Decreasing miss ratio with k-way associativity

Equivalent to having k direct mapped caches, with the option of putting the data in any of the k caches

Example (2-way set associative, Block size = 1)

A cache set is defined as the corresponding blocks from the k caches. In the above example, we have 4 sets, each being 2-way associative.
Degree of associativity can vary from 1-way to full

Example: a cache with 8 data blocks

- One-way set associative (direct mapped)
- Two-way set associative
- Four-way set associative
- Eight-way set associative (fully associative)

Need a block replacement policy to determine which block to evict in case a new block is to be cached into a full set.

- LRU replacement → evict the least recently used block

Example: 4KB cache, 4-way associative, block size = 1 word

- Equivalent to four, 1KB, direct mapped caches.
- Each 1KB direct mapped cache has 256 = 2^8 blocks.
- Byte offset = 2 bits
- Block offset = 0 bits
- Index = 8 bits
- Tag = remaining 22 bits

- Need to compare tags in all four ways
- If one matches, return the corresponding stored data,
- May use block size larger than one word.
Example: 4KB cache, 2-way associative, block size = 2 words

Equivalent to two, 2KB, direct mapped caches.

Each 2KB direct mapped cache has $256 = 2^8$ blocks.
(Each block = $2^1 = 2$ word)

$\rightarrow$

Byte offset = 2 bits
Block offset = 1 bits
Index = 8 bits
Tag = remaining 21 bits

Performance analysis

Cache access time = 1 cycle
Memory access time = 100 cycles
Average memory access time

$= 1 + \text{miss rate} \times \text{miss penalty}$

$= 1 + 0.05 \times 100 = 6$ cycles

Effect of cache misses on pipelined execution: Pipeline will stall on a miss to the Instruction or Data caches – thus increasing the CPI.

Example: Assume a pipeline where 36% of the instructions are lw or sw
Assume the instruction cache miss rate = 2% and data cache miss rate = 4%

Effective miss rate = $0.02 + 0.36 \times 0.04 = 0.0344$ misses per instruction

$\text{CPI} = \text{CPI ignoring cache misses} + 0.0344 \times 100 = \text{CPI no misses} + 3.376$ cycles
Effect of block size on miss rate

- Because of access locality, larger blocks decreases the miss rate.
- For random accesses, larger blocks increases miss rate.
- Memory accesses are a mix of sequential and non-sequential access,
- Hence, larger blocks, up to a given optimum size, decreases the miss rate.

Effect of cache size on miss rate and hit time

- If miss rate = 10%, average memory access time = 1 + 0.1 * 100 = 11 cycles.
- Larger cache leads to lower miss rate but larger hit time.
- If miss rate = 8% and hit time = 4 cycles, then average memory access time = 4 + 0.08 * 100 = 12 cycles.
Improving performance with multilevel caches

Example (a 1GHz machine \(\Rightarrow\) 1ns cycles)
- L1 hit time = 1 cycle,
- L2 hit time = 4 cycles
- DRAM access time = 100ns (100 cycles)
- L1 miss rate = 10%
- 70% of L1 misses are hits in L2 (hence 30% are misses)

- Average memory access time = \(1 + 0.1 \times (4 + 0.3 \times 100)\) = 4.4 cycles.

Note that all L1 misses are looked up in L2 and only the L2 misses are sent to memory.

Multilevel On-Chip Caches for Cortex-A8 and Core-i7

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>ARM Cortex-A8</th>
<th>Intel Nehalem</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 cache organization</td>
<td>Split instruction and data caches</td>
<td>Split instruction and data caches</td>
</tr>
<tr>
<td>L1 cache size</td>
<td>32 KB each for instructions/data per core</td>
<td>32 KB each for instructions/data per core</td>
</tr>
<tr>
<td>L1 cache associativity</td>
<td>4-way (L), 4-way (D) set associative</td>
<td>4-way (L), 4-way (D) set associative</td>
</tr>
<tr>
<td>L1 replacement</td>
<td>Random</td>
<td>Approximated LRU</td>
</tr>
<tr>
<td>L1 block size</td>
<td>64 bytes</td>
<td>64 bytes</td>
</tr>
<tr>
<td>L1 write policy</td>
<td>Write-back, Write-allocate (?)</td>
<td>Write-back, No-write-allocate</td>
</tr>
<tr>
<td>L1 hit time (boxed out)</td>
<td>1 clock cycle</td>
<td>4 clock cycles, pipelined</td>
</tr>
<tr>
<td>L2 cache organization</td>
<td>Unified (instruction and data)</td>
<td>Unified (instruction and data) per core</td>
</tr>
<tr>
<td>L2 cache size</td>
<td>128 KB to 1 MB</td>
<td>256 KB (0.25 MB)</td>
</tr>
<tr>
<td>L2 cache associativity</td>
<td>8-way set associative</td>
<td>8-way set associative</td>
</tr>
<tr>
<td>L2 replacement</td>
<td>Random (?)</td>
<td>Approximated LRU</td>
</tr>
<tr>
<td>L2 block size</td>
<td>64 bytes</td>
<td>64 bytes</td>
</tr>
<tr>
<td>L2 write policy</td>
<td>Write-back, Write-allocate (?)</td>
<td>Write-back, Write-allocate</td>
</tr>
<tr>
<td>L2 hit time</td>
<td>11 clock cycles</td>
<td>10 clock cycles</td>
</tr>
<tr>
<td>L3 cache organization</td>
<td>-</td>
<td>Unified (instruction and data)</td>
</tr>
<tr>
<td>L3 cache size</td>
<td>-</td>
<td>8 MB, shared</td>
</tr>
<tr>
<td>L3 cache associativity</td>
<td>-</td>
<td>16-way set associative</td>
</tr>
<tr>
<td>L3 replacement</td>
<td>-</td>
<td>Approximated LRU</td>
</tr>
<tr>
<td>L3 block size</td>
<td>-</td>
<td>64 bytes</td>
</tr>
<tr>
<td>L3 write policy</td>
<td>-</td>
<td>Write-back, Write-allocate</td>
</tr>
<tr>
<td>L3 hit time</td>
<td>-</td>
<td>35 clock cycles</td>
</tr>
</tbody>
</table>
The 3C cache miss model

Cache misses can be classified into one of the following:

1) Compulsory misses
   - Result from referencing a block for the first time
   - Unavoidable

2) Capacity misses
   - Result from the limited cache size (assuming a fully associative cache)
   - May be avoided if a larger cache is used

3) Conflict misses
   - Result from hash collisions if cache is not fully associative

Remember to send any question that you had while listening to this presentation to me before class time. I will answer the question that I receive in class.