Friday, March 22, 2019

Fully associative caches (continued)

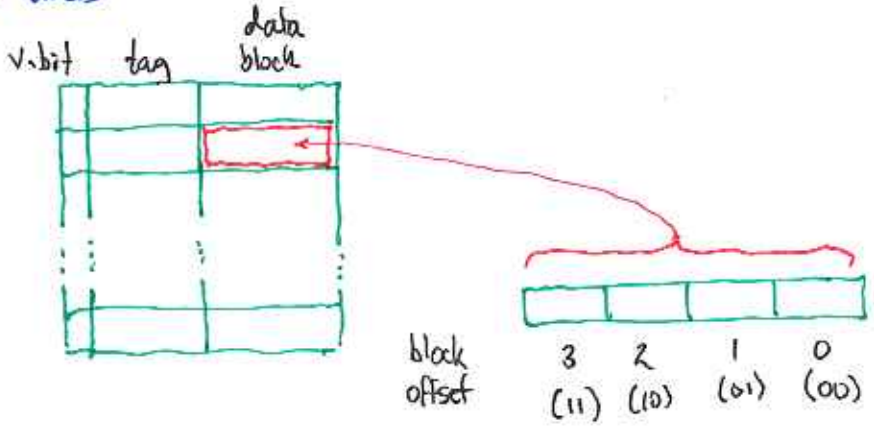
Well-suited to very small caches (e.g., virtual memory systems)

- Very fast lookups (simultaneous v-bit and tag checks)
- needs more hardware than a conventional cache.

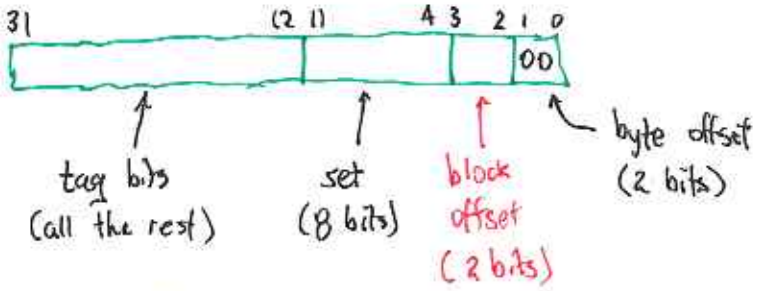
Caches with multi-word blocks

Suppose we have a direct-mapped cache, but now with multi-word blocks.

E.g., four-word blocks



Suppose there are 256 sets in the cache with a block size of 4.
Address is now broken down as:



Hits and misses in caches with multi-word blocks.

- Cache hit (v-bit = 1, tags match)
  - give word from the selected set to processor (block offset tells which word to use)

- Cache miss
  - all words of the block retrieved from main memory.
  - these words will receive a common v-bit and tag.

Which words get replaced on a cache miss:

Example.

- Block size is 4 words, 256 sets.
- Suppose address is 0x0040_0118 for a read request.

Address: 0000_0000_0100_0000_0000 | 0001_0001 | 10 | 00

- 0000_0000_0100_0000_0000 → tag = 0x00400
- 0001_0001 → set bits = 0x11
- 10 → block offset
- 00 → byte offset.

The following 4 words will be fetched:

0x00400 (tag) 0x11 (set)

block offset	byte offset	
00	00	
01	00	
10	00	← address of word requested
11	00	

(four words with block offsets 00 - 11)

Why multi-word blocks?

- much faster in modern DRAM to read a "burst" of N words (4, 8, 16, etc.) than N single words.
- most programs have spatial locality, so it makes sense to grab N words.

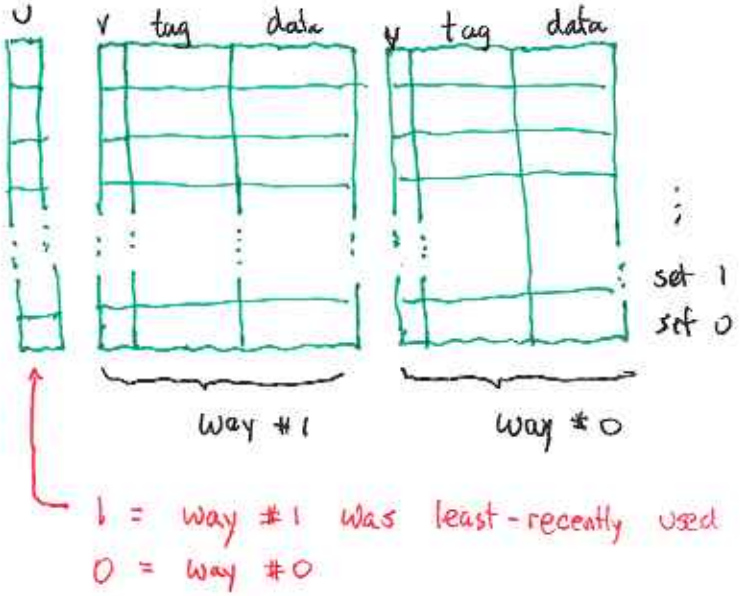
Example: program calling a procedure for the first time.

- I-cache misses on first instruction
- If procedure is short, the whole procedure may be read into the cache.

Replacement strategies in set-associate caches

In a 2-way set-associative cache, we may have to "evict" a previous entry to make room for a new one on a cache miss.

Strategy is called Least Recently Used (LRU). A U-bit is added to each set.



The U-bit points to which way gets the boot.

- Once a way is replaced, U points to the other
- For more than 2 ways, they are grouped. LRU group is selected, and a way within the group is selected at random.

Cache write policies

We have only considered loads from memory so far. What's involved is storing to memory?

Two strategies:

1. Write back
2. Write through.