How we store in cache

How to cut cost in cache storing

Size of cache

Book

Lecture – 19

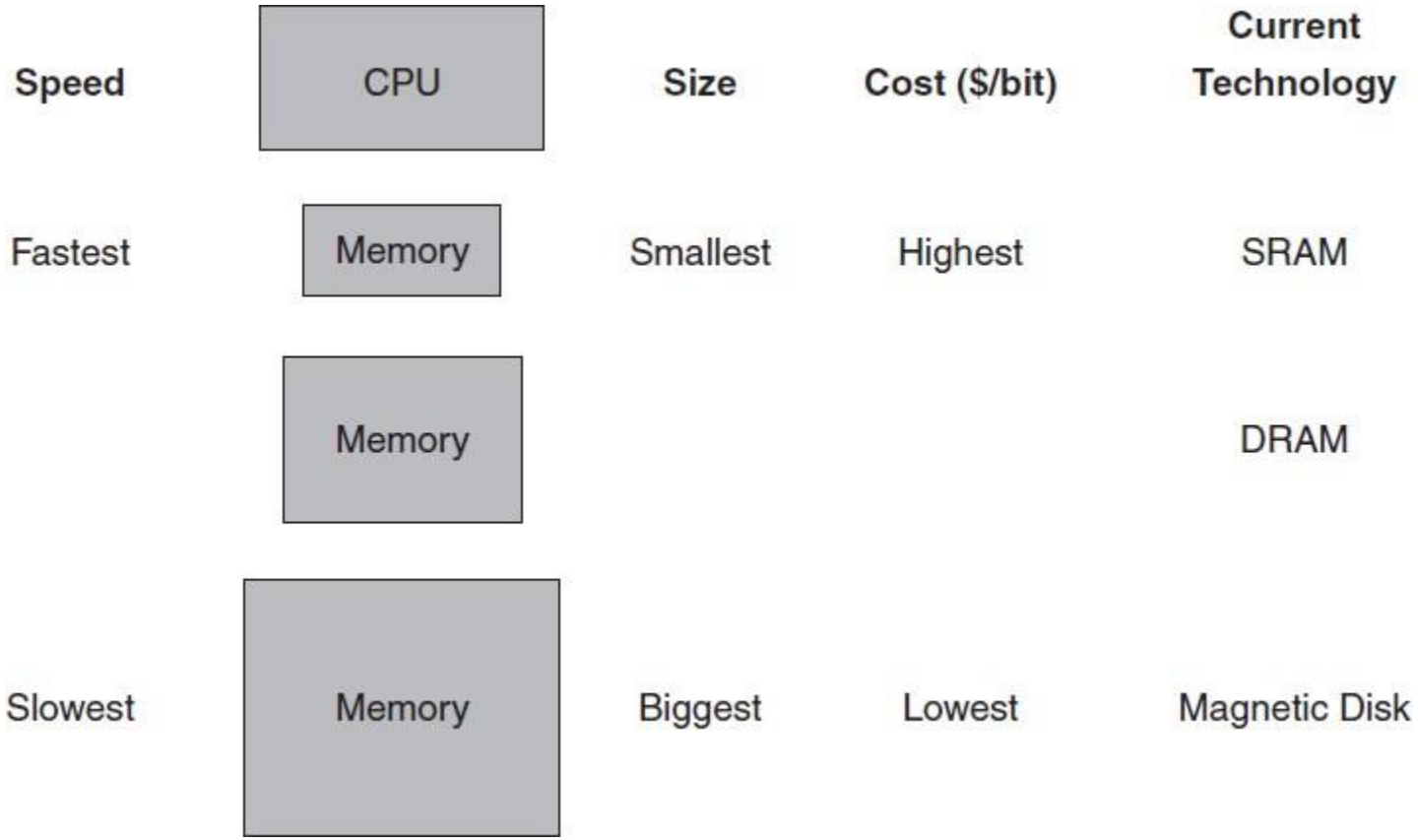
Memory Hierarchy

- A memory hierarchy consists of multiple levels of memory with different speeds and sizes.

- The faster memories are more expensive per bit than the slower memories and thus smaller.

- Three technologies used in building memory hierarchies:
  1. DRAM (Main memory)
  2. SRAM (Cache)
  3. Magnetic disk (Hard disk)

The Basic Structure of Memory Hierarchy

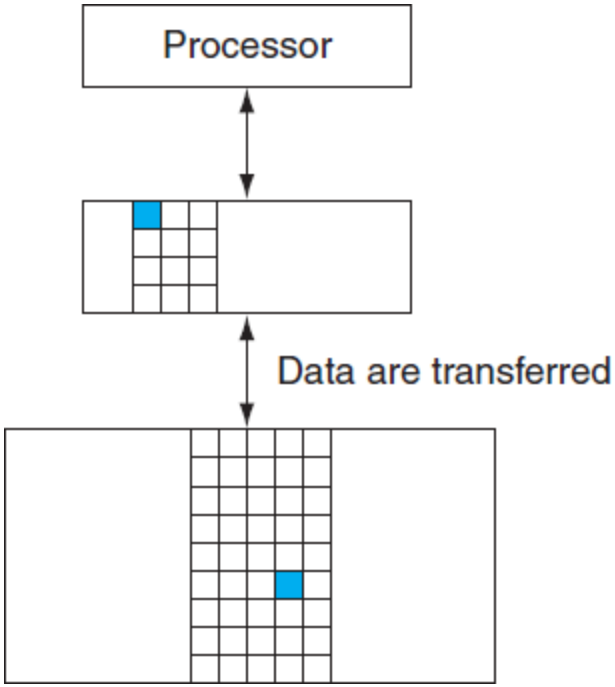


Memory Hierarchy

Memory Technology	Typical Access Time	$ Per GB in 2008
SRAM	0.5 – 2.5 ns	2000 – 5000
DRAM	50 – 70 ns	20 – 75
Magnetic Disk	5,000,000 – 20, 000,000 ns	0.20 – 2

- The goal is to provide the user with as much memory as is available in the cheapest technology, while providing access at the speed offered by the fastest memory.

Two level Hierarchy



Terminology

- **Block:**
  The minimum amount of information that can be either present or not present in the two level hierarchy.

- **Hit:**
  Data requested by the processor appears in some block in the upper level.

- **Miss:** → access data is absent & have to fetch from lower level
  Data requested by the processor is not present in the upper level.

- **Hit rate / Hit ratio:**
  The fraction of memory accesses found in the upper level.

- **Miss rate:**
  The fraction of memory accesses not found in the upper level. (1-Hit rate)

Terminology

- **Hit Time:**

  The time needed to access a level of the memory hierarchy, including the time required to determine whether the access is a hit or a miss.

- **Miss Penalty:** time to fetch data from lower level

  The time to replace a block in the upper level with the corresponding block from the lower level, plus the time to deliver this block to the processor.

Miss ratio = 1 − hit ratio

Principle of Locality

- It states that program access a relatively small portion of their address space at any instant of time.

- **Temporal Locality:**

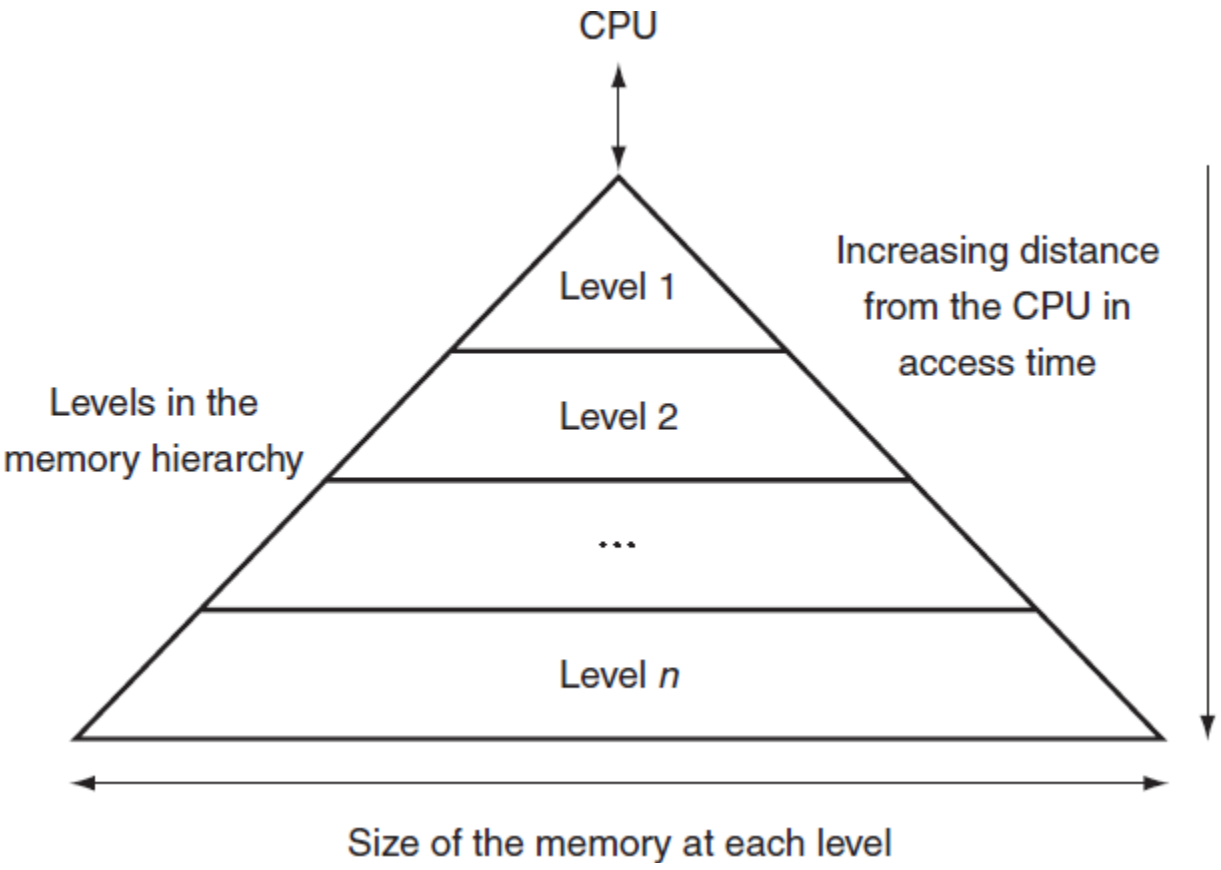
  If an item is referenced, it will tend to be referenced again soon. / can be accessed soon

- **Spatial Locality:**

  If an item is referenced, items whose addresses are close by will tend to be referenced soon.

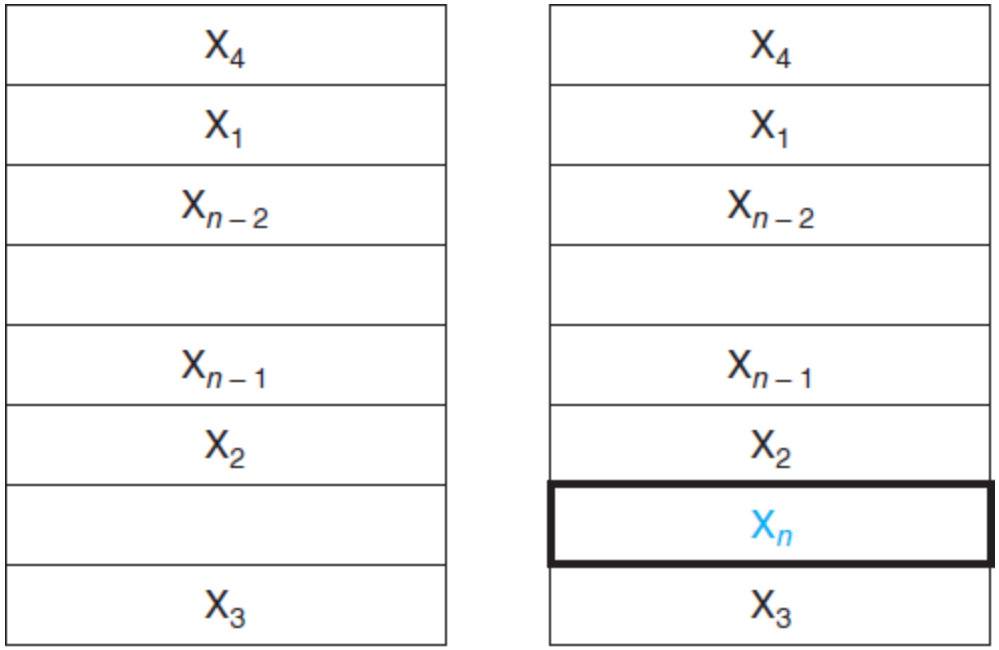
items near those accessed recently are likely to be accessed soon

Memory Hierarchy



Cache

- It refers to the level of memory hierarchy between the processor and main memory.



a. Before the reference to X_n b. After the reference to X_n

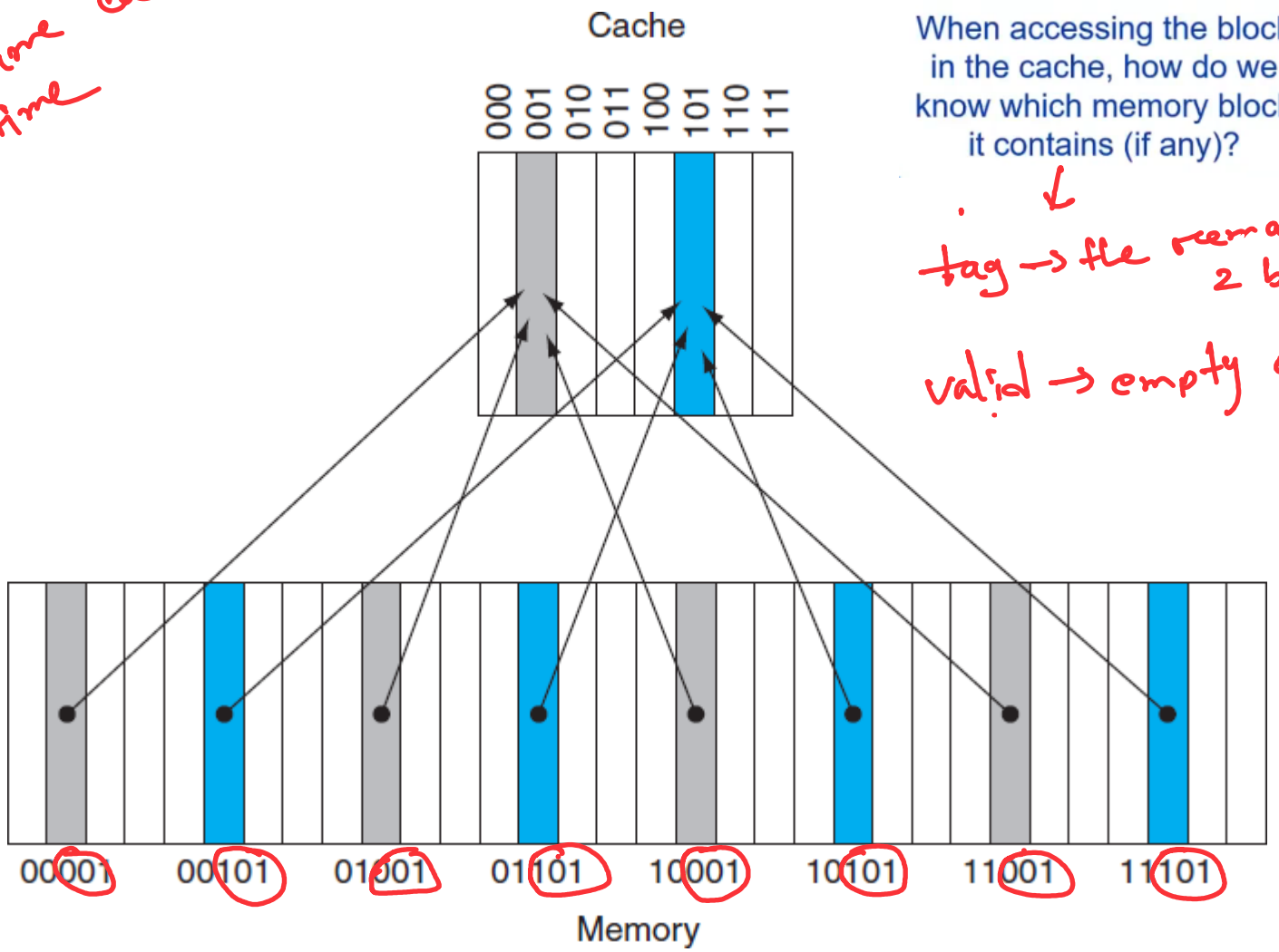
Direct Mapped Cache

Maps every block in memory to 1 specific block in the cache

- A cache structure in which each memory location is mapped to exactly one location in the cache.
- Assign cache location based on the address of the word in the memory.
- **Mapping:**
  (Block address) modulo (Number of cache blocks in the cache).
- Can accessed directly with the lower order bits.
- Each cache location can contain the contents of a number of different memory locations.

A Direct Mapped Cache

one of them use the same cache at a time



Tag and Valid Bit

- A field contains the address information required to identify whether a word in the cache corresponds to the requested word.

- It indicates that the associated block contains valid data.

Cache Example

Address	Hit/miss	Cache block
10110	miss	110
11010	miss	010
10110	hit	110
10010	miss	010

Index	V	Tag	Data
000	N		
001	N		
010	Y	10	Mem[10110]
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Accessing A Cache

Decimal address of reference	Binary address of reference	Hit or miss in cache	Assigned cache block (where found or placed)
22	10110_{two}	miss (7.6b)	$(10110_{two} \bmod 8) = 110_{two}$
26	11010_{two}	miss (7.6c)	$(11010_{two} \bmod 8) = 010_{two}$
22	10110_{two}	hit	$(10110_{two} \bmod 8) = 110_{two}$
26	11010_{two}	hit	$(11010_{two} \bmod 8) = 010_{two}$
16	10000_{two}	miss (7.6d)	$(10000_{two} \bmod 8) = 000_{two}$
3	00011_{two}	miss (7.6e)	$(00011_{two} \bmod 8) = 011_{two}$
16	10000_{two}	hit	$(10000_{two} \bmod 8) = 000_{two}$
18	10010_{two}	miss (7.6f)	$(10010_{two} \bmod 8) = 010_{two}$

Accessing A Cache

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

a. The initial state of the cache after power-on

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		

b. After handling a miss of address (10110_{two})

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

c. After handling a miss of address (11010_{two})

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		

d. After handling a miss of address (10000_{two})

Accessing A Cache

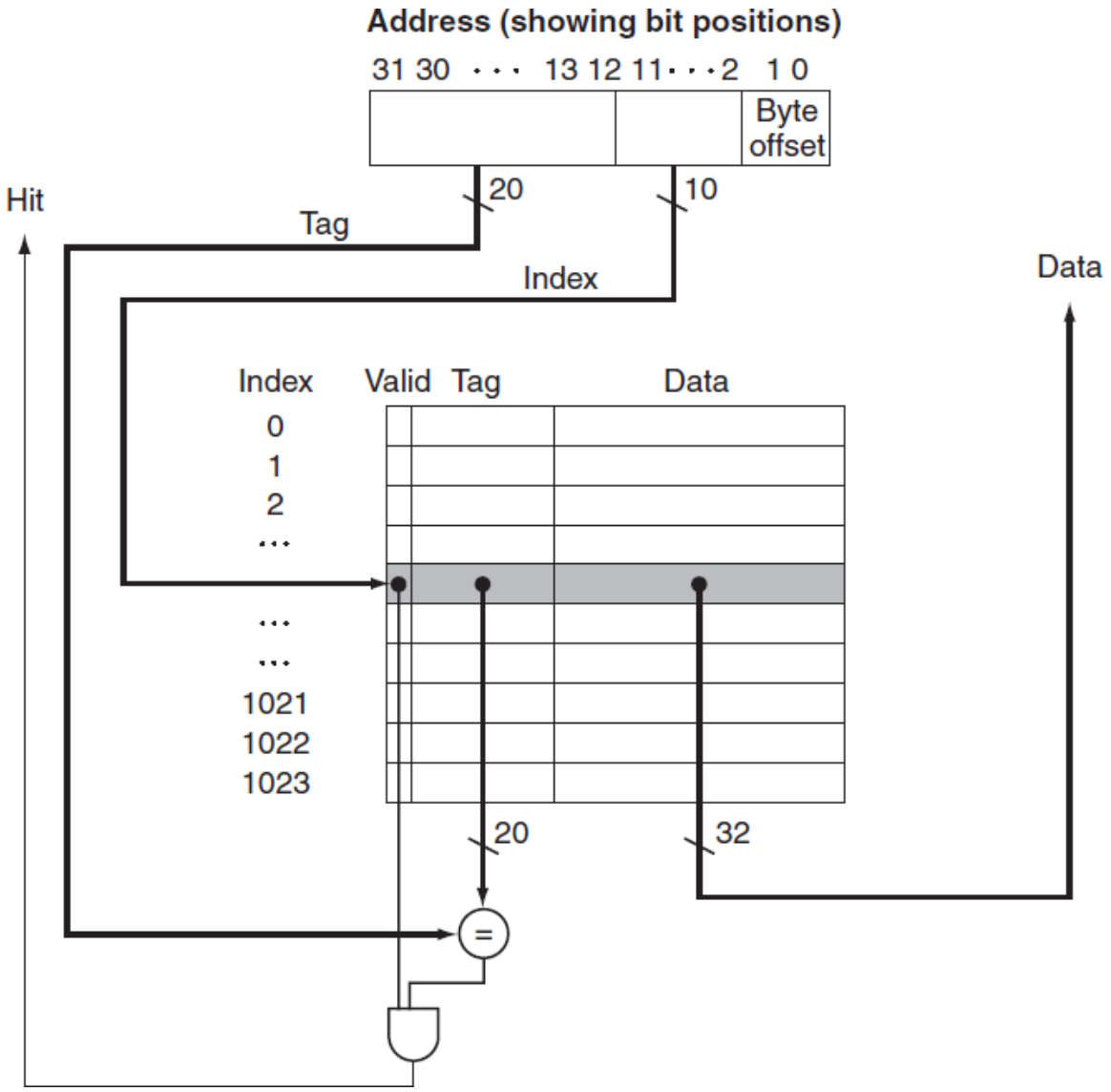
Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	Y	00_{two}	Memory (00011_{two})
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

e. After handling a miss of address (00011_{two})

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
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		

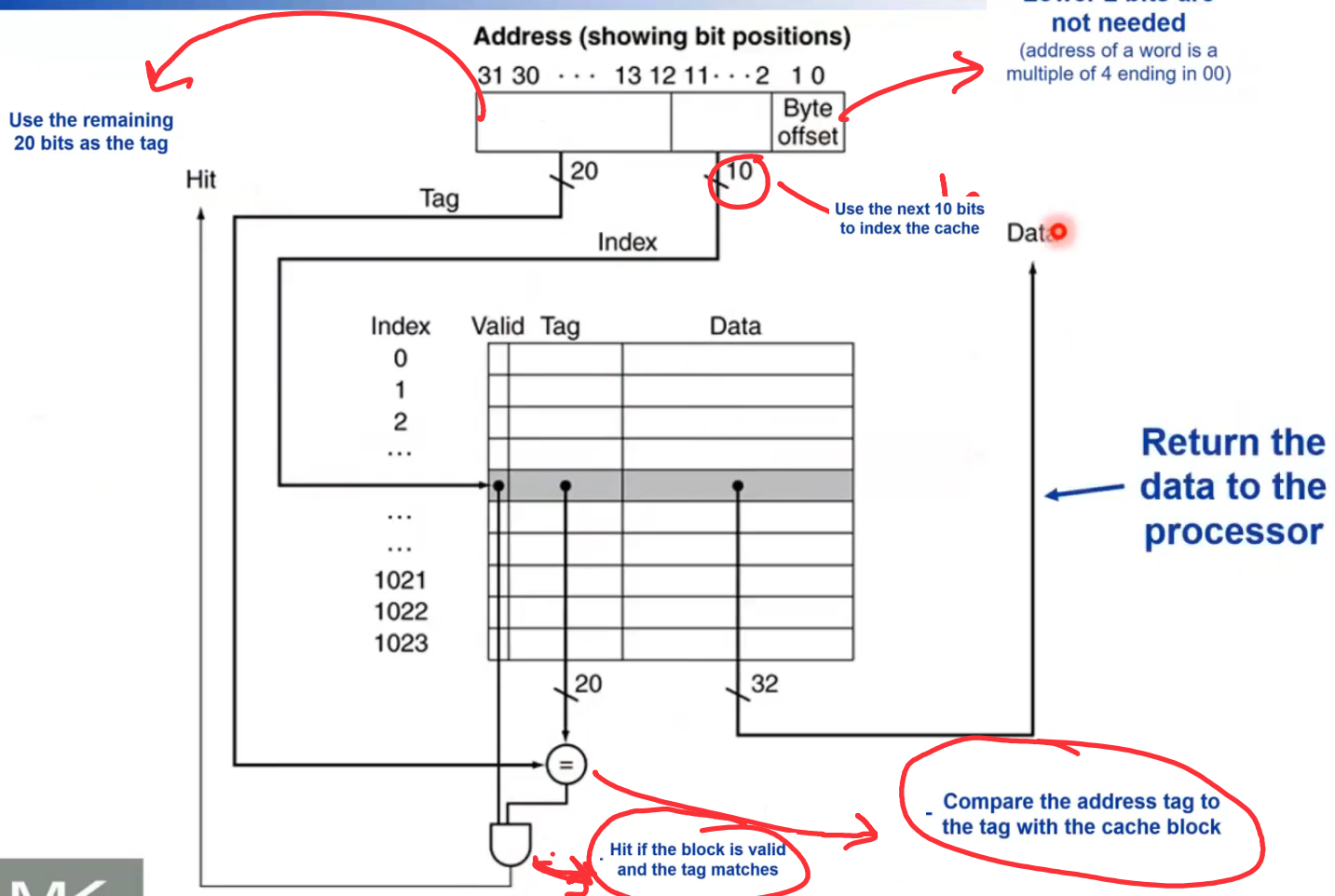
f. After handling a miss of address (10010_{two})

Referencing a Cache Block



Address Subdivision

Address Subdivision



Cache Size

- The total number of bits needed for a cache is a function of the cache size and the address size.

- Let address = 32 bits
  Cache size = 2^n blocks with 2^m words.
  Tag size = 32 – (n+m+2)

$$\text{Efficiency} = \frac{\text{data bits}}{\text{Total bits}}$$

Total cache size = block number × (words per block × 32 + tag bit + 1)

1. Word offset (2 bits)

- A word = 4 bytes (since word = 32 bits).
- To select **one byte inside a word**, we need:

$$\log_2(4) = 2 \text{ bits}$$

These **2 bits** are the **byte offset** (sometimes called word offset).

2. Block offset (m bits)

- Each block has 2^m words.
- To choose **one word inside the block**, we need:

$$m \text{ bits}$$

3. Index (n bits)

- Cache has 2^n blocks.
- To select which **cache block** the address maps to, we need:

$$n \text{ bits}$$

4. Tag (remaining bits)

Now we've already used:

- 2 bits for **word offset**
- m bits for **block offset**
- n bits for **index**

So the remaining bits are the **tag**:

$$\text{Tag size} = 32 - (n + m + 2)$$

Cache size = 2^n blocks with 2^m words,

they are describing:

- total cache words = 2^{n+m},
- total cache bytes = 2^{n+m+2},
- and address fields follow directly from this structure.

Bits in a Cache

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

Data size = 16 KB = 4K words = 2^{12} words.

Block size = 4 words (2^2). $m=2$

Cache Entries = 2^{10} blocks $n=10$

Block size = 4 × 32 = 128 bits.

Tag = 32-10-2-2 = 18 bits.

Valid bit = 1 bit

Total Cache size = 2^{10} × (128+18+1) = 147Kbits = 18.4 KB

Efficiency = $\frac{\text{data bits}}{\text{total bits}} = \frac{197 \times 10^3}{16 \times 8029 \times 8}$

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

data = 16 KB = (16 × 1024) bytes = 16384 bytes = $\frac{16384}{4}$ words

= 4096 words

= 2^{12} words

Each block = 4 words = 2^2 words (m word)

block = $\frac{2^{12}}{2^2}$ = 2^{10} blocks

= 1024 blocks

$m = 2$ $[2^m \text{ word} = 2^2 \text{ word}]$ $n = 10$ $[2^n = 2^{10} \text{ block}]$

tag bit = 32 − (2 + 10 + 2) = 18 bits

Cache size = blocks size (number) × (4 × 32 + tag + valid)

A direct-mapped cache uses a 32-bit address and has a total data size of **32 KB**. Each block contains **8 words**. Assuming each word is 4 bytes (32 bits), answer the following:

1. Calculate the number of blocks in the cache.
2. Find the number of bits for the tag, index, block offset, and byte offset.
3. Calculate the total number of bits required to implement the cache, including data bits, tag bits, and 1 valid bit per block.
4. Convert the total bits into KB.

① data = 32 Kb = (32×1024) byte = $\frac{32 \times 1024}{4}$ word = 8192 words = 2^{13} word

each block has 8 words = 2^3 words

∴ blocks = $\frac{2^{13}}{2^3}$ = 2^{10} blocks

② blocks = 2^{10} (2^n) | $n = 10$ (index)

Each block has 2^m words = 8 words | $m = 3$ (block offset)

tag = $32 - (n + m + 2)$ = $32 - (10 + 3 + 2)$

= 17

③ cache = $2^{10}\left(8 \times 32 + 17 + 1\right)$

= 9)

$\frac{2^{10}(8 \times 32 + 17 + 1)}{8 \times 1024}$ KB

Example Question:

A direct-mapped cache uses a 32-bit address and has a total data size of **8 KB**. Each block contains **2 words**. Each word is 4 bytes (32 bits).

Calculate:

1. Number of blocks in the cache
2. Number of bits for the tag, index, block offset, and byte offset
3. Total bits required for the cache (data + tag + valid bits)
4. Total cache size in KB

Effect of Larger Blocks

- Increasing the block size usually decreases the miss rate.

1. **Miss Rate Reduction**: Larger block sizes generally decrease the miss rate because they capture more spatial locality (nearby memory locations are likely to be accessed together).

- But the miss rate increases if the block size becomes too large.

2. **Miss Rate Increase**: If the block size becomes too large, the miss rate can increase due to:
   - Fewer blocks fitting in the cache (reduced cache capacity).
   - Blocks being evicted before many of their words are accessed (underutilization).

1. The number of blocks in the cache will be small.

2. A block will be bumped out of the cache before many of it's words are accessed.

- Increasing the block size will increase the cost of miss.

3. **Miss Cost**: Larger blocks also increase the cost of a miss because more data must be transferred from main memory on each miss.

Improving Miss Penalty due to Larger Block

- **Early Restart:**

  Restart execution as soon as the requested word from a block is available.

- **Requested Word First / Critical Word First:**

  Requested word is delivered first and then the rest of the block is delivered.

Impact of Block Size

- Advantages of larger blocks:
  - Reduces miss rate due to spatial locality
  - Amortize the overhead of the tag bits
- Disadvantages (assuming fixed-size cache):
  - Fewer blocks (increases miss rate)
  - Underutilizes blocks (pollution) if no spatial locality
  - Larger miss penalty (more bytes to fetch)
    - Can override benefit of reduced miss rate
    - Early restart and critical-word-first can help