

Answer to HW4

7.9

2–miss, 3–miss, 11–miss, 16–miss, 21–miss, 13–miss, 64–miss, 48–miss, 19–miss, 11–hit, 3–miss, 22–miss, 4–miss, 27–miss, 6–miss, 11–set.

Cache set	Address
0000	48
0001	
0010	2
0011	3
0100	4
0101	21
0110	6
0111	
1000	
1001	27
1010	
1011	11
1100	
1101	13
1110	
1111	

7.10

2–miss, 3–hit, 11–miss, 16–miss, 21–miss, 13–miss, 64–miss, 48–miss, 19–miss, 11–hit, 3–miss, 22–hit, 4–miss, 27–miss, 6–hit, 11–miss

Cache set	Address
00	[0, 1, 2, 3]
01	[4, 5, 6, 7]
10	[8, 9, 10, 11]
11	[12, 13, 14, 15]

7.14

The miss penalty is the time to transfer one block from main memory to the cache. Assume that it takes 1 clock cycle to send the address to the main memory.

a. Configuration (a) requires 16 main memory accesses to retrieve a cache block, and words of the block are transferred 1 at a time.

$$\text{Miss penalty} = 1 + 16 \cdot 10 + 16 \cdot 1 = 177 \text{ clock cycles.}$$

b. Configuration (b) requires 4 main memory accesses to retrieve a cache block and words of the block are transferred 4 at a time.

$$\text{Miss penalty} = 1 + 4 \cdot 10 + 4 \cdot 1 = 45 \text{ clock cycles.}$$

c. Configuration (c) requires 4 main memory accesses to retrieve a cache block, and words of the block are transferred 1 at a time.

$$\text{Miss penalty} = 1 + 4 \cdot 10 + 16 \cdot 1 = 57 \text{ clock cycles}$$

7.19

Execution time = Clock cycle·IC·(CPI_{base}+ Cache miss cycles per instruction)

Execution time_{original} = 2·IC·(2 + 1.5·20·0.05) = 7 IC

Execution time_{new} = 2.4·IC·(2 + 1.5·20·0.03) = 6.96 IC

Hence, doubling the cache size to improve miss rate at the expense of stretching the clock cycle results in essentially no net gain.

7.29

Address size:	k bits
Cache size:	S bytes/cache
Block size:	$B = 2^b$ bytes/block
Associativity:	A blocks/set

Number of sets in the cache:

$$\begin{aligned} \text{Sets/cache} &= \frac{(\text{Bytes/cache})}{(\text{Blocks/set}) \times (\text{Bytes/block})} \\ &= \frac{S}{AB} \end{aligned}$$

Number of address bits needed to index a particular set of the cache:

$$\begin{aligned} \text{Cache set index bits} &= \log_2 (\text{Sets/cache}) \\ &= \log_2 \left(\frac{S}{AB} \right) \\ &= \log_2 \left(\frac{S}{A} \right) - b \end{aligned}$$

Number of bits needed to implement the cache:

Tag address bits/block = (Total address bits) – (Cache set index bits)

$$\begin{aligned} & - (\text{Block offset bits}) \\ &= k - \left(\log_2 \left(\frac{S}{A} \right) - b \right) - b \\ &= k - \log_2 \left(\frac{S}{A} \right) \end{aligned}$$

Number of bits needed to implement the cache = sets/cache × associativity × (data + tag + valid):

$$\begin{aligned} &= \frac{S}{AB} \times A \times \left(8 \times B + k - \log_2 \left(\frac{S}{A} \right) + 1 \right) \\ &= \frac{S}{B} \times \left(8B + k - \log_2 \left(\frac{S}{A} \right) + 1 \right) \end{aligned}$$

7.41

The TLB will have a high miss rate because it can only access 64 KB (16 · 4 KB) directly. Performance could be improved by increasing the page size if the architecture allows it.

7.42

Virtual memory can be used to mark the heap and stack as read and write only. In the above example, the hardware will not execute the malicious instructions because the stack memory locations are marked as read and write only and not execute.

7.45

Less memory—fewer compulsory misses. (Note that while you might assume that capacity misses would also decrease, both capacity and conflict misses could increase or decrease based on the locality changes in the rewritten program. There is not enough information to definitively state the effect on capacity and conflict misses.)

Increased clock rate—in general there is no effect on the miss numbers; instead, the miss penalty increases. (Note for advanced students: since memory access timing would be accelerated, there could be secondary effects on the miss numbers if the hardware implements prefetch or early restart.)

Increased associativity—fewer conflict misses.

8.18

For 4-word block transfers, the bus bandwidth was 71.11 MB/sec. For 16-word block transfers, the bus bandwidth was 224.56 MB/sec. The disk drive has a transfer rate of 50 MB/sec. Thus for 4-word blocks we could sustain $71/50 = 1$ simultaneous disk transfers, and for 16-word blocks we could sustain $224/50 = 4$ simultaneous disk transfers. The number of simultaneous disk transfers is inherently an integer and we want the sustainable value. Thus, we take the floor of the quotient of bus bandwidth divided by disk transfer rate.

8.19

For the 4-word block transfers, each block now takes

- 1) 1 cycle to send an address to memory
- 2) $150 \text{ ns} / 5 \text{ ns} = 30$ cycles to read memory
- 3) 2 cycles to send the data
- 4) 2 idle cycles between transfers

This is a total of 35 cycles, so the total transfer takes $35 \cdot 64 = 2240$ cycles. Modifying the calculations in the example, we have a latency of 11,200 ns, 5.71M transactions/second, and a bus bandwidth of 91.43 MB/sec.

For the 16-word block transfers, each block now takes

- 1) 1 cycle to send an address to memory
- 2) 150 ns or 30 cycles to read memory
- 3) 2 cycles to send the data
- 4) 4 idle cycles between transfers, during which the read of the next block is completed

Each of the next two remaining 4-word blocks requires repeating the last two steps. The last 4-word block needs only 2 idle cycles before the next bus transfer. This is a total of $1 + 20 + 3 \cdot (2 + 4) + (2 + 2) = 53$ cycles, so the transfer takes $53 \cdot 16 = 848$ cycles. We now have a latency of 4240 ns, 3.77M transactions/second, and a bus bandwidth of 241.5 MB/sec.

Note that the bandwidth for the larger block size is only 2.64 times higher given the new read times. This is because the 30 ns for subsequent reads results in fewer opportunities for overlap, and the larger block size performs (relatively) worse in this situation.

8.20

The key advantage would be that a single transaction takes only 45 cycles, as compared with 57 cycles for the larger block size. If because of poor locality we were not able to make use of the extra data brought in, it might make sense to go with a smaller block size. Said again, the example assumes we want to access 256 words of data, and clearly larger block sizes will be better. (If it could support it, we'd like to do a single 256-word transaction!)

8.21

Assume that only the 4-word reads described in the example are provided by the memory system, even if fewer than 4 words remain when transferring a block. Then,

	Block size (words)												
	4	5	6	7	8	9	10	11	12	13	14	15	16
Number of 4-word transfers to send the block	1	2	2	2	2	3	3	3	3	4	4	4	4
Time to send address to memory (bus cycles)	1	1	1	1	1	1	1	1	1	1	1	1	1
Time to read first 4 words in memory (bus cycles)	40	40	40	40	40	40	40	40	40	40	40	40	40
Block transfer time, at 2 transfer bus cycles and 2 idle bus cycles per 4-word transfer (bus cycles)	4	8	8	8	8	12	12	12	12	16	16	16	16
Total time to transfer one block (bus cycles)	45	49	49	49	49	53	53	53	53	57	57	57	57
Number of bus transactions to read 256 words using the given block size	64	52	43	37	32	29	26	24	22	20	19	18	16
Time for 256-word transfer (bus cycles)	2880	2548	2107	1813	1568	1537	1378	1272	1166	1140	1083	1026	912
Latency (ns)	14400	12740	10535	9065	7840	7625	6890	6360	5830	5700	5415	5130	4560
Number of bus transactions (millions per second)	4.444	4.082	4.082	4.082	4.082	3.774	3.774	3.774	3.774	3.509	3.509	3.509	3.509
Bandwidth (MB/sec)	71.1	80.4	97.2	113.0	130.6	133.2	148.6	161.0	175.6	179.6	189.1	199.6	224.6