M6 – Memory Hierarchy

Module Outline

- CPU – Memory interaction
- Organization of memory modules
- Cache memory – Mapping and replacement policies.

Principle of Locality

- Programs access a small proportion of their address space at any time
- Temporal locality
  - Items accessed recently are likely to be accessed again soon
  - e. g., instructions in a loop, induction variables
- Spatial locality
  - Items near those accessed recently are likely to be accessed soon
  - E.g., sequential instruction access, array data

Question

- Your program contains 1000 instructions. 35% are loads and stores. How many references are made to the cache?

Question

- Your program contains 1000 instructions. 35% are loads and stores. How many references are made to the cache?
- 1000I + 350D

Cache Design Hints from LoR

- Temporal locality
  -
- Spatial locality
  -

Cache Design Hints from LoR

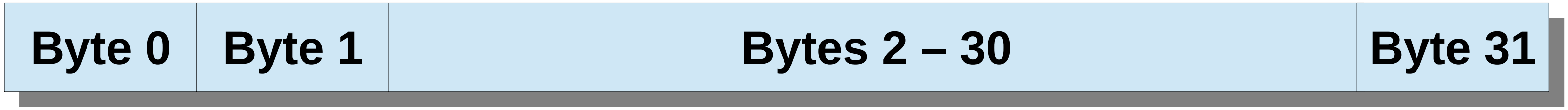
- Temporal locality
  - Keep the data brought into the cache as long as possible
- Spatial locality
  - If address 0x1000 is accessed, 0x1004, 0x1008, ... will also be accessed (almost certainly).

Cache Design Hints from LoR

- Temporal locality
  - Keep the data brought into the cache as long as possible
- Spatial locality
  - If address 0x1000 is accessed, 0x1004, 0x1008, ... will also be accessed (almost certainly).
  - Good idea to bring in a **block** of items from Main memory to cache – Cache Block (4B – 64B)

Block

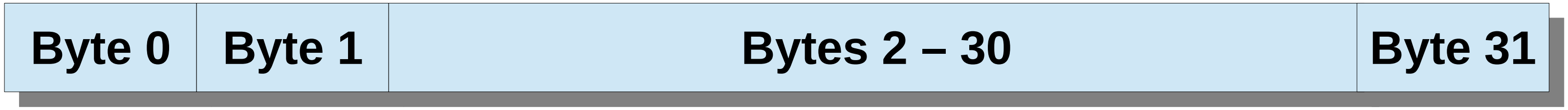
- 32B Block (Byte 0, ..., Byte 31)

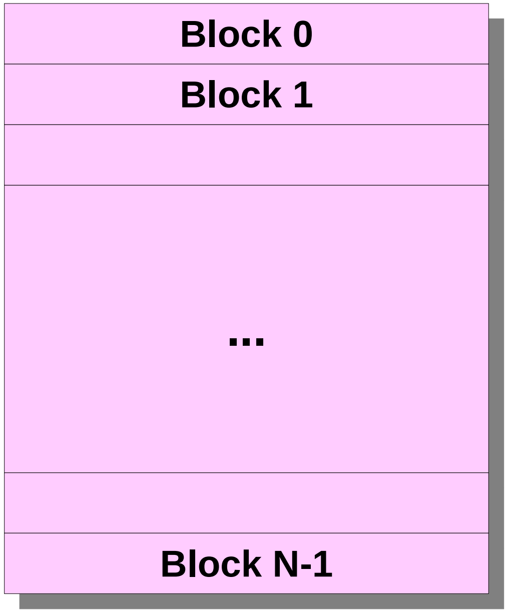


Byte 0	Byte 1	Bytes 2 – 30	Byte 31

Block

- 32B Block (Byte 0, ..., Byte 31)

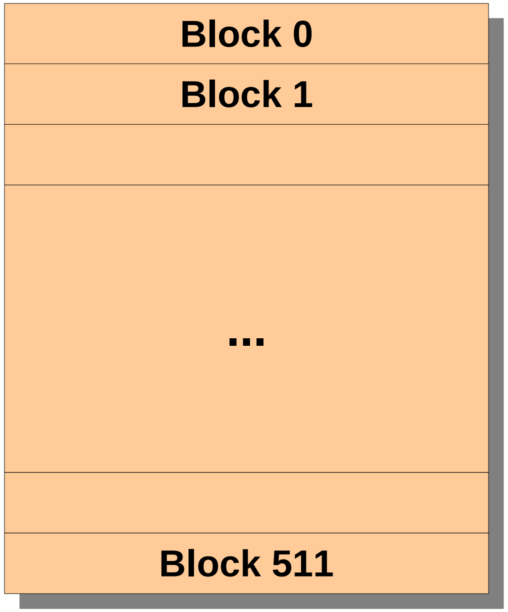




Main Memory houses N Blocks

Block

- Consider a 32KB cache containing 512 blocks. What is the block size?



Block 0

Block 1

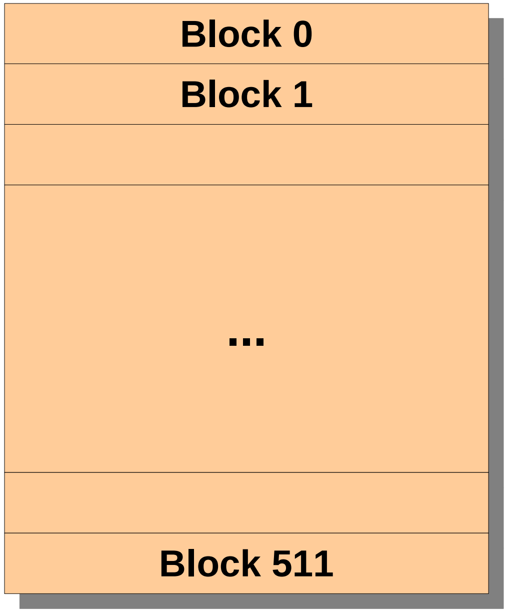
...

Block 511

Block

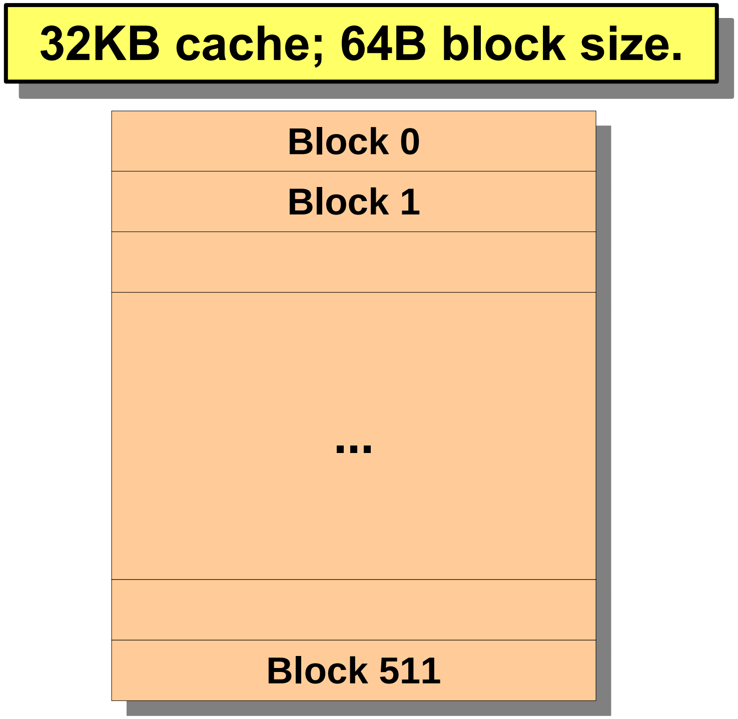
- Consider a 32KB cache containing 512 blocks. What is the block size?

32KB cache; 64B block size.



Block

- Consider a 32KB cache containing 512 blocks. What is the block size?
- How many blocks of the above size fit in a 8GB main memory?



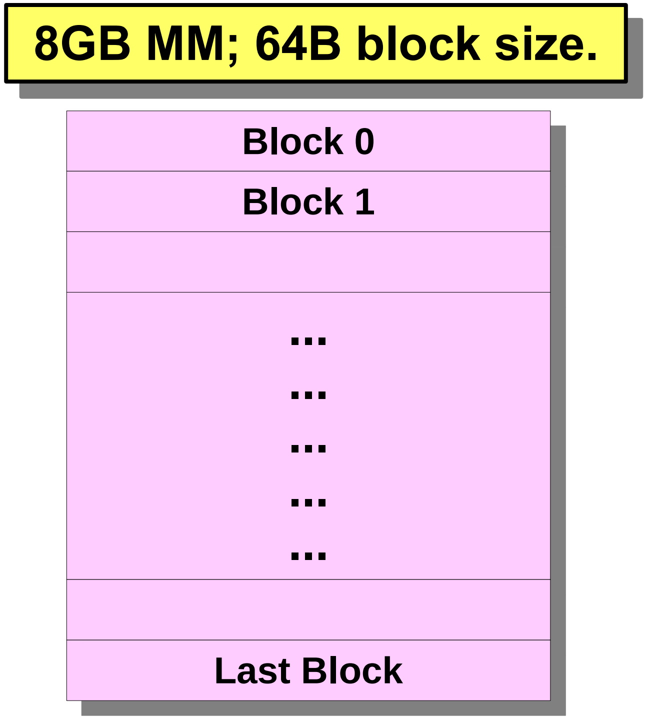
32KB cache; 64B block size.

Block 0

Block 1

...

Block 511



8GB MM; 64B block size.

Block 0

Block 1

...

...

...

...

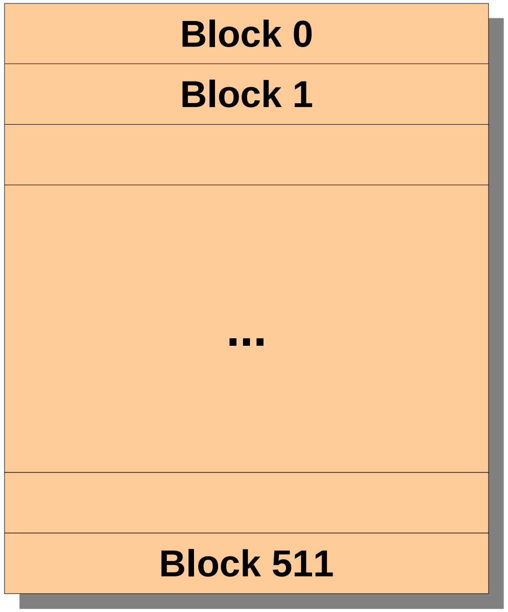
...

Last Block

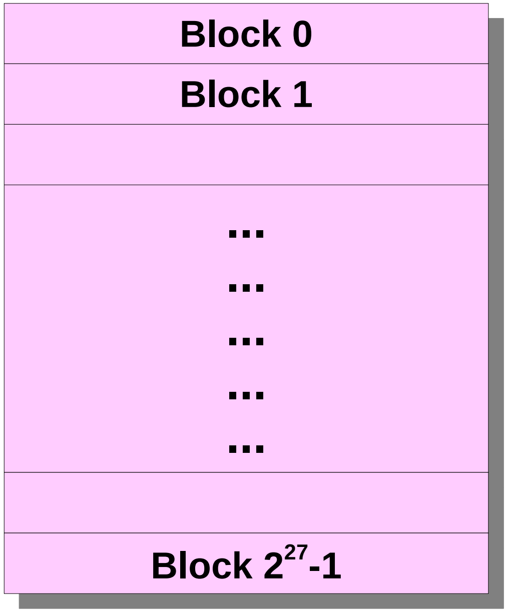
Block

- Consider a 32KB cache containing 512 blocks. What is the block size?
- How many blocks of the above size fit in a 8GB main memory?

32KB cache; 64B block size.



8GB MM; 64B block size.



Block

- Consider a 1024B Main Memory with 32B blocks
  - What is the size of the block address?

Block

- Consider a 1024B Main Memory with 32B blocks
  - What is the size of the block address?
- Which block does Byte 324 belong to?

Blocks and Block Addresses

Byte Address	Block Address	Address in Binary
0	0	00 0000 0000

Blocks and Block Addresses

Byte Address	Block Address	Address in Binary
0	0	00 0000 0000
1	0	00 0000 0001

Blocks and Block Addresses

Byte Address	Block Address	Address in Binary
0	0	00 0000 0000
1	0	00 0000 0001
...	...	...
31	0	00 0001 1111

Blocks and Block Addresses

Byte Address	Block Address	Address in Binary
0	0	00 0000 0000
1	0	00 0000 0001
...	...	...
31	0	00 0001 1111
32	1	00 0010 0000
...	...	...
63	1	00 0011 1111

Blocks and Block Addresses

Byte Address	Block Address	Address in Binary
0	0	00 0000 0000
1	0	00 0000 0001
...	...	...
31	0	00 0001 1111
32	1	00 0010 0000
...	...	...
63	1	00 0011 1111
64	2	00 0100 0000
...	...	...

Blocks and Block Addresses

Byte Address	Block Address	Address in Binary
0	0	00 0000 0000
1	0	00 0000 0001
...	...	...
31	0	00 0001 1111
32	1	00 0010 0000
...	...	...
63	1	00 0011 1111
64	2	00 0100 0000
...	...	...
96	3	00 0110 0000
...	...	...
...	...	...
...	...	...

Blocks and Block Address

- 32B per block.

Blocks and Block Address

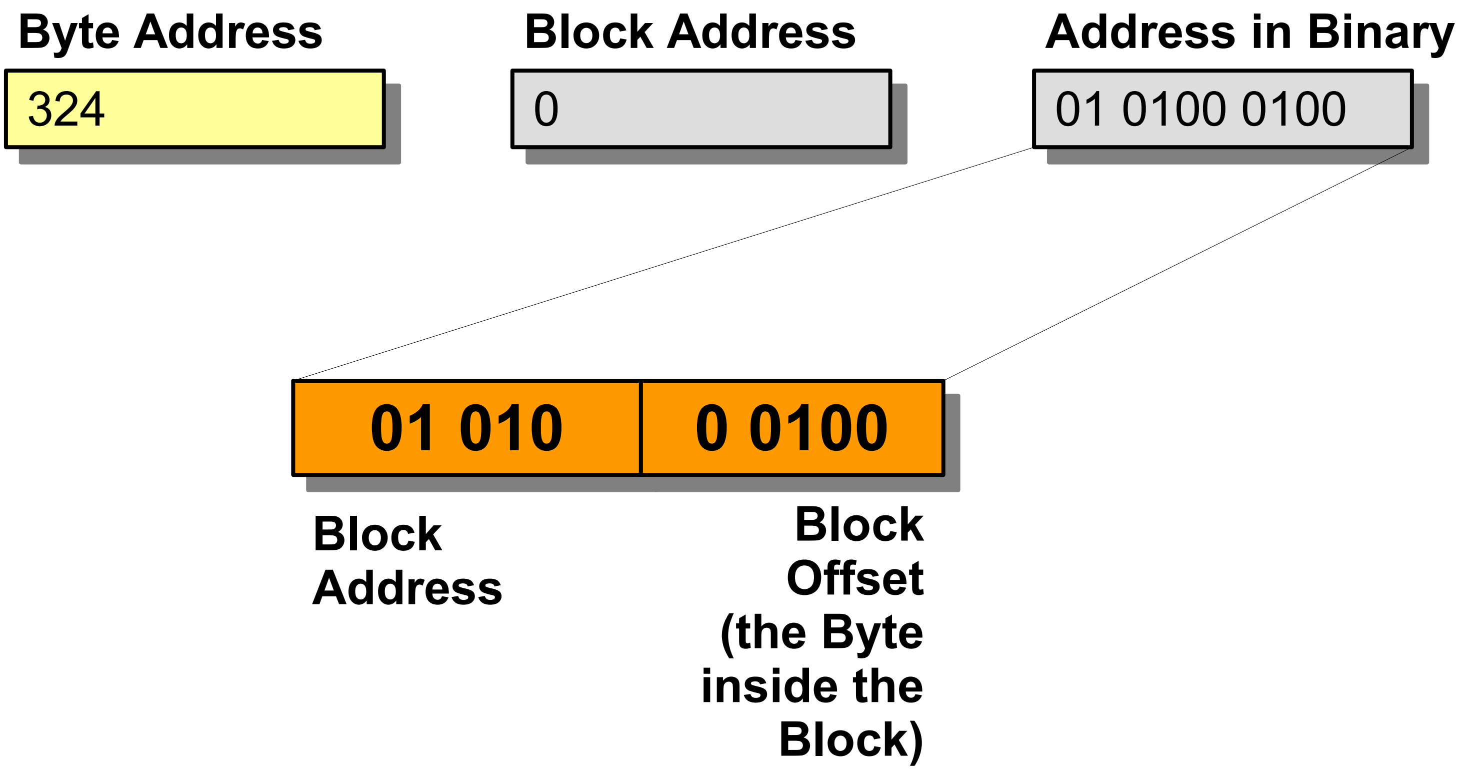
- 32B per block.
- Block 0 contains Bytes 0 – 31; Block 1 contains Bytes 32 – 63; ...

Blocks and Block Address

- 32B per block.
- Block 0 contains Bytes 0 – 31; Block 1 contains Bytes 32 – 63; ...
- Byte B will be housed in:

$$Block\ Number = \frac{Byte\ Address}{Block\ S\ ize}$$

Blocks and Block Addresses



Byte Address

324

Block Address

0

Address in Binary

01 0100 0100

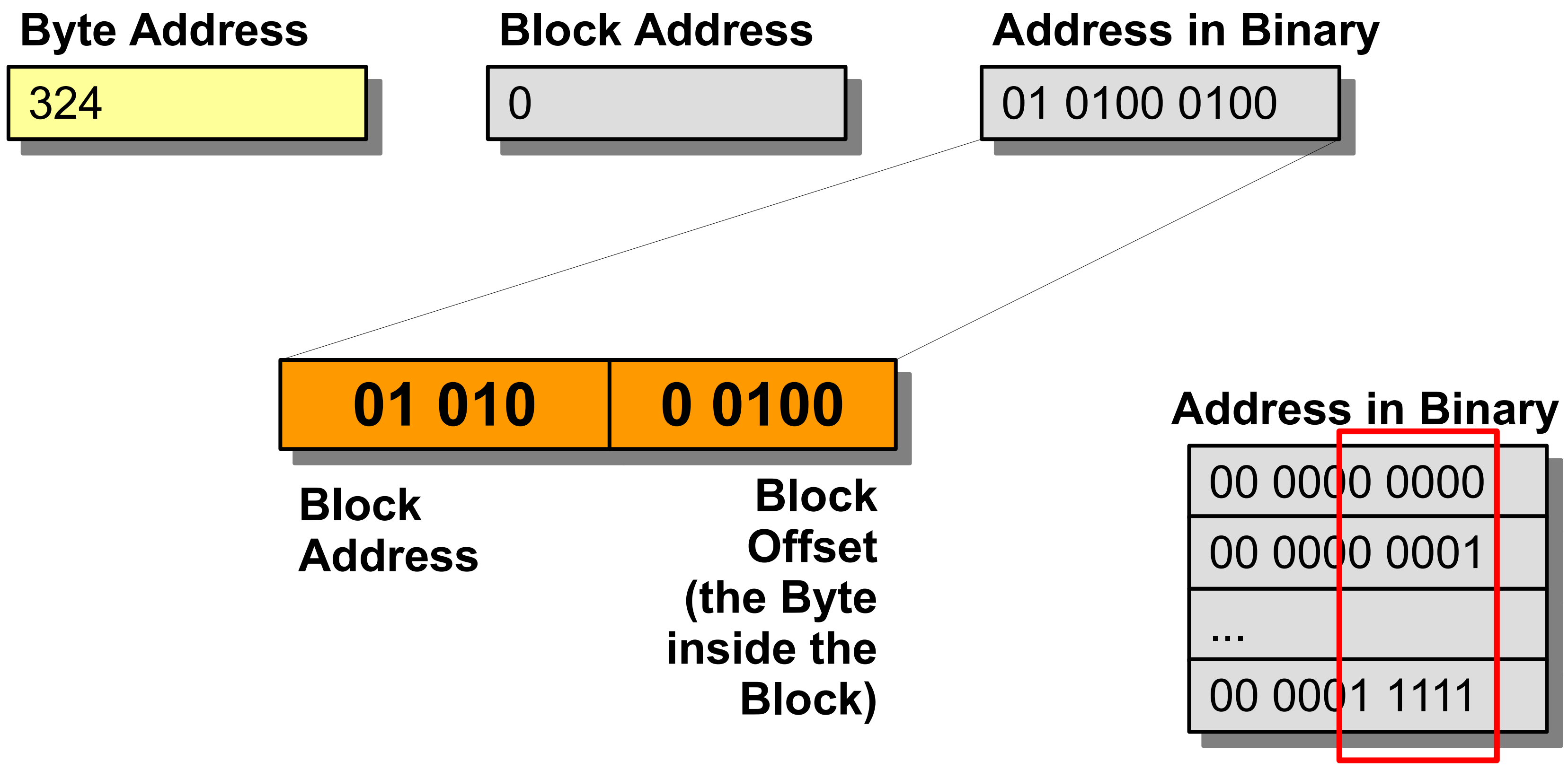
01 010

0 0100

Block Address

Block Offset (the Byte inside the Block)

Blocks and Block Addresses

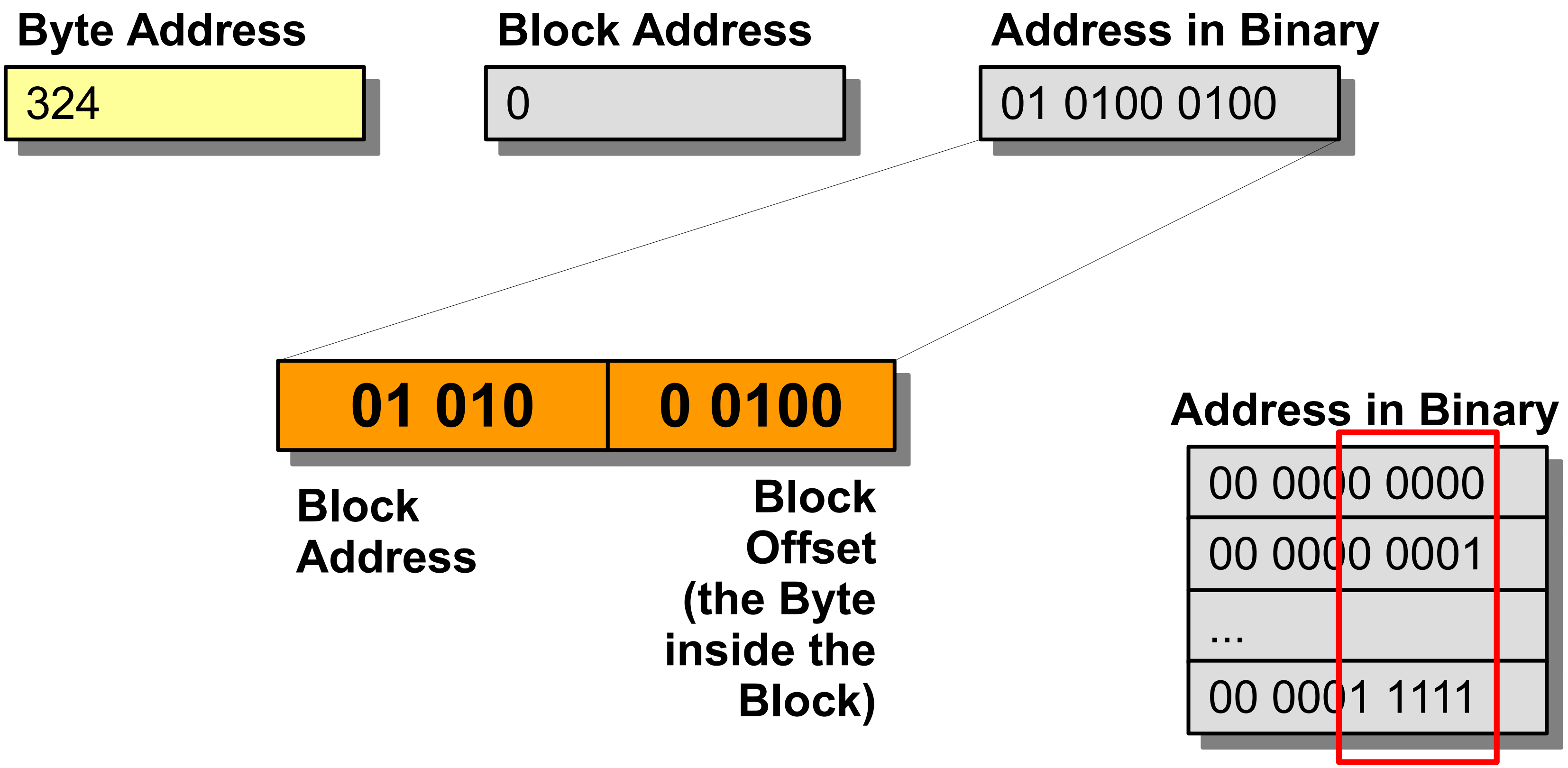


Byte Address	Block Address	Address in Binary
324	0	01 0100 0100

01 010	0 0100
Block Address	Block Offset (the Byte inside the Block)

Address in Binary
00 0000 0000
00 0000 0001
...
00 0001 1111

Blocks and Block Addresses



Byte Address

324

Block Address

0

Address in Binary

01 0100 0100

01 010 | **0 0100**

Block Address

Block Offset (the Byte inside the Block)

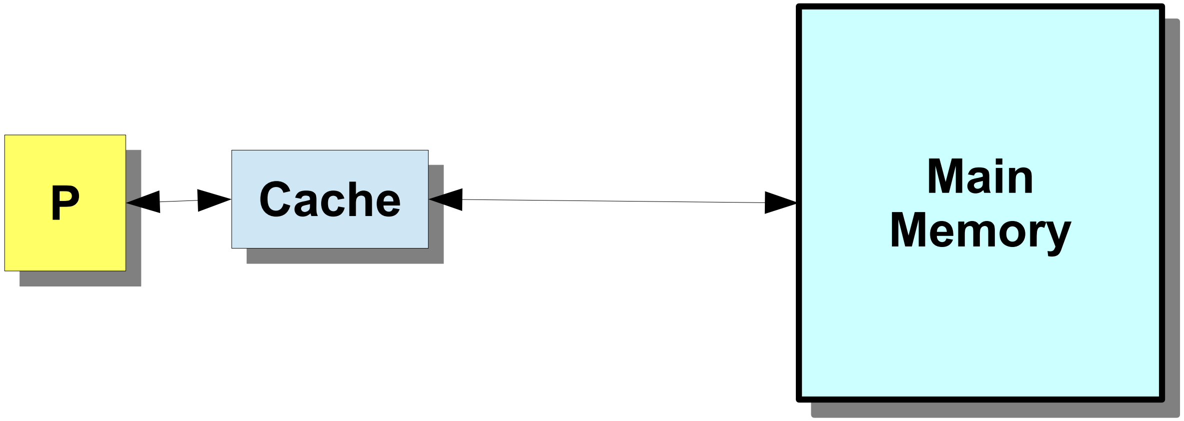
Address in Binary
00 0000 0000
00 0000 0001
…
00 0001 1111

32B block.
5b Block Offset

Cache Design

Assume: Cache is empty

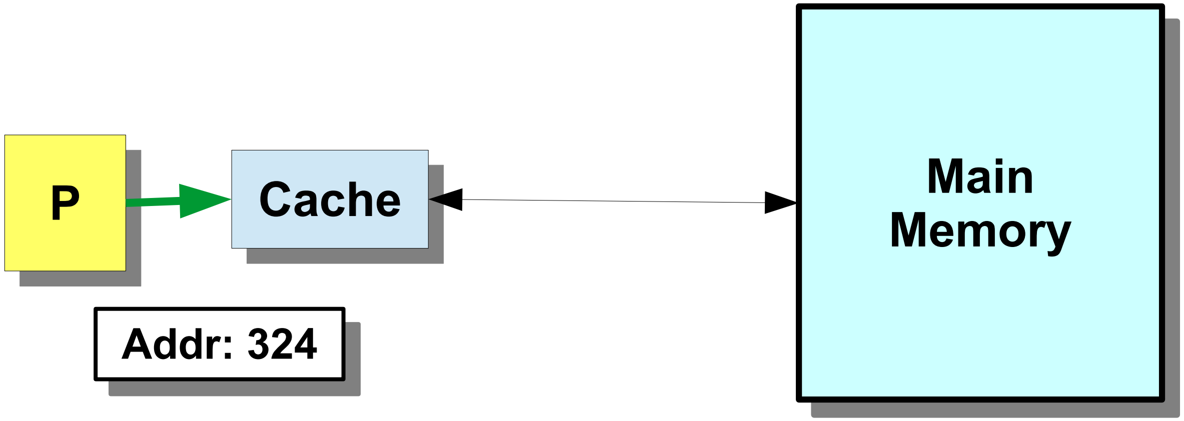
Processor asks for Byte 324



Cache Design

Assume: Cache is empty

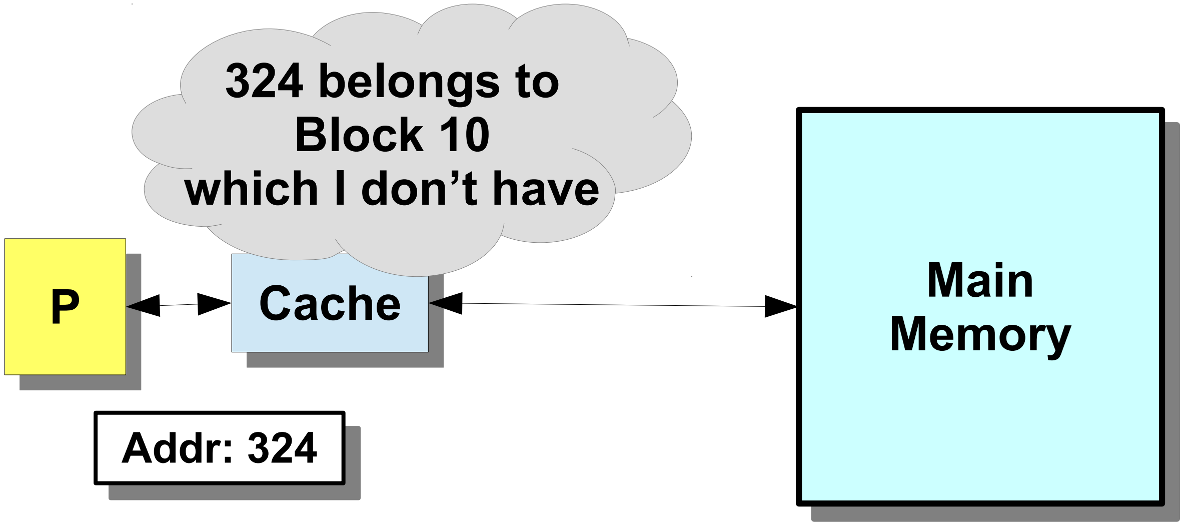
Processor asks for Byte 324



Cache Design

Assume: Cache is empty

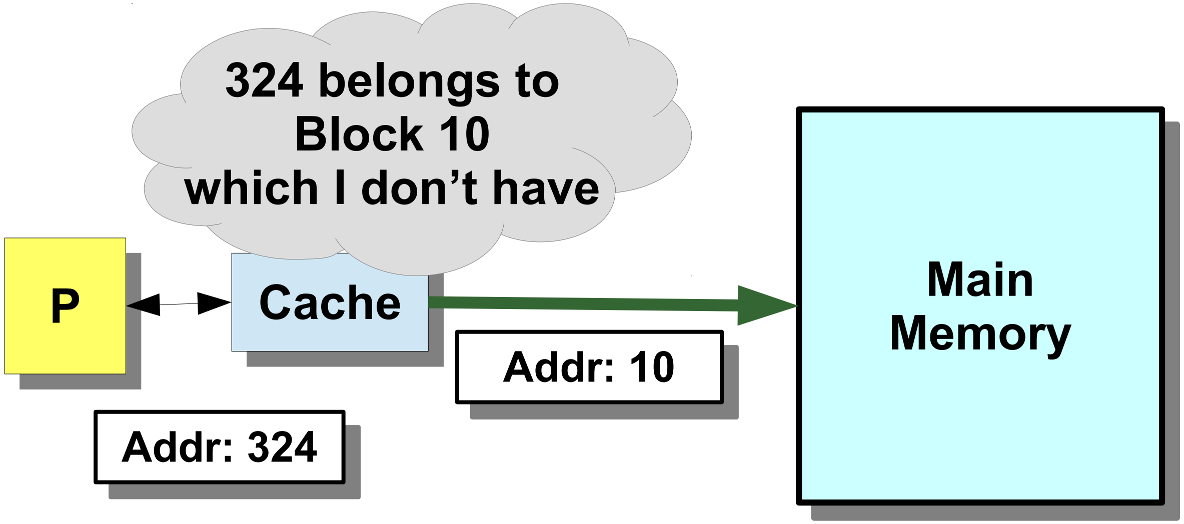
Processor asks for Byte 324



Cache Design

Assume: Cache is empty

Processor asks for Byte 324

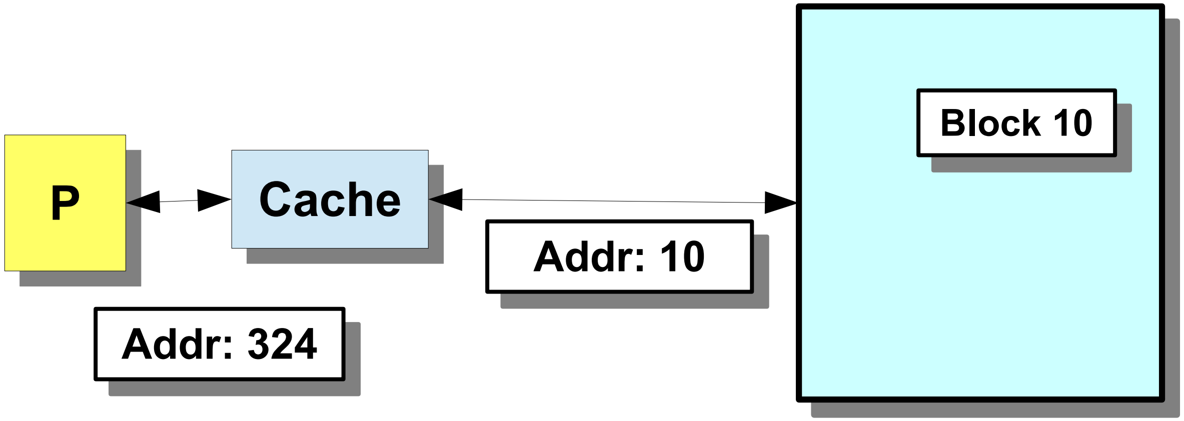


324 belongs to Block 10 which I don't have

Cache Design

Assume: Cache is empty

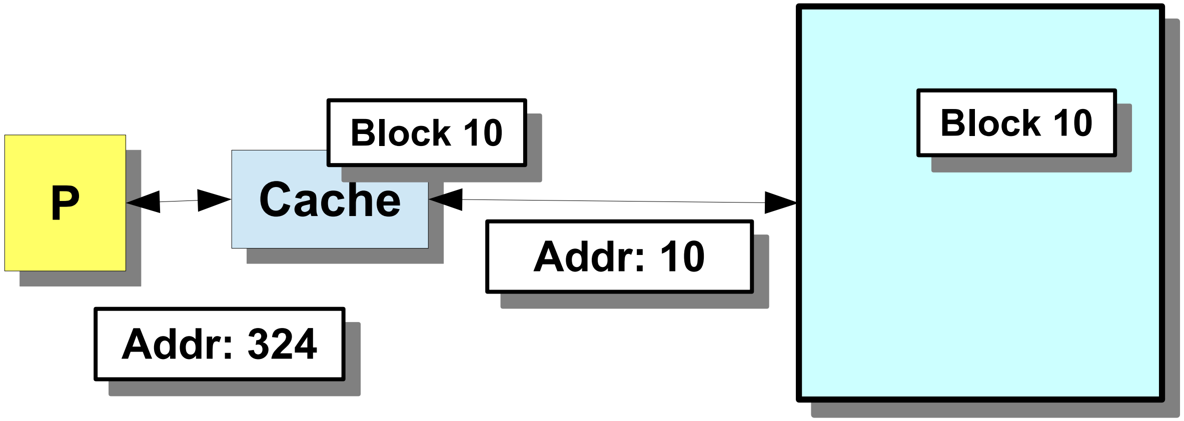
Processor asks for Byte 324



Cache Design

Assume: Cache is empty

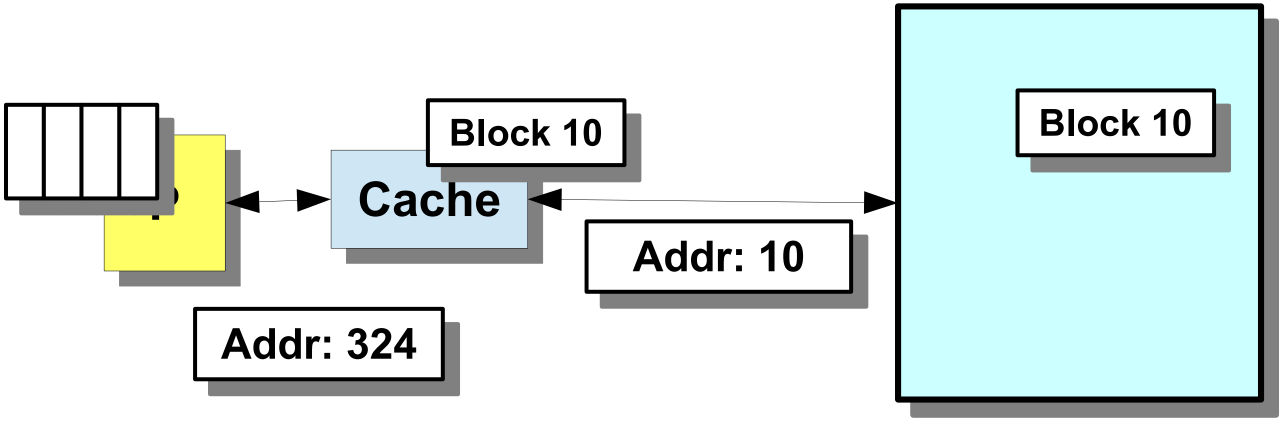
Processor asks for Byte 324



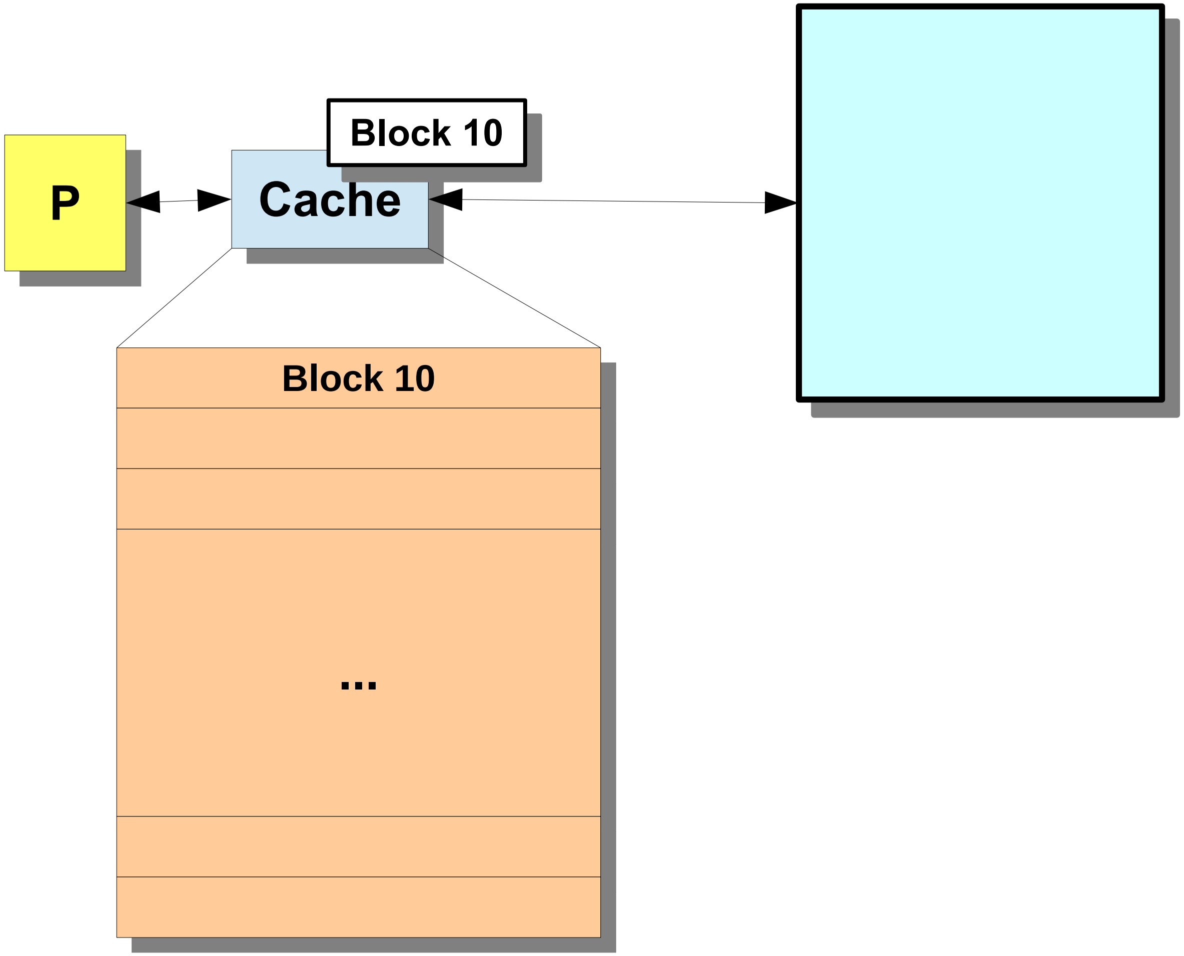
Cache Design

Assume: Cache is empty

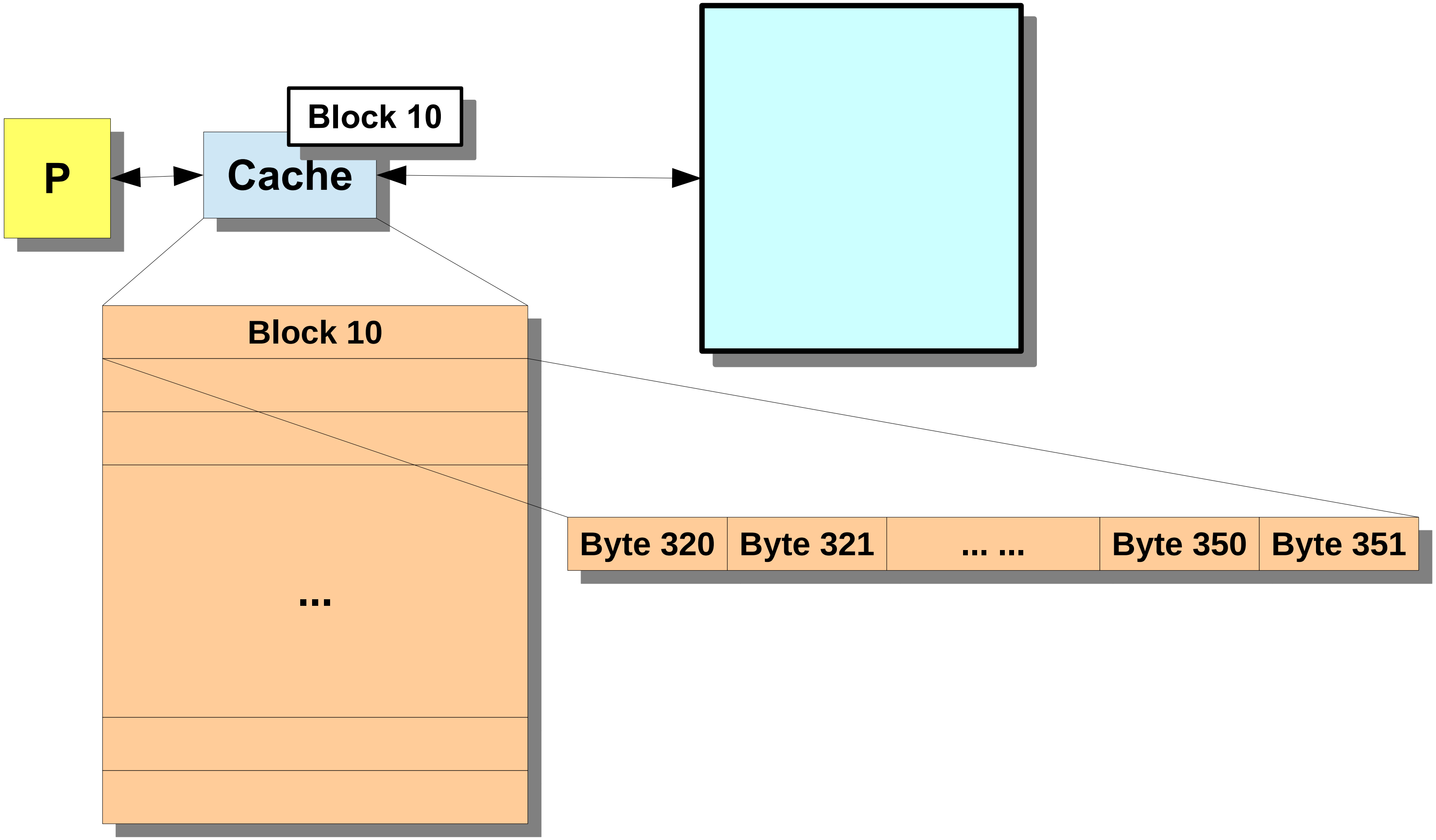
Processor asks for Byte 324



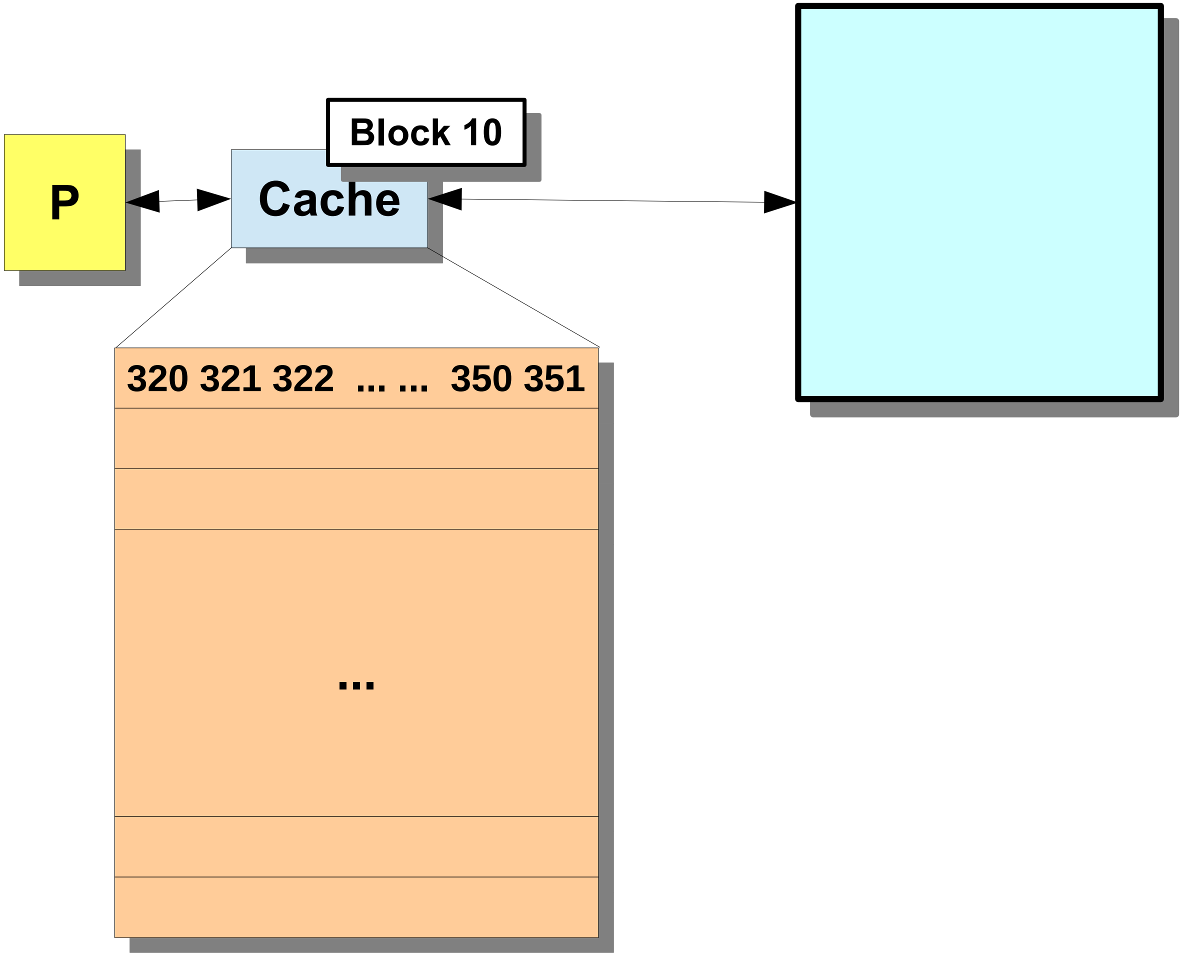
Cache Design



Cache Design

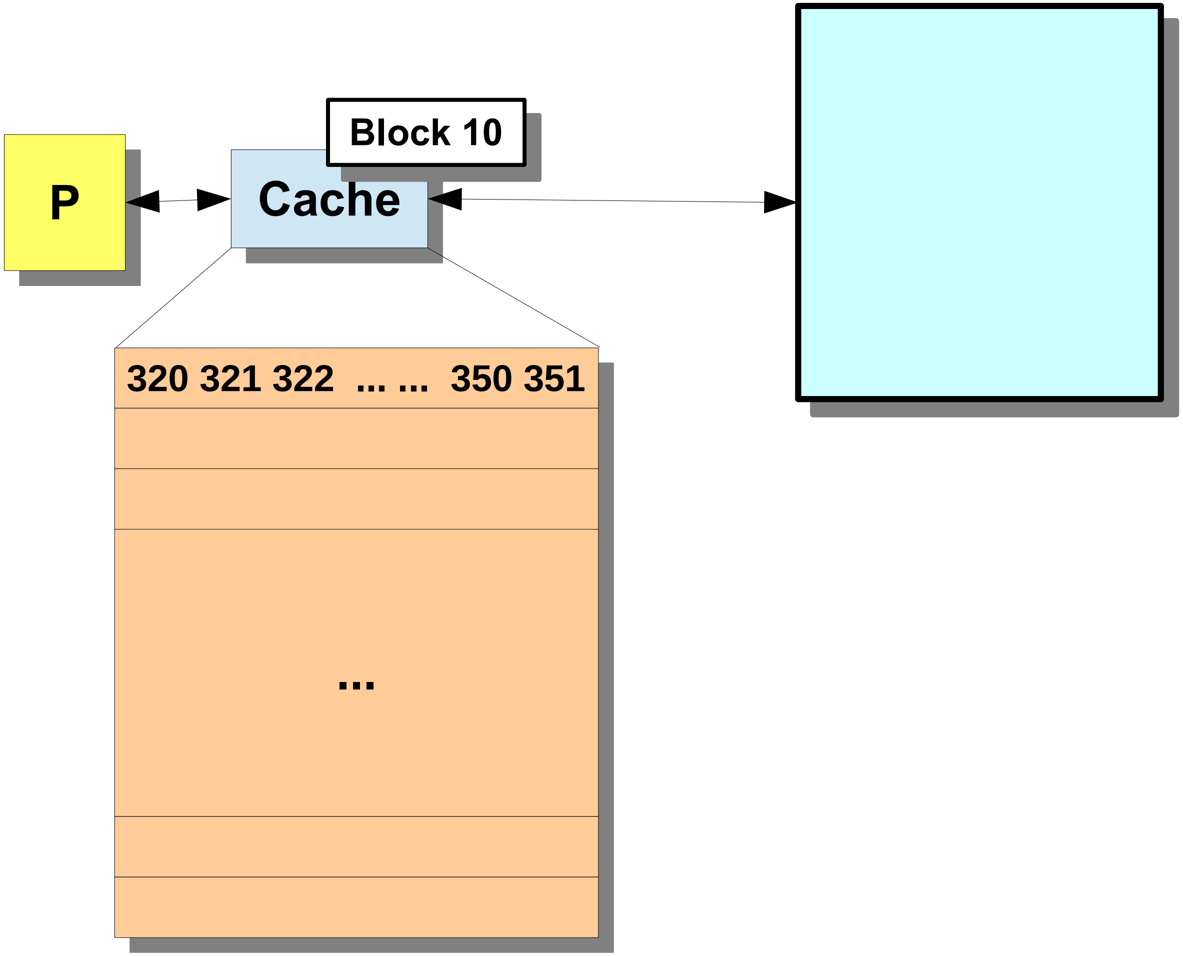


Cache Design



Cache Design

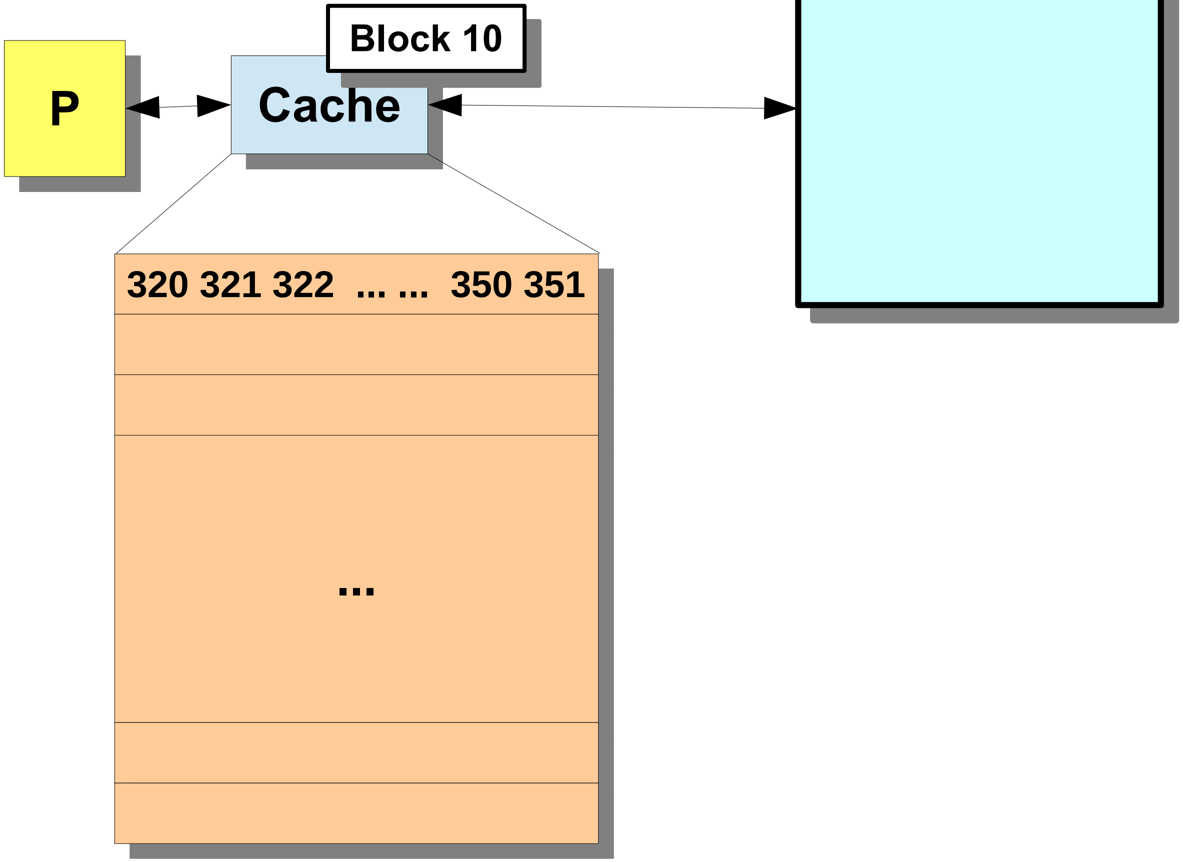
Processor asks for Byte 328



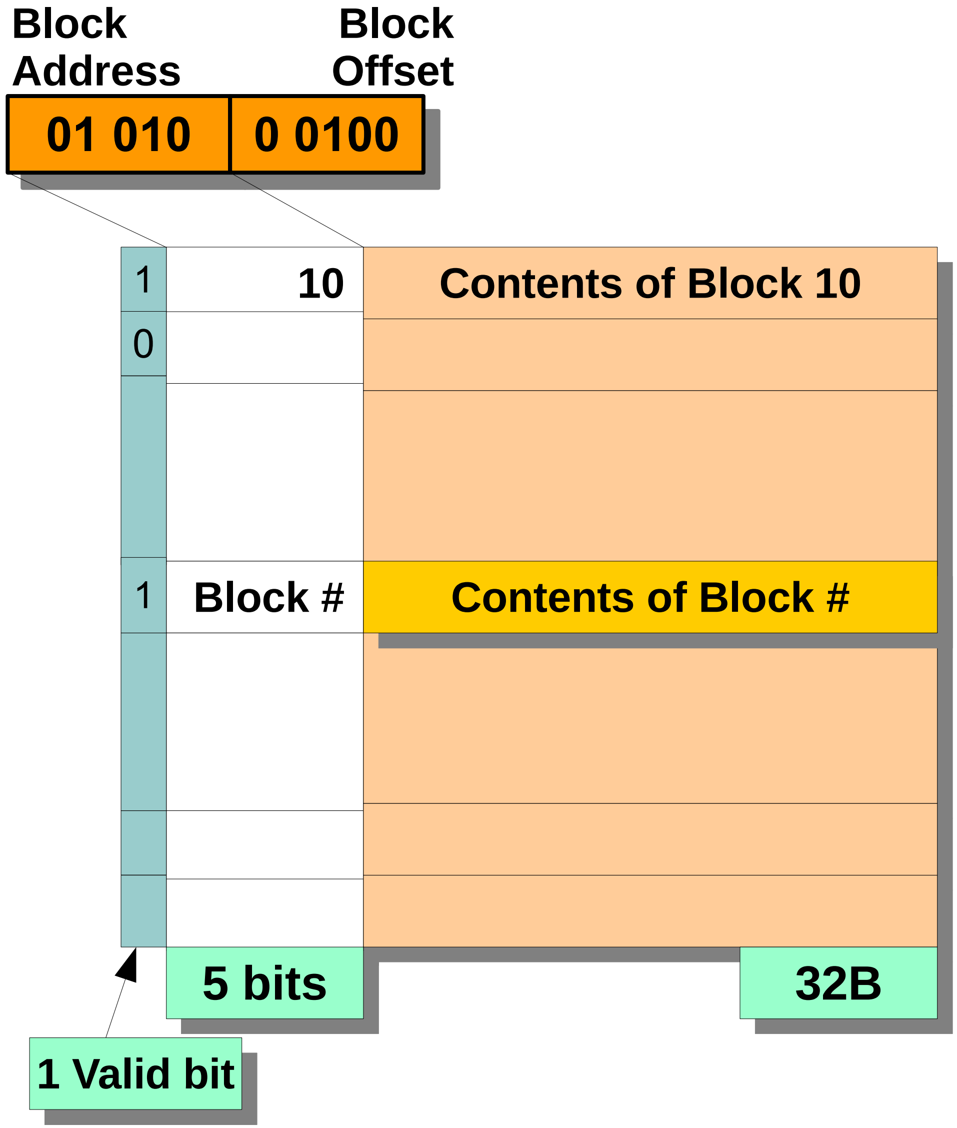
Cache Design

Processor asks for Byte 328

How does the cache know it already has contents of 328?



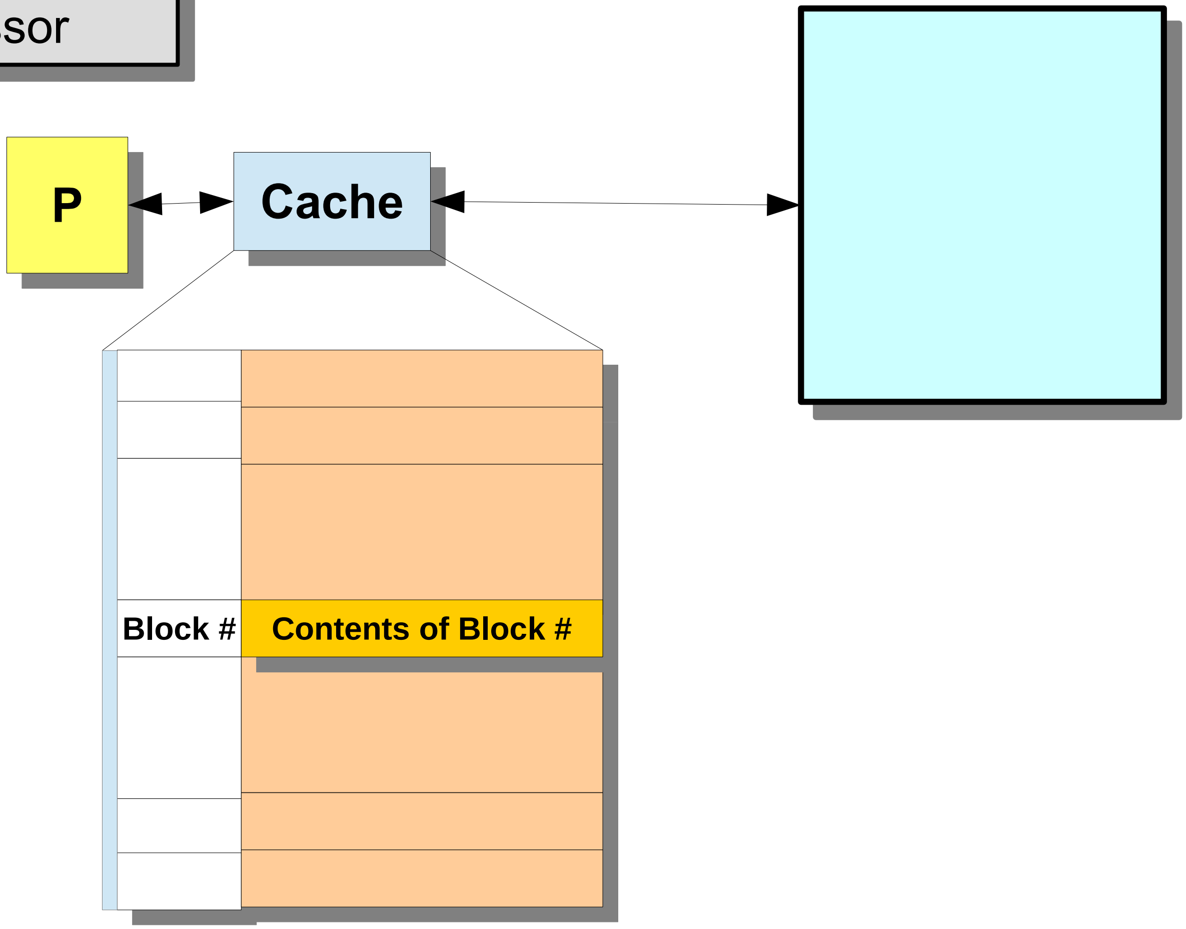
Cache Contents



Cache Design

Processor asks for Byte #

Cache searches through its block numbers; If block found, return data to processor



Question

- 32KB cache contains 64B cache lines. The main memory address is 32b. Each cache line stores 2 bits: Valid and Dirty. What is the size of the bookkeeping (block numbers + flags) bits in the cache?

Block Search in Cache

- A just-arrived block is **placed** in the next available line in the cache.

1	10	Contents of Block 10
1	23	Contents of Block 23
1	13	Contents of Block 13
1	30	Contents of Block 30
1	20	Contents of Block 20
1	7	Contents of Block 7
1	3	Contents of Block 3
1	12	Contents of Block 12

Block Search in Cache

- A just-arrived block is **placed** in the next available line in the cache.
- Time complexity of searching for a block address in this cache = *O(n)*

1	10	Contents of Block 10
1	23	Contents of Block 23
1	13	Contents of Block 13
1	30	Contents of Block 30
1	20	Contents of Block 20
1	7	Contents of Block 7
1	3	Contents of Block 3
1	12	Contents of Block 12

Block Search in Cache

- A just-arrived block is **placed** in the next available line in the cache.
- Time complexity of searching for a block address in this cache = *O(n)*
- Linear Search
  - Because a block can be **placed** anywhere in the cache

1	10	**Contents of Block 10**
1	23	**Contents of Block 23**
1	13	**Contents of Block 13**
1	30	**Contents of Block 30**
1	20	**Contents of Block 20**
1	7	**Contents of Block 7**
1	3	**Contents of Block 3**
1	12	**Contents of Block 12**

To Improve Search Time

- Linear search: *O(n)*
  - Block was placed in the next available cache line

To Improve Search Time

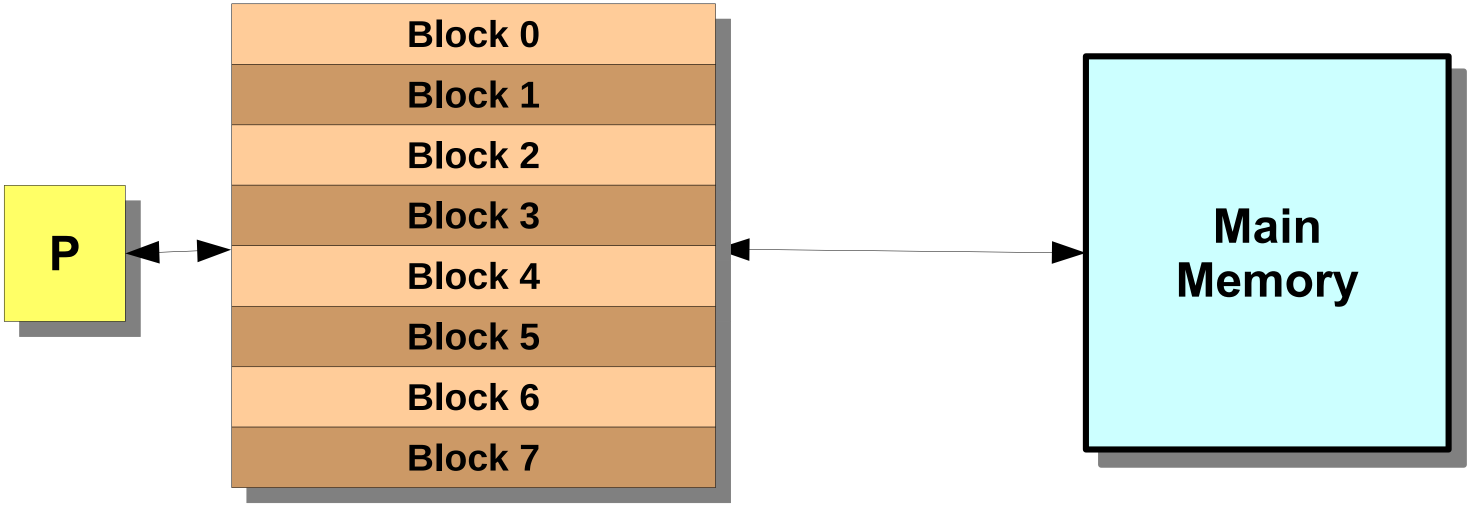
- Linear search: *O(n)*
  - Block was placed in the next available cache line
- Place just-arrived blocks in cache lines identified by the block address

To Improve Search Time

- Linear search: *O(n)*
  - Block was placed in the next available cache line
- Place just-arrived blocks in cache lines identified by the block address
  - Block addresses are unique – locations of cache blocks will also be unique in the cache

To Improve Search Time

1024B Main Memory; 256B cache; 32B block size.

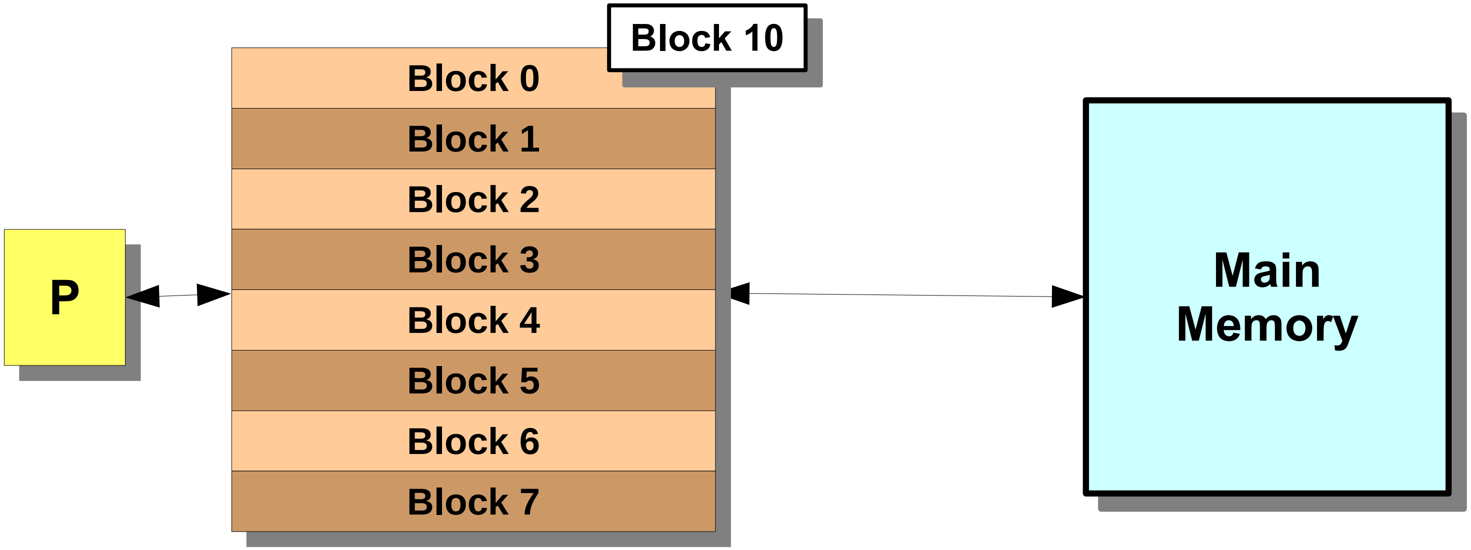


To Improve Search Time

1024B Main Memory; 256B cache; 32B block size.

Processor requests B 324

Block 10 arrives at the Cache



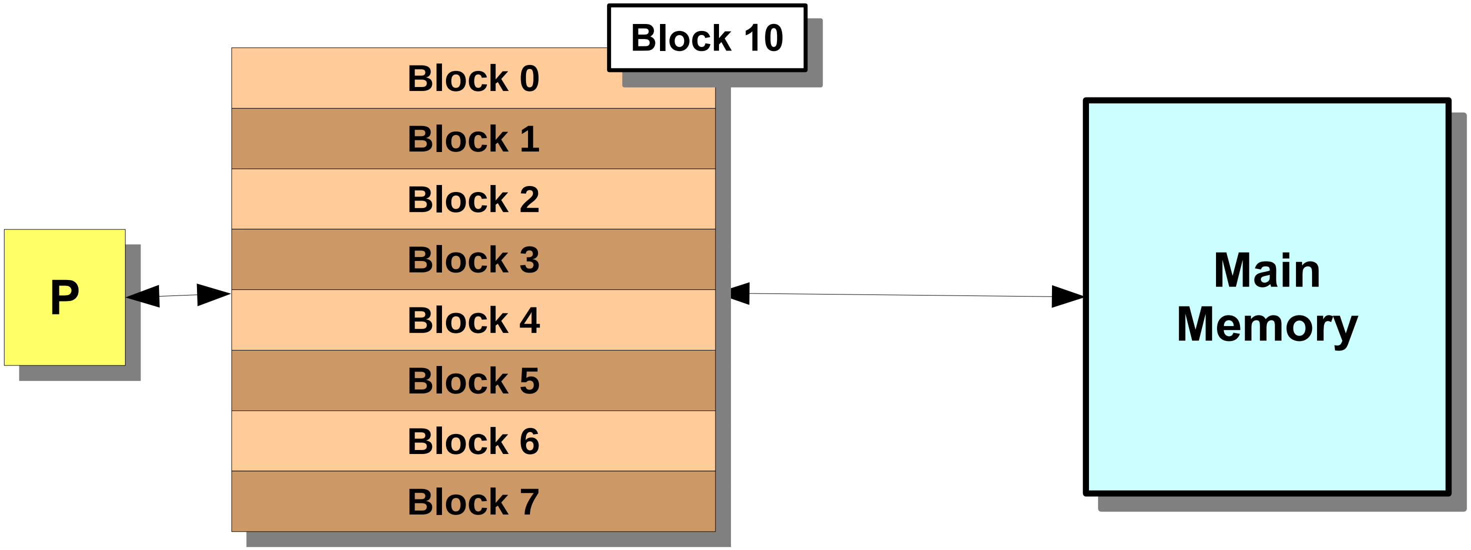
To Improve Search Time

1024B Main Memory; 256B cache; 32B block size.

Processor requests B 324

Block 10 arrives at the Cache

Simple Scheme: Place Block 10 in cache line 10%8 = 2



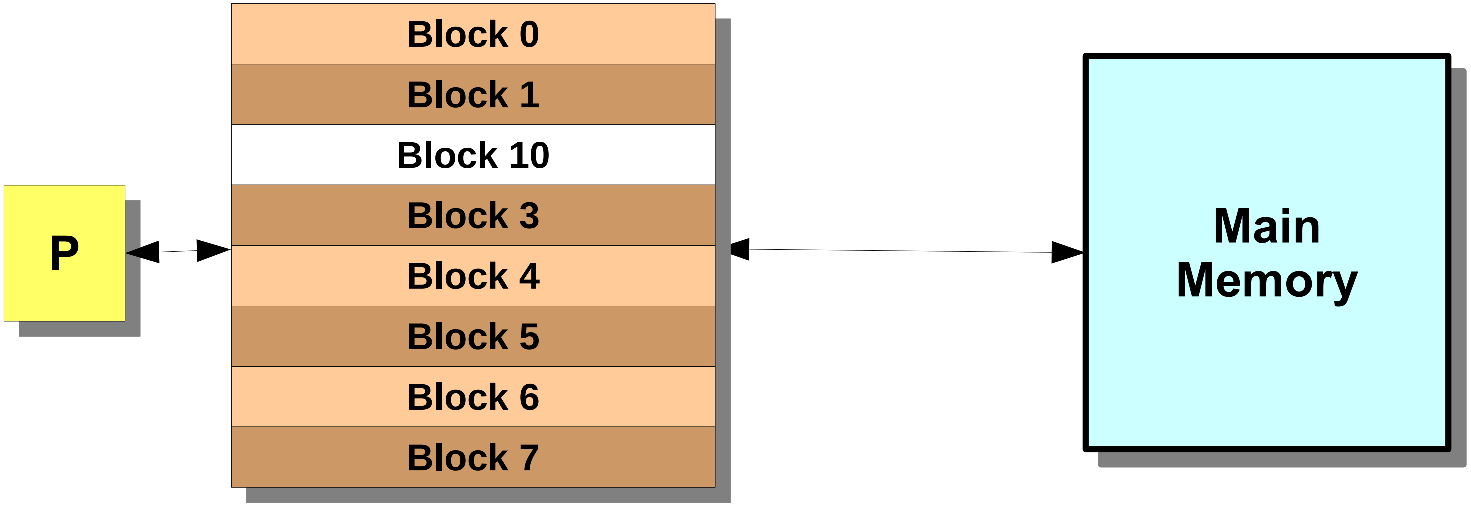
To Improve Search Time

1024B Main Memory; 256B cache; 32B block size.

Processor requests B 324

Block 10 arrives at the Cache

Simple Scheme: Place Block 10 in cache line 10%8 = 2



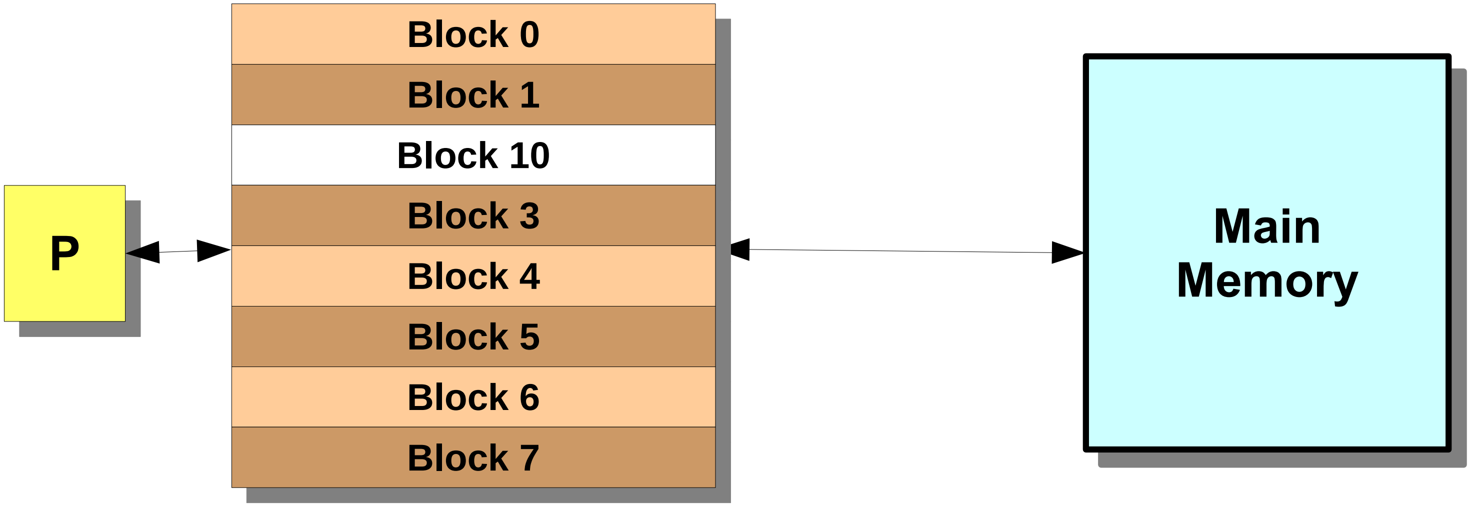
To Improve Search Time

1024B Main Memory; 256B cache; 32B block size.

Processor requests B 324

Block 10 arrives at the Cache

Simple Scheme: Place Block 10 in cache line 10%8 = 2



Block Location = (Block Address) % (Cache size in Blocks)

To Improve Search Time

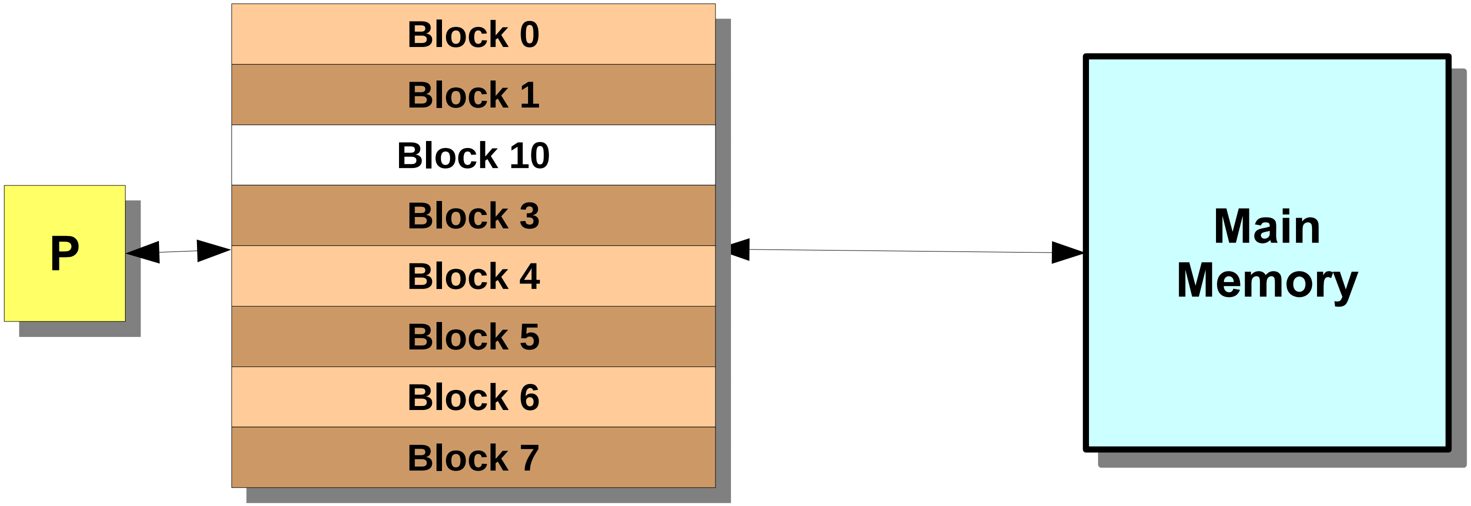
1024B Main Memory; 256B cache; 32B block size.

Processor requests B 324

Block 10 arrives at the Cache

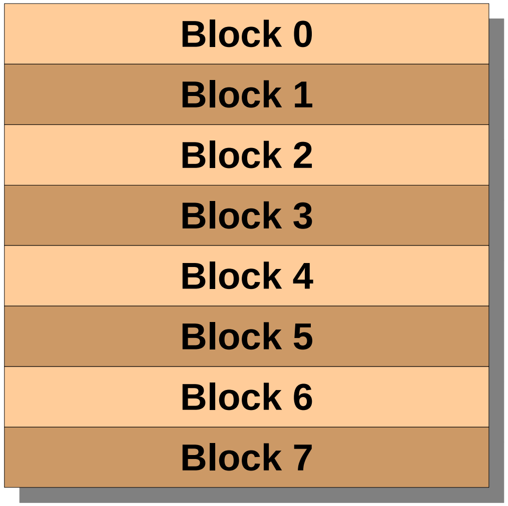
Simple Scheme: Place Block 10 in cache line 10%8 = 2

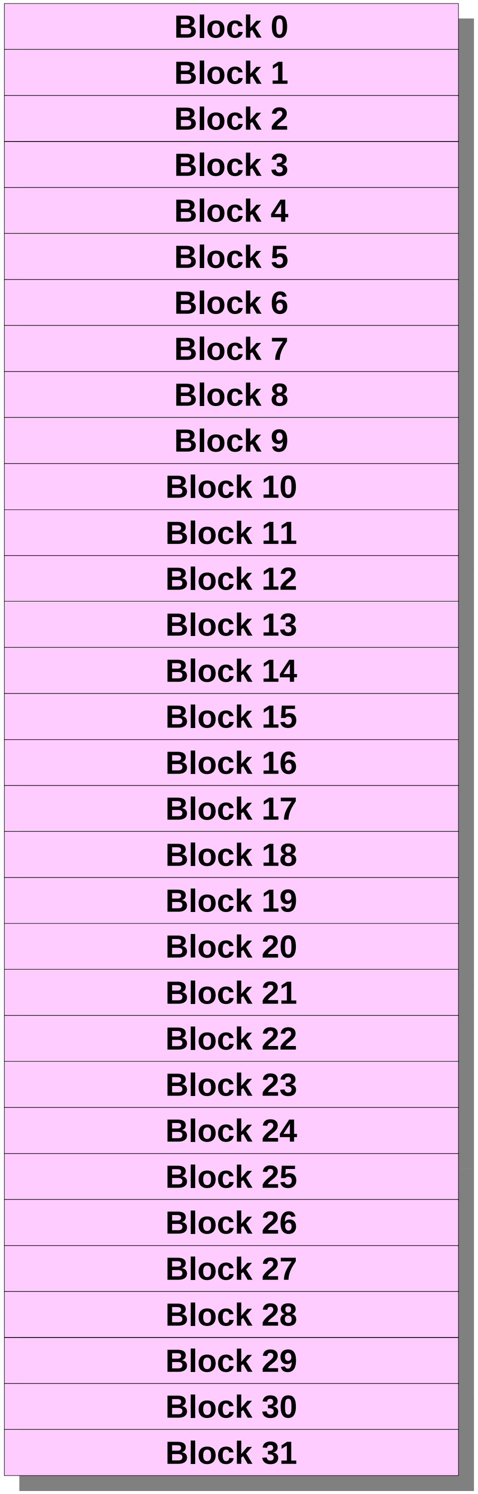
Block Address	Block Offset
01 010	0 0100



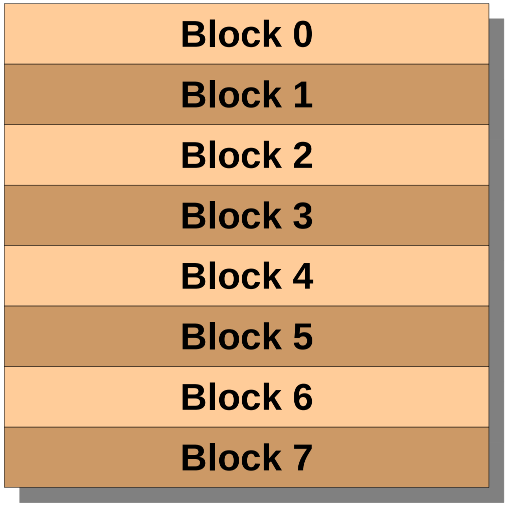
Block Location = (Block Address) % (Cache size in Blocks)

1024B Main Memory; 256B cache; 32B block size.

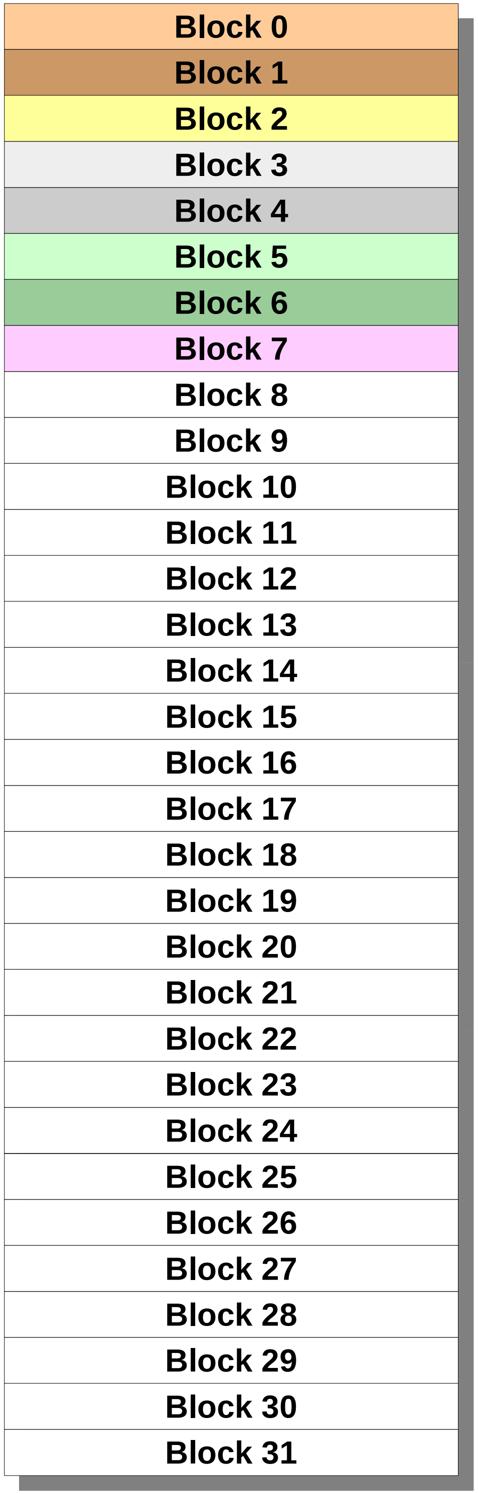




1024B Main Memory; 256B cache; 32B block size.

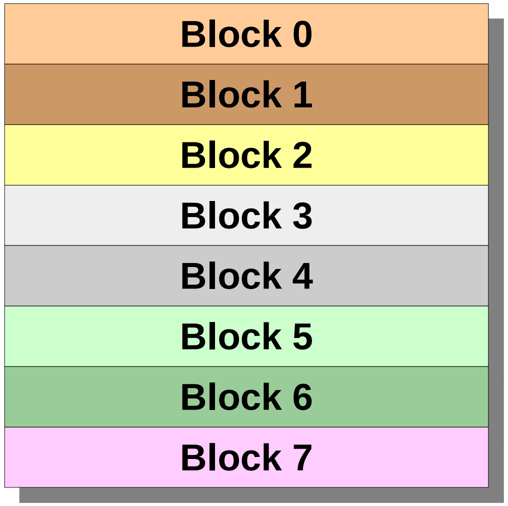


Block 0
Block 1
Block 2
Block 3
Block 4
Block 5
Block 6
Block 7

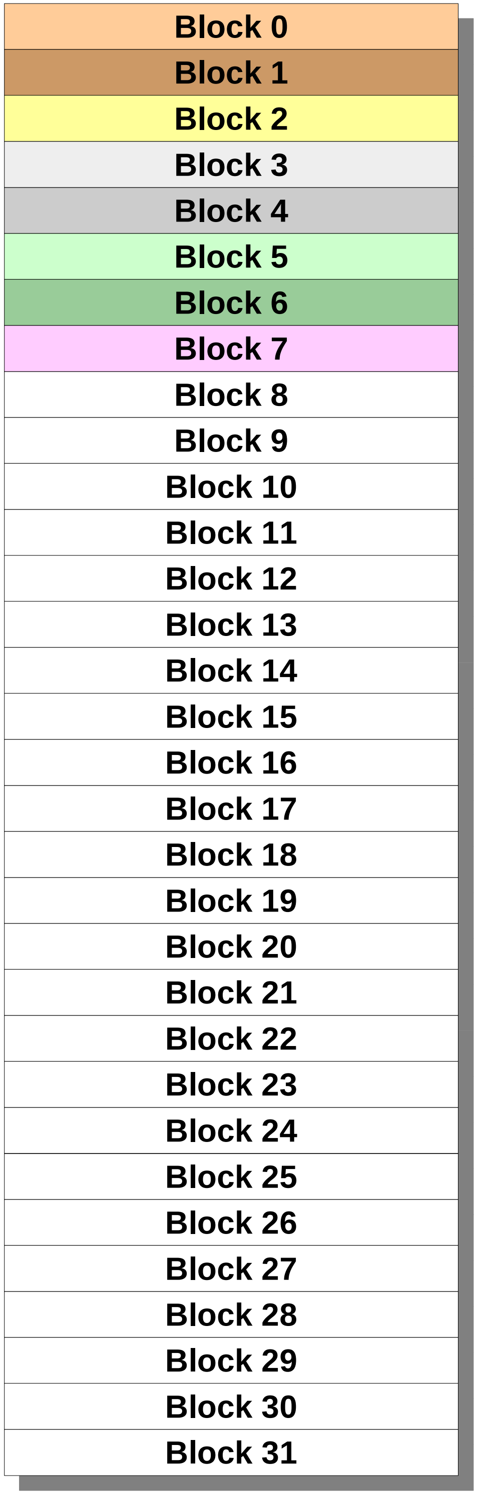


Block 0
Block 1
Block 2
Block 3
Block 4
Block 5
Block 6
Block 7
Block 8
Block 9
Block 10
Block 11
Block 12
Block 13
Block 14
Block 15
Block 16
Block 17
Block 18
Block 19
Block 20
Block 21
Block 22
Block 23
Block 24
Block 25
Block 26
Block 27
Block 28
Block 29
Block 30
Block 31

1024B Main Memory; 256B cache; 32B block size.

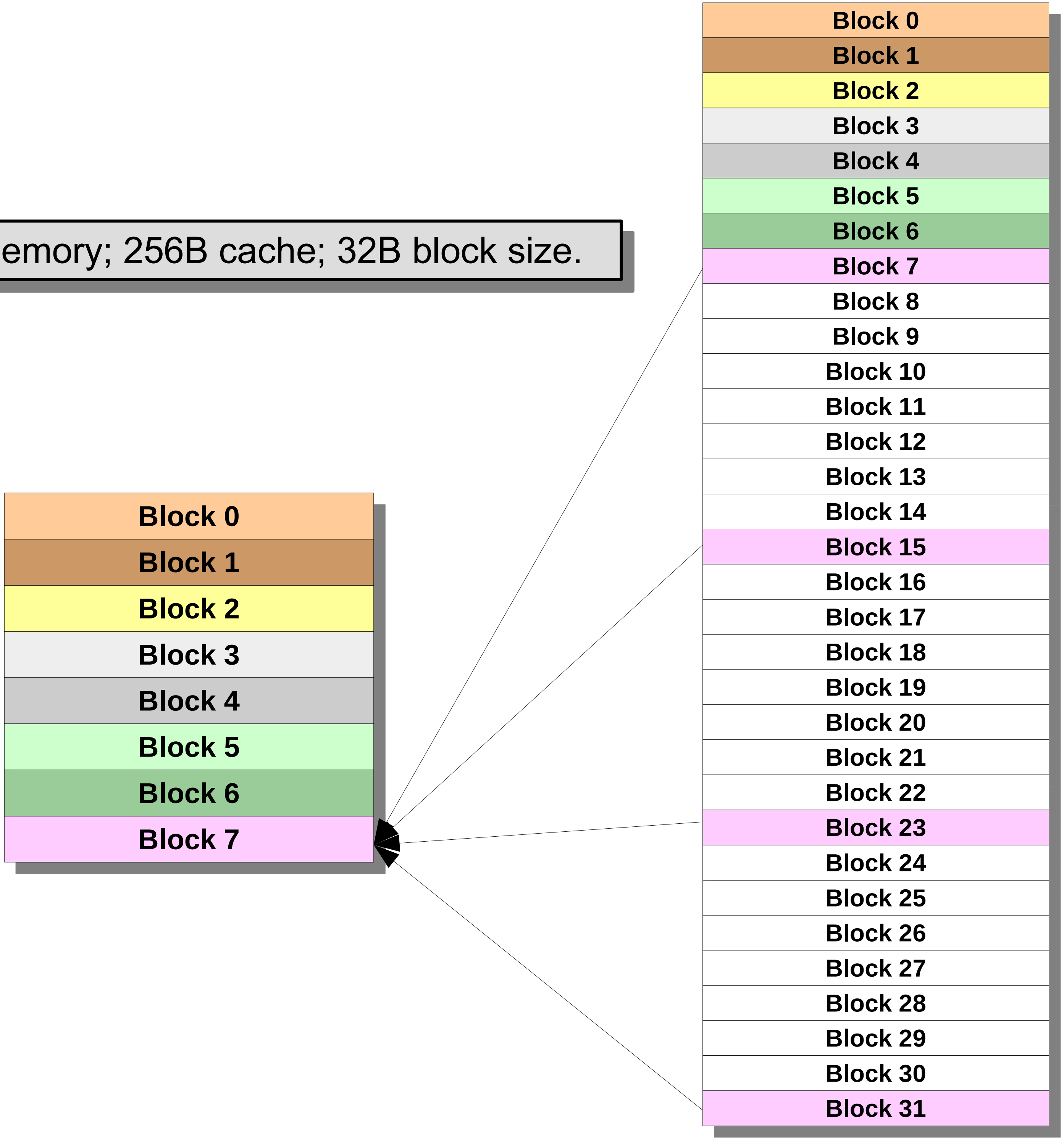


Block 0
Block 1
Block 2
Block 3
Block 4
Block 5
Block 6
Block 7



Block 0
Block 1
Block 2
Block 3
Block 4
Block 5
Block 6
Block 7
Block 8
Block 9
Block 10
Block 11
Block 12
Block 13
Block 14
Block 15
Block 16
Block 17
Block 18
Block 19
Block 20
Block 21
Block 22
Block 23
Block 24
Block 25
Block 26
Block 27
Block 28
Block 29
Block 30
Block 31

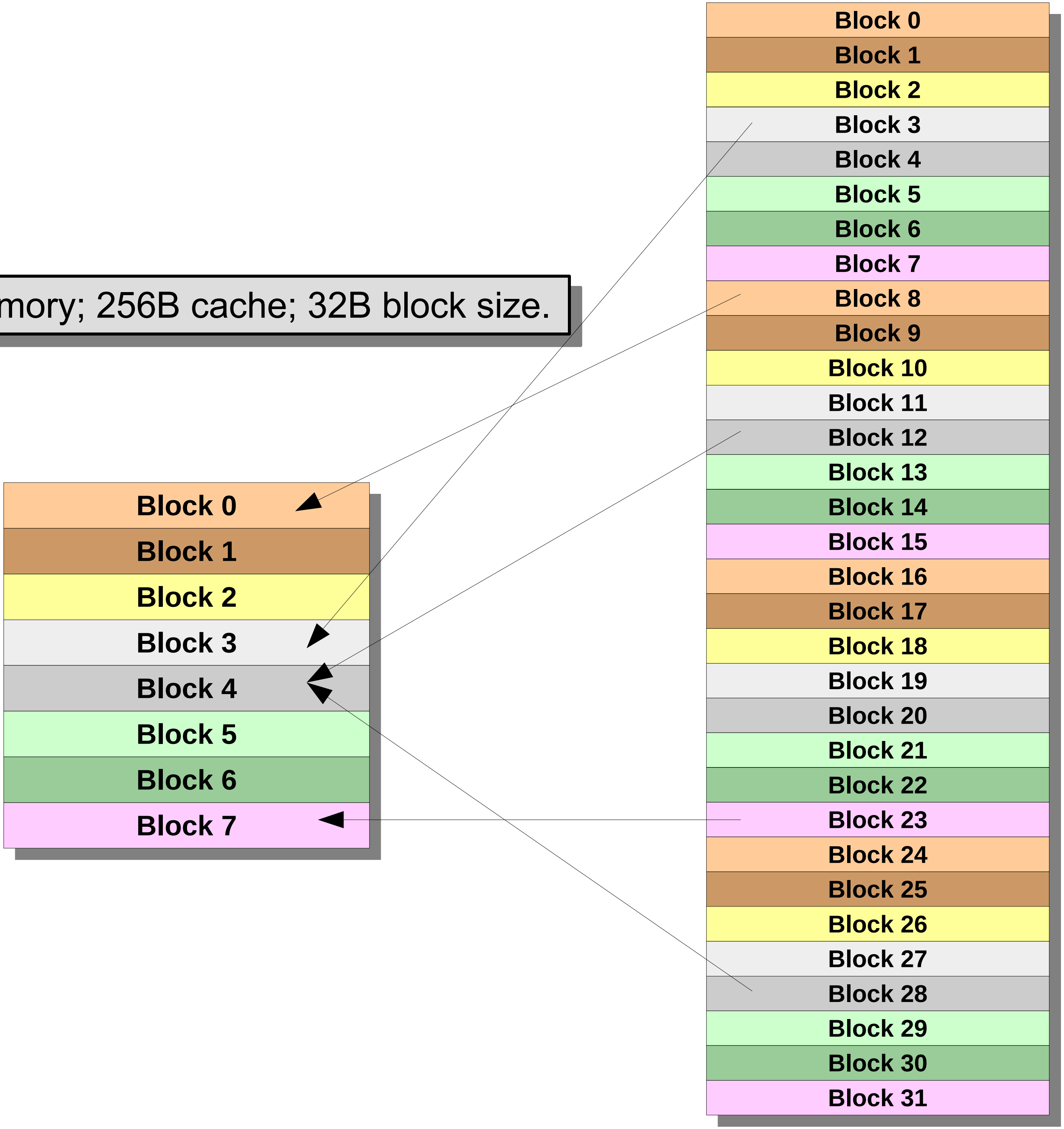
1024B Main Memory; 256B cache; 32B block size.



Block 0
Block 1
Block 2
Block 3
Block 4
Block 5
Block 6
Block 7

Block 0
Block 1
Block 2
Block 3
Block 4
Block 5
Block 6
Block 7
Block 8
Block 9
Block 10
Block 11
Block 12
Block 13
Block 14
Block 15
Block 16
Block 17
Block 18
Block 19
Block 20
Block 21
Block 22
Block 23
Block 24
Block 25
Block 26
Block 27
Block 28
Block 29
Block 30
Block 31

1024B Main Memory; 256B cache; 32B block size.

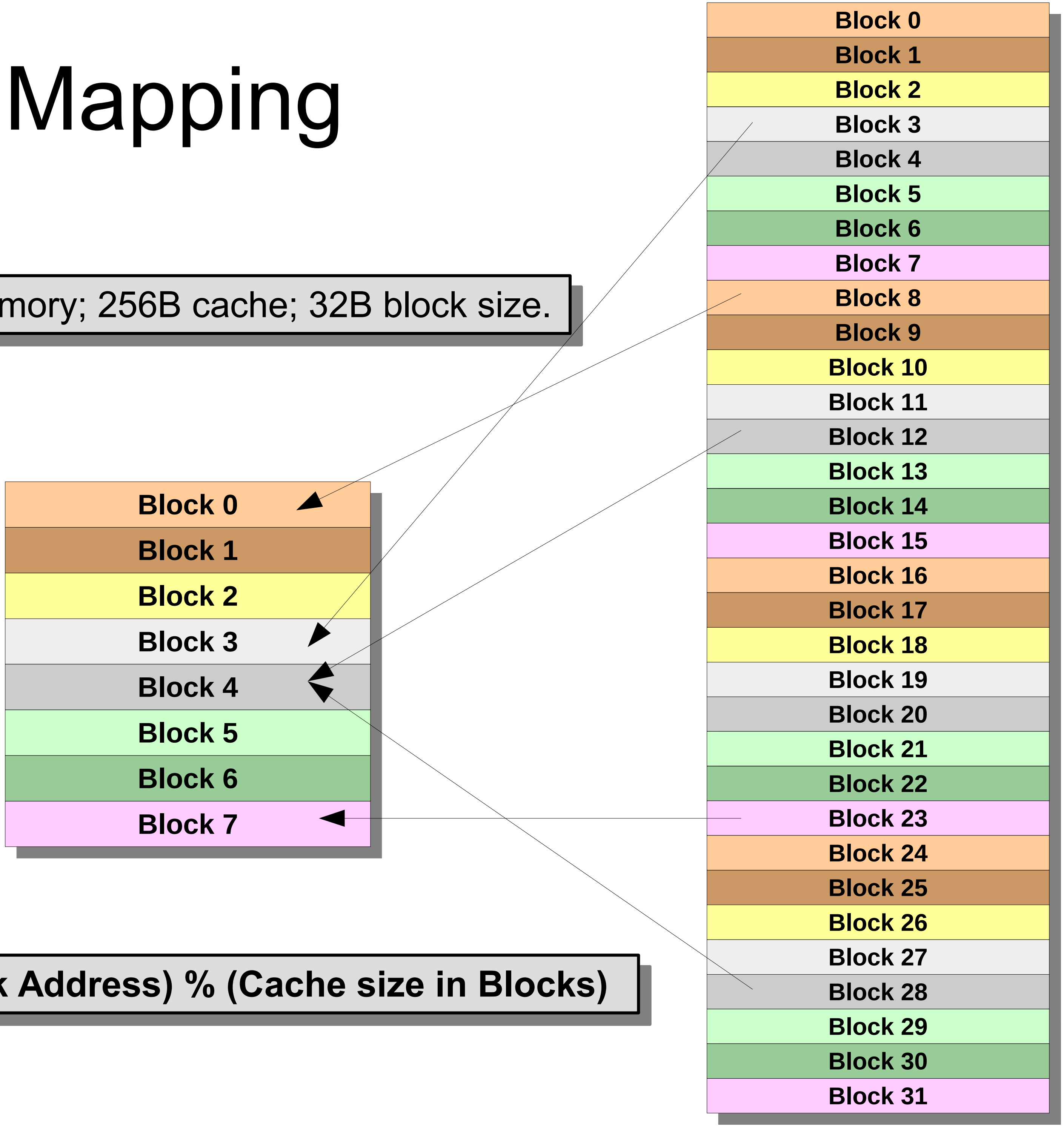


Block 0
Block 1
Block 2
Block 3
Block 4
Block 5
Block 6
Block 7

Block 0
Block 1
Block 2
Block 3
Block 4
Block 5
Block 6
Block 7
Block 8
Block 9
Block 10
Block 11
Block 12
Block 13
Block 14
Block 15
Block 16
Block 17
Block 18
Block 19
Block 20
Block 21
Block 22
Block 23
Block 24
Block 25
Block 26
Block 27
Block 28
Block 29
Block 30
Block 31

Direct Mapping

1024B Main Memory; 256B cache; 32B block size.



Index = (Block Address) % (Cache size in Blocks)

Contents of the DM Cache

- The following cache blocks are accessed in sequence. Show the state of the Cache after each block is placed in the cache. Assume that the cache is empty at start. Show Block Address bits.
- 10, 23, 13, 8, 18, 27

Contents of the DM Cache

	Block 0
	Block 1
	Block 2
	Block 3
	Block 4
	Block 5
	Block 6
	Block 7

10, 23, 13, 8, 18, 27

Contents of the DM Cache

	Block 0
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
	Block 5
	Block 6
	Block 7

10 mod 8 = 2

10, 23, 13, 8, 18, 27

Contents of the DM Cache

	Block 0
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
	Block 5
	Block 6
	Block 7

10 mod 8 = 2

	Block 0
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
	Block 5
	Block 6
10111	Contents of Block 23

23 mod 8 = 7

	Block 0
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
01101	Contents of Block 13
	Block 6
10111	Contents of Block 23

13 mod 8 = 5

01000	Contents of Block 8
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
01101	Contents of Block 13
	Block 6
10111	Contents of Block 23

8 mod 8 = 0

01000	Contents of Block 8
	Block 1
10010	Contents of Block 18
	Block 3
	Block 4
01101	Contents of Block 13
	Block 6
10111	Contents of Block 23

18 mod 8 = 2

Eviction due to address conflict
4 Blocks are empty

Contents of the DM Cache

	Block 0
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
	Block 5
	Block 6
	Block 7

10 mod 8 = 2

	Block 0
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
	Block 5
	Block 6
10111	Contents of Block 23

23 mod 8 = 7

10, 23, 13, 8, 18, 27

Contents of the DM Cache

	Block 0
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
	Block 5
	Block 6
	Block 7

10 mod 8 = 2

	Block 0
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
	Block 5
	Block 6
10111	Contents of Block 23

23 mod 8 = 7

	Block 0
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
01101	Contents of Block 13
	Block 6
10111	Contents of Block 23

13 mod 8 = 5

10, 23, 13, 8, 18, 27

Contents of the DM Cache

	Block 0
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
	Block 5
	Block 6
	Block 7

10 mod 8 = 2

	Block 0
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
	Block 5
	Block 6
10111	Contents of Block 23

23 mod 8 = 7

	Block 0
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
01101	Contents of Block 13
	Block 6
10111	Contents of Block 23

13 mod 8 = 5

01000	Contents of Block 8
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
01101	Contents of Block 13
	Block 6
10111	Contents of Block 23

8 mod 8 = 0

10, 23, 13, 8, 18, 27

Contents of the DM Cache

	Block 0
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
	Block 5
	Block 6
	Block 7

10 mod 8 = 2

	Block 0
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
	Block 5
	Block 6
10111	Contents of Block 23

23 mod 8 = 7

	Block 0
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
01101	Contents of Block 13
	Block 6
10111	Contents of Block 23

13 mod 8 = 5

01000	Contents of Block 8
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
01101	Contents of Block 13
	Block 6
10111	Contents of Block 23

8 mod 8 = 0

01000	Contents of Block 8
	Block 1
10010	Contents of Block 18
	Block 3
	Block 4
01101	Contents of Block 13
	Block 6
10111	Contents of Block 23

18 mod 8 = 2

10, 23, 13, 8, 18, 27

Contents of the DM Cache

	Block 0
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
	Block 5
	Block 6
	Block 7

10 mod 8 = 2

	Block 0
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
	Block 5
	Block 6
10111	Contents of Block 23

23 mod 8 = 7

	Block 0
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
01101	Contents of Block 13
	Block 6
10111	Contents of Block 23

13 mod 8 = 5

01000	Contents of Block 8
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
01101	Contents of Block 13
	Block 6
10111	Contents of Block 23

8 mod 8 = 0

01000	Contents of Block 8
	Block 1
10010	Contents of Block 18
	Block 3
	Block 4
01101	Contents of Block 13
	Block 6
10111	Contents of Block 23

18 mod 8 = 2

Eviction due to address conflict
4 Blocks are empty

10, 23, 13, 8, 18, 27

Contents of the DM Cache

	Block 0
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
	Block 5
	Block 6
	Block 7

10 mod 8 = 2

	Block 0
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
	Block 5
	Block 6
10111	Contents of Block 23

23 mod 8 = 7

	Block 0
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
01101	Contents of Block 13
	Block 6
10111	Contents of Block 23

13 mod 8 = 5

01000	Contents of Block 8
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
01101	Contents of Block 13
	Block 6
10111	Contents of Block 23

8 mod 8 = 0

01000	Contents of Block 8
	Block 1
10010	Contents of Block 18
	Block 3
	Block 4
01101	Contents of Block 13
	Block 6
10111	Contents of Block 23

18 mod 8 = 2

01000	Contents of Block 8
	Block 1
10010	Contents of Block 18
11011	Contents of Block 27
	Block 4
01101	Contents of Block 13
	Block 6
10111	Contents of Block 23

27 mod 8 = 3

Contents of the DM Cache

	Block 0
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
	Block 5
	Block 6
	Block 7

10 mod 8 = 2

	Block 0
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
	Block 5
	Block 6
10111	Contents of Block 23

23 mod 8 = 7

	Block 0
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
01101	Contents of Block 13
	Block 6
10111	Contents of Block 23

13 mod 8 = 5

01000	Contents of Block 8
	Block 1
01010	Contents of Block 10
	Block 3
	Block 4
01101	Contents of Block 13
	Block 6
10111	Contents of Block 23

8 mod 8 = 0

01000	Contents of Block 8
	Block 1
10010	Contents of Block 18
	Block 3
	Block 4
01101	Contents of Block 13
	Block 6
10111	Contents of Block 23

18 mod 8 = 2

01000	Contents of Block 8
	Block 1
10010	Contents of Block 18
11011	Contents of Block 27
	Block 4
01101	Contents of Block 13
	Block 6
10111	Contents of Block 23

27 mod 8 = 3

Mod Operation

1001 01011	mod 2	=	1
1001 01011	mod 4	=	11
1001 01011	mod 8	=	011
1001 01011	mod 16	=	1011
1001 01011	mod 32	=	01011
1001 01011	mod 64	=	1 01011
1001 01011	mod 128	=	01 01011

For A mod 2^m,
extract $\log_2 m$
least significant bits

Question

- Identify the Block Index in a 32KB Direct Mapped cache with 32B cache lines, using 32b addresses. Address = 0xABCD1010.

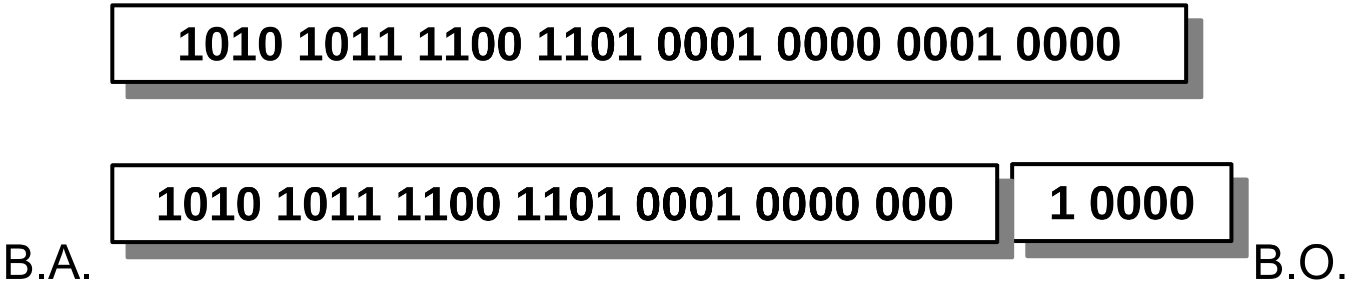
Question

- Identify the Block Index in a 32KB Direct Mapped cache with 32B cache lines, using 32b addresses. Address = 0xABCD1010.

1010 1011 1100 1101 0001 0000 0001 0000

Question

- Identify the Block Index in a 32KB Direct Mapped cache with 32B cache lines, using 32b addresses. Address = 0xABCD1010.
- 1024 cache lines



1010 1011 1100 1101 0001 0000 0001 0000

1010 1011 1100 1101 0001 0000 000

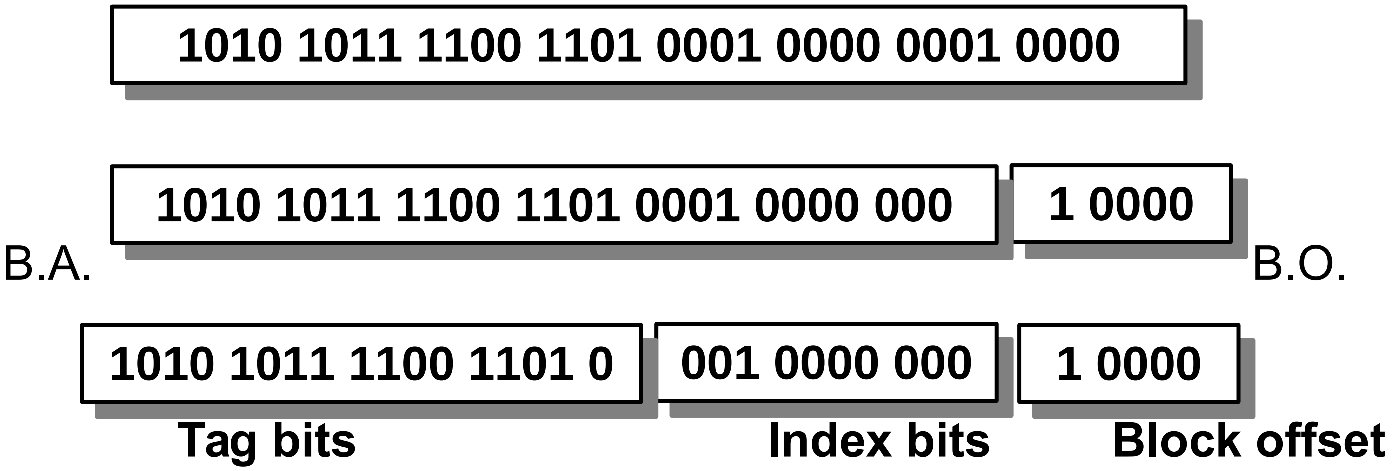
1 0000

B.A.

B.O.

Question

- Identify the Block Index in a 32KB Direct Mapped cache with 32B cache lines, using 32b addresses. Address = 0xABCD1010.
- 1024 cache lines



1010 1011 1100 1101 0001 0000 0001 0000

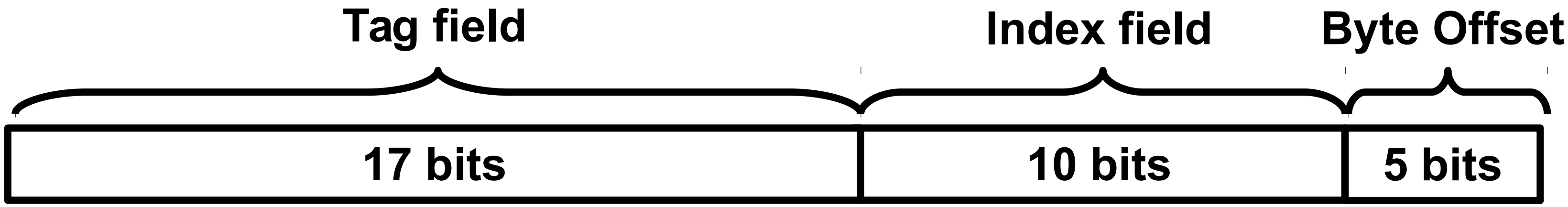
B.A. **1010 1011 1100 1101 0001 0000 000** **1 0000** B.O.

1010 1011 1100 1101 0 **001 0000 000** **1 0000**

Tag bits **Index bits** **Block offset**

Question

- Identify the Block Index in a 32KB Direct Mapped cache with 32B cache lines, using 32b addresses. Address = 0xABCD1010.
- 1024 cache lines



Tag field

Index field

Byte Offset

17 bits

10 bits

5 bits

Direct Mapped Cache Organization

Index	Valid	Tag	Data
0			
1			
2			
.			
.			
.			
1022			
1023			

Direct Mapped Cache Organization

32 bit address from Processor

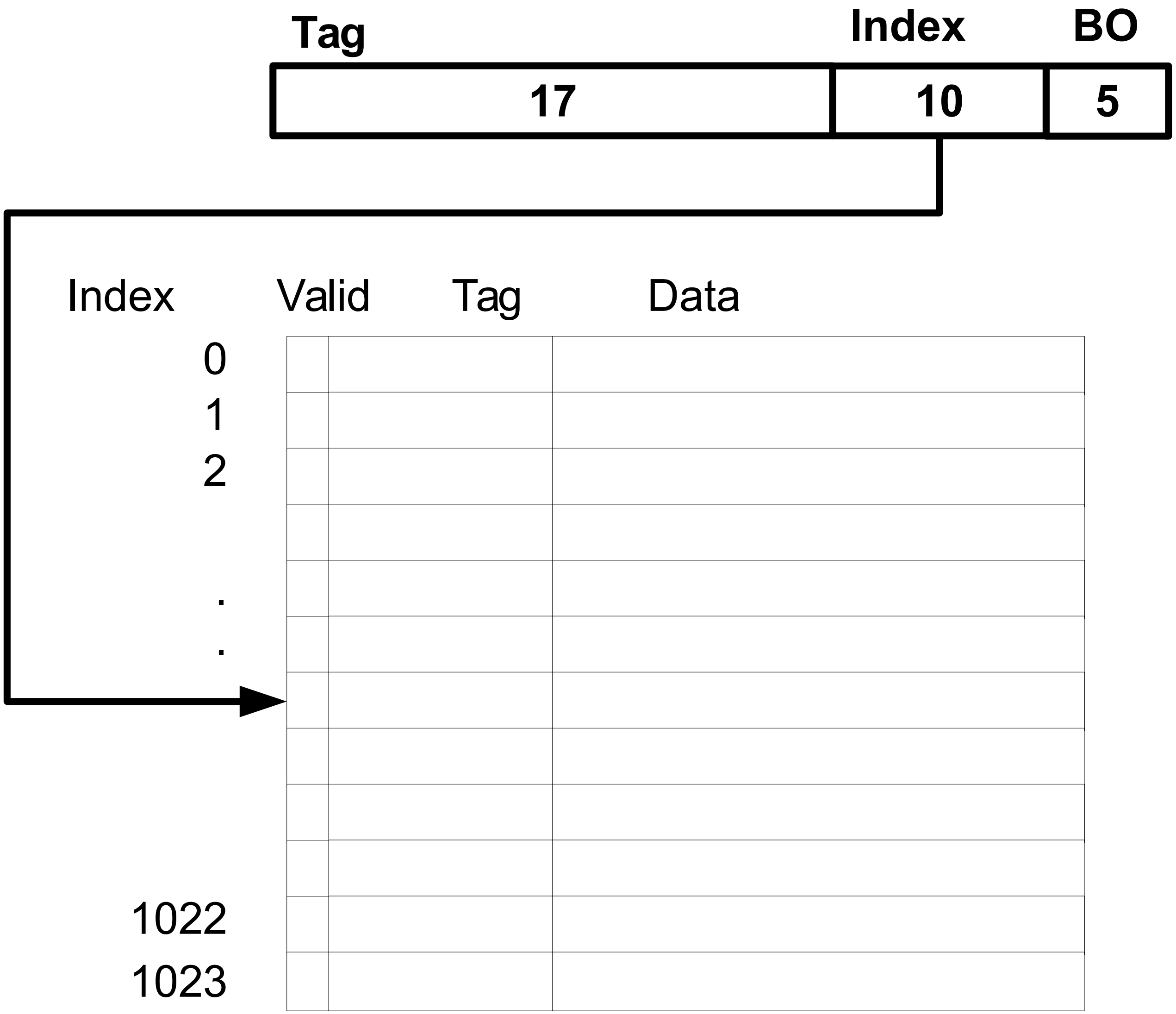
Index	Valid	Tag	Data
0			
1			
2			
.			
.			
.			
1022			
1023			

Direct Mapped Cache Organization

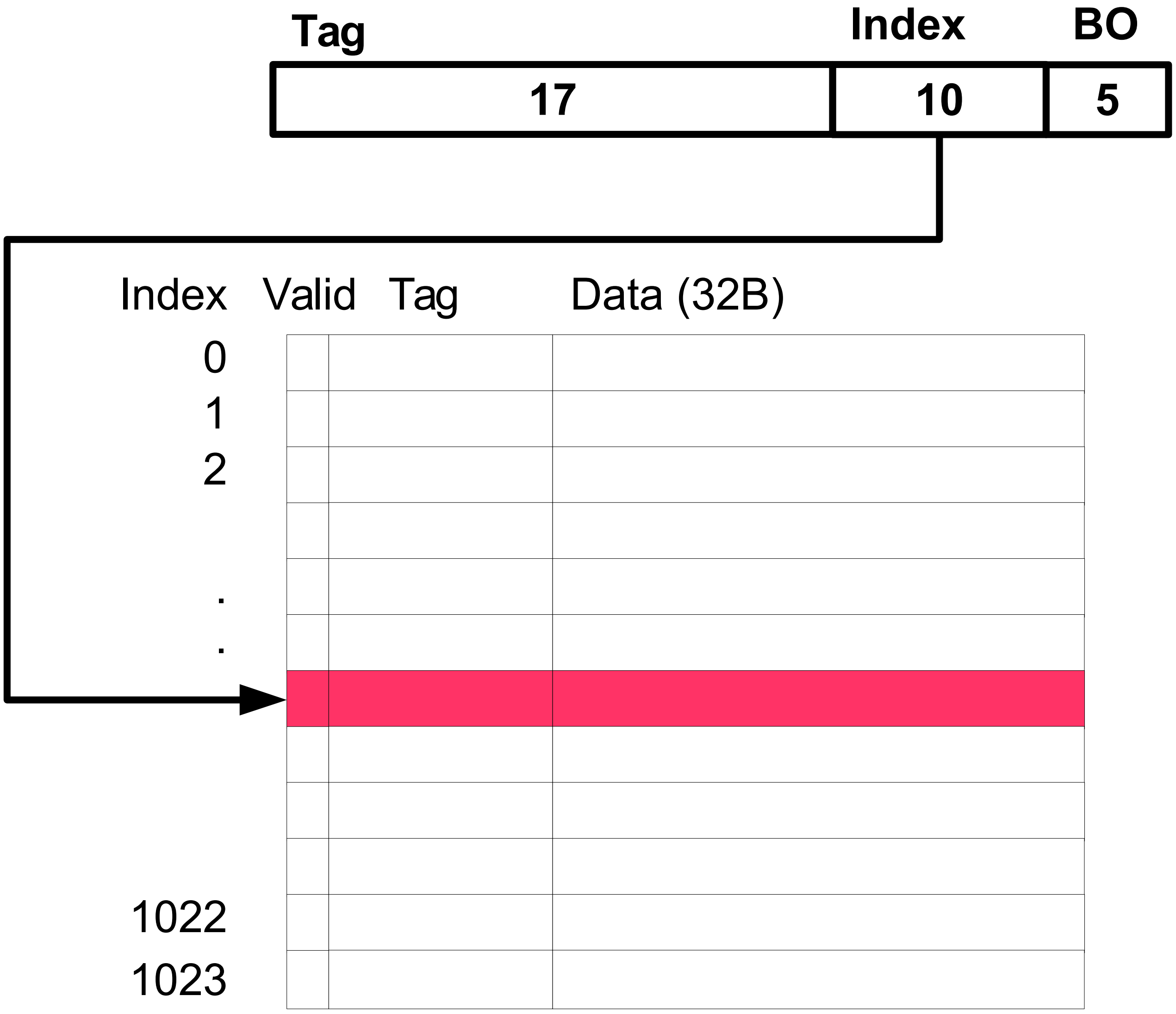
Tag	Index	BO
17	10	5

Index	Valid	Tag	Data
0			
1			
2			
.			
.			
.			
1022			
1023			

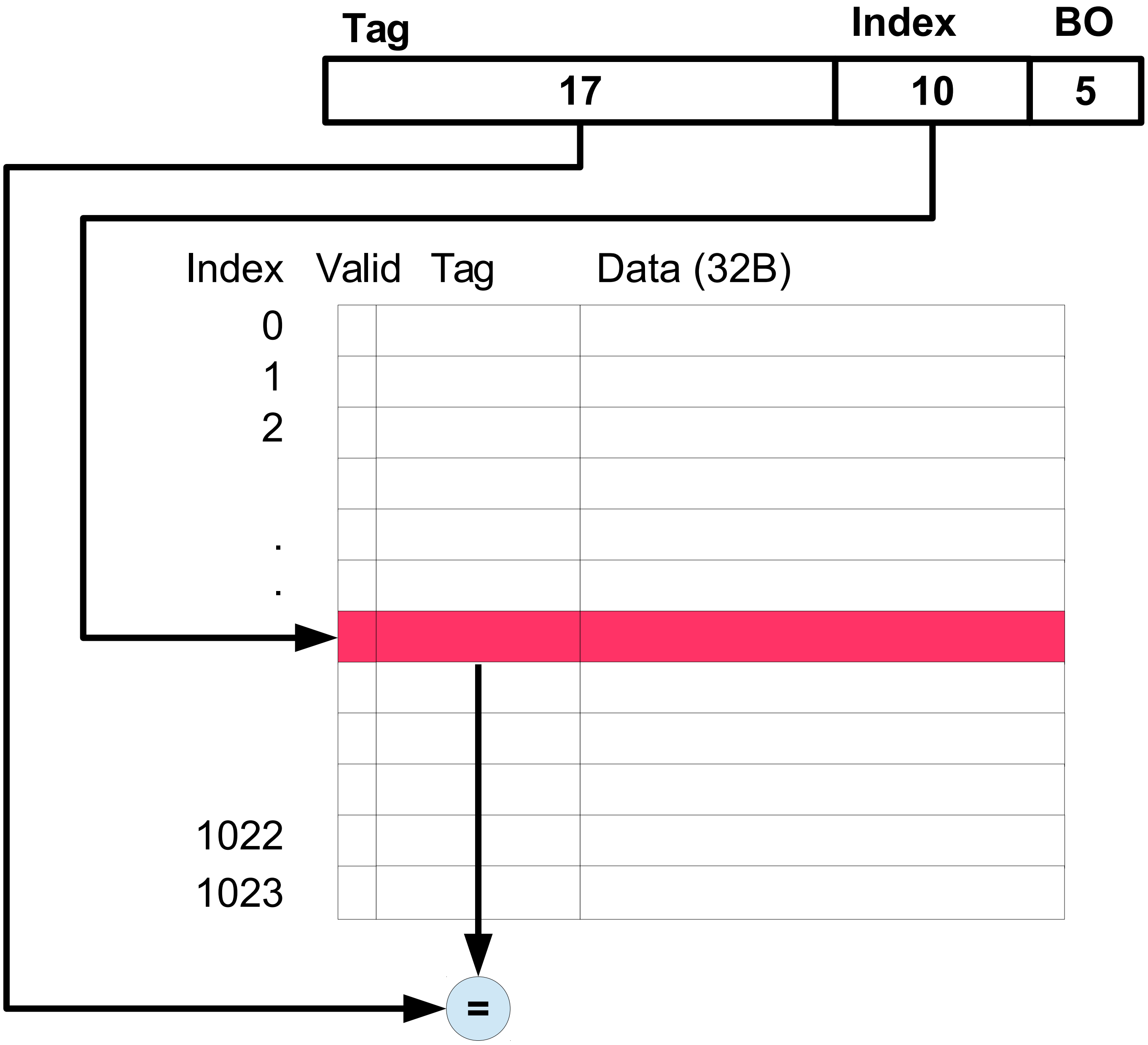
Direct Mapped Cache Organization



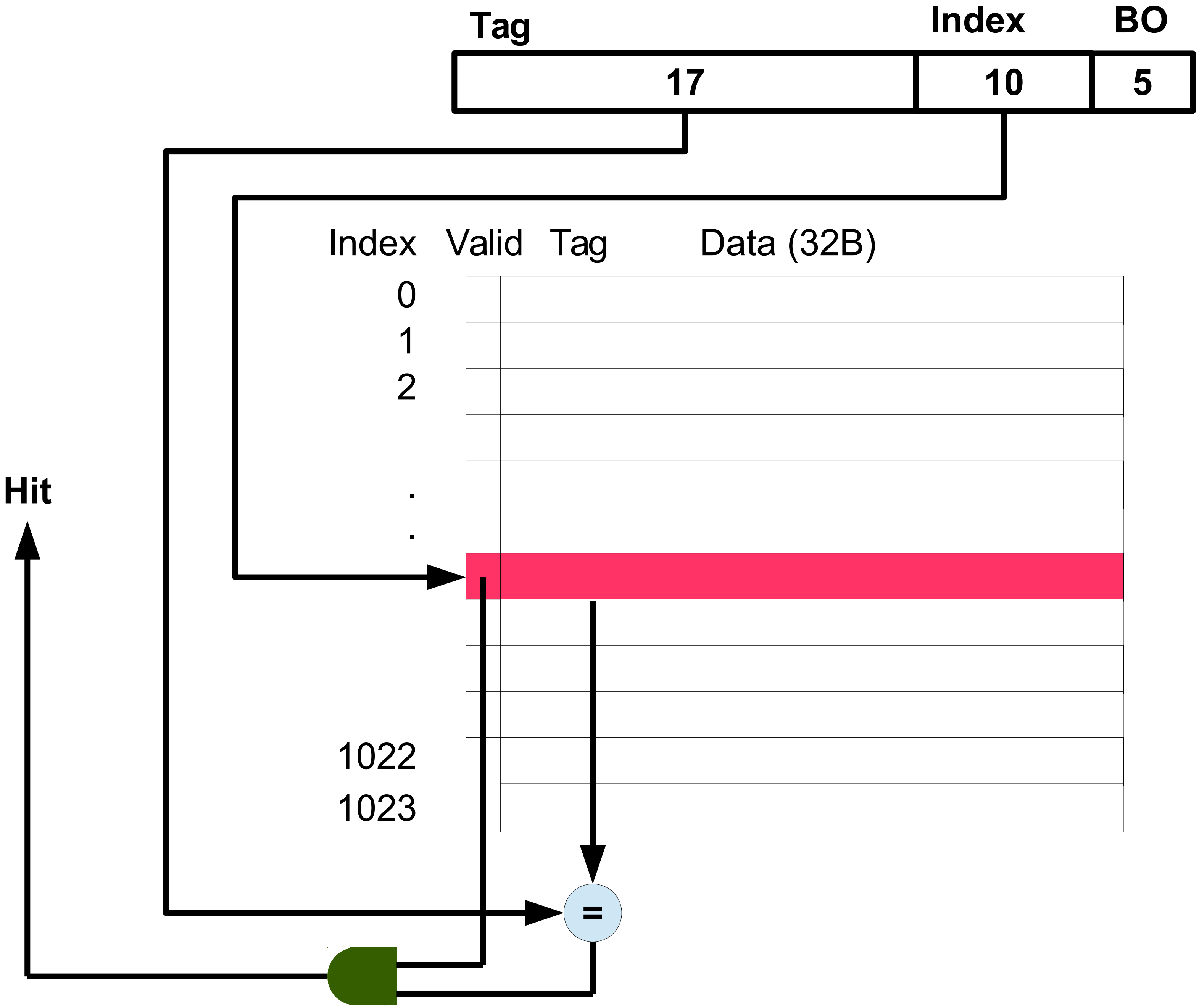
Direct Mapped Cache Organization



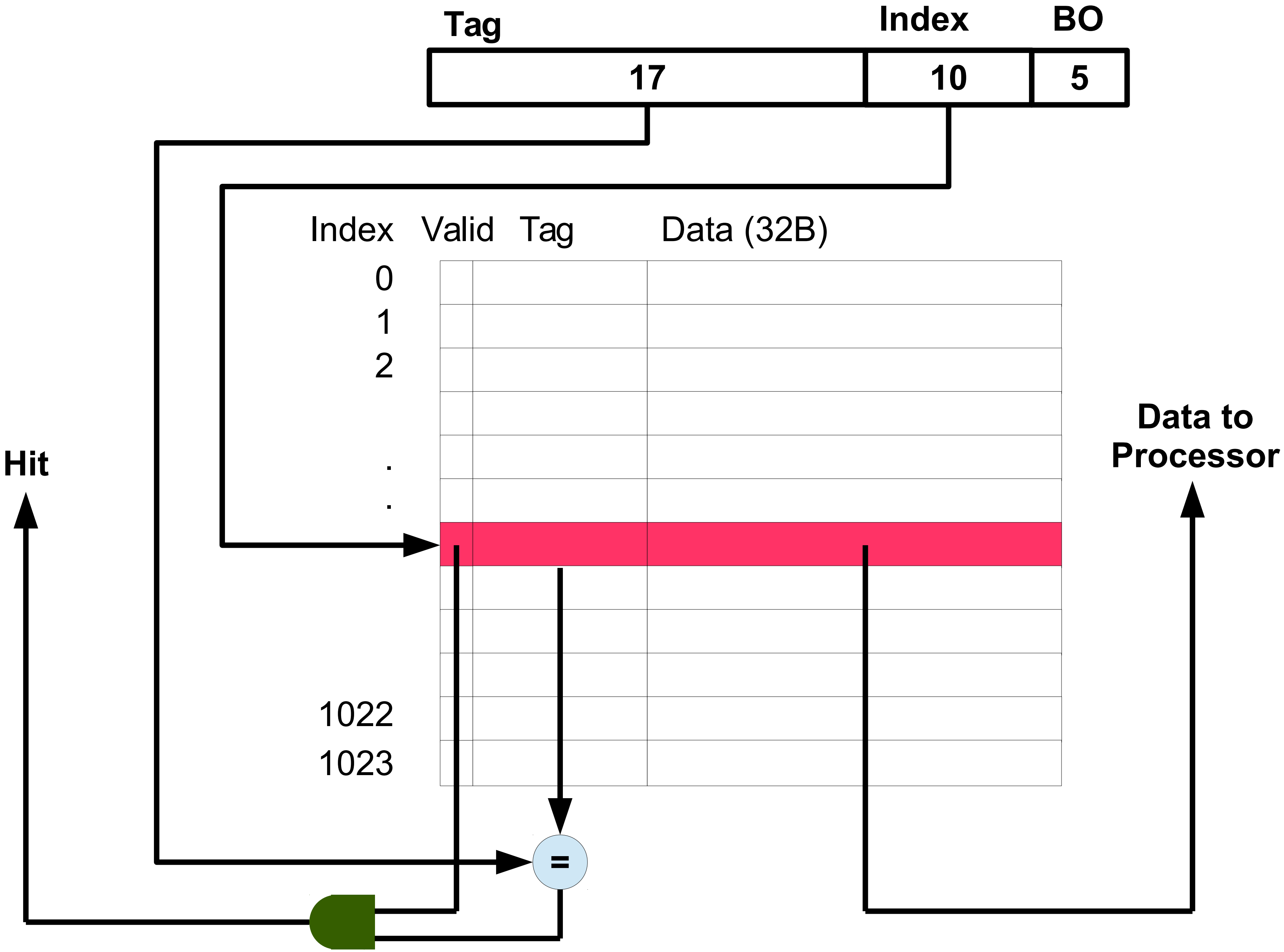
Direct Mapped Cache Organization



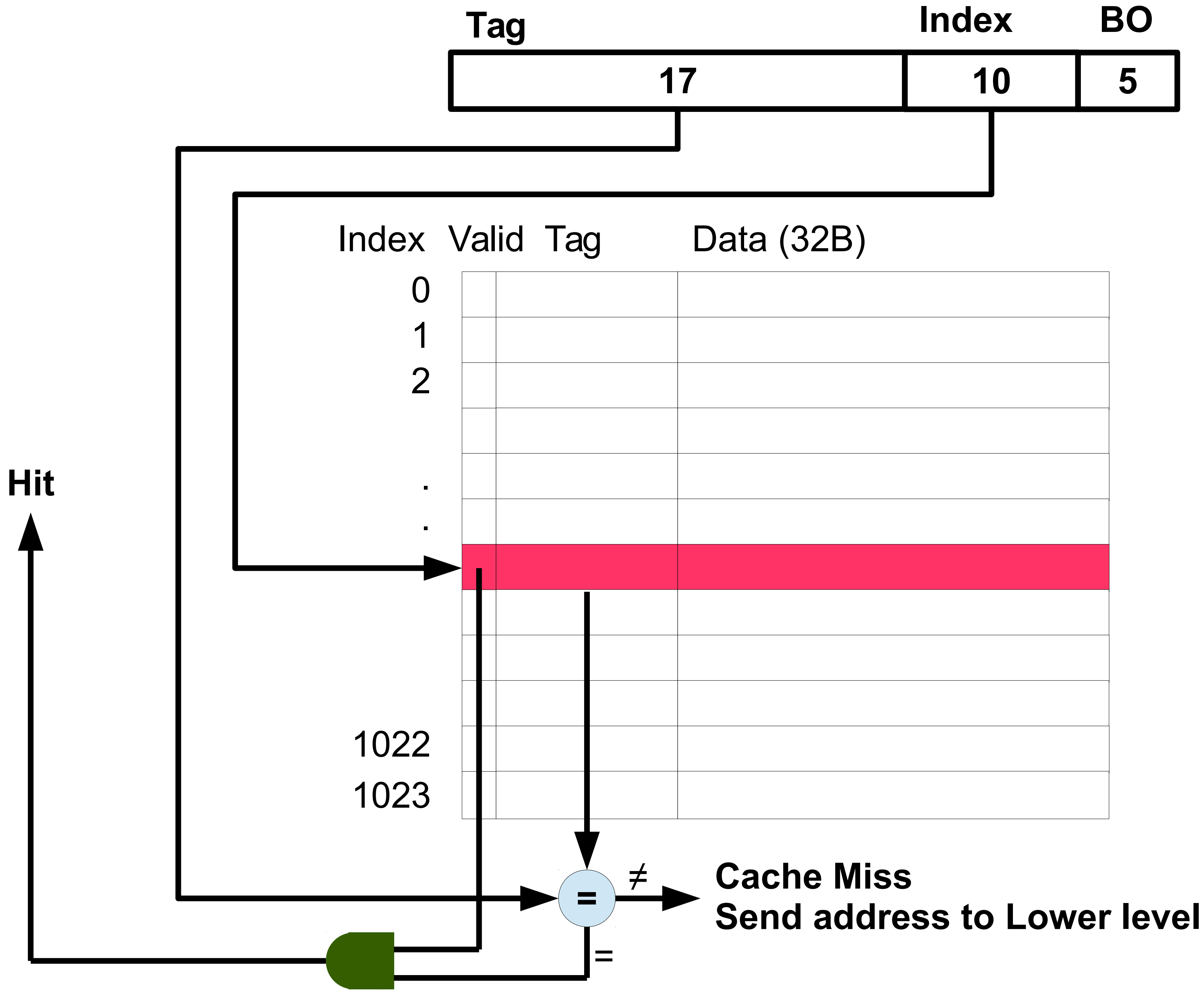
Direct Mapped Cache Organization



Direct Mapped Cache Organization



Direct Mapped Cache Organization



Tag

Index

BO

17

10

5

Index Valid Tag Data (32B)

0

1

2

.

.

1022

1023

Hit

=

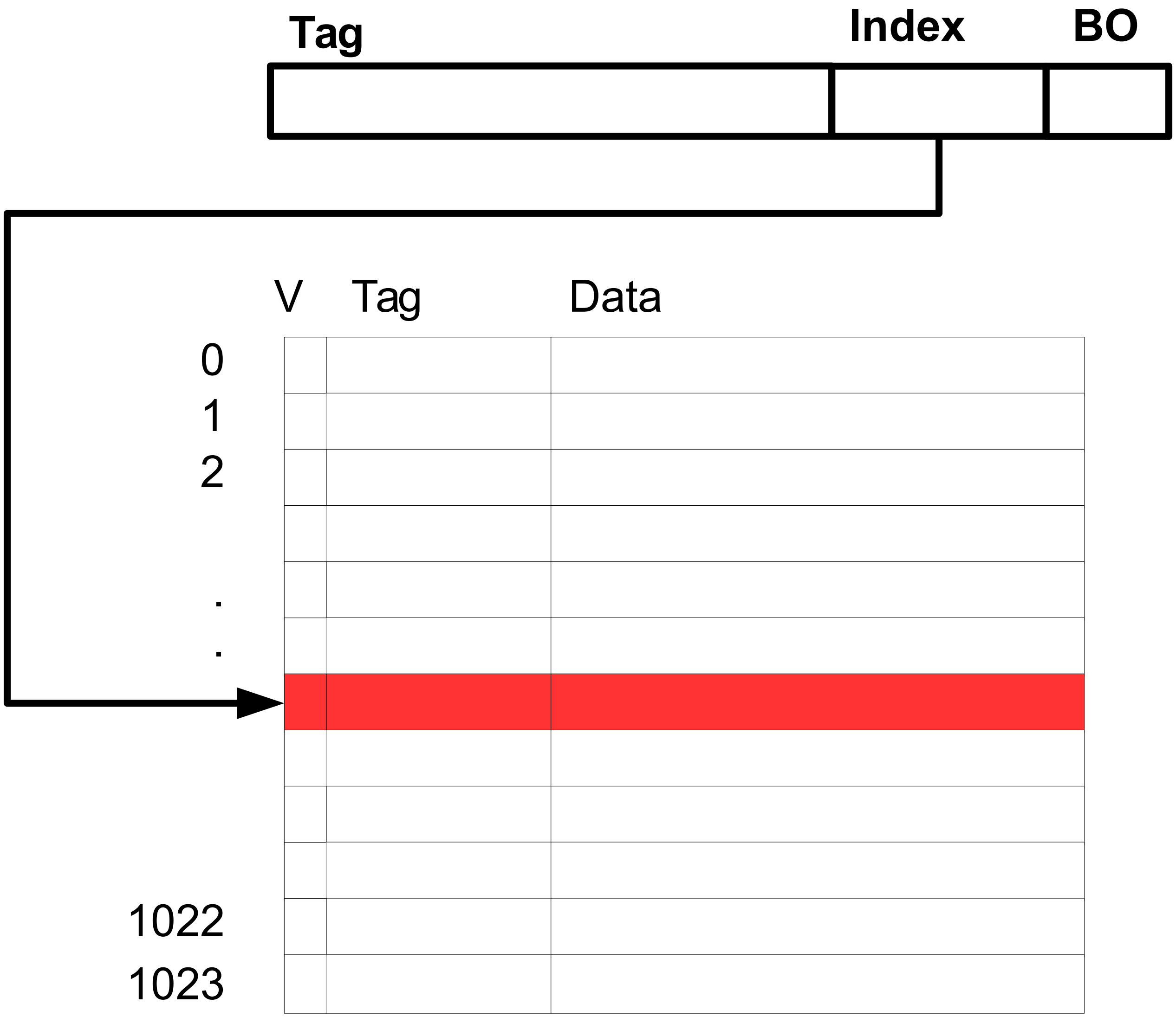
≠

=

Cache Miss

Send address to Lower level

Block Placement



Direct Mapped Cache

Index bits Identify a unique Cache line

To Improve Search Time

- Linear search: *O(n)*
  - Block was placed in the next available cache line
  - Fully Associative mapping
- Hashing: *O(1)*
  - Block will be placed in a unique location identified by the block address itself
  - Direct mapping

Direct Mapped Cache

- Block placement done using index bits

Direct Mapped Cache

- Block placement done using index bits
- Search is faster than FA cache

Direct Mapped Cache

- Block placement done using index bits
- Search is faster than FA cache
- Conflicts may result in under utilization of the available cache space.
  - Access sequence in our example: 10, 23, 13, 30, 18, 27.
  - Block 18 evicted Block 10
-

Direct Mapped Cache

- Block placement done using index bits
- Search is faster than FA cache
- Conflicts may result in under utilization of the available cache space.
  - Access sequence in our example: 10, 23, 13, 30, 18, 27.
  - Block 18 evicted Block 10
- Improve hit rate by reducing conflicts

Set Associative Mapping

- Index bits uniquely identify a set of blocks

Set Associative Mapping

- Index bits uniquely identify a set of blocks
- Block can be placed anywhere in the set.

Set Associative Mapping

- Index bits uniquely identify a set of blocks
- Block can be placed anywhere in the set.
- No. of Index bits are calculated as follows:

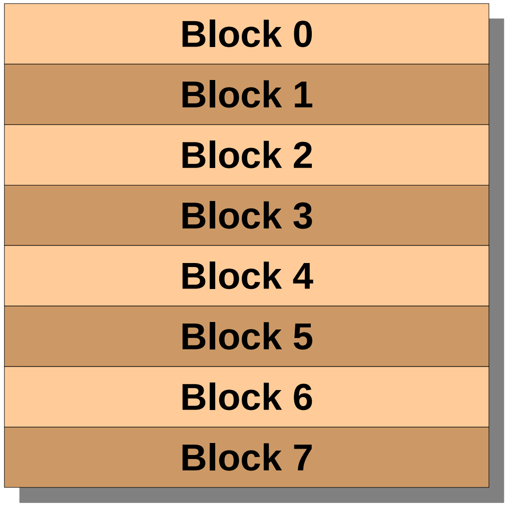
$$\boxed{2^{Index} = \frac{CacheSize}{BlockSize \times SetAssociativity}}$$

Set Associative Mapping

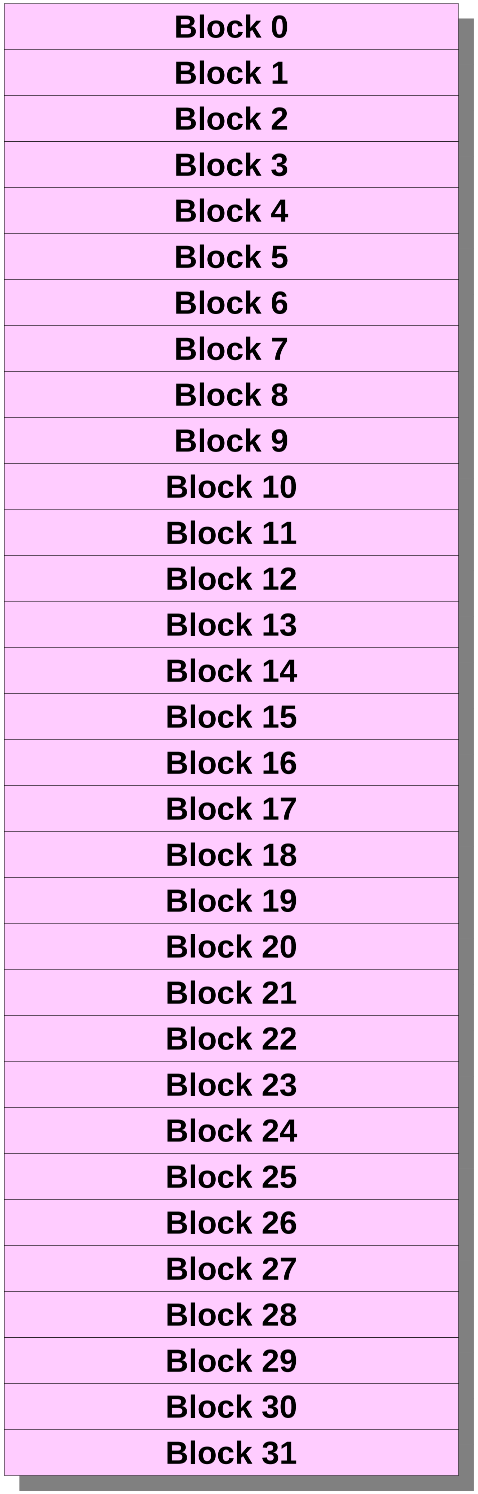
- 64KB 2-way SA cache with 32B cache lines, using 32b addresses. Address = 0xABCD1010. What is the index field?

2-way SA Mapping

1024B Main Memory; 256B cache; 32B block size.



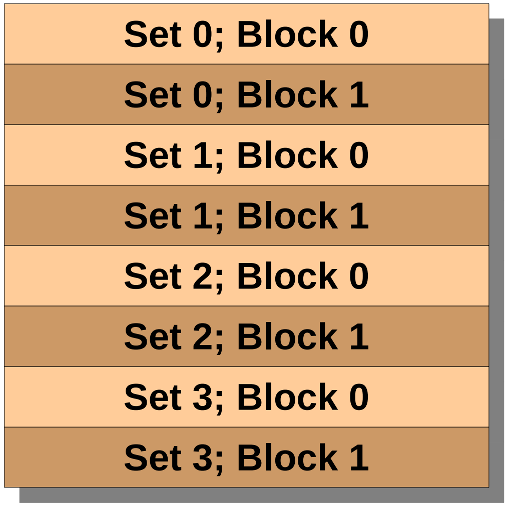
Block 0
Block 1
Block 2
Block 3
Block 4
Block 5
Block 6
Block 7



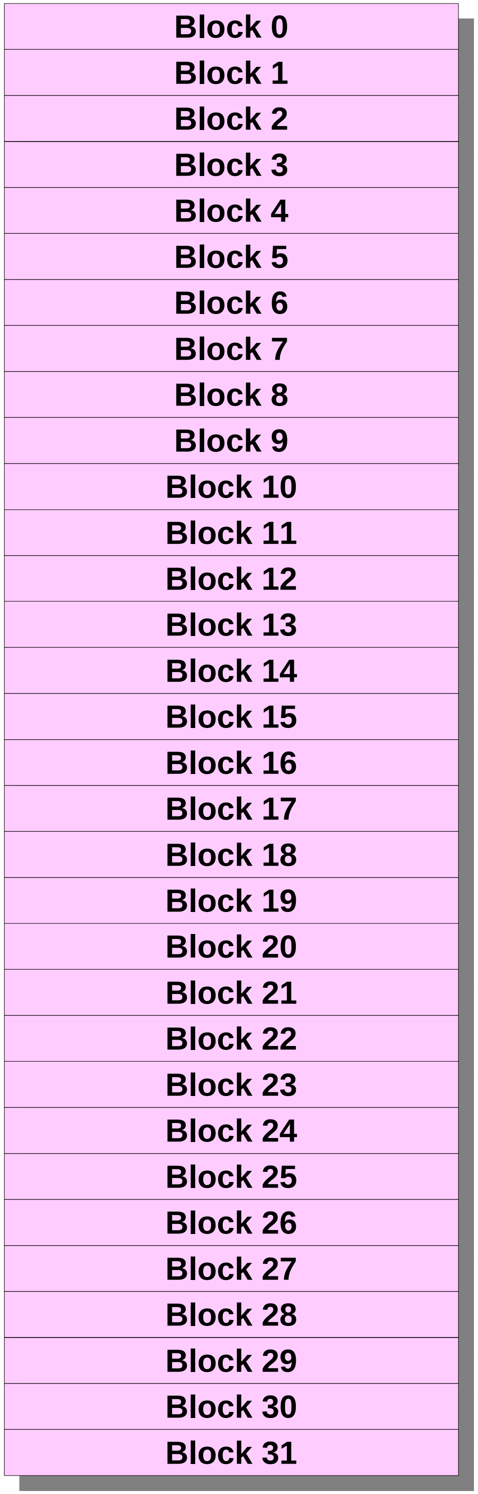
Block 0
Block 1
Block 2
Block 3
Block 4
Block 5
Block 6
Block 7
Block 8
Block 9
Block 10
Block 11
Block 12
Block 13
Block 14
Block 15
Block 16
Block 17
Block 18
Block 19
Block 20
Block 21
Block 22
Block 23
Block 24
Block 25
Block 26
Block 27
Block 28
Block 29
Block 30
Block 31

2-way SA Mapping

1024B Main Memory; 256B cache; 32B block size.



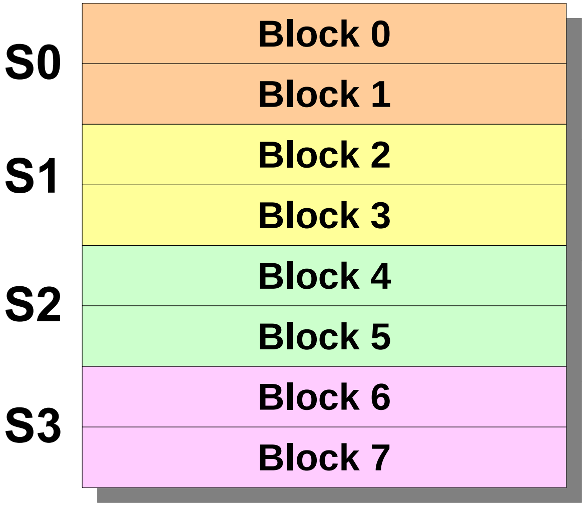
Set 0; Block 0
Set 0; Block 1
Set 1; Block 0
Set 1; Block 1
Set 2; Block 0
Set 2; Block 1
Set 3; Block 0
Set 3; Block 1



Block 0
Block 1
Block 2
Block 3
Block 4
Block 5
Block 6
Block 7
Block 8
Block 9
Block 10
Block 11
Block 12
Block 13
Block 14
Block 15
Block 16
Block 17
Block 18
Block 19
Block 20
Block 21
Block 22
Block 23
Block 24
Block 25
Block 26
Block 27
Block 28
Block 29
Block 30
Block 31

2-way SA Mapping

1024B Main Memory; 256B cache; 32B block size.



S0

Block 0

Block 1

S1

Block 2

Block 3

S2

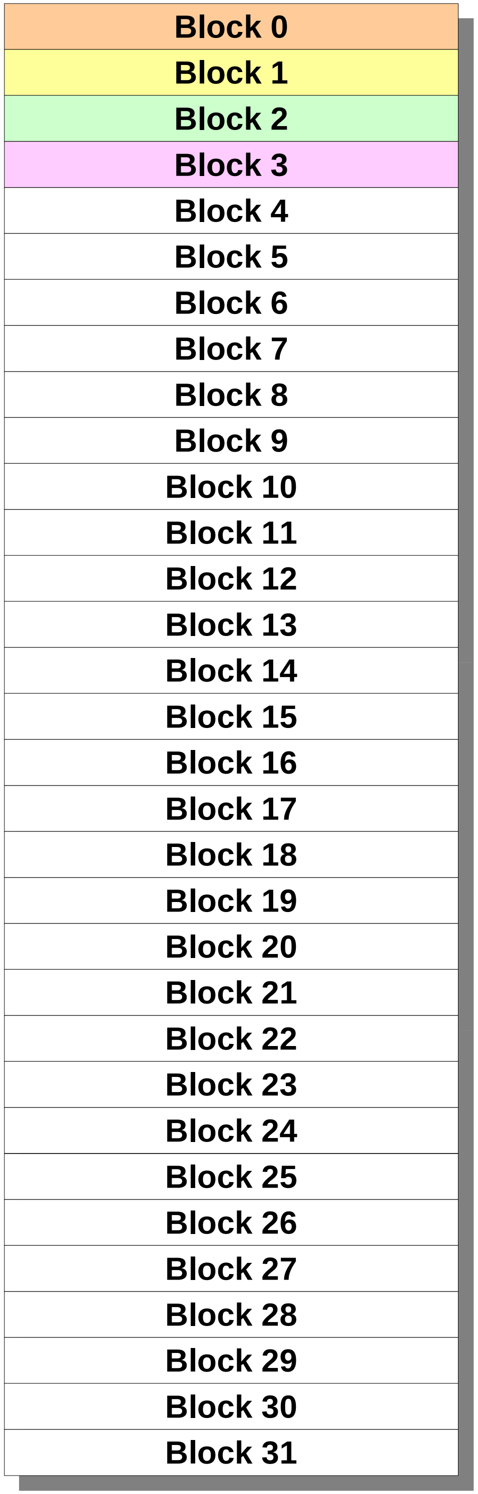
Block 4

Block 5

S3

Block 6

Block 7



Block 0

Block 1

Block 2

Block 3

Block 4

Block 5

Block 6

Block 7

Block 8

Block 9

Block 10

Block 11

Block 12

Block 13

Block 14

Block 15

Block 16

Block 17

Block 18

Block 19

Block 20

Block 21

Block 22

Block 23

Block 24

Block 25

Block 26

Block 27

Block 28

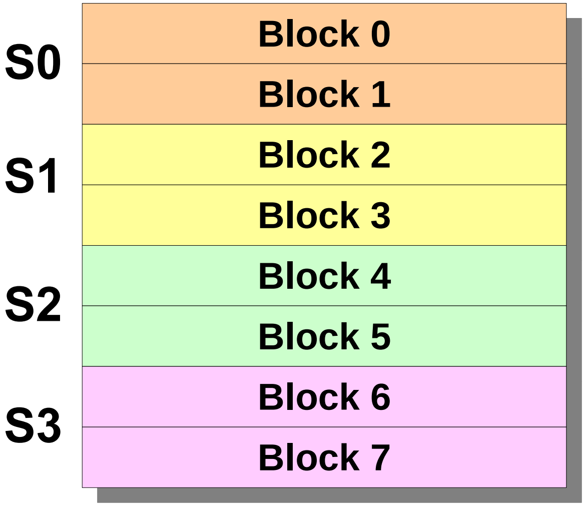
Block 29

Block 30

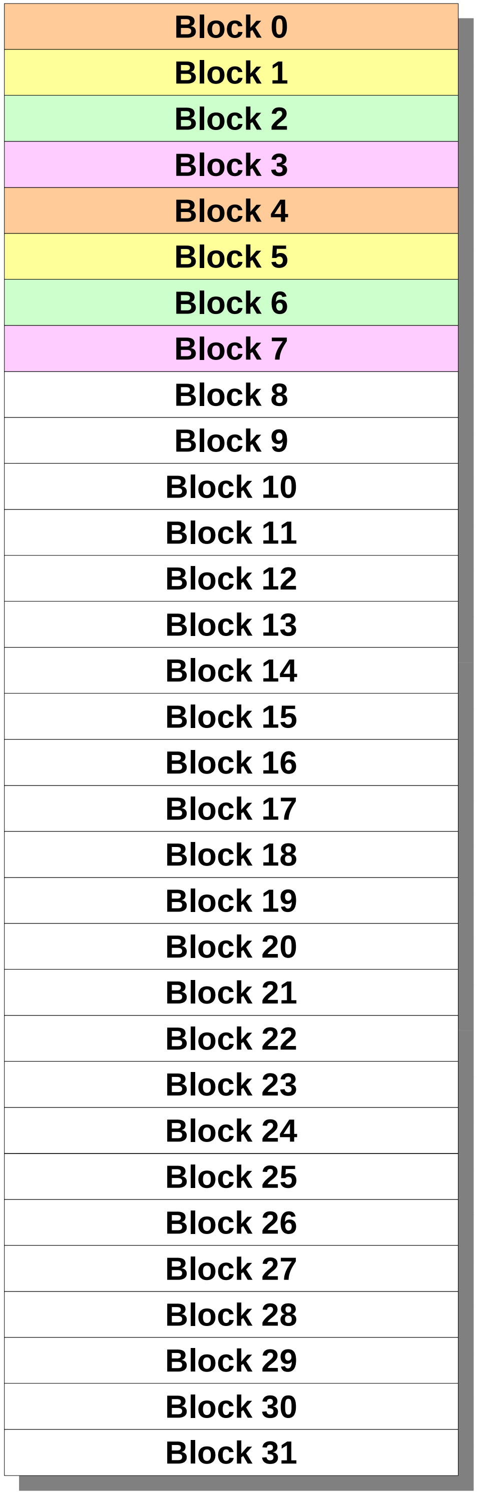
Block 31

2-way SA Mapping

1024B Main Memory; 256B cache; 32B block size.



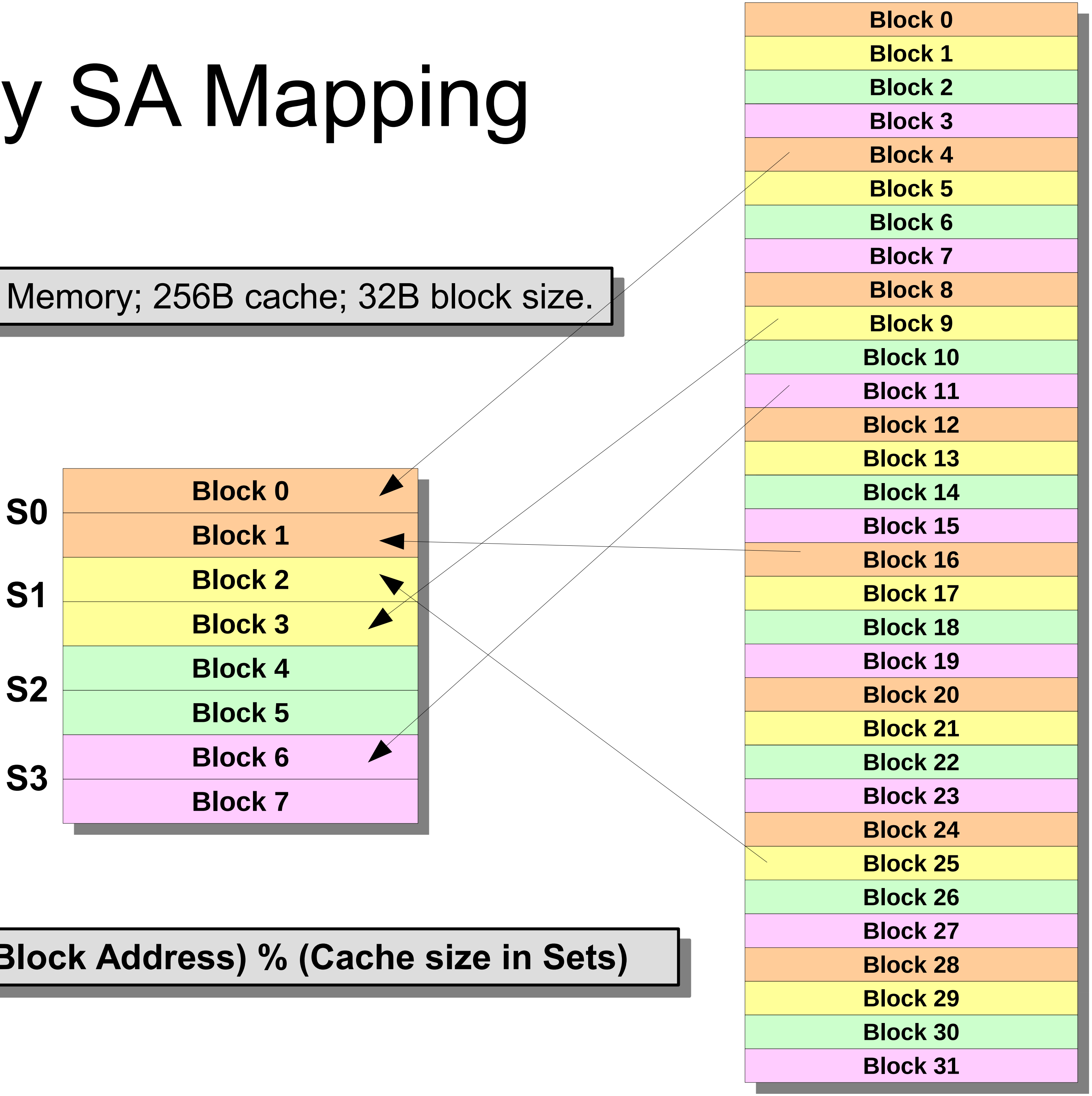
S0
Block 0
Block 1
S1
Block 2
Block 3
S2
Block 4
Block 5
S3
Block 6
Block 7



Block 0
Block 1
Block 2
Block 3
Block 4
Block 5
Block 6
Block 7
Block 8
Block 9
Block 10
Block 11
Block 12
Block 13
Block 14
Block 15
Block 16
Block 17
Block 18
Block 19
Block 20
Block 21
Block 22
Block 23
Block 24
Block 25
Block 26
Block 27
Block 28
Block 29
Block 30
Block 31

2-way SA Mapping

1024B Main Memory; 256B cache; 32B block size.



S0

Block 0

Block 1

S1

Block 2

Block 3

S2

Block 4

Block 5

S3

Block 6

Block 7

Block 0

Block 1

Block 2

Block 3

Block 4

Block 5

Block 6

Block 7

Block 8

Block 9

Block 10

Block 11

Block 12

Block 13

Block 14

Block 15

Block 16

Block 17

Block 18

Block 19

Block 20

Block 21

Block 22

Block 23

Block 24

Block 25

Block 26

Block 27

Block 28

Block 29

Block 30

Block 31

Index = (Block Address) % (Cache size in Sets)

Contents of the SA Cache

- The following cache blocks are accessed in sequence. Show the state of the 2-way SA Cache after each block is placed in the cache. Assume that the cache is empty at start. Total sets in the Cache = 4.
- 10, 23, 13, 8, 18, 27

Contents of the DM Cache

010	Contents of Block 10

10 mod 4 = 2

10, 23, 13, 8, 18, 27

Contents of the DM Cache

010	Contents of Block 10

10 mod 4 = 2

010	Contents of Block 10
101	Contents of Block 23

23 mod 4 = 3

10, 23, 13, 8, 18, 27

Contents of the DM Cache

010	Contents of Block 10

10 mod 4 = 2

010	Contents of Block 10
101	Contents of Block 23

23 mod 4 = 3

011	Contents of Block 13
010	Contents of Block 10
101	Contents of Block 23

13 mod 4 = 1

10, 23, 13, 8, 18, 27

Contents of the DM Cache

010	Contents of Block 10

10 mod 4 = 2

010	Contents of Block 10
101	Contents of Block 23

23 mod 4 = 3

011	Contents of Block 13
010	Contents of Block 10
101	Contents of Block 23

13 mod 4 = 1

010	Contents of Block 8
011	Contents of Block 13
010	Contents of Block 10
101	Contents of Block 23

8 mod 4 = 0

10, 23, 13, 8, 18, 27

Contents of the DM Cache

010	Contents of Block 10

10 mod 4 = 2

010	Contents of Block 10
101	Contents of Block 23

23 mod 4 = 3

011	Contents of Block 13
010	Contents of Block 10
101	Contents of Block 23

13 mod 4 = 1

010	Contents of Block 8
011	Contents of Block 13
010	Contents of Block 10
101	Contents of Block 23

8 mod 4 = 0

010	Contents of Block 8
011	Contents of Block 13
010	Contents of Block 10
100	Contents of Block 18
101	Contents of Block 23

18 mod 4 = 2

10, 23, 13, 8, 18, 27

Contents of the DM Cache

010	Contents of Block 10

10 mod 4 = 2

010	Contents of Block 10
101	Contents of Block 23

23 mod 4 = 3

011	Contents of Block 13
010	Contents of Block 10
101	Contents of Block 23

13 mod 4 = 1

010	Contents of Block 8
011	Contents of Block 13
010	Contents of Block 10
101	Contents of Block 23

8 mod 4 = 0

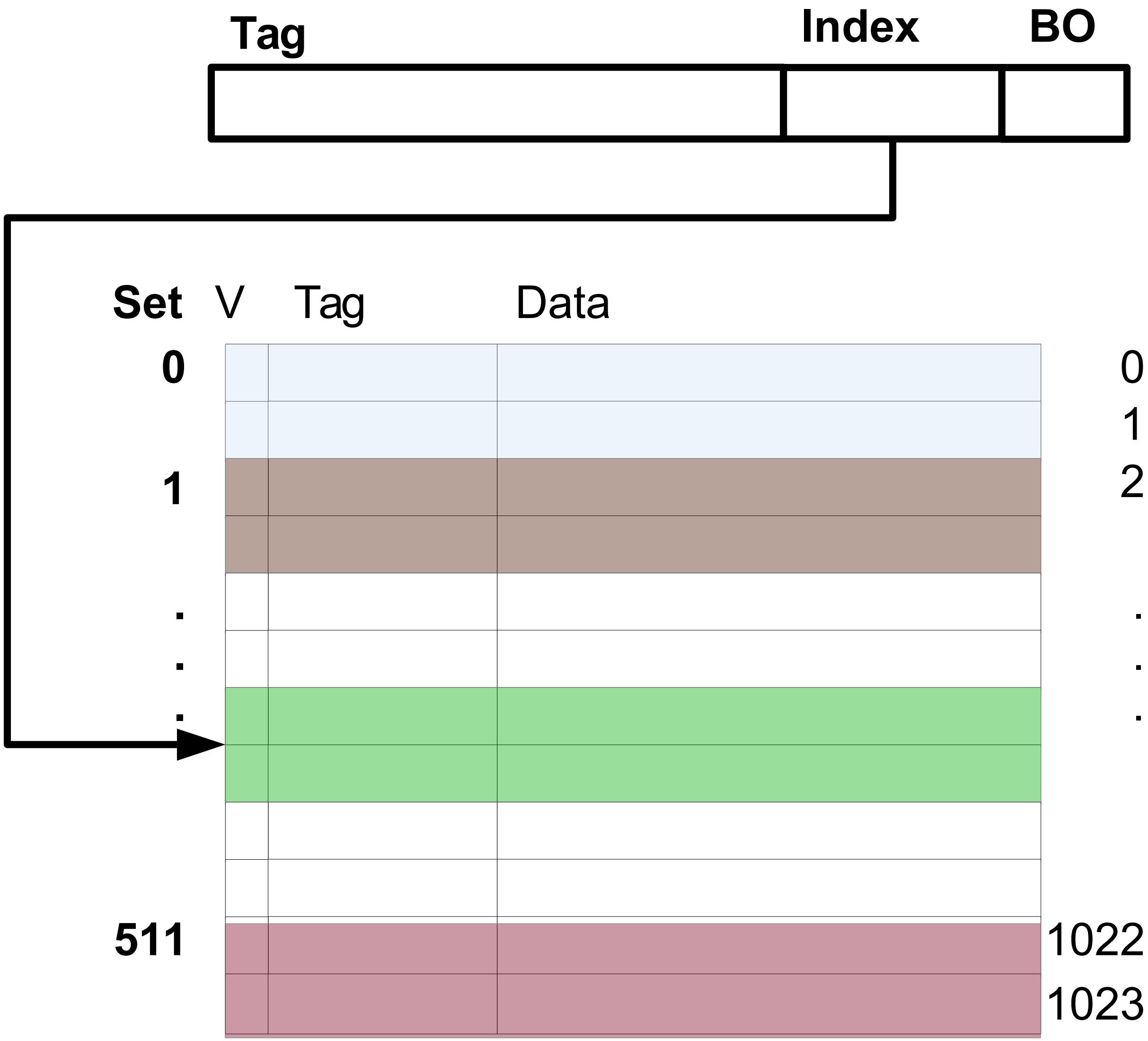
010	Contents of Block 8
011	Contents of Block 13
010	Contents of Block 10
100	Contents of Block 18
101	Contents of Block 23

18 mod 4 = 2

010	Contents of Block 8
011	Contents of Block 13
010	Contents of Block 10
100	Contents of Block 18
101	Contents of Block 23
110	Contents of Block 27

27 mod 4 = 3

Block Placement



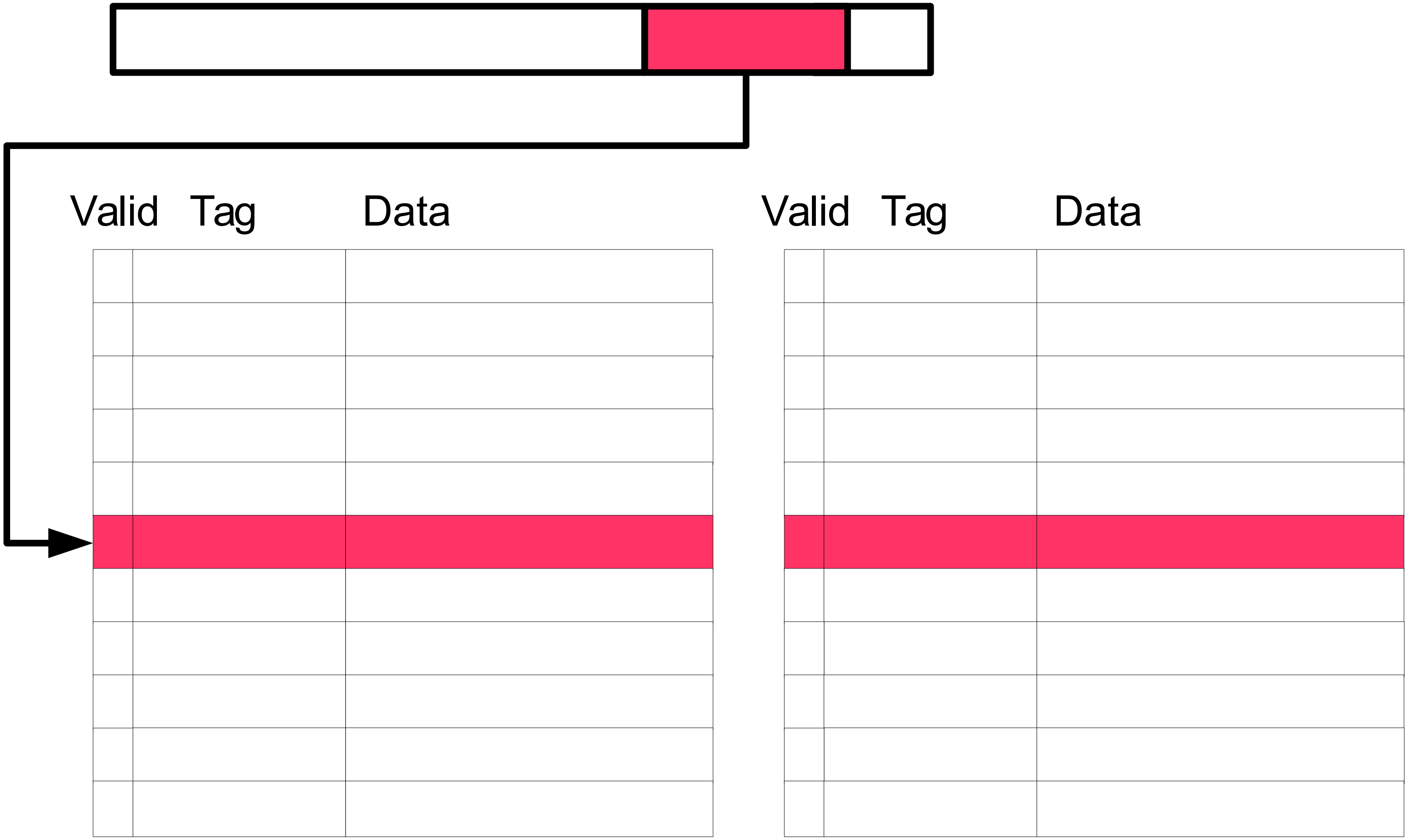
Set Associative Cache

Index bits Identify a unique SET

A Set contains multiple cache lines

2-way Set Associative Cache

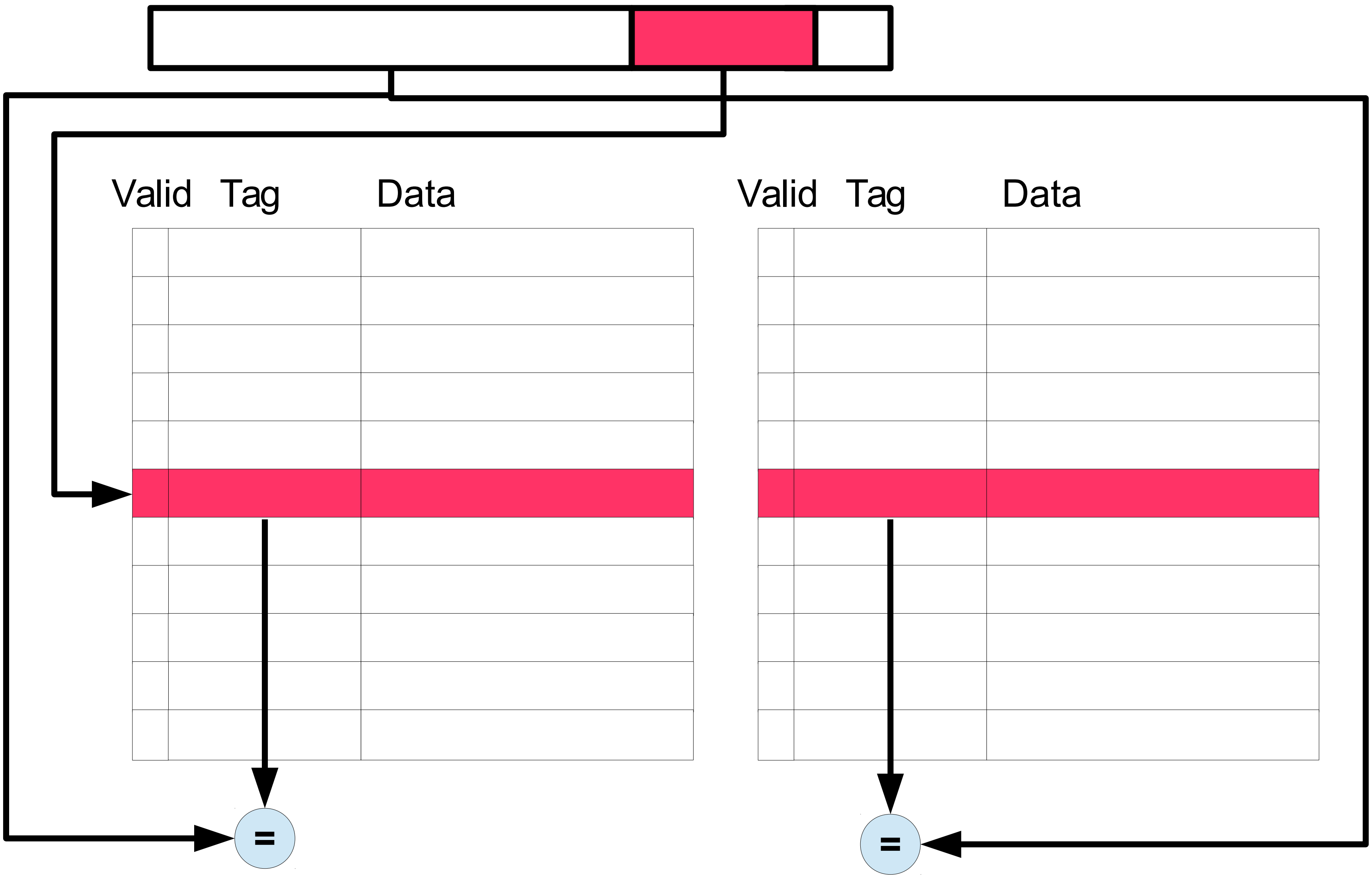
2–way Set Associative Cache



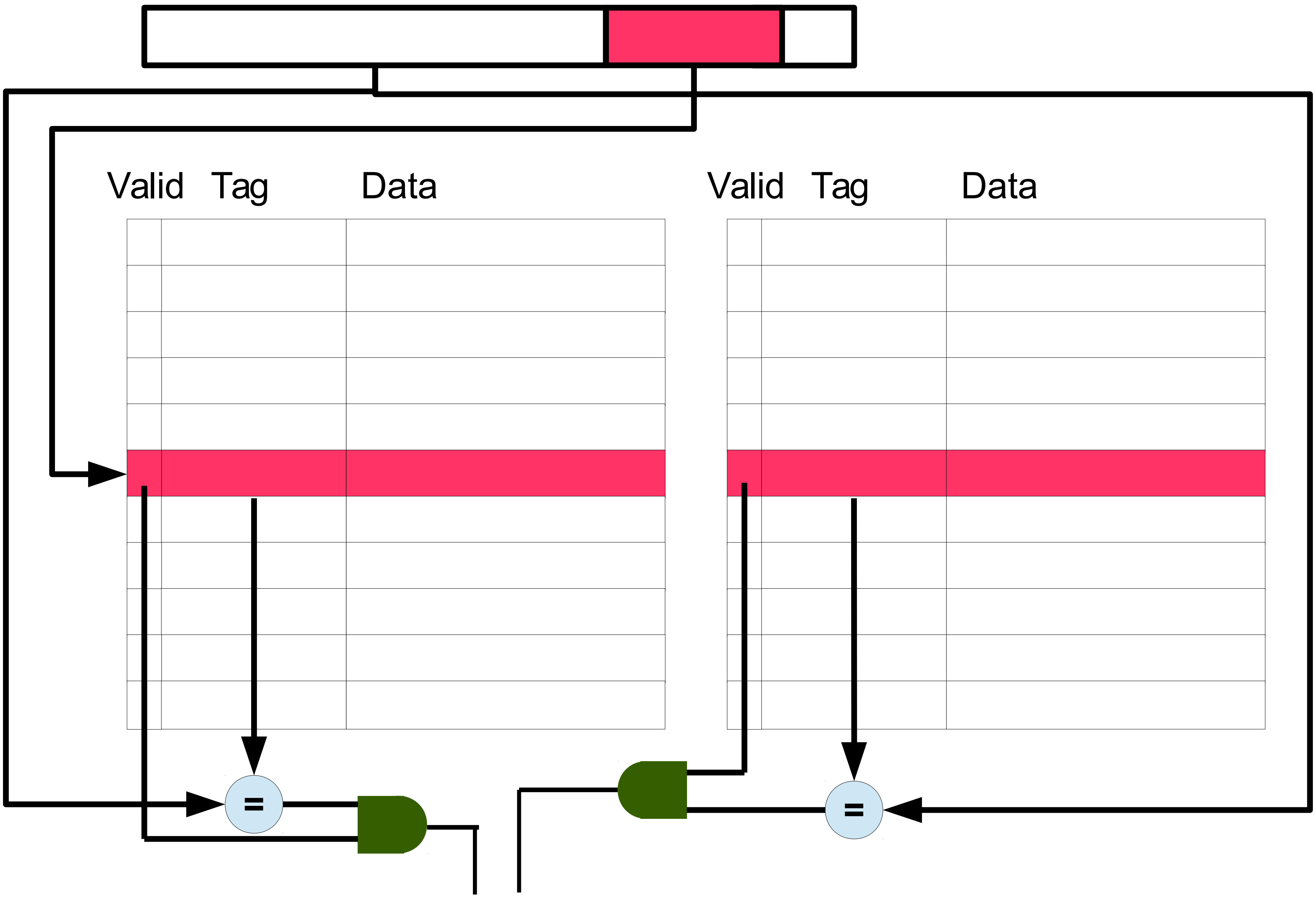
Valid Tag Data

Valid Tag Data

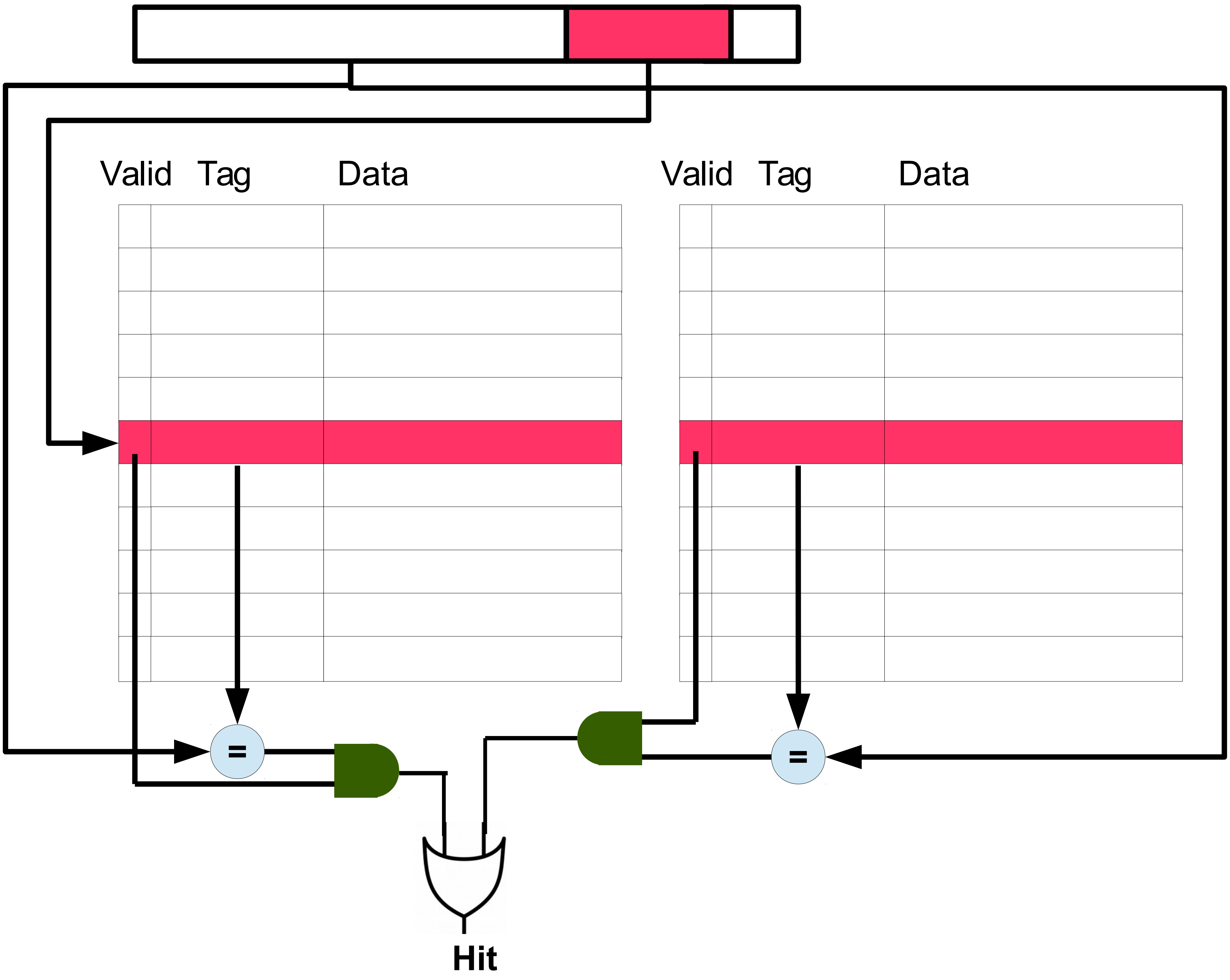
2–way Set Associative Cache



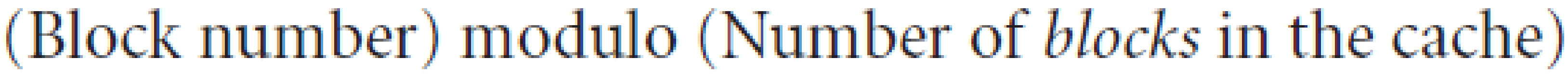
2–way Set Associative Cache



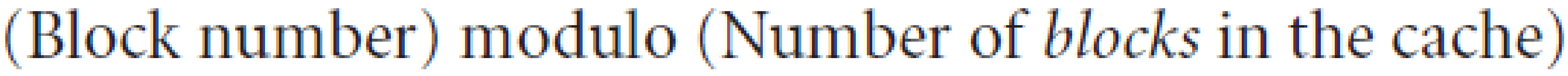
2–way Set Associative Cache



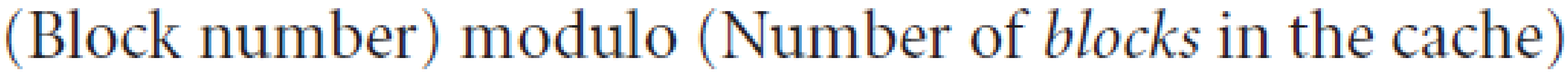
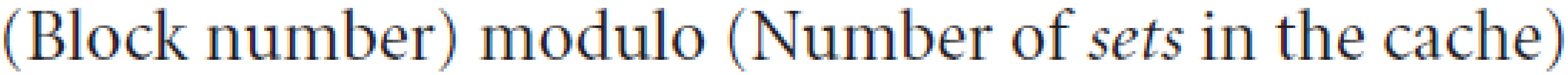
Block Placement

- Direct Mapped Cache
  - A block can be placed in exactly one location in the cache
  - 
    (Block number) modulo (Number of *blocks* in the cache)

Block Placement

- Direct Mapped Cache
  - A block can be placed in exactly one location in the cache
  - 
    (Block number) modulo (Number of *blocks* in the cache)
- Fully Associative Cache
  - A block can be placed in any location in the cache

Block Placement

- Direct Mapped Cache
  - A block can be placed in exactly one location in the cache
  - 
    (Block number) modulo (Number of *blocks* in the cache)
- Fully Associative Cache
  - A block can be placed in any location in the cache
- Set Associative Cache
  - A block can be placed in any location inside a set in the cache
  - 
    (Block number) modulo (Number of *sets* in the cache)

Examples

Consider a cache with 64 blocks and a block size of 16 bytes. To what block number does byte address 1200 map?

- DM Cache
- 4-way SA Cache
- Fully SA CAche

Examples

Consider a cache with 64 blocks and a block size of 16 bytes. To what block number does byte address 1200 map?

- DM Cache
- 4-way SA Cache
- Fully SA CAche

What is the size of the Cache RAM (in bits)? 16 KB of data. 32-bit address.

1. Direct-mapped cache with 4-word blocks.
2. 2-way Set Associative cache with 8-word blocks
3. Fully Set Associative with 8-word blocks

Block Replacement

- Which block should be evicted when a new block is about to be fetched into the cache?
  - In Direct Mapped Cache?

Block Replacement

- Which block should be evicted when a new block is about to be fetched into the cache?
  - In Direct Mapped Cache?
- Least Recently Used (LRU)

Block Replacement

- Which block should be evicted when a new block is about to be fetched into the cache?
  - In Direct Mapped Cache?
- Least Recently Used (LRU)
- First in First Out (FIFO)

Block Replacement

- Which block should be evicted when a new block is about to be fetched into the cache?
  - In Direct Mapped Cache?
- Least Recently Used (LRU)
- First in First Out (FIFO)
- Random Replacement Policy

Cache Writes

- On a Write Hit

Cache Writes

- On a Write Hit
  - When does the cache update the modified block in the lower level?

Cache Writes

- On a Write Hit
  - When does the cache update the modified block in the lower level?
  - As soon as a write occurs: **Write through policy**.
  - When the block is replaced: **Write back policy**.

Cache Writes

- On a Write Hit
  - When does the cache update the modified block in the lower level?
  - As soon as a write occurs: **Write through policy**.
    - Large stall time
    - Write buffer
  - When the block is replaced: **Write back policy**.
    - Multiple writes within a block require only one write to the lower level in the hierarchy.

Cache Writes

- On a Write Miss
  - –
  - –

Cache Writes

- On a Write Miss
  - **No write-allocate**: write the data to memory only.
  - **Write-allocate**: read the entire block into the cache; update the word in the cache

Four Memory Hierarchy Questions

- Block Placement
  - –
  - –

Four Memory Hierarchy Questions

- Block Placement
  - Where can a block be placed in a cache?
  - Direct mapped, Set associative, Fully associative

Four Memory Hierarchy Questions

- Block Identification
  - How is a block found if it is in cache?

Four Memory Hierarchy Questions

- Block Replacement
  - Which block should be replaced on a miss?

Four Memory Hierarchy Questions

- Write Strategy
  - What happens on a write?