MIDIENDO Y MEJORANDO EL DESEMPEÑO CON CACHE

ARQUITECTURA DEL PROCESADOR II

En esta sección[1], se comienzan a examinar las formas de medir y mejorar el desempeño de la cache. Después se exploran dos técnicas diferentes para mejorar el desempeño de la cache. Una se concentra en reducir la tasa de fallos reduciendo la probabilidad de que dos bloques distintos de memoria compitan por la misma ubicación en la cache. La segunda técnica reduce la penalidad adicionando un nivel más en la jerarquía. Esta técnica llamada cache multinivel, apareció primero en ordenadores de alto desempeño, vendidos en más de U$A100.000 en 1990; desde entonces se ha hecho común en dispositivos móviles personales que se venden por unos pocos cientos de dólares!. El tiempo de CPU puede ser dividido en ciclos de reloj que la CPU gasta ejecutando el programa y los ciclos de reloj que la CPU gasta esperando por el sistema de memoria. Normalmente se asume que los costos por accesos exitosos a cache son parte del ciclo normal de ejecución de la CPU. Por eso:
$Tiempo\ de\ CPU =$

$$(Ciclos\ de\ ejecución\ +\ Ciclos\ por\ atasco\ de\ memoria) \times\ Tiempo\ de\ ciclo$$

Los ciclos de reloj por atasco de la memoria provienen principalmente de los fallos de cache, y es lo que se asume aquí. También se restringe la discusión a un modelo simplificado del sistema de memoria. En los procesadores reales, los atascos generados por las lecturas y por las escrituras pueden ser bastantes complejos, y predecir el desempeño exacto usualmente requiere una simulación muy detallada del sistema de memoria y procesador.

Los ciclos de reloj por atasco de la memoria pueden ser definidos como la suma de los ciclos de atasco que provienen de las lecturas más los ciclos de atascos que provienen de las escrituras:
$Ciclos\ por\ atasco\ de\ memoria =$

$$(Ciclos\ de\ atasco\ por\ lectura\ +\ Ciclos\ de\ atasco\ por\ escritura)$$

Los ciclos de atasco por lectura pueden ser definidos en términos del número de accesos de lectura por programa, la penalidad por miss en ciclos de reloj debido a lecturas y la tasa de fallo por lectura:
$Ciclos\ de\ atasco\ por\ lectura =$

$$\frac{Lecturas}{Programa} \times Tasa\ de\ fallos\ por\ lectura \times Penalidad\ por\ fallo\ de\ lectura$$

Las escrituras son más complicadas. Para un esquema de escritura directa existen dos fuentes de atascos: fallos de escritura, los cuales usualmente requieren que se recupere el bloque antes de continuar con la escritura y los atascos por buffer de escritura, los cuales

[1]Este material de estudio es la traducción de la sección 5.4 del libro de Patterson y Hennessy [1]

ocurren cuando el buffer de escritura está lleno al momento de hacer la escritura. De esa manera, el total de los ciclos de atasco por escritura es igual a la siguiente suma:
Ciclos por atasco de escritura =

$$(\frac{Escrituras}{Programa} \times Tasa\ de\ fallos\ por\ escritura \times Penalidad\ por\ fallo\ de\ escritura)$$

$$+\ Atascos\ por\ buffer\ de\ escritura$$

Como los atascos por buffer de escritura dependen de la cercanía (proximidad) de las escrituras y no solamente de la frecuencia, no es posible dar una ecuación simple para calcular esta clase de atascos. Afortunadamente, en los sistemas con buffers de escritura de una profundidad suficiente (cuatro o más palabras) y una memoria capaz de aceptar escrituras a una tasa que exceda significativamente la frecuencia promedio de escritura en programas (por un factor de dos) los atascos por buffers de escritura serán pocos y pueden ser ignorados sin problema. Si un sistema no cumple estos criterios, podría no estar bien diseñado; quiere decir que el diseñador debiera haber usado un buffer de escritura más profundo o una organización de escritura demorada.

El esquema de escritura demorada también tiene potenciales atascos adicionales que surgen de la necesidad de escribir un bloque de cache actualizado en memoria cuando el bloque es reemplazado. Esto se discutirá más adelante. En muchas organizaciones de cache de escritura directa, las penalidades por fallos de escritura y lectura son iguales (el tiempo para recuperar el bloque desde la memoria). Si se asume que los atascos por buffer de escritura son despreciados, es posible combinar las lecturas y las escrituras usando una única tasa de fallos y penalidad por miss:
Ciclos de reloj por atasco de memoria =

$$\frac{Accesos\ a\ memoria}{Programa} \times Tasa\ de\ fallos\ \times\ Penalidad\ por\ fallos$$

Se puede factorizar como:
Ciclos de reloj por atasco de memoria =

$$\frac{Instrucciones}{Programa} \times \frac{Fallos}{Instrucción} \times Penalidad\ por\ fallos$$

Conviene analizar un ejemplo simple para ayudar a entender el impacto del desempeño de la cache sobre el desempeño del procesador.

Ejemplo.Cálculo del desempeño de la cache. Asuma que la tasa de fallo de una cache de instrucciones es del 2% y la la tasa de fallo de una cache de datos es 4%. Si un procesador tiene un CPI de 2 sin atascos por memoria y la penalidad por fallo es de 100 ciclos para toda clase de miss, determine cuanto más rápido podría correr un procesador con una cache perfecta que nunca tiene miss. Asuma que la frecuencia de loads y stores es del 36%.
Respuesta. El número de ciclos debido a miss por instrucciones, en términos de cantidad de instrucciones (I) es:

$$Ciclos\ por\ miss\ de\ instrucciones = I \times 2\% \times 100\ =\ 2 \times I$$

como la frecuencia de todos los loads y stores es del 36%, es posible encontrar el número de ciclos debidos a miss de memoria para referencias a datos:

$$Ciclos\ por\ miss\ de\ datos = I \times 36\% \times 4\% \times 100 = 1,44 \times I$$

El número total de ciclos debidos a atascos por memoria es: $2,00\ I\ +\ 1,44\ I\ =\ 3,44\ I$. Es decir más de tres ciclos de atasco por instrucción. De acuerdo a eso, el CPI total incluyendo los atascos por memoria es $2 + 3,44 = 5,44$. Debido a que no hay cambios en la cantidad de instrucciones ni en la frecuencia del reloj, la relación entre los dos tiempos de CPU es:

$$\frac{Tiempo\ de\ CPU\ con\ atascos}{Tiempo\ de\ CPU\ sin\ atascos} = \frac{I \times CPI_{atascos} \times Ciclo\ de\ reloj}{I \times CPI_{perfecta} \times Ciclo\ de\ reloj} =$$

$$\frac{CPI_{atascos}}{CPI_{perfecta}} = \frac{5,44}{2} = 2,72$$

□

¿Qué sucedería si se acelera el procesador pero no el sistema de memoria?. La cantidad de tiempo gastado por los atascos del sistema de memoria se convertirá en una fracción mayor del tiempo de ejección. La ley de Amdahl, que ya hemos analizado, nos recuerda este hecho. Un ejemplo simple nos muestra lo serio que puede ser este problema. Supongamos que se logra acelerar la computadora del ejemplo anterior de manera que se reduce el CPI de 2 a 1 sin cambiar la frecuencia del reloj, eso podría ser alcanzado con un pipeline mejorado. El sistema con fallos de cache podra tener un CPI=1+3,44=4,44 y el sistema con una cache perfecta podría ser

$$\frac{4,44}{1} = 4,44\ veces\ más\ rápido$$

La cantidad de tiempo de ejecución consumida por atascos podría aumentar de

$$\frac{3,44}{5,44} = 63\%$$

a

$$\frac{3,44}{4,44} = 77\%$$

De la misma manera, incrementar la frecuencia de reloj sin cambiar el sistema de memoria también incrementa la pérdida de desempeño debido a las fallas de cache. Las ecuaciones y ejemplos previos asumen que el tiempo de hit no es un factor en la determinación del desempeño de la cache. Claramante, si el tiempo de hit se incrementa, el tiempo total para acceder a una palabra desde el sistema de memoria, se incrementa, causando posiblemente un incremento en el tiempo de ciclo del procesasdor. Si bien en breve veremos ejemplos adicionales de que es lo que puede incrementar el tiempo de hit, un ejemplo es incrementar el tamaño de la cache. Una cache más grande claramente puede tener un tiempo de acceso mayor, algo similar a lo que sería en el ejemplo de la biblioteca tener un escritorio muy grande (de unos 3 metros cuadrados), podría tomar mucho tiempo ubicar un libro en este escritorio. Un incremento en el tiempo de hit probablemente agregue una etapa más a la

segmentación, debido a que un hit de cache puede tomar multiples ciclos del procesador. Aunque es más complicado calcular el impacto en el desempeño de un procesador con más etapas, en parte el aumento en el tiempo de hit para una cache más grande puede superar a la mejora por la tasa de hit, llevando a un decremento en el desempeño del procesador.

Para capturar el hecho de que el tiempo de acceso a los datos tanto en hit como miss afecta el desempeño, los diseñadores a veces usan el *tiempo promedio de acceso a la memoria* AMAT por sus siglas en inglés, como una manera de examinar alternativas en el diseño de caches. El tiempo promedio de acceso a la memoria es el tiempo promedio para acceder la memoria, considerando tanto hits como misses y la frecuencia de diferentes accesos, es igual a lo siguiente:

$$AMAT = Tiempo\ de\ hit + Tasa\ de\ fallo \times Penalidad\ por\ fallo$$

Ejemplo. Calculando el tiempo promedio de acceso a la memoria. Encontrar el AMAT para un procesador con un tiempo de ciclo de reloj de 1ns, una penalidad por miss de 20 ciclos, una tasa de fallo de 0,05 miss por instrucción y un tiempo de acceso a cache (incluyendo el tiempo de hit) de 1 ciclo de reloj. Asuma que la penalidad de lectura y de escritura son las mismas e ignore cualquier otro atasco de escritura.
Respuesta. El tiempo promedio de acceso a memoria por instrucción es:

$$\begin{aligned} AMAT &= Tiempo\ de\ hit + Tasa\ de\ fallo \times Penalidad\ por\ miss \\ &= 1 + 0,05 \times 20 \\ &= 2\ ciclos\ de\ reloj \end{aligned}$$

o 2 ns □
La siguiente subsección discute orgnizaciones de cache alternativas que decrementan la tasa de fallo pero a veces pueden incrementar el tiempo de hit.

1. Reducción de los misses de cache por una ubicación más flexible de los bloques.

Hasta ahora, para ubicar un bloque en la cache se usó un método simple: un bloque puede ir exactamente en un lugar en la cache. Como se mencionó anteriormente, es llamado *correspondencia directa* porque hay una transformación directa de la dirección de cualquier bloque de memoria a una única ubicación en el nivel superior de la jerarquía. Sin embargo, realmente existe un rango entero de esquemas para ubicar bloques. La correspondencia directa, donde un bloque puede ser ubicado en exactamente un lugar, es un extremo.

Concepto 1. *Cache totalmente asociativa. Una estructura de cache en la cual un bloque puede ser ubicado en cualquier posición de la cache.*

En el otro extremo existe un esquema en el cual un bloque puede ser ubicado en *cualquier* lugar de la cache. Este esquema es llamado *totalmente asociativo*, porque un bloque en memoria puede ser asociado con cualquier entrada en la cache. Para encontrar un bloque determinado en una cache totalmente asociativa, todas las entradas de la cache deben ser consideradas porque un bloque puede ser ubicado en cualquiera de ellas. Para hacer que la búsqueda sea práctica, se hace en paralelo con un comparador asociado con cada entrada

en la cache. Estos comparadores incrementan significativamente el costo del hardware, lo que resulta en que la localización completamente asociativa sea practica solamente para caches pequeñas.

Concepto 2. *Cache de conjunto asociativo. Una cache que tiene un número fijo de ubicaciones (al menos dos) donde cada bloque puede ser ubicado.*

El rango intermedio de diseños entre correspondencia directa y completamente asociativa es llamado *asociativa por conjunto*. En una cache asociativa por conjunto, existe un número fijo de ubicaciones donde cada bloque puede ser localizado. Una cache asociativa por conjunto con n posiciones para un bloque es llamada asociativa por conjunto de *n-vías*. Una cache asociativa por conjuntos de n-vías consiste en un número de conjuntos, cada uno de los cuales tiene n bloques. Cada bloque en la memoria se corresponde con un único conjunto en la cache de acuerdo al campo índice y un bloque puede ser ubicado en cualquier lugar dentro del conjunto. De esa manera la ubicación asociativa por conjunto, combina la ubicación de correspondencia directa y la ubicación totalmente asociativa: Un bloque se corresponde directamente con un conjunto y luego todos los bloques dentro del conjunto son considerados para ver si hay coincidencia. Por ejemplo la Figura 1 muestra donde se puede ubicar el bloque 12 en una cache con un total de 8 bloques, de acuerdo a las tres políticas de ubicación de bloques.

FIGURE 1. **Ubicación de un bloque de memoria cuya dirección es 12 en una cache de 8 entradas, variando para las distintas localizaciones: correspondencia directa, asociativa por conjunto y completamente asociativa.** En localización por correspondencia directa, hay solamente un bloque de cache donde el bloque 12 puede ser encontrado y aquel bloque está dado por (12 mod 8) = 4. En una cache asociativa por conjunto de dos vías habrá cuatro conjuntos y el bloque de memoria 12 estará en el conjunto (12 mod 4) = 0; el bloque de memoria podría estar en cualquier elemento del conjunto. En una localización completamente asociativa, el bloque de memoria para la dirección de bloque 12 puede aparecer en cualquiera de los ocho bloques de cache

Recordar que en una cache de correspondencia directa, la posición de un bloque de memeoria está dada por:

$$(\textit{Número de bloque en memoria})\ \textit{módulo}\ (\textit{Número de}\ \textbf{bloques}\ \textit{en la cache})$$

En una cache de conjunto asociativo, el conjunto que contiene un bloque de memoria está dado por:

$$(\textit{Número de bloque en memoria})\ \textit{módulo}\ (\textit{Número de}\ \textbf{conjuntos}\ \textit{en la cache})$$

Considerando que los bloques pueden ser ubicados en cualquier elemento del conjunto, *todos los tags de todos los elementos del conjunto* deben ser comparados. En una cache totalmente asociativa, el bloque puede ir donde sea y *todos los tags de todos los elementos de la cache* deben ser comparados. Es posible considerar que todas las estrategias de ubicación de bloques son variantes de conjunto asociativo. La Figura 2 muestra las estructuras asociativas posibles para una cache de ocho bloques. Una cache de correspondencia directa, es simplemente una cache de conjunto asociativo de una vía: cada entrada en la cache mantiene un bloque y cada conjunto tiene un único elemento. Una cache totalmente asociativa con m entradas es simplemente una cache asociativa por conjunto de m-vías, tiene un conjunto con m bloques y una entrada puede residir en cualquier bloque dentro de ese conjunto.

La ventaja de aumentar el grado de asociatividad es que habitualmente decrementa la tasa de fallo, como el siguiente ejemplo lo muestra. La desventaja principal, que será discutida más adelante, es un potencial incremento en el tiempo de hit.

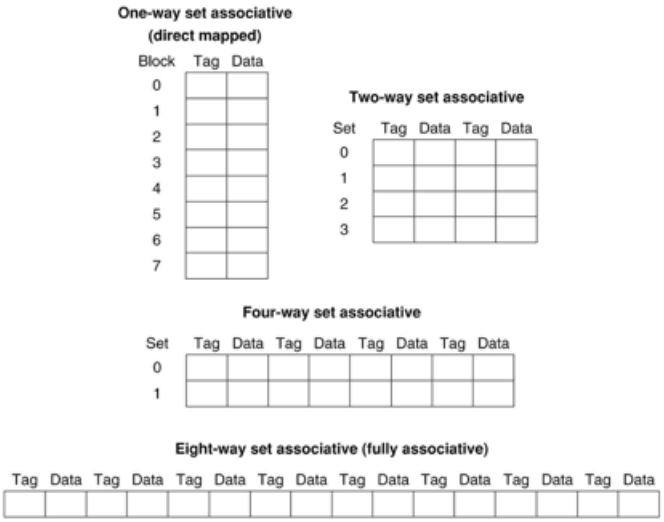

FIGURE 2. **Una cache de ocho bloques configurada como correspondencia directa, asociativa por conjunto de dos vías, asociativa por conjunto de cuatro vías, y completamente asociativa.** El tamaño total de la cache en bloques es igual al número de conjuntos multiplicado por la asociatividad. De esa manera, para un tamaño de cache fijo, incrementar la asociatividad disminuye el número de conjuntos, mientras se incrementa el número de elementos por conjunto. Con ocho bloques, una cache asociativa por conjunto de ocho vías es lo mismo que una cache completamente asociativa.

Ejemplo. Fallos y asociatividad en caches. Asumiendo que hay tres caches pequeñas, cada una consistente de cuatro bloques de una palabra. Una cache es totalmente asociativa, otra es asociativa por conjunto de dos vías y la tercera de correspondencia directa. Encontrar el

número de fallos para cada organización de cache para la siguiente secuencia de direcciones de bloques: 0, 8, 0, 6 y 8.
Respuesta. El caso de correspondencia directa es el primero. Primero, hay que determinar a cual bloque de cache corresponde cada dirección de bloque:

Dirección de bloque	Bloque de cache
0	(0 modulo 4)=0
6	(6 modulo 4)=2
8	(8 modulo 4)=0

Ahora es posible completar los contenidos de la cache después de cada referencia, usando una entrada en blanco para significar que el bloque no es válido, el texto en colores muestra que se agregó una nueva entrada en la cache, para la referencia asociada y el texto habitual muestra una entrada antigua en la cache.

Dirección de memoria del bloque accedido	hit o miss	Bloques de cache después de cada referencia			
		0	1	2	3
0	miss	Memoria[0]			
8	miss	Memoria[8]			
0	miss	Memoria[0]			
6	miss	Memoria[0]		Memoria[6]	
8	miss	Memoria[8]		Memoria[6]	

La cache de correspondencia directa genera cinco fallos para los cinco accesos.

La cache asociativa por conjunto tiene dos conjuntos (con índices 0 y 1) con dos elementos por conjunto. Lo primero es determinar a que conjunto corresponde cada bloque:

Dirección de bloque	Conjunto en cache
0	(0 modulo 2)=0
6	(6 modulo 2)=0
8	(8 modulo 2)=0

Debido a que se puede elegir que entrada del conjunto reemplazar en un miss, se necesita una regla de reemplazo. La cache de conjunto asociativo usualmente reemplaza el bloque menos recientemente referenciado dentro del conjunto; es decir que se reemplaza el bloque que fue usado hace más tiempo en el pasado. Usando esta regla de reemplazo, los contenidos de la cache de conjunto asociativo después de cada referencia se ven de este modo:

Notar que cuando el bloque 6 es referenciado, reemplaza al bloque 8, debido a que el bloque 8 ha sido menos recientemente referenciado que el bloque 0. Las cache asociativas de dos vías, tiene cuatro misses, uno menos que la cache de correspondencia directa.

La cache totalmente asociativa tiene cuatro bloques de cache (en un único conjunto); cualquier bloque de memoria puede ser almacenado en cualquier bloque de cache. La cache totalmente asociativa tiene la mejor performance, con solamente tres misses:

Para esta serie de referencias, tres fallos es lo mejor que se puede conseguir, debido a que se accede a tres direcciones de bloques distintas. Notar que si se tuviesen ocho bloques en

Dirección de memoria del bloque accedido	hit o miss	Bloques de cache después de cada referencia			
		Set 0	Set 0	Set 1	Set 1
0	miss	Memoria[0]			
8	miss	Memoria[0]	Memoria[8]		
0	hit	Memoria[0]	Memoria[8]		
6	miss	Memoria[0]	Memoria[6]		
8	miss	Memoria[8]	Memoria[6]		

Dirección de memoria del bloque accedido	hit o miss	Bloques de cache después de cada referencia			
		Bloque 0	bloque 1	Bloque 2	Bloque 3
0	miss	Memoria[0]			
8	miss	Memoria[0]	Memoria[8]		
0	hit	Memoria[0]	Memoria[8]		
6	miss	Memoria[0]	Memoria[8]	Memoria[6]	
8	hit	Memoria[0]	Memoria[8]	Memoria[6]	

la cache, podría no haber reemplazo en la cache asociativa por conjunto (el lector debiera verificarlo) y podría tener el mismo número de fallos que la cache totalmente asociativa. De igual manera si se cuenta con 16 bloques, las tres caches podrían tener el mismo número de fallos. Con este ejemplo simple se observa que el tamaño de la cache y la asociatividad no son independientes en la determinación del desempeño de la cache.□

¿Cuánta mejora se consigue en la tasa de fallo por medio de la asociatividad? La Figura 3 muestra la mejora para una cache de datos de 64 KiB con un bloque de 16 palabras, una asociatividad variando desde correspondencia directa a 8-vías. El paso de una-vía a dos-vías decrementa la tasa de fallos en un 15%, pero luego hay poca ganancia adicional al ir a una mayor asociatividad.

Associativity	Data miss rate
1	10.3%
2	8.6%
4	8.3%
8	8.1%

FIGURE 3. **Tasa de fallos de la cache de datos para una organización de cache como la del procesador FastMATH para el benchmark SPEC CPU2000 con la asociatividad variando de una a ocho-vías.**

1.1. **Localizando un bloque en la cache.** Es momento de considerar la tarea de encontrar un bloque en una cache que es asociativa por conjunto. Al igual que en la cache de correspondencia directa, cada bloque en una cache de conjunto asociativo incluye una etiqueta de acuerdo a la dirección del bloque. La etiqueta de cada bloque de cache dentro del conjunto correspondiente, es comparada para ver si coincide con la dirección de bloque proveniente del procesador. La Figura 4 descompone la dirección. El valor del índice se usa para seleccionar el conjunto que contiene la dirección de interés y las etiquetas de todos los bloques en el conjunto que deben ser comparadas. Debido a que la velocidad es escencial,

todas las etiquetas en el conjunto seleccionado son consideradas en paralelo. Al igual que en una cache totalmente asociativa, una búsqueda secuencial podría hacer demasiado lento el tiempo de hit de una cache asociativa por conjunto.

FIGURE 4. **Las tres porciones de una dirección en una cache asociativa por conjunto o correspondencia directa.** El índice es usado para seleccionar el conjunto, luego el tag se usa para elegir el bloque por comparación con los bloques en el conjunto seleccionado. El desplazamiento del bloque, es la dirección del dato deseado dentro del bloque.

Si el tamaño total de cache se mantiene, el incremento de la asociatividad incrementa el número de bloques por conjunto, lo cual es el número de comparadores simultáneos necesarios para hacer la búsqueda en paralelo: cada incremento en la asociatividad por un factor de 2, duplica el número de bloques por conjunto y divide por 2 el número de conjuntos. De acuerdo a eso, cada incremento en un factor de dos de la asociatividad, decrementa en un bit el tamaño del índice e incrementa el tamaño del tag en un bit. En una cache totalmente asociativa, hay realmente un único conjunto y todos los bloques deben ser comparados en paralelo. Así, no hay índice y la dirección completa, excluyendo el desplazamiento en el bloque, debe ser comparada con la etiqueta de cada bloque. En otras palabras, se busca en la cache completa sin ninguna indexación.

En una cache de correspondencia directa, solamente se necesita un comparador, debido a que la entrada puede estar solamente en un bloque y se accede a la cache mediante el índice. La Figura 5 muestra que en una cache asociativa por conjuntos de cuatro vías, se necesitan cuatro comparadores conjuntamente con un multiplexor de 4 a 1 para elegir entre los cuatro potenciales miembros del conjunto seleccionado. Los accesos a cache consisten en indexar el conjunto apropiado y luego buscar el tag en el conjunto. Los costos de una cache asociativa son los comparadores extras y toda la demora impuesta por tener que hacer la comparación y seleccionar entre los elementos del conjunto.

La elección entre correspondencia directa, asociativa por conjunto y completamente asociativa en una jerarquía de memoria depende del costo de un fallo vs el costo de implementar la asociatividad, tanto en tiempo como en hardware extra.

Elaboración. Una memoria direccionable por contenido (CAM por sus siglas en inglés) es un circuito que combina comparaciones y almacenamiento en un solo dispositivo. En lugar de proveer una dirección y leer una palabra como ocurre en una RAM, se provee el dato y la CAM busca para ver si tiene una copia y retorna el índice de la fila que coincide. Las CAMs permitieron a los diseñadores implementar caches con una asociatividad por conjunto mucho más alta, en comparación a la alcanzada cuando tenían que construir el hardware partiendo de SRAMs y comparadores. En 2013 era habitual que empleando SRAMs y comparadores se alcanzara una asociatividad de conjunto de 2-vías o 4-vías, con el actual incremento en tamaño y poder de las CAM se llega a 8-vías o más.□

1.2. **Eligiendo que bloque reemplazar.** Cando un miss ocurre en una cache de correspondencia directa, el bloque requerido puede ir exactamente en una posición y el bloque

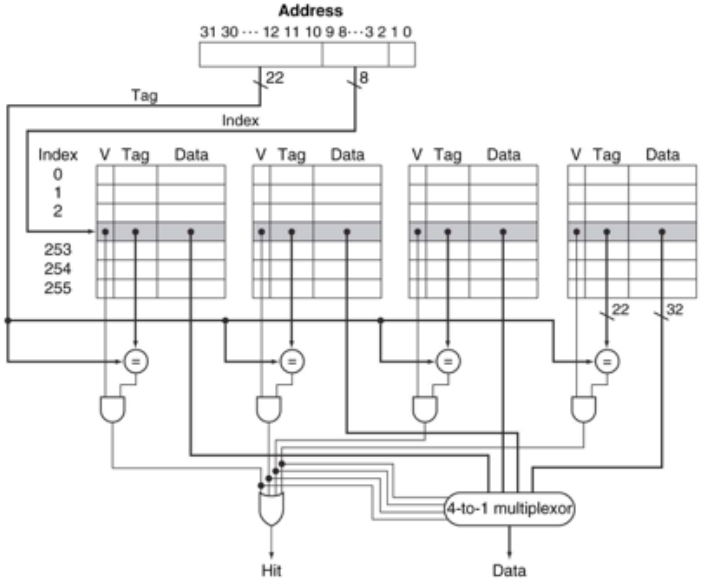

FIGURE 5. **La implementación de una cache asociativa de cuatro vías requiere cuatro comparadores y un multiplexor de 4 a 1.** Los comparadores determinan que elemento del conjunto seleccionado (si hay alguno) se corresponde con el tag. La salida de los comparadores se usa para seleccionar el dato de uno de los cuatro bloques del conjunto indexado, usando un multiplexor con una señal decodificada. En algunas implementaciones, las señales de habilitación de salida en las partes de datos de las RAM de cache se pueden usar para seleccionar la entrada en el conjunto que controla la salida. La señal de habilitación de salida proviene de los comparadores, lo que hace que el elemento que coincide conduzca las salidas de datos. Esta organización elimina la necesidad del multiplexor.

que ocupa esa posición debe ser reemplazado. En una cache asociativa, se tiene que elegir donde ubicar el bloque requerido y en consecuencia elegir que bloque reemplazar. En una cache totalmente asociativa, todos los bloques son candidatos para reemplazar. En una cache asociativa por conjunto, se debe elegir entre los bloques del conjunto seleccionado.

El esquema más comúnmente usado es LRU (por sus siglas en inglés), el *menos recientemente usado*, que fue aplicado en el ejemplo previo. En un esquema LRU, el bloque reemplazado es el que ha estado más tiempo sin ser usado. El ejemplo de conjunto asociativo de la página 6, usa LRU, esa es la razón por la cual se reemplaza Memoria[0] en lugar de Memoria[6].

El reemplazo LRU se implementa manteniendo información de cuando cada elemento de un conjunto fue usado en relación a otros elementos del conjunto. Para una cache asociativa de dos-vías, mantener información de cuando los dos elementos fueron usados, puede ser implementado mediante un único bit en cada conjunto y alterando el bit para indicar un elemento cada vez que el elemento es referenciado. A medida que se aumenta la asociatividad, la implementación de LRU se hace más difícil, existen otros esquemas alternativos para el reemplazo.

References

[1] Patterson, David A. and Hennessy, John L., Computer Organization and Design, Fifth Edition: The Hardware/Software Interface, Morgan Kaufmann Publishers Inc., 2014