CS61C Summer 2016 Discussion 8 – 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.

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

CPU Cache	Offset			
	3	2	1	0

Tag bits	Index bits	Offset bits	Total
			32

What type of cache is this?

CPU Cache	Index Number	Offset			
		3	2	1	0
	0				
	1				

Tag bits	Index bits	Offset bits	Total
			32

CPU Cache	Index Number	Offset			
		3	2	1	0
	0				
	1				

What type of cache is this?

CPU Cache	Index Number	Offset			
		3	2	1	0
	0				
	1				
	2				
	3				

Tag bits	Index bits	Offset bits	Total
			32

What type of cache is this?

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?

FE	ED	F0	0D
1111 1110	1110 1101	1111 0000	0000 1101

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.

Address size (bits)	Cache size	Block size	Tag bits	Index bits	Offset bits	Bits per row
16	4KiB	4B				
32	32KiB	16B				
32			16	12		
64	2048KiB			14		1068

2. Cache hits and misses

Assume we have the following 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).

	Index Number	Offset							
		7	6	5	4	3	2	1	0
CPU Cache	0								
	1								
	2								
	3								

1. 0x00000004
2. 0x00000005
3. 0x00000068
4. 0x000000C8
5. 0x00000068
6. 0x000000DD
7. 0x00000045
8. 0x00000004
9. 0x000000C8

3. Analyzing C Code

```
#define NUM_INTS 8192
int A[NUM_INTS];  /* A  lives  at  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 */
```

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:

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?

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

3. Associativity usually improves the miss ratio, but not always. Give a short series of address references for which a two-way set-associative cache with LRU replacement would experience more misses than a direct-mapped cache of the same size.