Computer Systems Organization

Caching - II

Girish Varma

Reference: Chapter 10, in Sarangi

https://www.dropbox.com/s/vxq6cqnma92jdgd/Chapter-10.pdf?dl=0

Memory Technologies

Tech	Speed	Power/Area/Price
Flip Flop	< 1	Highiest
SRAM	10	Cheaper
DRAM	100	Cheapest

What to pick for processor design?

Caching

Use a combination of different technologies, to maximise speed and power consumption/price.

Based on some principles of memory access, that are observed in all programs:

Caching

Use a combination of different technologies, to maximise speed and power consumption/price.

Based on some principles of memory access, that are observed in all programs:

- **Temporal Locality**: If a memory location is accessed, with high probability it will be accessed again.

Caching

Use a combination of different technologies, to maximise speed and power consumption/price.

Based on some principles of memory access, that are observed in all programs:

- **Temporal Locality**: If a memory location is accessed, with high probability it will be accessed again.
- **Spatial Locality**: If a memory location is accessed, with high probability nearby locations will also be accessed again.

Caching: Memory Hierarchy

- **L1 Cache**: small SRAM array (8-64 KB)
- **L2 Cache**: larger SRAM array (128 KB - 4 MB)
- **Main Memory**: Large DRAM (4GB in a 32 bit system)

Caching: Memory Hierarchy

- **L1 Cache**: small SRAM array (8-64 KB)
- **L2 Cache**: larger SRAM array (128 KB - 4 MB)
- **Main Memory**: Large DRAM (4GB in a 32 bit system)

values(L1) ⊂ values(L2) ⊂ values(main memory)

Example

32 bit system (4GB main memory) and 8KB L1 Cache with block size of 64 bytes.

- Total no of blocks =

Example

32 bit system (4GB main memory) and 8KB L1 Cache with block size of 64 bytes.

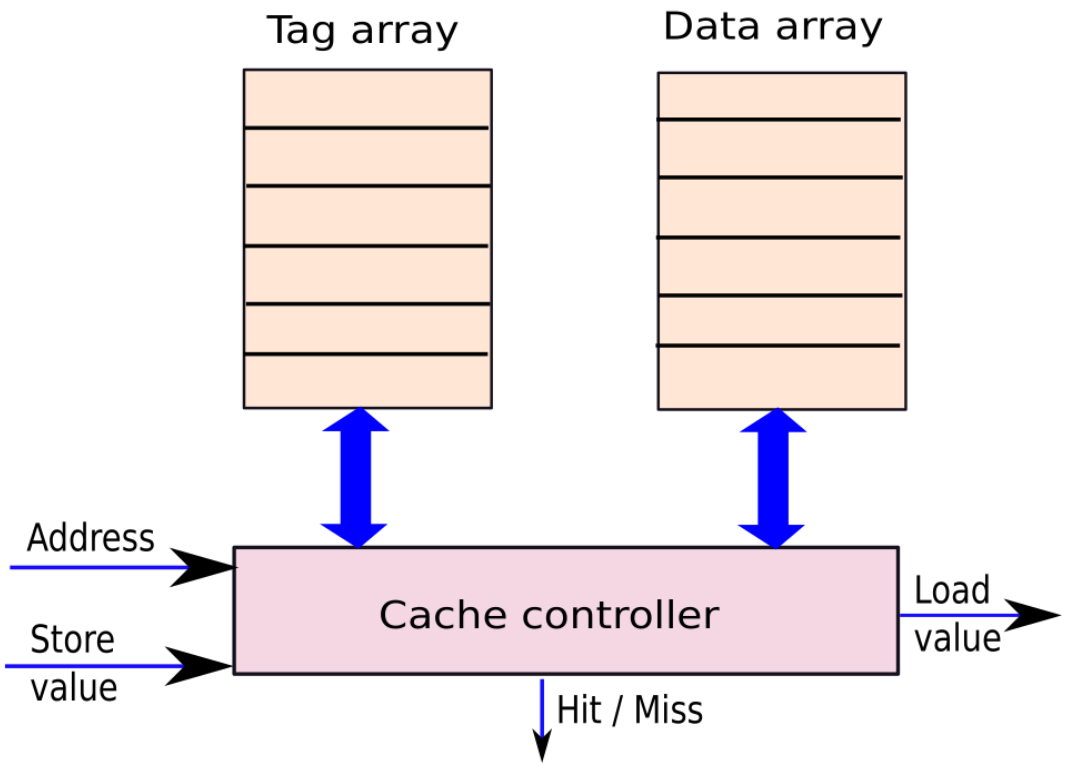
- Total no of blocks $= 2^{32}/64 = 2^{26}$ blocks.
- No. of blocks in L1 Cache =

Example

32 bit system (4GB main memory) and 8KB L1 Cache with block size of 64 bytes.

- Total no of blocks $= 2^{32}/64 = 2^{26}$ blocks.
- No. of blocks in L1 Cache $= 8KB/64B = 2^7 = 128$ blocks.

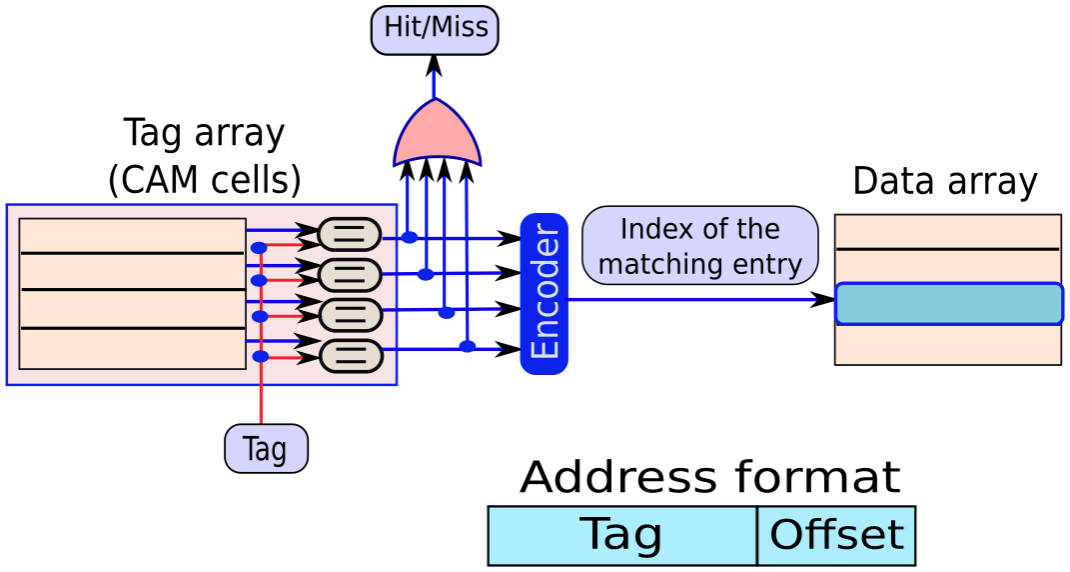
Cache Structure



Tag stores the address values, and data store the corresponding memory values.

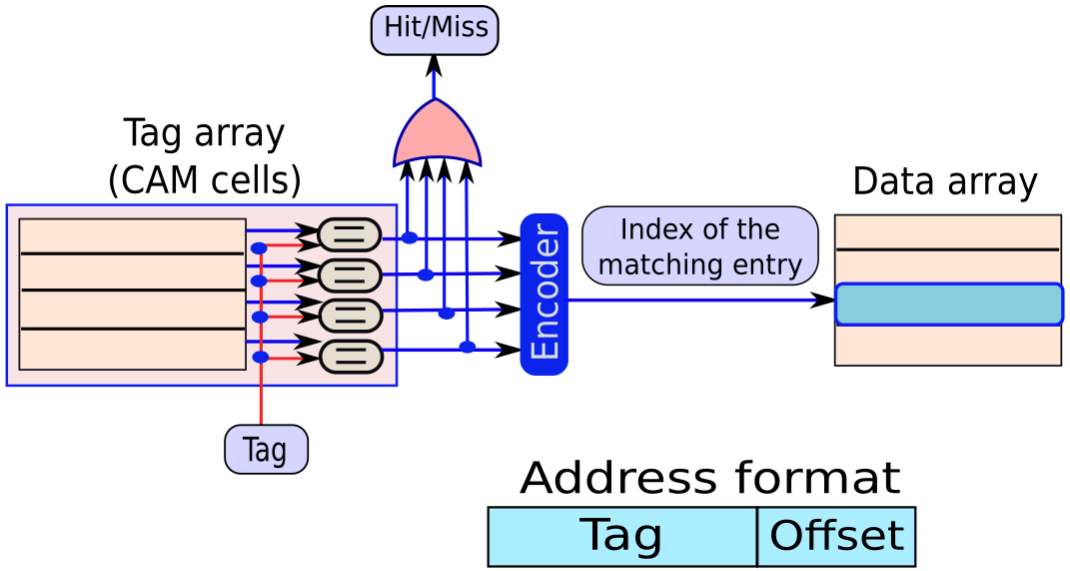
Fully Associative (FA) Cache

- Any memory location can taken any of the 128 positions in the cache.
- Tag is 26 bit block address.
- Compare all 128 tags to find a match.



Fully Associative (FA) Cache

- Any memory location can taken any of the 128 positions in the cache.
- Tag is 26 bit block address.
- Compare all 128 tags to find a match.



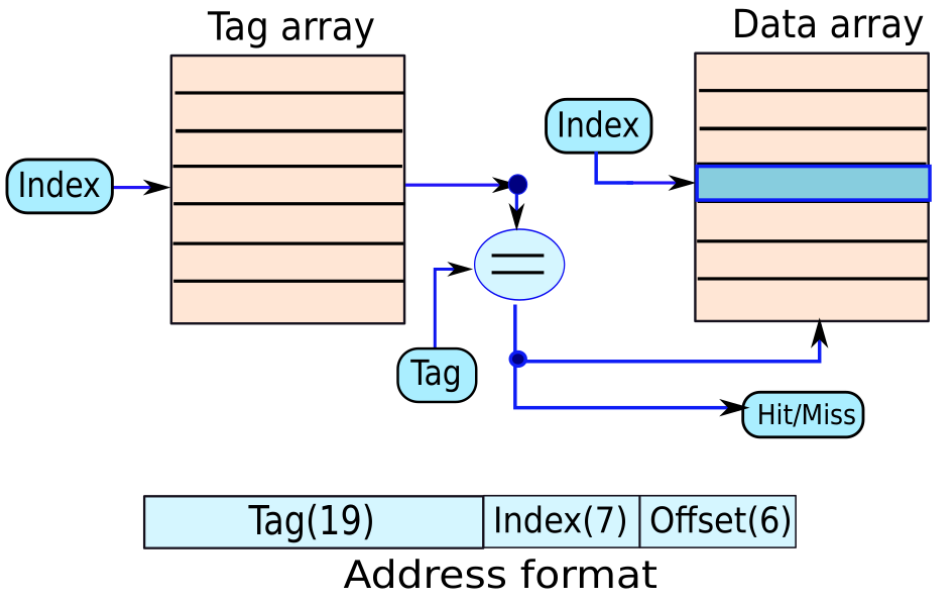
- Only possible for small caches.

Direct Mapped (DM) Cache

- A memory block add A can only 1 of the 128 positions in the cache given by

$$A\%128.$$

- That is extract 7 LSB out of 26 bit block address.
- Tag is $26 - 7 = 19$ bits.

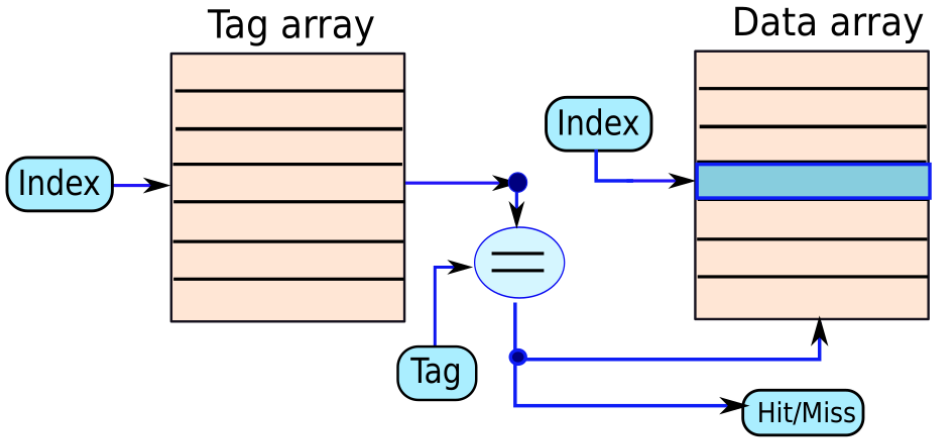


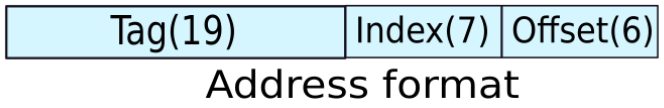
Direct Mapped (DM) Cache

- A memory block add A can only 1 of the 128 positions in the cache given by

$$A\%128.$$

- That is extract 7 LSB out of 26 bit block address.
- Tag is $26 - 7 = 19$ bits.



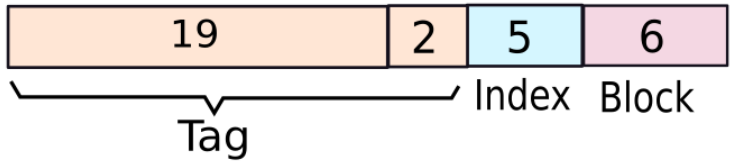


Address format

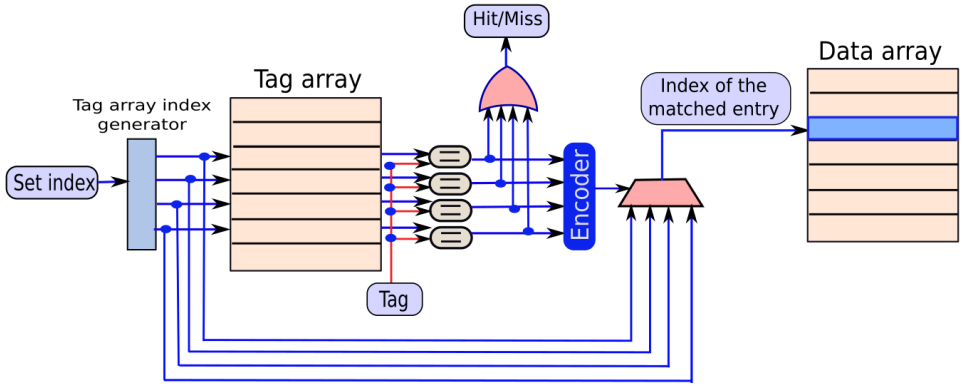
- High probability of cache miss.

Set Associative Cache (Hybrid)

- A memory block can reside in a small subset (say of size 4) of locations in the cache.
- Have to check all (4) locations in the set to declare hit or miss.



- 5 bit set index



Problem

A cache has the following parameters in a 32-bit system.

Parameter	*Value*
Size	N
Associativity	K
Block Size	B

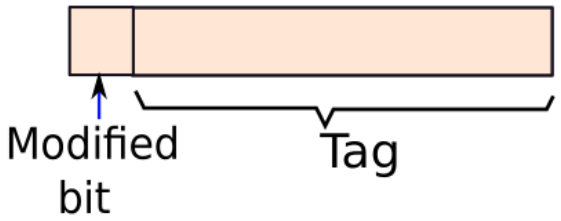
What is the size of the tag?

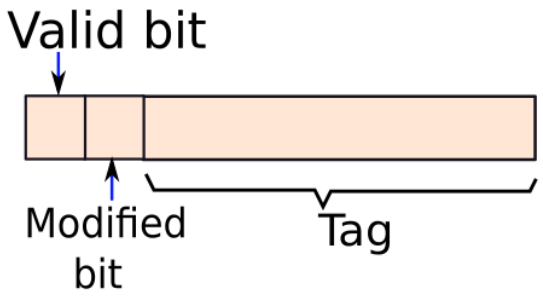
Cache: Operations

- data read
- data write
- insert
- replace
- evict

Cache write

- Write Through
- Write back





Cache replacement policy

- Random
- FIFO
- LRU

Cache Read/Write