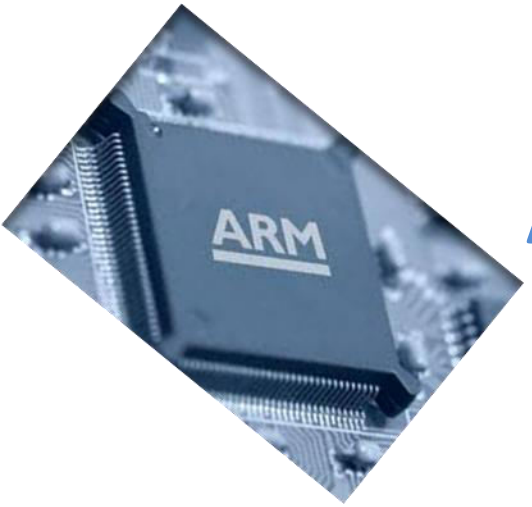


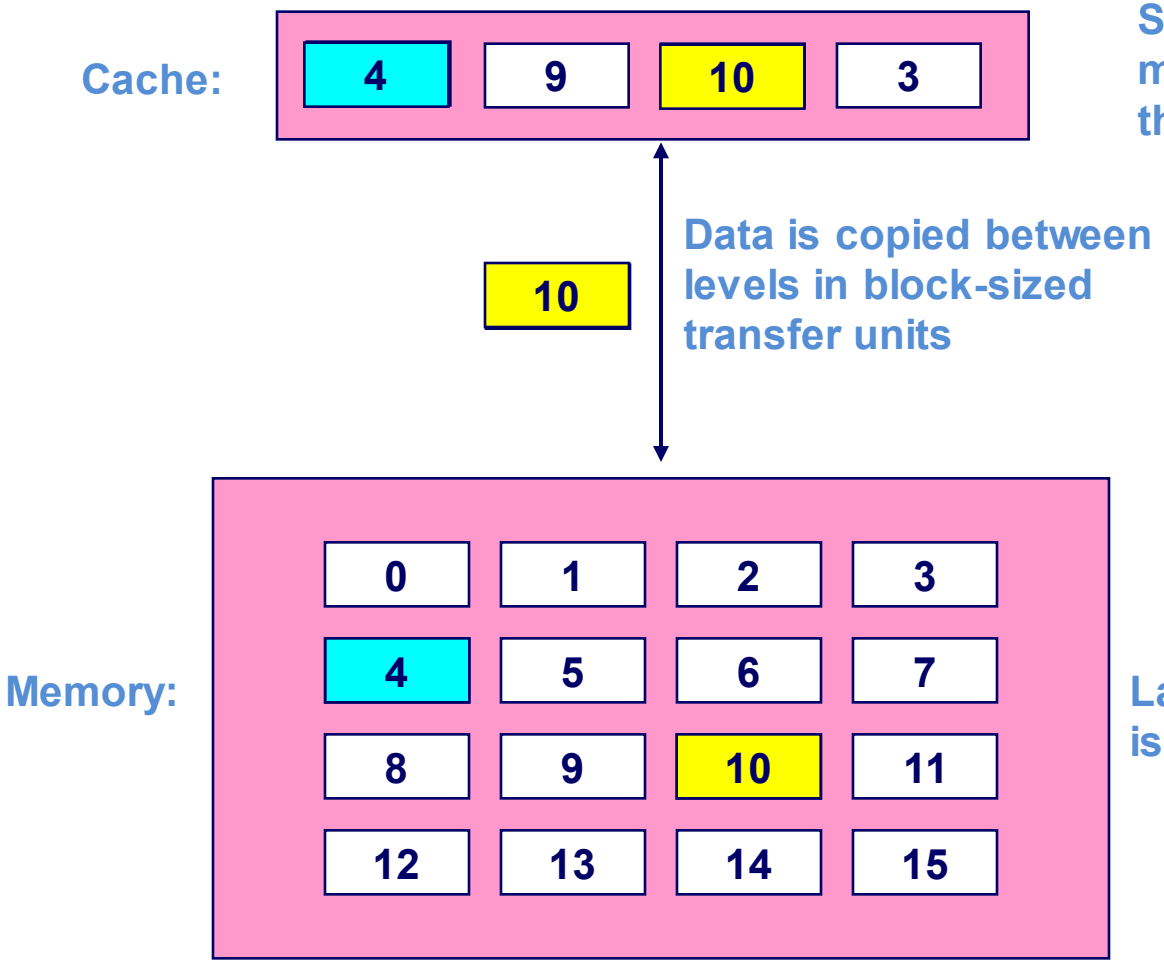
MICROPROCESSOR AND COMPUTER ARCHITECTURE

CACHE MEMORY

Generic cache memory organization
Direct mapped caches
Set associative caches

Credits:
MPCA Team

General Cache Mechanics



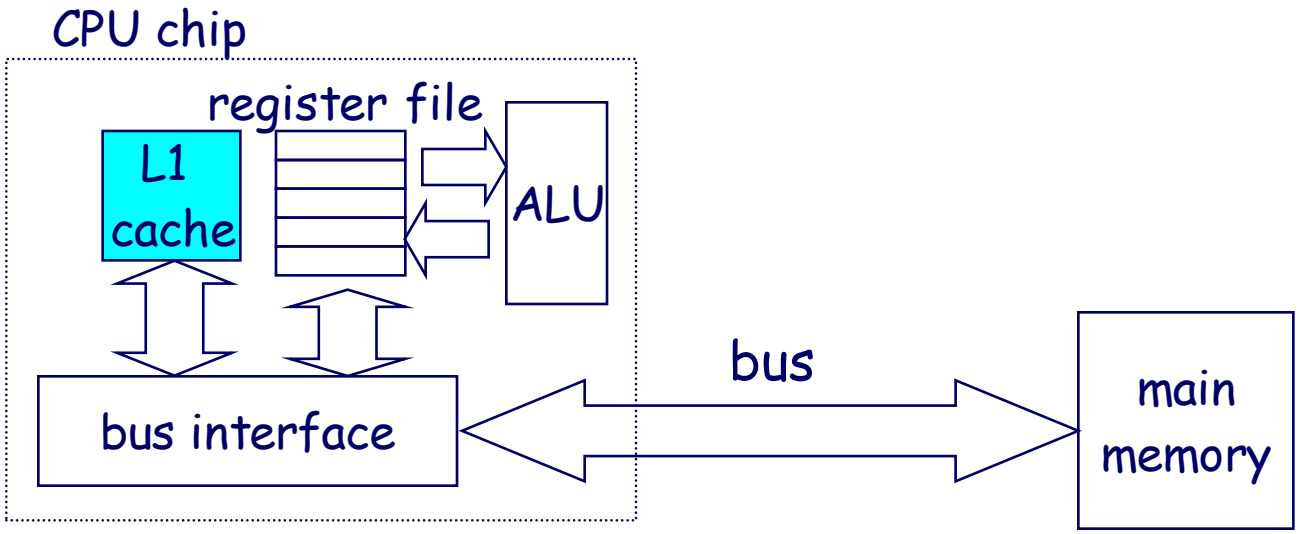
Hardware Cache Memories

Cache memories are small, fast SRAM-based memories managed automatically in hardware

- Hold frequently accessed blocks of main memory

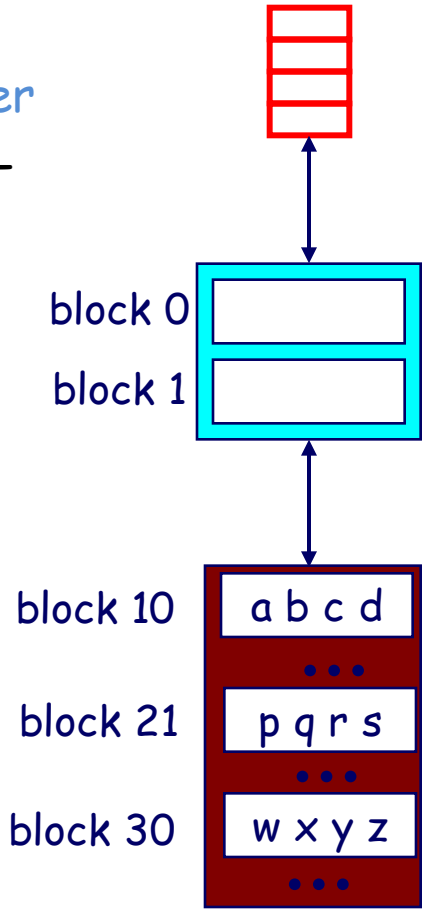
CPU looks first for data in L1, then in main memory

Typical system structure:



Inserting an L1 cache between the CPU and Main Memory

The transfer unit between the CPU register file and the cache is a 4-byte word



The tiny, very fast CPU register file has room for four 4-byte words

The small fast L1 cache has room for two 4-word blocks

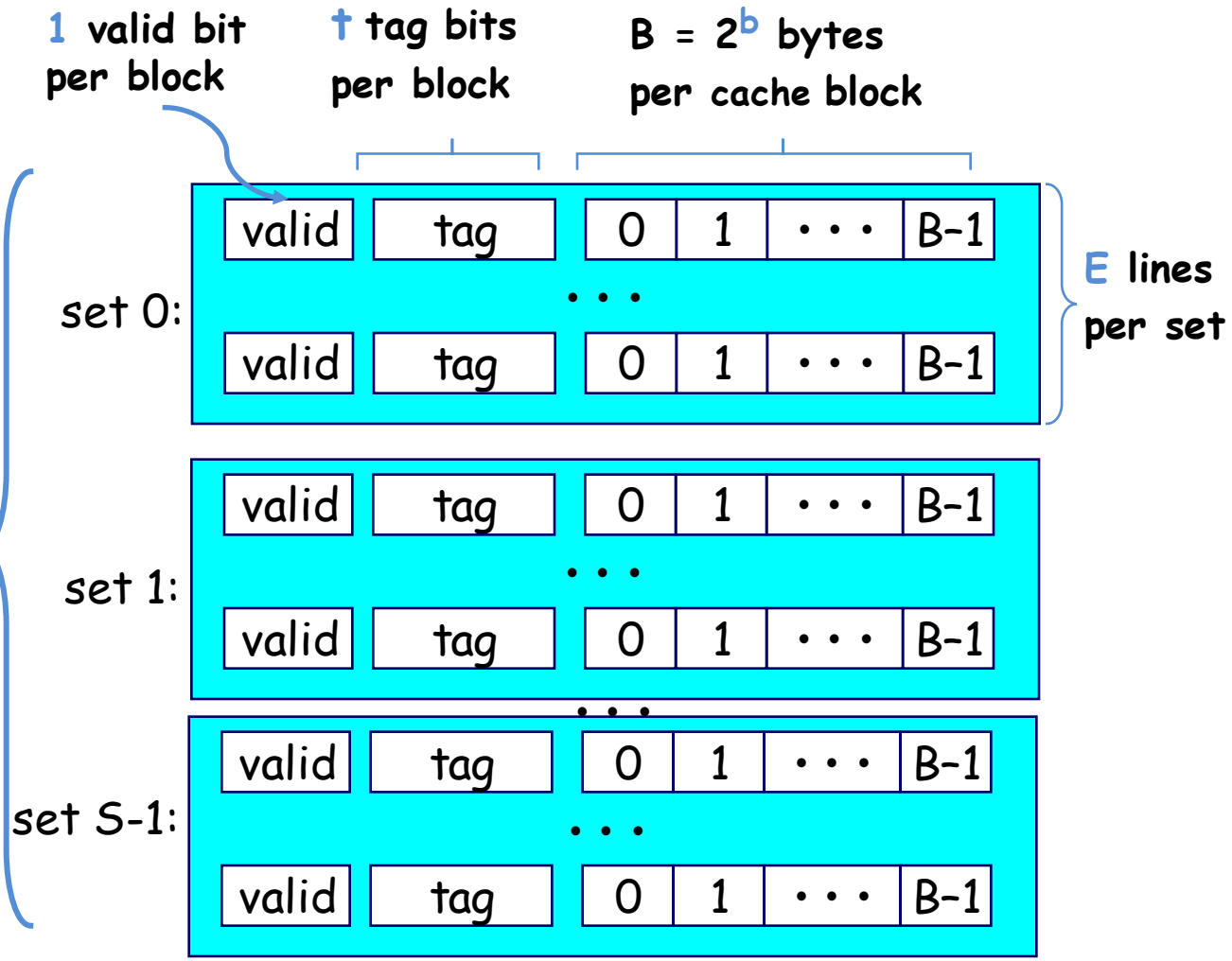
The transfer unit between the cache and main memory is a 4-word block (16 bytes)

The big slow main memory has room for many 4-word blocks

General Organization Of Cache

Cache is an array of sets.
Each set contains one or more blocks

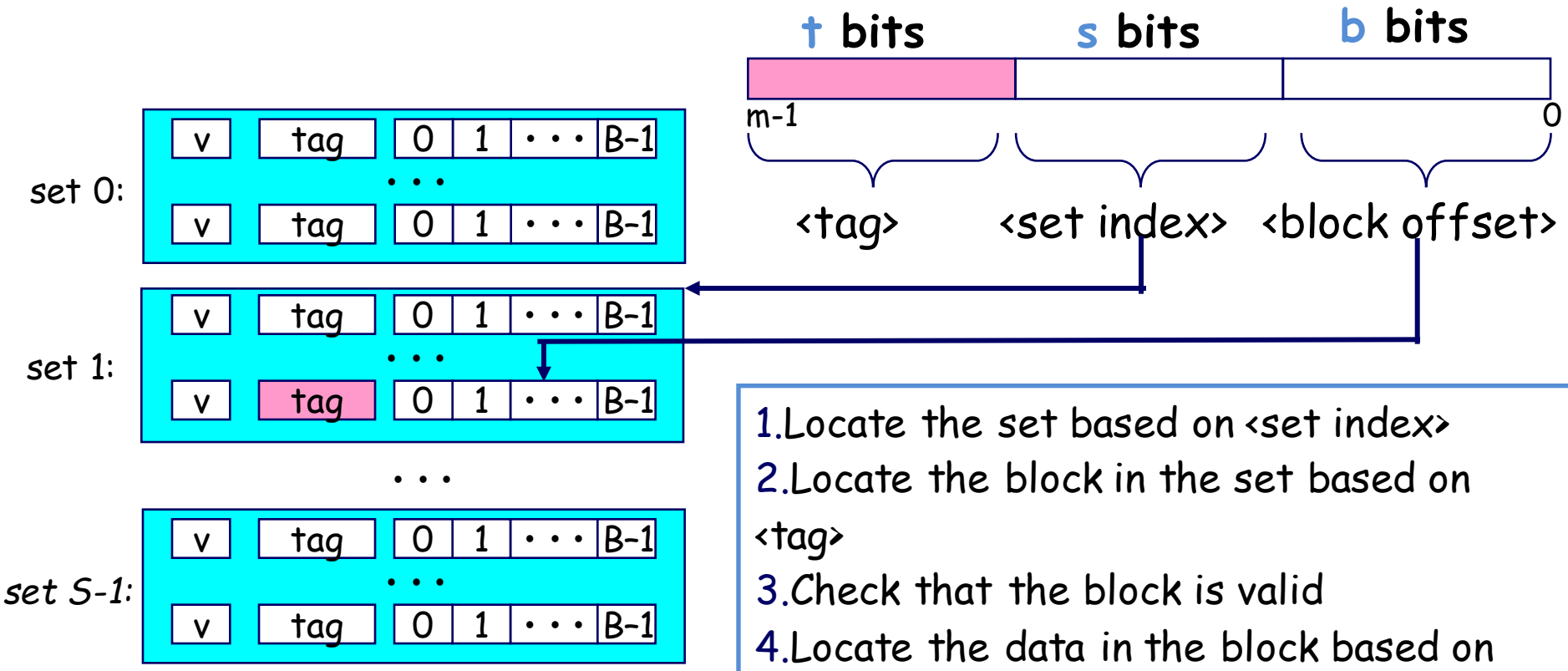
$S = 2^s$ sets



Cache size:
$C = B \times E \times S$ data bytes

Addressing Cache

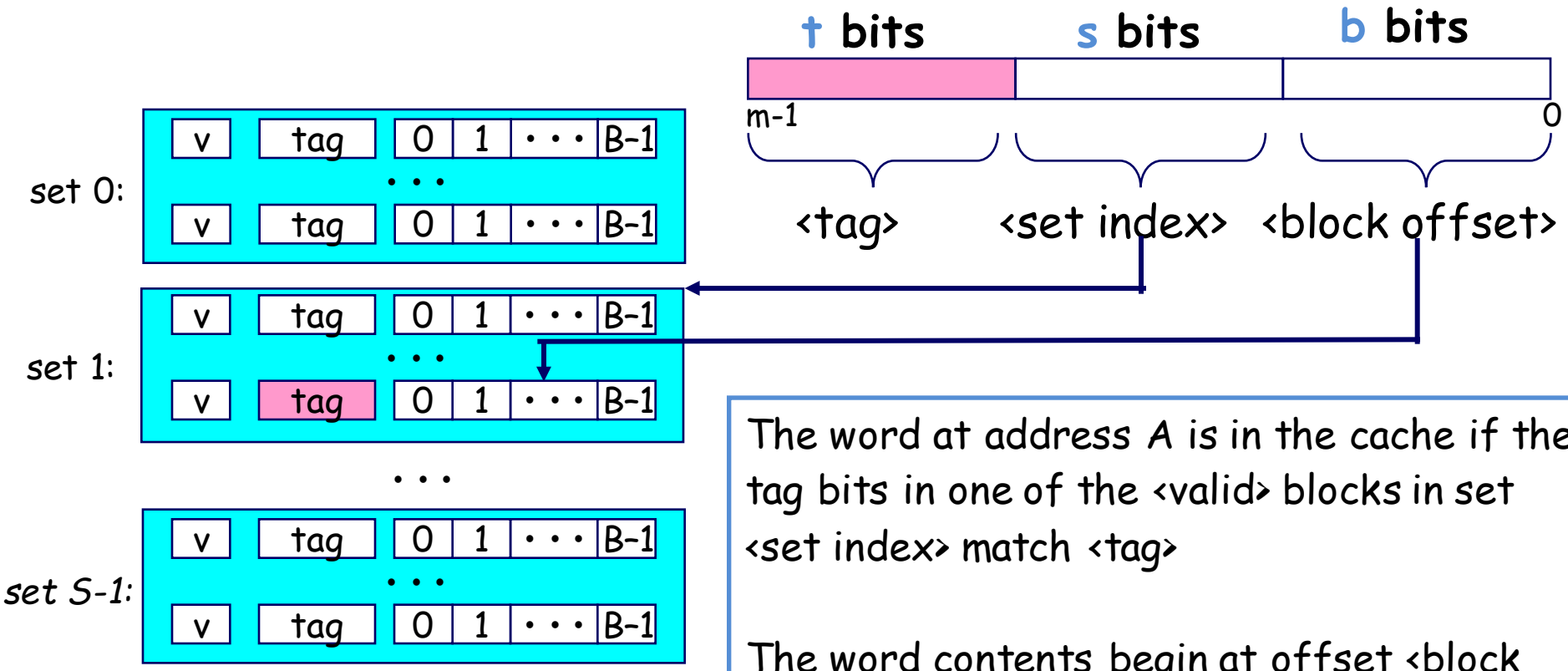
Address A:



1. Locate the set based on <set index>
2. Locate the block in the set based on <tag>
3. Check that the block is valid
4. Locate the data in the block based on <block offset>

Addressing Cache

Address A:



The word at address A is in the cache if the tag bits in one of the <valid> blocks in set <set index> match <tag>

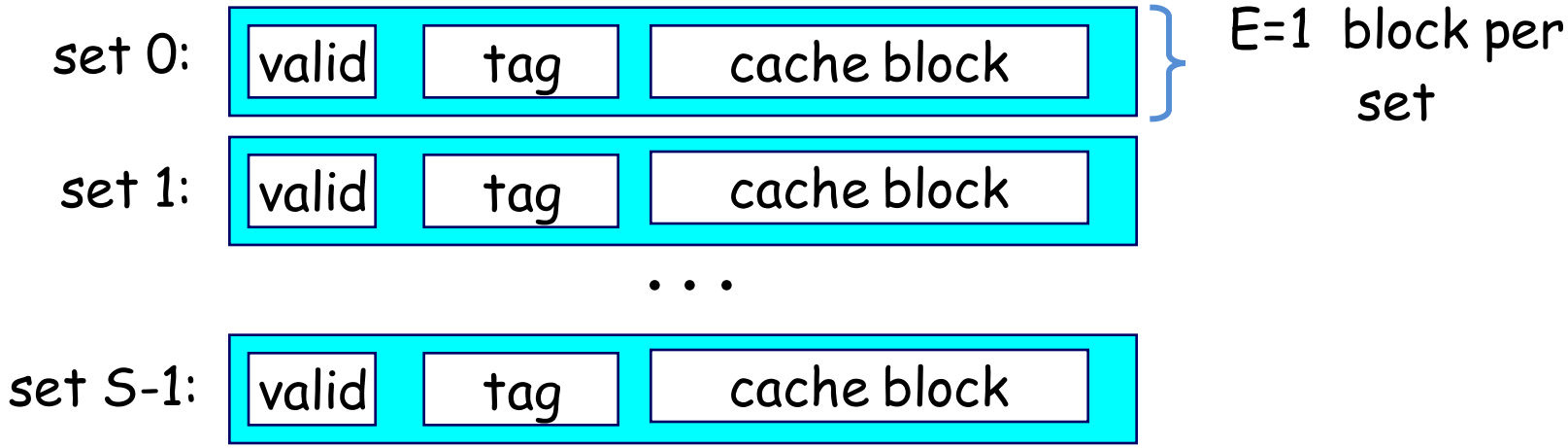
The word contents begin at offset <block offset> bytes from the beginning of the block

Example: Direct-Mapped Cache

Simplest kind of cache, easy to build

Only 1 tag compare required per access

Characterized by exactly one block per set.

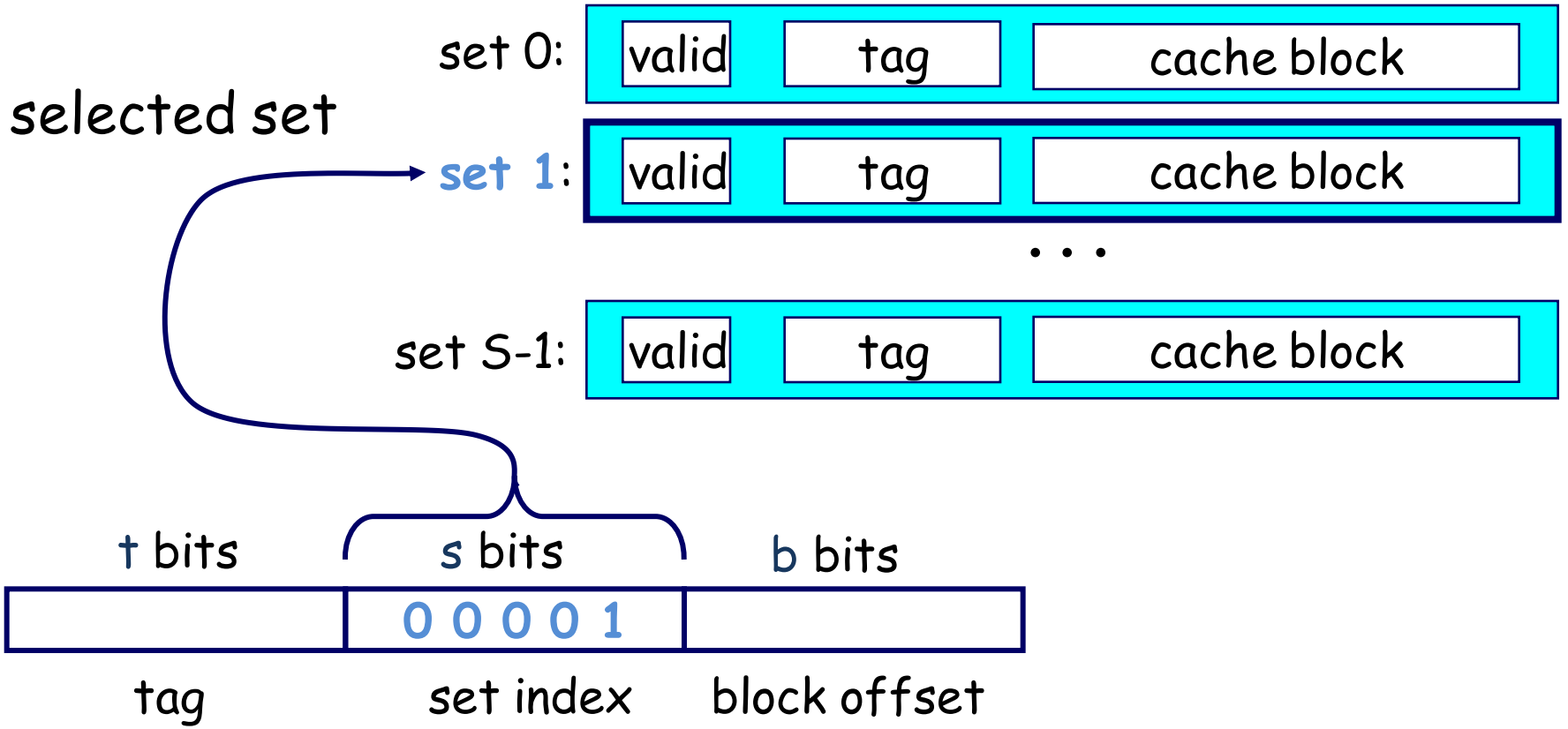


Cache size: $C = B \times S$ data bytes

Accessing Direct-Mapped Cache

Set selection

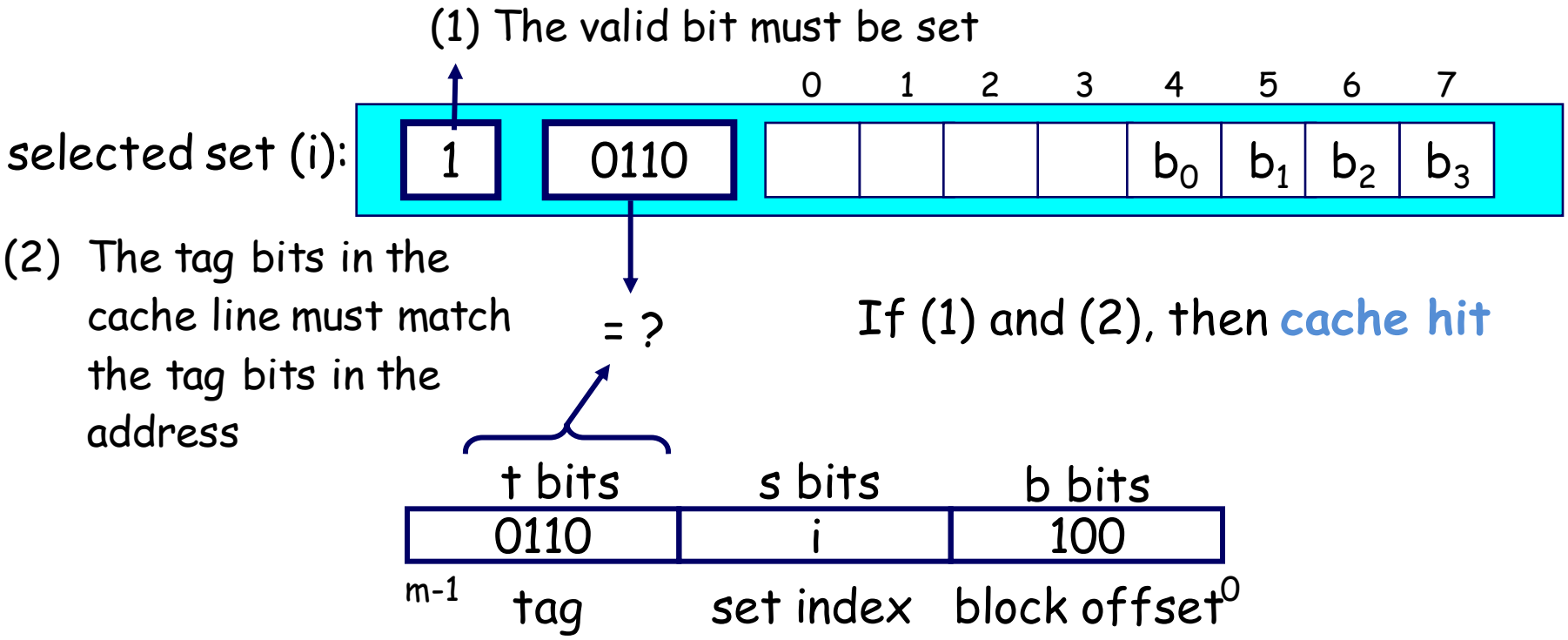
– Use the set index bits to determine the set of interest.



Accessing Direct-Mapped Cache

Block matching and word selection

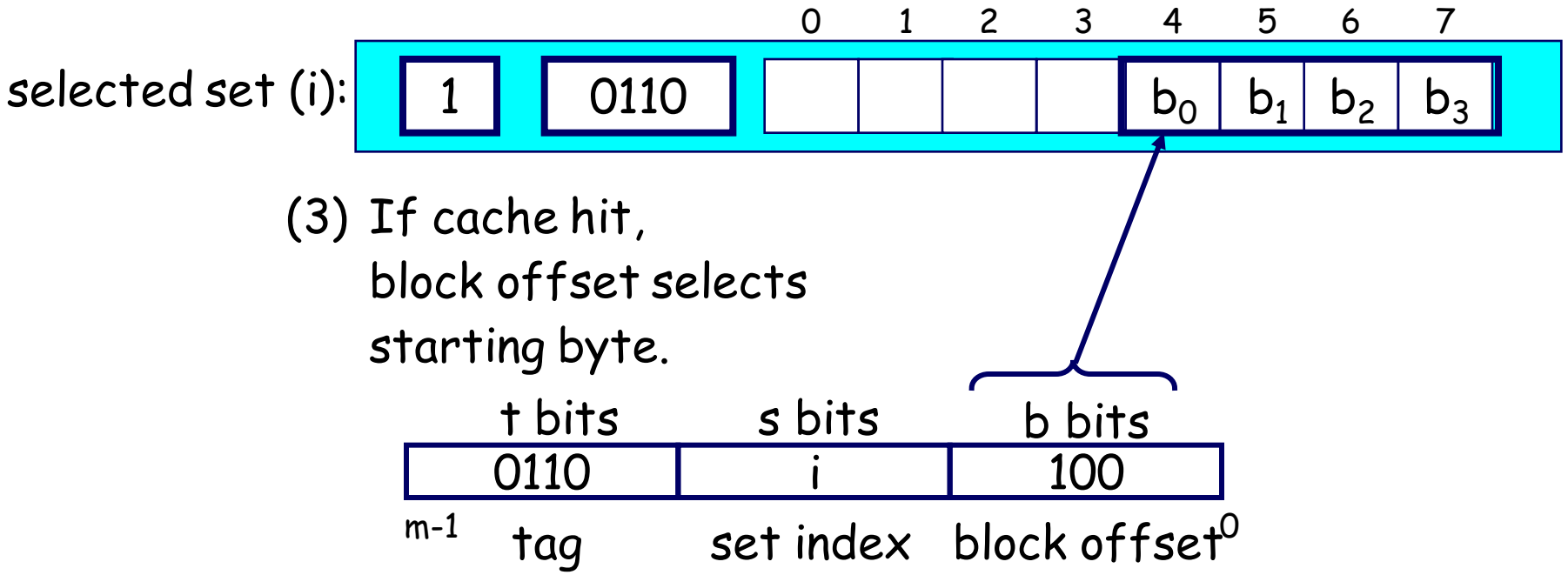
- **Block matching**: Find a valid line in the selected set with a matching tag
- **Word selection**: Then extract the word



Accessing Direct-Mapped Cache

Block matching and word selection

- **Block matching**: Find a valid line in the selected set with a matching tag
- **Word selection**: Then extract the word



Direct-Mapped Cache Simulation

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

M=16 byte addresses, B=2 bytes/block, S=4 sets, E=1 entry/set

Main memory addresses

0	$[0000_2]$	miss
1	$[0001_2]$	hit
7	$[0111_2]$	miss
8	$[1000_2]$	miss
0	$[0000_2]$	miss

Cache memory

v	tag	data
1	0	M[0-1]
1	0	M[6-7]

Miss Ratio: 4/5 = 0.8

Hit Ratio: 1- Miss Ratio = 0.2

Example 1

Consider a direct mapped cache of size 512 KB with block size 1 KB. There are 7 bits in the tag. Find the Size of main memory

Given:

Cache memory size = 512 KB

Block size = Line size = 1 KB = 2^10, i.e offset is 10 bits

Number of bits in tag = 7 bits

Total number of blocks= 512 kb / 1 kb = 512 blocks or line = 2^9 . i.e 9 bits

The physical address = TAG+ Line number + offset

= 7 bits + 9 bits + 10 bits

= 26 bits

Size of memory is 2^26= 67,108 bytes

Example 2

A computer system uses 16-bit memory addresses. It has a 2K-byte cache organized in a direct-mapped manner with 64 bytes per cache block. Assume that the size of each memory word is 1 byte.

(a) Calculate the number of bits in each of the Tag, Block, and Word fields of the memory address.

(b) When a program is executed, the processor reads data sequentially from the following word addresses: **128, 144, 2176, 2180, 128, 2176**

All the above addresses are shown in decimal values. Assume that the cache is initially empty.

For each of the above addresses, indicate whether the cache access will result in a hit or a miss.

Example 2

Block size = 64 bytes = 2^6 bytes
Therefore, **Number of bits in the *Word* field = 6**
Cache size = 2K-byte = 2^{11} bytes
Number of cache blocks = Cache size / Block size = $2^{11}/2^6 = 2^5$
Therefore, **Number of bits in the *Block* field = 5**
Total number of address bits = 16
Therefore, **Number of bits in the *Tag* field = 16 - 6 - 5 = 5**
For a given 16-bit address, the 5 most significant bits, represent the *Tag*, the next 5 bits represent the *Block*, and the 6 least significant bits represent the *Word*.

Example 2

Access # 1: The cache is initially empty. Therefore, all the cache blocks are invalid

Address = $(128)_{10}$ = $(0000000010000000)_2$
For this address, *Tag* = 00000, *Block* = 00010, *Word* = 000000

Since the cache is empty before this access, this will be a cache **miss**
After this access, **Tag field for cache block 00010 is set to 00000**

Access # 2: Address = $(144)_{10}$ = $(0000000010010000)_2$

For this address, *Tag* = **00000**, *Block* = **00010**, *Word* = 010000

Since tag field for cache block **00010** is **00000** before this access, this will be a cache **hit** (because address tag = block tag)

Example 2

Access # 3: Address = $(2176)_{10}$ = $(\mathbf{00001}\mathbf{00010}00000)_2$
For this address, *Tag* = **00001**, *Block* = **00010**, *Word* = 000000
Since tag field for cache block **00010** is **00000** before this access, this will be a cache **miss**
(address tag ≠ block tag)
After this access, **Tag field for cache block 00010 is set to 00001**

Access # 4:
Address = $(2180)_{10}$ = $(\mathbf{00001}00010\mathbf{00}100)_2$
For this address, *Tag* = **00001**, *Block* = **00010**, *Word* = 000100
Since tag field for cache block **00010** is **00001** before this access, this will be a cache **hit** (address tag = block tag)

Example 2

Access # 5:

Address = $(128)_{10}$ = $(\mathbf{00000}\mathbf{00010}00000)_2$

For this address, *Tag* = **00000**, *Block* = **00010**, *Word* = 000000

Since tag field for cache block **00010** is **00001** before this access, this will be a cache **miss**

(address tag ≠ block tag)

After this access, **Tag field for cache block 00010 is set to 00000**

Access # 6:

Address = $(2176)_{10}$ = $(\mathbf{00001}\mathbf{00010}00000)_2$

For this address, *Tag* = **00001**, *Block* = **00010**, *Word* = 000000

Since tag field for cache block **00010** is **00000** before this access, this will be a cache **miss**

(address tag ≠ block tag)

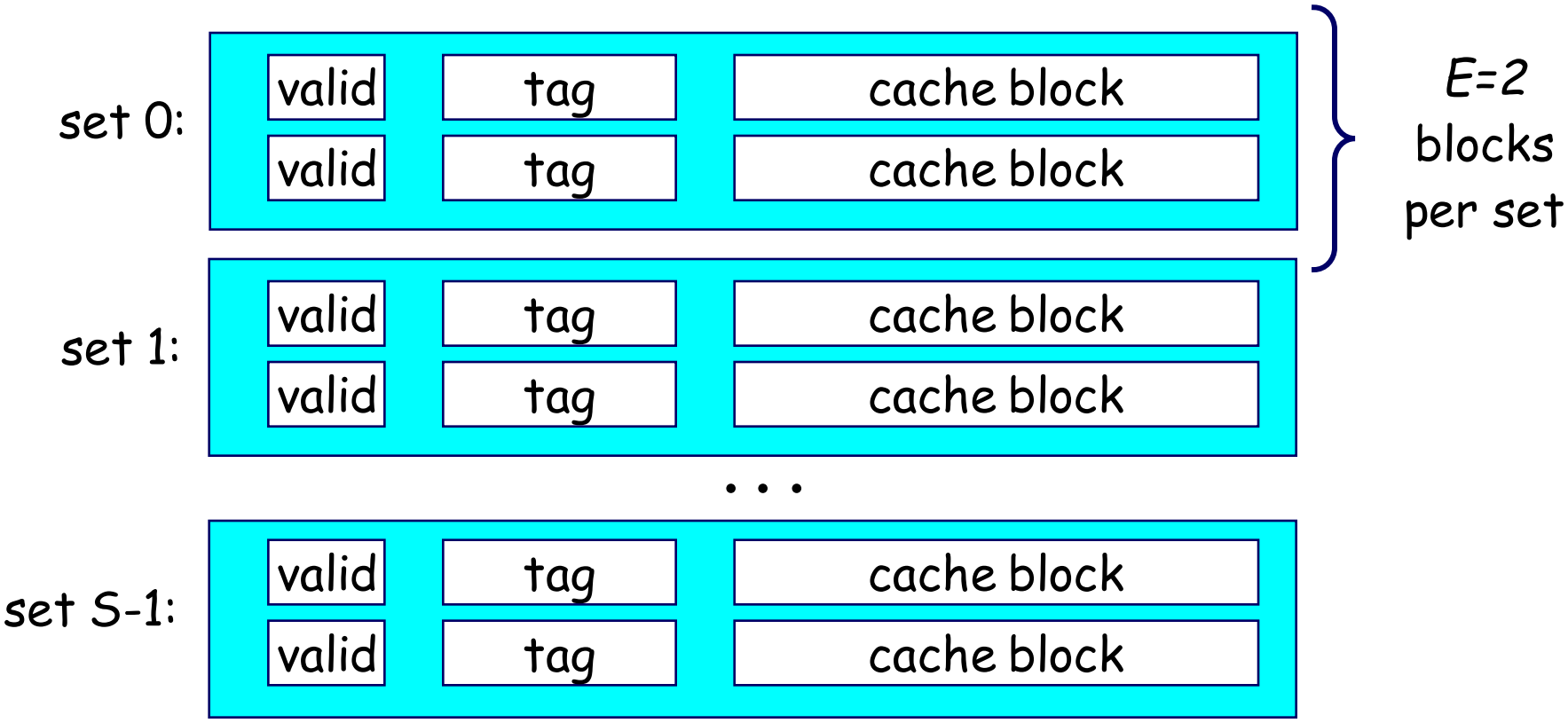
After this access, **Tag field for cache block 00010 is set to 00001**

Cache hit rate = Number of hits / Number of accesses = 2/6

= 0.333

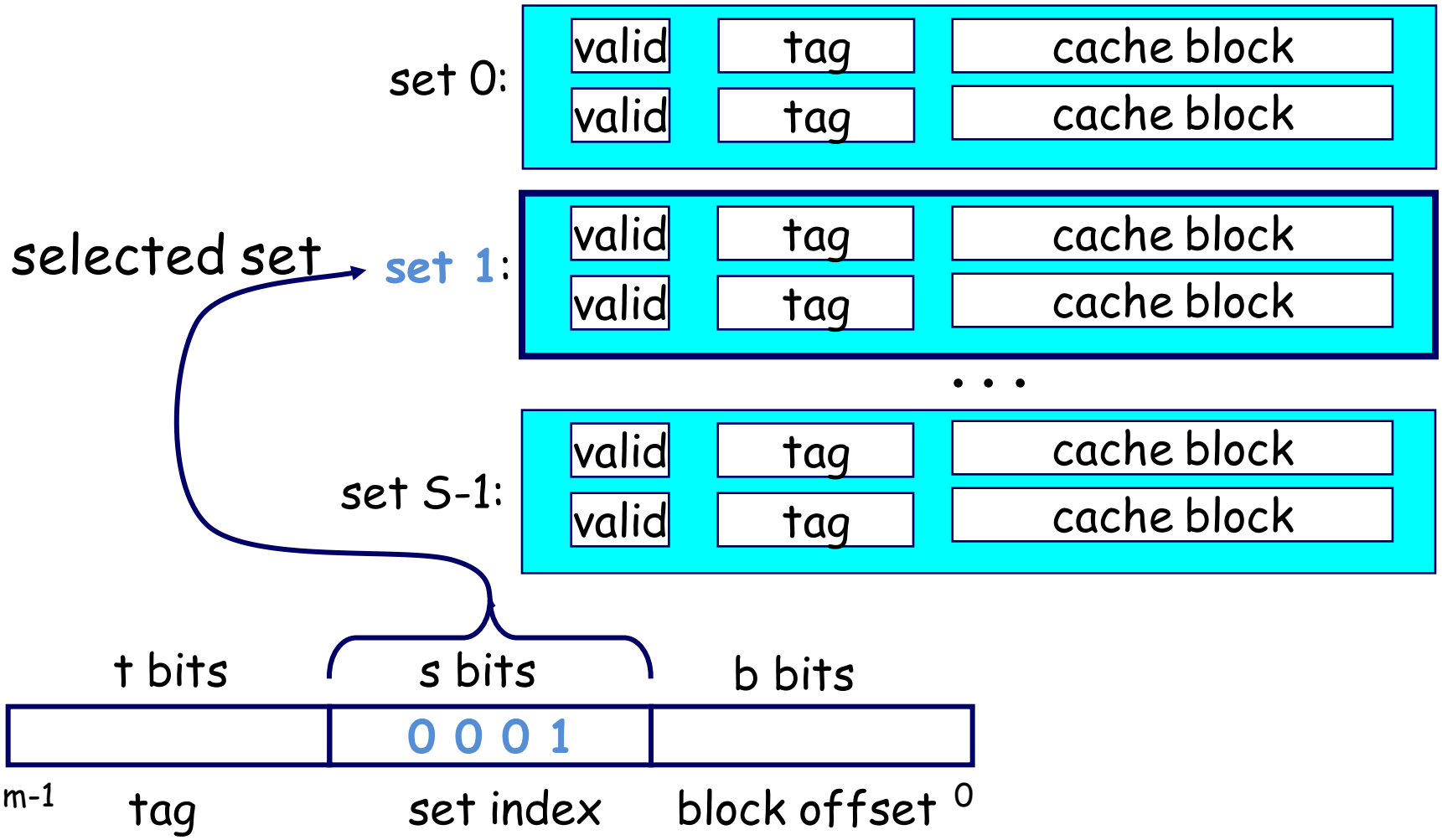
Example: Set Associative mapping

Characterized by more than one block per set



E-way associative cache

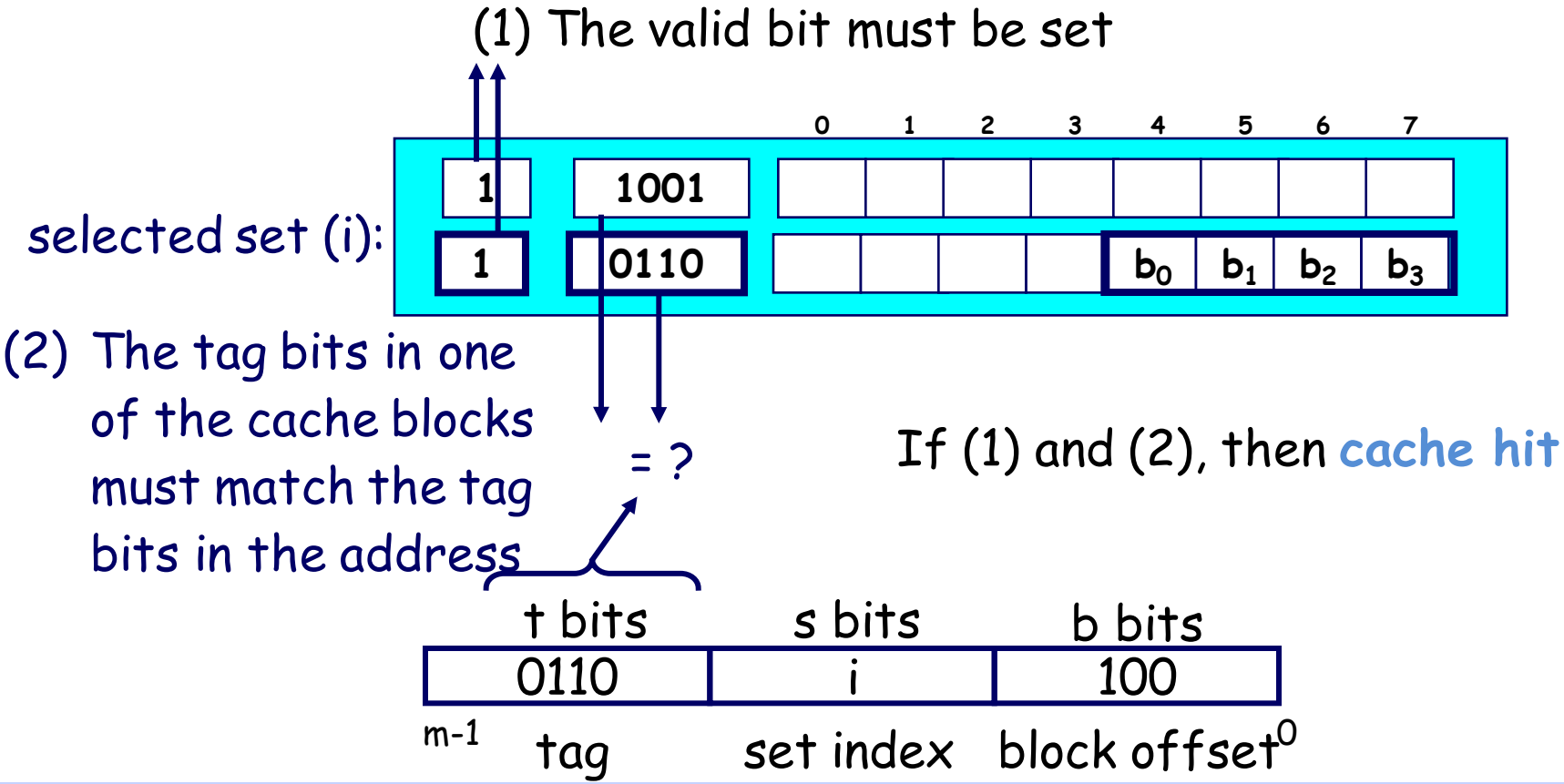
Accessing Set Associative Caches – Set Selection



Accessing Set Associative Caches

Block matching and word selection:

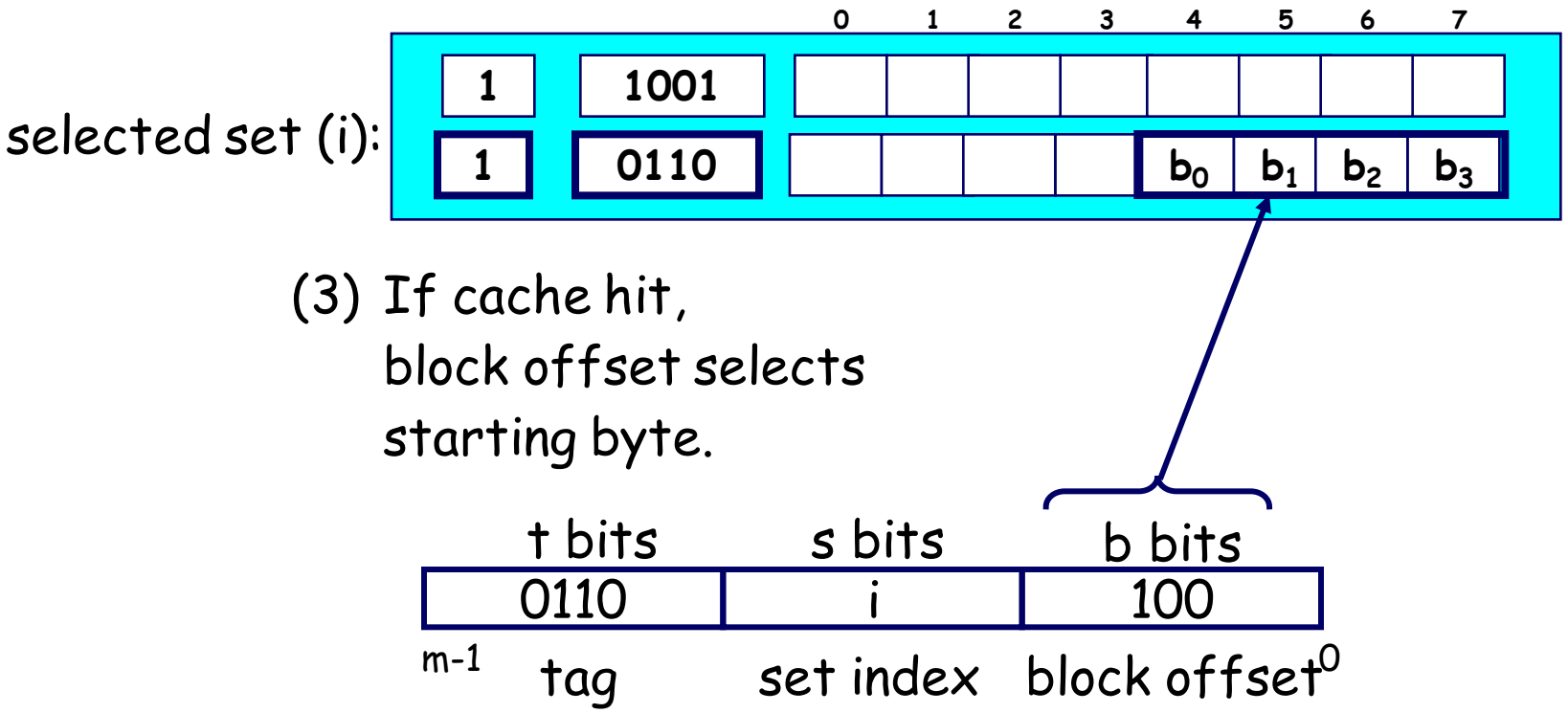
– must compare the tag in each valid block in the selected set.



Accessing Set Associative Caches

Block matching and word selection:

– must compare the tag in each valid block in the selected set.



2 Way Associative Cache Simulation

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

M=16 byte addresses, B=2 bytes/block, S=2 sets, E=2 entry/set

Main memory addresses

0	$[0000_2]$
1	$[0001_2]$
7	$[0111_2]$
8	$[1000_2]$
0	$[0000_2]$

miss
hit
miss
miss
hit

Cache memory

	v	tag	data
Set 0	1	00	M[0-1]
	1	10	M[8-9]
Set 1	1	01	M[6-7]
	0		

Miss Ratio: 3/5 = 0.6

Hit Ratio: 1- Miss Ratio = 0.4

Example 1

Consider a 4 way set associative cache with block size 4 KB. The size of main memory is 16 GB and there are 10 bits in the tag. Find the Size of cache memory

Given-

- Block size = Line size = 4 KB = 2^12 i.e. offset = 12 bits
- Size of main memory = 16 GB = 2^34 bytes i.e Physical address = 34 bits
- Number of bits in tag = 10 bits

- Set number = 34- (10+12)=12 bits
- Number of blocks or lines = 2^12 x 4 = 2^14 lines
- Size of Cache= 2^14 * 4 kb = 2^16 kb= 64 MB

Example 2

A computer system uses 16-bit memory addresses. It has a 2K-byte cache organized in a 2-way set associative manner with 64 bytes per cache block. Assume that the size of each memory word is 1 byte.

(a) Calculate the number of bits in each of the Tag, Block, and Word fields of the memory address.

(b) When a program is executed, the processor reads data sequentially from the following word addresses: **128, 144, 2176, 2180, 128, 2176**

All the above addresses are shown in decimal values. Assume that the cache is initially empty.

For each of the above addresses, indicate whether the cache access will result in a hit or a miss.

Example 2

Block size = 64 bytes = 2^6 bytes

Therefore, **Number of bits in the *Word* field = 6**

Cache size = 2K-byte = 2^{11} bytes

Number of cache blocks = Cache size / (Block size x Number of blocks per set)

= $2^{11}/(2^6 \times 2) = 2^4$

Therefore, **Number of bits in the *set* field = 4**

Total number of address bits = 16

Therefore, **Number of bits in the *Tag* field = 16 – (6+4) = 6**

For a given 16-bit address, the 6 most significant bits, represent the *Tag*, the next 4 bits represent the *Block*, and the 6 least significant bits represent the *Word*.

Example 2

Access # 1:

Address = $(128)_{10}$ = $(0000000010000000)_2$

For this address, *Tag* = **000000**, *Set* = **0010**, *Word* = 000000

Since the cache is empty before this access, this will be a cache **miss**

After this access, **Tag field for the first block in set 0010 is set to 000000**

Access # 2:

Address = $(144)_{10}$ = $(0000000010010000)_2$

For this address, *Tag* = **000000**, *Set* = **0010**, *Word* = 010000

The tag field for this address matches the tag field for the first block in set 0010.

Therefore, this access will be a cache **hit**.

Example 2

Access # 3:

Address = $(2176)_{10}$ = $(000010\,0010\,000000)_2$

For this address, *Tag* = **000010**, *Set*= **0010**, *Word* = 000000

The tag field for this address does not match the tag field for the first block in set 0010.

The second block in set 0010 is empty. Therefore, this access will be a cache **miss**.

After this access, **Tag field for the second block in set 0010 is set to 000010**

Access # 4:

Address = $(2180)_{10}$ = $(000010\,0010\,000100)_2$

For this address, *Tag* = **000010**, *Set* = **0010**, *Word* = 000100

The tag field for this address matches the tag field for the second block in set 0010.

Therefore, this access will be a cache **hit**.

Example 2

Access # 5:

Address = $(128)_{10}$ = $(0000000010000000)_2$

For this address, *Tag* = **000000**, *Set* = **0010**, *Word* = 000000

The tag field for this address matches the tag field for the first block in set 0010.

Therefore, this access will be a cache **hit**.

Access # 6:

Address = $(2176)_{10}$ = $(0000100010000000)_2$

For this address, *Tag* = **000010**, *Set* = **0010**, *Word* = 000000

The tag field for this address matches the tag field for the second block in set 0010.

Therefore, this access will be a cache **hit**.

Example 2

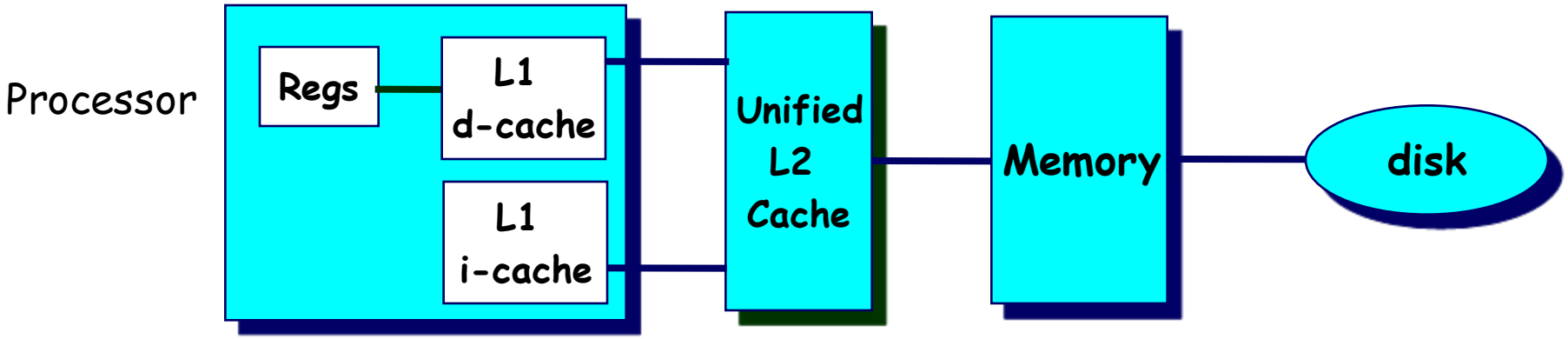
Cache hit rate = Number of hits / Number of accesses = 4/6 = 0.666

Compare with direct mapping

Cache hit rate = Number of hits / Number of accesses
= 2/6
= 0.333

Sidebar: Multi-Level Caches

Options: separate data and instruction caches, or a unified cache



size:	200 B	8-64 KB	1-4MB SRAM	128 MB DRAM	30 GB
speed:	3 ns	3 ns	6 ns	60 ns	8 ms
$/Mbyte:			$100/MB	$1.50/MB	$0.05/MB
line size:	8 B	32 B	32 B	8 KB	

larger, slower, cheaper

Q & A

CACHE MEMORY MAPPING