Set-Associative Mapping

- Cache contains of $2^n$ sets of $m$ lines
- select set from $n$ dedicated bits in address by direct mapping
- within the $m$ lines of this set, perform an associative mapping

Example: 4 sets of 3 lines à 8 words (16 bit)

<table>
<thead>
<tr>
<th>tag</th>
<th>set</th>
<th>word</th>
<th>×</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>4</td>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
</tbody>
</table>

Example (cont’d)

Accessing: 312E, 4B78, 233C, 12C0

Set-Associative Mapping

Advantages

- set of lines immediately available
- limited search for a cache line only
- a line is only replaced if set is full
- provides usually good performance
- overall, good compromise!

Disadvantages

- complex mapping function
- sets may remain unused
- cache line still has to be searched
- tag field is slightly larger as compared to direct mapping
- additional counter for LRU
- worst case: difficult to specify

Example for On-Chip Cache

MC 68040:

- separate caches for instruction and data
- size per cache: 4 KB
- four-way set-associative
- 64 sets à 4 lines
- each line contains 4 words à 32 bits
- $64 \times 4 \times (4 \times 32) = 4096$ √
- each line has tag and valid fields
- each word has its own dirty bit
- write-back and write-through
- flushing of selected lines or entire cache
- random replacement scheme
Example for On-Chip Cache (2)

ARM720T:
- unified cache for instruction and data
- size of cache: 8 KB
- four-way set-associative
- 128 sets à 4 lines
- each line contains 4 words à 32 bits
- $128 \times 4 \times (4 \times 32/8) = 8192$
- reads can be cachable or uncachable
- writes are bufferable or unbufferable
- write-through only to buffer
- random replacement scheme

Example for On-Chip Cache (3)

Pentium III and 4:
- separate L1 caches for instruction and data
- size per L1 cache: 16 KB
- data: four-way set-associative
- instr: two-way set-associative
- unified L2 cache up to 512 KB
- on-chip or off-chip (earlier)
- fast 64-to-256 bit wide cache bus
- P4 caches instructions pre-interpreted
- write-through for data in L1 cache
- optional L3 cache for servers

State of the Art
- Fast primary on-chip and secondary off-chip caches (access optimization)
- instruction and data caches separated in primary cache (pipelining)
- usually set-associative caching
- write-through (with optional write-buffer) or write-back selectable
- read-through for optimization on misses
- fine-tuning of cache parameters and line replacement scheme crucial
- close connection between memory cache and memory management unit (virtual memory)

Cache Performance
- Hit rate: Probability of cache hit
  \[ h = \frac{\text{average cache hits}}{\text{total accesses}} \]
- Average access time:
  \[ t_{\text{ave}} = hC + (1-h)M \]
- Performance improvement:
  \[ \text{time without cache}/\text{time with cache} \]

Two-level Cache
- L1 hit rate $h_1$: in primary cache
- L2 hit rate $h_2$: in secondary cache
- Three probabilities:
  1. in L1: $h_1$
  2. in L2: $(1-h_1) \cdot h_2$
  3. in M: $(1-h_1) \cdot (1-h_2)$
- Average access time:
  \[ t_{\text{ave}} = h_1 C_1 + (1-h_1) h_2 C_2 + (1-h_1) (1-h_2) M \]