CS 61C Spring 2016 Discussion 7 – Caches

In the following diagram, each blank box in the CPU Cache represents 8 bits (1 byte) of data. Our memory is byte-addressed, meaning that there is one address for each byte. Compare this to word-addressed, which means that there is one address for each word.

<table>
<thead>
<tr>
<th>CPU Cache</th>
<th>Index Number</th>
<th>Offset</th>
<th>Tag bits</th>
<th>Index bits</th>
<th>Offset bits</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3 2 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>32</td>
</tr>
</tbody>
</table>

Index bits = \( \log_2 \) (Number of index rows) Offset bits = \( \log_2 \) (Number of offsets columns)

1. **Direct mapped caches**
   1. How many bytes of data can our cache hold? How many words?
   2. Fill in the “Tag bits, Index bits, Offset bits” with the correct T:I:O breakdown according to the diagram.
   3. Let’s say we have a 8192KiB cache with an 128B block size, what is the tag, index, and offset of 0xFEEDF00D?

<table>
<thead>
<tr>
<th>FE</th>
<th>ED</th>
<th>F0</th>
<th>OD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>1110</td>
<td>1110</td>
<td>0000</td>
</tr>
<tr>
<td>0000</td>
<td>1111</td>
<td>1101</td>
<td></td>
</tr>
</tbody>
</table>

Tag: Index: Offset:

4. Fill in the table below. Assume we have a write-through cache, so the number of bits per row includes only the cache data, the tag, and the valid bit.

<table>
<thead>
<tr>
<th>Address size (bits)</th>
<th>Cache size</th>
<th>Block size</th>
<th>Tag bits</th>
<th>Index bits</th>
<th>Offset bits</th>
<th>Bits per row</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>4KiB</td>
<td>4B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>32KiB</td>
<td>16B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
<td>16</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>2048KiB</td>
<td></td>
<td></td>
<td></td>
<td>14</td>
<td>1068</td>
</tr>
</tbody>
</table>

2. **Cache hits and misses**

Assume we have the following byte-addressed cache. Of the 32 bits in each address, which bits do we use to find the row of the cache to use?

Classify each of the following byte memory accesses as a cache hit (H), cache miss (M), or cache miss with replacement (R).
3. 3C’s of Caches

3 types of cache misses:

1. Compulsory: Miss to an address not seen before. Reduce compulsory misses by having a longer cache line, which brings in locations before we ask for them.
2. Conflict: Increasing the associativity or improving the replacement policy would remove the miss.
3. Capacity: The only way to remove the miss is to increase the cache capacity.

Classify each M and R above as one of the 3 misses above.

4. Analyzing C Code

```c
#define NUM_INTS 8192
int A[NUM_INTS]; /* A lives at 0x10000 */
int i, total = 0;
for (i = 0; i < NUM_INTS; i += 128) { A[i] = i; } /* Line 1 */
for (i = 0; i < NUM_INTS; i += 128) { total += A[i]; } /* Line 2 */
```

Let’s say you have a byte-addressed computer with a total memory of 1MiB. It features a 16KiB CPU cache with 1KiB blocks.

1. How many bits make up a memory address on this computer?
2. What is the T:I:O breakdown? tag bits: index bits: offset bits:
3. Calculate the cache hit rate for the line marked Line 1:
4. Calculate the cache hit rate for the line marked Line 2:

5. Average Memory Access Time

AMAT is the average (expected) time it takes for memory access. It can be calculated using the formula:

\[ \text{AMAT} = \text{hit}\_\text{time} + \text{miss}\_\text{rate} \times \text{miss}\_\text{penalty} \]

Remember that the miss penalty is the additional time it takes for memory access in the event of a cache miss. Therefore, a cache miss takes (hit_time + miss_penalty) time.

1. Suppose that you have a cache system with the following properties. What is the AMAT?
   a) L1$ hits in 1 cycle (local miss rate 25%)
   b) L2$ hits in 10 cycles (local miss rate 40%)
   c) L3$ hits in 50 cycles (global miss rate 6%)
   d) Main memory hits in 100 cycles (always hits)