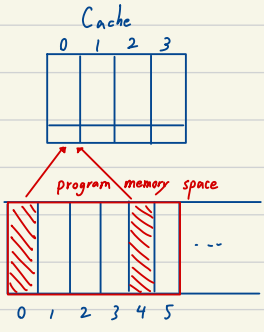
Set Associative Cache



設 0 号 block 最後一個 word 為 branch 指令會跳至 4 号 block

且 4 号 block 最後一個 word 為 jump 指令會跳至 0 号 block

根據 direct mapped 的分配方式，0 号 block 會對應到 0 号 cache block

但 4 号 block 會對應到 0 号 cache block，設使用的是 Write-back cache

這樣一來，交錯執行的指令會競爭相同 cache block，使 miss rate 很高

且根據 direct mapped，cache block 1, 2, 3 皆 idle

∴ 把 cache block 切割成多個集合 ⇒ 集合內多個 cache blocks ⇒ 當競爭發生時，分配不同 cache block，降低 miss rate 也.

∴ MEM 到 cache 的方式變成：memory block address % number of sets

得到餘數為分配到的集合，可分配 set 內任意 block 給它

商數為 tag

而 associativity 為一個集合內的 block 總數，又稱 N-way

Associativity 越高，miss rate 就越低，但 hit time 會增加 (∵ 需比對的 tag 數上升)

Question：設 Memory 中有 32 個 blocks，若使用 direct mapped cache，則：0, 4, 8, 12, 16, 20, 24, 28 會競爭 block 1

而使用 2-way associative cache，則：偶數号競爭 set 0 的 2 個 block，競爭比例相同，為何可降低 cache miss？

∵ principle of locality 的關係，每個 block 被用到的机率不均等

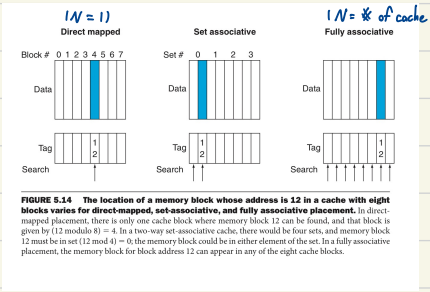


FIGURE 5.14 The location of a memory block whose address is 12 in a cache with eight blocks varies for direct-mapped, set-associative, and fully associative placement. In direct-mapped placement, there is only one cache block where memory block 12 can be found, and that block is given by (12 modulo 8) = 4. In a two-way set-associative cache, there would be four sets, and memory block 12 must be in set (12 mod 4) = 0; the memory block could be in either element of the set. In a fully associative placement, the memory block for block address 12 can appear in any of the eight cache blocks.

MEM byte address 至 cache address 的轉換

做 2 次除法：① 看一個 block 有几个 byte：得 MEM block address 和 offset

② 看 cache 有几个集合：得 tag 和 index

Note：tag 需和所有 set 中的 block 同時做比对 ⇒ 使用比較器平行比較所有 entry

當要從 MEM 中搬一個 block 資料至 cache，但对應的 set 全滿時

需挑選 cache block 置換，法則：① random choice，類似 OS 中 page replacement

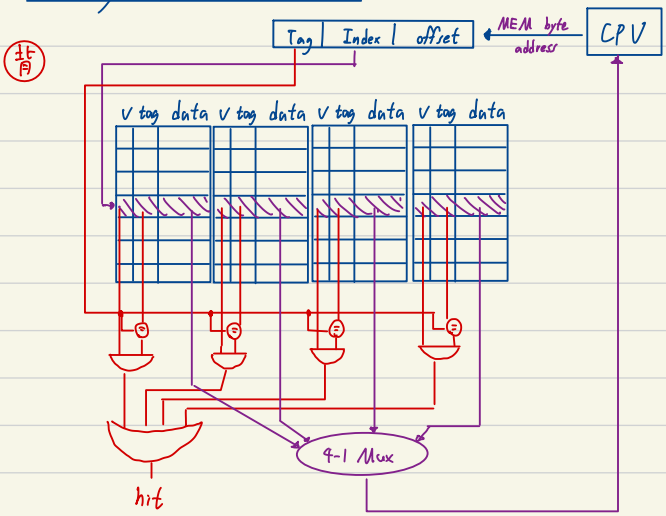
② LRU

<Review>：Cache block 總數 = Cache size / Block size

公式：Cache block 總數 = # of sets × associativity

可先求出 cache block 總數，再求得集合總數

① 畫 4-way set associative cache



② Set Associative Cache 計算題

給定：<1>. Cache size

<2>. Block size

<3>. Address length

<4>. Associativity

① 求 Block 總數：$\frac{<1>}{<2>}$

② 求 Set 總數：$\frac{①}{<4>}$ ⇒ 先算第二小

③. 切位址求 tag bit 數

④. 求 Cache 的 bit 總數

[v + tag + data] × Associativity × 集合總數

[v + tag + data] × block 總數

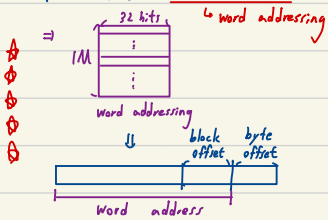
<Ex P.110 16題 (同類型最難)

已知：①. block size：4 個 words

②. associativity = 4

③. Cache size = 16 K words

④. MEM size：1M × 32 bits ↳ word addressing

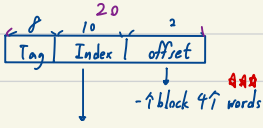


∴ CPU 丟出來的為 MEM word address

∵ MEM 共有 1M 個位址，

word address 共 20 個 bits

集合總數 = $\frac{16K}{4}$ ← block 總數 / 4 ← 4-way



集合總數：1K

<Ex P.111 18題> 已知：address length：k bits

block size：B bytes，$B = 2^b$

cache size：S bytes

associativity：A

求 cache 的需總 bits 數 in terms of k, B, b, S, A

①. block 總數 in cache：$\frac{S}{B}$

②. set 總數：$\frac{S}{AB}$

③. index 的 bits 數：$\lg\left(\frac{S}{AB}\right)$

④. tag 的 bits 數：$k - b - \lg\left(\frac{S}{AB}\right)$

∴ Cache 的需總 bits 數：$\left[k - b - \lg\left(\frac{S}{AB}\right) + 1 + 8B\right] \cdot \frac{S}{B}$

☆☆☆

<Ex>: Given 2^S-byte, 2^L-byte-per-line cache in an M-bit, byte-addressable memory system

①. 求 range of index field size in number of bits.

②. 求 range of index field size in number of bits.

極端 極端

Note: range: direct-mapped ⇔ fully associative

已知: cache size: 2^S byte

block size: 2^L byte

address length: M-bit

	Tag	Index	offset
direct	M-S	S-L	L
fully	M-L	0	L

= 次 of blocks: 2^{S-L}

Note: ① write-back cache 中, cache 有 dirty bit 欄位來確認是否被寫入過資料 以節省 copy-back 時間

② 使用 LRU 法則的話, 需 LRU bit 記錄 least recently used 的 block

統稱這些 bits 為 extra bit & control bit

③. entry: fully / direct-mapped / set associative | block / block / set — 依題目判斷

Associativity 和 tag bits 的關係

設: 共有 4K 個 blocks, block size 為 4 words 且 address length 為 32 bits

	Direct	2-way	4-way	fully
# of sets	$\frac{4K}{1} = 4K$	$\frac{4K}{2} = 2K$	$\frac{4K}{4} = K$	1
	tag 16 | index 12 | offset 4	tag 17 | index 11 | offset 4	tag 18 | index 10 | offset 4	tag 28 | index 0 | offset 4
Total tag bits	$16 \times 4K = 64K$	$17 \times 4K = 68K$	$18 \times 4K = 72K$	$28 \times 4K = 112K$

可知: 當 associativity 越高 tag bits 數越多, 硬件成本越高

且比較器個數也增加

(15% total) In Table 2, we have given you four different sequences of addresses generated by a program running on a processor with a data cache. Cache hit ratio for each sequence is also shown in Table 2. Assuming that the cache is initially empty at the beginning of each sequence, find out the following parameters of the processors' data cache:

1 (3%) Associativity? (1, 2, or 4 ways)
2 (4%) Block size? (1, 2, 4, 8, 16, or 32 bytes)
3 (4%) Total cache size? (256 bytes, or 512 bytes)
4 (4%) Replacement policy? (LRU, or FIFO)

Table 2.

Sequence No.	Address Sequence	Hit Ratio
1	0, 2, 4, 8, 16, 32	0.33
2	0, 512, 1024, 1536, 2048, 1536, 1024, 512, 0	0.33
3	0, 64, 128, 256, 512, 256, 128, 64, 0	0.33
4	0, 512, 1024, 0, 1536, 0, 2048, 512	0.25

m h h m m m

sequence 1. hit ratio: $\frac{1}{3} = \frac{2}{6}$ 0, 2, 4, 8, 16, 32 ⇒ block size: 8 bytes

sequence 2. hit ratio: $\frac{3}{9}$ 0, 512, 1024, 1536, 2048, 1536, 1024, 512, 0

m m m m m 3個 hit

block address: 192, 128, 64, 0

case I. 4-way, 0, 512, 1024, 1536, 2048 皆為同一 set

case II. 2-way 0, 512, 1024 皆為同一 set

sequence 3. hit ratio: $\frac{2}{9}$ 0, 64, 128, 256, 512, 256, 128, 64, 0

m m m m m

block address: 32, 16, 8, 0

sequence 4. hit ratio: $\frac{2}{8}$ 0, 512, 1024, 0, 1536, 0, 2048, 512

m m m □ m □ m □

若為 case II, 0 皆為 miss

∴ 為 case I, 4-way

block address:

已知: block size: 4 way, 又 set 總數為 $\frac{\text{cache size}}{32} = \frac{C}{32}$ 又 $16\% \frac{C}{32} == 8\% \frac{C}{32}$

∴ C ≠ 512 bytes C = 256 bytes

再由 sequence 4 知為 lru