Block addressing

An address in main memory points to the first byte of a block when all of the block offset bits are set to zero.

Example:

Given a block size of 8 bytes, 3 bits are required to represent the block offset.

Hence, the address 0x71 points to the second byte in the block:

Address in binary: 01110001

The first byte of the block is determined by setting the last three bits to zeroes:

Start of block in binary: 01110000

Cache simulations

We are going to simulate a series of instructions being executed by the CPU.

Each instruction is trying to copy one byte from main memory into a register (MOV).

Instead of reading directly from main memory, however, our instructions are going to read from the cache.

Thus, we must first check whether the desired data already exists in the cache.

If the data is not present in the cache, we must copy the data from main memory into the cache.

We then return the byte to the CPU from the cache.

Direct mapped cache simulation

Assumptions:

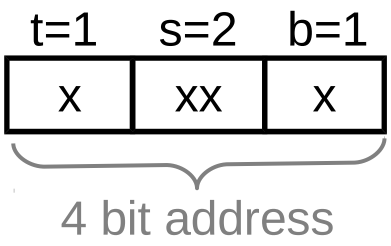
Size of main memory: 16 bytes (really small!)
Address size needed: 4 bits

Number of sets: 4
Number of set bits needed: 2

Block size: 2
Number of block offset bits needed: 1

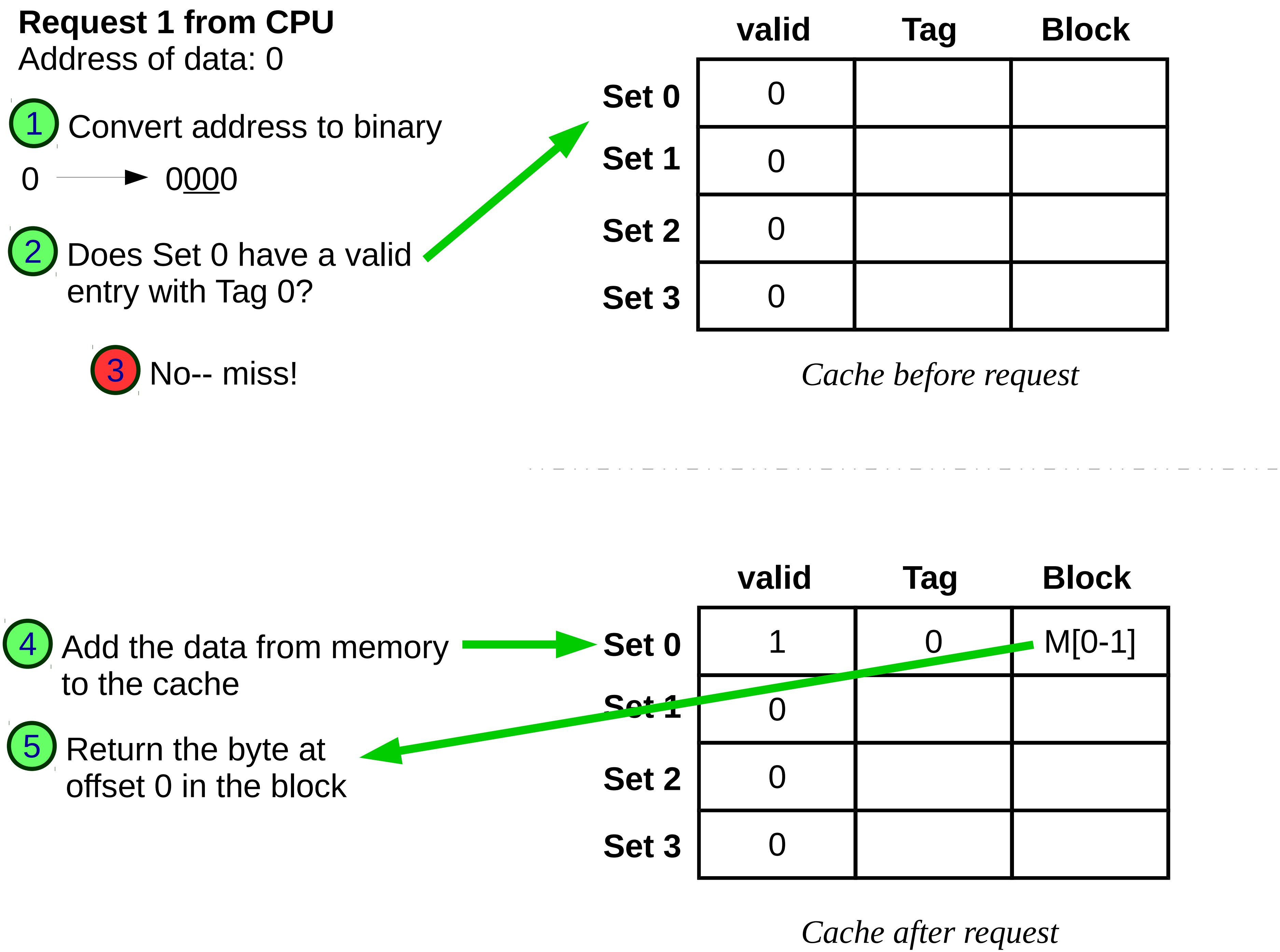
Tag size: 1 bit (4 total bits - 2 set bits - 1 block bit)
Entries per set: 1 (direct mapped)

Address to cache conversion:



Initial state of the cache (cold); all entries are invalid

	valid	**Tag**	**Block**
Set 0	0		
Set 1	0		
Set 2	0		
Set 3	0		



Request 1 from CPU
Address of data: 0

1 Convert address to binary

0 → $0\underline{00}0$

2 Does Set 0 have a valid entry with Tag 0?

3 No-- miss!

	valid	Tag	Block
Set 0	0		
Set 1	0		
Set 2	0		
Set 3	0		

Cache before request

4 Add the data from memory to the cache

5 Return the byte at offset 0 in the block

	valid	Tag	Block
Set 0	1	0	M[0-1]
Set 1	0		
Set 2	0		
Set 3	0		

Cache after request

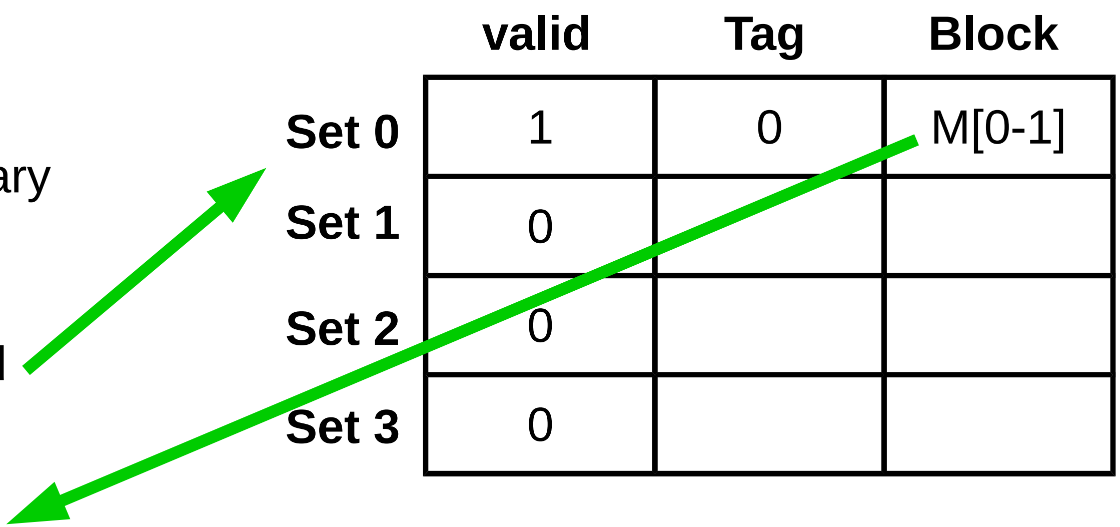
Request 2 from CPU
Address of data: 1

1 Convert address to binary

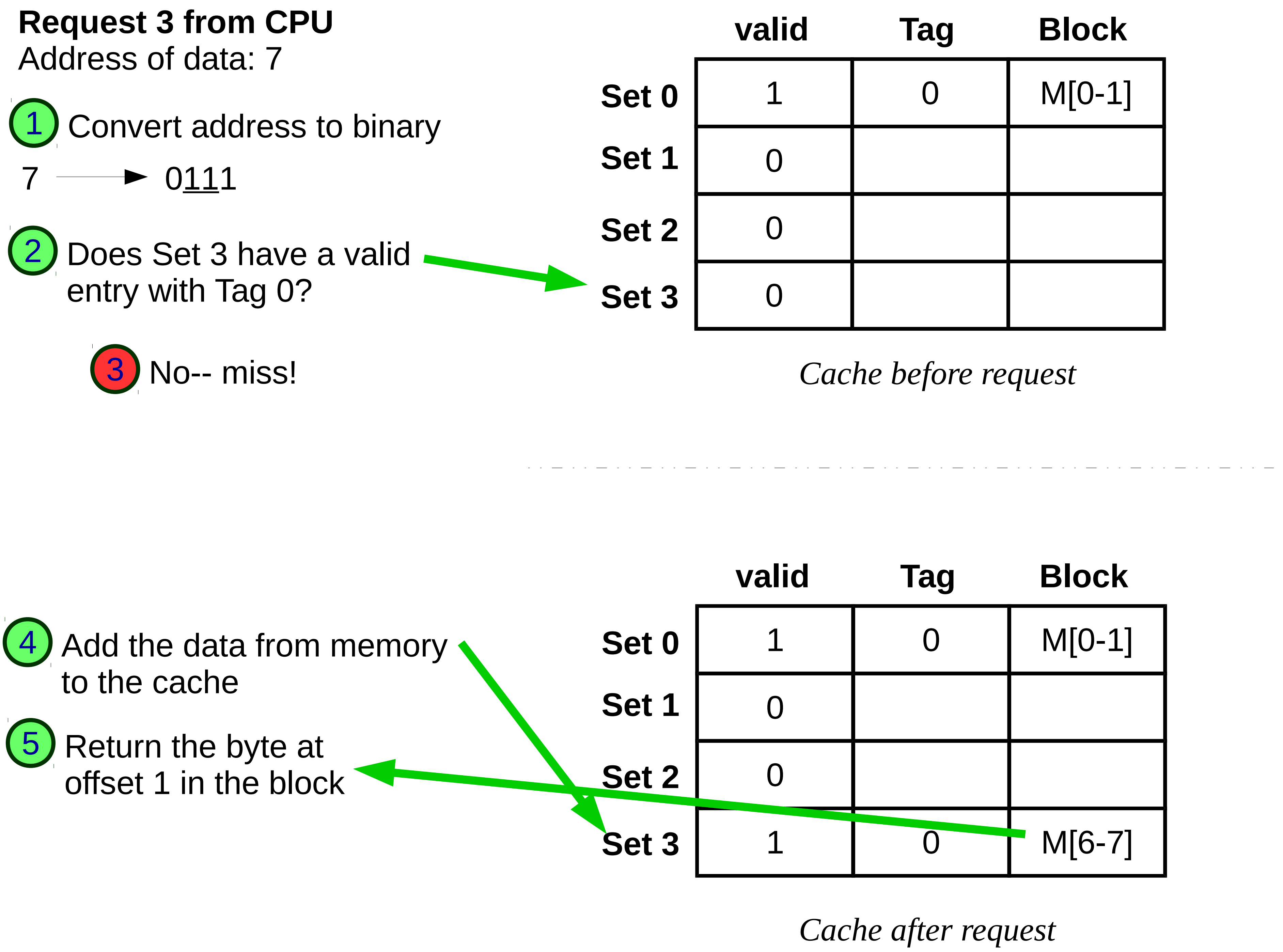
1 → 0001

2 Does Set 0 have a valid entry with Tag 0?

3 Yes! Return the byte at offset 1 in the block



	valid	Tag	Block
Set 0	1	0	M[0-1]
Set 1	0		
Set 2	0		
Set 3	0		



Request 3 from CPU
Address of data: 7

1 Convert address to binary

7 → $0\underline{11}1$

2 Does Set 3 have a valid entry with Tag 0?

3 No-- miss!

	valid	Tag	Block
Set 0	1	0	M[0-1]
Set 1	0		
Set 2	0		
Set 3	0		

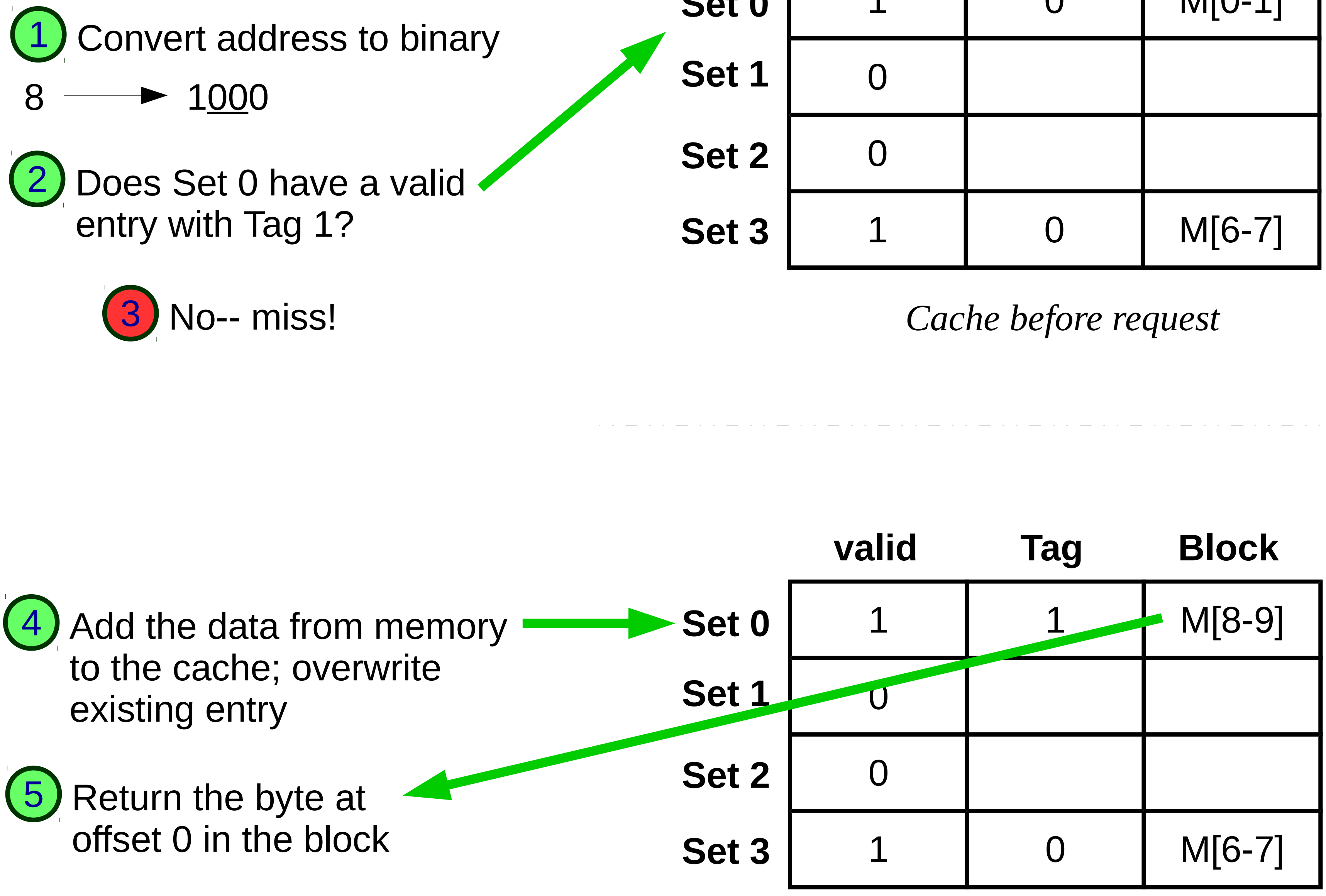
Cache before request

4 Add the data from memory to the cache

5 Return the byte at offset 1 in the block

	valid	Tag	Block
Set 0	1	0	M[0-1]
Set 1	0		
Set 2	0		
Set 3	1	0	M[6-7]

Cache after request



Request 4 from CPU
Address of data: 8

1 Convert address to binary

8 → 1$\underline{00}$0

2 Does Set 0 have a valid entry with Tag 1?

3 No-- miss!

	valid	Tag	Block
Set 0	1	0	M[0-1]
Set 1	0		
Set 2	0		
Set 3	1	0	M[6-7]

Cache before request

4 Add the data from memory to the cache; overwrite existing entry

5 Return the byte at offset 0 in the block

	valid	Tag	Block
Set 0	1	1	M[8-9]
Set 1	0		
Set 2	0		
Set 3	1	0	M[6-7]

Cache after request

Request 5 from CPU
Address of data: 6

1 Convert address to binary

6 → $0\underline{11}0$

2 Does Set 3 have a valid entry with Tag 0?

3 Yes! Return the byte at offset 0 in the block

	valid	Tag	Block
Set 0	1	1	M[8-9]
Set 1	0		
Set 2	0		
Set 3	1	0	M[6-7]

Request 6 from CPU
Address of data: 0

1 Convert address to binary

0 → 0000

2 Does Set 0 have a valid entry with Tag 0?

3 No-- miss!

	valid	Tag	Block
Set 0	1	1	M[8-9]
Set 1	0		
Set 2	0		
Set 3	1	0	M[6-7]

Cache before request

4 Add the data from memory to the cache; overwrite existing entry

5 Return the byte at offset 0 in the block

	valid	Tag	Block
Set 0	1	0	M[0-1]
Set 1	0		
Set 2	0		
Set 3	1	0	M[6-7]

Cache after request

2-Way associative cache simulation

Assumptions:

Size of main memory: 16 bytes (really small!)
Address size needed: 4 bits

Number of sets: 2
Number of set bits needed: 1

Block size: 2
Number of block offset bits needed: 1

Tag size: 2 bits (4 total bits - 1 set bit - 1 block bit)
Entries per set: 2 (2-way associative)

Address to cache conversion:

t=2	s=1	b=1
xx	x	x

4 bit address

Initial state of the cache (cold); all entries are invalid

	valid	Tag	Block
Set 0	0		
	0		
Set 1	0		
	0		

Request 1 from CPU
Address of data: 0

1 Convert address to binary

0 → $00\underline{0}0$

2 Does Set 0 have a valid entry with Tag 00?

3 No-- miss!

	valid	Tag	Block
Set 0	0		
	0		
Set 1	0		
	0		

Cache before request

4 Add the data from memory to the cache

5 Return the byte at offset 0 in the block

	valid	Tag	Block
Set 0	1	00	M[0-1]
	0		
Set 1	0		
	0		

Cache after request

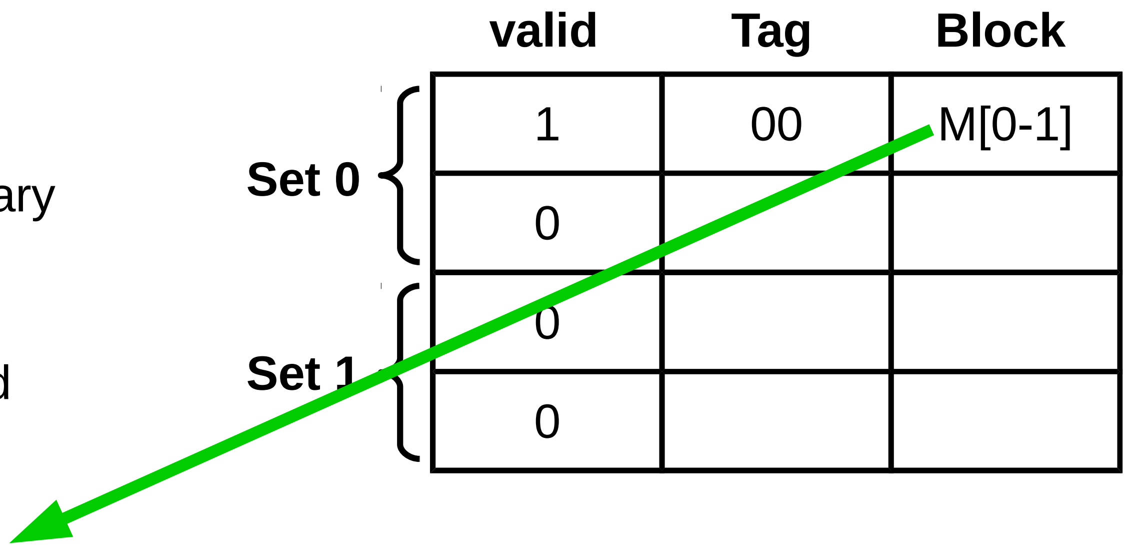
Request 2 from CPU
Address of data: 1

1 Convert address to binary

$0 \longrightarrow 00\underline{0}1$

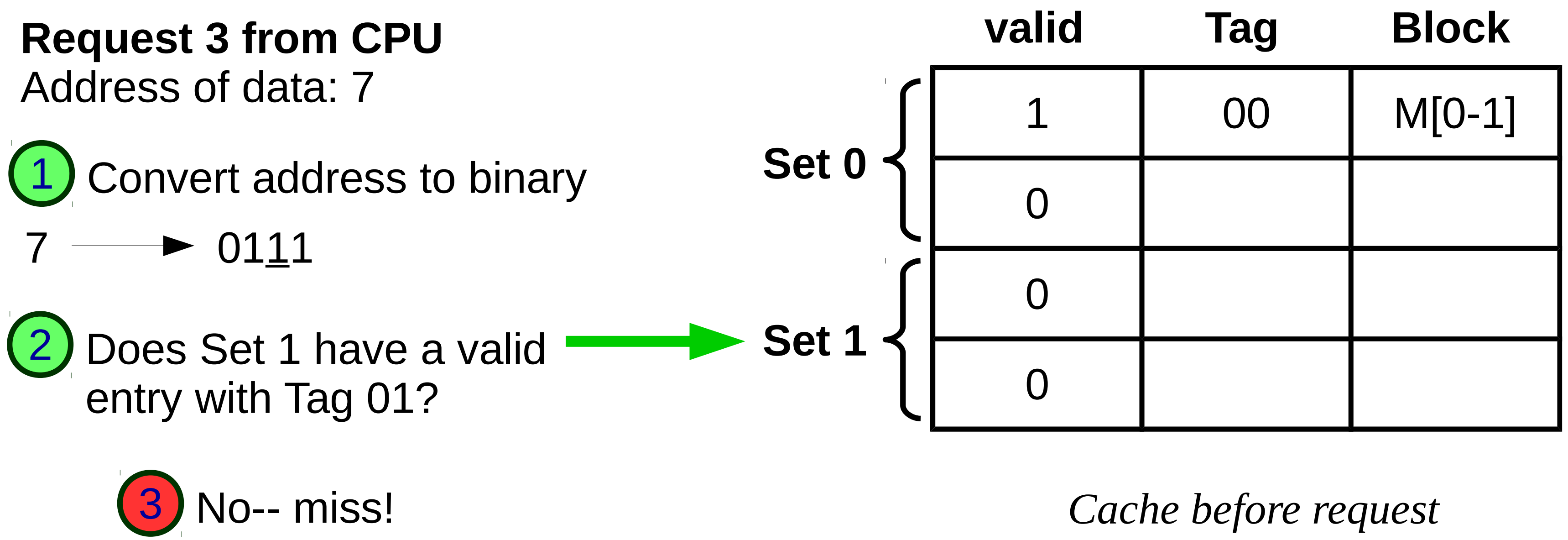
2 Does Set 0 have a valid entry with Tag 00?

3 Yes! Return the byte at offset 1 in the block



	valid	Tag	Block
Set 0	1	00	M[0-1]
	0		
Set 1	0		
	0		

Request 3 from CPU
Address of data: 7



1 Convert address to binary

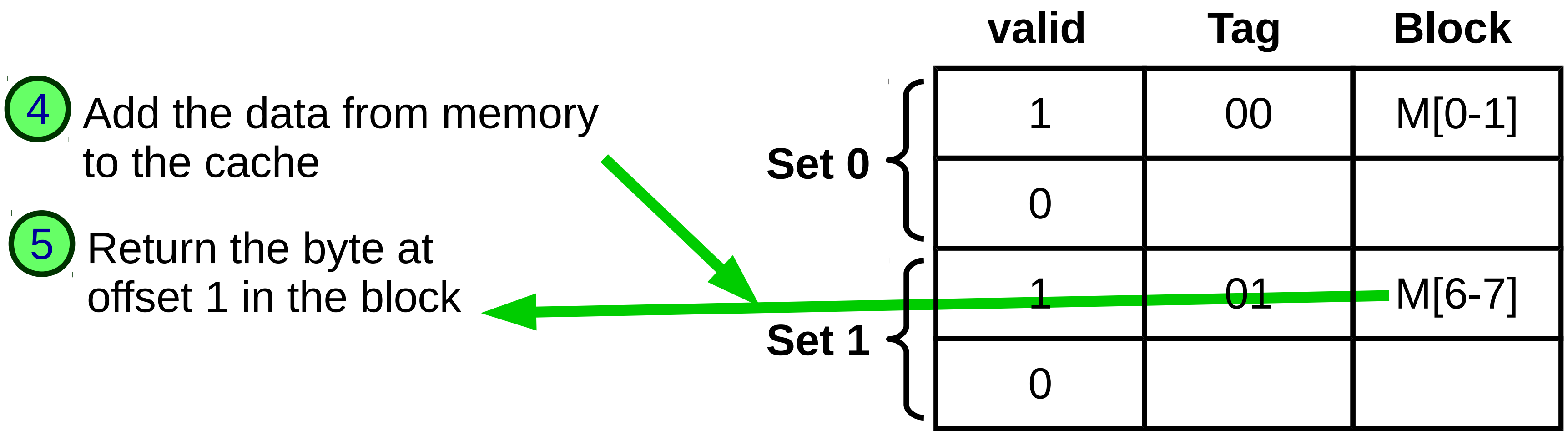
7 → $01\underline{1}1$

2 Does Set 1 have a valid entry with Tag 01?

3 No-- miss!

	valid	Tag	Block
Set 0	1	00	M[0-1]
	0		
Set 1	0		
	0		

Cache before request



4 Add the data from memory to the cache

5 Return the byte at offset 1 in the block

	valid	Tag	Block
Set 0	1	00	M[0-1]
	0		
Set 1	1	01	M[6-7]
	0		

Cache after request

Request 4 from CPU
Address of data: 8

1 Convert address to binary

8 → $10\underline{0}0$

2 Does Set 0 have a valid entry with Tag 10?

3 No-- miss!

	valid	Tag	Block
Set 0	1	00	M[0-1]
	0		
Set 1	1	01	M[6-7]
	0		

Cache before request

4 Add the data from memory to the cache

5 Return the byte at offset 0 in the block

	valid	Tag	Block
Set 0	1	00	M[0-1]
	1	10	M[8-9]
Set 1	1	01	M[6-7]
	0		

Cache after request

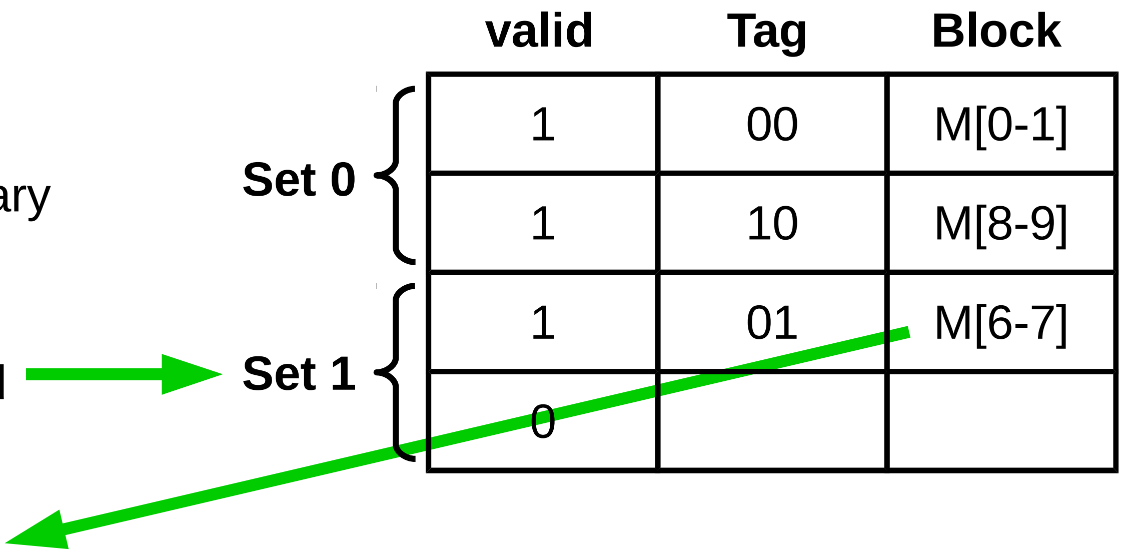
Request 5 from CPU
Address of data: 6

1 Convert address to binary

6 → 01$\underline{1}$0

2 Does Set 1 have a valid entry with Tag 01?

3 Yes! Return the byte at offset 0 in the block



	valid	Tag	Block
Set 0	1	00	M[0-1]
	1	10	M[8-9]
Set 1	1	01	M[6-7]
	0		

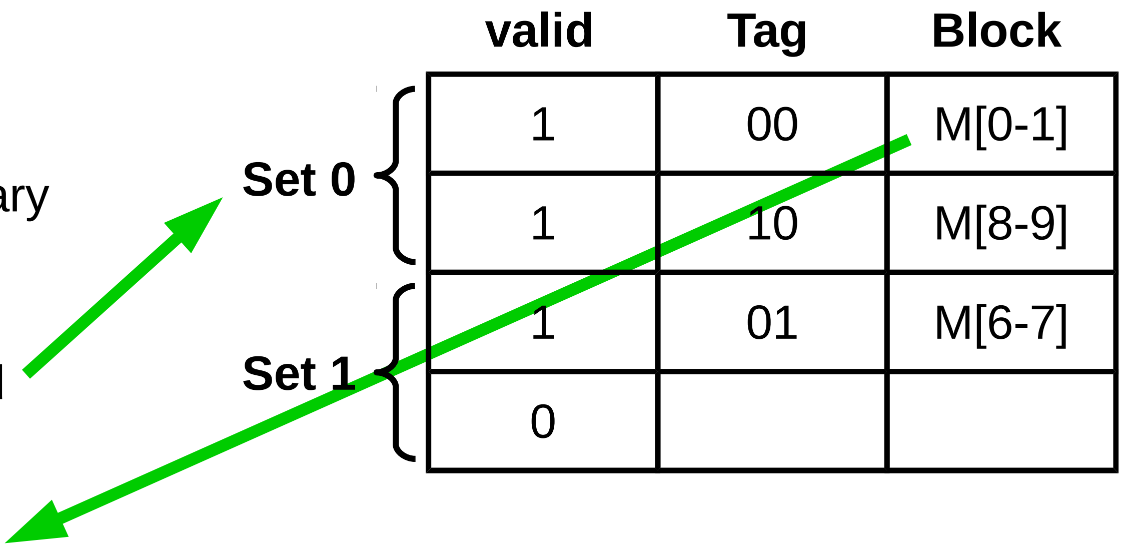
Request 6 from CPU
Address of data: 0

1 Convert address to binary

$0 \longrightarrow 00\underline{0}0$

2 Does Set 0 have a valid entry with Tag 00?

3 Yes! Return the byte at offset 0 in the block



	valid	Tag	Block
Set 0	1	00	M[0-1]
	1	10	M[8-9]
Set 1	1	01	M[6-7]
	0		