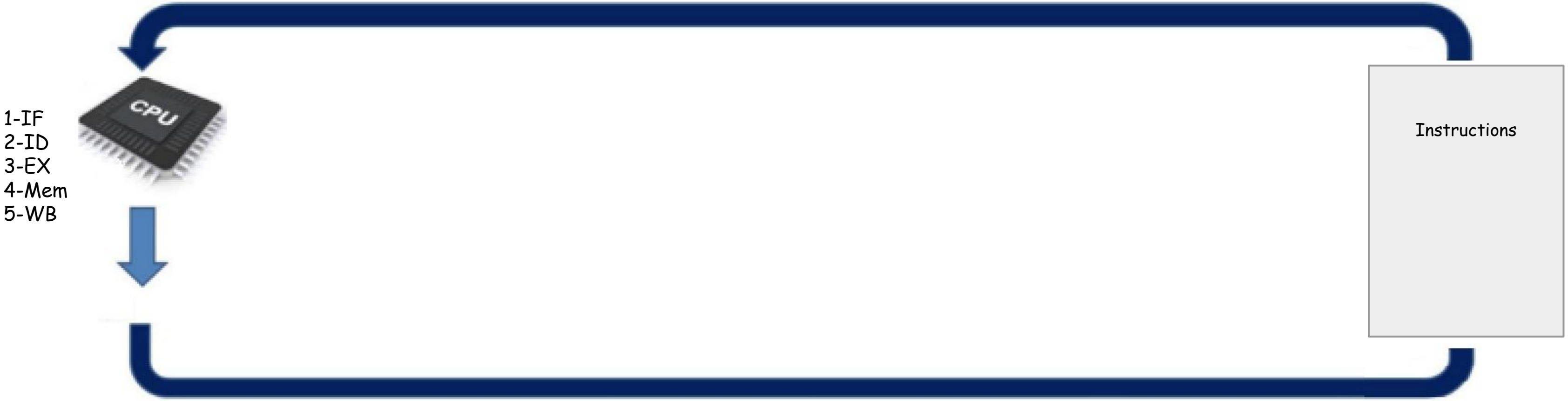
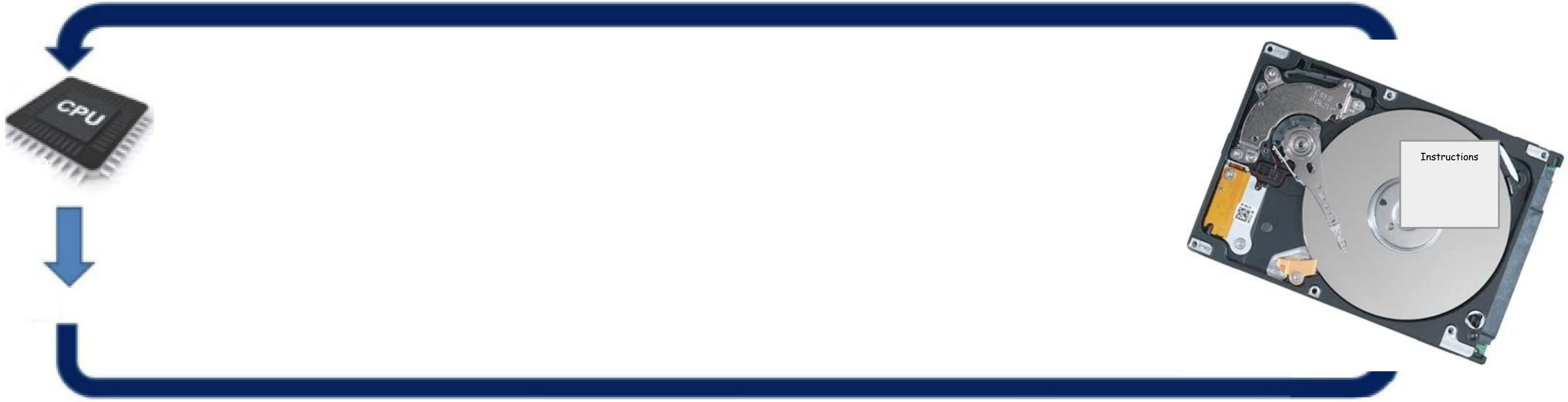
Cache Memory



1-IF
2-ID
3-EX
4-Mem
5-WB

Instructions

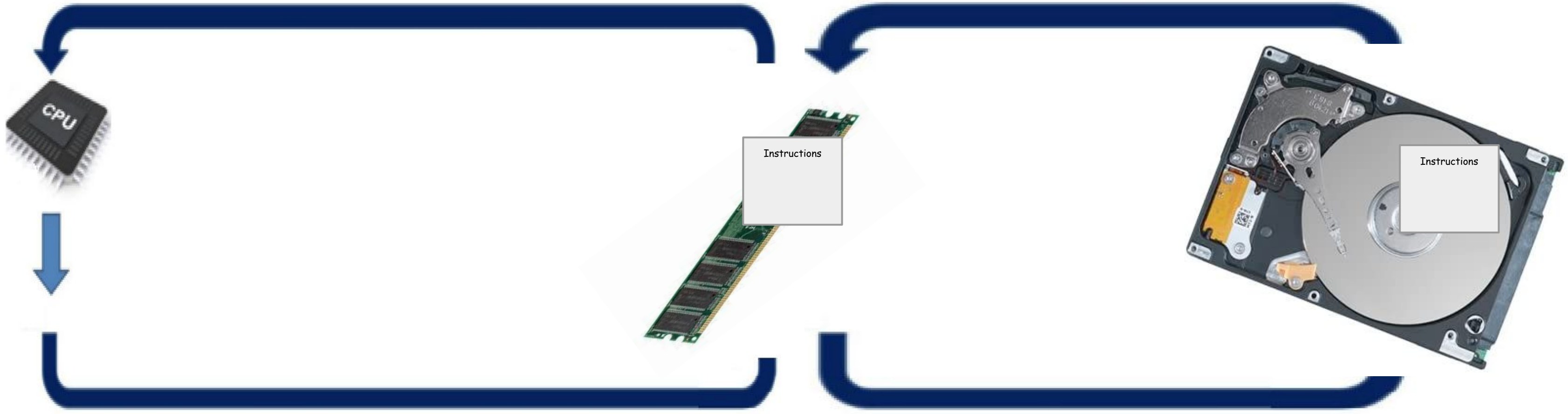
1-IF
2-ID
3-EX
4-Mem
5-WB



0.5ns

5~ 10 millisecond
($5x10^6$ ns)

The Von Neumann Bottleneck



1-IF
2-ID
3-EX
4-Mem
5-WB

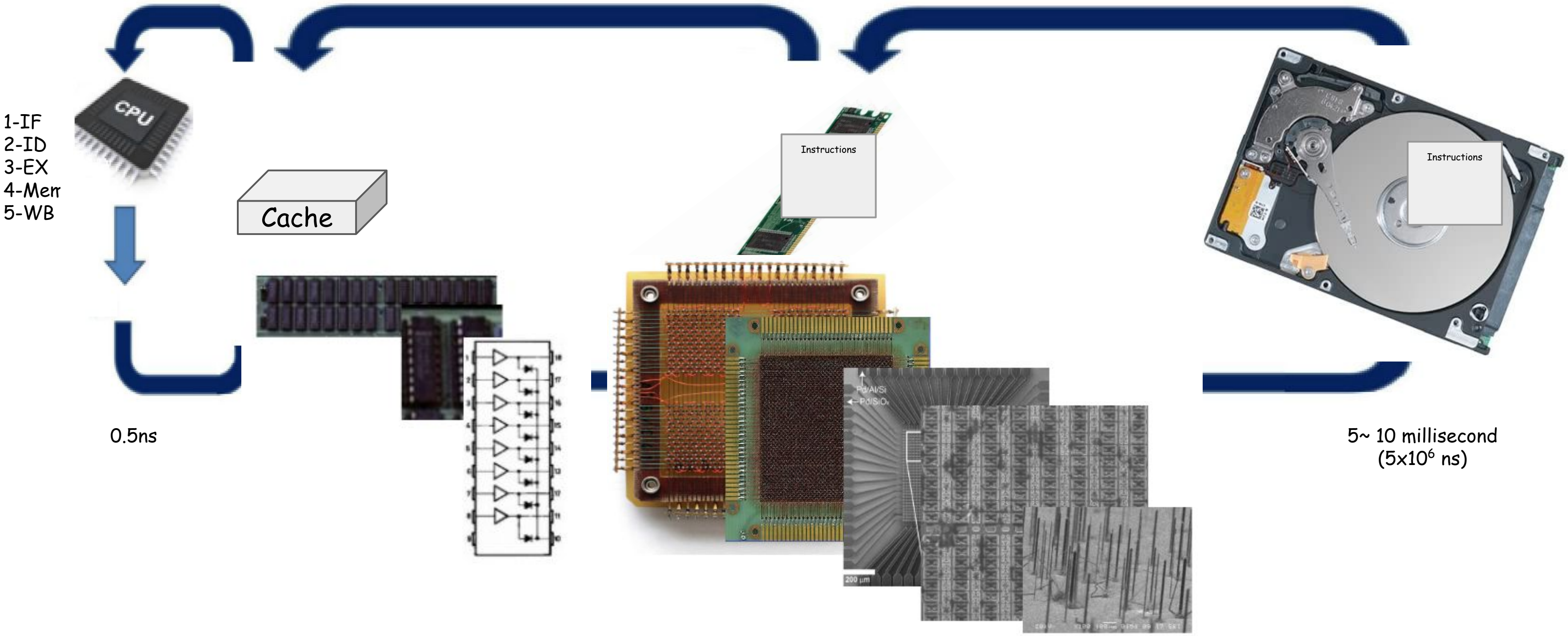
Instructions

Instructions

0.5ns

50 ns

5~ 10 millisecond
($5x10^6$ ns)

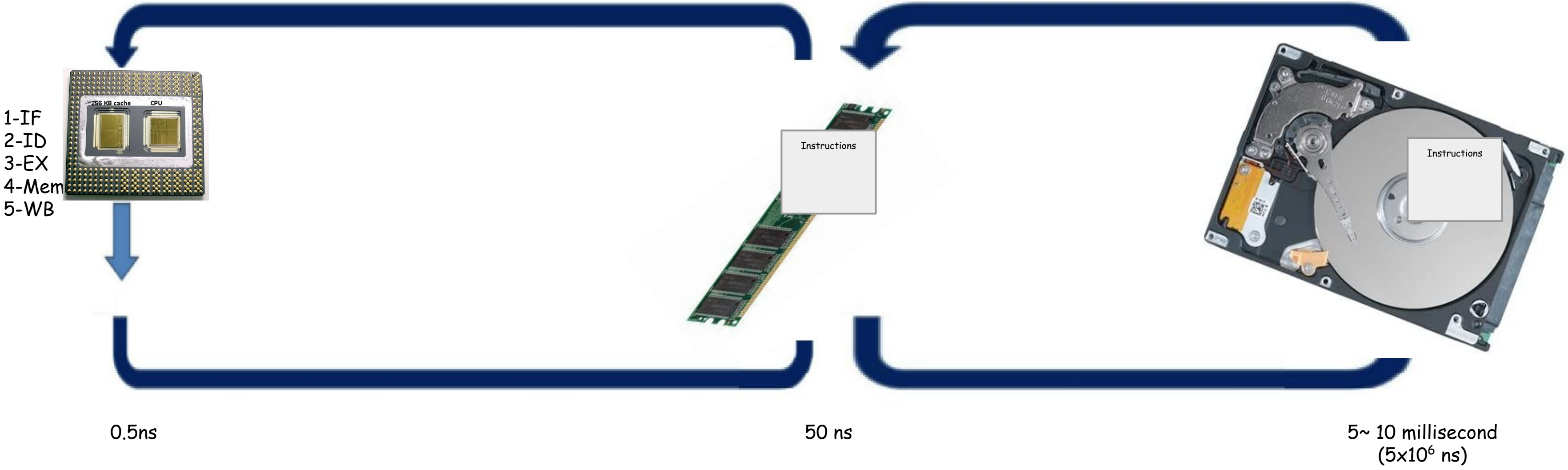


1-IF
2-ID
3-EX
4-Mem
5-WB

Cache

0.5ns

5~ 10 millisecond
(5x10^6 ns)



Spatial Locality: (Need for adjacent Data)

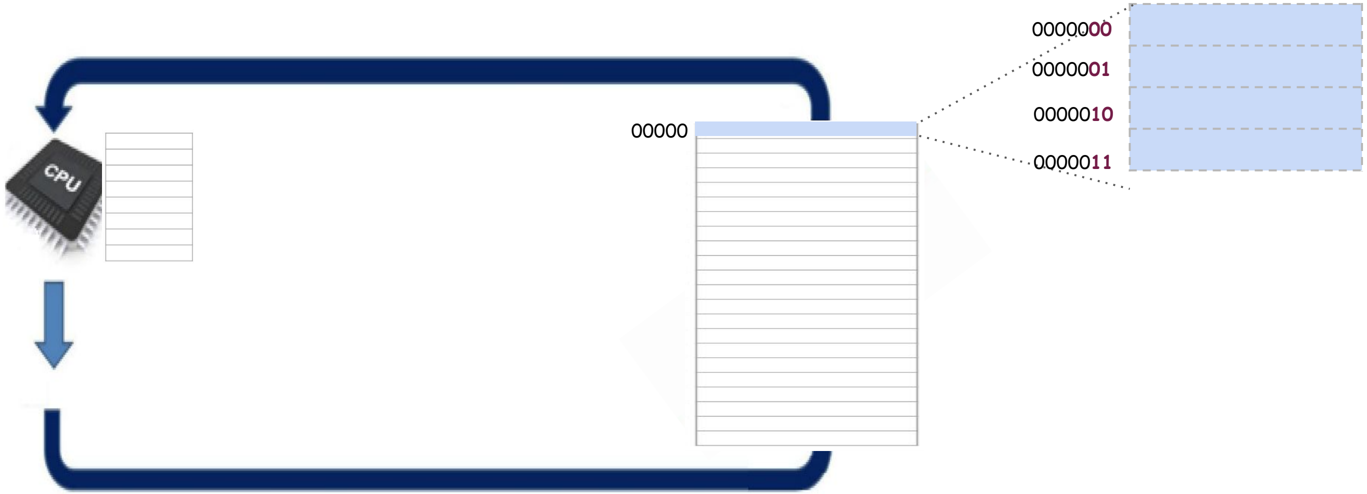
Temporal Locality: (Need to be close to data for some time)

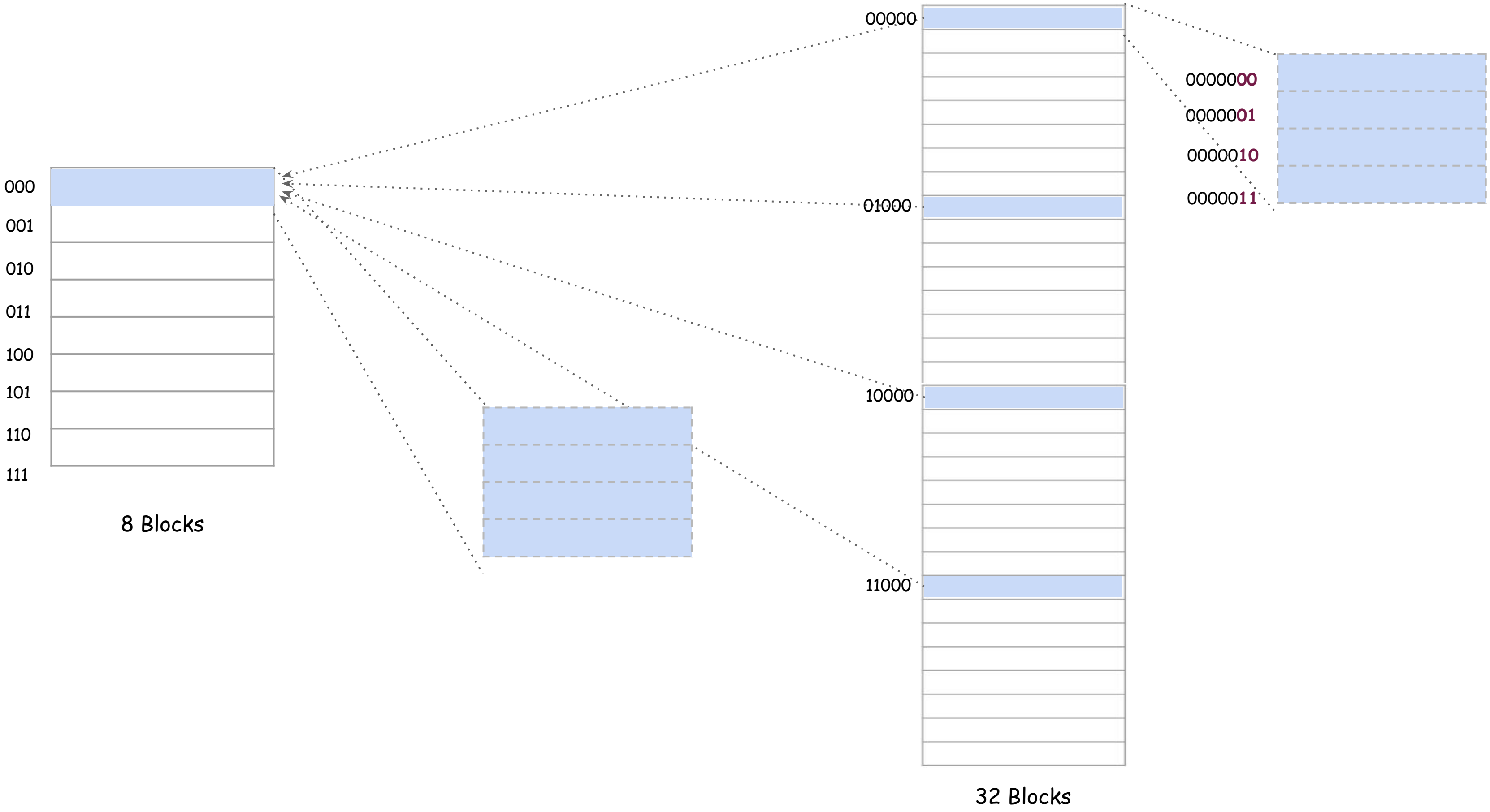
```
int a={3,4,5,6,7,1,2,3,8,3}

total=0

for(int i=0; i<10;i++){

        total+=a[i]
```

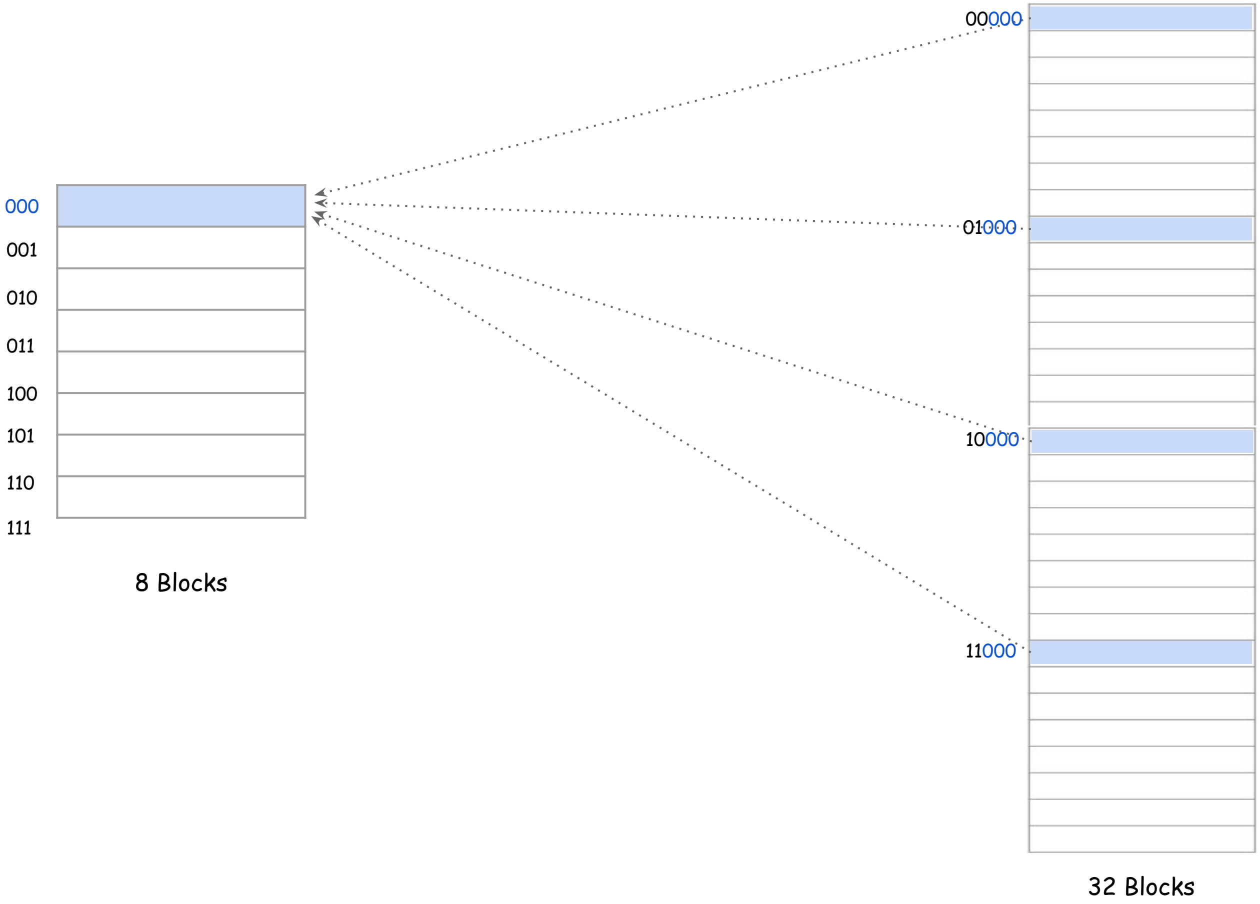
Cache is a small amount of memory that's on the CPU itself or right next to it.

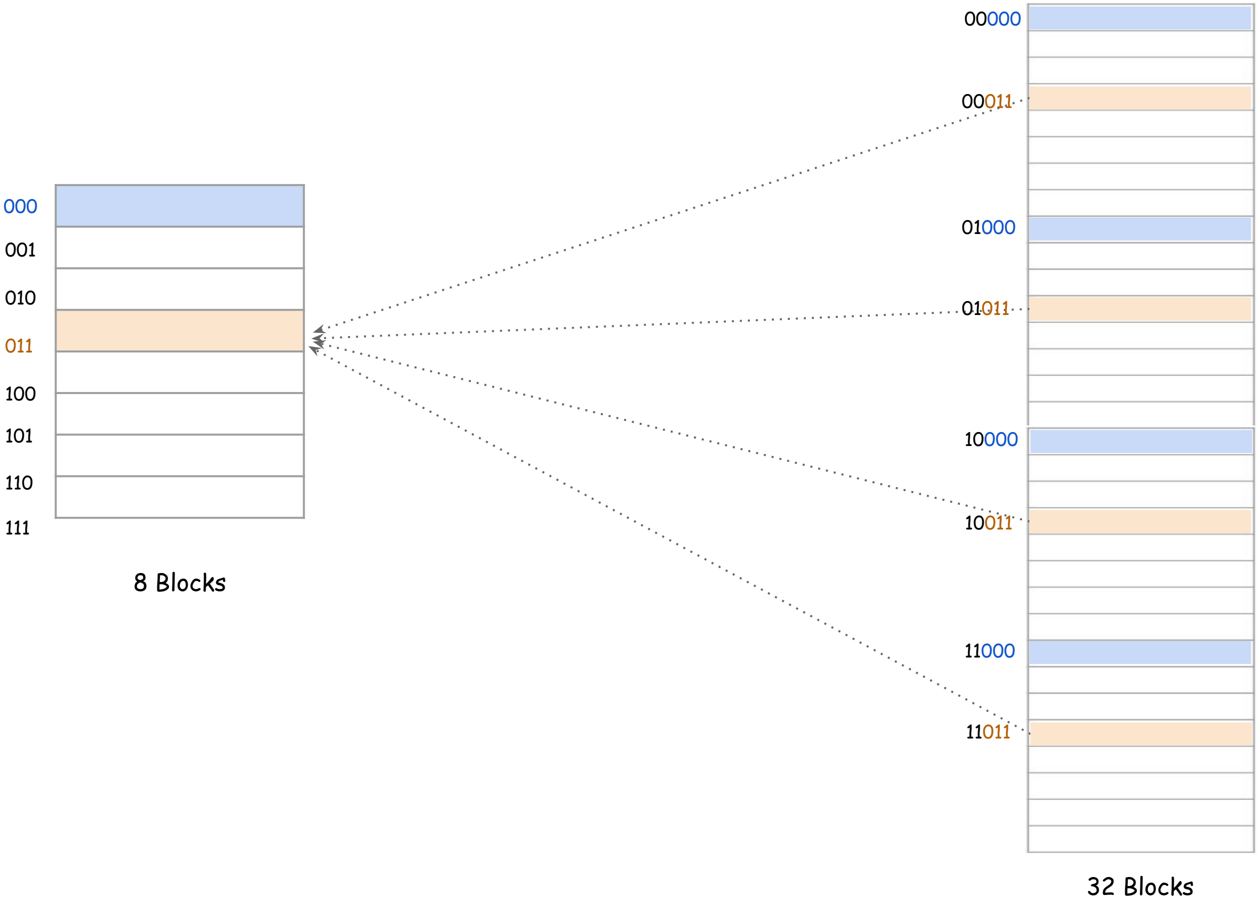
It can provide the cpu with same of its speed.

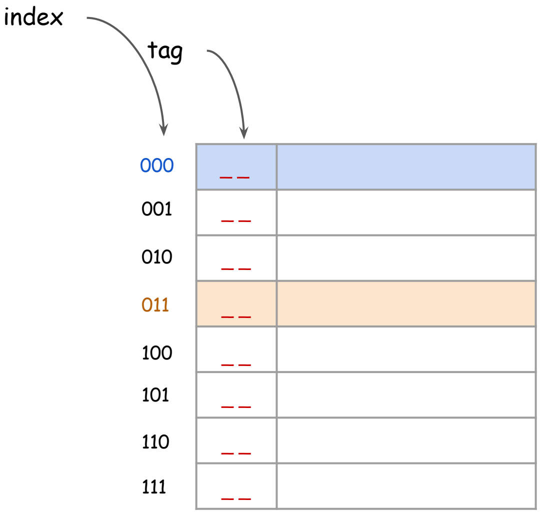
- It stores a copy of info. From the main memory.
- CPU asks cache if yes (cache hit) if not (cache miss)
- The greater the cache hits ⇒ the greater the performance
- The greater the cache misses ⇒ the lower the performance


CPU
00000
0000000
0000001
0000010
0000011

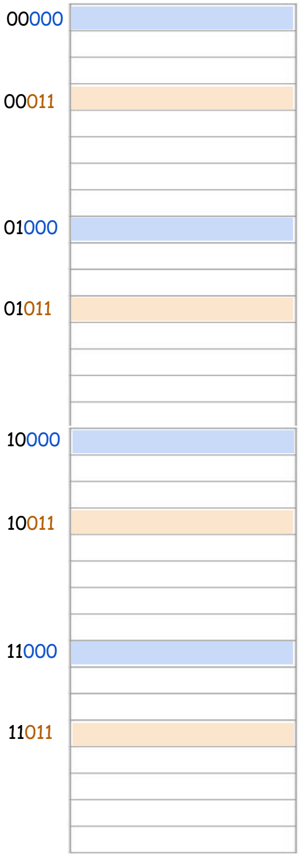

000
001
010
011
100
101
110
111
8 Blocks
00000
01000
10000
11000
32 Blocks
0000000
0000001
0000010
0000011


000
001
010
011
100
101
110
111
8 Blocks
00000
01000
10000
11000
32 Blocks

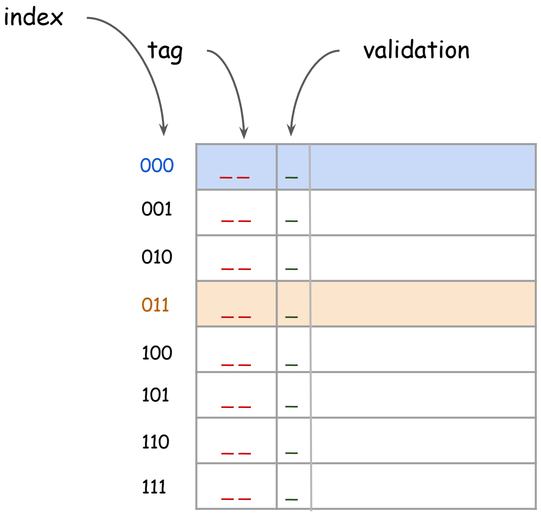

000
001
010
011
100
101
110
111
8 Blocks
00000
00011
01000
01011
10000
10011
11000
11011
32 Blocks



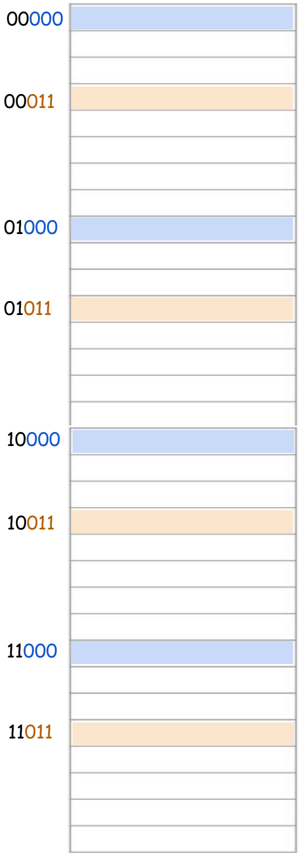
8 Blocks



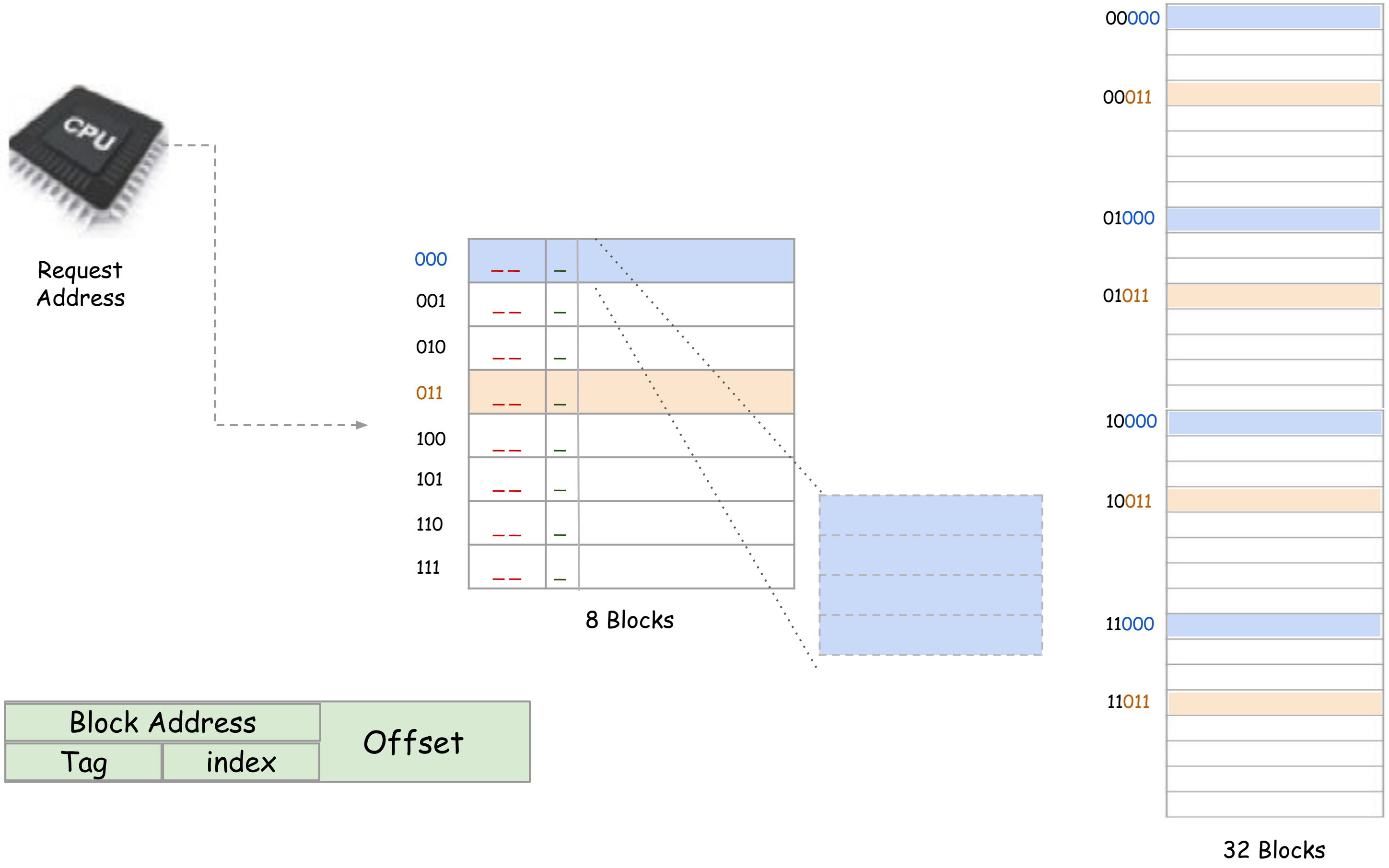
32 Blocks



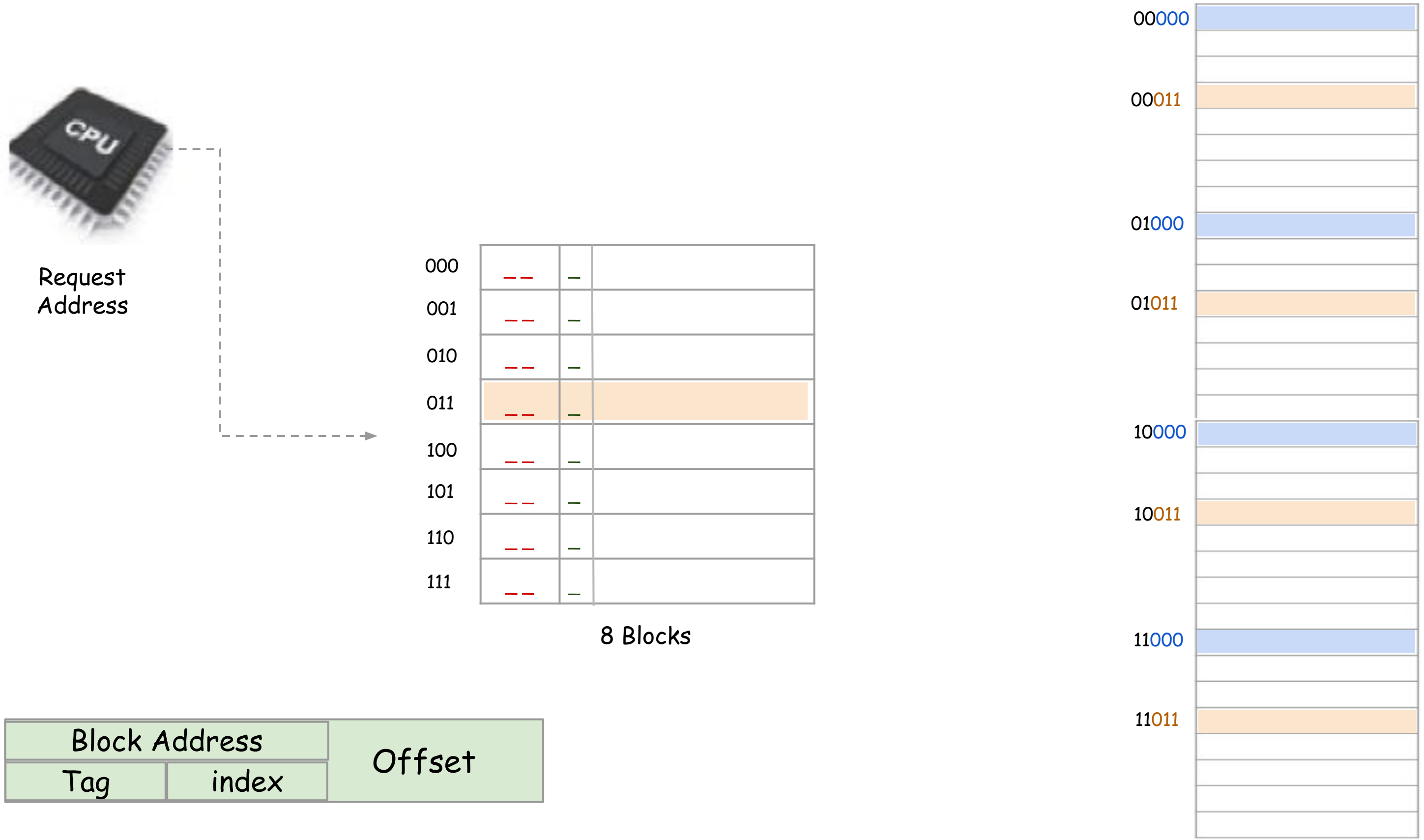
8 Blocks



32 Blocks



Block Address		Offset
Tag	index	



Given ByteAddress (0x59), find its location in cache (which block address)?
(Each block has 4 bytes, and the cache has 8 Blocks in total)

Exercise:

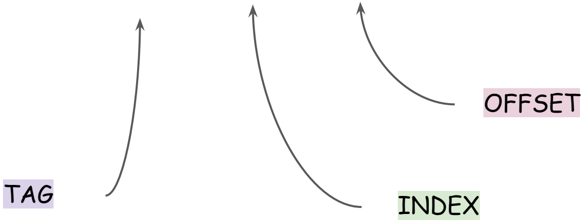
Consider a 64 Blocks cache and a block size of 16 bytes and address length is 16 bits. To which Block number does byte address 0x4B0 map?

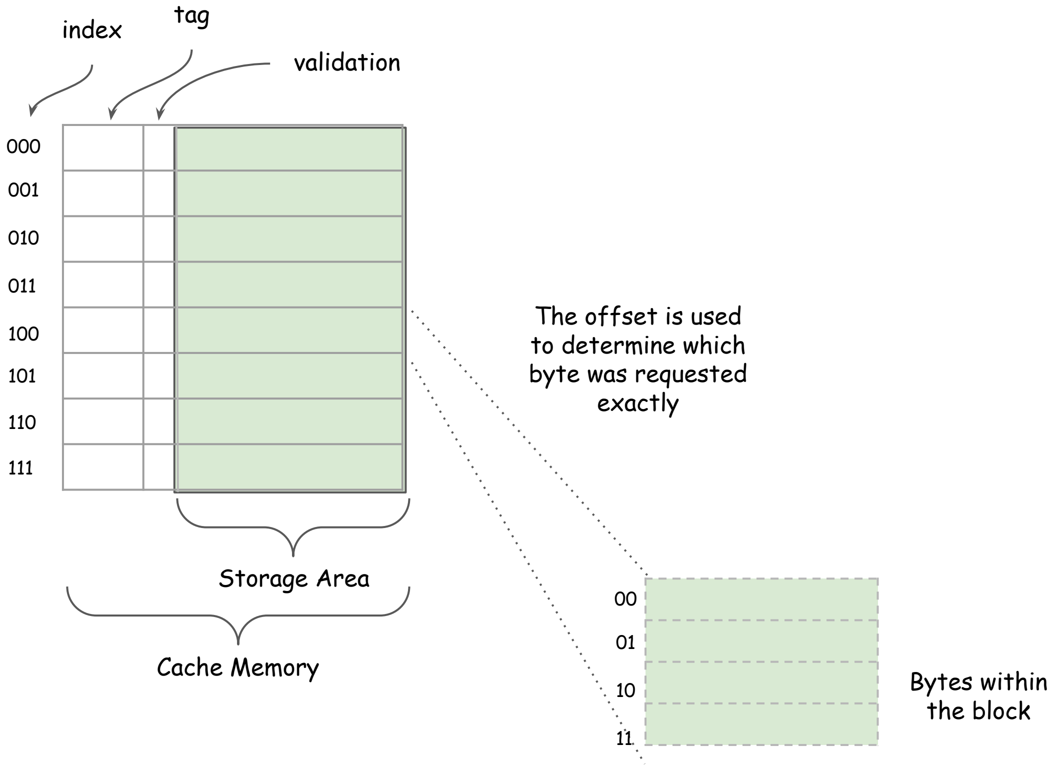
Exercise Solution:

CacheSize is 64 blocks = 2^6 ⇒ so the INDEX is 6 bits

BlockSize is 16 bytes = 2^4 ⇒ so the OFFSET is 4 bits

Address is $(4B0)_{hex}$ =$(0000010010110000)_2$ ⇒ the length of the address is 16 ⇒ tag is 6 bits





The offset is used to determine which byte was requested exactly

Exercise:

How many total bits are required for a direct-mapped cache with 16KB of data and 4-word blocks, assuming 32-bit address (word size=32 bits)?

Exercise Solution:

BlockSize = 4 words = 4 x (32 bits) = 16 bytes = 2^4 ⇒ the OFFSET is 4

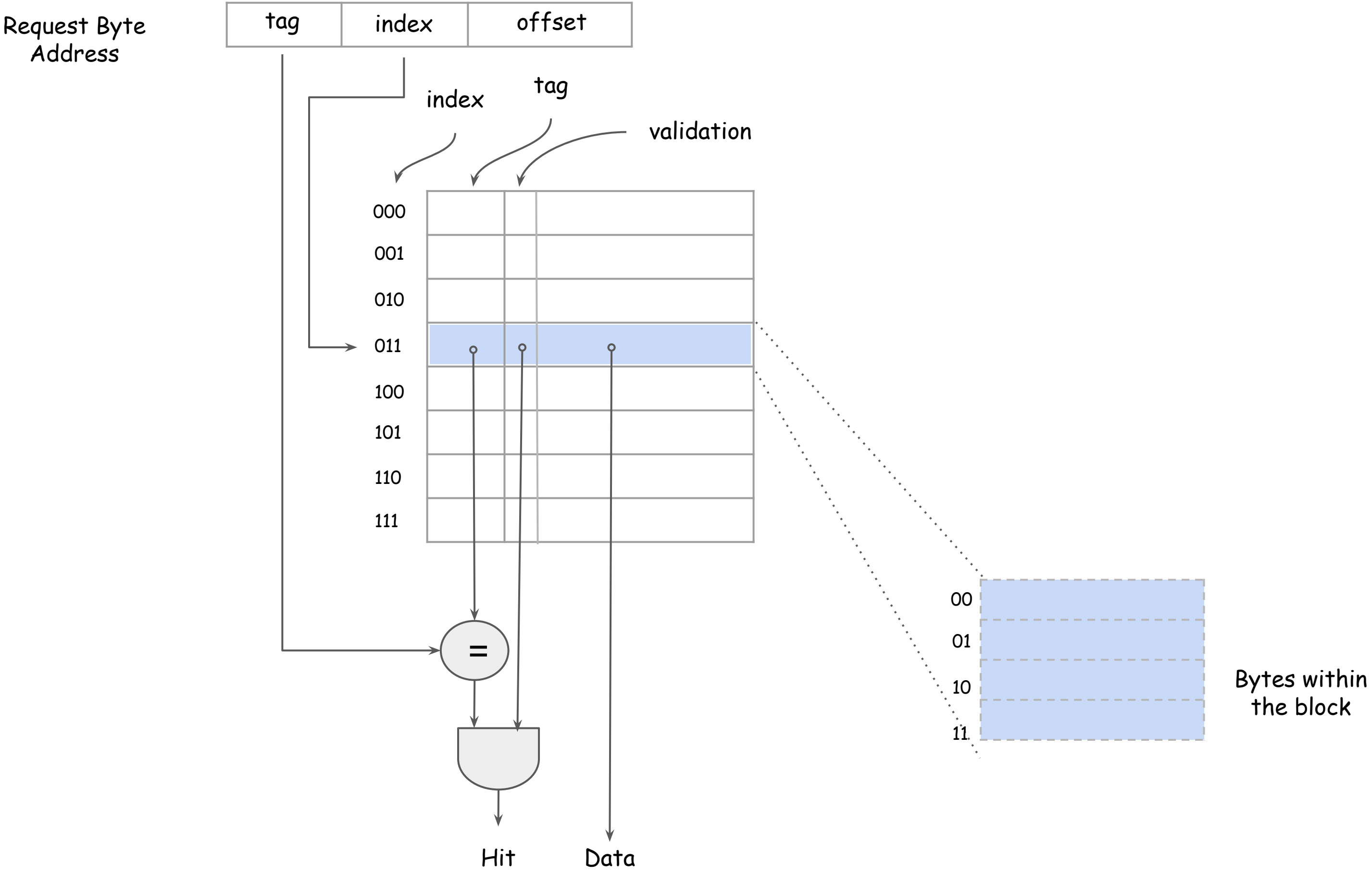
CacheSize = 16kB, Blocksize = 16 bytes ⇒ No. of blocks in cache = 16kB/16 = 2^{10} ⇒ the INDEX is 10 bits

Address = TAG + INDEX + OFFSET ⇒ TAG = 32 - (10 + 4) = 18 bits

⇒ Total bits in the cache = (BlockSizeinBits + validationBit + TAG) x (No. of Blocks)

= $(128 + 1 + 18) \times 2^{10}$

= 147 kbits


Request Byte Address
tag
index
offset
index
tag
validation
000
001
010
011
100
101
110
111
=
Hit
Data
00
01
10
11
Bytes within the block

1-Initial State

Request Byte
Address

000	– –	–	
001	– –	–	
010	– –	–	
011	– –	–	
100	– –	–	
101	– –	–	
110	– –	–	
111	– –	–	

8 Blocks

00000	
00011	
01000	
01011	
10000	
10011	
11000	
11011	

32 Blocks

2.

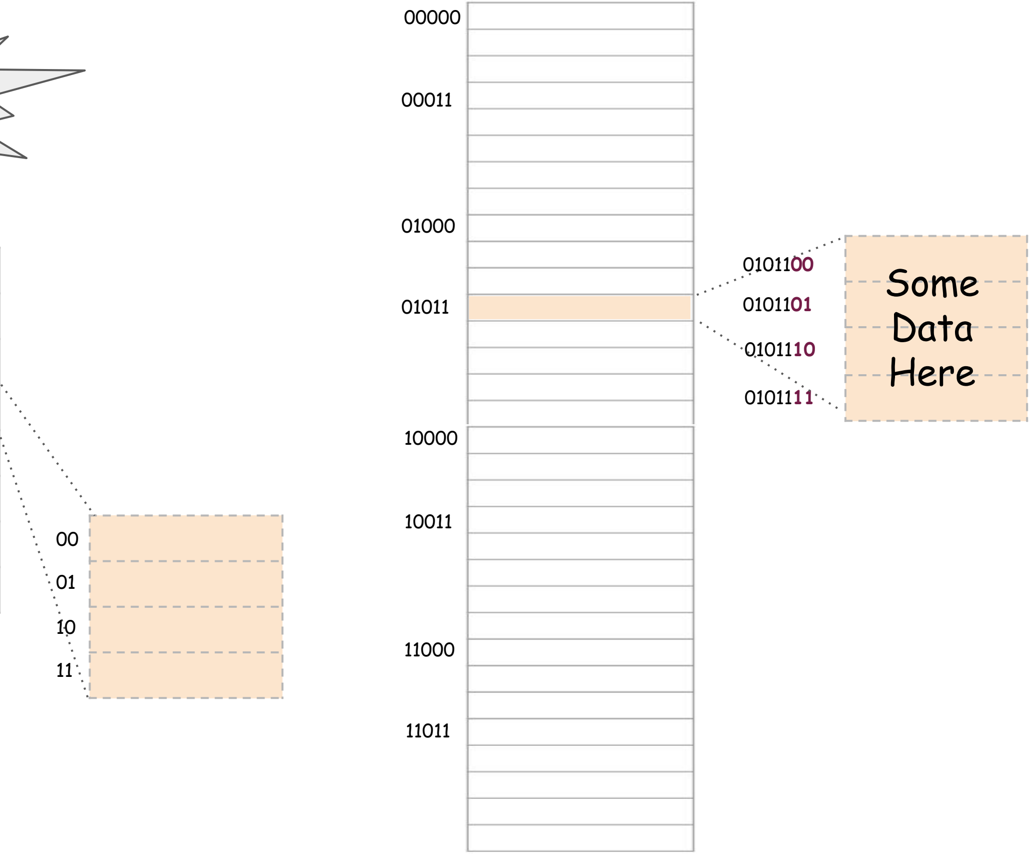
Request Byte Address

0101110

Cache Miss

000	--	0	
001	--	0	
010	--	0	
011	--	0	
100	--	0	
101	--	0	
110	--	0	
111	--	0	

8 Blocks

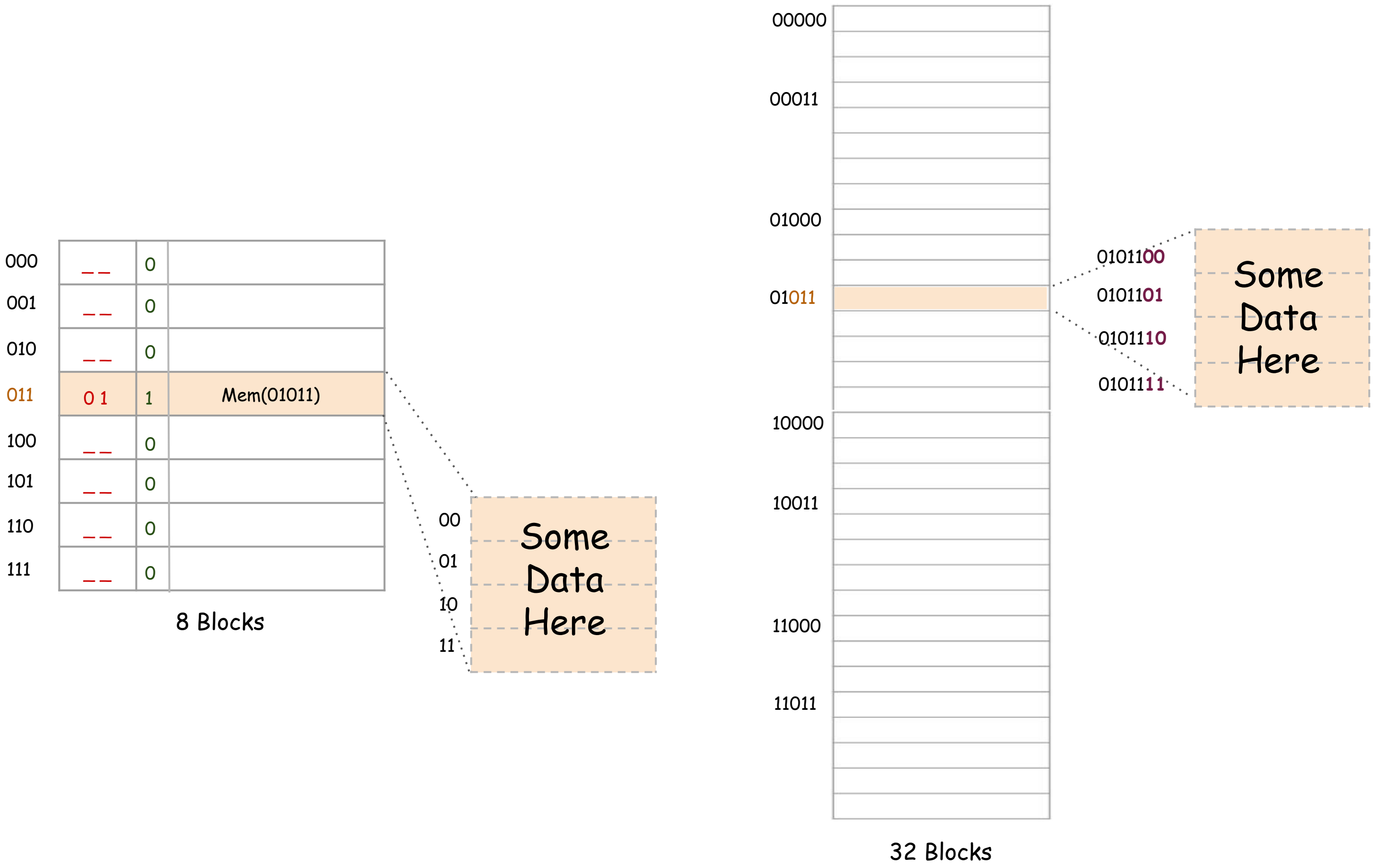


32 Blocks

3.

Request Byte Address

0101110



000	--	0	
001	--	0	
010	--	0	
011	0 1	1	Mem(01011)
100	--	0	
101	--	0	
110	--	0	
111	--	0	

4.

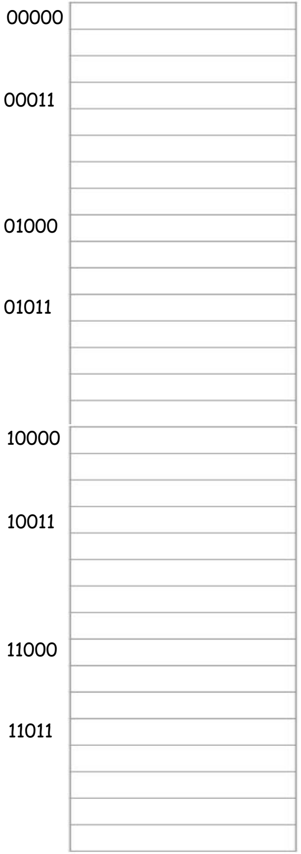
Request Byte Address

1000001

Cache Miss

000	_ _	0	
001	_ _	0	
010	_ _	0	
011	0 1	1	Mem(01011)
100	_ _	0	
101	_ _	0	
110	_ _	0	
111	_ _	0	

8 Blocks



32 Blocks

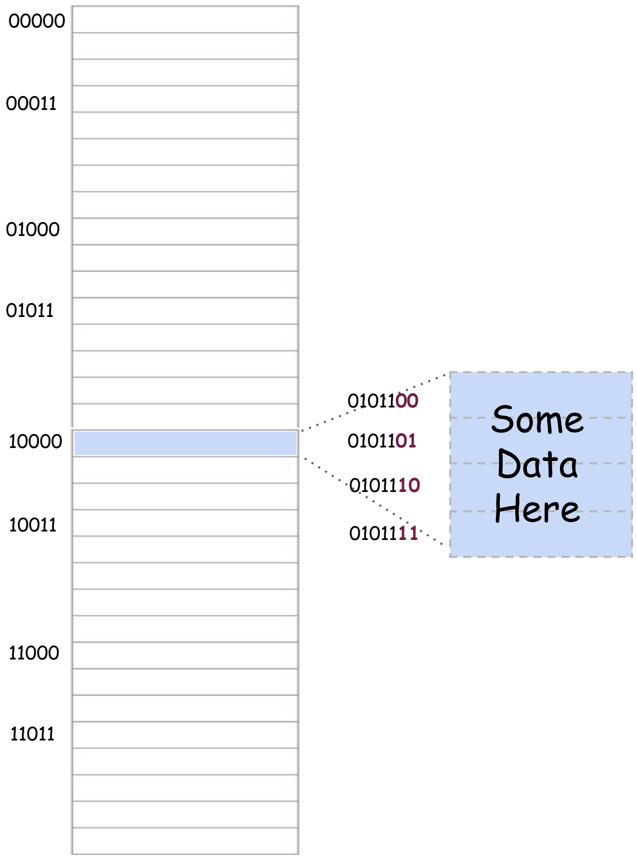
5.

Request Byte Address

1000001

000	1 0	1	Mem(10000)
001	_ _	0	
010	_ _	0	
011	0 1	1	Mem(01011)
100	_ _	0	
101	_ _	0	
110	_ _	0	
111	_ _	0	

8 Blocks



32 Blocks

6.

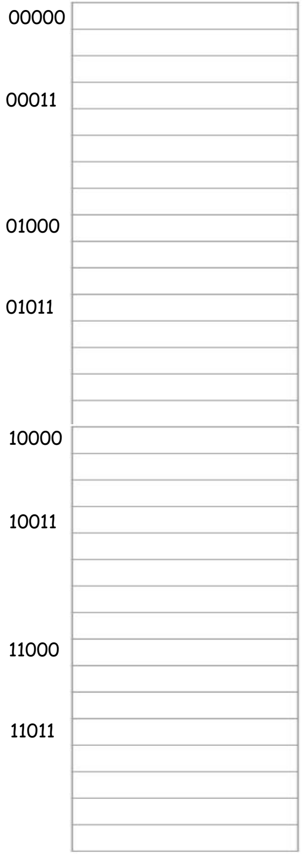
Request Byte Address

0101100

Cache Hit 🙂

000	1 0	1	Mem(10000)
001	– –	0	
010	– –	0	
011	0 1	1	Mem(01011)
100	– –	0	
101	– –	0	
110	– –	0	
111	– –	0	

8 Blocks



32 Blocks

7.

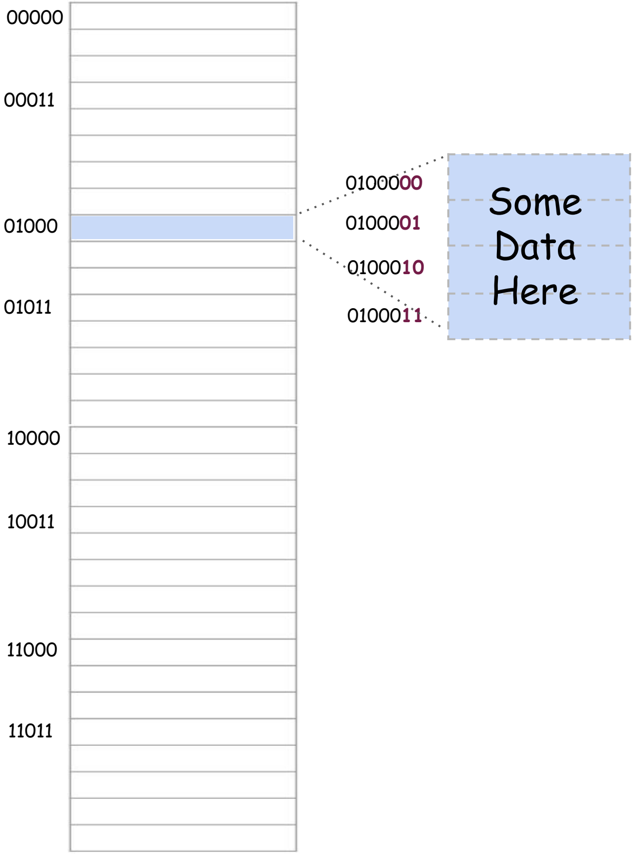
Request Byte Address

0100011

Cache Miss

000	1 0	1	Mem(10000)
001	_ _	0	
010	_ _	0	
011	0 1	1	Mem(01011)
100	_ _	0	
101	_ _	0	
110	_ _	0	
111	_ _	0	

8 Blocks

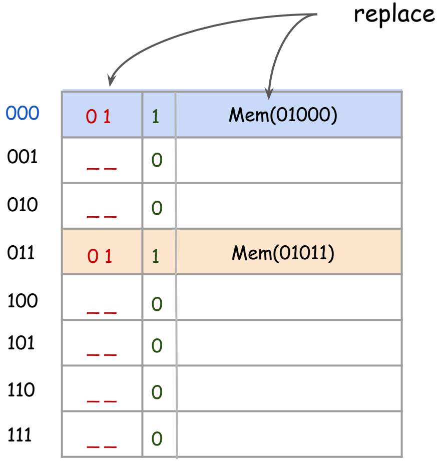


32 Blocks

7.

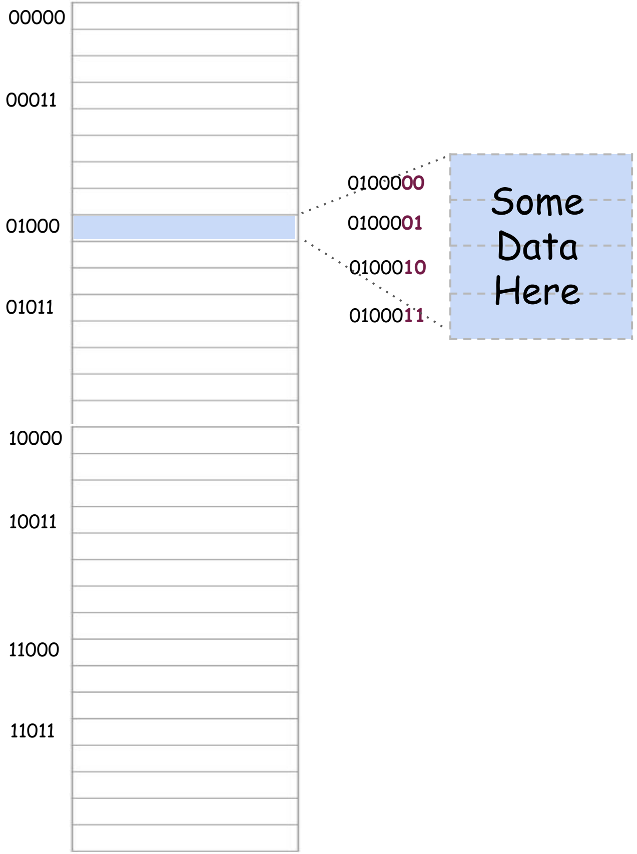
Request Byte Address

0100011



000	0 1	1	Mem(01000)
001	_ _	0	
010	_ _	0	
011	0 1	1	Mem(01011)
100	_ _	0	
101	_ _	0	
110	_ _	0	
111	_ _	0	

8 Blocks



32 Blocks

Cache Performance

MissRate = #CacheMisses / #CacheAccesses

Average Memory Access Time

$$\boxed{\text{AMAT} = \text{HitTime} + \text{MissRate} \times \text{MissPenalty}}$$

Average Memory Access Time

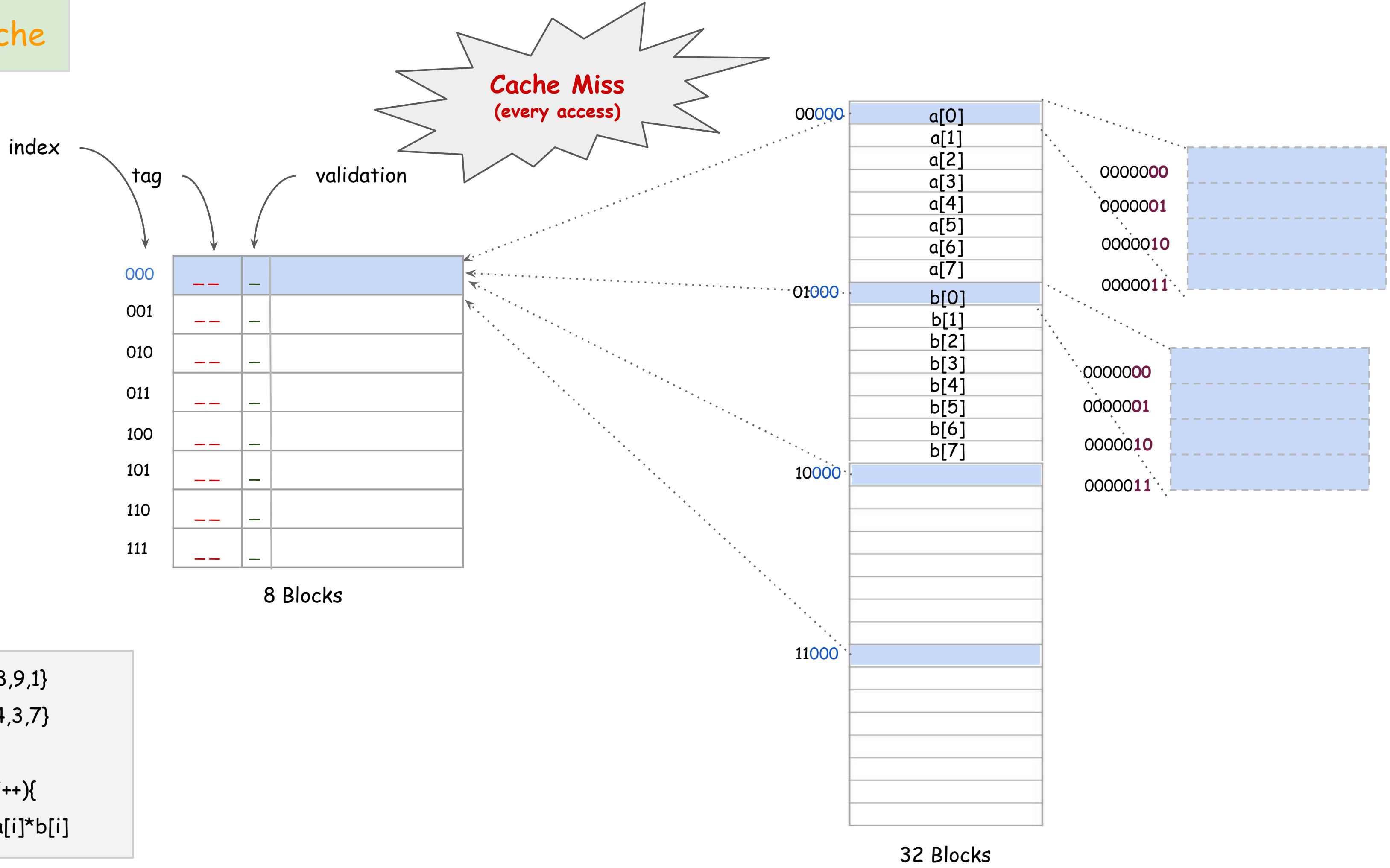
AMAT = HitTime + MissRate x MissPenalty

MissRate = #CacheMisses / #CacheAccesses

Cache Miss Categories (3Cs)

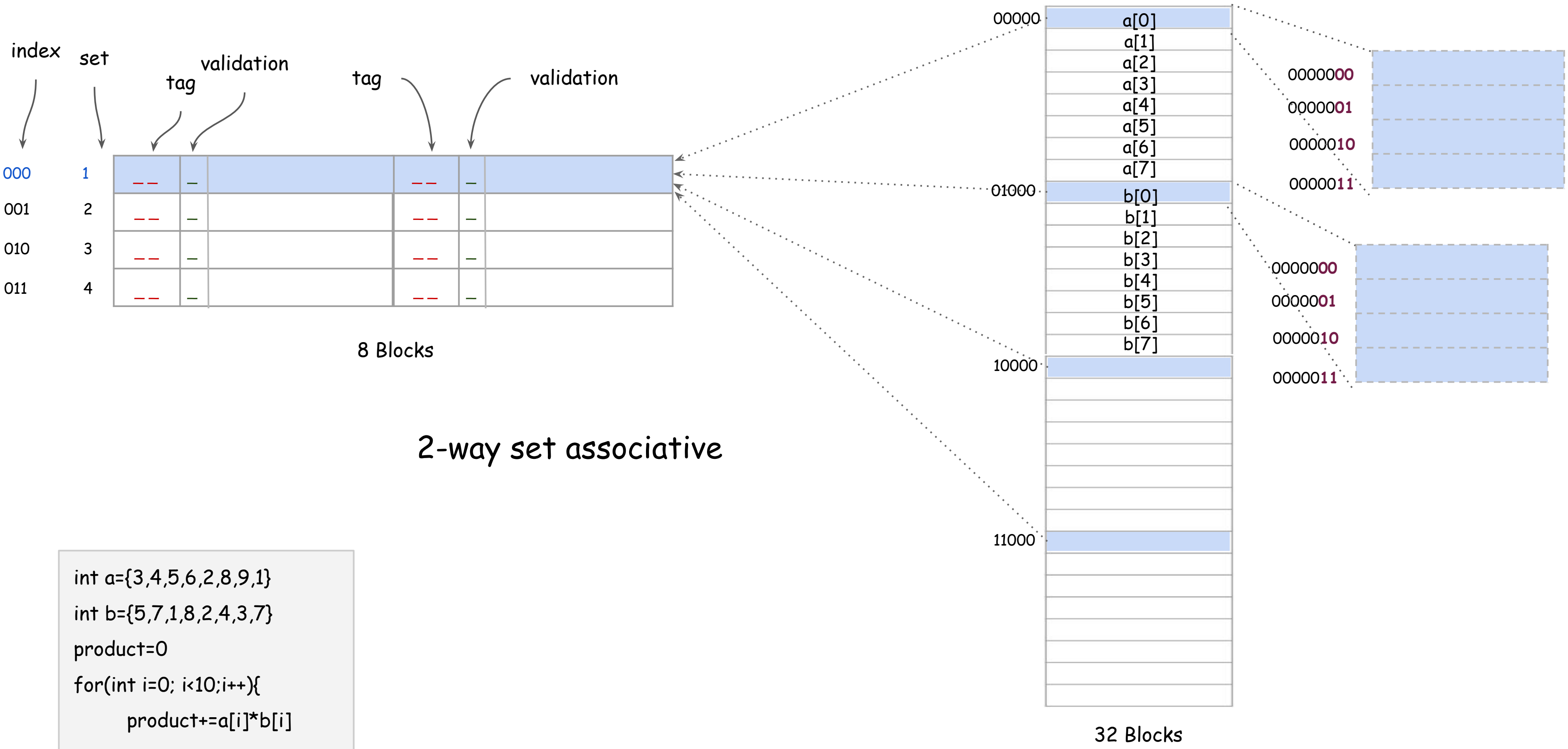
- Compulsory (First Access is always a miss)
- Capacity (program working set is larger than cache capacity)
- Conflict (several blocks are mapped to same block frame)

Associative Cache



```
int a={3,4,5,6,2,8,9,1}
int b={5,7,1,8,2,4,3,7}
product=0
for(int i=0; i<10;i++){
        product+=a[i]*b[i]
```

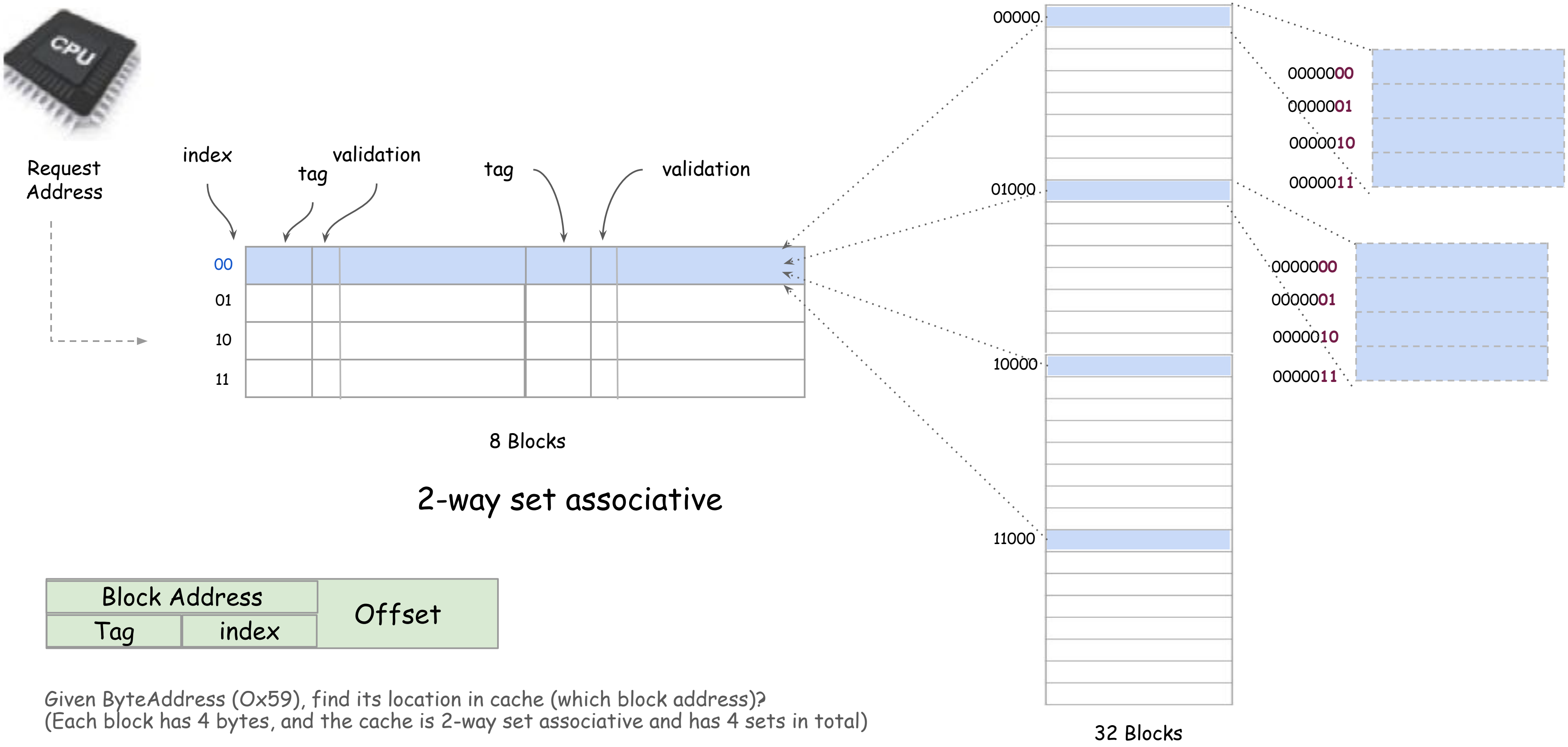
Associative Cache



2-way set associative

```
int a={3,4,5,6,2,8,9,1}
int b={5,7,1,8,2,4,3,7}
product=0
for(int i=0; i<10;i++){
        product+=a[i]*b[i]
```

Associative Cache



2-way set associative

Block Address		Offset
Tag	index	

Given ByteAddress (0x59), find its location in cache (which block address)?
(Each block has 4 bytes, and the cache is 2-way set associative and has 4 sets in total)

Associative Cache

Exercise:

Consider a 64 Blocks 2-way associative cache and a block size of 16 bytes. To what Block number does byte address $(0x4B0)_{16}$=$(0000010010110000)_2$ map to (assuming 16 bits address)?

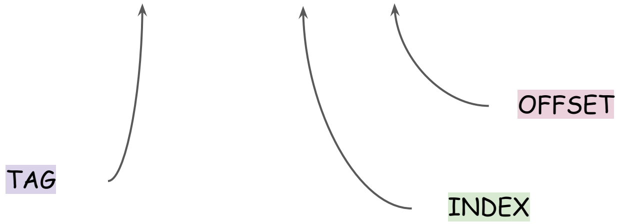
Exercise Solution:

Consider a 64 Blocks 2-way associative cache and a block size of 16 bytes. To what Block number does byte address $(0x4B0)_{16}$=$(0000010010110000)_2$ map (assuming 16 bits address)?

CacheSize is 64 blocks (2-ways) ⇒ so the is INDEX 5 bits which comes from $64/2 = 32 = 2^5$

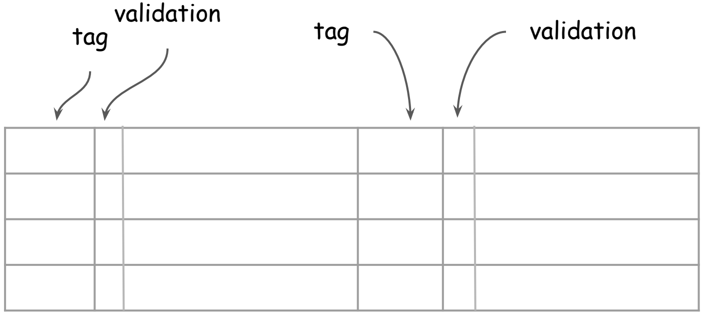
BlockSize is 16 bytes = 2^4 ⇒ so the OFFSET is 4 bits

Address is $(0x4B0)_{16}$ =$(0000010010110000)_2$



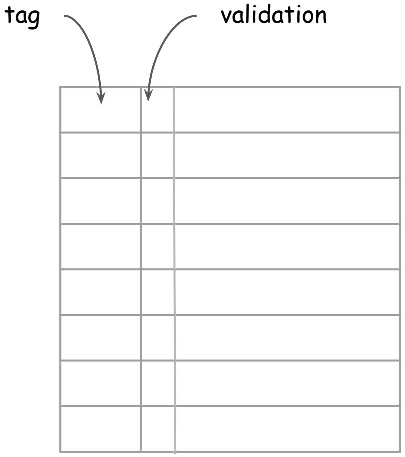
Associative Cache

2-ways set associative



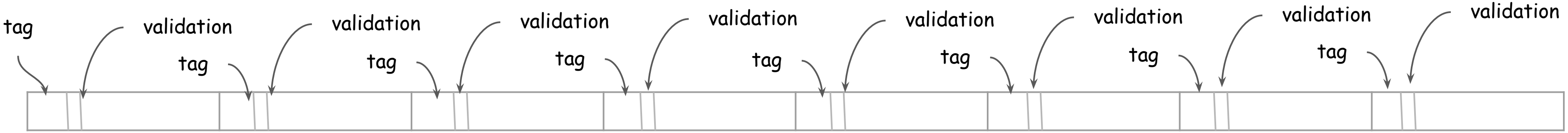
#Sets ≠ #Blocks

1-way set associative
(direct mapping)



#Sets = #Blocks

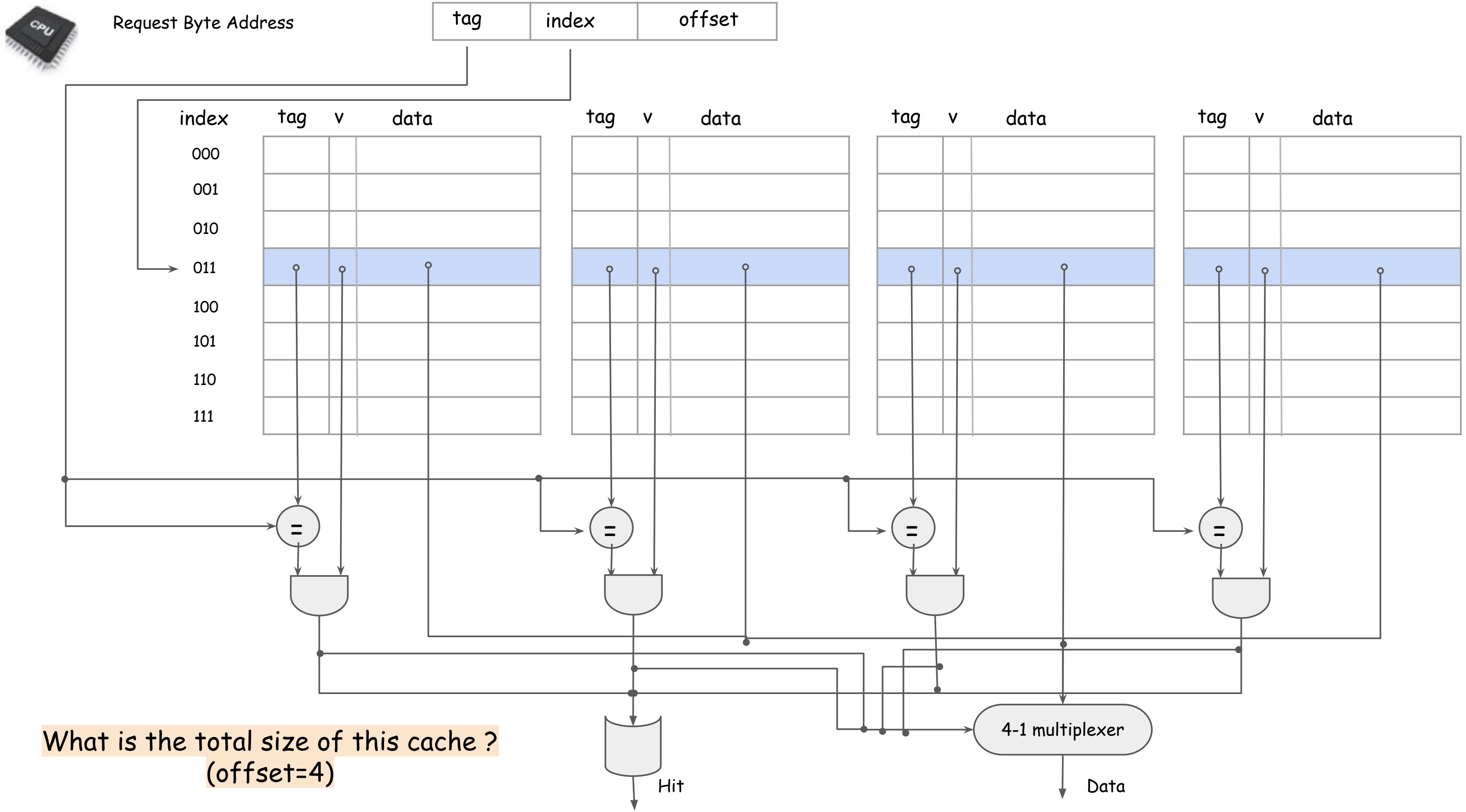
8-ways set associative
(fully associative)



#Sets = 1
(cacheIndex=0)

CacheSize = #Sets x Ways x #BlockSize

Associative Cache



What is the total size of this cache ? (offset=4)

Cache Performance

MissRate = #CacheMisses / #CacheAccesses

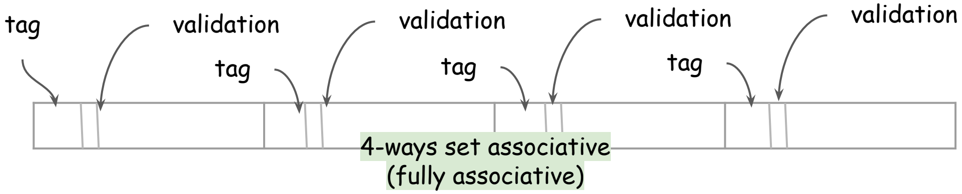
Average Memory
Access Time

$$\text{AMAT} = \text{HitTime} + \text{MissRate} \times \text{MissPenalty}$$

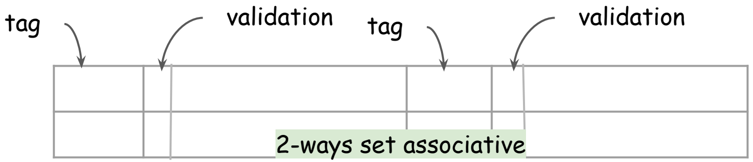
Associative Cache

Which cache organization reduces misses?

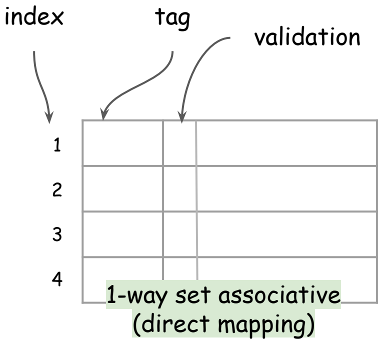
What type of misses can this organization reduce (compulsory, capacity, conflict) ?



4-ways set associative
(fully associative)



2-ways set associative



1-way set associative
(direct mapping)