Computer Architecture, Fall 2017

Homework 5

DUE DATE: DECEMBER 13, 2017

學號：b03902129　系級：資工四　姓名：陳鵬宇

5.2 Caches are important to providing a high-performance memory hierarchy to processors. Below is a list of 32-bit memory address references, given as word addresses.

5.2.1

Word Address	Binary Address	Tag	Index	Hit/Miss
3	0000 0011	0	3	Miss
180	1011 0100	11	4	Miss
43	0010 1011	2	11	Miss
2	0000 0010	0	2	Miss
191	1011 1111	11	15	Miss
88	0101 1000	5	8	Miss
190	1011 1110	11	14	Miss
14	0000 1110	0	14	Miss
181	1011 0101	11	5	Miss
44	0010 1100	2	12	Miss
186	1011 1010	11	10	Miss
253	1111 1101	15	13	Miss

5.2.2

Word Address	Binary Address	Tag	Index	Hit/Miss
3	0000 0011	0	1	Miss
180	1011 0100	11	2	Miss
43	0010 1011	2	5	Miss
2	0000 0010	0	1	Hit
191	1011 1111	11	7	Miss
88	0101 1000	5	4	Miss
190	1011 1110	11	7	Hit
14	0000 1110	0	7	Miss
181	1011 0101	11	2	Hit
44	0010 1100	2	6	Miss
186	1011 1010	11	5	Miss
253	1111 1101	15	6	Miss

5.2.3

Word Address	Binary Address	Tag	Cache 1		Cache 2		Cache 3	
			index	hit/miss	index	hit/miss	index	hit/miss
3	0000 0011	0	3	Miss	1	Miss	0	Miss
180	1011 0100	22	4	Miss	2	Miss	1	Miss
43	0010 1011	5	3	Miss	1	Miss	0	Miss
2	0000 0010	0	2	Miss	1	Miss	0	Miss
191	1011 1111	23	7	Miss	3	Miss	1	Miss
88	0101 1000	11	0	Miss	0	Miss	0	Miss
190	1011 1110	23	6	Miss	3	Hit	1	Hit
14	0000 1110	1	6	Miss	3	Miss	1	Miss
181	1011 0101	22	5	Miss	2	Hit	1	Miss
44	0010 1100	5	4	Miss	2	Miss	1	Miss
186	1011 1010	23	2	Miss	1	Miss	0	Miss
253	1111 1101	31	5	Miss	2	Miss	1	Miss

Cache 1 miss rate = 100%

Cache 1 total cycles = $12 \times 25 \times +12 \times 2 = 324$

Cache 2 miss rate = $10/12 = 83\%$

Cache 2 total cycles = $10 \times 25 + 12 \times 3 = 286$

Cache 3 miss rate = $11/12 = 92\%$

Cache 3 total cycles = $11 \times 25 + 12 \times 5 = 335$
Cache 2 provides the best performance.

5.2.4 First we must compute the number of cache blocks in the initial cache configuration. For this, we divide 32 KiB by 4 (for the number of bytes per word) and again by 2 (for the number of words per block). This gives us 4096 blocks and a resulting index field width of 12 bits. We also have a word offset size of 1 bit and a byte offset size of 2 bits. This gives us a tag field size of 32 - 15 = 17 bits. These tag bits, along with one valid bit per block, will require $18 \times 4096 = 73728$ bits or 9216 bytes. The total cache size is thus 9216 + 32768 = 41984 bytes.

The total cache size can be generalized to
totalsize = datasize + (validbitsize - tagsize) × blocks
totalsize = 41984
datasize = blocks × blocksize× wordsize
wordsize = 4
tagsize = $32 - \log 2(\text{blocks}) - \log 2(\text{blocksize}) - \log 2(\text{wordsize})$
validbitsize = 1

Increasing from 2-word blocks to 16-word blocks will reduce the tag size from 17 bits to 14 bits.

In order to determine the number of blocks, we solve the inequality:

$$41984 \leq 64 \times \text{blocks} + 15 \times \text{blocks}.$$

Solving this inequality gives us 531 blocks, and rounding to the next power of two gives us a 1024-block cache.

The larger block size may require an increased hit time and an increased miss penalty than the original cache. The fewer number of blocks may cause a higher conflict miss rate than the original cache.

5.2.5 Associative caches are designed to reduce the rate of conflict misses. As such, a sequence of read requests with the same 12-bit index field but a different tag field will generate many misses. For the cache described above, the sequence $0, 32768, 0, 32768, 0, 32768, \ldots$, would miss on every access, while a 2-way set associate cache with LRU replacement, even one with a significantly smaller overall capacity, would hit on every access after the first two.

5.2.6 Yes, it is possible to use this function to index the cache. However, information about the five bits is lost because the bits are XOR'd, so you must include more tag bits to identify the address in the cache.

5.4

5.4.1 The L1 cache has a low write miss penalty while the L2 cache has a high write miss penalty. A write buffer between the L1 and L2 cache would hide the write miss latency of the L2 cache. The L2 cache would benefit from write buffers when replacing a dirty block, since the new block would be read in before the dirty block is physically written to memory.

5.4.2 On an L1 write miss, the word is written directly to L2 without bringing its block into the L1 cache. If this results in an L2 miss, its block must be brought into the L2 cache, possibly replacing a dirty block which must first be written to memory.

5.4.3 After an L1 write miss, the block will reside in L2 but not in L1. A subsequent read miss on the same block will require that the block in L2 be written back to memory, transferred to L1, and invalidated in L2.

5.4.4 One in four instructions is a data read, one in ten instructions is a data write. For a CPI of 2, there are 0.5 instruction accesses per cycle, 12.5% of cycles will require a data read, and 5% of cycles will require a data write. The instruction bandwidth is thus $(0.0030 \times 64) \times 0.5 = 0.096$ bytes/cycle. The data read bandwidth is thus $0.02 \times (0.13 + 0.050) \times 64 = 0.23$ bytes/cycle. The total read bandwidth requirement is 0.33 bytes/cycle. The data write bandwidth requirement is $0.05 \times 4 = 0.2$ bytes/cycle.

5.4.5 The instruction and data read bandwidth requirement is the same as in 5.4.4. The data write bandwidth requirement becomes $0.02 \times 0.30 \times (0.13 + 0.050) \times 64 = 0.069$ bytes/cycle.

5.4.6 For CPI = 1.5 the instruction throughput becomes $1/1.5 = 0.67$ instructions per cycle. The data read frequency becomes $0.25/1.5 = 0.17$ and the write frequency becomes $0.10/1.5 = 0.067$.

The instruction bandwidth is $(0.0030 \times 64) \times 0.67 = 0.13$ bytes/cycle.

For the write-through cache, the data read bandwidth is $0.02 \times (0.17 + 0.067) \times 64 = 0.22$ bytes/cycle. The total read bandwidth is 0.35 bytes/cycle. The data write bandwidth is $0.067 \times 4 = 0.27$ bytes/cycle.

For the write-back cache, the data write bandwidth becomes $0.02 \times 0.30 \times (0.17 + 0.067) \times 64 = 0.091$ bytes/cycle.

Address	0	4	16	132	232	160	1024	30	140	3100	180	2180
Line ID	0	0	1	8	14	10	0	1	9	1	11	8
Hit/Miss	Miss	Hit	Miss	Miss	Miss	Miss	Miss	Hit	Hit	Miss	Miss	Miss
Replace	N	N	N	N	N	N	Y	N	N	Y	N	Y

5.12

5.12.1 Worst case is 2^{43-12} entries, requiring $2^{43-12} \times 4$ bytes $= 2^{33} = 8$ GB.

5.12.2 With only two levels, the designer can select the size of each page table segment. In a multi-level scheme, reading a PTE requires an access to each level of the table.

5.12.3 In an inverted page table, the number of PTEs can be reduced to the size of the hash table plus the cost of collisions. In this case, serving a TLB miss requires an extra reference to compare the tag or tags stored in the hash table.

5.12.4 It would be invalid if it was paged out to disk.

5.12.5 A write to page 30 would generate a TLB miss. Software-managed TLBs are faster in cases where the software can pre-fetch TLB entries.

5.12.6 When an instruction writes to VA page 200, and interrupt would be generated because the page is marked as read only.