Jon Discussion 8

Sunday, July 15, 2018 9:57 PM

CS61C Discussion 8 – Caches

1 An Overview

At first, caches seem like a confusing mixture of concepts: addresses, memory, bits, data, locality, etc. Truthfully, caches are just beautiful data structures. Here, we will break down all the key terms to keep in mind:

Consider the example given in lecture: books. In this situation, we are looking for a certain book, but the trip of going all the way from your house to the shelf in the library is tedious and becomes redundant. So what do we do?

1. We will consider the library to be **main memory** and our house as the **cache**. When we go to grab the book we want, we grab it plus several books adjacent to it and put it into a box. In the language of caches, a book would be a **byte** and the box is known as a **block**. Grabbing several books along with the one we originally wanted exploits **spatial locality**.
2. We arrive home with our box of books, but now need to put it somewhere. In our house, we have many different rooms to choose from. Each of these rooms is known as a **set**, capable of holding several **blocks**. The number of boxes we can fit in each room is analagous to the number of **ways** a **block** can be put into a **set**.
3. Now when we need a book, we first start by finding the room it's in, or in cache terms, the **set**. These are what the **index bits** are for. Then, we need to find which box contains our book; this is what the **tag bits** are for. Lastly, we need to know where the book is in the box; the **offset bits** tell us this information.

Check your understanding:

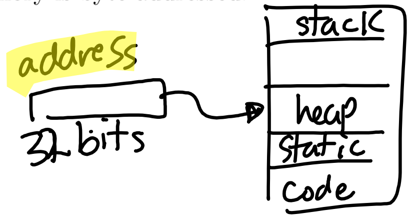
1. What does a direct mapped cache mean in terms of boxes, rooms, and books? A fully associative cache? An N-way set associative cache?
2. What is the relationship between ways, sets, and blocks?

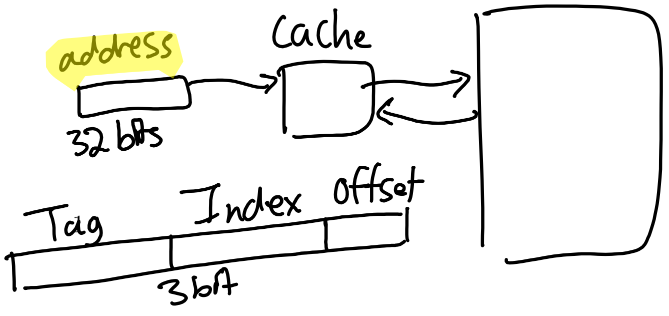
2 Tag Index Offset

The cache holds data fetched from main memory, but how does it decide where those bytes should be stored? (In the language of the analogy above, how do we know where a book is in our house?) We use portions of the memory address it was loaded from, breaking it down into three groups of bits: Tag, Index, and Offset.

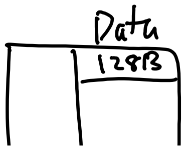
Tag - Used to distinguish different blocks that use the same index - Number of bits: leftovers
Index - The set that this piece of memory will be placed in - Number of bits: $\log_2$(# of sets)
Offset - The location of the byte in the block - Number of bits: $\log_2$(size of block)

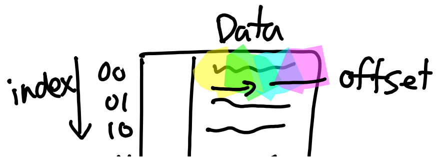
Therefore, given the details of main memory and our cache, we are able to devise a scheme for locating individual bytes within it. In the following diagrams, each blank box represents 1 byte (8 bits) of data. All of memory is byte addressed.





$8192 = 2^{13}$





3 Direct mapped Cache

3 Direct Mapped Cache

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

$\log_2(128) = 7$

	Tag	Index	Offset
Number of bits	9	16	7
Bits of address	1111 1110	1110 1101 1111 0000	0000 1101

$32 - 16 - 7 = 9$ $\quad 2^{13}\cdot 2^{10} \to 2^{23}/2^{7} = 2^{16}$ $\quad \log 2^{16} = 16$ bits

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. Recall that in addition to the data itself, our cache can store other information such as whether an entry has a valid piece of data.

Address size (bits)	Cache Size	Block Size	Tag Bits	Index Bits	Offset Bits	Bits per row
16	4KiB	4B	4	10	2	
32	32KiB	16B	17	11	4	
32	2^{16}B	16B	16	12	4	
64	2048KiB	2^7	43	14	7	1068

$\frac{C}{16} = 2^{12}$ $\quad$ Cache $= 2^{12}\cdot 2^{4} = 2^{16}$

$2^{21}/2^{14} = 2^{7}$

4 Cache Hits and Misses

4.1 Finding Hits and Misses

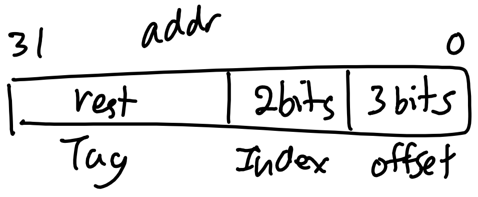
Assume we have a direct-mapped byte-addressed cache with capacity 32B and block size of 8B. Of the 32 bits in each address, which bits do we use to find the index of the cache to use?

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 0000 0100 M, comp
2. 0x00000005 0000 0101 H
3. 0x00000068 0110 1000 M, comp
4. 0x000000C8 1100 1000 M, comp
5. 0x00000068 0110 1000 M, conflict
6. 0x000000DD 1101 1101 M, comp
7. 0x00000045 0100 0101 M, comp
8. 0x00000004 0000 0100 M, Cap
9. 0x000000C8 1100 1000 M, cap

Cache

index	Tag	Data	valid
00	0		1
01	6		1
10			0
11	6		1



4.2 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.

5 N-Way Set Associative Caches

$2^2 \cdot 2^{10}$

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?

Offset : $\log_2$ (Block) = $\log_2(16) = 4$

Index : 5 bits

Tag 23 bits

2. How many total bits of storage are required for the cache if it uses write back and LRU replacement?

$2^{12}/2^4 = 2^8$

$2^8/8 = 2^8/2^3$
$= 2^5$

$32 - 5 - 4 = 23$

index Data

Index	Data
00	
01	
10	
11	

index	data
00	
01	
10	
11	