CS 219: Homework #11

Due on December 7th, 2016 at 4:00pm

Dr. Egbert

Matthew J. Berger

Problem 1

Consider a machine with a byte addressable main memory of 2^{16} bytes and block size of 8 bytes. Assume that a direct mapped cache consisting of 32 lines is used with this machine.

a.) How is a 16-bit memory address divided into tag, line number, and byte number?

b.) Into what line would bytes with each of the following addresses be stored?
0001 0001 0001 1011
1100 0011 0011 0100
1101 0000 0001 1101
1010 1010 1010 1010

c.) Suppose the byte with address 0001 1010 0001 1010 is stored in the cache.What are the addresses of the other bytes stored along with it?

d.) How many total bytes of memory can be stored in the cache?

e.) Why is the tag also stored in the cache?

Solution:

a.) The first 8 bits are the tag, the 5 middle bits are the line, and the 3 rightmost bits are the byte number.

b.) Line 3, Line 6, Line 3, and then Line 21.

c.) All of the bytes that have addresses in the range of 0001 1010 0001 1000 through 0001 1010 0001 1111 are stored in the cache.

d.) 256 total bytes of memory can be stored in the cache.

e.) The tag is also stored in the cache because two objects with two different memory addresses can technically be stored in the same exact place in the cache. The tag is necessary to tell them apart.

Problem 2

Consider a memory system that uses a 32-bit address to address at the byte level, plus a cache that uses a 64-byte line size.

a.) Assume a direct mapped cache with a tag field in the address of 20 bits. Show the address format and determine the following parameters: number of addressable units, number of blocks in main memory, number of lines in cache, size of tag.

b.) Assume an associative cache. Show the address format and determine the following parameters: number of addressable units, number of blocks in main memory, number of lines in cache, size of tag.

c.) Assume a four-way set-associative cache with a tag field in the address of 9 bits. Show the address format and determine the following parameters: number of addressable units, number of blocks in main memory, number of lines in set, number of sets in cache, number of lines in cache, size of tag.

Solution:

a.)
- Address format: Tag = 20 bits. Line = 6 bits. Word = 6 bits.
- Number of addressable units: $2^{S+W} = 2^{32}$ bytes.
- Number of blocks in main memory: $2^S = 2^{26}$
- Number of lines in cache: $2^R = 2^6 = 64$
- Size of tag: 20 bits

b.)
- Address format: Tag = 26 bits. Word = 6 bits.
- Number of addressable units: $2^{s+w} = 2^{32}$ bytes.
- Number of blocks in main memory: $2^s = 2^{26}$
- Number of lines in cache: Undetermined
- Size of tag: 26 bits

c.)
- Address format: Tag = 9 bits. Set = 17 bits. Word = 6 bits.
- Number of addressable units: $2^{S+W} = 2^{32}$ bytes.
- Number of blocks in main memory: $2^S = 2^{26}$
- Number of lines in set: $k = 4$
- Number of sets in cache: $2^d = 2^{17}$
- Number of lines in cache: $k \times 2^d = 2^{19}$
- Size of tag: 9 bits

Problem 3

Suppose an 8-bit data word stored in memory is 11000010. Using the Hamming algorithm, determine what check bits would be stored in memory with the data word. Show how you got your answer.

Solution:

Value 1 data bits are in the bit positions 12, 11, 5, 4, 2, and 1 as demonstrated below.

Position	12	11	10	9	8	7	6	5	4	3	2	1
Bits	D8	D7	D6	D5	C8	D4	D3	D2	C4	D1	C2	C1
Block	1	1	0	0		0	0	1		0		
Codes	1100	1011						0101				

The check bits are in bits 12, 11, 10, and 9.
Check bit 8 was calculated using values in bits: 12, 11, 10, and 9.
Check bit 4 was calculated using values in bits: 12, 7, 6, and 5.
Check bit 2 was calculated using values in bits: 11, 10, 7, 6, and 3.
Check bit 1 was calculated using values in bits: 11, 9, 7, 5, and 3.

Problem 4

For the 8-bit word 00111001, the check bits stored with it would be 0111. Suppose when the word is read from memory, the check bits are calculated to be 1101.What is the data word that was read from memory?

Solution:

The Hamming Word was initially this:

12	11	10	9	8	7	6	5	4	3	2	1
0	0	1	1	0	1	0	0	1	1	1	1

Doing an XOR of 0111 and 1101 yields 1010, which means there was an error in bit 10 of our Hamming Word. This means the data word that was read from memory was **00011001**.