Memory (III)

Mapping Functions (II)

Direct Mapping (cont.)

- 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:

| Tag: \(s\) bits | Word: \(w\) bits |

  - 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.

- If we use the same example we used in Direct Mapping, what does the address structure look like if using associative mapping?
  • Cache size: 64 KB; block size: 4 bytes; addressable unit: byte
  • Main memory size: 16 MB; address length: 24 bits

    - Num. of lines: \( \frac{64KB}{4B} = 16K = 2^{14} \)
    - Num. of blocks: \( \frac{16MB}{4B} = 4M \)

  • 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 the tag? [22, as 24 - 2 = 22]

  | Tag: 22 bits | Word: 2 bits |

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

- Pros & Cons
  • A block can load to any line of cache
  • Every line’s tag must be examined for a match
  • Cache searching gets expensive and slow

Set-Associative Mapping
- A comprise that exhibits the strengths of direct mapping (simple, inexpensive) and associative mapping (flexibility that blocks can be loaded to any lines) while reducing their disadvantages (fixed location for a given block - high cache miss ratio, examine every line’s tag for a match - cache searching is expensive and slow).

- It introduces a new concept cache set.
  • Cache is divided into a number of sets, $v$
  • Each set contains $k$ lines
  • $k$ lines in a set is called a $k$-way set associative mapping
  • Number of lines in a cache, $m = v \times k$
  • The idea of set-associative mapping is that a block is always mapped to a specific cache set if it's swapped into the cache. But, it can be loaded into any line of that cache set.

• Way to calculate the cache set number of a block $i = j \mod v$
  where
  $i = \text{cache set number}$
  $j = \text{main memory block number}$
  $m = \text{number of lines in the cache}$
  $v = \text{number of sets}$
  $k = \text{number of lines in each set}$

• Note:
  - $k = 1$, this is direct mapping
- $v = 1$, this is associative mapping
- A given block maps to a line within its specified set

- Its address structure has three fields

<table>
<thead>
<tr>
<th>Tag: $s - r$ bits</th>
<th>Set: $r$ bits</th>
<th>Word: $w$ bits</th>
</tr>
</thead>
</table>

- The rightmost $w$ bits uniquely identify a word within a block
- The rightmost $r$ bits of the remaining $s$ bits identify which set in the cache
- The leftmost $s - r$ bits uniquely identify the block within a set

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