Memory (III)

Mapping Functions (II)

Direct Mapping (cont.)

- Address structure for direct mapping is

<table>
<thead>
<tr>
<th>Tag</th>
<th>Line: Bits identifying line in cache</th>
<th>Word: Bits identifying word offset into block</th>
</tr>
</thead>
<tbody>
<tr>
<td>s - r bits</td>
<td>r bits</td>
<td>w bits</td>
</tr>
</tbody>
</table>

- 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 ($2^{24} = 16$ 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]

- 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 |

- 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, $24 - 2 - 14 = 8$]

- 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

- Pros & Cons
  - Simple
  - Inexpensive
  - Fixed location for given block - if a program accesses 2 blocks that map to the same line repeatedly, cache misses are very high.

**Associative Mapping**

- To provide more flexibility, associative mapping allows a main memory block to be loaded into any line of cache.
- Its address structure has two fields:

\[
\begin{array}{|c|c|}
\hline
\text{Tag: } s \text{ bits} & \text{Word: } w \text{ bits} \\
\hline
\end{array}
\]

- The rightmost \( w \) bits are the word position within a block.
- The leftmost \( s \) bits are used to identify which block is stored in a particular cache line.

- The way to check for hit is
  - compare the “tag” field of the target address with the “tag” of every line of the cache.
  - if a cache line has the same “tag”, use the “word” field of the target address to find the target word.
  - otherwise, the target word is missed in the 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.