* posibilidades dentro de un conjunto
  - Totalmente asociativo
    - Puede alojarse en cualquier lugar
- **Una asociatividad mayor reduce la tasa de fallos**
  - Aumenta la complejidad, el coste y el tiempo de acceso

La localización de un bloque (continúa el resumen) (3)

Asociatividad	Método de localización	Comparaciones de etiqueta
Correspondencia directa	Índice	1
Asociativa por conjuntos de n vías	Índice de conjunto, después búsqueda dentro de las entradas del conjunto	n
Totalmente asociativa	Búsqueda en todas las entradas	Número de entradas
	Tabla de búsqueda completa	0

- **Cachés en hardware**
  - Reducir el número de comparaciones para reducir coste
- **Memoria virtual**
  - La búsquda en tablas completas hace realizable la asociatividad total
  - El beneficio es una menor tasa de fallos

El reemplazo (continúa el resumen) (4)

- **Elegir una entrada a reemplazar cuando se produce un fallo**
  - Menos recientemente usado (LRU)
    - Hardware complejo y costoso para asociatividad elevada
  - Aleatorio
    - Se aproxima a LRU en prestaciones, más fácil de implementar
- **Memoria virtual**
  - Se hace una aproximación a LRU con soporte del hardware

La política de escrituras (continúa el resumen) (5)

- **Escritura directa**
  - Actualiza tanto el nivel superior como el inferior
  - Simplifica el reemplazo, pero requiere un buffer de escrituras
- **Escritura retardada**
  - Actualiza sólo el nivel superior
  - Actualiza el nivel inferior cuando el bloque es reemplazado
  - Necesita mantener más información de estado
- **Memoria virtual**
  - Sólo es viable la escritura retardada debido a la latencia de escritura del disco

Los compromisos de diseño de las cachés (continúa el resumen) (6)

Cambio en el diseño	**Efecto sobre la tasa de fallos**	**Efecto negativo sobre el rendimiento**
Aumentar el tamaño de la caché	Disminuyen los fallos de capacidad	Puede aumentar el tiempo de acceso
Aumentar la asociatividad	Disminuyen los fallos de conflicto	Puede aumentar el tiempo de acceso
Aumentar el tamaño de bloque	Disminuyen los fallos forzosos	Aumenta la penalización por fallo. Para tamaños de bloque muy grandes, puede aumentar la tasa de fallos (“polución”)