# Caches are critical for Performance

- Caches reduce average data access time
  - Automatic replication of data closer to the processor
- Reduce bandwidth requirement
  - Data accesses issued by the process that are satisfied by the cache, do not appear on the interconnect
- Each processor has at least one level of cache
- Data is logically transferred from producer to consumer to main memory
  - ST Reg --> (to) Mem
  - LD Reg <-- (from) Mem

# Basics of Cache

# Caches - basic idea

Small, fast memory

Stores frequently-accessed *blocks* of memory.

Level of temporary memory storage between CPU and main memory. Improves overall memory speed by taking advantage of the principle of locality

When it fills up, discard some blocks and replace them with others.

Works well if we reuse data blocks

Examples:
Incrementing a variable
Loops
Function calls

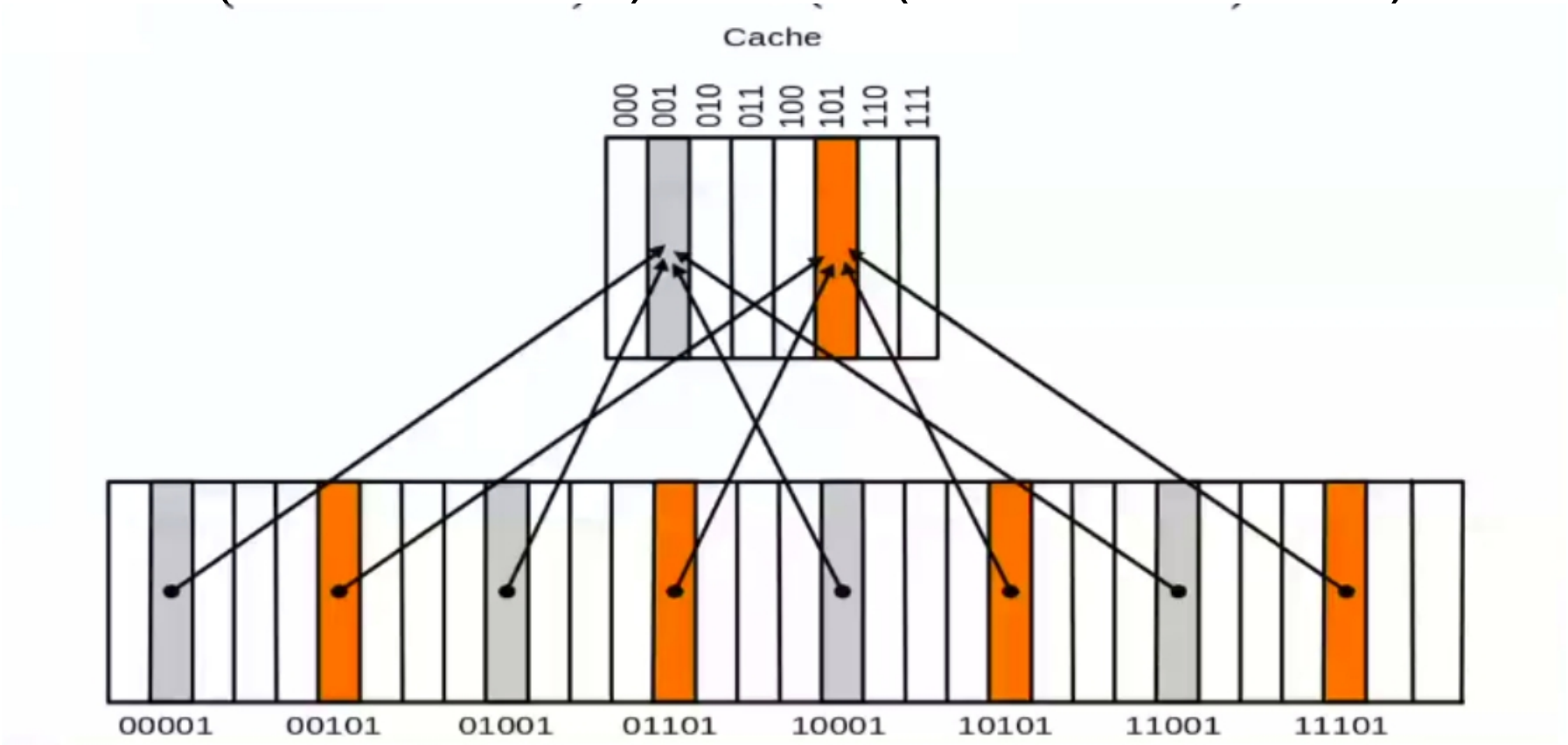
# Q1: Block Placement

- Where can block be placed in cache?
  - — In one predetermined place - direct-mapped
    - » Use part of address to calculate block location in cache
    - » Compare cache block with tag to check if block present
  - — Anywhere in cache - fully associative
    - » Compare tag to every block in cache
  - — in a limited set of places - set-associative
    - » Use portion of address to calculate set (like direct-mapped)
    - » Place in any block in the set
    - » Compare tag to every block in set
    - » Hybrid of direct mapped and fully associative

# Direct mapped Cache

Mapping

- Cache address is Memory address modulo the number of blocks in the cache
- Find a cache location:
  - (Block address) modulo (#Blocks in cache)



# Example: Accessing A Direct- Mapped Cache

DM cache contains 4 1-word blocks. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

DM Memory Access 1: Mapping: 0 mod 4 = 0

| Mem Block | DM Hit/Miss | | |
|---|---|---|---|
| | | Block 0 | Mem[0] |
| 0 | MISS | | |
| | | Block 1 | |
| | | Block 2 | |
| | | Block 3 | |

Set 0 is empty: write Mem[0]

# Example: Accessing A Direct- Mapped Cache

DM cache contains 4 1-word blocks. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

Memory Access 3: Mapping: 0 mod 4 = 0

| Mem Block | DM Hit/Miss | Block 0 | Mem[0] |
|---|---|---|---|
| 0 | miss | Block 1 | |
| 8 | miss | Block 2 | |
| 0 | miss | Block 3 | |

Set 0 contains Mem[8]. Overwrite with Mem[0]

# Example: Accessing A Direct- Mapped Cache

DM cache contains 4 1-word blocks. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

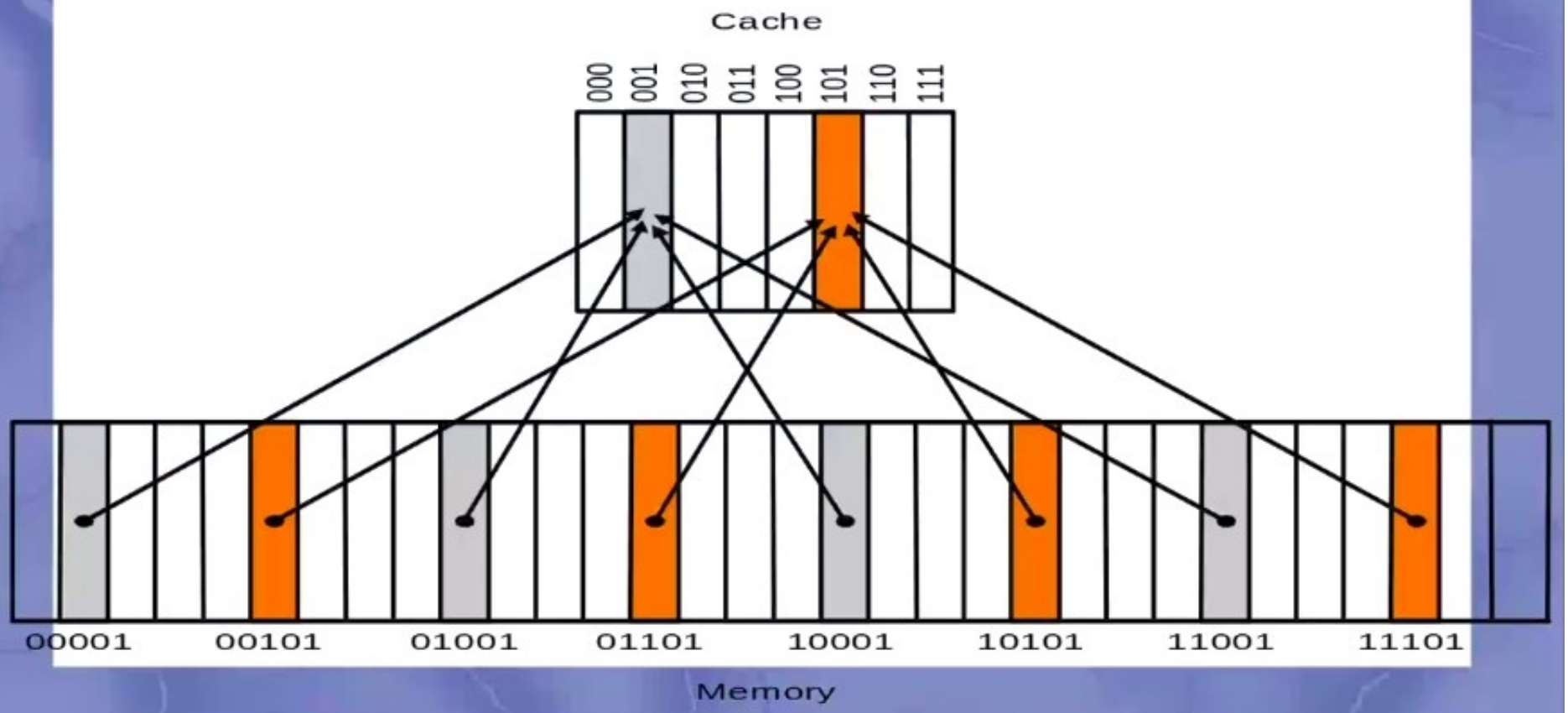
Memory Access 5: Mapping: 8 mod 4 = 0

| Mem Block | DM Hit/Miss |
|---|---|
| 0 | miss |
| 0 | miss |
| 0 | miss |
| 6 | miss |
| 8 | |

| | |
|---|---|
| Block 0 | Mem[0] |
| Block 1 | |
| Block 2 | Mem[6] |
| Block 3 | |

# Direct-Mapped Cache with n one-word blocks

- Pros: find data fast
- Con: What if access 00001 and 10001 repeatedly?



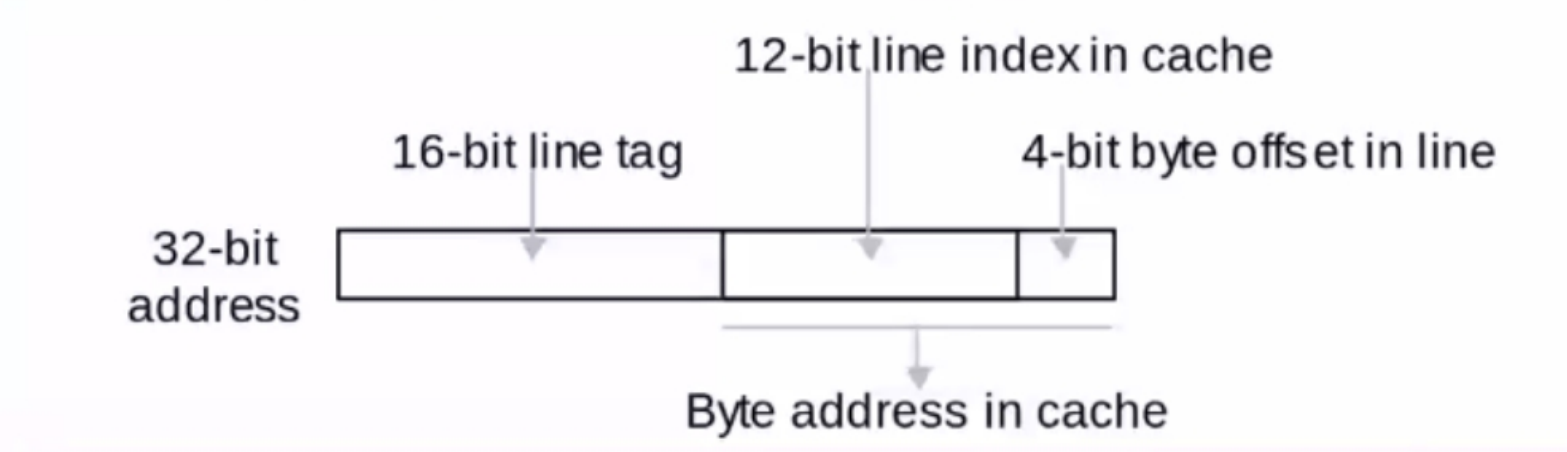
# Accessing a Direct-Mapped Cache

## Example

Show cache addressing for a byte-addressable memory with 32-bit addresses. Cache line *W* = 16 B. Cache size L = 4096 lines (64 KB).

Solution

Byte offset in line is log 16 = 4 b. Cache line index is $\log_2 4096 = 12$ b. This leaves 32 — 12 — 4 = 16 b for the tag.




Components of the 32-bit address in an example direct-mapped cache with byte addressing.