Tutorial 9 – Week of Nov. 15

Question: 5.2

Caches are important to providing a high-performance memory hierarchy to processors. Below is a list of 64-bit memory address references, given as word addresses.
0x03, 0xb4, 0x2b, 0x02, 0xbf, 0x58, 0xbe, 0x0e, 0xb5, 0x2c, 0xba, 0xfd

1 word = 4 bytes

5.2.1 For each of these references, identify the binary word address, the tag, and the index given a direct-mapped cache with 16 one-word blocks. Also list whether each reference is a hit or a miss, assuming the cache is initially empty.

word addressable
assume we only wan to access data by word
(i,e, no byte offset)
(normally: offset = block offset + byte offset
in this case: offset = block offset)

5.2.2 For each of these references, identify the binary word address, the tag, the index, and the offset given a direct mapped cache with two-word blocks and a total size of eight blocks. Also list if each reference is a hit or a miss, assuming the cache is initially empty.

5.2.3 You are asked to optimize a cache design for the given references. There are three direct-mapped cache designs possible, all with a total of eight words of data:

C1 has 1-word blocks,	#blocks = 8 / 1 = 8.	index field = 3 bits.	block offset = 0 bits.	tag = 8 - 3 - 0 = 5
C2 has 2-word blocks, and	#blocks = 8 / 2 = 4.	index field = 2 bits.	block offset = 1 bits.	tag = 8 - 2 - 1 = 5
C3 has 4-word blocks.	#blocks = 8 / 4 = 2.	index field = 1 bits.	block offset = 2 bits.	tag = 8 - 1 - 2 = 5

Solution:

5.2

5.2.1

Word Address	Binary Address	Tag	Index	Hit/Miss
0x03	0000 0011	0	3	M
0xb4	1011 0100	b	4	M
0x2b	0010 1011	2	b	M
0x02	0000 0010	0	2	M
0xbf	1011 1111	b	f	M
0x58	0101 1000	5	8	M
0xbe	1011 1110	b	e	M
0x0e	0000 1110	0	e	M
0xb5	1011 0101	b	5	M
0x2c	0010 1100	2	c	M
0xba	1011 1010	b	a	M
0xfd	1111 1101	f	d	M

5.2.2

Word Address	Binary Address	Tag	Index	Offset	Hit/Miss
0x03	0000 0011	0	1	1	M
0xb4	1011 0100	b	2	0	M
0x2b	0010 1011	2	5	1	M
0x02	0000 0010	0	1	0	H
0xbf	1011 1111	b	7	1	M
0x58	0101 1000	5	4	0	M
0xbe	1011 1110	b	6	0	H
0x0e	0000 1110	0	7	0	M
0xb5	1011 0101	b	2	1	H
0x2c	0010 1100	2	6	0	M
0xba	1011 1010	b	5	0	M
0xfd	1111 1101	f	6	1	M

Tag	Index	offset

5.2.3

Word Address	Binary Address	5 bits Tag	Cache 1		Cache 2		Cache 3	
			3 bits Index	Hit/miss	2 bits Index	Hit/miss	1 bits Index	Hit/miss
0x03	0000 0011	0x00	3	M	1	M	0	M
0xb4	1011 0100	0x16	4	M	2	M	1	M
0x2b	0010 1011	0x05	3	M	1	M	0	M
0x02	0000 0010	0x00	2	M	1	M	0	M
0xbf	1011 1111	0x17	7	M	3	M	1	M
0x58	0101 1000	0x0b	0	M	0	M	0	M
0xbe	1011 1110	0x17	6	M	3	H	1	H
0x0e	0000 1110	0x01	6	M	3	M	1	M
0xb5	1011 0101	0x16	5	M	2	H	1	M
0x2c	0010 1100	0x05	4	M	2	M	1	M
0xba	1011 1010	0x17	2	M	1	M	0	M
0xfd	1111 1101	0x1F	5	M	2	M	1	M

Cache 1 miss rate = 100%

~~Cache 1 total cycles = 12 × 25 + 12 × 2 = 324~~

Cache 2 miss rate = 10/12 = 83%

~~Cache 2 total cycles = 10 × 25 + 12 × 3 = 286~~

Cache 3 miss rate = 11/12 = 92%

~~Cache 3 total cycles = 11 × 25 + 12 × 5 = 335~~

Cache 2 provides the best performance.

Question: 5.3

By convention, a cache is named according to the amount of data it contains (i.e., a 4 KiB cache can hold 4 KiB of data); however, caches also require SRAM to store metadata such as tags and valid bits. For this exercise, you will examine how a cache's configuration affects the total amount of SRAM needed to implement it as well as the performance of the cache. For all parts, assume that the caches are byte addressable, and that addresses and words are 64 bits.

5.3.1 Calculate the total number of bits required to implement a 32 KiB cache with two-word blocks.

5.3.2 Calculate the total number of bits required to implement a 64 KiB cache with 16-word blocks. How much bigger is this cache than the 32 KiB cache described in Exercise 5.3.1? (Notice that, by changing the block size, we doubled the amount of data without doubling the total size of the cache.)

5.3.3 Explain why this 64 KiB cache, despite its larger data size, might provide slower performance than the first cache.

5.3.4 Generate a series of read requests that have a lower miss rate on a 32 KiB two-way set associative cache than on the cache described in Exercise 5.3.1.

This solution is NOT correct because it says "each word is 8 bytes"

corrected version:
5.3.1
A word is 4 bytes. Two words are 8 bytes = 2^3 bytes.
The cache contains 2^15/2^3=2^12 lines of data.
Each 64 bit address is divided
2 bits byte offset
1 bit word offset
12 bits index
and 48 bits tag (64-12-1-3)

For 32 bit address, everything is the same except the number of tag bits, in this case, it is 16 bits.

Solution:

5.3

5.3.1 Total size is 364,544 bits = 45,568 bytes

Each word is 8 bytes; each block contains two words; thus, each block contains 16 = 2^4 bytes.

The cache contains 32KiB = 2^15 bytes of data. Thus, it has 2^15/2^4 = 2^11 lines of data.

Each 64-bit address is divided into: (1) a 3-bit word offset, (2) a 1-bit block offset, (3) an 11-bit index (because there are 2^11 lines), and (4) a 49-bit tag (64 − 3 − 1 − 11 = 49).

The cache is composed of: 2^15 * 8 bits of data + 2^11*49 bits of tag + 2^11*1 valid bits = 364,544 bits.

X

5.3.2 549,376 bits = 68,672 bytes. This is a 51% increase.

Each word is 8 bytes; each block contains 16 words; thus, each block contains 128 = 2^7 bytes.

The cache contains 64KiB = 2^16 bytes of data. Thus, it has 2^16/2^7 = 2^9 lines of data.

Each 64-bit address is divided into: (1) a 3-bit word offset, (2) a 4-bit block offset, (3) a 9-bit index (because there are 2^9 lines), and (4) a 48-bit tag (64 − 3 − 4 − 9 = 48).

The cache is composed of: 2^16 * 8 bits of data + 2^9*48 bits of tag + 2^9*1 valid bits = 549,376 bits

X

5.3.3 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.

5.3.4 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, …, would miss on every access, while a two-way set associate cache with LRU replacement, even one with a significantly smaller overall capacity, would hit on every access after the first two.