Caches

Why do we cache? What is the end result of our caching, in terms of capability?

What are temporal and spatial locality? Give high level examples in software of when these occur.

TIO breakdown of an address:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Index</th>
<th>Offset</th>
</tr>
</thead>
</table>

Offset: “column index” (O bits) – location of address within block
Index: “row index” (I bits) – location (row) of block within cache
Tag: “block identifier” (T bits) – is this the right block?

Cache Misses – aka The Three alliterated C’s

Compulsory – Misses that are the result of the address having never been seen before.
Capacity – Misses that result from the cache being full. Only for associative!
Conflict – Misses that result from addresses “aliasing” to the same row in the cache. These are misses that occur not because the cache is full, but because the addresses map to the same row in the cache. Can’t occur on fully-associative!

Define the following cache terms:

Cache hit –
Cache miss –
Cache miss, block replacement –

Cache hit rate –
Cache miss rate –

Assume 16 B of memory and an 8B direct-mapped cache with 2-byte blocks. Classify each of the following memory accesses as hit (H), miss (M), or miss with replacement (R).

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
a. 4 | e. 1 |
b. 5 | f. 10 |
c. 2 | g. 7 |
d. 6 | h. 2 |
Fill this one in... everything here is direct-mapped!

<table>
<thead>
<tr>
<th>Address 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>4KB</td>
<td>4B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>16KB</td>
<td>8B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>64KB</td>
<td>16</td>
<td>12</td>
<td>4</td>
<td>146</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>512KB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>64</td>
<td>64B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>2048KB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1069</td>
</tr>
</tbody>
</table>

You know you have 1 MiB of memory (maxed out for processor address size) and a 16 KiB direct-mapped cache (data size only, not counting extra bits) with 1 KiB blocks.

```c
#define NUM_INTS 8192
int *A = malloc(NUM_INTS * sizeof(int)); // returns address 0x100000
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
```

a) What is the T:I:O breakup for the cache (assuming byte addressing)?

b) Calculate the hit percentage for the cache for the line marked “Line 1”.

c) Calculate the hit percentage for the cache for the line marked “Line 2”.

d) How could you optimize the computation?

**Average Memory Access Time (AMAT) = Hit time + Miss Rate * Miss Penalty**

Questions (Variable modification with all other variables unchanged):

1) How will increasing the cache size change hit time?

2) How will increasing the cache size change the miss rate (be careful!)?

3) What variable(s) will adding a secondary (L2) cache modify?

4) How will increasing block size change miss penalty? Miss rate?

5) Math. 😊 You want your AMAT to be <= 2 cycles. You have two levels of cache.
   - L1 Hit Time is 1 cycle
   - L1 miss rate is 10%
   - L2 Hit Time is 3 cycles
   - L2 Miss Penalty is 100 cycles
   What must you optimize your L2 miss rate to be?