Computer Science M151B, Homework 8

Michael Wu
UID: 404751542

May 30th, 2018

Problem 1

The average number of cycles that a memory access will take is

$$0.94 \times 1 \text{ cycles} + 0.06 \times 55 \text{ cycles} = 4.24 \text{ cycles}$$

Each cycle takes 0.5 ns, which means that the effective access time is

$$2.12 \times 10^{-9} \text{ s}$$

Problem 2

The number of sets in the cache would decrease to 128, which would require 7 index bits and 23 tag bits. Each cache entry has 1 valid bit, 23 tag bits, and 32 data bits, which is 56 bits total. There are $128 \times 8 = 1024$ cache entries total which means that the bits of storage required to implement the cache is 57344 bits.

Problem 3

In a fully associative cache, there would be no index which leaves us with 15 bits of block offset. Thus the block size at the maximum could be 32768 words. In a direct-mapped cache, there would be 11 index bits which leaves 4 bits of block offset. Thus the block size at a minimum could be 16 words.

Problem 4

The cache information is shown below. Note that since these are 32-bit addresses, both the binary address and the tag would be extended by an additional 24 bits that are entirely zero. I have left them out for readability.

Word Address	Binary Address	Tag	Index	Hit/Miss
3	00000011	0000	001	Miss
180	10110100	1011	010	Miss
43	00101011	0010	101	Miss
2	00000010	0000	001	Hit
191	10111111	1011	111	Miss
88	01011000	0101	100	Miss
190	10111110	1011	111	Hit
14	00001110	0000	111	Miss
181	10110101	1011	010	Hit
44	00101100	0010	110	Miss
186	10111010	1011	101	Miss
253	11111101	1111	110	Miss

The final cache contents are shown below.

Block Number	Word Addresses	Tag
1	2-3	0000
2	180-181	1011
4	88-89	0101
5	186-187	1011
6	252-253	1111
7	14-15	0000

Problem 5

Assuming the cache is byte addressable, one miss will occur for every 16 accesses. Thus the miss rate is 6.25%. The miss rate does not depend on the working set size, since no memory location is referenced more than once. So replacing block frames is not the reason for our misses, and we do not need to hold the working set in memory. The miss rate can vary with the size of the cache, but only if the cache size changes due to changing the block size. Adding or removing block frames will not change the miss rate. Increasing

the block size will reduce the miss rate, as more accesses can occur before the program moves onto the next block of memory. The types of misses that occur in this program are compulsory misses.

Problem 6

The miss rate with a 16 byte block size will be 12.5%, the miss rate with a 64 byte block size will be 3.125%, and the miss rate with a 128 byte block size will be 1.5625%. The workload is exploiting spatial locality.

Problem 7

Since only the first access will be a miss and the working set size is 524288 bytes, one out of 524288 accesses will be a miss. Thus the miss rate will be 0.000190735%.

Problem 8

a) Each set will have $8 \times 4 = 32$ words in it, so there are a maximum of 2048 sets that the chip can hold. I assume that the cache attempts to use all of these sets. Then there will be 3 bits of block offset and 11 bits for the index. The cache controller will generate 2 bits that indicate which block frame in the set is a hit. Together this provides the 16 bits necessary to generate an address into the RAM chip. I assume that the address into the RAM chip has the 11 index bits as the highest order bits, then the 3 block offset bits, then the 2 block frame bits as the lowest order bits.

Taking the lowest order 3 bits of the CPU generated address as block offset, our block offset will be `001` in binary. Taking the next 11 bits of the CPU generated address as the index, our index will be `00001100100`. Then by concatenating these values with the possible 2 bits that the cache controller generates, we obtain the RAM chip addresses `0000110010000100`, `0000110010000101`, `0000110010000110`, and `0000110010000111`. Equivalently this would be `0xC84`, `0xC85`, `0xC86`, and `0xC87` in hex.

b) With the same assumptions about the cache implementation above, we get `011` as our block offset and `10010101000` as our index. Then we have the

RAM chip addresses `1001010100001100`, `1001010100001101`, `1001010100001110`, and `1001010100001111`, which would be `0x950C`, `0x950D`, `0x950E`, and `0x950F`.

Problem 9

a) Loads take 5 cycles, stores take 4 cycles, R-format instructions take 4 cycles, branches take 3 cycles, and jumps take 3 cycles. Thus the average CPI is

$$0.27 \times 5 + 0.13 \times 4 + 0.38 \times 4 + 0.2 \times 3 + 0.02 \times 3 = 4.05 \text{ cycles}$$

b) Let the average CPI for memory instructions be x. In order to decrease performance by a factor of 2, we must increase the overall average CPI to 8.1. Thus we need to have

$$\begin{aligned} 0.4x + 0.38 \times 4 + 0.2 \times 3 + 0.02 \times 3 &= 8.1 \text{ cycles} \\ 0.4x &= 5.92 \text{ cycles} \\ x &= 14.8 \text{ cycles} \end{aligned}$$

We have that 32.5% of memory accesses are stores and 67.5% of memory accesses are loads. Assume that misses are evenly distributed among loads and stores. A store that is a cache hit takes 4 cycles. A store that is a cache miss takes 9 cycles. A load that is a cache hit takes 5 cycles. A load that is a cache miss on a clean block takes 10 cycles. A load that is a cache miss on a dirty block takes 16 cycles, since two memory accesses must be made.

Let our miss rate be m. Our average CPI for memory accesses will then be

$$0.325(9m+4(1-m))+0.675(16\times0.3m+10\times0.7m+5(1-m)) = 6.215m+4.675$$

Solving $6.215m + 4.675 = 14.8$ for m yields $m = 1.62912$. Because this is greater than 1, this means that even if every memory access is a cache miss, the performance decrease will still be less than a factor of 2. So for any cache hit or miss rate, our performance will not decrease by a factor of 2.