Tu Lam

CS 472 / Justin Goins

April 24th, 2021

Homework #2

(Cache)

1. *Suppose the existence of a direct-mapped cache design that utilizes a 26-bit address. Assume that the addressable memory space is byte-indexed. The cache uses the following breakdown of address bits*:

Tag	Index	Offset
25-8	7-5	4-0

a) *What is the cache block size (in bytes)?*

Answer: To calculate the cache block size, we can follow the calculation of cache block size = $2^{(offset)}$. Following this we see that offset to have 5 bits so we will have $2^{(5)}$ = **32 bytes** of block size.

b) *How many blocks can the cache contain?*

Answer: The index field has 3 bits (7-5), so the cache can contain the number of blocks by 2^{index} which is 2^3 = **8 blocks.**

c) *Including space for the valid bits, tags, and actual block data, how many bits would be required to implement this hypothetical cache?*

Answer: If we included the actual block data (256 bytes = 32 bytes * 8 blocks), the valid bit (1 bytes), and the tag (18 bytes), then we would have a total of (1 + 18 + 256) * 8 bits = **2,200 bits** roughly to implement this hypothetical cache.

2. *Using the cache described in problem 1, assume the following byte-addressed cache references are recorded. Assume that the cache is initially empty. Also assume that accesses occurred from left to right (e.g. address 71 was requested, then address 65, followed by address 1927, etc).*

Byte Address											
71	65	1927	244	585	225	900	1616	1410	81	590	1942

a) *For each memory access, indicate whether the activity generated a "hit" or a "miss". If a memory access triggered a block eviction, be sure to indicate this.*

Answer: Below is a table showing the "miss" or "hit" for each byte address:

Address	Tag	Index	Offset	Hit or Miss
71	00 0000 0000 0000 0000	010	0 0111	Miss
65	00 0000 0000 0000 0000	010	0 0001	Hit
1927	00 0000 0000 0000 0111	100	0 0111	Miss
244	00 0000 0000 0000 0000	111	1 0100	Miss
585	00 0000 0000 0000 0010	010	0 1001	Miss
225	00 0000 0000 0000 0000	111	0 0001	Hit
900	00 0000 0000 0000 0011	100	0 0100	Miss
1616	00 0000 0000 0000 0110	010	1 0000	Miss
1410	00 0000 0000 0000 0101	100	0 0010	Miss
81	00 0000 0000 0000 0000	010	1 0001	Miss
590	00 0000 0000 0000 0010	010	0 1110	Miss
1942	00 0000 0000 0000 0111	100	1 0110	Miss

b) *Create a table to show the final state of the cache including the value of each Index, Valid bit, and Tag. If a value is unknown, you may leave it blank.*

Answer: Below is a table that show index, valid bit, and tag at the final state:

Index	Valid Bit	Tag
000	0	-
001	0	-
010	1	00 0000 0000 0000 0010
011	0	-
100	1	00 0000 0000 0000 0111
101	0	-
110	0	-
111	1	00 0000 0000 0000 0000

3. *Now imagine the existence of a 4-way set-associative cache (n=4) with a 26-bit address. The cache uses a "least recently used" replacement scheme. Assume that the addressable memory space is byte-indexed. The cache uses the following breakdown of address bits (same arrangement as problem #1):*

Tag	Index	Offset
25-8	7-5	4-0

a) *What is the cache block size (in bytes)?*

Answer: To calculate the cache block size, we can follow the calculation of cache block size = $2^{(offset)}$ and from there we can calculate it as 2^5 = **32 bytes** of cache block size.

b) *How many blocks can the cache contain?*

Answer: The index field has 3 bits (7-5), so the cache can contain the number of blocks by 2^{index} which is 2^3 = **8 blocks**.

c) *Including space for the valid bits, tags, and actual block data, how many bits would be required to implement this hypothetical cache?*

Answer: If we included the actual block data (256 bytes = 32 bytes * 8 blocks), the valid bit (1 bytes), and the tag (18 bytes), then we would have a total of (1 + 18 + 256) * 8 bits = 2,200 bits then times it by 4 to get **8,800 bits** roughly to implement this hypothetical cache.

4. *Using the cache described in problem 3, assume the following byte-addressed cache references are recorded. Assume that the cache is initially empty. As before, assume that accesses occurred from left to right (e.g. address 71 was requested, then address 65, followed by address 1927, etc)*

Byte Address											
71	65	1927	244	585	225	900	1616	1410	81	590	1942

a) *For each memory access, indicate whether the activity generated a "hit" or a "miss". If a memory access triggered a block eviction, be sure to indicate this.*

Answer: Below is a table that will determine hit or miss:

Address	Index	Tag1	Tag2	Tag3	Tag4	H or M
71	010	00 0000 0000 0000 0000				Miss
65	010	00 0000 0000 0000 0000				Hit
1927	100	00 0000 0000 0000 0111				Miss
244	111	00 0000 0000 0000 0000				Miss
585	010		00 0000 0000 0000 0010			Miss
225	111	00 0000 0000 0000 0000				Hit
900	100		00 0000 0000 0000 0011			Miss
1616	010			00 0000 0000 0000 0110		Miss
1410	100			00 0000 0000 0000 0101		Miss

81	010	00 0000 0000 0000 0000				Hit
590	010		00 0000 0000 0000 0010			Hit
1942	100	00 0000 0000 0000 0111				Hit

b) *Create a table to show the final state of the cache including the value of each Index, Valid bit, and Tag. If a value is unknown, you may leave it blank.*

Answer: Below is the final state of the cache including the value of each index, valid bit, and tag.

		4-Ways Set Associative			
Index	**Valid Bit**	**Tag 1**	**Tag 2**	**Tag 3**	**Tag4**
000	0	-	-	-	-
001	0	-	-	-	-
010	1	00 0000 0000 0000 0000	00 0000 0000 0000 0010	00 0000 0000 0000 0110	-
011	0	-	-	-	-
100	1	00 0000 0000 0000 0111	00 0000 0000 0000 0011	00 0000 0000 0000 0101	-
101	0	-	-	-	-
110	0	-	-	-	-
111	1	00 0000 0000 0000 0000	-	-	-

5. *Suppose we are working with a processor that exhibits an average CPI (cycles per instruction) of 1.2, assuming all references hit the primary cache, and a clock speed of 3 GHz. Assume a main memory access time of 120 cycles (including all miss handling). Imagine that the miss rate per instruction at the primary cache is 3.5%. How much faster will the system operate (on average) if we add a secondary cache with an access time (hit or miss) of 5 ns? Assume that the new secondary cache exhibits a miss rate of 0.8%.*

Answer: Given the information, we know that:

CPI = 1.2
Clock Rate = 3 GHz
Miss Rate for Primary Cache = 3.5%
Main Memory Access Time = 120 cycles

From this, we can calculate the primary cache effective CPI as:

Effective CPI: 1.2 + (0.035 * 120) = **5.4 for primary cache**

Now adding in secondary cache, we know that:

Access Time (Hit or Miss) = 5 ns
Miss Rate for Secondary Cache = 0.8%

Then we can calculate the prime hit with secondary cache as, 5 / 0.4 = 12.5 cycles. Then we can plug in to find the effective CPI for the secondary cache is:

Effective CPI: 1.2 + (0.035 * 12.5) + (0.008 * 120) = **2.6 for the secondary cache**

Performance Ratio: 5.4 / 2.6 = **2.1 perform faster** if add a secondary cache.