Computer Systems Lecture 16

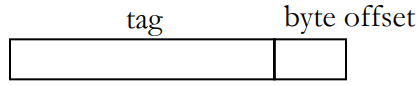
Cache Basics

In main memory, data is identified by its full 32-bit address

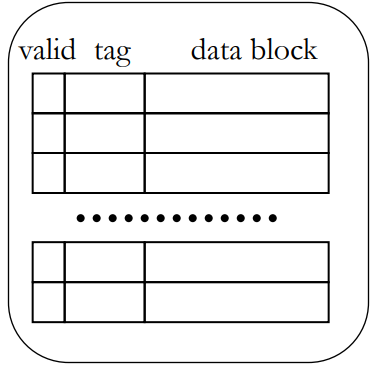
How do we map a 32-bit address to a much smaller memory, such as cache? We associate each data block in the cache with a tag word, indicating the address of the main memory block it holds and a valid bit, indicating whether the block is in use.

Fully-Associative Cache

Requested address:



Cache:

The correct cache block is identified by matching the tags, then the desired word/byte within the block is selected by the byte offset.

The Address tag can potentially match the tag of any cache block so we need to check all of them.

Each block consists of 16 words, making the byte offset 6 bits long (as the cache needs to be byte addressable). This leaves 28 bytes left for the tag.

Cache Replacement

* Least Recently Used (LRU)
  + Evict the cache clock that hasn’t been accessed for the longest time
  + Relies on past behaviour as a predictor of the future
* FIFO (first in first out, replace in the same order in which it was used)
  + This is simple to implement

Example:

* Cache has 4 blocks
* ACCESS ADDRESS: 0 2 6 0 7 8

In LRU first we’d add 0, then 2, then 6, then we access the 0 again, then we add the 7 and our cache is full, so when we try and add 8 we look at our least recently used element which is the 2 and replace it.

In FIFO it goes the same way until we try and add 8, at which point we replace the 0 instead.

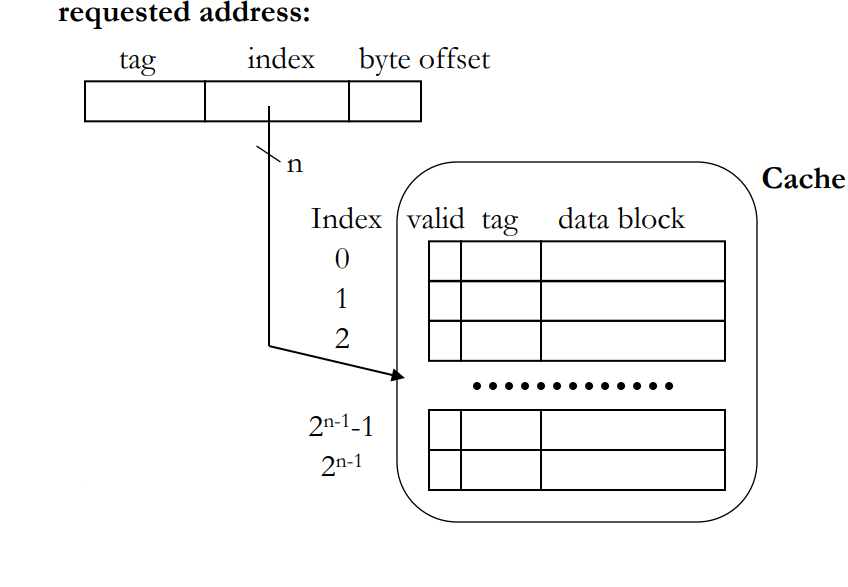
Direct-Mapped Cache

In a fully-associative cache, the search for matching tags is either very slow, or requires a very expensive memory type called Content Addressable Memory (CAM)

By restricting the cache location where a data item can be stored, we can simply the cahce

In a direct-mapped cache, a data item can be stored in one location only, determined by its address (we use some of the address bits as an index to the cache array).

Address Mapping for Direct-Mapped Cache



Example Problem

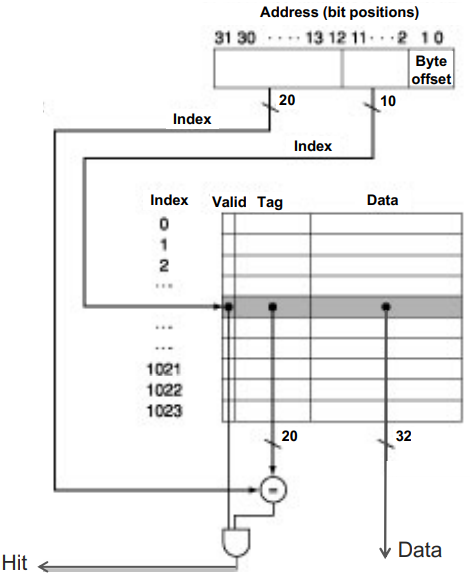
Given a 4KB direct-mapped cache with 4-byte blocks and 32-bit addresses, how many tag, index and offset bits does the address decompose into?

If we have a 4 KB cache and use 4 bytes per block, then we have space for 1K blocks, this will require a 10-bit index

A 4 byte block will require a 2-bit offset (to index each byte)

This leaves the tag with 32-10-2 = 20 bits.

Direct-Mapped Cache in Detail

In this method, we use the 10 bit index to find the correct ‘row’ in the cache, but just because it matches, doesn’t mean it’s the correct data, so we then compare the tag stored in the cache with the tag from the address and AND the result with the valid bit, this will tell use if we get the right data and its available (is a hit), if it’s a miss, then we go and we get the block from main memory.

Note that this is using the example problem.