CS 61C Spring 2015 Discussion 5 – Direct Mapped Caches

In the following diagram, each block 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.

The Geometry of Caches

Main Memory
...
6
5
4
3
2
1
0

Cache Number	Main Memory			
0	3	2	1	0
	7	6	5	4
1	11	10	9	8
	15	14	13	12
2	19	18	17	16
	23	22	21	20
3	27	26	25	24
	31	30	29	28

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

Tag bits	Index bits	Offset bits	**Total**
29	1	2	**32**

$1\,\text{word} = 4\,\text{bytes} = 32\,\text{bits}$

$\text{Index bits} = \log_2(\text{Number of index rows})$

$\text{Offset bits} = \log_2(\text{Number of offsets columns})$

1. Direct mapped caches

1. How many bytes of data can our cache hold? 8 bytes How many words? 2 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: 1 1111 1101 (0x1FD) Index: 1101 1011 1110 0000 (0xDBE0) Offset: 000 1101 (0x0D)
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	4	10	2	37
32	32KiB	16B	17	11	4	146
32	64KiB	16B	16	12	4	145
64	2048KiB	128B	43	14	7	1068

2. Cache hits and misses

Assume we have the following cache. 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 M
2. 0x00000005 H
3. 0x00000068 M
4. 0x000000C8 R
5. 0x000000DD M
6. 0x00000045 R
7. 0x00000004 R
8. 0x000000C8 H

Self check: Of the 32 bits in each address, which bits do we use to find the *row* of the cache to use?

We use these bits: 0b…0000 0000 0100

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 address space of 1MiB. It features a 16KiB CPU cache with 1KiB blocks.

1. How many bits make up a memory address on this computer? 20
2. What is the T:I:O breakdown? 6 tag bits 4 index bits 10 offset bits
3. Calculate the cache hit rate for the line marked `Line 1`:
   50%. The integers are 4×128 = 512 bytes apart, which means that there are two accesses per block. The first access is a cache miss, but the second access is a cache hit, because A[i] and A[i + 128] are in the same cache block.
4. Calculate the cache hit rate for the line marked `Line 2`:
   50%. At the end of line 1, we now have the second half of `A` inside our cache, so we get the same hit rate as before. Note that we do not have to consider cache hits for `total`, since the compiler will probably leave it in a register.

4. Average Memory Access Time

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

$$\text{AMAT} = \text{hit time} + \text{miss rate} \times \text{miss penalty}$$

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 $\text{hit time} + \text{miss penalty}$ 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)

   The AMAT is 1 + 25%×(10 + 40%×(50)) + 6%×(100) = 14.5 cycles.

   Alternatively, you could have calculated the following global hit rates for each of the caches:
   - 75% L1$
   - 15% L2$
   - 4% L3$
   - 6% Main Memory

   And the following hit times:
   - 1 cycle L1$
   - 11 cycles L2$
   - 61 cycles L3$
   - 161 cycles Main Memory

   Then compute:

   AMAT = 75%×1 + 15%×11 + 4%×61 + 6%×161 = 14.5 cycles.