Organización del caché

Organización del caché

- Hasta ahora solo se ha visto la estrategia de mapeo directo.
- Un bloque solo puede ir en un lugar en el caché.
- Otras organizaciones pueden reducir la tasa de fallas:
  - Fully associative cache. El bloque puede ir en cualquier parte en el caché.
  - Set associative cache. El bloque de memoria puede ir en cualquier parte dentro del conjunto que le toque.

Fully associative cache

- Un bloque en memoria puede estar asociado con cualquier línea en el caché.
- Para encontrar un bloque se debe buscar en todo el caché.
- Para ser práctica, la búsqueda se hace en paralelo asociando un comparador con cada línea del caché.
- Los comparadores incrementan el costo del hardware.
- Son prácticos solo para cachés pequeños.

Set associative cache

- El caché está dividido en *m* conjuntos.
- Cada conjunto consta de *n* bloques.
- Se le llama n-way set associative.
- Un bloque en memoria solo puede ir en un conjunto.
- Dentro del conjunto, el bloque puede ir en cualquier lugar.
- Un dato se busca en todos los bloques del conjunto.
- Las fórmulas para calcular el bloque en el caché son *casi* las mismas que en mapeo directo.

Set associative cache

- La diferencia es que se usa el número de conjunto y no el número de bloque.
- $d = a$ **div** k
- *d* es la dirección de bloque
- *a* es la dirección del dato en la memoria
- *k* es el tamaño del bloque en bytes
- $b = d$ **mod** n
- *b* es el número de bloque
- *n* es el número de ***conjuntos*** que tiene el caché

Set associative cache

- La división de la dirección en binario funciona *casi* igual que en mapeo directo:
- Offset. La posición del byte dentro del bloque. Ocupa los bits bajos de la dirección. Tamaño = $\log_2(k)$, k es el tamaño del bloque en bytes.
- Índice. El número de conjunto en donde se guarda el dato. No hay forma de saber en que bloque se guarda el dato. Ocupa los bits intermedios de la dirección. Tamaño = $\log_2(n)$, n es el número de conjuntos que tiene el caché.

Set associative caché

- Etiqueta. Ocupa los bits altos de la dirección.

Ejemplo

- Suponer un caché 4-way con capacidad de 4K bytes y bloques de 4 palabras. Las direcciones son de 32 bits.
- ¿Cuántos bloques y cuántos conjuntos tiene el caché?
- ¿Cómo se divide la dirección?
- ¿Qué conjunto en el caché le toca a la dirección en la memoria 1714?

Ejemplo

- ¿Cuántos bloques y cuántos conjuntos tiene el caché?
- Número de bloques:

$$\text{bloques} = \frac{4K}{4W} = \frac{4 \times 1024}{4 \times 4} = 256 \text{ bloques}$$

- Es un cache 4-way, por lo tanto cada conjunto tiene 4 bloques, es decir:

$$\text{conjuntos} = \frac{256}{4} = 64 \text{ conjuntos}$$

Ejemplo

- ¿Cómo se divide la dirección?

- Tamaño del offset = $\log_2(16) = 4$ bits
- Tamaño del índice = $\log_2(64) = 6$ bits
- Tamaño de la etiqueta = $32 - (6 + 4) = 22$ bits

Ejemplo

- ¿Qué conjunto en el caché le toca a la dirección en la memoria 1714?
- Método 1 – usando las fórmulas.
- Datos: a = 1714; k = 16; n = 64

- d = 1714 **div** 16 = 107
- b = 107 **mod** 64 = 43
- Conclusión: la dirección 1714 se guarda en el conjunto 43.

Ejemplo

- ¿Qué conjunto en el caché le toca a la dirección en la memoria 1714?
- Método 2 – analizando la dirección en binario.
- $1714_{10} = 11010110010_2 = 1\text{-}101011\text{-}0010$
- Se convierte el índice a base 10.
- $101011_2 = 43_{10}$
- Conclusión: la dirección 1714 se guarda en el conjunto 43.

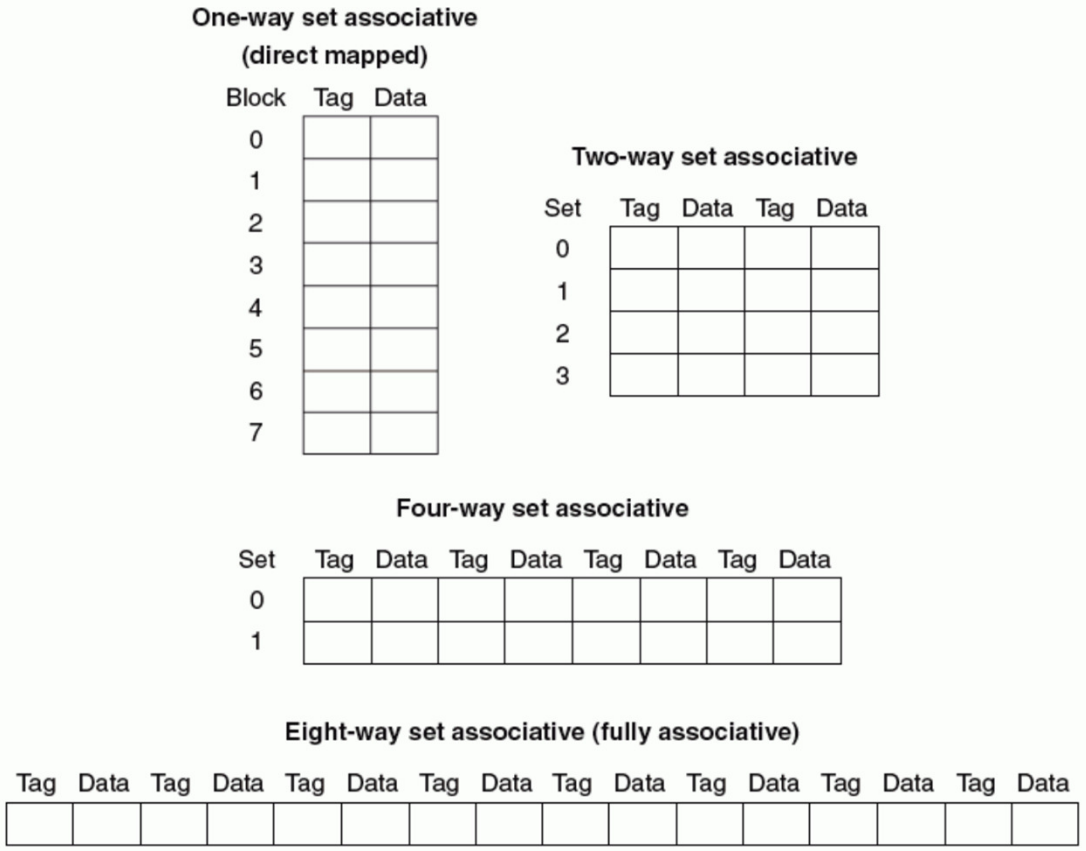
Variantes de una forma

- Mapeo directo y fully associative se pueden ver como variantes de set associative.

1. Mapeo directo es 1-way set associative.
   - Un caché de *n* bloques se puede ver como un caché de *n* conjuntos en donde cada conjunto tiene 1 bloque.
2. Fully associative es n-way set associative.
   - Un caché de *n* bloques se puede ver como un caché de 1 conjunto de *n* bloques.

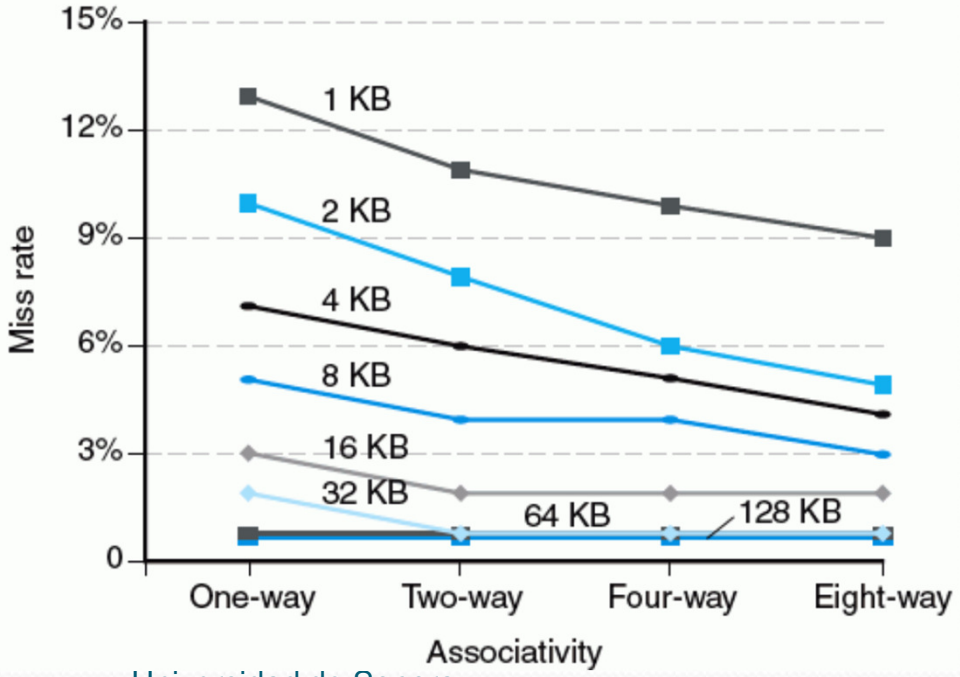
Variantes de una forma

- Variantes para un caché de 8 bloques.



Ventajas y desventajas…

- de incrementar el grado de asociatividad.
- Ventaja: generalmente decrementa la tasa de fallas.
- Desventaja: incrementa el tiempo de éxito (hit time).



Ejemplo

- Hay 3 cachés.
- Cada uno tiene 4 bloques de 1 byte.
- Un caché es de mapeo directo, otro 2-way set associative y otro fully associative.
- Encontrar el número de fallas para cada organización dada la siguiente secuencia de direcciones:
- *a* = 0, 8, 0, 6, 8.

Mapeo directo

- Determinar en que bloque se mapea cada dirección.

Block address	Cache block
0	(0 modulo 4) = 0
6	(6 modulo 4) = 2
8	(8 modulo 4) = 0

Address of memory block accessed	Hit or miss	Contents of cache blocks after reference			
		0	1	2	3
0	miss	Memory[0]			
8	miss	Memory[8]			
0	miss	Memory[0]			
6	miss	Memory[0]		Memory[6]	
8	miss	Memory[8]		Memory[6]	

- 5 fallas.

2-way set associative

- Hay dos conjuntos (0 y 1).
- Determinar en que conjunto se mapea cada dirección.

Block address	Cache set
0	(0 modulo 2) = 0
6	(6 modulo 2) = 0
8	(8 modulo 2) = 0

- Se necesita una regla de reemplazo.
- Se supone LRU (el menos usado recientemente).

2-way set associative

Address of memory block accessed	Hit or miss	Contents of cache blocks after reference			
		Set 0	Set 0	Set 1	Set 1
0	miss	Memory[0]			
8	miss	Memory[0]	Memory[8]		
0	hit	Memory[0]	Memory[8]		
6	miss	Memory[0]	Memory[6]		
8	miss	Memory[8]	Memory[6]		

- El bloque 6 reemplaza al 8 por ser el menos usado recientemente.
- 4 fallas.

Fully associative

- Los bloques pueden ir donde sea.

Address of memory block accessed	Hit or miss	Contents of cache blocks after reference			
		Block 0	Block 1	Block 2	Block 3
0	miss	Memory[0]			
8	miss	Memory[0]	Memory[8]		
0	hit	Memory[0]	Memory[8]		
6	miss	Memory[0]	Memory[8]	Memory[6]	
8	hit	Memory[0]	Memory[8]	Memory[6]	

- 3 fallas.

Efecto de la asociatividad

- Caché de datos de 64 KB con bloques de 16 palabras.
- Asociatividad varía desde 1-way (mapeo directo) hasta 8-way.
- Benchmark SPEC2000.

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

Comparación

Characteristic	ARM Cortex-A8	Intel Nehalem
L1 cache organization	Split instruction and data caches	Split instruction and data caches
L1 cache size	32 KiB each for instructions/data	32 KiB each for instructions/data per core
L1 cache associativity	4-way (I), 4-way (D) set associative	4-way (I), 8-way (D) set associative
L1 replacement	Random	Approximated LRU
L1 block size	64 bytes	64 bytes
L1 write policy	Write-back, Write-allocate(?)	Write-back, No-write-allocate
L1 hit time (load-use)	1 clock cycle	4 clock cycles, pipelined
L2 cache organization	Unified (instruction and data)	Unified (instruction and data) per core
L2 cache size	128 KiB to 1 MiB	256 KiB (0.25 MiB)
L2 cache associativity	8-way set associative	8-way set associative
L2 replacement	Random(?)	Approximated LRU
L2 block size	64 bytes	64 bytes
L2 write policy	Write-back, Write-allocate (?)	Write-back, Write-allocate
L2 hit time	11 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

Conclusión

- Los cachés n-way set associative (n > 1) por lo general tienen tasas de fallas menores que los cachés de mapeo directo.
- Los cachés n-way set associative (n > 1) tienen mayor tiempo de éxito que los cachés de mapeo directo.
- Se puede reducir el tiempo de éxito usando n comparadores.