CS-200
Computer Architecture
—
Part 3b. Memory Hierarchy
Simple Cache Examples

Paolo Ienne
<paolo.ienne@epfl.ch>

1. Compulsory Misses of a Direct Mapped $

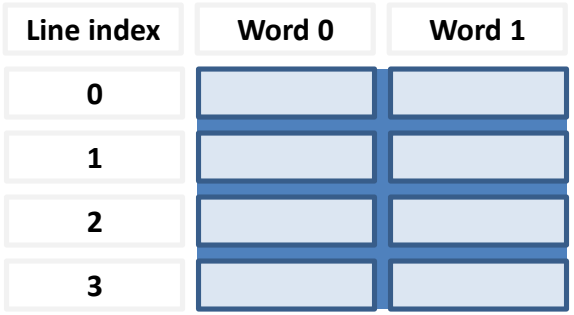
- Given an initially empty **direct-mapped cache** with **4 lines** and **2 words per line**, find the total number of compulsory misses for the following memory access sequence (in decimal):

  12, 15, 15, 11, 1, 17, 3, 17, 11, 17

- The memory is word-addressable
- The least recently used (LRU) replacement policy is used

Solution

- Given that we have a **direct-mapped** cache, replacement policy is **inapplicable**
- Direct-mapped cache with 4 lines and 2 words per line



- Compulsory misses are all those misses that **could not have been avoided**; hence, the # of compulsory misses equals the # of **unique cache blocks** accessed
  - There are 6 unique cache blocks accessed: (**12**,13), (14, **15**), (10, **11**), (0, **1**), (16, **17**), (2, **3**)
  - Hence, there are 6 compulsory misses

2. Hits in a 2-Way Set-Associative $

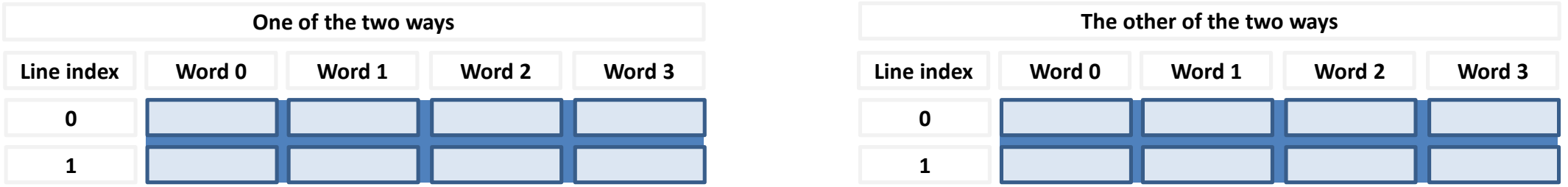
- Given an initially empty **2-way set-associative cache** with **2 sets** and **4 words per block**, find the total number of hits for the following memory access sequence (in decimal):

  11, 15, 4, 16, 3, 10, 12, 24, 19, 25

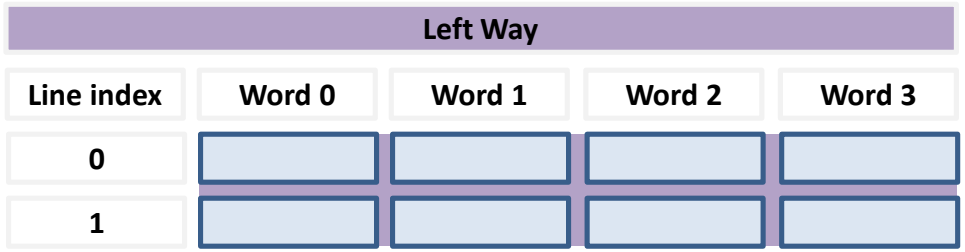
- The memory is word-addressable
- The least recently used (LRU) replacement policy is used

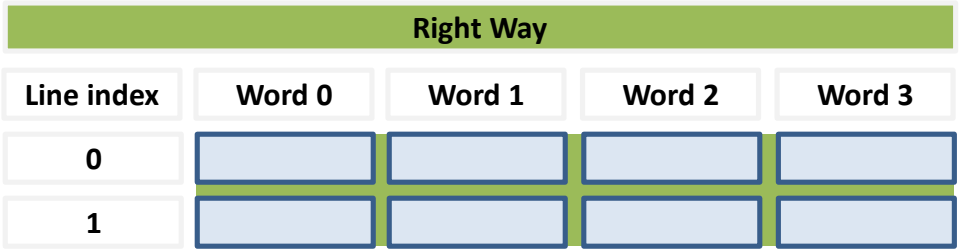
Solution

- Given that we have a **set-associative** cache, replacement policy is **relevant**
- 2-way set-associative cache with 2 sets and 4 words per block



- Bits 0 and 1 of the address (i.e., the last two bits) select the word in the line (0, 1, 2, or 3)
- Bit 2 of the address corresponds to the line/set index (0 or 1)





Address sequence	Set index	Way	Words	Hit? Miss?	Anything evicted to make space for this?
11 = $(0\ 1011)_2$	0	L	8, 9, 10, 11	Miss	No.
15 = $(0\ 1111)_2$	1	L	12, 13, 14, 15	Miss	No
4 = $(0\ 0100)_2$	1	R	4, 5, 6, 7	Miss	No
16 = $(1\ 0000)_2$	0	R	16, 17, 18, 19	Miss	No
3 = $(0\ 0011)_2$	0	L	0, 1, 2, 3	Miss	(8, 9, 10, 11) @ L
10 = $(0\ 1010)_2$	0	R	8, 9, 10, 11	Miss	(16, 17, 18, 19) @ R
12 = $(0\ 1100)_2$	1	L	12, 13, 14, 15	Hit	-
24 = $(1\ 1000)_2$	0	L	24, 25, 26, 27	Miss	(0, 1, 2, 3) @ L
19 = $(1\ 0011)_2$	0	R	16, 17, 18, 19	Miss	(8, 9, 10, 11) @ R
25 = $(1\ 1001)_2$	0	L	24, 25, 26, 27	Hit	-

Answer: 2

3. Miss Rate in a 2-Way Set-Associative $

- Given an initially empty **2-way set-associative cache** with **2 sets**, find the miss rate for the following **block address** access sequence (in decimal):

  6, 0, 0, 5, 5, 6, 1, 2, 2, 0

- The memory is word-addressable
- The least recently used (LRU) replacement policy is used

Solution

- The least significant bit of the block address is the set

Block Address sequence	Set index	Way	Hit? Miss?	Anything evicted to make space for this?
6	0	L	Miss	No
0	0	R	Miss	No
0	0	R	Hit	-
5	1	L	Miss	No
5	1	L	Hit	-
6	0	L	Hit	-
1	1	R	Miss	No
2	0	R	Miss	Yes, 0 @ R (least recently accessed)
2	0	R	Hit	-
0	0	L	Miss	Yes, 6 @ L (least recently accessed)

Miss Rate
= #misses / #accesses =
60%

4. Direct-Mapped $

- Consider a **direct-mapped cache** with **a capacity of 128 KiB** and **2 words per line**
- The memory is word-addressable (one word = 4 bytes)
- The memory address has 32 bits
- Show how the address bits are used in the cache

Solution

- No bits are used to select the byte (word-addressed)
- Bit 0 used to select the word (2 words/line)
- Bits 1-14 used for the index
  - Total storage: 128 KiB = 2^{17} B
  - Each line = 2 words × 4 B/word = 2^3 B
  - Number of lines = 2^{17} / 2^3 = 2^{14}
  - Bits for the index = 14
- Bits 15-31 are the tag

5. Tag of a 2-Way Set-Associative $

- Consider a **2-way set-associative cache** with a **capacity of 128 KiB** and **8 words per line**
- The memory is byte-addressable (one word = 4 bytes)
- The memory address has 32 bits
- Find the **tag** for memory address `0xa8f4bfc6`

Solution

- Bits 0-1 used to select the byte (byte-addressed)
- Bits 2-4 used to select the word (8 words/line)
- Bits 5-15 used for the index
  - Total storage: 128 KiB = 2^{17} B
  - Each line = 8 words × 4 B/word = 2^5 B
  - Number of total lines = 2^{17} / 2^5 = 2^{12}
  - Number of lines per way = 2^{12} / 2 = 2^{11}
  - Bits for the index = 11
- Bits 16-31 are the tag: `0xa8f4bfc6` → `0xa8f4`