CS61C Discussion 8 – Caches

1 Tag Index Offset

When working with caches, we have to be able to break down the memory addresses we work with to understand where they fit into our caches. There are three fields:

Tag - Used to distinguish different blocks that use the same index - Number of bits: leftovers
Index - The set that this piece of memory will be placed in - Number of bits: $\log_2$(# of indices)
Offset - The location of the byte in the block - Number of bits: $\log_2$(size of block)

In the following diagrams, each blank box represents 1 byte (8 bits) of data. All of memory is byte addressed.

2 Direct mapped Cache

1. Let's say we have a 8192KiB cache with an 128B block size, how many bits are in tag, index, and offset? What parts of the address of 0xFEEDF00D fit into which sections?

	Tag	Index	Offset
Number of bits	9	16	7
Bits of address	111111101 (0x1FD)	1101101111100000 (0xDBE0)	0001101 (0x0D)

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

Address size (bits)	Cache Size	Block Size	Tag Bits	Index Bits	Offset Bits	Bits per row
16	4KiB	4B	4	10	2	32 + 4 + 1 = 37
32	32KiB	16B	17	11	4	128 + 17 + 1 = 146
32	64KiB	16B	16	12	4	128 + 16 + 1 = 145
64	2048KiB	128B	43	14	7	1068

3 Cache Hits and Misses

3.1 Finding Hits and Misses

Assume we have a direct-mapped byte-addressed cache with capacity 32B and block size of 8B. Of the 32 bits in each address, which bits do we use to find the index of the cache to use? We use the 4^{th} and 5^{th} least significant bit since the offset is 3 bits
Classify each of the following byte memory accesses as a cache hit (H), cache miss (M), or cache miss with replacement(R). It is probably best to try drawing out the cache before going through so that you can have an easier time seeing the replacements in the cache. The following white space is to do this:

1. 0x00000004 Index 0, Tag 0: M, Compulsory
2. 0x00000005 Index 0, Tag 0: H
3. 0x00000068 Index 1, Tag 3: M, Compulsory
4. 0x000000C8 Index 1, Tag 6: R, Compulsory
5. 0x00000068 Index 1, Tag 3: R, Conflict
6. 0x000000DD Index 3, Tag 6: M, Compulsory
7. 0x00000045 Index 0, Tag 2: R, Compulsory
8. 0x00000004 Index 0, Tag 0: R, Capacity
9. 0x000000C8 Index 1, Tag 6: R, Capacity

3.2 The 3 C's of Misses

3 types of cache misses:

I. Compulsory: First time you ask the cache for a certain block. A miss that must occur when you first bring in a block. Reduce compulsory misses by having a longer cache lines (bigger blocks), which bring in the surrounding addresses along with our requested data. Can also pre-fetch blocks beforehand using a hardware prefetcher (a special circuit that tries to guess the next few blocks that you will want).

II. Conflict: Occurs if you hypothetically went through the ENTIRE string of accesses with a fully associative cache and wouldn't have missed for that specific access. Increasing the associativity or improving the replacement policy would remove the miss.

III. Capacity: The only way to remove the miss is to increase the cache capacity, as even with a fully associative cache, we had to kick a block out at some point.

Note: There are many different ways of fixing misses. The name of the miss doesn't necessarily tell us the best way to reduce the number of misses.

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

4 N-Way Set Associative Caches

1. Assuming 32 bits of physical memory, for an 8-way set associative 4KiB cache with 16B blocks, how big are the T, I, and O fields? T = 23, I = 5, O = 4
2. How many total bits of storage are required for the cache if it uses write back and LRU replacement? 2^8 rows, each row has Tag, Data, Valid, LRU Type of Dirty ($\log_2$(Associativity)) = 23 + 128 + 1 + 1 + 3 = 156 bits. $156(2^8) = 39936$ bits.