Cache Organization

Consider the following function:

```
void vec_add (int x[16], int y[16], int s[16])
{
  int i;
  for (i=0; i < 16; i++)
     s[i] = x[i] + y[i];
  for (i=0; i < 16; i++)
     s[i] = s[i] + 1;
}
```

Simple direct-mapped cache: 128 byte cache, 16 byte blocks. Miss rate?

CSL

Addresses

Address of x: 0xc0 (tag=1, index=4)

00000000000000000000000001	100	00	00
25	3	2	2

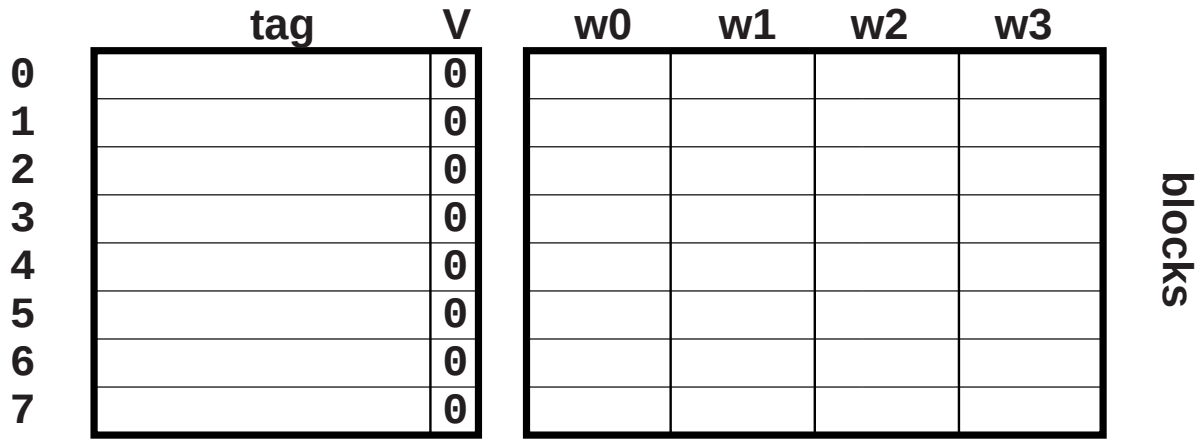
Address of y: 0x200 (tag=4, index=0)

00000000000000000000000100	000	00	00
25	3	2	2

Address of s: 0x2a0 (tag=5, index=2)

00000000000000000000000101	010	00	00
25	3	2	2

Direct-Mapped Cache



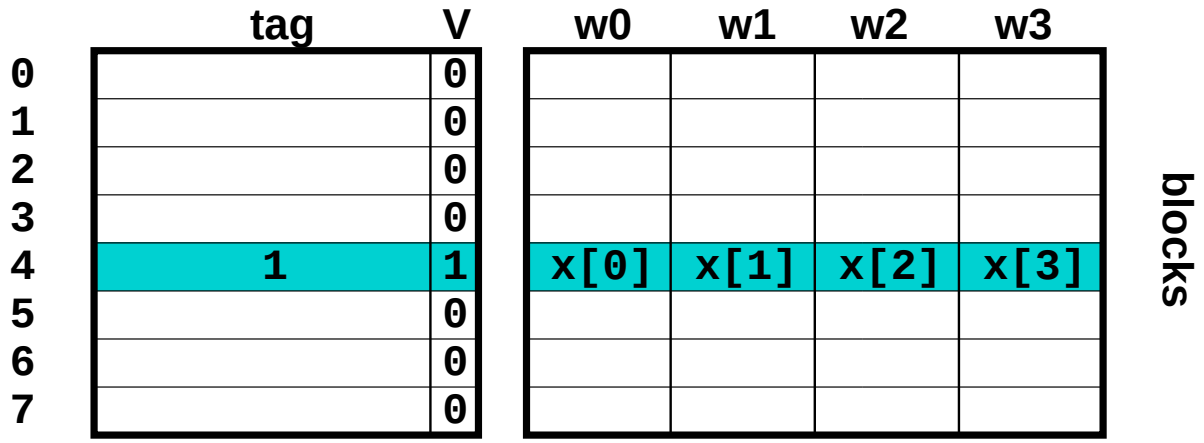
Request: load `x[0]`

Addresses:

x: tag=1, index=4, y: tag=4, index=0

s: tag=5, index=2

Direct-Mapped Cache



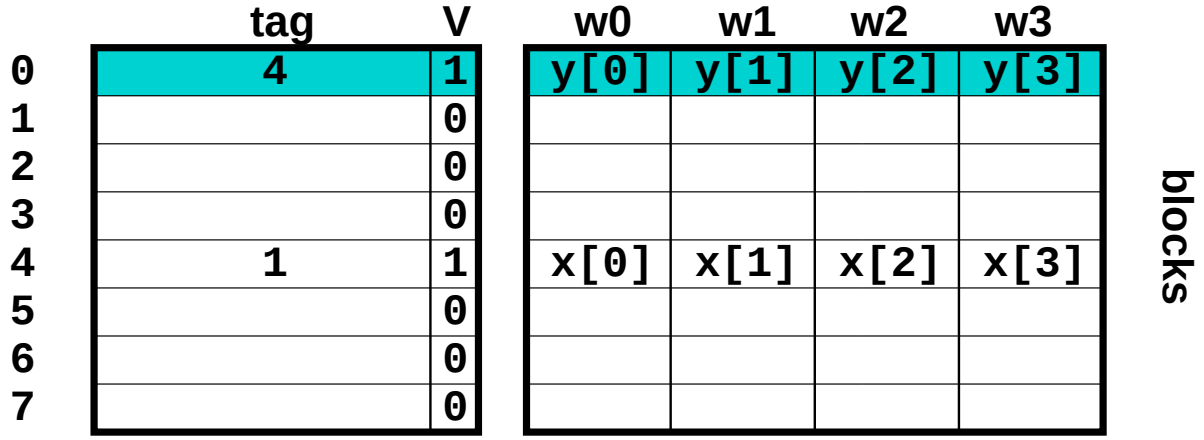
Request: load `y[0]`

Addresses:

x: tag=1, index=4, y: tag=4, index=0

s: tag=5, index=2

Direct-Mapped Cache



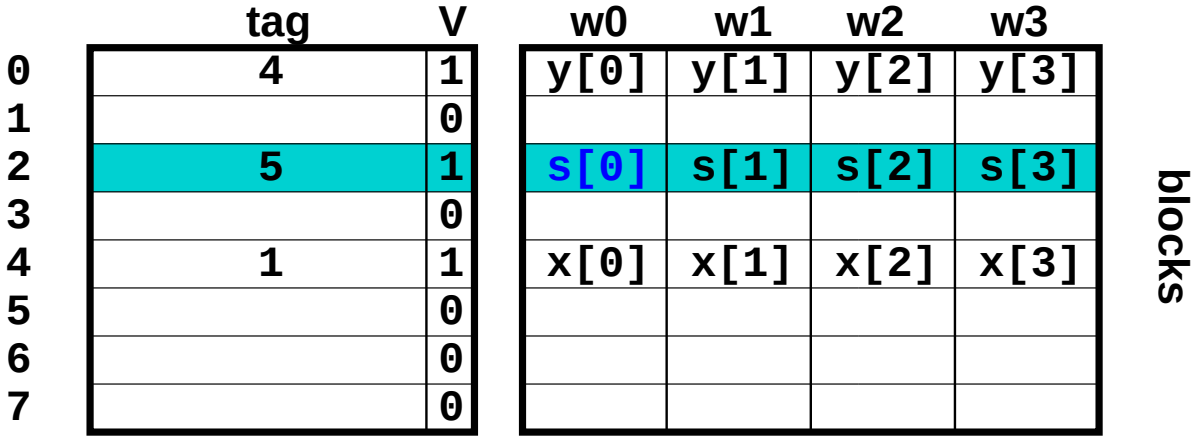
Request: **store** `s[0]`

Addresses:

x: tag=1, index=4, y: tag=4, index=0

s: tag=5, index=2

Direct-Mapped Cache



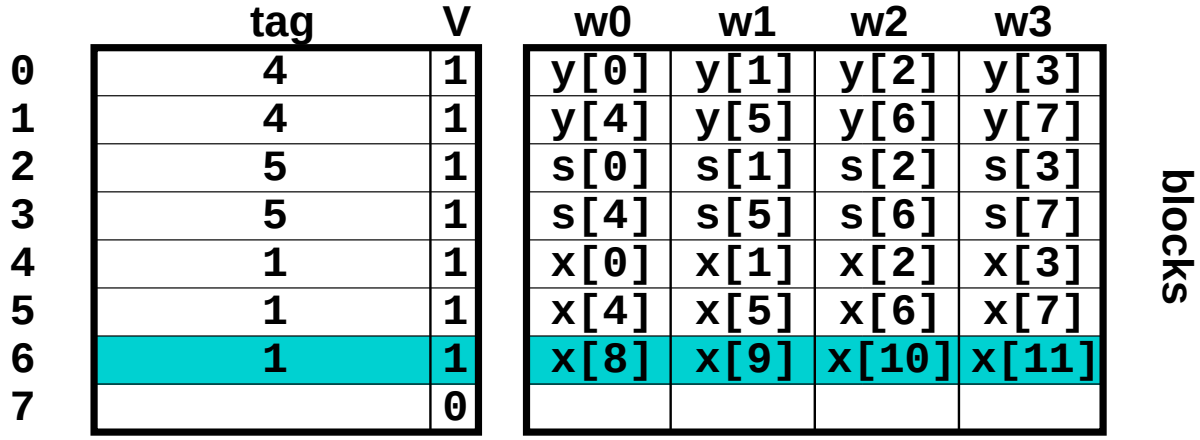
Request: load `x[1]`

Addresses:

x: tag=1, index=4, y: tag=4, index=0

s: tag=5, index=2

Direct-Mapped Cache



	tag	V	w0	w1	w2	w3
0	4	1	y[0]	y[1]	y[2]	y[3]
1	4	1	y[4]	y[5]	y[6]	y[7]
2	5	1	s[0]	s[1]	s[2]	s[3]
3	5	1	s[4]	s[5]	s[6]	s[7]
4	1	1	x[0]	x[1]	x[2]	x[3]
5	1	1	x[4]	x[5]	x[6]	x[7]
6	1	1	x[8]	x[9]	x[10]	x[11]
7		0				

blocks

Request: load `y[8]`

Addresses:

x: tag=1, index=4, y: tag=4, index=0

s: tag=5, index=2

Direct-Mapped Cache

	tag	V	w0	w1	w2	w3
0	4	1	y[0]	y[1]	y[2]	y[3]
1	4	1	y[4]	y[5]	y[6]	y[7]
2	4	1	y[8]	y[9]	y[10]	y[11]
3	5	1	s[4]	s[5]	s[6]	s[7]
4	1	1	x[0]	x[1]	x[2]	x[3]
5	1	1	x[4]	x[5]	x[6]	x[7]
6	1	1	x[8]	x[9]	x[10]	x[11]
7		0				

blocks

Request: **store** `s[8]`

Addresses:

x: tag=1, index=4, y: tag=4, index=0

s: tag=5, index=2

Direct-Mapped Cache

	tag	V	w0	w1	w2	w3
0	4	1	y[0]	y[1]	y[2]	y[3]
1	4	1	y[4]	y[5]	y[6]	y[7]
2	4	1	y[8]	y[9]	y[10]	y[11]
3	4	1	y[12]	y[13]	y[14]	y[15]
4	5	1	s[8]	s[9]	s[10]	s[11]
5	5	1	s[12]	s[13]	s[14]	s[15]
6	1	1	x[8]	x[9]	x[10]	x[11]
7	1	1	x[12]	x[13]	x[14]	x[15]

blocks

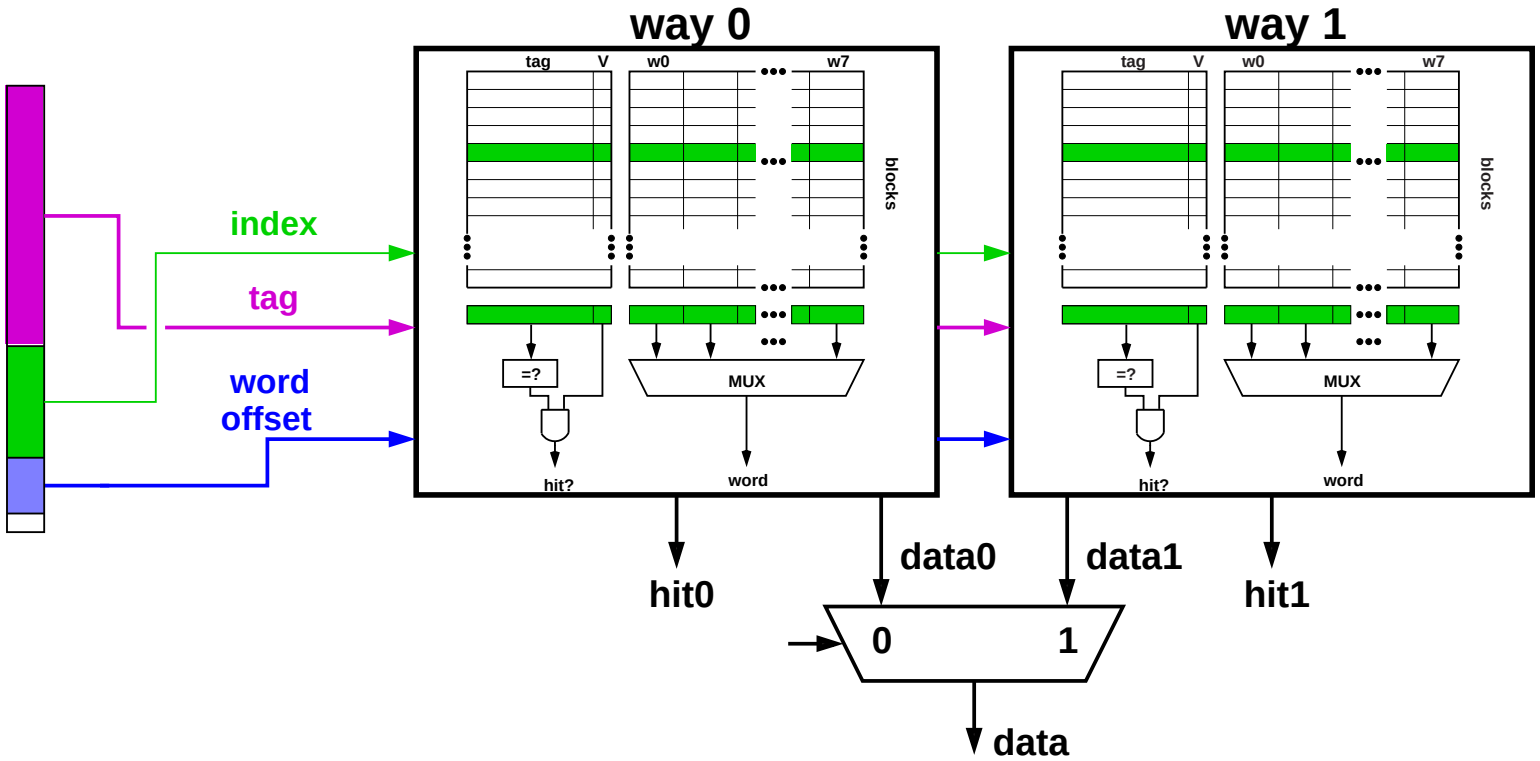
Request: load `s[0]`

Addresses:

s: tag=5, index=2

What if x, y, and s had the same index?

Set-Associative Cache



For a 4KB 2-way set associative cache, each set stores 2KB of data.

Fully associative: # of ways = # of blocks

Replacement Policy

Which block gets replaced on a miss?

- FIFO
  - round-robin replacement of blocks
- LRU (least recently used)
  - replace block that has been unused for the longest period
- Random
  - replace random block

Cache Misses

Classified into three different types.

The 3 C's:

- Compulsory (cold start)
  - first access to a piece of data
- Capacity
  - working set of program larger than cache size
- Conflict
  - collisions, multiple blocks competing for the same set