Cache Structure and Access

Caiwen Ding
Department of Computer Science and Engineering
University of Connecticut

Adapted from *Computer Organization and Design*
by Patterson & Hennessy

Cache

- Review access to a direct-mapped cache
- Direct-mapped cache
  - Diagram
  - Hardware cost
  - Handling hit and miss, on read and write
    - Write through/write back

Reading: 5.3.

Review

- What is cache hit?
- What is cache miss?
- How does cache check if a request is a hit?

Terms

Hit: data found in cache

Hit Rate: number of hits divided by number of memory accesses

Miss: data NOT in cache

Miss Rate: number of misses divided by number of memory accesses

What is hit rate plus miss rate?

Direct-Mapped Cache Example

Start with an empty cache (all blocks are not valid/blank).

Access the words at the following addresses in sequence.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[0]

4 miss

00	Mem[0]
00	Mem[4]

8 miss

00	Mem[0]
00	Mem[4]
00	Mem[8]

12

16

12

16

60

Direct-Mapped Cache Example

Start with an empty cache (all blocks are not valid/blank).

Access the words at the following addresses in sequence.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[0]

4 miss

00	Mem[0]
00	Mem[4]

8 miss

00	Mem[0]
00	Mem[4]
00	Mem[8]

12

00	Mem[0]
00	Mem[4]
00	Mem[8]

16

12

16

60

Direct-Mapped Cache Example

Start with an empty cache (all blocks are not valid/blank).

Access the words at the following addresses in sequence.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[0]

4 miss

00	Mem[0]
00	Mem[4]

8 miss

00	Mem[0]
00	Mem[4]
00	Mem[8]

12 miss

00	Mem[0]
00	Mem[4]
00	Mem[8]

16

12

16

60

Direct-Mapped Cache Example

Start with an empty cache (all blocks are not valid/blank).

Access the words at the following addresses in sequence.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[0]

4 miss

00	Mem[0]
00	Mem[4]

8 miss

00	Mem[0]
00	Mem[4]
00	Mem[8]

12 miss

00	Mem[0]
00	Mem[4]
00	Mem[8]
00	Mem[12]

16

12

16

60

Direct-Mapped Cache Example

Start with an empty cache (all blocks are not valid/blank).

Access the words at the following addresses in sequence.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[0]

4 miss

00	Mem[0]
00	Mem[4]

8 miss

00	Mem[0]
00	Mem[4]
00	Mem[8]

12 miss

00	Mem[0]
00	Mem[4]
00	Mem[8]
00	Mem[12]

16

00	Mem[0]
00	Mem[4]
00	Mem[8]
00	Mem[12]

12

16

60

Direct-Mapped Cache Example

Start with an empty cache (all blocks are not valid/blank).

Access the words at the following addresses in sequence.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[0]

4 miss

00	Mem[0]
00	Mem[4]

8 miss

00	Mem[0]
00	Mem[4]
00	Mem[8]

12 miss

00	Mem[0]
00	Mem[4]
00	Mem[8]
00	Mem[12]

16 miss

00	Mem[0]
00	Mem[4]
00	Mem[8]
00	Mem[12]

12

16

60

Direct-Mapped Cache Example

Start with an empty cache (all blocks are not valid/blank).

Access the words at the following addresses in sequence.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[0]

4 miss

00	Mem[0]
00	Mem[4]

8 miss

00	Mem[0]
00	Mem[4]
00	Mem[8]

12 miss

00	Mem[0]
00	Mem[4]
00	Mem[8]
00	Mem[12]

16 miss

~~00~~ 01	~~Mem[0]~~ 16
00	Mem[4]
00	Mem[8]
00	Mem[12]

12

16

60

Direct-Mapped Cache Example

Start with an empty cache (all blocks are not valid/blank).

Access the words at the following addresses in sequence.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[0]

4 miss

00	Mem[0]
00	Mem[4]

8 miss

00	Mem[0]
00	Mem[4]
00	Mem[8]

12 miss

00	Mem[0]
00	Mem[4]
00	Mem[8]
00	Mem[12]

16 miss

01 ~~00~~	~~Mem[0]~~ 16
00	Mem[4]
00	Mem[8]
00	Mem[12]

12

01	Mem[16]
00	Mem[4]
00	Mem[8]
00	Mem[12]

16

60

Direct-Mapped Cache Example

Start with an empty cache (all blocks are not valid/blank).

Access the words at the following addresses in sequence.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[0]

4 miss

00	Mem[0]
00	Mem[4]

8 miss

00	Mem[0]
00	Mem[4]
00	Mem[8]

12 miss

00	Mem[0]
00	Mem[4]
00	Mem[8]
00	Mem[12]

16 miss

01 ~~00~~	~~Mem[0]~~ 16
00	Mem[4]
00	Mem[8]
00	Mem[12]

12 hit

01	Mem[16]
00	Mem[4]
00	Mem[8]
00	Mem[12]

16

60

Direct-Mapped Cache Example

Start with an empty cache (all blocks are not valid/blank).

Access the words at the following addresses in sequence.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[0]

4 miss

00	Mem[0]
00	Mem[4]

8 miss

00	Mem[0]
00	Mem[4]
00	Mem[8]

12 miss

00	Mem[0]
00	Mem[4]
00	Mem[8]
00	Mem[12]

16 miss

01 ~~00~~	~~Mem[0]~~ 16
00	Mem[4]
00	Mem[8]
00	Mem[12]

12 hit

01	Mem[16]
00	Mem[4]
00	Mem[8]
00	Mem[12]

16

01	Mem[16]
00	Mem[4]
00	Mem[8]
00	Mem[12]

60

Direct-Mapped Cache Example

Start with an empty cache (all blocks are not valid/blank).

Access the words at the following addresses in sequence.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[0]

4 miss

00	Mem[0]
00	Mem[4]

8 miss

00	Mem[0]
00	Mem[4]
00	Mem[8]

12 miss

00	Mem[0]
00	Mem[4]
00	Mem[8]
00	Mem[12]

16 miss

01 ~~00~~	~~Mem[0]~~ 16
00	Mem[4]
00	Mem[8]
00	Mem[12]

12 hit

01	Mem[16]
00	Mem[4]
00	Mem[8]
00	Mem[12]

16 hit

01	Mem[16]
00	Mem[4]
00	Mem[8]
00	Mem[12]

60

Direct-Mapped Cache Example

Start with an empty cache (all blocks are not valid/blank).

Access the words at the following addresses in sequence.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[0]

4 miss

00	Mem[0]
00	Mem[4]

8 miss

00	Mem[0]
00	Mem[4]
00	Mem[8]

12 miss

00	Mem[0]
00	Mem[4]
00	Mem[8]
00	Mem[12]

16 miss

01 ~~00~~	~~Mem[0]~~ 16
00	Mem[4]
00	Mem[8]
00	Mem[12]

12 hit

01	Mem[16]
00	Mem[4]
00	Mem[8]
00	Mem[12]

16 hit

01	Mem[16]
00	Mem[4]
00	Mem[8]
00	Mem[12]

60

01	Mem[16]
00	Mem[4]
00	Mem[8]
00	Mem[12]

Direct-Mapped Cache Example

Start with an empty cache (all blocks are not valid/blank).

Access the words at the following addresses in sequence.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[0]

4 miss

00	Mem[0]
00	Mem[4]

8 miss

00	Mem[0]
00	Mem[4]
00	Mem[8]

12 miss

00	Mem[0]
00	Mem[4]
00	Mem[8]
00	Mem[12]

16 miss

01 ~~00~~	~~Mem[0]~~ 16
00	Mem[4]
00	Mem[8]
00	Mem[12]

12 hit

01	Mem[16]
00	Mem[4]
00	Mem[8]
00	Mem[12]

16 hit

01	Mem[16]
00	Mem[4]
00	Mem[8]
00	Mem[12]

60 miss

01	Mem[16]
00	Mem[4]
00	Mem[8]
00	Mem[12]

Direct-Mapped Cache Example

Start with an empty cache (all blocks are not valid/blank).

Access the words at the following addresses in sequence.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[0]

4 miss

00	Mem[0]
00	Mem[4]

8 miss

00	Mem[0]
00	Mem[4]
00	Mem[8]

12 miss

00	Mem[0]
00	Mem[4]
00	Mem[8]
00	Mem[12]

16 miss

01 ~~00~~	~~Mem[0]~~ 16
00	Mem[4]
00	Mem[8]
00	Mem[12]

12 hit

01	Mem[16]
00	Mem[4]
00	Mem[8]
00	Mem[12]

16 hit

01	Mem[16]
00	Mem[4]
00	Mem[8]
00	Mem[12]

60 miss

01	Mem[16]
00	Mem[4]
00	Mem[8]
11 ~~00~~	~~Mem[12]~~ 60

Discussion

- 6 misses out of 8 requests

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100
M	M	M	M	M	H	H	M

- What is the miss rate?

- What kind of locality are we taking advantage of?

  A: Temporal.

  B: Spatial.

Design parameters

- Let us keep the cache size the same, but increase block size

Previously,

Cache size: 4 words

Block size: 1 word

Number of blocks: 4

Now,

Cache size: 4 words

Block size: 2 words

Number of blocks: ?

Number of bits in cache index: ?

Another cache of the same size

Let cache block hold two words (8 bytes). Again, start with an empty cache.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0

4

8

12

16

12

16

60

Another cache of the same size

Let cache block hold two words (8 bytes). Again, start with an empty cache.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

4

8

12

16

12

16

60

Another cache of the same size

Let cache block hold two words (8 bytes). Again, start with an empty cache.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[4]	Mem[0]

4

8

12

16

12

16

60

Another cache of the same size

Let cache block hold two words (8 bytes). Again, start with an empty cache.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[4]	Mem[0]

4

00	Mem[4]	Mem[0]

8

12

16

12

16

60

Another cache of the same size

Let cache block hold two words (8 bytes). Again, start with an empty cache.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[4]	Mem[0]

4 hit

00	Mem[4]	Mem[0]

8

12

16

12

16

60

Another cache of the same size

Let cache block hold two words (8 bytes). Again, start with an empty cache.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[4]	Mem[0]

4 hit

00	Mem[4]	Mem[0]

8

00	Mem[4]	Mem[0]

12

16

12

16

60

Another cache of the same size

Let cache block hold two words (8 bytes). Again, start with an empty cache.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[4]	Mem[0]

4 hit

00	Mem[4]	Mem[0]

8 miss

00	Mem[4]	Mem[0]

12

16

12

16

60

Another cache of the same size

Let cache block hold two words (8 bytes). Again, start with an empty cache.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[4]	Mem[0]

4 hit

00	Mem[4]	Mem[0]

8 miss

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

12

16

12

16

60

Another cache of the same size

Let cache block hold two words (8 bytes). Again, start with an empty cache.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[4]	Mem[0]

4 hit

00	Mem[4]	Mem[0]

8 miss

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

12

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

16

12

16

60

Another cache of the same size

Let cache block hold two words (8 bytes). Again, start with an empty cache.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[4]	Mem[0]

4 hit

00	Mem[4]	Mem[0]

8 miss

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

12 hit

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

16

12

16

60

Another cache of the same size

Let cache block hold two words (8 bytes). Again, start with an empty cache.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[4]	Mem[0]

4 hit

00	Mem[4]	Mem[0]

8 miss

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

12 hit

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

16

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

12

16

60

Another cache of the same size

Let cache block hold two words (8 bytes). Again, start with an empty cache.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[4]	Mem[0]

4 hit

00	Mem[4]	Mem[0]

8 miss

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

12 hit

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

16 miss

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

12

16

60

Another cache of the same size

Let cache block hold two words (8 bytes). Again, start with an empty cache.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[4]	Mem[0]

4 hit

00	Mem[4]	Mem[0]

8 miss

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

12 hit

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

16 miss

~~00~~ 01	Mem[~~4~~] 20	Mem[~~0~~] 16
00	Mem[12]	Mem[8]

12

16

60

Another cache of the same size

Let cache block hold two words (8 bytes). Again, start with an empty cache.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[4]	Mem[0]

4 hit

00	Mem[4]	Mem[0]

8 miss

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

12 hit

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

16 miss

01 ~~00~~	20 ~~Mem[4]~~	16 ~~Mem[0]~~
00	Mem[12]	Mem[8]

12

01	Mem[20]	Mem[16]
00	Mem[12]	Mem[8]

16

60

Another cache of the same size

Let cache block hold two words (8 bytes). Again, start with an empty cache.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[4]	Mem[0]

4 hit

00	Mem[4]	Mem[0]

8 miss

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

12 hit

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

16 miss

01 ~~00~~	20 ~~Mem[4]~~	16 ~~Mem[0]~~
00	Mem[12]	Mem[8]

12 hit

01	Mem[20]	Mem[16]
00	Mem[12]	Mem[8]

16

60

Another cache of the same size

Let cache block hold two words (8 bytes). Again, start with an empty cache.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[4]	Mem[0]

4 hit

00	Mem[4]	Mem[0]

8 miss

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

12 hit

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

16 miss

01 ~~00~~	20 ~~Mem[4]~~	16 ~~Mem[0]~~
00	Mem[12]	Mem[8]

12 hit

01	Mem[20]	Mem[16]
00	Mem[12]	Mem[8]

16

01	Mem[20]	Mem[16]
00	Mem[12]	Mem[8]

60

Another cache of the same size

Let cache block hold two words (8 bytes). Again, start with an empty cache.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[4]	Mem[0]

4 hit

00	Mem[4]	Mem[0]

8 miss

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

12 hit

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

16 miss

01 ~~00~~	20 Mem[~~4~~]	16 Mem[~~0~~]
00	Mem[12]	Mem[8]

12 hit

01	Mem[20]	Mem[16]
00	Mem[12]	Mem[8]

16 hit

01	Mem[20]	Mem[16]
00	Mem[12]	Mem[8]

60

Another cache of the same size

Let cache block hold two words (8 bytes). Again, start with an empty cache.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[4]	Mem[0]

4 hit

00	Mem[4]	Mem[0]

8 miss

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

12 hit

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

16 miss

01 ~~00~~	20 ~~Mem[4]~~	16 ~~Mem[0]~~
00	Mem[12]	Mem[8]

12 hit

01	Mem[20]	Mem[16]
00	Mem[12]	Mem[8]

16 hit

01	Mem[20]	Mem[16]
00	Mem[12]	Mem[8]

60

01	Mem[20]	Mem[16]
00	Mem[12]	Mem[8]

Another cache of the same size

Let cache block hold two words (8 bytes). Again, start with an empty cache.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[4]	Mem[0]

4 hit

00	Mem[4]	Mem[0]

8 miss

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

12 hit

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

16 miss

01 ~~00~~	20 ~~Mem[4]~~	16 ~~Mem[0]~~
00	Mem[12]	Mem[8]

12 hit

01	Mem[20]	Mem[16]
00	Mem[12]	Mem[8]

16 hit

01	Mem[20]	Mem[16]
00	Mem[12]	Mem[8]

60 miss

01	Mem[20]	Mem[16]
00	Mem[12]	Mem[8]

Another cache of the same size

Let cache block hold two words (8 bytes). Again, start with an empty cache.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[4]	Mem[0]

4 hit

00	Mem[4]	Mem[0]

8 miss

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

12 hit

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

16 miss

~~00~~ 01	~~Mem[4]~~ 20	~~Mem[0]~~ 16
00	Mem[12]	Mem[8]

12 hit

01	Mem[20]	Mem[16]
00	Mem[12]	Mem[8]

16 hit

01	Mem[20]	Mem[16]
00	Mem[12]	Mem[8]

60 miss

01	Mem[20]	Mem[16]
~~00~~ 11	~~Mem[12]~~ 60	~~Mem[8]~~ 56

Another cache of the same size

Let cache block hold two words (8 bytes). Again, start with an empty cache.

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100

0 miss

00	Mem[4]	Mem[0]

4 hit

00	Mem[4]	Mem[0]

8 miss

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

12 hit

00	Mem[4]	Mem[0]
00	Mem[12]	Mem[8]

16 miss

01 ~~00~~	20 Mem[~~4~~]	16 Mem[~~0~~]
00	Mem[12]	Mem[8]

12 hit

01	Mem[20]	Mem[16]
00	Mem[12]	Mem[8]

16 hit

01	Mem[20]	Mem[16]
00	Mem[12]	Mem[8]

60 miss

01	Mem[20]	Mem[16]
~~00~~ 11	Mem[~~12~~] 60	Mem[~~8~~] 56

8 requests, 4 misses.

Discussion

- 6 misses out of 8 requests

0	4	8	12	16	12	16	60
000000	000100	001000	001100	010000	001100	010000	111100
M	H	M	H	M	H	H	M

- What is the miss rate?

- Why do we have fewer missies?

What kind of locality are we taking advantage of?

A: Temporal.

B: Spatial.

Building a cache

- Cache size = 4KiB
- Block size = one word

Cache size

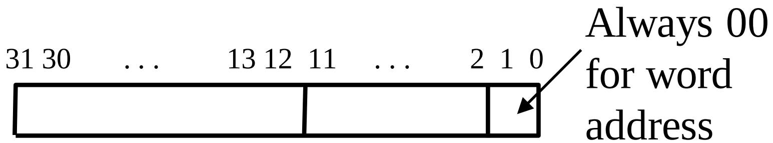
Amount of data that can be stored in cache

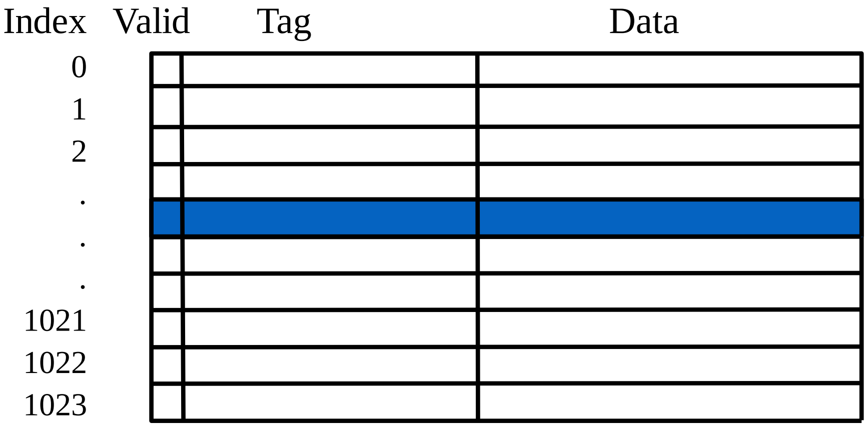
- How many blocks?
- Size of each field in cache ?
  - Block offset, cache index, and tag
- How would you select a block?

Suppose we only want to support LW/SW.

Direct-Mapped Cache Diagram – Single-word Block

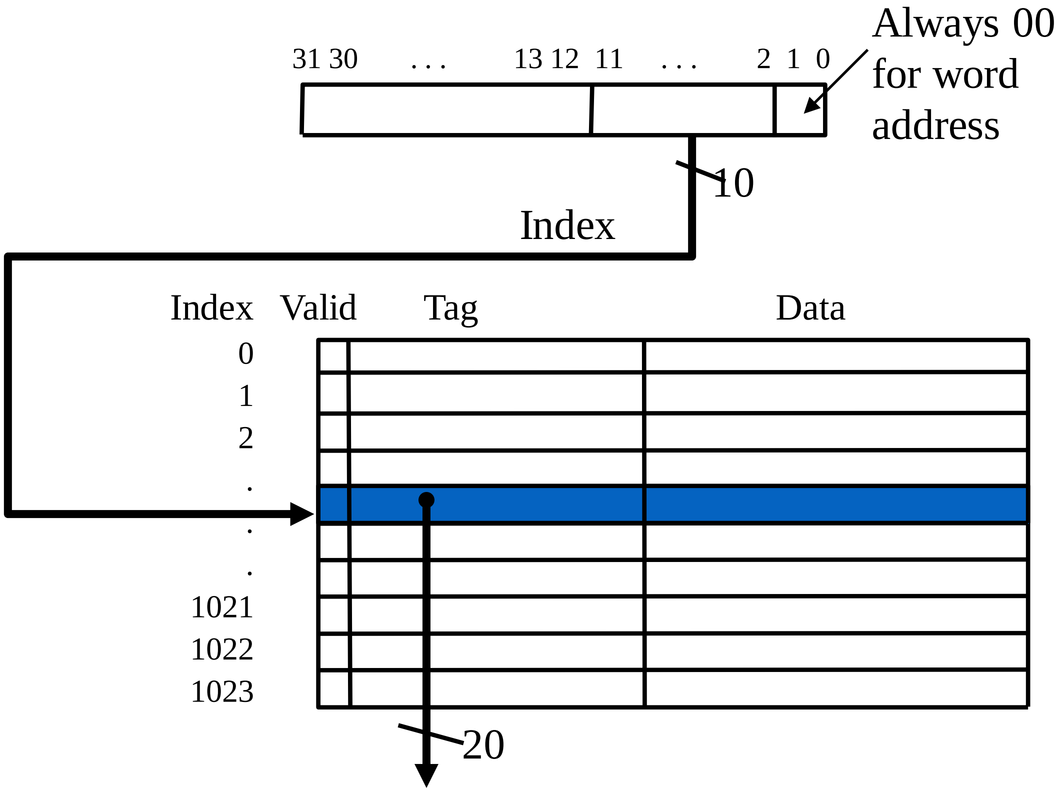
Cache size = 4KiB, block size = one word (4 bytes)





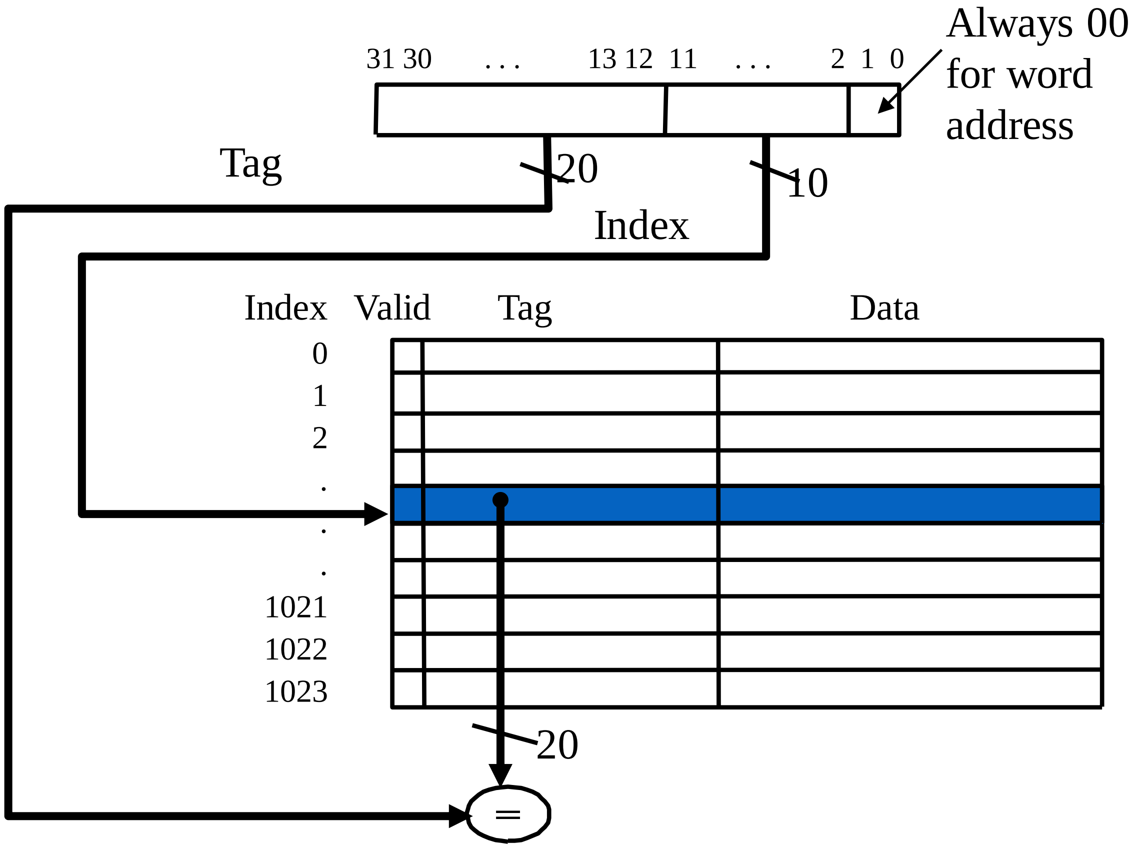
Direct-Mapped Cache Diagram – Single-word Block

Cache size = 4KiB, block size = one word (4 bytes)



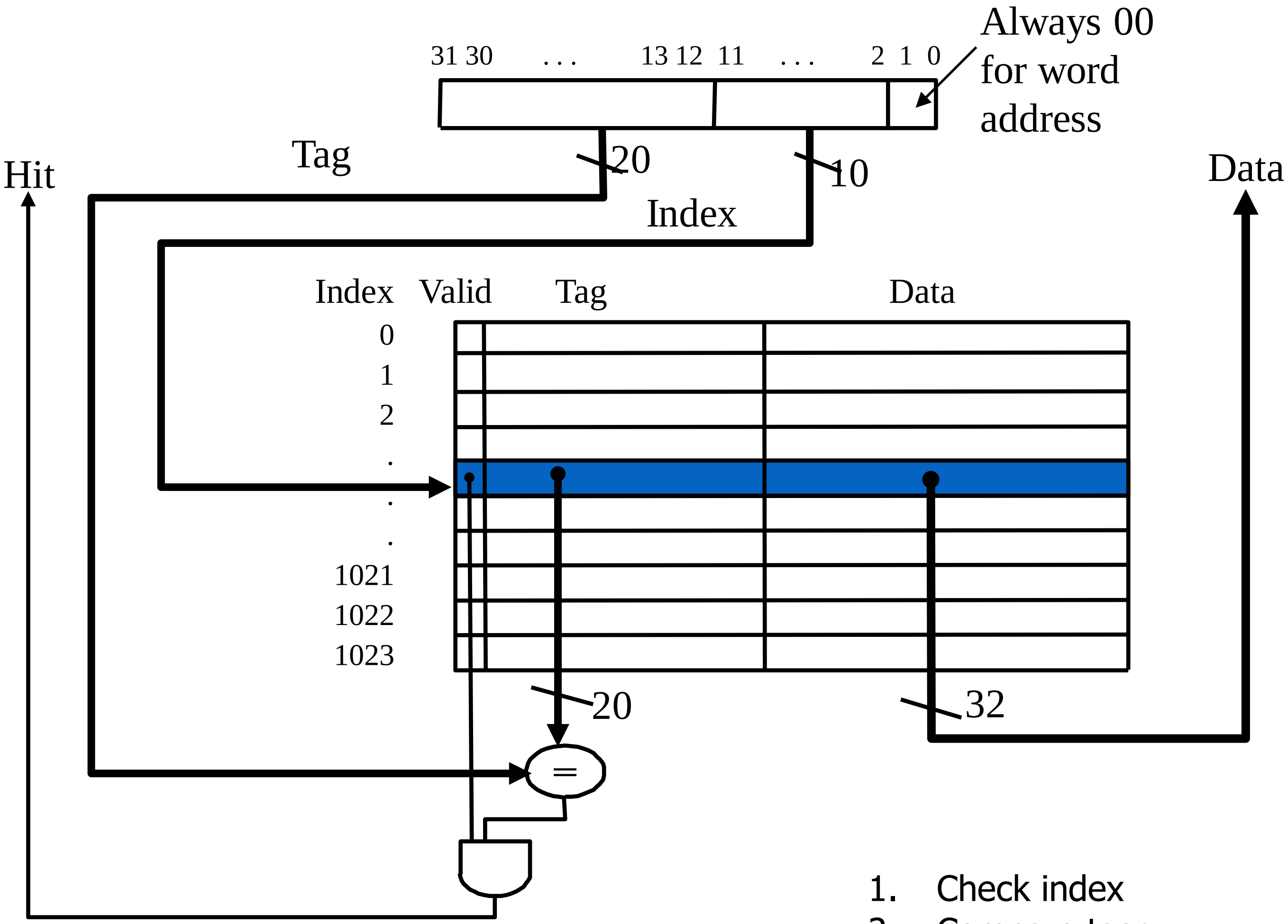
Direct-Mapped Cache Diagram – Single-word Block

Cache size = 4KiB, block size = one word (4 bytes)



Direct-Mapped Cache Diagram – Single-word Block

Cache size = 4KiB, block size = one word (4 bytes)



1. Check index
2. Compare tags
3. Check valid bit

Discussion

Assume we only access words.

What kind of locality can we take advantage of with the cache on the previous slide?

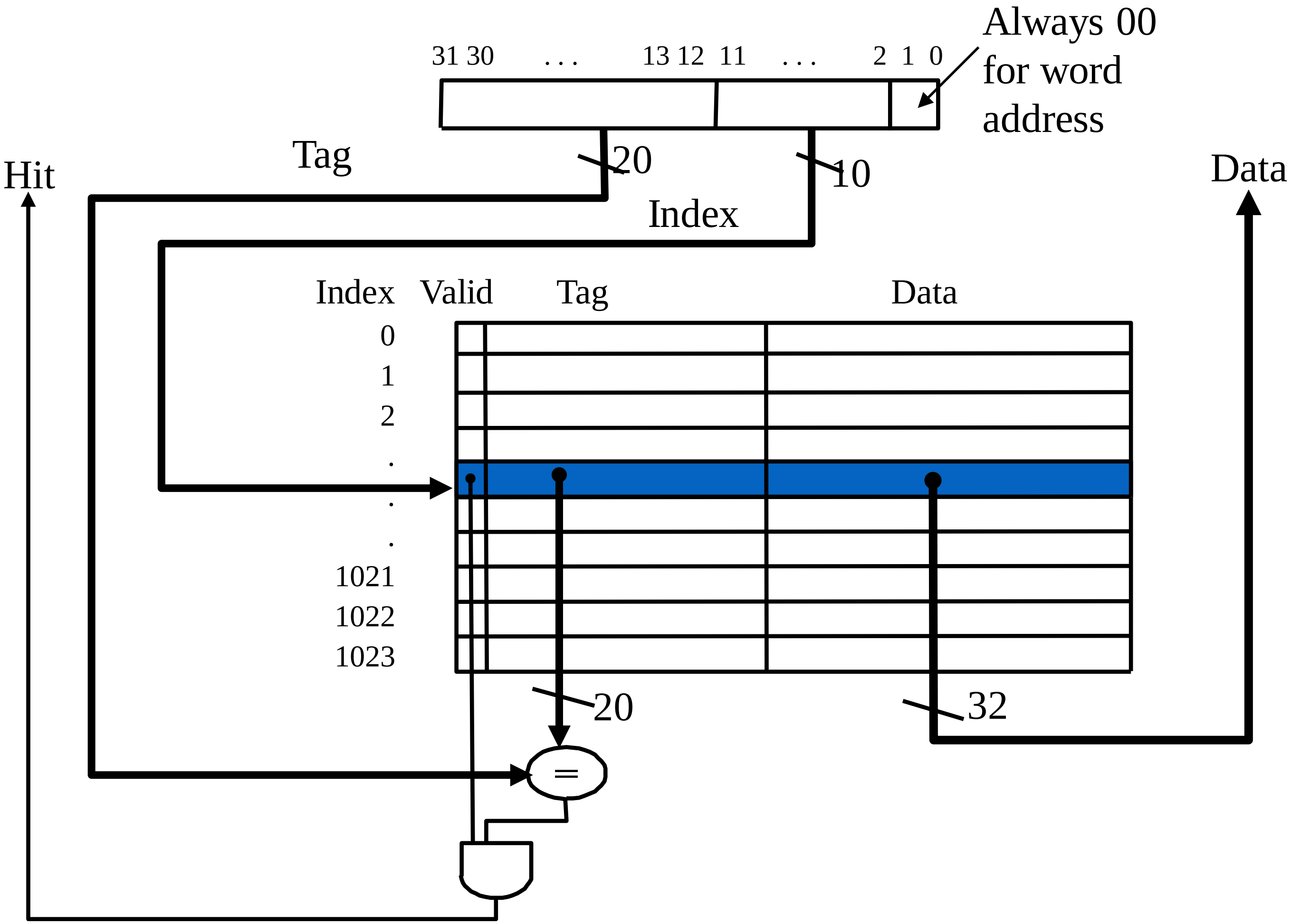
A: Temporal

B: Spatial

C: Both

Questions

- What is the total number of bits in the cache?



Design parameters

- Let us keep the cache size the same, but increase block size

Cache size: 4 KiB

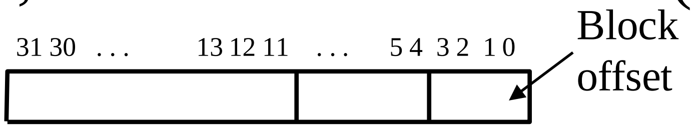
Previously, block size: 4 bytes (one word)

Now, Block size: 16 bytes (four words)

- How many blocks?
- Size of each field in cache ?
  - Block offset, cache index, and tag
- How would you select a block? And then a word?

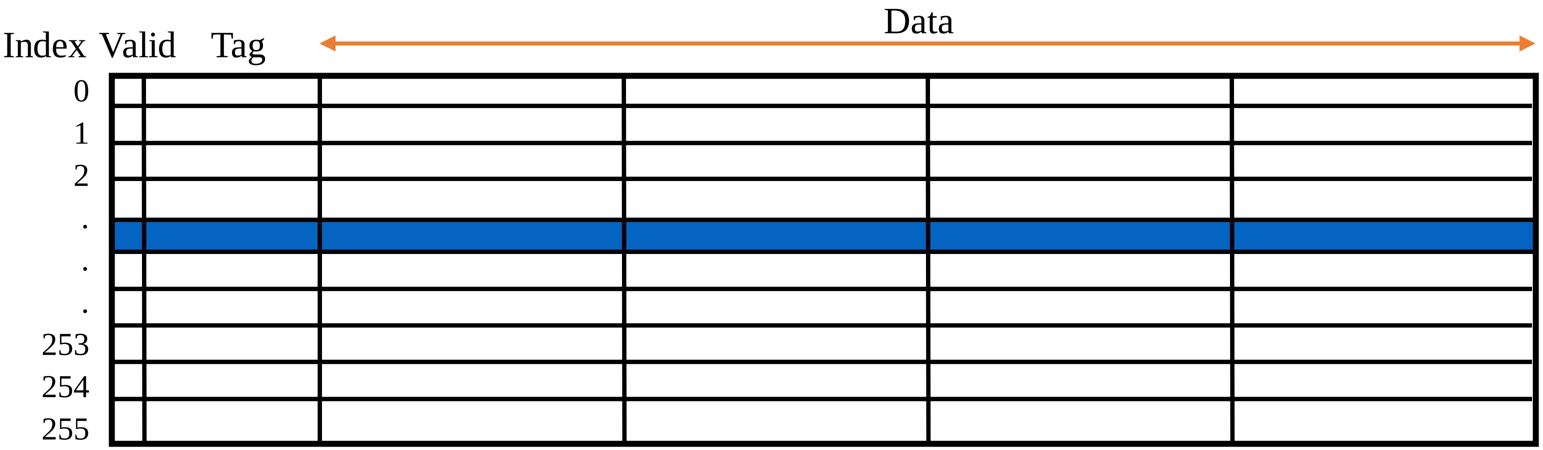
Direct-Mapped Cache Diagram – Multiword Block

Cache size = 4KiB, block size = four words (16 bytes)



31 30 . . . 13 12 11 . . . 5 4 3 2 1 0

Block offset



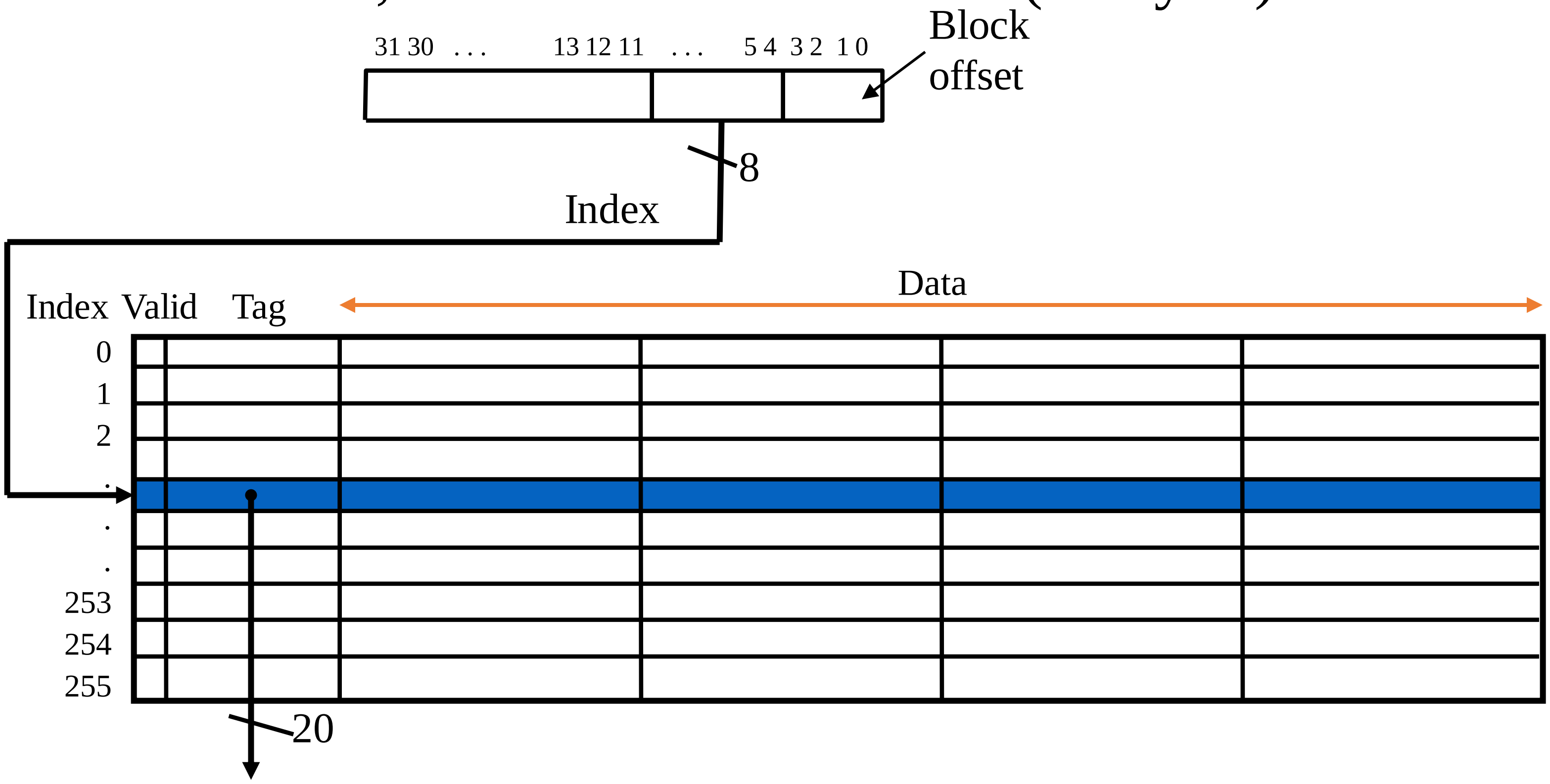
Index Valid Tag

Data

0
1
2
.
.
.
253
254
255

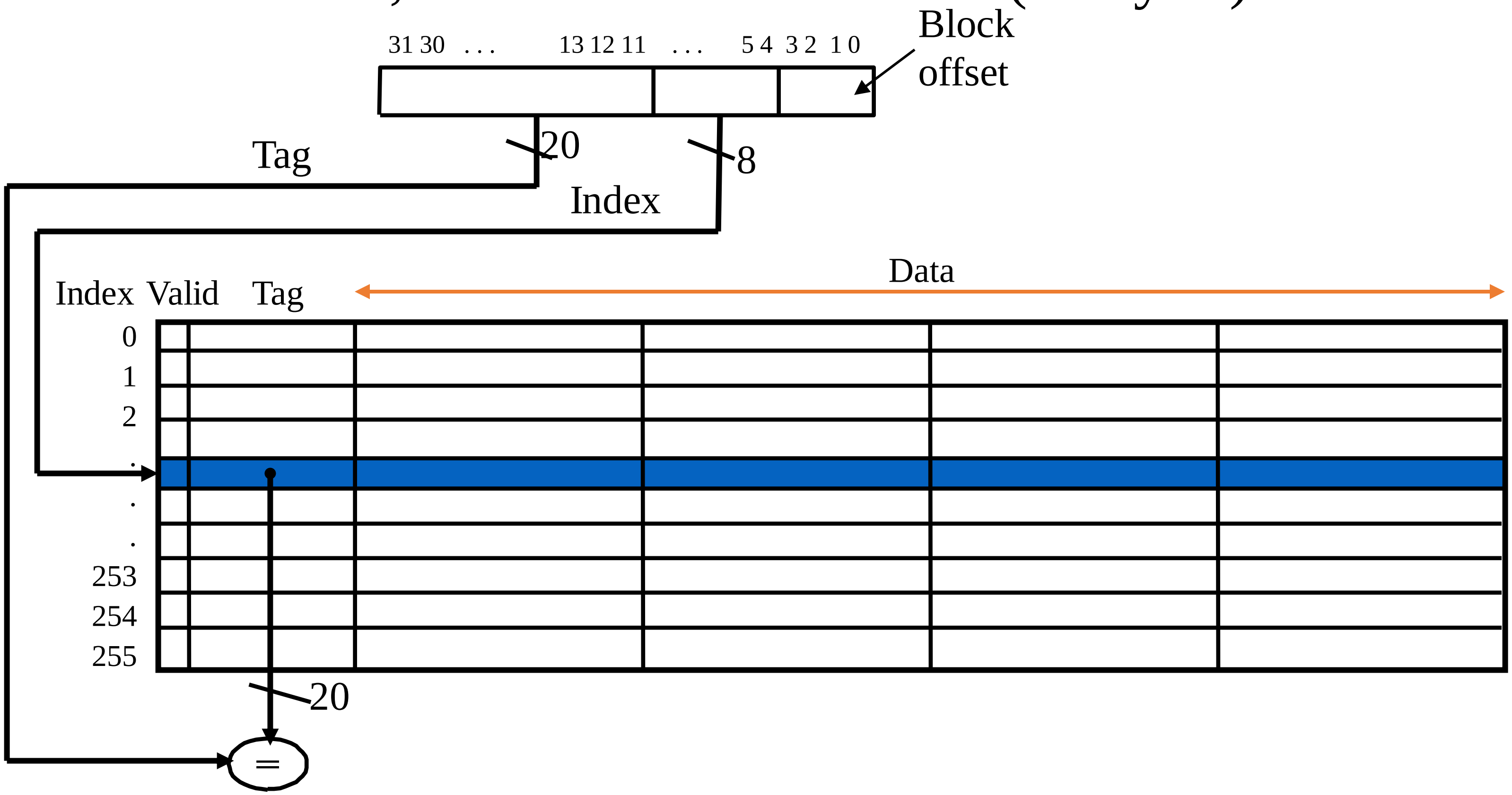
Direct-Mapped Cache Diagram – Multiword Block

Cache size = 4KiB, block size = four words (16 bytes)



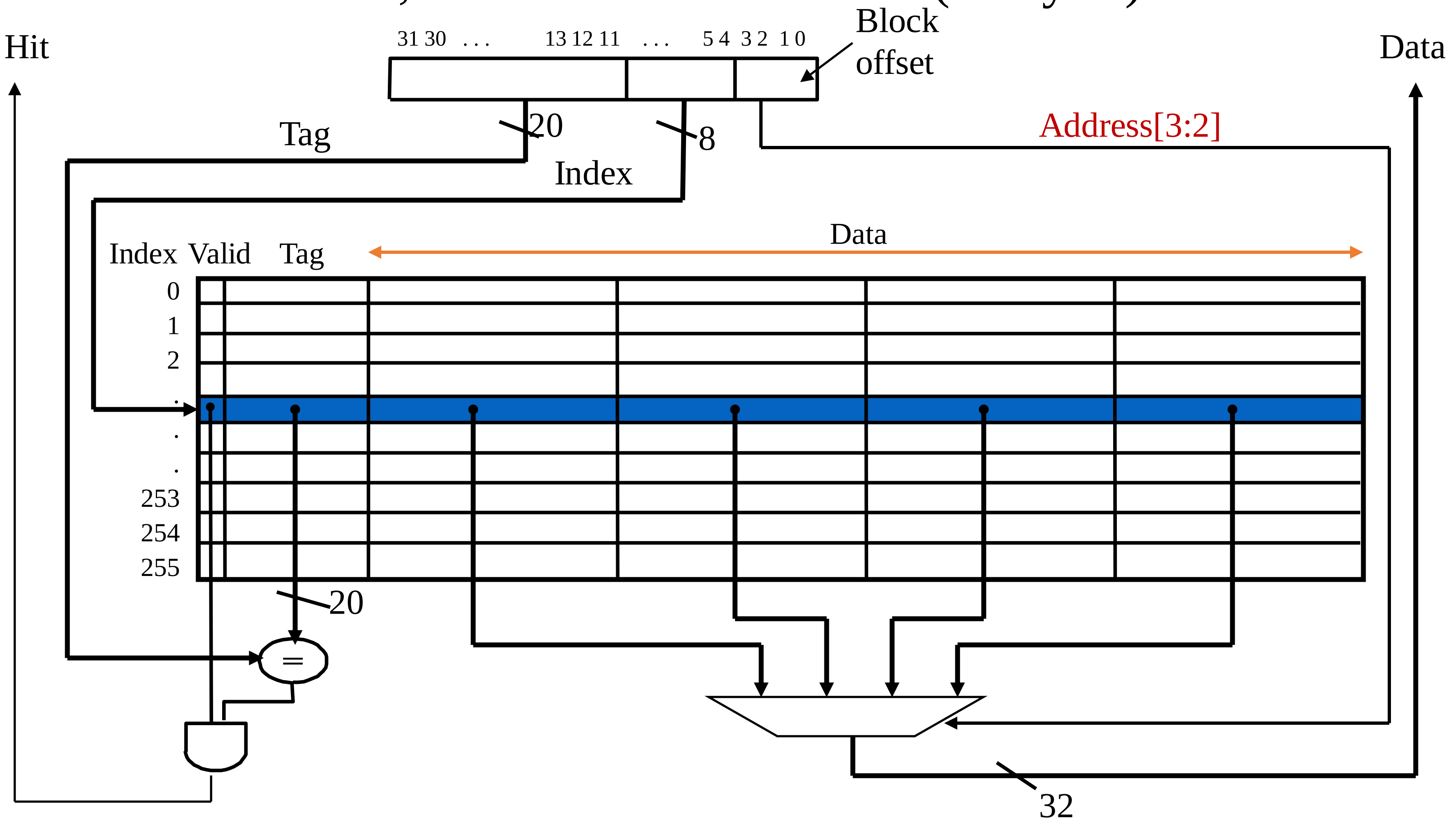
Direct-Mapped Cache Diagram – Multiword Block

Cache size = 4KiB, block size = four words (16 bytes)



Direct-Mapped Cache Diagram – Multiword Block

Cache size = 4KiB, block size = four words (16 bytes)



Discussion

Assume we only access words.

What kind of locality can we take advantage of with the cache on the previous slide?

A: Temporal

B: Spatial

C: Both

Questions

- What is the total number of bits in the cache?

- What percent of the bits are used for tags and status?

Number of Bits in a Direct Mapped Cache

Cache needs to store both data and tags

- Although we normally just use data size, e.g., a 4KB cache.

32-bit byte address

- offset has m bits for a block size of 2^m bytes, to identify any byte within the block. The word address has lower 2 bits set to 0
- cache index has n bits for a direct mapped cache with 2^n blocks
- the rest of the bits are the tag

- The total number of bits in a direct-mapped cache is then

$$2^n \times (\text{block size} + \text{tag field size} + \text{number of status bits})$$

Example: Number of Bits in a Direct Mapped Cache

Assume an address has 32 bits and there is only one status bit for each block. How many bits in total are in a direct mapped cache with 16KiB of data and 16-byte blocks?

Solutions:

Example: Number of Bits in a Direct Mapped Cache

Assume an address has 32 bits and there is only one status bit for each block. How many bits in total are in a direct mapped cache with 16KiB of data and 16-byte blocks?

Solutions:

Block size: 16 bytes

Example: Number of Bits in a Direct Mapped Cache

Assume an address has 32 bits and there is only one status bit for each block. How many bits in total are in a direct mapped cache with 16KiB of data and 16-byte blocks?

Solutions:

Block size: 16 bytes

→ Number of bits for selecting bytes in a block: 4 bits

Example: Number of Bits in a Direct Mapped Cache

Assume an address has 32 bits and there is only one status bit for each block. How many bits in total are in a direct mapped cache with 16KiB of data and 16-byte blocks?

Solutions:

Block size: 16 bytes

→ Number of bits for selecting bytes in a block: 4 bits

Cache size: 16 KiB

Example: Number of Bits in a Direct Mapped Cache

Assume an address has 32 bits and there is only one status bit for each block. How many bits in total are in a direct mapped cache with 16KiB of data and 16-byte blocks?

Solutions:

Block size: 16 bytes

→ Number of bits for selecting bytes in a block: 4 bits

Cache size: 16 KiB

→ Number of blocks = 16 K / 16 = 1K (1024)

Example: Number of Bits in a Direct Mapped Cache

Assume an address has 32 bits and there is only one status bit for each block. How many bits in total are in a direct mapped cache with 16KiB of data and 16-byte blocks?

Solutions:

Block size: 16 bytes

→ Number of bits for selecting bytes in a block: 4 bits

Cache size: 16 KiB

→ Number of blocks = 16 K / 16 = 1K (1024)

→ Number of bits in index: 10 bits (for 1K blocks)

Example: Number of Bits in a Direct Mapped Cache

Assume an address has 32 bits and there is only one status bit for each block. How many bits in total are in a direct mapped cache with 16KiB of data and 16-byte blocks?

Solutions:

Block size: 16 bytes

→ Number of bits for selecting bytes in a block: 4 bits

Cache size: 16 KiB

→ Number of blocks = 16 K / 16 = 1K (1024)

→ Number of bits in index: 10 bits (for 1K blocks)

Number of bits in tag: 32 – 10 – 4 = 18 (the rest of address bits)

Example: Number of Bits in a Direct Mapped Cache

Assume an address has 32 bits and there is only one status bit for each block. How many bits in total are in a direct mapped cache with 16KiB of data and 16-byte blocks?

Solutions:

Block size: 16 bytes

→ Number of bits for selecting bytes in a block: 4 bits

Cache size: 16 KiB

→ Number of blocks = 16 K / 16 = 1K (1024)

→ Number of bits in index: 10 bits (for 1K blocks)

Number of bits in tag: 32 – 10 – 4 = 18 (the rest of address bits)

Status bits: 1 (only the valid bit)

Example: Number of Bits in a Direct Mapped Cache

Assume an address has 32 bits and there is only one status bit for each block. How many bits in total are in a direct mapped cache with 16KiB of data and 16-byte blocks?

Solutions:

Block size: 16 bytes

→ Number of bits for selecting bytes in a block: 4 bits

Cache size: 16 KiB

→ Number of blocks = 16 K / 16 = 1K (1024)

→ Number of bits in index: 10 bits (for 1K blocks)

Number of bits in tag: 32 – 10 – 4 = 18 (the rest of address bits)

Status bits: 1 (only the valid bit)

The number of bits needed for a block: 16 × 8 + 18 + 1 = 147 bits

Example: Number of Bits in a Direct Mapped Cache

Assume an address has 32 bits and there is only one status bit for each block. How many bits in total are in a direct mapped cache with 16KiB of data and 16-byte blocks?

Solutions:

Block size: 16 bytes

→ Number of bits for selecting bytes in a block: 4 bits

Cache size: 16 KiB

→ Number of blocks = 16 K / 16 = 1K (1024)

→ Number of bits in index: 10 bits (for 1K blocks)

Number of bits in tag: 32 – 10 – 4 = 18 (the rest of address bits)

Status bits: 1 (only the valid bit)

The number of bits needed for a block: 16 × 8 + 18 + 1 = 147 bits

Total number of bits for 1K blocks:

Example: Number of Bits in a Direct Mapped Cache

Assume an address has 32 bits and there is only one status bit for each block. How many bits in total are in a direct mapped cache with 16KiB of data and 16-byte blocks?

Solutions:

Block size: 16 bytes

→ Number of bits for selecting bytes in a block: 4 bits

Cache size: 16 KiB

→ Number of blocks = 16 K / 16 = 1K (1024)

→ Number of bits in index: 10 bits (for 1K blocks)

Number of bits in tag: 32 – 10 – 4 = 18 (the rest of address bits)

Status bits: 1 (only the valid bit)

The number of bits needed for a block: 16 × 8 + 18 + 1 = 147 bits

Total number of bits for 1K blocks:

147 × 1K = 147 × 1024 = 150528

Question

The number of bits needed for a block: 16 × 8 + 18 + 1 = 147 bits

Common mistake:
Forget to multiply 8

Total number of bits for 1K blocks:

$$147 \times 1K = 147 \times 1024 = 150528$$

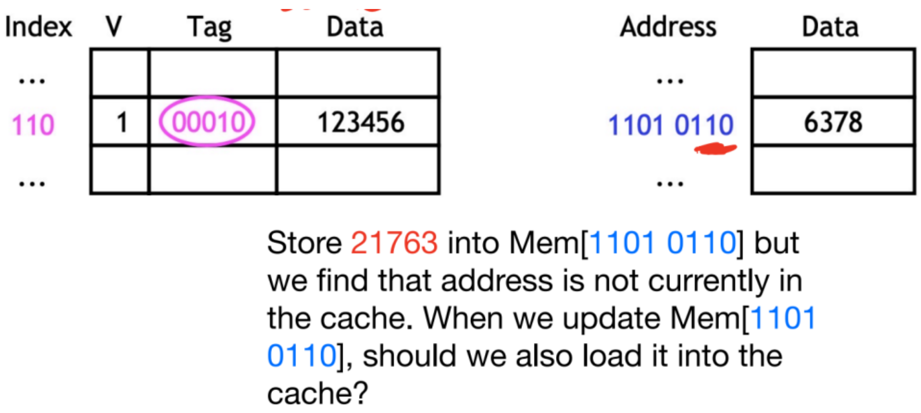
What percent of the bits are used for tags and status?

Round to the nearest whole percent. For example, 16 for 15.5%.

Cache operations

- Read (for both I$ and D$)
  - Read hit
    - This is what we want!
  - Read miss
    - Stall the pipeline

- Write (D$ only)
  - Hit
    - Write-through or write-back
  - Handling Miss
    - Write allocate or No write allocate



Index	V	Tag	Data
...			
110	1	00010	123456
...			

Address	Data
...	
1101 0110	6378
...	

Store 21763 into Mem[1101 0110] but we find that address is not currently in the cache. When we update Mem[1101 0110], should we also load it into the cache?

credit to: Howard Huang, Cache writes and examples

Handling Read Misses

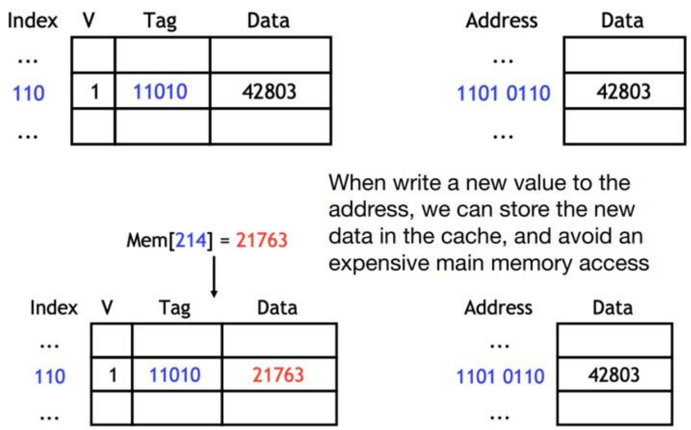
- Read misses (I$ and D$)
  - Stall the pipeline
  - Fetch the block from the next level in the memory hierarchy into cache
  - Send the requested word to the processor
  - Let the pipeline resume

Textbook has more detailed explanations on handling instruction cache miss

Write Hits

- The data to be updated is found in cache
- The problem? It has two copies
  - One in cache and one in memory

The address we want to write to is loaded in the cache (assume direct-mapped cache.)



Index	V	Tag	Data
…			
110	1	11010	42803
…			

Address	Data
…	
1101 0110	42803
…	

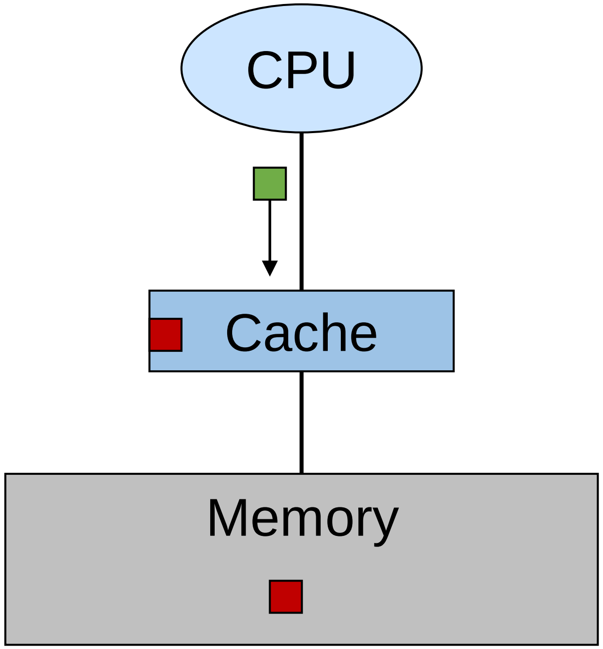
When write a new value to the address, we can store the new data in the cache, and avoid an expensive main memory access

Mem[214] = 21763

Index	V	Tag	Data
…			
110	1	11010	21763
…			

Address	Data
…	
1101 0110	42803
…	

credit to: Howard Huang, *Cache writes and examples*

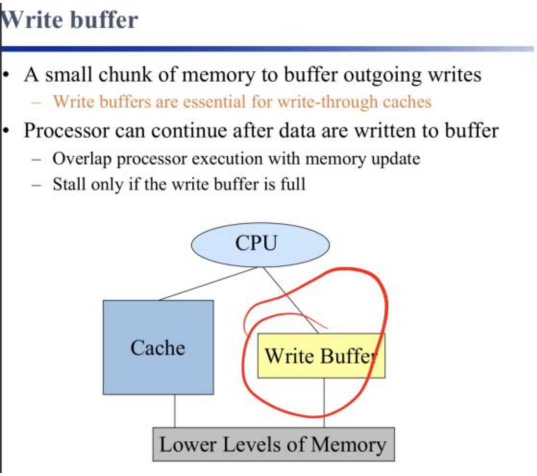


Now the cache and memory contain different, inconsistent data!

Handling Write Hits: Write-through

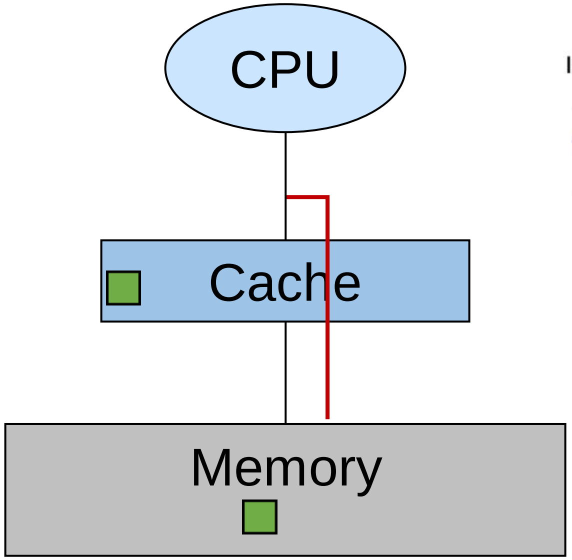
Write-through: update both cache and memory

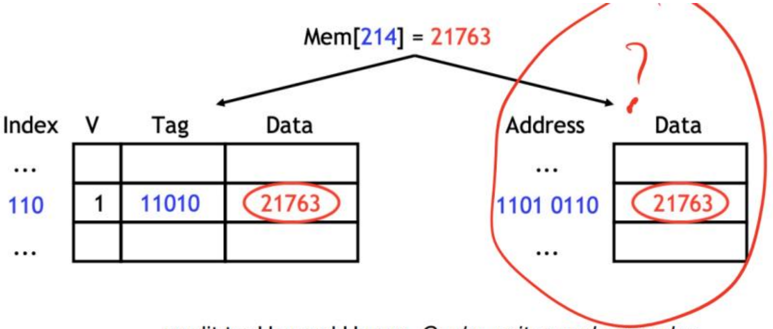
- Pros: The cache and memory are consistent
  - Always write the data into both the cache block and the next level in the memory hierarchy (write-through)
- Cons: Writes run at the speed of the memory – very slow!
  - Think about initialization of an array in your code



Write buffer

- A small chunk of memory to buffer outgoing writes
  - Write buffers are essential for write-through caches
- Processor can continue after data are written to buffer
  - Overlap processor execution with memory update
  - Stall only if the write buffer is full





Index	V	Tag	Data
...			
110	1	11010	21763
...			

Address	Data
...	
1101 0110	21763
...	

credit to: Howard Huang, *Cache writes and examples*

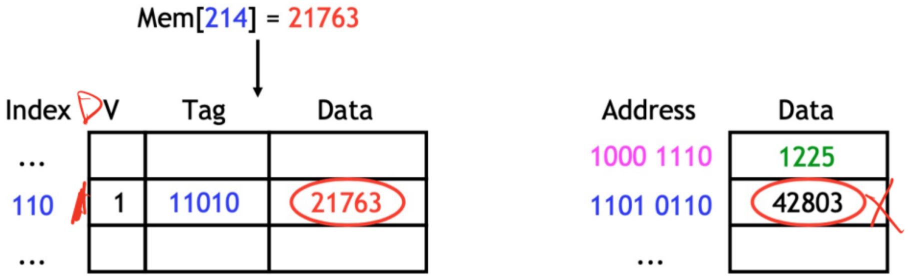
Handling Write Hits: Write Back

Write-Back: update cache only.

Update memory later, when that cache block is “evicted”

Cache and memory are inconsistent.

They have different data

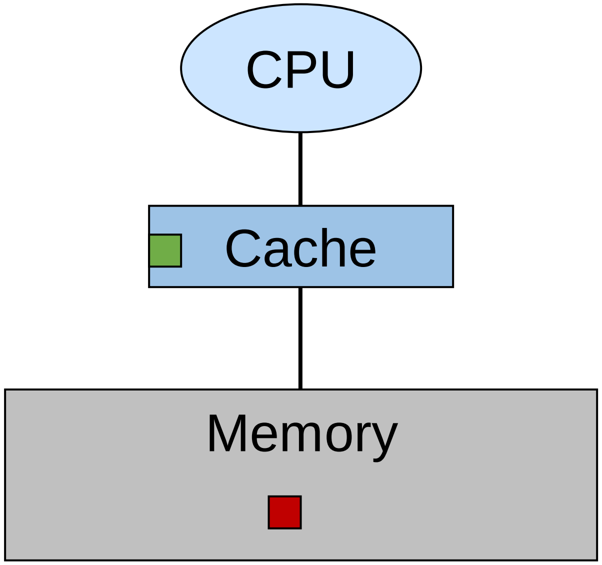


Mem[214] = 21763

Index	D	V	Tag	Data
...				
110	1	1	11010	21763
...				

Address	Data
1000 1110	1225
1101 0110	42803
...	

credit to: Howard Huang, *Cache writes and examples*



CPU

Cache

Memory

Each cache block has a dirty bit that indicates if the block has changed

If a block is dirty, it is the only copy!

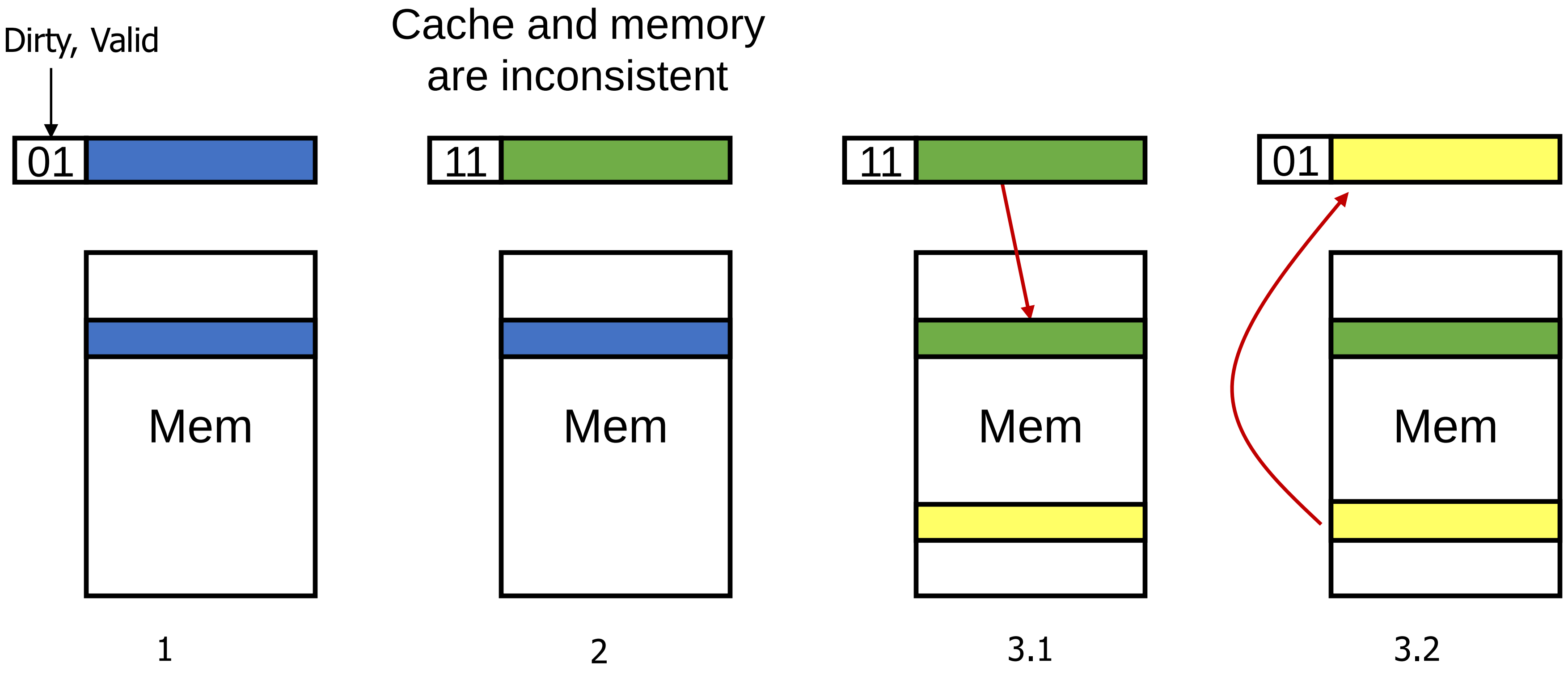
Illustration: Replacing a dirty block on read

1. The blue block is loaded in cache

2. CPU updates the block. The cache block is dirty

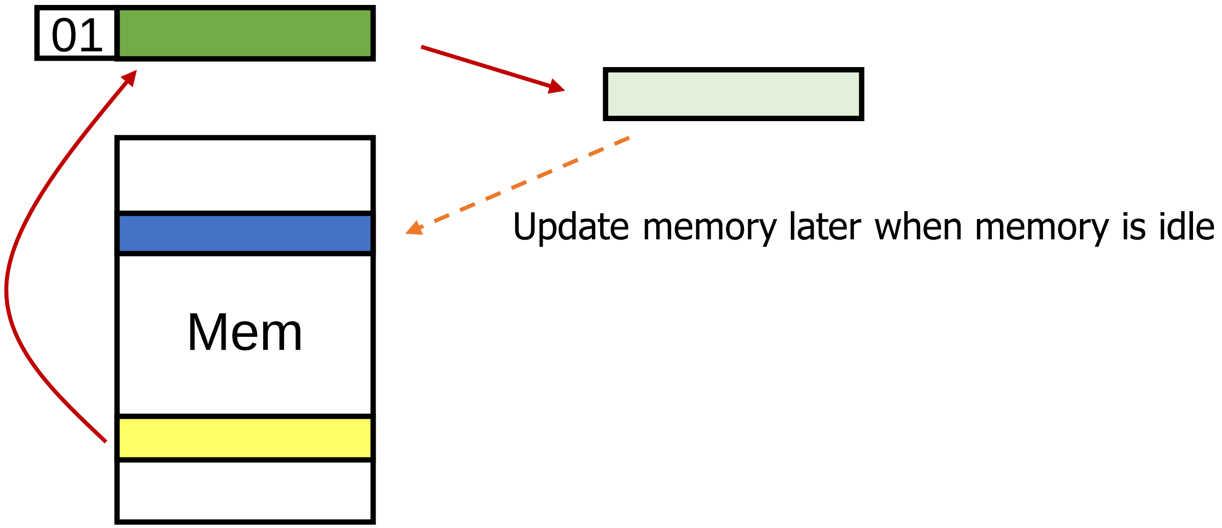
3. To load the yellow block from memory into the same cache block

    1) write the green block to memory and then 2) load yellow block in cache



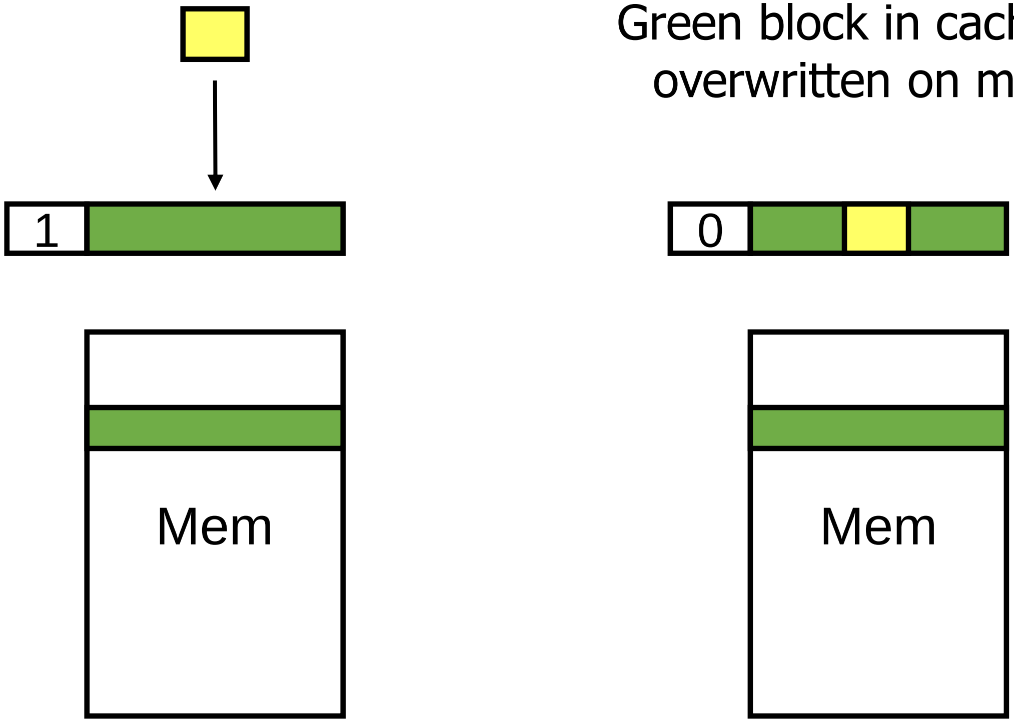
Reduce Replacement Time

- We can do the following in parallel:
  - Load the yellow block into cache
  - Save the green block in an additional buffer (write-back buffer)



Write Operation: Write-through

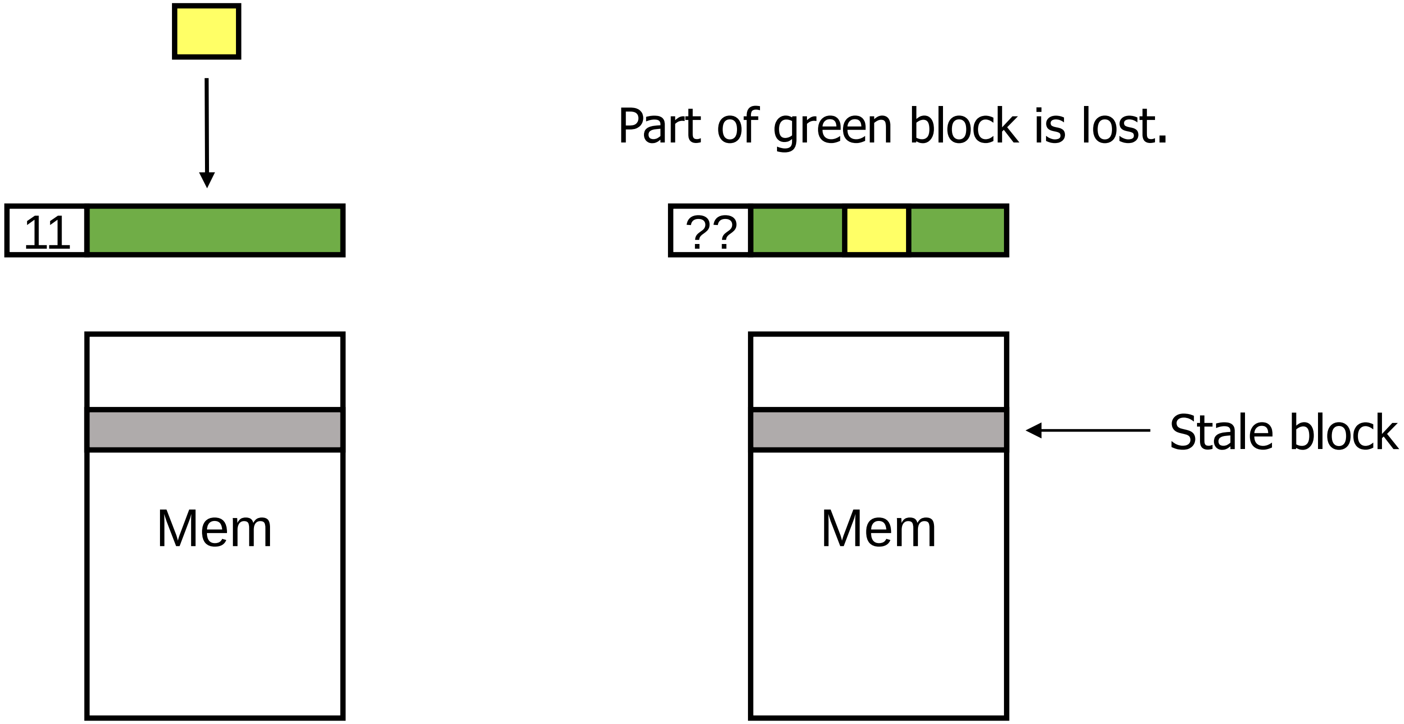
- We can update cache block and check tags at the same time.
  - It is fine even if it is a miss because there is a copy in memory



Green block in cache is overwritten on miss

Write Operation: Write-back

- With write-back, we cannot update before we know it is a hit
  - Need two steps: Is it a hit? If yes, then update cache
    - Takes two cycles, or requires a buffer
  - Otherwise, we may overwrite a dirty block on miss



Handling Write Misses (D$ only)

- Write allocate: load the block in cache first
  - Stall the pipeline
  - Fetch the block from next level in the memory hierarchy into cache
    - May have to evict a dirty block if using a write-back cache
  - Write the word from the processor to the cache and set the dirty bit
  - Let the pipeline resume

- No write allocate: Skip the cache, write to memory directly
  - Just write the word to the write buffer
  - No need to stall if the write buffer is not full

Write Back Summary

- Updates cache only.
  - A dirty bit for each cache block indicates if the block is changed
- When placing a block in cache, we need to check if it replaces a dirty block
  - Write the dirty block to memory first
  - A miss may result in two memory operations: write dirty block and load the new block. Longer access time!
- When writing to a cache block, we must sure it is a hit before updating blocks
  - Cannot overwrite the only copy of the data in cache
  - At least 2 cycles!

Questions about cache

Q1: Where do we place a block in cache?

Q2: How do we find a data item is in the cache?

Where to find it? How to check it is the block?

Q3: How do we replace a block in cache?

Q4: How do we handle write?