QUESTION: Now provide an example with a data block of 8 bytes and 4 cache lines, but use example memory addresses aligned to 4 bytes, instead of A,B,C,D,E. Also, explain how the tag is derived from the address. (To Google Gemini, Oct. 15, 2025)

That's a much more realistic example. Let's use memory addresses that are aligned to the 8-byte block size and explicitly show how the address is split into the Tag and Block Offset for this fully associative cache.

Fully Associative Cache Example (4-Byte Aligned Hex Addresses)

This example uses a fully associative cache with 4 lines, 8-byte blocks, and the **Least Recently Used (LRU)** replacement policy. The CPU performs 4-byte accesses, but the cache always loads/evicts 8-byte blocks.

Address Breakdown and Tag Derivation

For an **8-byte block**, the **Block Offset** uses the lowest 3 bits of the address ($\log_2(8) = 3$ bits). The **Tag** is the remainder, which identifies the 8-byte memory block.

Address Field	Bits (Position)	Value for Address **0xC**	Value for Address **0x10**
Tag (Block Identifier)	31 down to 3	**0x1** (From 0xC = 0001 100_2)	**0x2** (From 0x10 = 0010 000_2)
Block Offset (Byte within Block)	2 down to 0	100_2 **(4)**	000_2 **(0)**
Full Address (Hex)	31 down to 0	**0xC** (12)	**0x10** (16)

Memory Access Trace

The initial state has all 4 lines (L0-L3) as **Invalid** ($V = 0$). (Note that MRU refers to the cache line currently most recently used, and LRU refers to the

cache line currently least recently used.)

Step	Access	Full Address (Hex)	Block Offset	Block Tag (Hex)	Hit/Miss	Eviction Action (If Miss)	Final LRU State Change
1	**Read**	**0x8**	000_2	**0x1**	**Miss**	Load block **0x1** (0x8 to 0xF) into L3.	L3 → 1 (MRU)
2	**Read**	**0xC**	100_2	**0x1**	**Hit**	Match on **Tag = 0x1** in L3.	L3 → 1
3	**Read**	**0x10**	000_2	**0x2**	**Miss**	Load block **0x2** (0x10 to 0x17) into L2.	L2 → 1, L3 → 2
4	**Write**	**0x1C**	100_2	**0x3**	**Miss**	Load block **0x3** (0x18 to 0x1F) into L1, set **M = 1**.	L1 → 1, L2 → 2, L3 → 3
5	**Read**	**0x20**	000_2	**0x4**	**Miss**	Load block **0x4** (0x20 to 0x27) into L0. (Cache is full)	L0 → 1, L1 → 2, L2 → 3, L3 → 4 (LRU)
6	**Read**	**0x28**	000_2	**0x5**	**Miss**	**Evict L3 (Tag 0x1, M=0).** Load block **0x5**.	L3 → 1
7	**Read**	**0x14**	100_2	**0x2**	**Hit**	Match on **Tag = 0x2** in L2. Update LRU state.	L2 → 1, L3 → 2, L0 → 3, L1 → 4 (LRU)
8	**Write**	**0x30**	000_2	**0x6**	**Miss**	**Evict L1 (Tag 0x3, M=1). Write-back 0x3.** Load block **0x6**.	L1 → 1

Final Cache State (After Step 8)

Line	Tag (Hex)	Valid (V)	Modified (M)	LRU Rank (1=MRU, 4=LRU)
L0	**0x4**	1	0	3
L1	**0x6**	1	1	1 (MRU)
L2	**0x2**	1	0	2
L3	**0x5**	1	0	4 (LRU)

Key Eviction Detail (Step 8)

The **LRU line L1** was chosen for replacement. Because its **Modified bit (M) was 1** (containing **Tag = 0x3**), a **write-back** of the modified 8-byte block (addresses 0x18 to 0x1F) to main memory was required before the new block (**Tag = 0x6**) was loaded. This is the function of the Modified bit under a write-back policy.