

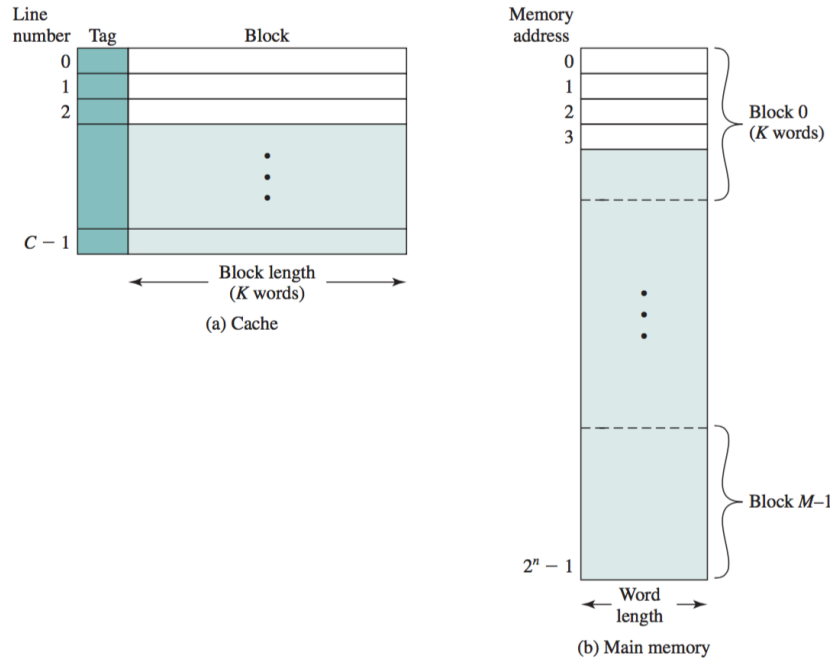
Memory (II)

Cache Overview

- Cache is a small amount of fast memory.
 - If the CPU requests contents, it will check cache for this data.
 - If the data present in the cache, the CPU get from the cache.
 - Otherwise, the required block will be read from main memory to cache, and the contents are delivered to the CPU.
 - Cache is to help improve the average access time, which is achieved by improving the hit ratio, letting requested data presented in the fast memory.
 - However, it is not the more the better.
 - Cost: more caches mean more expensive
 - Speed:
 - More cache is faster up to a point.
 - Beyond that point, larger circuits will be more complex and slow up the cache.
 - An algorithm is needed for mapping main memory blocks to lines in the cache. Beyond that point, the time it takes to look into the cache is not significantly less than to look into the RAM directly.
 - So the right amount of cache and appropriate algorithm lead to the good performance.
-

Cache Structure

- Main memory consists of 2^n addressable **words**, each word has a unique n -bit address.
- Main memory contains M **blocks**, Each block has K words. So, there are $M = \frac{2^n}{K}$ blocks in main memory.
- Cache contains C **lines** of blocks, each has a tag uniquely identifying the block of K words.
- $C \ll M$
- If a word in a block of memory is read, that block is transferred to one of the lines of the cache



Mapping Functions

- A mapping function is the method used to locate a word.
- To locate a word, the address associated with the word is used.
- Mapping functions define the address structure and how to use that structure to locate a word.
- It is used when copying a block from main memory to the cache and it is used again when trying to retrieve data from the cache.

Direct Mapping

- A straightforward way is to map each block of main memory to only one cache line. It says if a block is in a cache, it will always be found in the same place once it's been swapped into the cache.

Cache line	Main memory blocks assigned
0	$0, m, 2m, \dots, 2^s - m$
1	$1, m + 1, 2m + 1, \dots, 2^s - m + 1$
$\vdots$	$\vdots$
$m - 1$	$m - 1, 2m - 1, 3m - 1, \dots, 2^s - 1$

- Line number is calculated using the following function

$$i = j \bmod m$$

where

i = cache line number

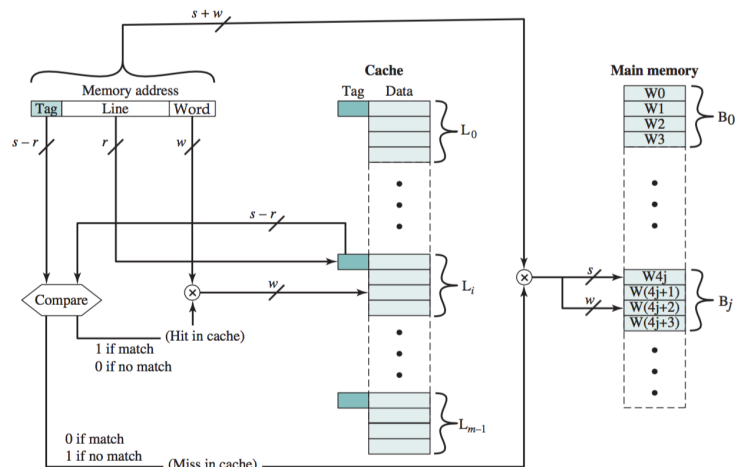
j = main memory block number

m = number of lines in the cache

- Address structure for direct mapping is



- Each main memory address can be divided into three fields
 - Rightmost w bits identify a unique word within a block
 - Remaining bits specify which block in memory. These bits are divided into two fields:
 - Rightmost r bits of these s bits identifies which line in the cache
 - Leftmost $s-r$ bits uniquely identifies the block within a line of cache, as a line can be used by multiple blocks.
 - Please note that this tag field is the tag uniquely identify a block, as no two blocks in the same line have the the same tag.
- The way Direct Mapping uses to check contents of cache is:
 - use the “line” field of the target address to find the particular line in cache
 - compare the “tag” field of the address with the “tag” of that particular cache line
 - if the two “tags” are the same, use the “word” field to find the target word
 - otherwise, the target word is missed in cache, and will need to use the target address to search in the main memory and replace a block in the cache with the block where the target word is in.



Example

- Assume the cache size is 64 KB. Data are transferred between main memory and the cache in blocks of 4 bytes each. This means that the cache is organized as $16K = 2^{14}$ lines of 4 bytes each. A word is 8-bit long. [Cache size = 64 KB, Block size = 4 B, number of lines 2^{14} , Word length = 1B]
- The main memory consists of 16 MB, with each byte directly addressable by a 24-bit address ($2^{24} = 16 \text{ M}$). Thus, for mapping purposes, we can consider main memory to consist of 4 M blocks of 4 bytes each. [Main memory size = 16 MB, address = 24 bits, numbers of blocks = 4M]
- How many bits for w? [2, as the block size is 4 bytes and each word is a byte, so need 2 bit to specify the 4 words.]
- How many bits for r? [14, as the cache has 2^{14} number of lines]
- How many bits for tag? [8, as the address is 24-bit long, $24 - 2 - 14 = 8$]
- So, if using direct mapping, the above **example** would have the address in the following format.

Tag: 8 bits	Line: 14 bits	Word: 2 bits
-------------	---------------	--------------

- Summary
 - Address length = (s + w) bits
 - Number of addressable units = $2^{(s+w)}$ words or bytes
 - Block size = line width = 2^w words or bytes
 - Number of blocks in main memory = $\frac{2^{(s+w)}}{2^w} = 2^s$
 - Number of lines in cache = 2^r
 - Size of tag = (s - r) bits