Memory Hierarchy Problems

April 27, 2023

1. Suppose we have a memory and a direct-mapped cache with the following characteristics.
   - Memory is byte addressable
   - Memory addresses are 16 bits (i.e., the total memory size is 2 16 = 65536 bytes)
   - The cache has 8 rows (i.e., 8 cache lines)
   - Each cache row (line) holds 16 bytes of data

   How 16 address bits will be allocated to the offset, index, and tag parts of the address used to reference the cache?

1. Suppose we have a memory and a direct-mapped cache with the following characteristics.
   - Memory is byte addressable
   - Memory addresses are 16 bits (i.e., the total memory size is 2^16 = 65536 bytes)
   - The cache has 8 rows (i.e., 8 cache lines)
   - Each cache row (line) holds 16 bytes of data

   How 16 address bits will be allocated to the offset, index, and tag parts of the address used to reference the cache?

   4 bits offset
   3 bit index
   9 bit tag

2. Below is a sequence of four binary memory addresses with 16 address bits in the order they are used to reference memory. Assume that the cache is initially empty. For each reference, write down the tag and index bits and indicate whether that reference is a hit or a miss.

0010 1101 1011 0011
0000 0110 1111 1100
0010 1101 1011 1000
1010 1010 1010 1011

2. Below is a sequence of four binary memory addresses with 16 address bits in the order they are used to reference memory. Assume that the cache is initially empty. For each reference, write down the tag and index bits and indicate whether that reference is a hit or a miss.

0010 1101 1011 0011
0000 0110 1111 1100
0010 1101 1011 1000
1010 1010 1010 1011

address	index	tag	hit/miss
0010 1101 1011 0011	011	0010 1101 1	miss
0000 0110 1111 1100	111	0000 0110 1	miss
0010 1101 1011 1000	011	0010 1101 1	hit
1010 1010 1010 1011	010	1010 1010 1	miss

3. Different storage devices have different access times and costs. For each of the following devices, circle the access time that is closest to the typical time needed to read information from that device given current hardware technologies. The important thing is to get the order of magnitude right; the exact number is not the issue. (ms = millisecond, μs = microsecond, ns = nanosecond). If it matters, assume that these numbers are for a computer with a 1GHz clock (i.e., a low-end current machine)

Main memory (RAM) : 1sec 100ms 10ms 1ms 100μs 10μs 1μs 100ns 10ns 1ns
Hard disk drive: 1sec 100ms 10ms 1ms 100μs 10μs 1μs 100ns 10ns 1ns
CPU register: 1sec 100ms 10ms 1ms 100μs 10μs 1μs 100ns 10ns 1ns

3. Different storage devices have different access times and costs. For each of the following devices, circle the access time that is closest to the typical time needed to read information from that device given current hardware technologies. The important thing is to get the order of magnitude right; the exact number is not the issue. (ms = millisecond, μs = microsecond, ns = nanosecond). If it matters, assume that these numbers are for a computer with a 1GHz clock (i.e., a low-end current machine)

Main memory (RAM) : 1sec 100ms 10ms 1ms 100μs 10μs 1μs 100ns 10ns 1ns
Hard disk drive: 1sec 100ms 10ms 1ms 100μs 10μs 1μs 100ns 10ns 1ns
CPU register: 1sec 100ms 10ms 1ms 100μs 10μs 1μs 100ns 10ns 1ns

4. Two of the design choices in a cache are the row size (number of bytes per row or line) and whether each row is organized as a single block of data (direct mapped cache) or as more than one block (2-way or 4-way set associative). The goal of a cache is to reduce overall memory access time. Suppose that we are designing a cache and we have a choice between a direct-mapped cache where each row has a single 64-byte block of data, or a 2-way set associative cache where each row has two 32-byte blocks of data. Which one would you choose and why? Give a brief technical justification for your answer. If the choice would make no difference in performance then explain why not.

4. Two of the design choices in a cache are the row size (number of bytes per row or line) and whether each row is organized as a single block of data (direct mapped cache) or as more than one block (2-way or 4-way set associative). The goal of a cache is to reduce overall memory access time. Suppose that we are designing a cache and we have a choice between a direct-mapped cache where each row has a single 64-byte block of data, or a 2-way set associative cache where each row has two 32-byte blocks of data. Which one would you choose and why? Give a brief technical justification for your answer. If the choice would make no difference in performance then explain why not.

Two blocks of 32-bytes each should give better performance. Reasons:

(a) it will reduce conflicts between cache lines that have the same index bits but different tags i.e. reduce conflict misses

(b) it also will probably avoid bringing in extra data in the wider cache line that is less likely to be used because it has lower spatial locality, and instead can cause more false sharing issues with other threads

(c) it will reduce memory traffic on a cache miss

5. Most memory systems contain both a translation lookaside buffer (TLB) used by the virtual memory system, and a cache memory. Both the cache and the TLB have similar functions: they store main memory data in a faster “cache-like” memory that can be accessed at nearly processor speeds. The question is why do these two separate mechanisms exist when they have such a similar purpose? Why have both a cache and a TLB? Why not just a single “super-cache” that would hold both regular memory data as well as frequently used page table entries?

5. Most memory systems contain both a translation lookaside buffer (TLB) used by the virtual memory system, and a cache memory. Both the cache and the TLB have similar functions: they store main memory data in a faster “cache-like” memory that can be accessed at nearly processor speeds. The question is why do these two separate mechanisms exist when they have such a similar purpose? Why have both a cache and a TLB? Why not just a single “super-cache” that would hold both regular memory data as well as frequently used page table entries?

Although the TLB and the cache both hold copies of main memory data, they are used in different parts of the memory subsystem, and accesses to them have different characteristics.

a. TLB and the cache are used in different parts of a memory access. A TLB uses the virtual address to locate data, while a cache normally uses the physical (translated) address as the index and tag.

b. Each entry in the TLB contains a pair of addresses: a virtual page number and a page frame address. Because individual TLB entries translate a large number of virtual addresses (a full page each), relatively few of them are active at any time. So a TLB tends to be fairly small, and the hardware to look up a virtual page number is fully associative.

   The regular cache, on the other hand, has much more data per row to exploit locality of instruction and data memory references. It also tends to be much larger than a TLB to get a reasonably large hit rate, so it is not practical to use fully associative addressing. Instead, most caches tend to be 2-way or 4-way set associative.

c. A TLB miss and a cache miss are also handled in quite different ways. A TLB miss requires walking the page table; a cache miss does not use a secondary data structure.

Because of these differences it makes sense to use specialized hardware optimized for each function rather than having a single “super cache”.

6. True or false: By having the cache act as a bridge between main memory and the CPU, the time it takes to retrieve a block of data from main memory is longer than it would be if the cache did not exist.

6. True or false: By having the cache act as a bridge between main memory and the CPU, the time it takes to retrieve a block of data from main memory is longer than it would be if the cache did not exist.

True. Added access time for the cache.

7. Assume a memory access to main memory on a cache "miss" takes 30 ns and a memory access to the cache on a cache "hit" takes 3 ns. If 80% of the processor's memory requests result in a cache "hit", what is the average memory access time?

(a) 8.4 nS (b) 33 nS (c) 24.6 nS (d) 27.0 nS (e) 2.4 nS (f) 3.0 nS

7. Assume a memory access to main memory on a cache "miss" takes 30 ns and a memory access to the cache on a cache "hit" takes 3 ns. If 80% of the processor's memory requests result in a cache "hit", what is the average memory access time?

(a) 8.4 nS (b) 33 nS (c) 24.6 nS (d) 27.0 nS (e) 2.4 nS (f) 3.0 nS

Since we know that not all of the memory accesses are going to the cache, the average access time must be greater than the cache access time of 3 ns. This eliminates answers 'e' and 'f'. Similarly, not all accesses are going to the main memory, so the average must be less than 30 ns eliminating 'b' as an answer. For the exact answer, you need to see that 80% of the time, the access will be 3 ns while the rest of the time (20%) the access time will be 30 ns. You should be able to see then that the average is going to be closer to 3 ns than 30 ns meaning the answer is a.

You could also see this with the following equation.

$(0.8 \times 3 \text{ ns}) + (0.2 \times 30 \text{ ns})$
$= 2.4 \text{ ns} + 6 \text{ ns}$
$= 8.4 \text{ ns}$

8. The following cache represents a 2-way set associative cache, i.e., there are two lines per set. Notice that the set ID values start at 01101101_2 and increment every other row. This is meant to imply that you are looking at a group of lines/sets toward the middle of the cache and not the entire cache. There are 14 bits for the tag, 8 bits for the set id, and 2 bits for the word id. Answer the following 3 questions based on this cache.

Tag (binary values)	Set ID (binary values)	Word within block			
		00	01	10	11
10100001001001	01101101	00_{16}	61_{16}	$C2_{16}$	23_{16}
11100001100100	01101101	10_{16}	71_{16}	$D2_{16}$	33_{16}
11001011010110	01101110	20_{16}	81_{16}	$E2_{16}$	43_{16}
11100101101011	01101110	30_{16}	91_{16}	$F2_{16}$	53_{16}
11110110110100	01101111	40_{16}	$A1_{16}$	02_{16}	63_{16}
10100111010101	01101111	50_{16}	$B1_{16}$	12_{16}	73_{16}
10101010111110	01110000	84_{16}	$E5_{16}$	46_{16}	$A7_{16}$
10101010010011	01110000	94_{16}	$F5_{16}$	56_{16}	$B7_{16}$
01110001001000	01110001	$A4_{16}$	$A5_{16}$	66_{16}	$C7_{16}$
00001101101101	01110001	$B4_{16}$	15_{16}	76_{16}	$D7_{16}$
01011010010010	01110010	$C4_{16}$	25_{16}	86_{16}	$E7_{16}$
10101111001011	01110010	$D4_{16}$	35_{16}	96_{16}	$F7_{16}$

a. A copy of the data from memory address 7121C5 (hex) is contained in the portion of the cache shown above. Enter the value that was retrieved from that address in the space below as a two-digit hexadecimal number with no base identification, e.g., 0x4F (hexadecimal 4F) should be entered as 4F.

Tag (binary values)	Set ID (binary values)	Word within block			
		00	01	10	11
10100001001001	01101101	00_{16}	61_{16}	$C2_{16}$	23_{16}
11100001100100	01101101	10_{16}	71_{16}	$D2_{16}$	33_{16}
11001011010110	01101110	20_{16}	81_{16}	$E2_{16}$	43_{16}
11100101101011	01101110	30_{16}	91_{16}	$F2_{16}$	53_{16}
11110110110100	01101111	40_{16}	$A1_{16}$	02_{16}	63_{16}
10100111010101	01101111	50_{16}	$B1_{16}$	12_{16}	73_{16}
10101010111110	01110000	84_{16}	$E5_{16}$	46_{16}	$A7_{16}$
10101010010011	01110000	94_{16}	$F5_{16}$	56_{16}	$B7_{16}$
01110001001000	01110001	$A4_{16}$	$A5_{16}$	66_{16}	$C7_{16}$
00001101101101	01110001	$B4_{16}$	15_{16}	76_{16}	$D7_{16}$
01011010010010	01110010	$C4_{16}$	25_{16}	86_{16}	$E7_{16}$
10101111001011	01110010	$D4_{16}$	35_{16}	96_{16}	$F7_{16}$

a. A copy of the data from memory address 7121C5 (hex) is contained in the portion of the cache shown above. Enter the value that was retrieved from that address in the space below as a two-digit hexadecimal number with no base identification, e.g., 0x4F (hexadecimal 4F) should be entered as 4F.

First, convert 7121C5 to binary. 7121C5 16 = 0111 0001 0010 0001 1100 0101

The last two bits, 01, identify the word position within the block. 01 means that it is in the second column of data.

The next eight bits, 01110001, should identify the set. 01110001 identifies the set consisting of the third and fourth rows from the bottom.

The fourth row from the bottom has the matching tag of 0111000100100.

Therefore, the data is in the second column of the fourth row from the bottom, i.e., A5.

Tag (binary values)	Set ID (binary values)	Word within block			
		00	01	10	11
10100001001001	01101101	00_{16}	61_{16}	$C2_{16}$	23_{16}
11100001100100	01101101	10_{16}	71_{16}	$D2_{16}$	33_{16}
11001011010110	01101110	20_{16}	81_{16}	$E2_{16}$	43_{16}
11100101101011	01101110	30_{16}	91_{16}	$F2_{16}$	53_{16}
11110110110100	01101111	40_{16}	$A1_{16}$	02_{16}	63_{16}
10100111010101	01101111	50_{16}	$B1_{16}$	12_{16}	73_{16}
10101010111110	01110000	84_{16}	$E5_{16}$	46_{16}	$A7_{16}$
10101010010011	01110000	94_{16}	$F5_{16}$	56_{16}	$B7_{16}$
01110001001000	01110001	$A4_{16}$	$A5_{16}$	66_{16}	$C7_{16}$
00001101101101	01110001	$B4_{16}$	15_{16}	76_{16}	$D7_{16}$
01011010010010	01110010	$C4_{16}$	25_{16}	86_{16}	$E7_{16}$
10101111001011	01110010	$D4_{16}$	35_{16}	96_{16}	$F7_{16}$

b. How many lines are contained in this cache?

(i) 8 (ii) 256 (iii) 512 (iv) 1024 (v) 16K (vi) Cannot be determined

Tag (binary values)	Set ID (binary values)	Word within block			
		00	01	10	11
10100001001001	01101101	00_{16}	61_{16}	$C2_{16}$	23_{16}
11100001100100	01101101	10_{16}	71_{16}	$D2_{16}$	33_{16}
11001011010110	01101110	20_{16}	81_{16}	$E2_{16}$	43_{16}
11100101101011	01101110	30_{16}	91_{16}	$F2_{16}$	53_{16}
11110110110100	01101111	40_{16}	$A1_{16}$	02_{16}	63_{16}
10100111010101	01101111	50_{16}	$B1_{16}$	12_{16}	73_{16}
10101010111110	01110000	84_{16}	$E5_{16}$	46_{16}	$A7_{16}$
10101010010011	01110000	94_{16}	$F5_{16}$	56_{16}	$B7_{16}$
01110001001000	01110001	$A4_{16}$	$A5_{16}$	66_{16}	$C7_{16}$
00001101101101	01110001	$B4_{16}$	15_{16}	76_{16}	$D7_{16}$
01011010010010	01110010	$C4_{16}$	25_{16}	86_{16}	$E7_{16}$
10101111001011	01110010	$D4_{16}$	35_{16}	96_{16}	$F7_{16}$

b. How many lines are contained in this cache?

(i) 8 (ii) 256 (iii) 512 (iv) 1024 (v) 16K (vi) Cannot be determined

Tag (binary values)	Set ID (binary values)	Word within block			
		00	01	10	11
10100001001001	01101101	00_{16}	61_{16}	$C2_{16}$	23_{16}
11100001100100	01101101	10_{16}	71_{16}	$D2_{16}$	33_{16}
11001011010110	01101110	20_{16}	81_{16}	$E2_{16}$	43_{16}
11100101101011	01101110	30_{16}	91_{16}	$F2_{16}$	53_{16}
11110110110100	01101111	40_{16}	$A1_{16}$	02_{16}	63_{16}
10100111010101	01101111	50_{16}	$B1_{16}$	12_{16}	73_{16}
10101010111110	01110000	84_{16}	$E5_{16}$	46_{16}	$A7_{16}$
10101010010011	01110000	94_{16}	$F5_{16}$	56_{16}	$B7_{16}$
01110001001000	01110001	$A4_{16}$	$A5_{16}$	66_{16}	$C7_{16}$
00001101101101	01110001	$B4_{16}$	15_{16}	76_{16}	$D7_{16}$
01011010010010	01110010	$C4_{16}$	25_{16}	86_{16}	$E7_{16}$
10101111001011	01110010	$D4_{16}$	35_{16}	96_{16}	$F7_{16}$

c. How many blocks are contained in the memory space (not the cache, but the memory) of the cache system defined above?
(i) 2^24 (ii) 2^22 (iii) 2^14 (iv) 2^8 (iv) 2^2 (v) Cannot be determined

Tag (binary values)	Set ID (binary values)	Word within block			
		00	01	10	11
10100001001001	01101101	00_{16}	61_{16}	$C2_{16}$	23_{16}
11100001100100	01101101	10_{16}	71_{16}	$D2_{16}$	33_{16}
11001011010110	01101110	20_{16}	81_{16}	$E2_{16}$	43_{16}
11100101101011	01101110	30_{16}	91_{16}	$F2_{16}$	53_{16}
11110110110100	01101111	40_{16}	$A1_{16}$	02_{16}	63_{16}
10100111010101	01101111	50_{16}	$B1_{16}$	12_{16}	73_{16}
10101010111110	01110000	84_{16}	$E5_{16}$	46_{16}	$A7_{16}$
10101010010011	01110000	94_{16}	$F5_{16}$	56_{16}	$B7_{16}$
01110001001000	01110001	$A4_{16}$	$A5_{16}$	66_{16}	$C7_{16}$
00001101101101	01110001	$B4_{16}$	15_{16}	76_{16}	$D7_{16}$
01011010010010	01110010	$C4_{16}$	25_{16}	86_{16}	$E7_{16}$
10101111001011	01110010	$D4_{16}$	35_{16}	96_{16}	$F7_{16}$

c. How many blocks are contained in the memory space (not the cache, but the memory) of the cache system defined above?

(i) 2^24 (ii) 2^22 (iii) 2^14 (iv) 2^8 (iv) 2^2 (v) Cannot be determined

9. Which is the fastest cache mapping function?

(a) Direct mapping (b) Set associative mapping (c) Fully associative mapping

10. Which cache mapping function is least likely to thrash, i.e., it has the lowest chance of two blocks contending with each other to be stored in the same line?

(a) Direct mapping (b) Set associative mapping (c) Fully associative mapping

11. Which cache mapping function does not require a replacement algorithm?

(a) Direct mapping (b) Set associative mapping (c) Fully associative mapping

12. True or false: The number of lines contained in a set associative cache can be calculated from the number of bits in the memory address, the number of bits assigned to the tag, the number of bits assigned to the word id (identifying the number of words per block), and the number of bits assigned to the set id (identifying the number of sets.)

13. Which cache write mechanism creates more bus traffic?

(a) Write through (b) Write back

14. Which cache write mechanism allows an updated memory location in the cache to remain out of date in memory until the block containing the updated memory location is replaced in the cache?

(a) Write through (b) Write back (c) Both (d) Neither

15. Which cache write mechanism best supports bus watching for use with multi-processor systems?

(a) Write through (b) Write back (c) does not make a difference

9. Which is the fastest cache mapping function?

(a) Direct mapping (b) Set associative mapping (c) Fully associative mapping

10. Which cache mapping function is least likely to thrash, i.e., it has the lowest chance of two blocks contending with each other to be stored in the same line?

(a) Direct mapping (b) Set associative mapping (c) Fully associative mapping

11. Which cache mapping function does not require a replacement algorithm?

(a) Direct mapping (b) Set associative mapping (c) Fully associative mapping

12. True or false: The number of lines contained in a set associative cache can be calculated from the number of bits in the memory address, the number of bits assigned to the tag, the number of bits assigned to the word id (identifying the number of words per block), and the number of bits assigned to the set id (identifying the number of sets.)

13. Which cache write mechanism creates more bus traffic?

(a) Write through (b) Write back

14. Which cache write mechanism allows an updated memory location in the cache to remain out of date in memory until the block containing the updated memory location is replaced in the cache?

(a) Write through (b) Write back (c) Both (d) Neither

15. Which cache write mechanism best supports bus watching for use with multi-processor systems?

(a) Write through (b) Write back (c) does not make a difference

16. Suppose a computer has a 4-way set associative cache with one-word blocks. It has a capacity of 256 bytes. Given the sequence of byte addresses 8, 64, 96, 128, 64, 96, 256, 192, 24 show the final cache contents and state the number of hits and misses.

16. Suppose a computer has a 4-way set associative cache with one-word blocks. It has a capacity of 256 bytes. Given the sequence of byte addresses 8, 64, 96, 128, 64, 96, 256, 192, 24 show the final cache contents and state the number of hits and misses.

There are 256/(4*4) = 16 sets. The sequence generates the following table. Empty sets are not shown. The currently accessed cache block is in boldface. The notation m[i] means the word located at memory address i.

byte #	block #	set #		set0	set0	set0	set0	set2	set6	set8
				block0	block1	block2	block3	block0	block0	block0
8	2	2	miss					**m[8]**		
64	16	0	miss	**m[64]**				m[8]		
96	24	8	miss	m[64]				m[8]		**m[96]**
128	32	0	miss	m[64]	**m[128]**			m[8]		m[96]
64	16	0	hit	**m[64]**	m[128]			m[8]		m[96]
96	24	8	hit	m[64]	m[128]			m[8]		**m[96]**
256	64	0	miss	m[64]	m[128]	**m[256]**		m[8]		**m[96]**
192	48	0	miss	m[64]	m[128]	m[256]	m[192]	m[8]		**m[96]**
24	6	6	miss	m[64]	m[128]	m[256]	m[192]	m[8]	**m[24]**	m[96]

17. A machine has a base CPI of 2 clock cycles. Measurements obtained show that the instruction miss rate is 12% and the data miss rate is 6%, and that on average, 30% of all instructions contain one data reference. The miss penalty for the cache is 10 cycles. What is the total CPI?

17. A machine has a base CPI of 2 clock cycles. Measurements obtained show that the instruction miss rate is 12% and the data miss rate is 6%, and that on average, 30% of all instructions contain one data reference. The miss penalty for the cache is 10 cycles. What is the total CPI?

Effective CPI = 2.0 + instruction miss cycles + data miss cycles

= 2.0 + 0.12*10 + 0.30*0.06*10 = 2.0 + 1.2 + 0.18

= 3.38

18. A machine has a 500MHz system clock. Memory takes 30 ns to access a word. How many clock cycles is this?

18. A machine has a 500MHz system clock. Memory takes 30 ns to access a word. How many clock cycles is this?

Clock cycles to access a word
= 30 ns/(1 clock cycle) * (500 * 10^6 clock cycles/sec) * (1 sec/10^9 ns)
= 30 * 500 * 10^6 * 10^-9
= 15000 * 10^-3 = 15 clock cycles

19. A machine with a two level cache has a base CPI of 1.5 when all references hit the primary cache. Given the following characteristics, what is the total CPI?

- a clock rate of 250MHz
- memory access time of 100ns
- miss rate at the primary cache of 5%
- secondary cache access time of 10ns
- miss rate at secondary cache of 1%

19. A machine with a two level cache has a base CPI of 1.5 when all references hit the primary cache. Given the following characteristics, what is the total CPI?

- a clock rate of 250MHz
- memory access time of 100ns
- miss rate at the primary cache of 5%
- secondary cache access time of 10ns
- miss rate at secondary cache of 1%

The clock rate of 250 MHz implies a clock cycle of length 4 ns (because 1/(250*10^6) = 4 * 10^-9.

Therefore the memory access time is 25 cycles and the secondary cache access time is 2.5 cycles.

The effective CPI = 1.5 + cycles missed by primary but caught by secondary + cycles missed by both caches

= 1.5 + (0.05 - 0.01)*2.5 + 0.01*(2.5 + 25)

= 1.5 + 0.1 + 0.275

= 1.875

20. Draw a picture showing the organization of a direct-mapped cache with 16 words per block, with a capacity of 128KB. Show any multiplexers, gates, needed. Show how a 32-bit physical address is mapped to a cache block.

Cache capacity is 128KB. Each block has 16*4 = 64 bytes. There are 128K/64 = 2K = 2048 blocks. The bytes offset is 2 bits, the word offset is 4 bits (16 = 2^4) and the index is 11 bits since 2K = 2^11. That leaves 15 bits for the tag.