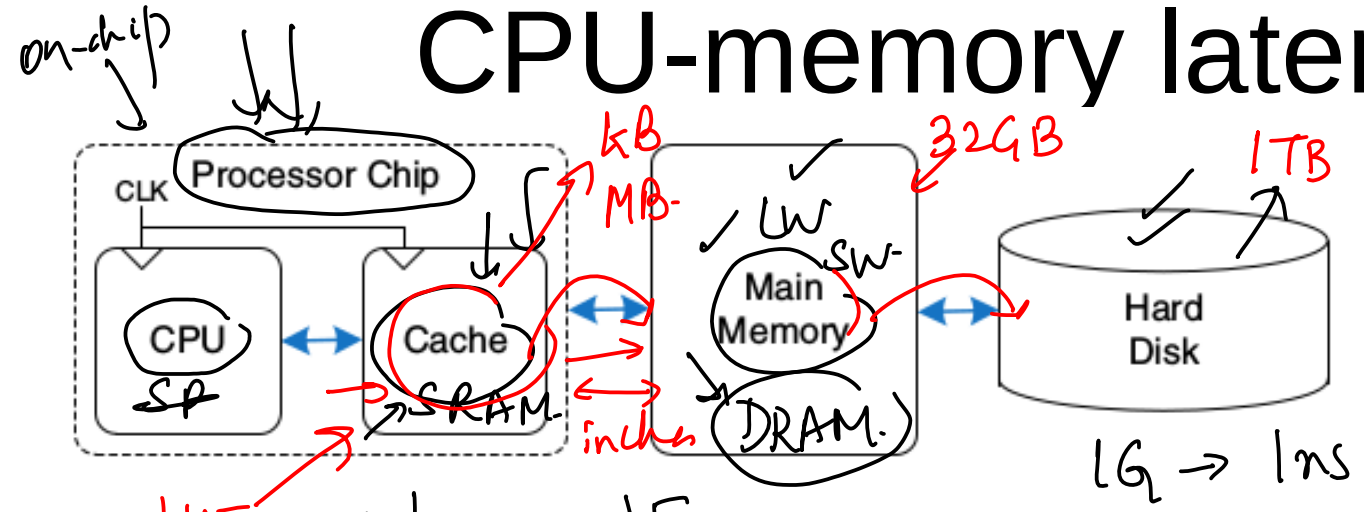
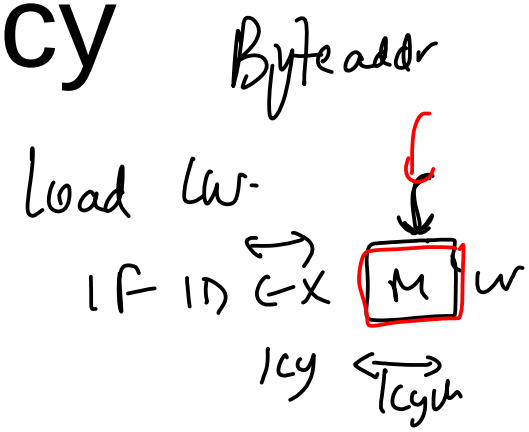
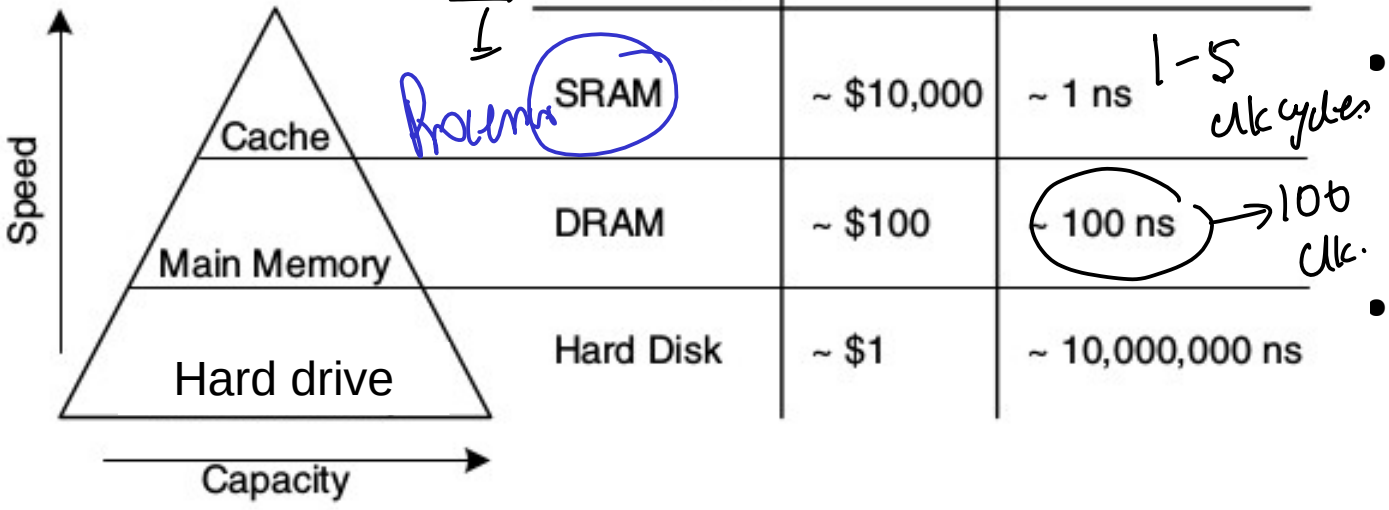
Introduction to Caches

CPU-memory latency





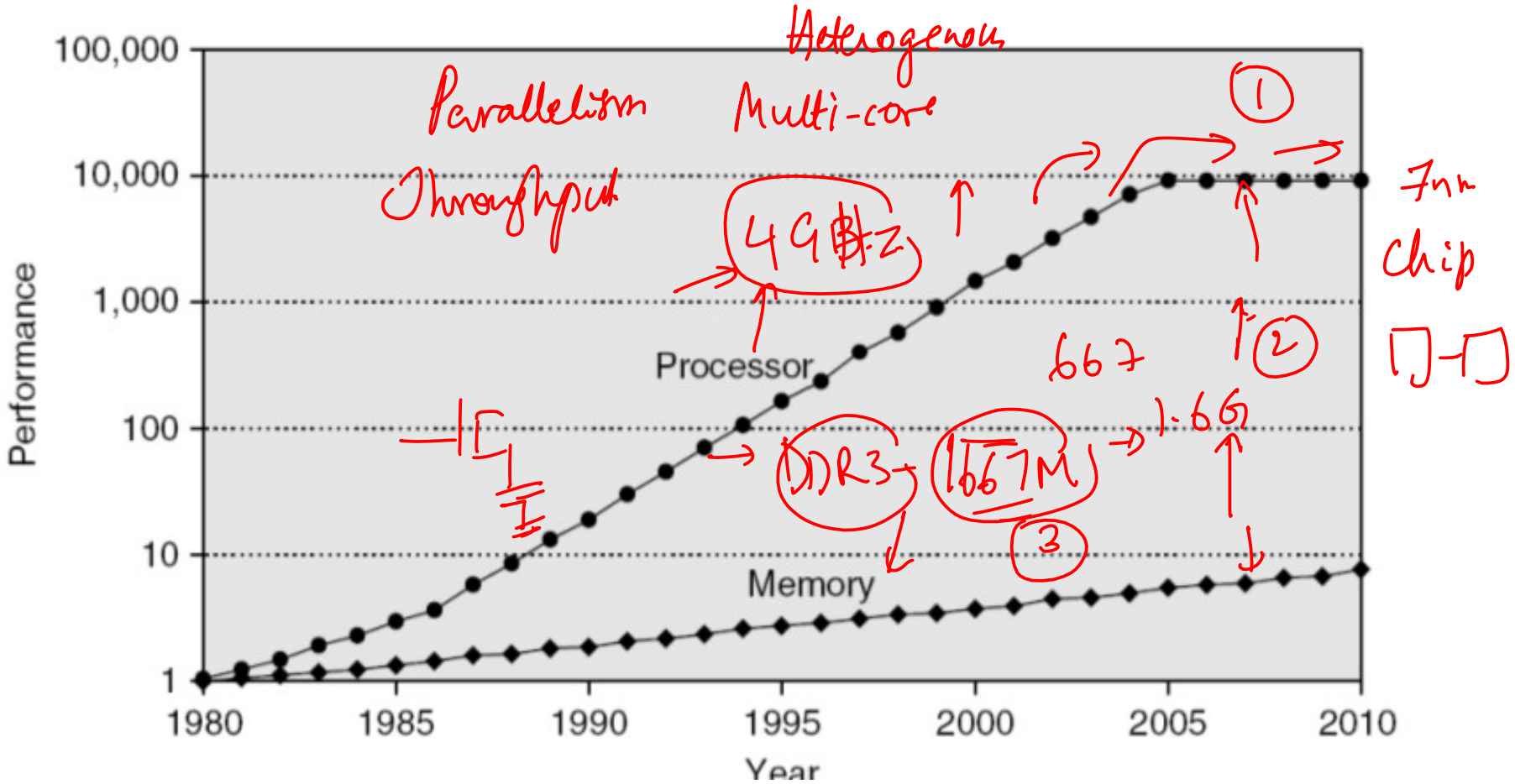


Technology	Cost/GB	Access Time
SRAM	~ $10,000	~ 1 ns
DRAM	~ $100	~ 100 ns
Hard Disk	~ $1	~ 10,000,000 ns

- Latency- time to access memory
- Bandwidth - number of accesses per unit time
- Capacity – memory size

https://techdifferences.com/difference-between-sram-and-dram.htm

Processor-memory performance gap – extremely slow access time



Four-issue 3GHz superscalar accessing 100ns DRAM could execute 1,200 instructions during the time for one memory access!

Hennessy and Patterson Edition 5

Need for a better memory

- Need large, fast, and less expensive memory
  - Large memories are slow
  - Fast memories are expensive
- Can we use a hierarchal system of memories?
- Most often-used data in a small, fast SRAM (on the CPU die)
- Go to Main Memory rarely

Locality – property of a program

Temporal Locality: If a location is referenced it is likely to be referenced again in the near future.

Eg- Loops : 3, 209, 5, 3, 12, 24, 3

Exploit temporal locality by remembering the contents of recently accessed addresses.

Spatial Locality: If a location is referenced it is likely that locations/addresses near it will be referenced in the near future.

Eg- Arrays. 4, 78, 34, 35, 36, 109

Exploit spatial locality by fetching data around recently accessed addresses.

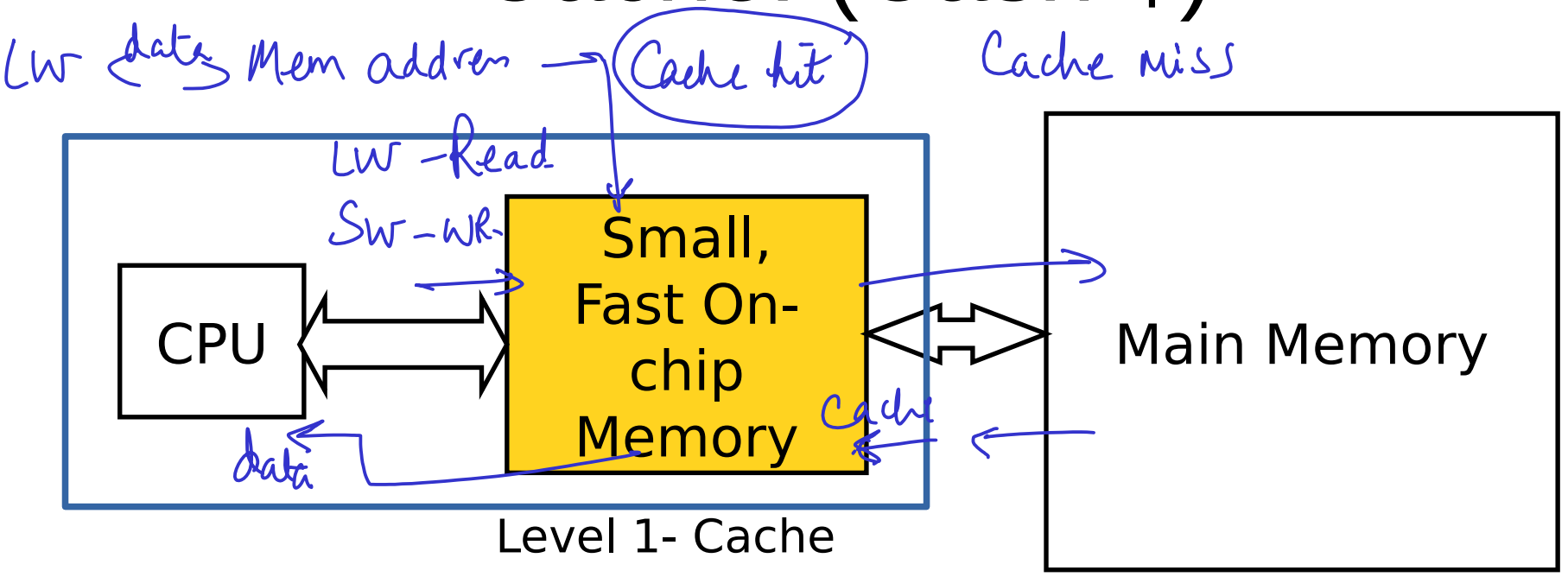
Example

- For (i=0;i<100;i++)

  A[i] = B[i] + 100

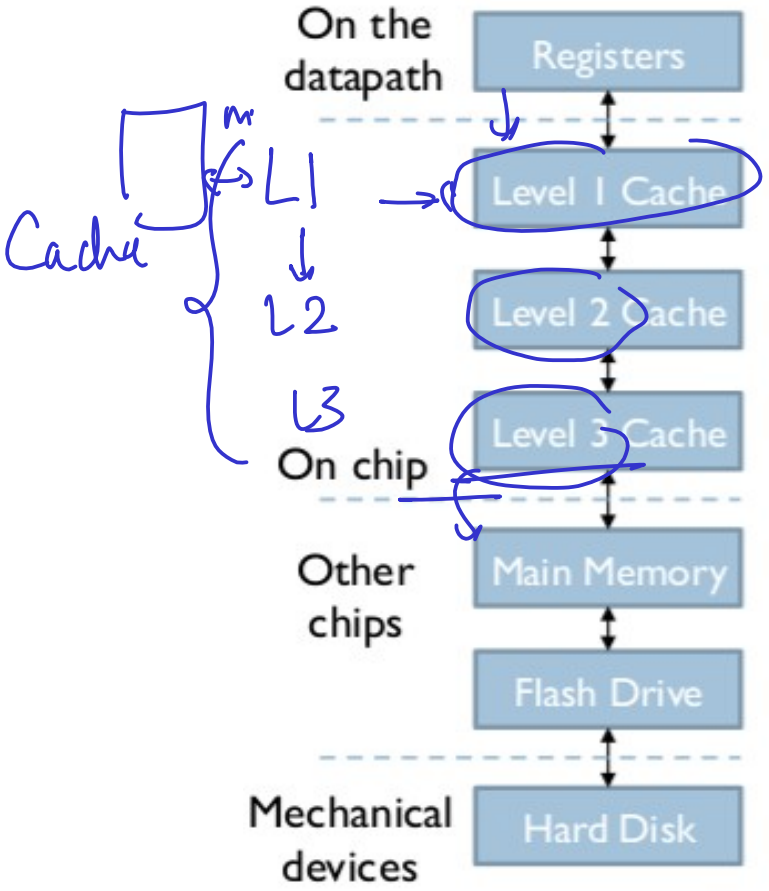
- B[i] – array – repeated reference to nearby address locations

Cache! (Cash $)

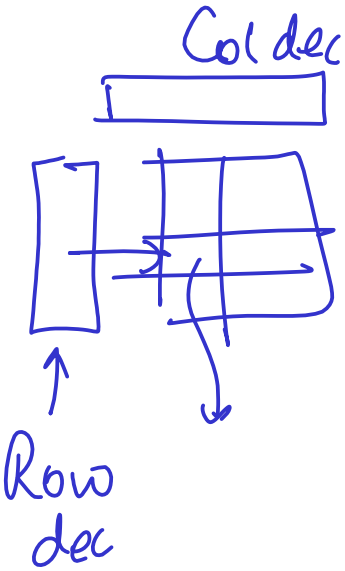


- Library analogy – frequently accessed books are on the desk
- Cache- retains data from recently accessed locations. On chip – faster than off-chip.
- Block (or line) - The minimum unit of information that can be either present/transferred in a cache

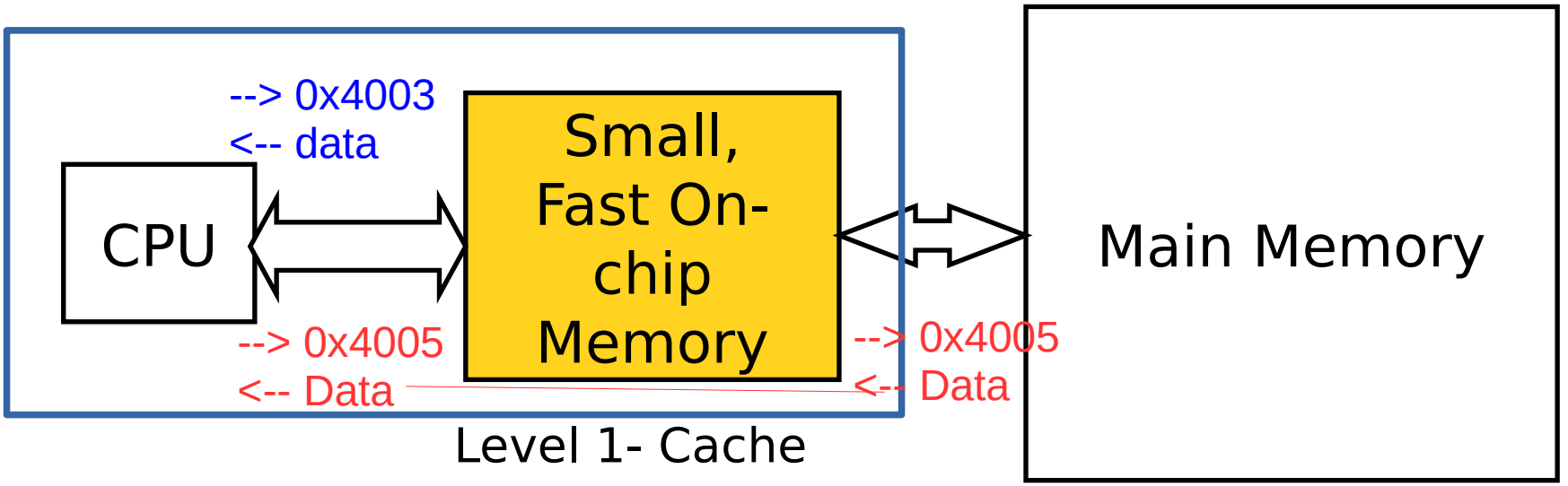
Hierarchies



Access time	Capacity
1 cycle	1 KB
2-4 cycles	32 KB
10 cycles	256 KB
40 cycles	10 MB
200 cycles	10 GB
10-100us	100 GB
10ms	1 TB



Accessing data



- Processor sends address to the cache
  - Cache hit: Data present, return it
  - Cache miss: Data not in cache
    - Fetch data from memory, send to processor
    - Retain this data in cache (replace some data)

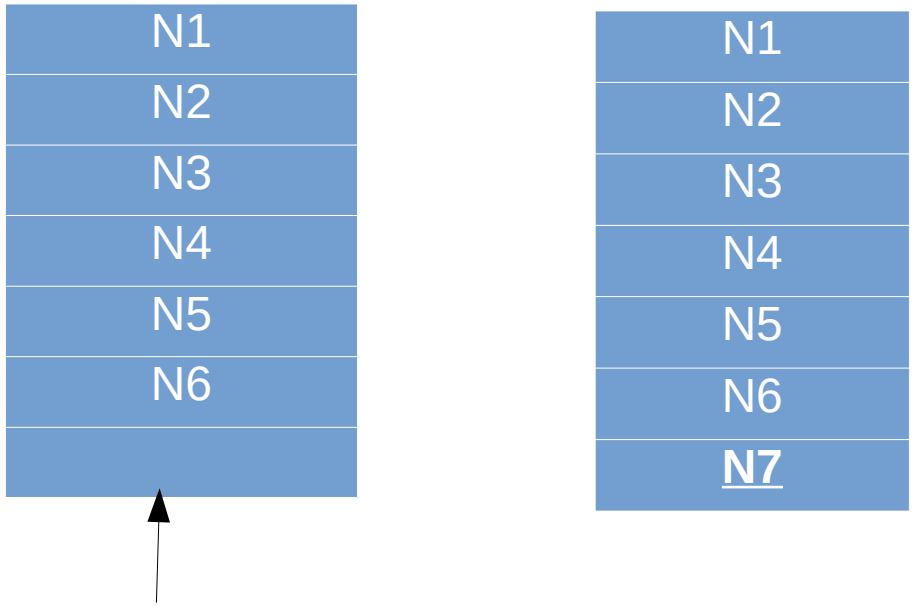
Performance

- To improve performance:
  - Reduce the hit time
  - Reduce the miss rate
  - Reduce the miss penalty

Types of caches

- Direct mapped caches
- Associative caches
  - Fully associative
  - N-way set associative

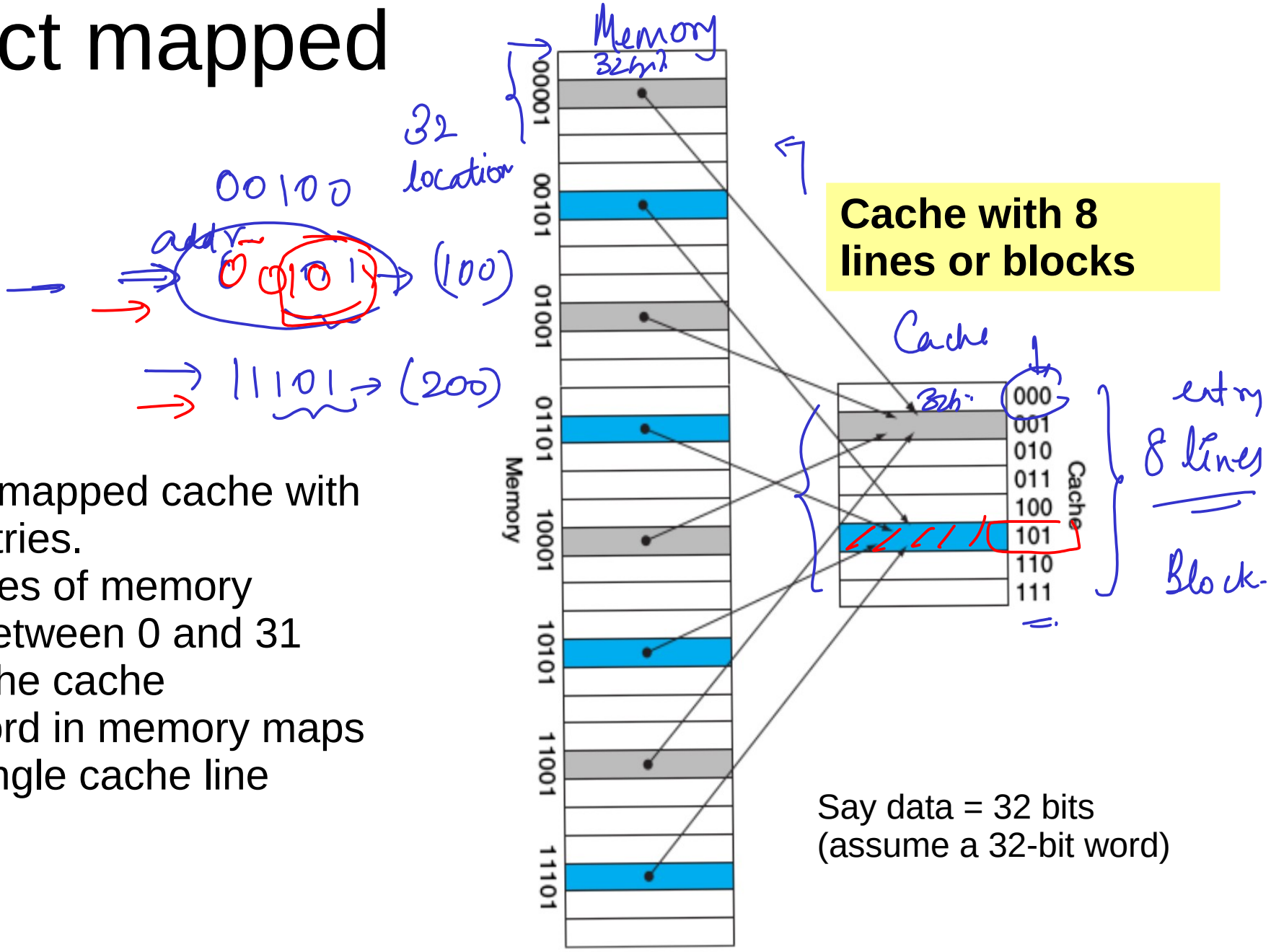
Basics of caches



How do we know if a data is in the cache?

How do we find it?

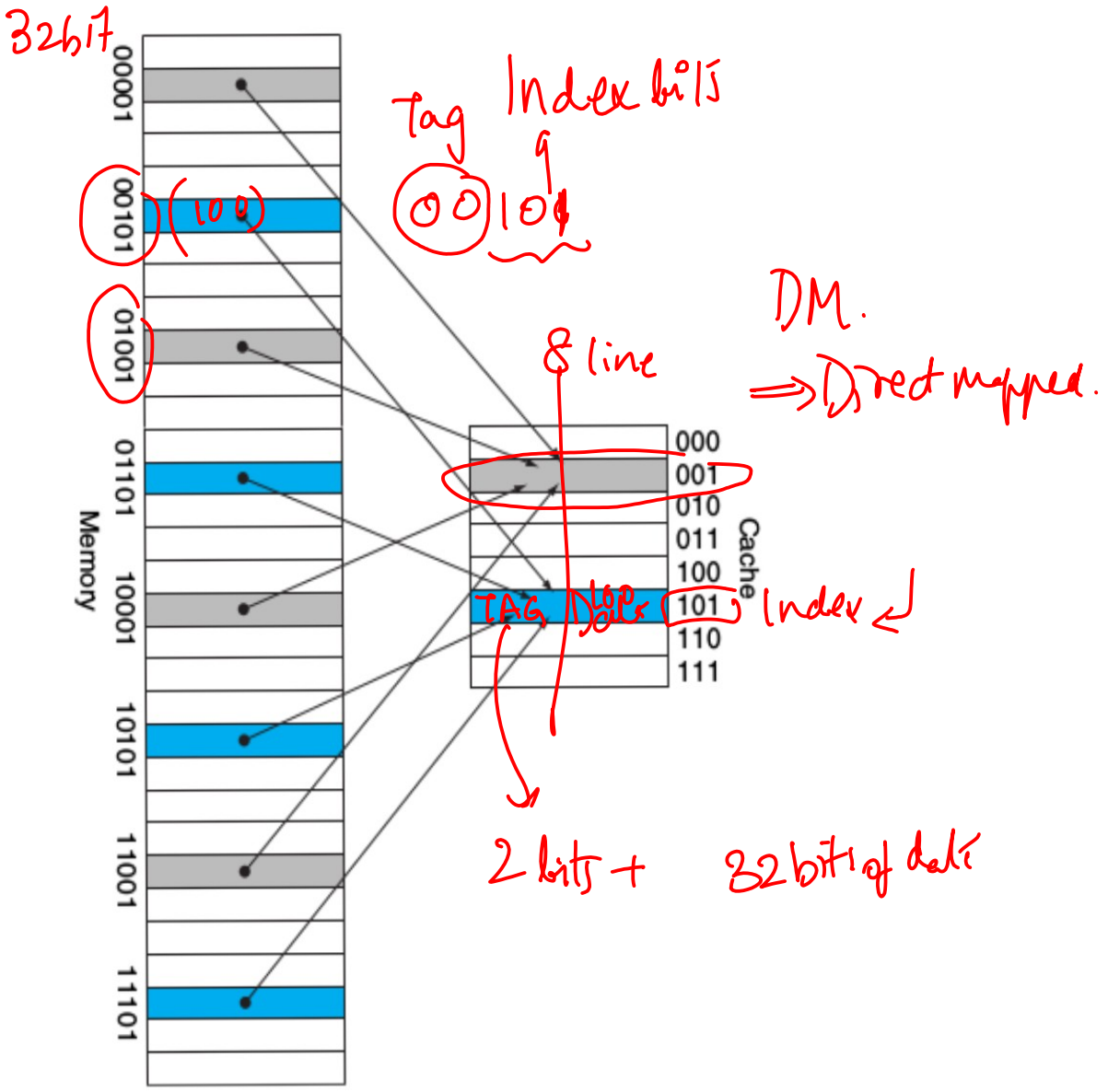
Direct mapped



- A direct-mapped cache with eight entries.
- Addresses of memory words between 0 and 31 map to the cache
- Each word in memory maps into a single cache line

Say data = 32 bits
(assume a 32-bit word)

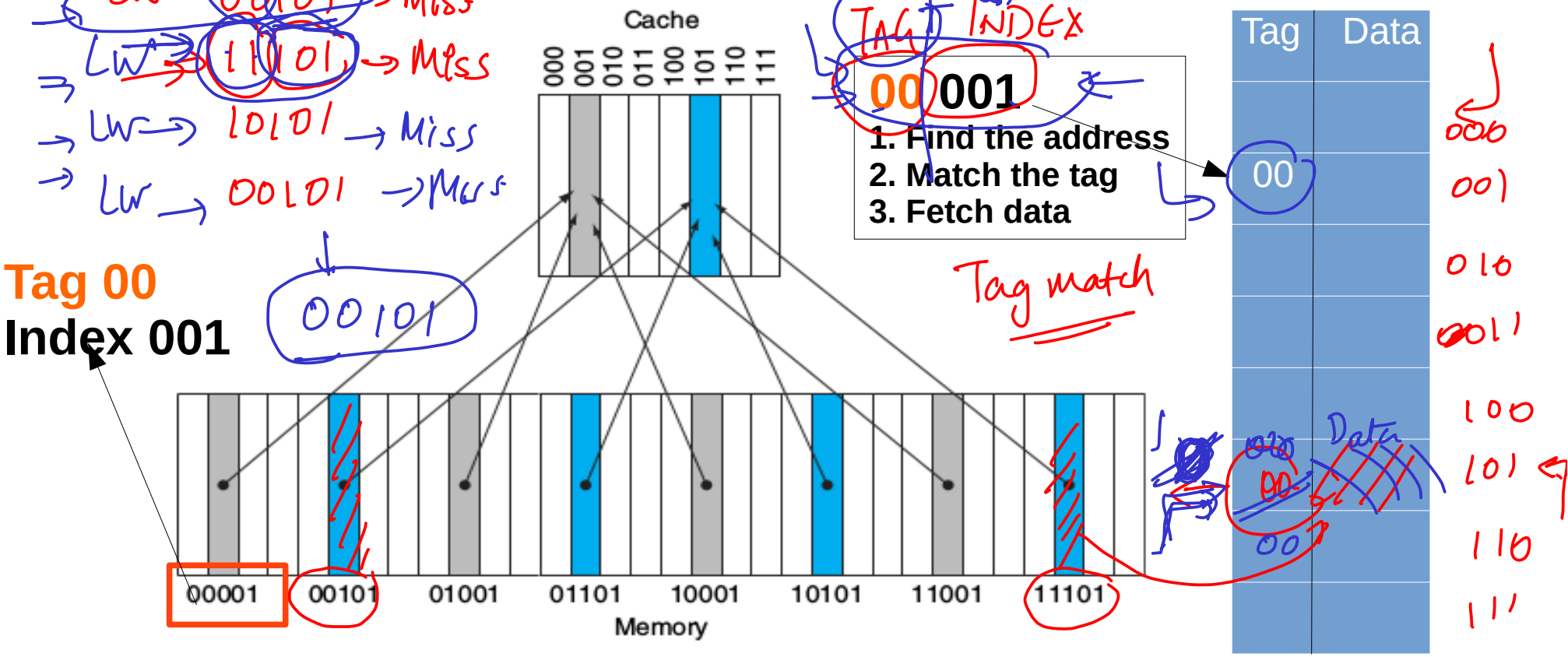
Mapping



- An address X maps to the direct-mapped cache word = X modulo 8
- **(Block address) modulo (Number of blocks in the cache)**
- $\log_2(8) = 3$ bits are used as the **cache index**
- Addresses 00001 , 01001 , 10001 and 11001 map to entry 001 of the cache

DM cache

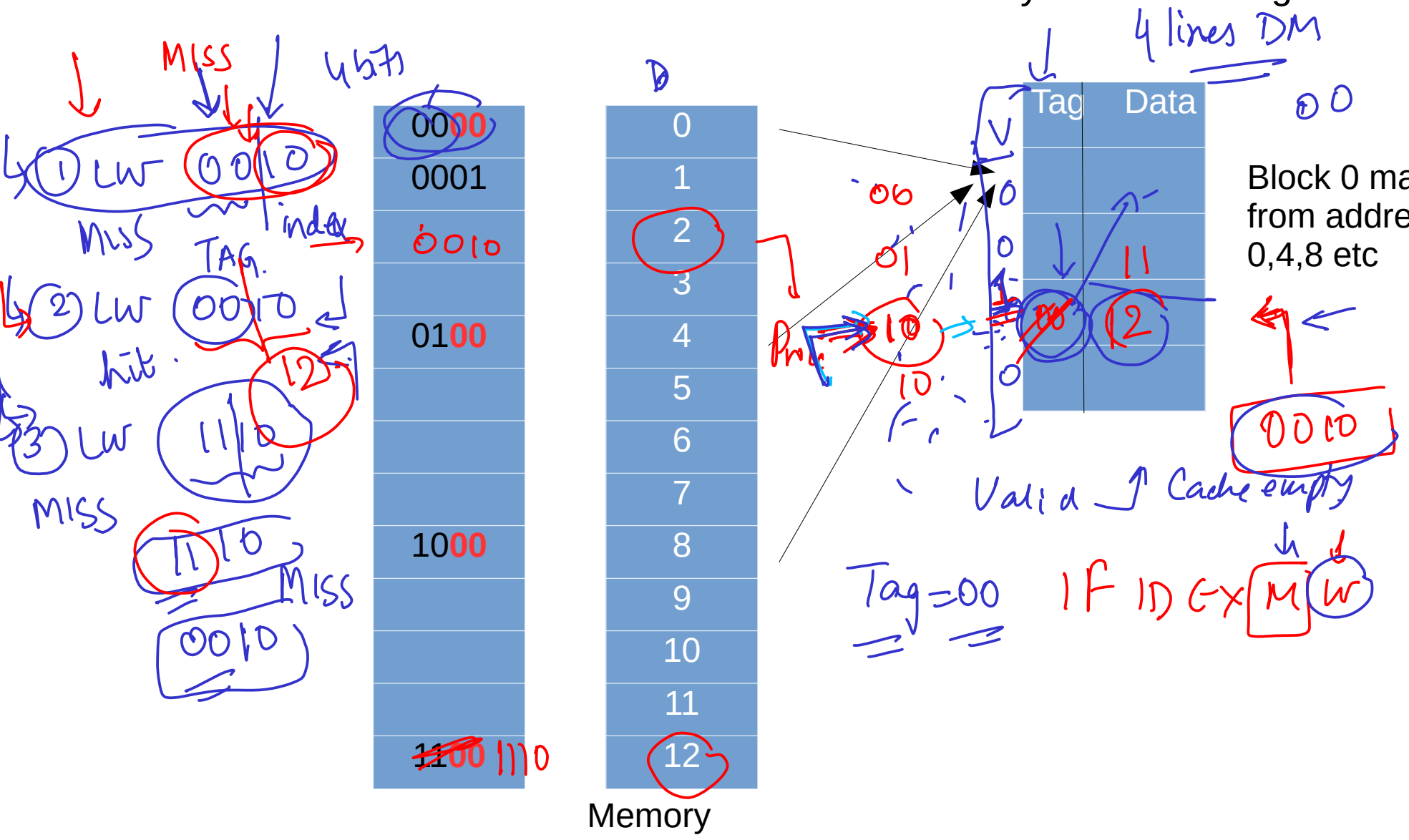
Cache with 8 lines or blocks



Each cache location can contain the contents of several memory locations.
How do we map the correct data?
Tag --> Higher order two bits of the memory
Problems: Collisions

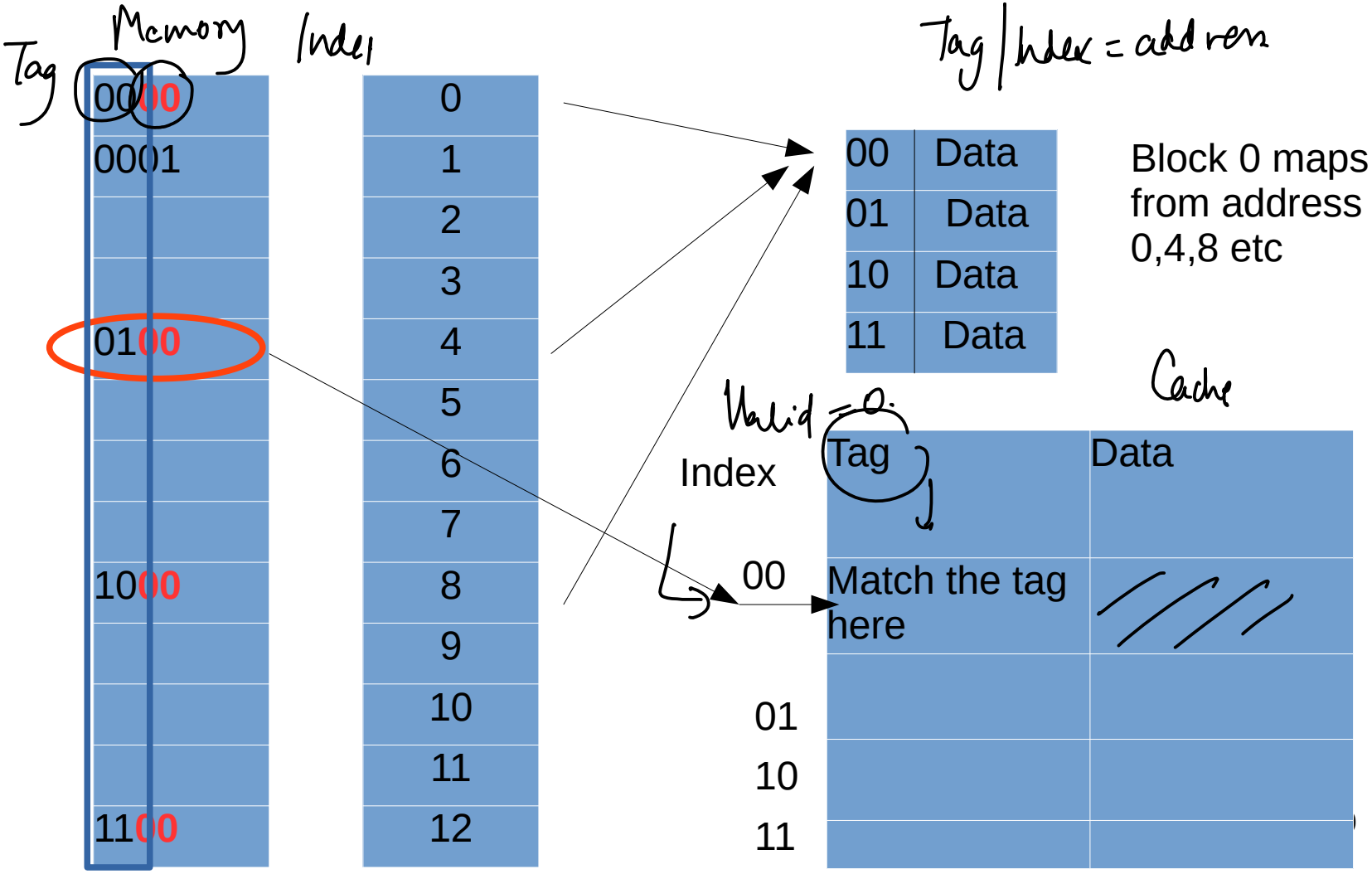
Direct mapped

Cache with 4 blocks uses 2 lower bits from main memory for addressing



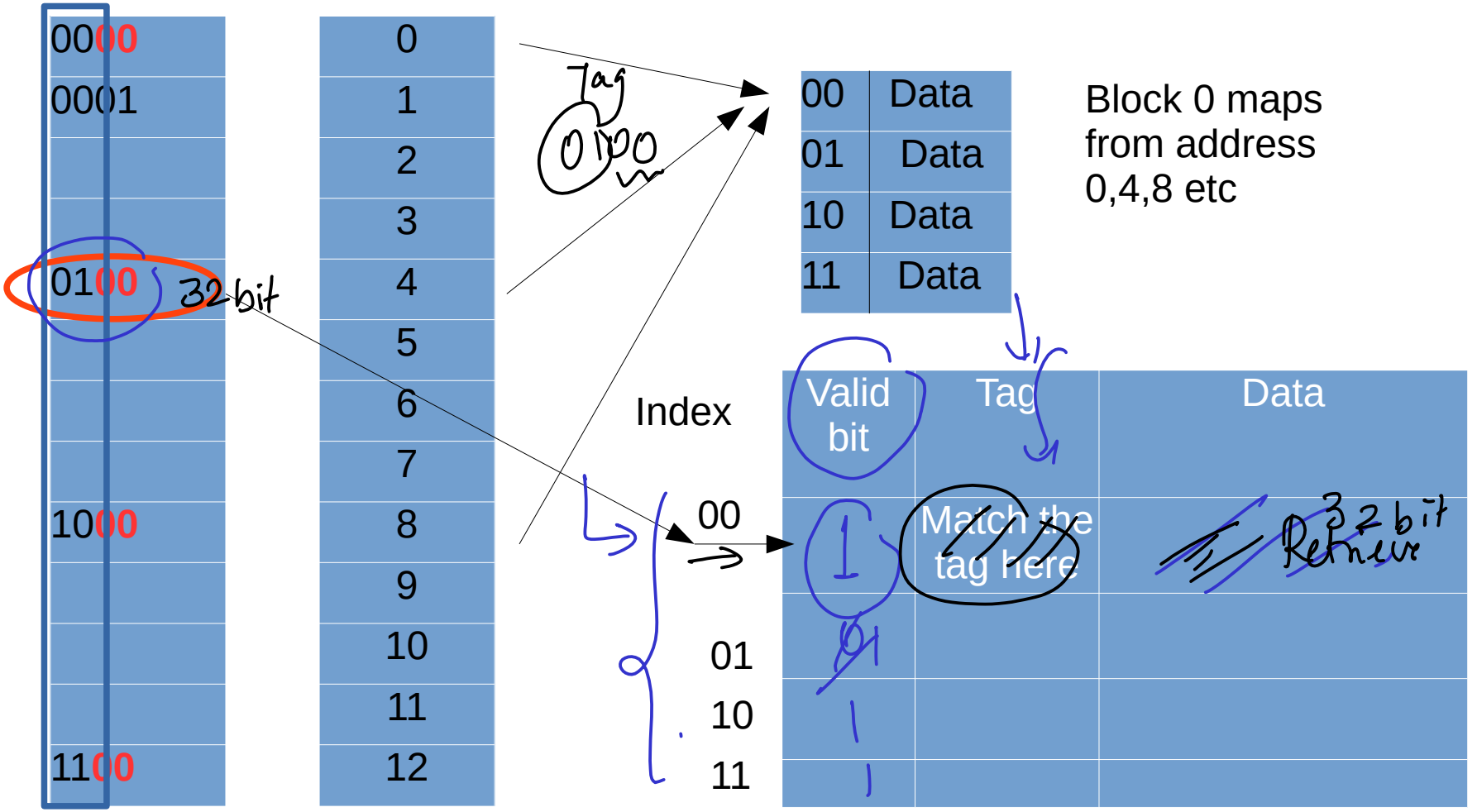
Direct mapped (DM)

- Concept of a **tag** and an index
- Usually a **valid bit** is added to check if the data is valid
  - For eg- on startup – cache entries are invalid

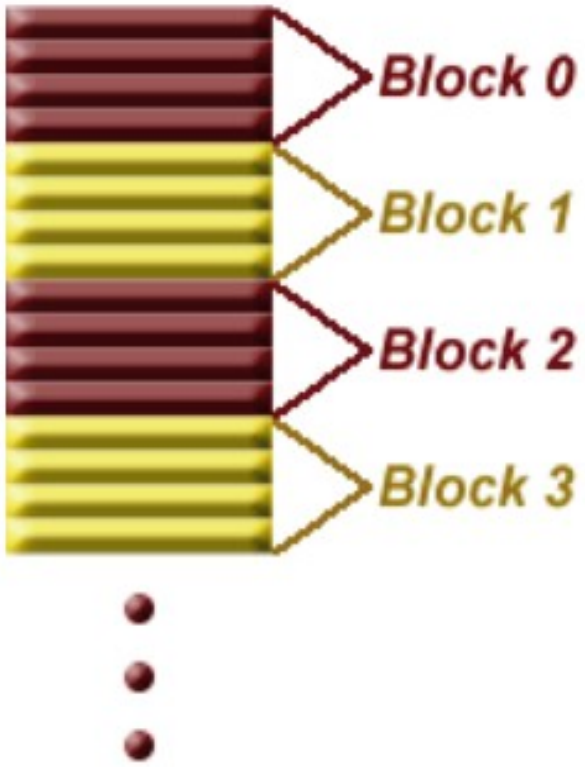


Direct mapped (DM)

- A **valid bit** is added to check if the data is valid



Concept of a block



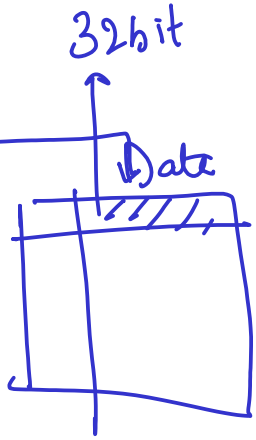
Main memory is divided into blocks of memory locations

Main memory

Block	Address
Block 0 (4 bytes)	0x00000
	0x00001
	0x00002
	0x00003
Block 1	0x00004
	0x00005
	0x00006
	0x00007
Block 2	0x00008
	0x00009
	0x0000A
	0x0000B

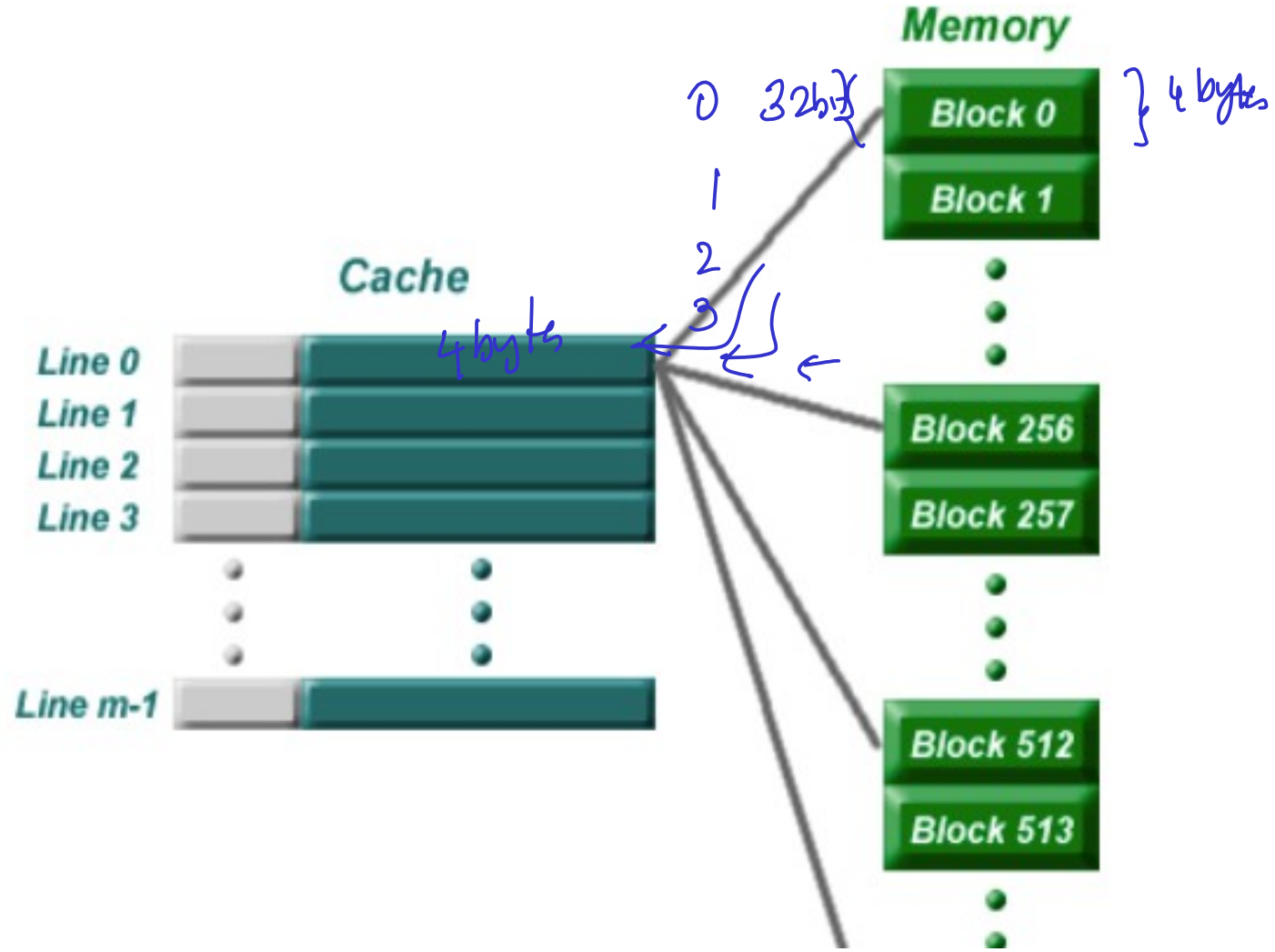
1 byte

Each block = 4 bytes of data.
Each address has one byte of data



Cache is organized into lines, each of which contains enough space to store **exactly one block** of data and a tag uniquely identifying where that block came from in memory

Concept of a block

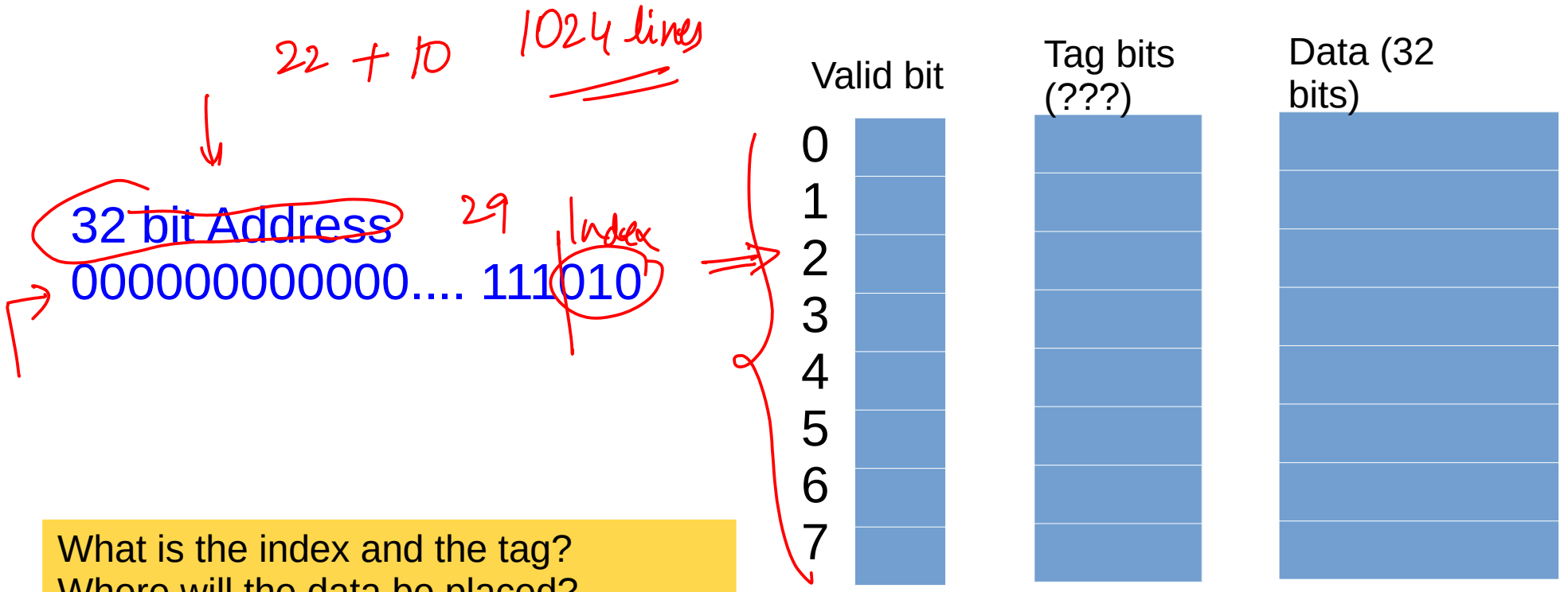


Each memory block has 4 bytes of data

Each block maps to a specific line in the cache

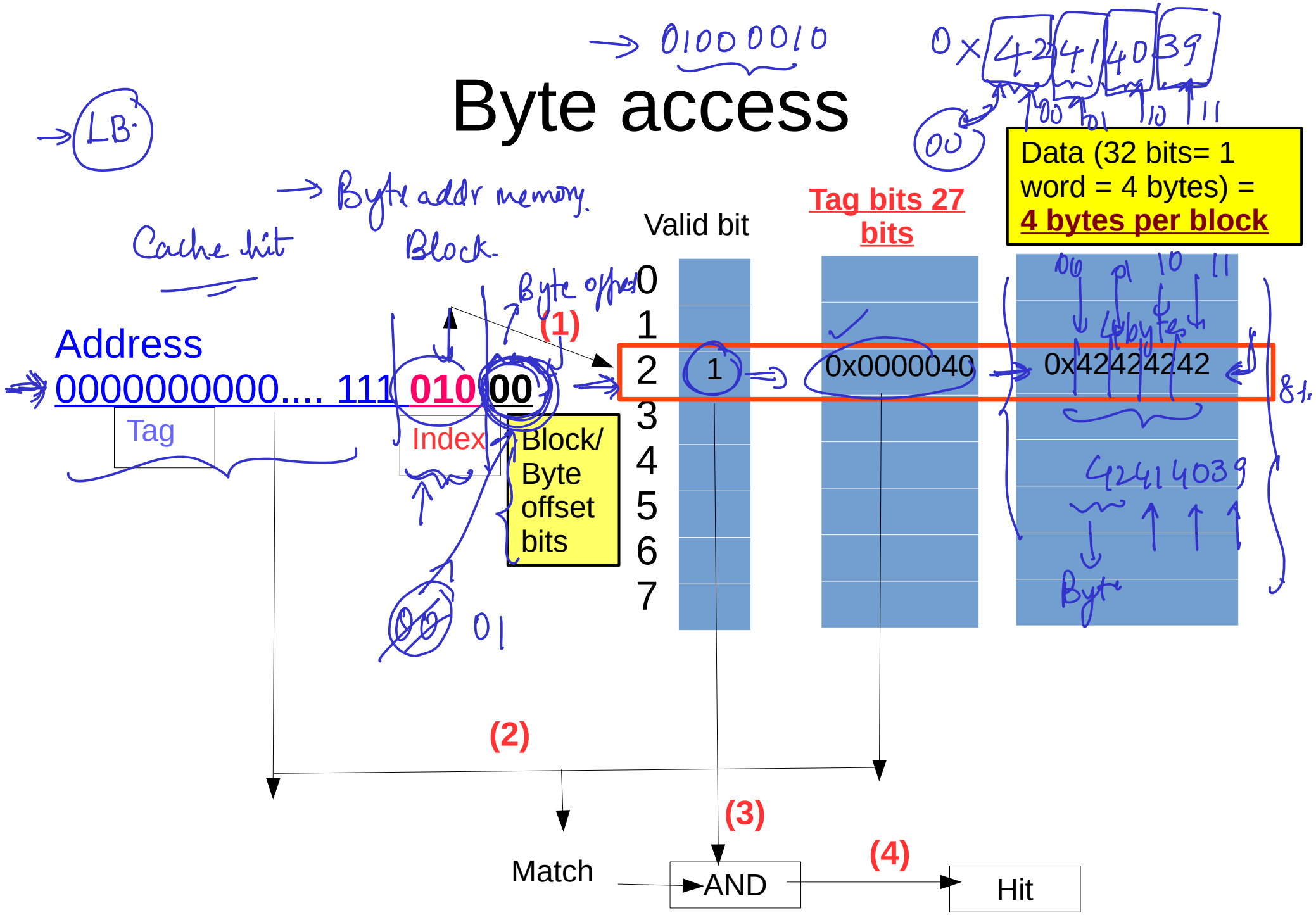
Thus a cache line contains a single word or 4 bytes of data

Example – 8 line cache



What is the index and the tag?
Where will the data be placed?

Byte access


Data (32 bits= 1 word = 4 bytes) = 4 bytes per block
Tag bits 27 bits
Valid bit
Address
0000000000.... 111 010 00
Tag
Index
Block/ Byte offset bits
(1)
(2)
(3)
(4)
0
1
2
3
4
5
6
7
1
0x0000040
0x42424242
Match
AND
Hit

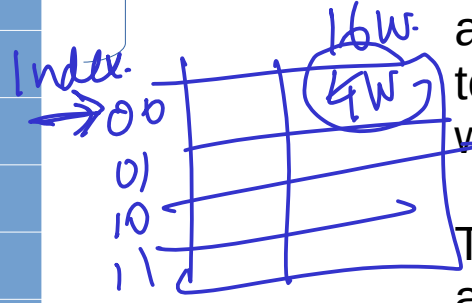
Concept of a block

16 bytes of data in 16 locations

Byte address	Access individual bytes inside the block
0000	Offset 00
0001	Offset 01
0010	Offset 10
0011	Offset 11
0100	
0101	
0110	
0111	

Block 0

Block 1



There are 4 blocks
The blocks can be addressed using the block address 0, 1, 2, 3--> which are the 2 MSB bits of the address.

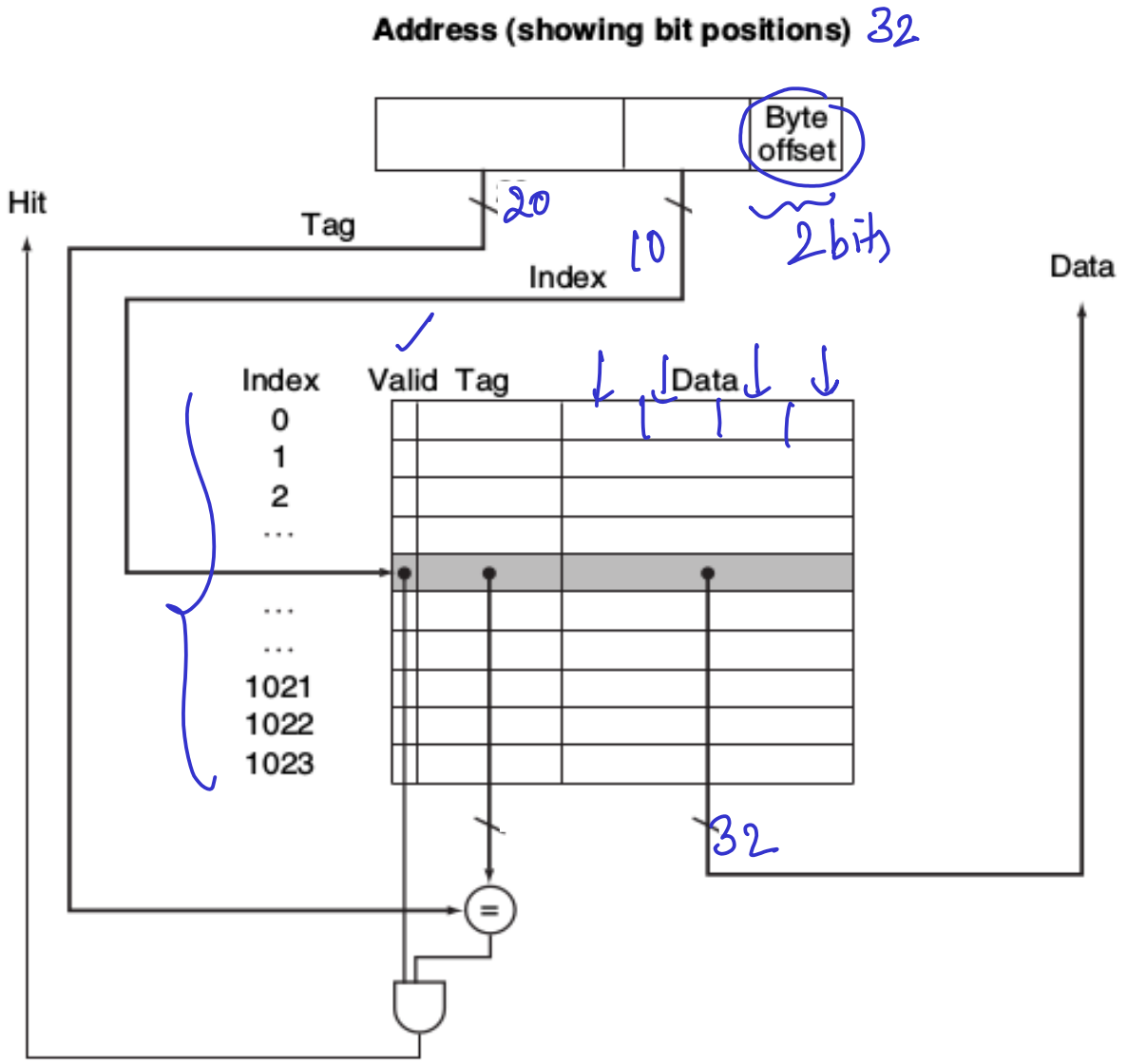
So, the 2 LSB bits of the address are not necessary to access the block as a whole.

The 2 LSB bits of the address are needed only if you need to access individual bytes.

DM cache implementation

Cache tag and address tag match : Hit Else Miss

How many tag bits?
Index bits?
Byte offset bits?



Accessing a cache of 8 block

Decimal address of reference	Binary address of reference
22	10110_{two}
26	11010_{two}
22	10110_{two}
26	11010_{two}
16	10000_{two}
3	00011_{two}
16	10000_{two}
18	10010_{two}
16	10000_{two}

8 block cache – initially empty

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

Address 22 – Miss or hit?
Cache content?

Accessing a cache of 8 lines

Decimal address of reference	Binary address of reference
22	10110_{two}
26	11010_{two}
22	10110_{two}
26	11010_{two}
16	10000_{two}
3	00011_{two}
16	10000_{two}
18	10010_{two}
16	10000_{two}

8 block cache – initially empty

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

Address 22 - Miss

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

Accessing a cache of 8 lines

Decimal address of reference	Binary address of reference
22	10110_{two}
26	11010_{two}
22	10110_{two}
26	11010_{two}
16	10000_{two}
3	00011_{two}
16	10000_{two}
18	10010_{two}
16	10000_{two}

Address 26 - Miss

Index	V	Tag	Data
000	N		
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

Next Address 22 and 26 - Hit

Accessing a cache of 8 lines

Decimal address of reference	Binary address of reference
22	10110_{two}
26	11010_{two}
22	10110_{two}
26	11010_{two}
16	10000_{two}
3	00011_{two}
16	10000_{two}
18	10010_{two}
16	10000_{two}

Address 16 - Miss

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

Address 3 - Miss

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	Y	00_{two}	Memory (00011_{two})
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

Accessing a cache of 8 lines

Decimal address of reference	Binary address of reference
22	10110_{two}
26	11010_{two}
22	10110_{two}
26	11010_{two}
16	10000_{two}
3	00011_{two}
16	10000_{two}
18	10010_{two}
16	10000_{two}

Temporal →

Spatial

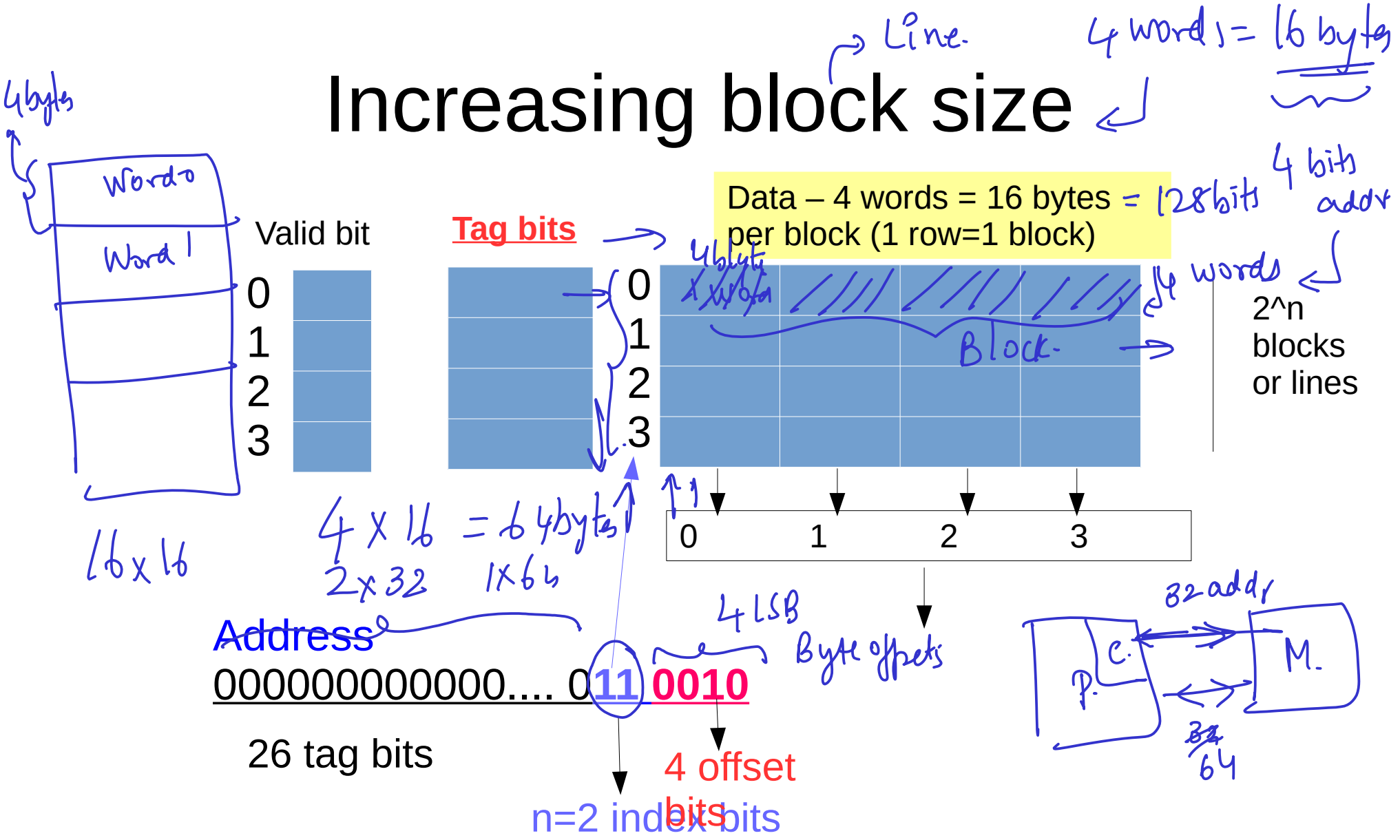
Address 16 - Hit

Address 18 - Miss

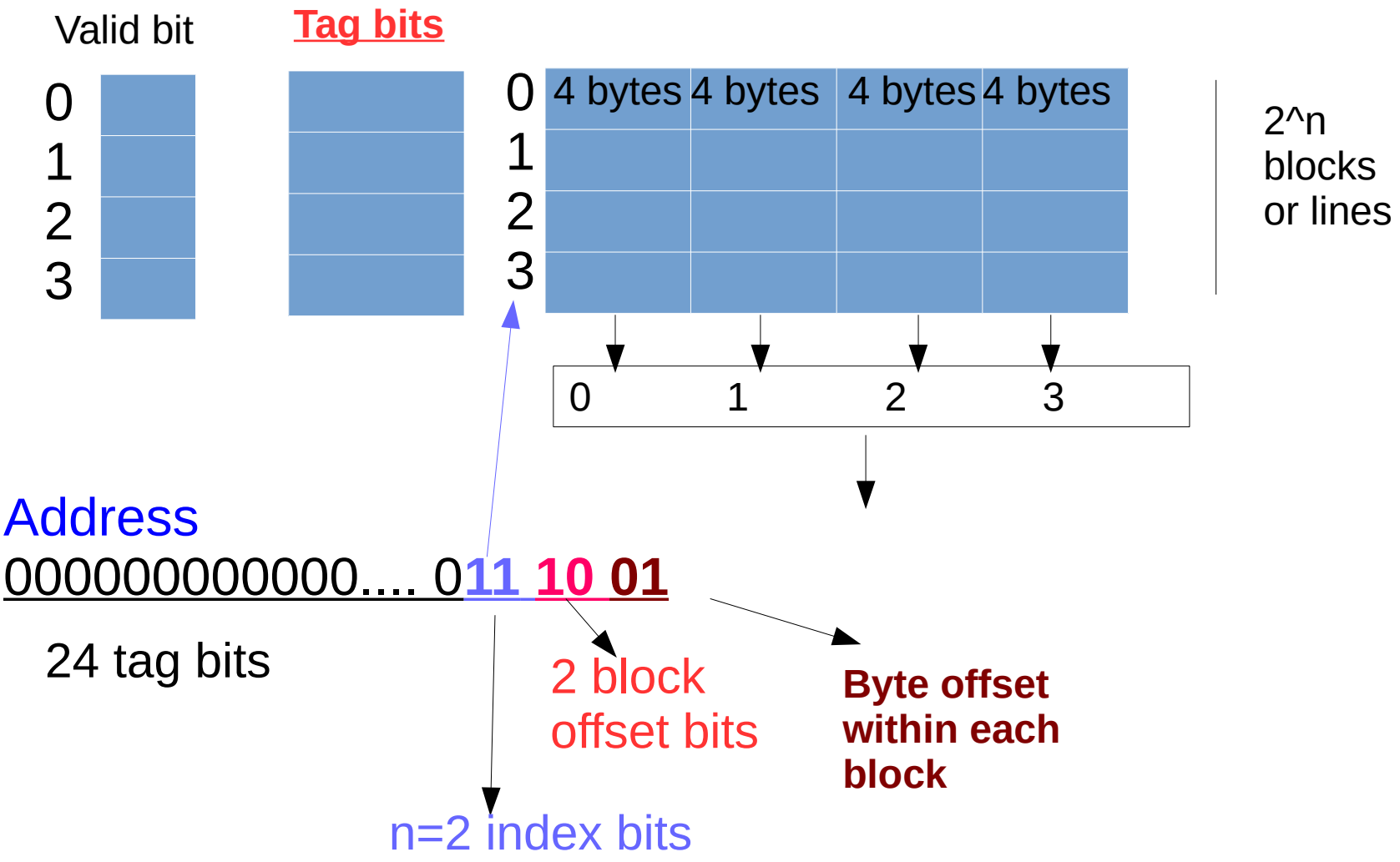
Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	Y	00_{two}	Memory (00011_{two})
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two}) → 1 Block.
001	N		
010	Y	10_{two}	Memory (10010_{two})
011	Y	00_{two}	Memory (00011_{two})
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

Increasing block size

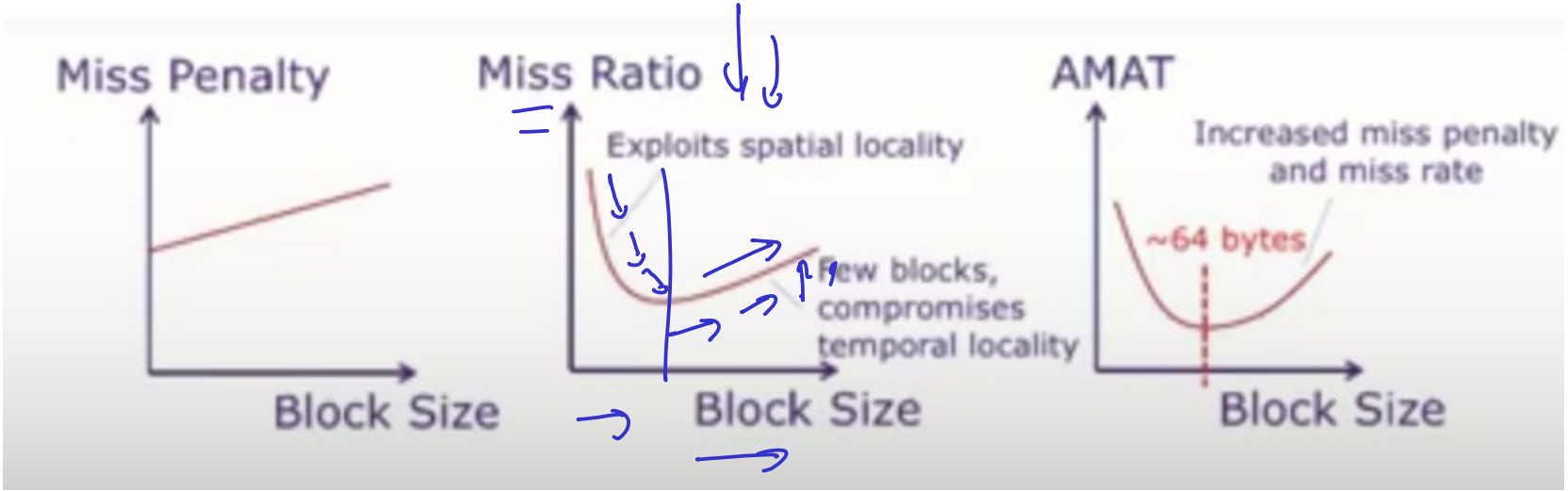


Increasing block size



Pros and cons

- Large blocks
  - Less tag overhead
  - Take advantage of spatial locality --> less Miss rate
  - Increase block size further --> Conflict misses, less data stored --> Miss rate increase
  - On a miss- replace entire block – wastage of bandwidth
  - Typical size = 16 words



Size of a cache - Number of bytes

- Cache size is 2^n lines = n bits for the index
- Block size is 2^m words = m bits for the word within the block = 2^(m+2) bytes
  - Tag size for a 32 bit address = 32 - (n+m+2)
- Number of bits in a DM cache:
  - 2^n * (block size + tag size + valid field size)
- Typically the tag and valid field are ignored

Cache size

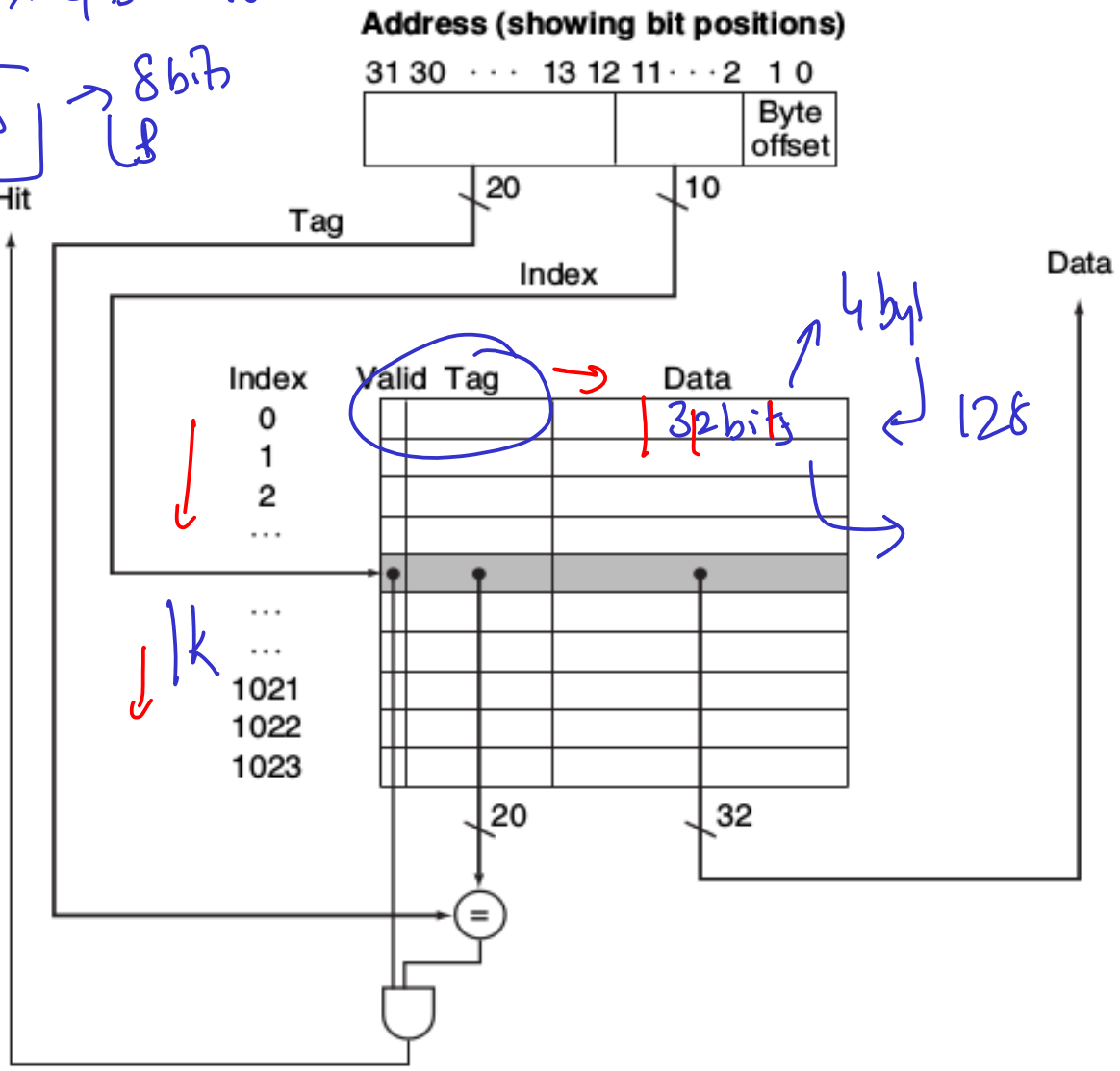
W = 32 bits

Block = Line = 1W; 2W, 4W, 16W.

4W x 4B = 16 B.

LW 32 bits → 8 bits LB

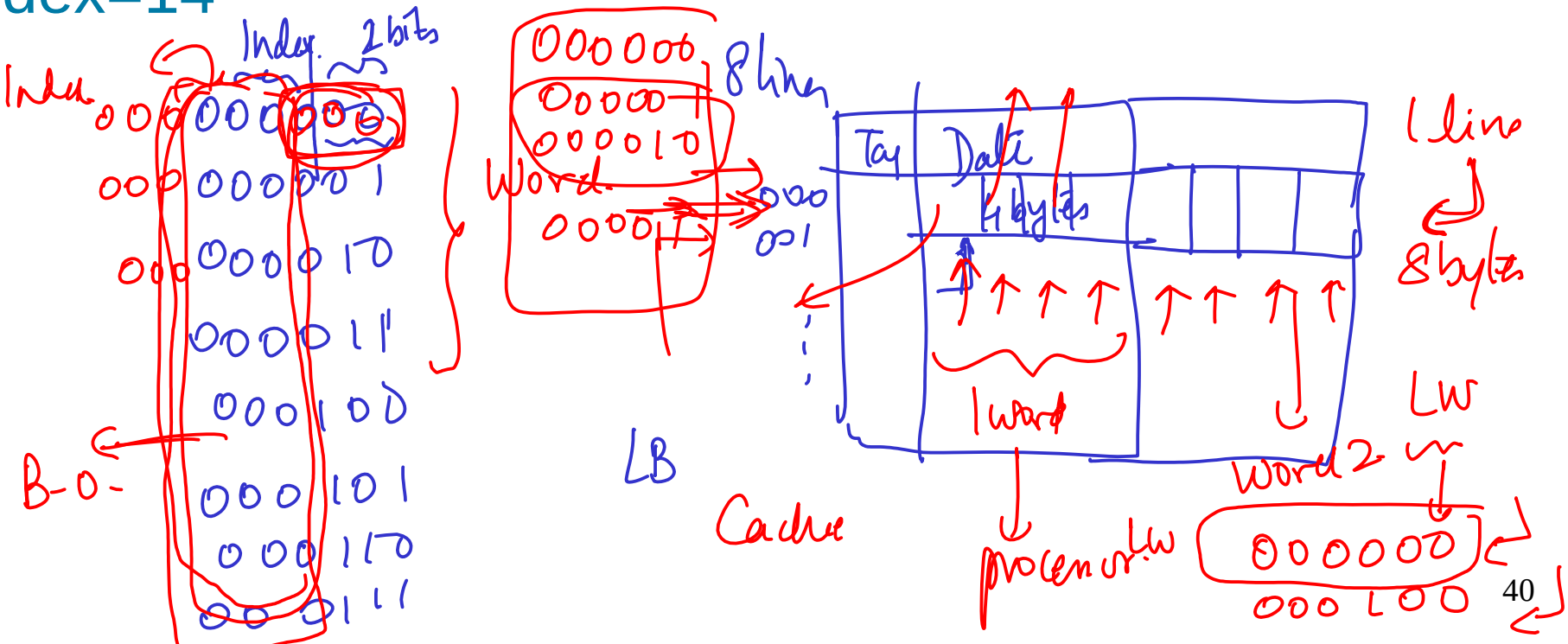
LB

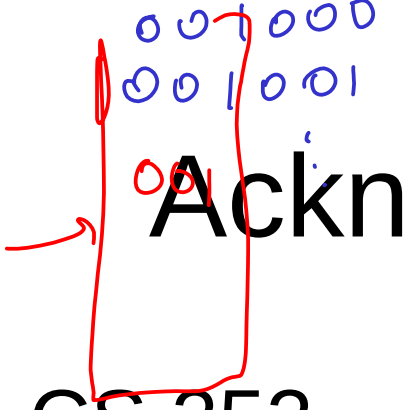


Size = 1024 * 4 bytes = 4kB cache

Acknowledgements

- MITx- 6.004- Computation Structures-
  - https://www.youtube.com/watch?v=hg0dTS10uSk&list=PLDSlqjcPpoL64CJdF0Qee5oWqGS6we_Yu&index=13
  - https://www.youtube.com/watch?v=-XqdkA931ag&list=PLDSlqjcPpoL64CJdF0Qee5oWqGS6we_Yu&index=14





Acknowledgements

- UCB- CS 252
  - Course page: http://inst.eecs.berkeley.edu/~cs152/sp18/
  - Memory: http://inst.eecs.berkeley.edu/~cs152/sp18/lectures/L05-Memory.pdf
  - http://inst.eecs.berkeley.edu/~cs152/sp18/lectures/L06-MemoryII.pdf
- Hennessey and Patterson
- Tech Review pages