Discussion 13

11/19/02 Tue

Locality

When instructions or data are accessed, they usually show locality.

- Temporal Locality:

  Temporal locality is the property that instructions or data are accessed multiple times over time.

  In a loop, instructions and data are accessed repeatedly.

  A piece of code called as a procedure.

- Spatial locality

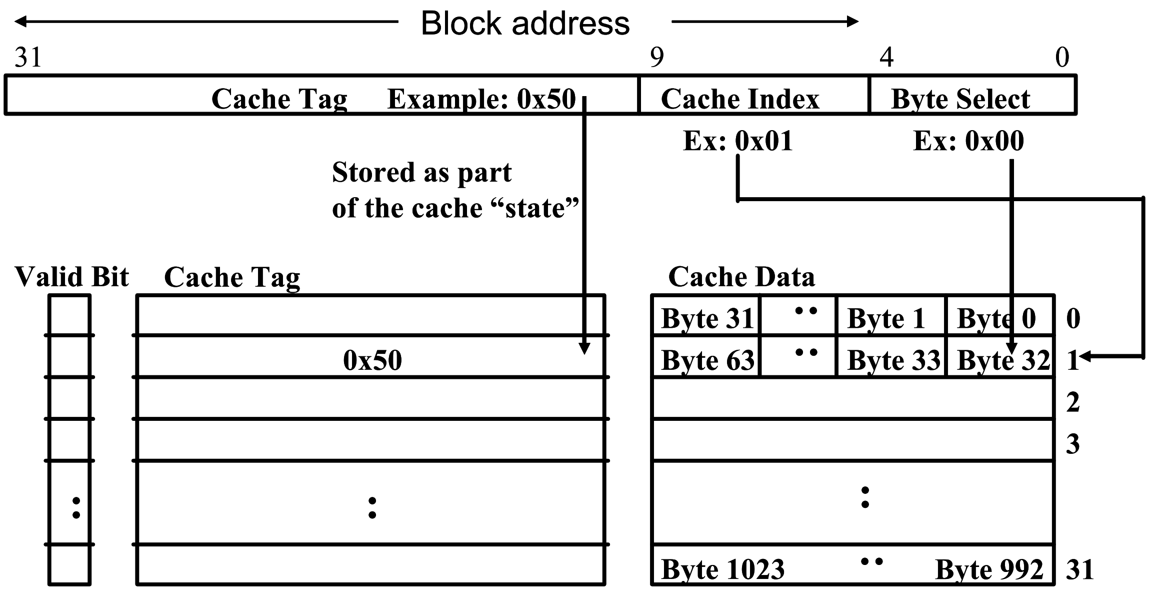
  Spatial locality is the property is that instructions or data in consecutive addresses are accessed over time.

  Instructions are usually executed in sequence.

  Data in an array are usually accessed together.

Most programs show locality. Thus it is possible to keep actively used instructions or data in a small subset.

Mapping memory address to cache



Cache is a rectangular table. Byte offset is used to identify a byte within a cache line. Index is used to identify the row in the cache.

All the addresses whose index bits and byte offset bits are the same are mapped to the same cache line. To tell the different addresses, we keep the remaining upper bits of address bits as a tag.

The relationship between metrics are as follows:

Cache size = Number of cache lines * cache line size (or block size)

Number of cache lines = 2^{index} bits

cache line size = 2^{byte} offset bits

Tag bits = Number of bits in a word – index bits – offset bits

Cache hit and miss.

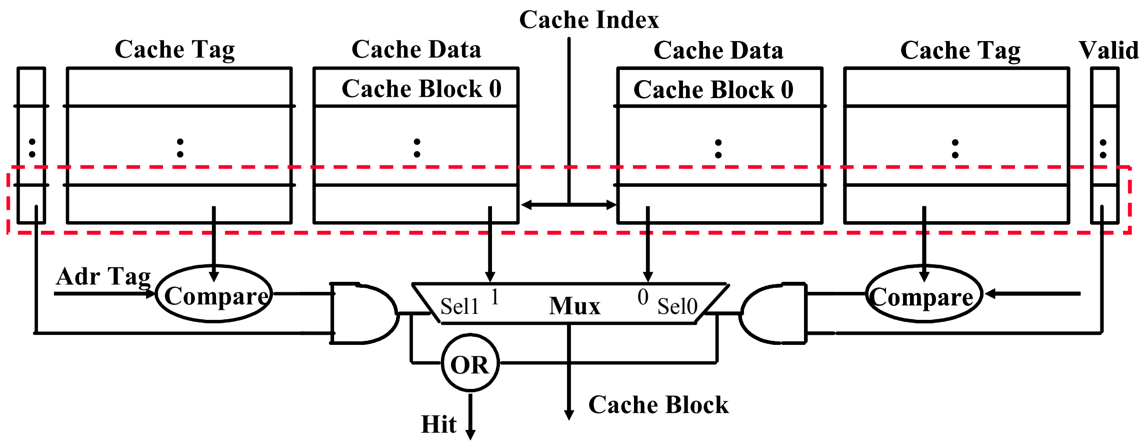
If the cache has the data when we access it, the cache is said to have a hit. Otherwise, it has a miss.

Hit rate = number of hits / number of cache accesses

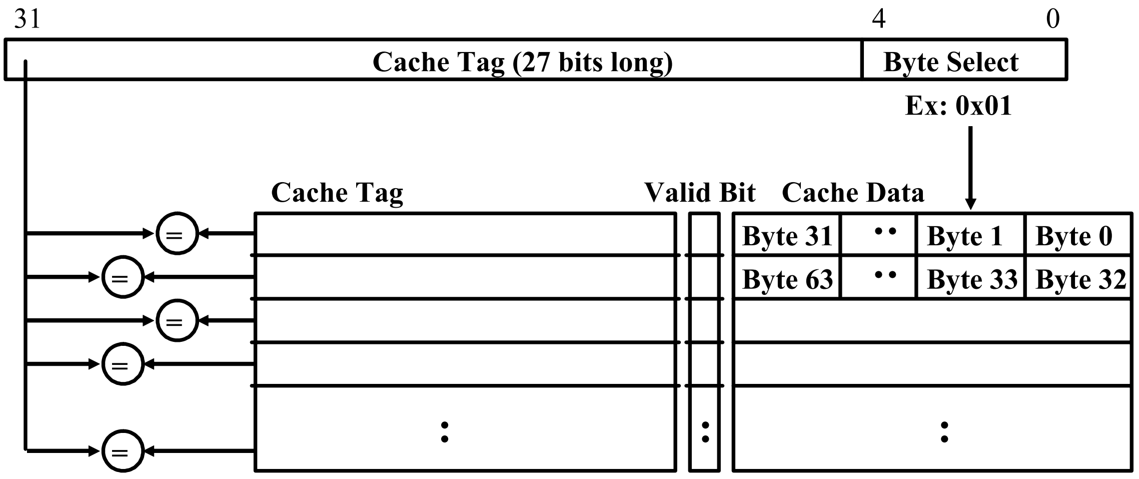
Miss rate = number of misses / number of cache accesses = 1 – hit rate

Set associative and fully associative cache

In a direct mapped cache, among the addresses mapped to a cache index, the cache can contain only one of the addresses. Thus, two addresses mapped to the same location is accessed alternately, then cache miss occurs each time. The idea of set associative cache is that to increase a cache block so that it contains multiple blocks and reduce the possibility of conflicts. In a set associative cache, the bigger block that contains multiple cache blocks are called a set.



Fully associative cache is extreme version of set associative cache. All the cache lines are in one row and they are looked up together for a machine address.



Example 1

Suppose we have a 16KB of data in a direct-mapped cache with 4 word blocks.

Cache size = 16KB = 16 * 2^10 bytes
cache line size = 4 words = 4 * 4 bytes = 16 bytes
Number of cache lines = 16 * 2^10 bytes / 16 bytes = 2^10

Index bits = 10
Offset bits = 4
Tag bits = 32 – 10 – 4 = 18

Suppose we have a 16KB of data in a 2-wat set associative cache with 4 word blocks. Find the size of index, offset and tag bits.

Cache size = 16 * 2^10 bytes
cache line size = 16 bytes
Set size = cache line size * set associativity = 16 bytes * 2 = 32 bytes

Number of sets = 16 * 2^10 bytes / 32 bytes = 2^9

Index bits = 9
Offset bits = 4
Tag bits = 32 – 9 – 4 = 19

Example 2:

Assume that we have a 32-bit processor (with 32-bit words) and that this processor is byte-addressed (i.e. addresses specify bytes).

Suppose that it has a 512-byte cache that is twoway set-associative, has 4-word cache lines, and uses LRU replacement.

Split the 32-bit address into “tag”, “index”, and “cache-line offset” pieces. Which address bits comprise each piece?

tag:

index:

cache-line offset:

How many sets does this cache have? Explain.

4 bits in the index $\Rightarrow$ 16 sets

Cause of cache misses

We can categorize the causes of cache misses in three. It is also called 3 C’s

- Compulsory miss:

  When a data is accessed first time, it is not in the cache. This is called compulsory miss.

- Conflict miss:

  Suppose a data loaded and replaced by some other data. hen it is accessed again, it is not in the cache because it is replaced. This is called conflict miss.

- Capacity miss:

  In a fully associative cache, we reduce the possibility of conflict misses by using all the available cache lines. But if all the lines are in use, then one of them should be replaced. And miss can occur when we try to access the replaced data. This is called capacity miss. The notion of capacity miss can be a little bit confusing. We need to know that capacity miss happens only in fully associative cache.

Example 3:

Below is a series of memory read references set to the cache from part example 2. Assume that the cache is initially empty and classify each memory references as a hit or a miss. Identify
each miss as either **compulsory, conflict, or capacity.** One example is shown. *Hint: start by*
splitting the address into components. Show your work.

Address	Hit/Miss?	Miss Type?
0x300		
0x1BC		
0x206		
0x109		
0x308		
0x1A1		
0x1B1		
0x2AE		
0x3B2		
0x10C		
0x205		
0x301		
0x3AE		
0x1A8		
0x3A1		
0x1BA		

Calculate the miss rate and hit rate.

Example 4:

You have a 500 MHz processor with 2-levels of cache.

Assume that it has a Harvard architecture (separate instruction and data cache at level 1).

Assume that the memory system has the following parameters:

First-level cache	Hit Time: 1 cycle Miss Rate: 4% data, 1 % instructions Block Size: 64 bytes
Second-level cache	Hit Time: 20 cycles + 1 cycle/64bits Miss Rate: 2% Miss Penalty: 200 cycles.

Instructions and data never cause TLB misses.

What is the average memory access time for instructions and data (assume all reads)?

Back to Top

Revised: 11/17/02.