

Comp 411 Computer Organization

Fall 2007

Problem Set #10 Solutions

Problem 1

a) There are 16 lines of 4 words each, so 64 words can be present in the cache at one time.

There are 16 lines, so the bits needed to select the cache line must be $\log_2 16 = 4$.

The remaining 24 upper bits are stored in the tag field.

b) The V bit indicates whether the line is valid.

c) First, partition 0x3328C into the cache components:

tag	line	block	
0000 0000 0000 0011 0011 0010	1000	11	00

Thus, the word would be in line 8, block 3 (DATA -11). The tag field will contain 0x000332.

d) Locations 0x12368 and 0x322F68 appear as follows in binary:

value	tag	index	block	
0x12368	0000 0000 0000 0001 0010 0011	0110	10	00
0x322F68	0000 0000 0011 0010 0010 1111	0110	10	00

Since the locations would both be on line 6, they cannot both be in the cache at the same time.

Locations 0x2536038 and 0x10F4 appear as follows in binary:

value	tag	index	block	
0x2536038	0000 0010 0101 0011 0110 0000	0011	10	00
0x10F4	0000 0000 0000 0000 0001 0000	1111	01	00

Locations 0x2536038 and 0x10F4 map to lines 3 and 15 respectively, and can both be in cache at the same time.

e) Since there are 4 words in a line, a miss would require that 4 words be fetched from memory.

Problem 2

- a) For all caches, since each word is 32 bits, the 2 LSBs are ignored. Since there are 16 words in each block, 4 bits are required to represent the block location.

One L1 cache holds 64k, however, since the cache is 2-way associative, there are 32k addressable locations. Given that the block size is 64 bytes, there are $\frac{32k}{64} = 512$ lines. The index must thus be 9 bits. This leaves 17 bits for the tag.

tag	index	block	alignment
17	9	4	2

The L2 cache holds 256k and is 16-way associative, resulting in 16k addressable locations. Given that the block size is 64 bytes, there are $\frac{16k}{64} = 256$ lines. The index must thus be 8 bits. This leaves 18 bits for the tag.

tag	index	block	alignment
18	8	4	2

- b) Recall that the block size for all caches was 64 bytes, or 512 bits.

L1 instruction cache:

$$512 \text{ lines} \times 2 \text{ sets} \times (17 \text{ tag bits} + 512 \text{ bit block}) + 512 \text{ lines} \times 1 \text{ LRU bit} = 542,208 \text{ bits}$$

L1 data cache:

$$512 \text{ lines} \times 2 \text{ sets} \times (4 \text{ dirty bits} + 17 \text{ tag bits} + 512 \text{ bit block}) + 512 \text{ lines} \times 1 \text{ LRU bit} = 546,304 \text{ bits}$$

L2 cache:

$$256 \text{ lines} * 16 \text{ sets} * (18 \text{ tag bits} + 512 \text{ bit block}) = 2,170,880 \text{ bits}$$

- c) Recall that if a cache has a hit rate of m , the miss rate is $(1 - m)$

The average instruction fetch cost is:

$$95\% \times 3 + 5\% \times 98\% \times (3 + 5) + 5\% \times 2\% \times (3 + 5 + 20) = 3.27 \text{ cycles}$$

The average data fetch cost is:

$$0.8 \times 3 + (1 - 0.8) \times (0.98 \times (3 + 5) + 0.02 \times (3 + 5 + 20)) = 4.08 \text{ cycles}$$

Every instruction incurs a cost of 3.27, but 20% of those instructions also access memory.

So, the overall effectiveness is:

$$3.27 \times 80\% + (3.27 + 4.08) \times 20\% = 4.09 \text{ cycles}$$

- d) After calculating similarly to problem c), the instruction fetch cost is 3.33 cycles. The data fetch cost is 4.32 cycles. The overall cost is 4.19 cycles.
- e) Answers will vary. Improving set-associativity will likely increase performance.

Problem 3

- a) The hit ratio will be $(1 - m)$ and each hit will take 1 cycle. The miss rate is m and each miss will cost $(1 + 2 + 3 + B)$ cycles. This leads to a general formula of:

$$(1 - m)1 + m(1 + 2 + 3 + B)$$

which can be simplified to: $1 + 5m + Bm$

- b) By testing all the configurations in the table, it can be found that a block size of 4 yields the fastest performance.

- c) Since now we can fetch 4 words at once, the formula changes to:

$$1 + 5m + \frac{1}{4}Bm$$

The fastest block size becomes 16.

- d) Since memory cost is now 12 cycles, the formula changes to:

$$(1 - m)1 + m(12 + \frac{1}{4}B)$$

or $1 + 11m + \frac{1}{4}Bm$

The fastest block size becomes 64.

Problem 4

a) The miss rate is $\frac{1}{4}$ as seen below:

Addr	Line	Miss?	Replace bits
100	2	m	1-01
1000	0	m	0-11
101	3	m	1-10
102	1	m	0-00
100	2		1-01
1001	0	m	0-11
101	3		1-10
102	1		0-00
100	2		1-01
1002	0	m	0-11
101	3		1-10
102	1		0-00
100	2		1-01
1003	0	m	0-11
101	3		1-10
102	1		0-00

- b) Lori's algorithm works by dividing the cache sets into groups half the size of the previous group. The first bit indicates half of cache is old. The second bit indicates which fourth of cache is old, and so on. Using this method, there will be $\log n$ levels. Each level needs twice as many bits as the previous level (1, 2, 4, 8, ...). The first level only needs 1 bit, so the total number is $n - 1$ bits.
- c) Consider if the replacement bits are 0-00 and this state was reached by the access pattern (0, 2, 3, 1). The LRU line is clearly 0, but since Lori's replacement bits are 0-00, line 2 will be replaced on the next miss.
- d) The worst case is described in c), that is, the second LRU line is discarded instead of the LRU line. The third LRU line can never be discarded out of line since it is either the least recently used line on the most recently accessed side, or the most recently used line on the least recently accessed side.