Homework #8

Due 11:59pm Monday, 21 November, 2016
Multiple submissions accepted.
Late homeworks are dinged 25% per full or partial day.

5.1 In this exercise we look at memory locality properties of matrix computation. The following code is written in C, where elements within the same row are stored contiguously. Assume each word is a 32-bit integer.

```
for (I=0; I<8; I++)
     for (J=0; J<8000; J++)
          A[I][J] = B[I][0] + A[J][I]
```

5.1.1 [5] <§5.1> How many 32-bit integers can be stored in a 16-byte cache block? Hint: don't worry about any of the extra stuff that has to accompany the integer like valid bits or tags, etc. Assume the memory is all for your data.

5.1.2 [5] <§5.1> References to which variables exhibit temporal locality?

5.1.3 [5] <§5.1> References to which variables exhibit spatial locality?
Hint: choose one from among `I`, `J`, `A[I][J]`, `B[I][0]`, and `A[J][I]`.

Locality is affected by both the reference order and data layout. The same computation can also be written below in Matlab, which differs from C by storing matrix elements within the same **column** contiguously in memory.

```
for I=1:8
     for J=1:8000
          A(I,J) = B(I,0) + A(J,I);
     end
end
```

5.1.4 [10] <§5.1> How many 16-byte cache blocks are needed to store all 32-bit matrix elements being referenced?
Hint: I assumed the that the entries of A on the left and right of the equal sign could be shared, so I needed space for A and for the much smaller B. You can fit four 32-bit elements in a 16-byte block.

5.1.5 [5] <§5.1> References to which variables exhibit temporal locality?

5.1.6 [5] <§5.1> References to which variables exhibit spatial locality?

©Kevin B Long. Portions ©Elsevier Inc. All rights reserved.

5.2 Caches are important to providing a high-performance memory hierarchy to processors. Below is a list of 32-bit memory address references, given as word addresses.

```
3, 180, 43, 2, 191, 88, 190, 14, 181, 44, 186, 253
```

5.2.1 [10] <§5.3> For each of these references, identify the binary address, the tag, and the index given a direct-mapped cache with 16 one-word blocks. Also list if each reference is a hit or a miss, assuming the cache is initially empty.

Hint: use the table below for your answers. Remember the Tag refers to the block or chunk of data from which the data element in the cache came. The Index tells you which row of the cache it's in.

Word Address	Binary Address	Tag	Index	Hit/Miss
3				
180				
43				
2				
191				
88				
190				
14				
181				
44				
186				
253				

5.2.2 [10] <§5.3> For each of these references, identify the binary address, the tag, and the index given a direct-mapped cache with two-word blocks and a total size of 8 blocks. Also list if each reference is a hit or a miss, assuming the cache is initially empty.

Hint: use this table again. A two-word block loads two words at the same time. Here, though, the Tag points to the start of a set of two adjacent words in memory that are being loaded into the cache. So if you load word n, you'll be loading word n+1 or n-1 along with it, whichever you need to make sure that you start off that 2-word block with even one (word 0, 2, 4, and so on). So you'll have a hit if either your intended word or its mate are requested.

©Kevin B Long. Portions ©Elsevier Inc. All rights reserved.

Word Address	Binary Address	Tag	Index	Hit/Miss
3				
180				
43				
2				
191				
88				
190				
14				
181				
44				
186				
253				

5.2.3 [20] <§§5.3, 5.4> 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 8 words of data: C1 has 1-word blocks, C2 has 2-word blocks, and C3 has 4-word blocks. In terms of miss rate, which cache design is the best? If the miss stall time is 25 cycles, and C1 has an access time of 2 cycles, C2 takes 3 cycles, and C3 takes 5 cycles, which is the best cache design?

Hint: Use the table below to map out your index numbers and hit/miss rate. This is just an extension of the prior exercise. When you have an n-word block, you always load all n words back to the nearest multiple of n. For example, if asked to find whether word 3 is present in a cache with 8-word blocks, you know that if it's a hit, you will find words 0 through 7 in that block, which of course includes 3.

Word Address	Binary Address	Tag	Cache 1		Cache 2		Cache 3	
			Index	Hit/Miss	Index	Hit/Miss	Index	Hit/Miss
3								
180								
43								
2								
191								
88								
190								
14								
181								

©Kevin B Long. Portions ©Elsevier Inc. All rights reserved.

44								
186								
253								

Once you have the hits and misses, calculate the miss rate for each cache design as a % of all accesses. All misses is 100% miss rate.

Then calculate total cycles per cache design. Use # misses x 25 cycles per miss to capture the stall time, but don't forget to add in that cache's access cycle which is per access.

	Cache 1	Cache 2	Cache 3
Miss rate:			
Total Cycles:			

Circle the winner

The following parameters apply to problems 5.2.4, .5, and .6:
There are many different design parameters that are important to a cache's overall performance. Below are listed parameters for different direct-mapped cache designs.

Cache Data Size: 32 KiB

Cache Block Size: 2 words

Cache Access Time: 1 cycle

5.2.4 [15] <§5.3> Calculate the total number of bits required for the cache listed above, assuming a 32-bit address. Given that total size, find the total size of the closest direct-mapped cache with 16-word blocks of equal size or greater. Explain why the second cache, despite its larger data size, might provide slower performance than the first cache.

Hint: Assume you have a 32KiB cache to start with. Figure out how many total blocks you have, and how many bits you are using for your index. Remember that the rest are all for your tag, less any you can ignore for word size or words per block. You must always add a valid bit to each block as well.

Changing from 2-word to 16-word blocks changes your tag size.

5.2.5 [20] <§§5.3, 5.4> Generate a series of read requests that have a lower miss rate on a 2 KiB 2-way set associative cache than the cache listed above. Identify one possible solution that would make the cache listed have an equal or lower miss rate than the 2 KiB cache. Discuss the advantages and disadvantages of such a solution.

Hint: It's all about entries with the same index field but with tag fields that change or not.

©Kevin B Long. Portions ©Elsevier Inc. All rights reserved.

5.2.6 [15] <§5.3> The formula shown in Section 5.3 shows the typical method to index a direct-mapped cache, specifically (Block address) modulo (Number of blocks in the cache). Assuming a 32-bit address and 1024 blocks in the cache, consider a different indexing function, specifically (Block address [31:27] XOR Block address [26:22]). Is it possible to use this to index a direct-mapped cache? If so, explain why and discuss any changes that might need to be made to the cache. If it is not possible, explain why.

5.3 For a direct-mapped cache design with a 32-bit address, the following bits of the address are used to access the cache.

Tag	Index	Offset
31-10	9-5	4-0

5.3.1 [5] <§5.3> What is the cache block size (in words)?
Hint: The offset bits are those which are simply carried around the cache and added back to the final address if needed to handle a fault. The cache doesn't bother storing them. Since the 4 bytes within a word account for 2 of those bits, the rest are ignored because every combination of them is part of the cache's data entry.

5.3.2 [5] <§5.3> How many entries does the cache have?
Hint: the tag bits are used to see if what is in the indexed row to which you've been pointed is the particular block you wanted, so it's not the tag bits.

5.3.3 [5] <§5.3> What is the ratio between total bits required for such a cache implementation over the data storage bits?
Hint: Ignore the presence of a valid bit. They didn't specify that it had one, so we'll assume it does not. Page 390 walks through an example, "Bits in a Cache." The one confusing thing to me was the formula, which I'll retype but with annotations here for you: 32 bits in an address that could be used for a tag – 10 bits that must be used to index to narrow down to which block in the cache we're dealing with – 2 bits for there being 4 words in a block – 2 bits for there being 4 bytes in a word + 1 valid bit.

Starting from power on, the following byte-addressed cache references are recorded.

Address											
0	4	16	132	232	160	1024	30	140	3100	180	2180

5.3.4 [10] <§5.3> How many blocks are replaced?

5.3.5 [10] <§5.3> What is the hit ratio?

©Kevin B Long. Portions ©Elsevier Inc. All rights reserved.

5.3.6 [20] <§5.3> List the final state of the cache, with each valid entry represented as a record of <index, tag, data>.

Index	Tag	Tag

5.7 This exercise examines the impact of different cache designs, specifically comparing associative caches to the direct-mapped caches from Section 5.4. For these exercises, refer to the address stream shown in Exercise 5.2.

5.7.1 [10] <§5.4> Using the sequence of references from Exercise 5.2, show the final cache contents for a three-way set associative cache with two-word blocks and a total size of 24 words. Use LRU replacement. For each reference identify the index bits, the tag bits, the block offset bits, and if it is a hit or a miss.

5.7.2 [10] <§5.4> Using the references from Exercise 5.2, show the final cache contents for a fully-associative cache with one-word blocks and a total size of 8 words. Use LRU replacement. For each reference identify the index bits, the tag bits, and if it is a hit or a miss.

5.7.3 [15] <§5.4> Using the references from Exercise 5.2, what is the miss rate for a fully-associative cache with two-word blocks and a total size of 8 words, using LRU replacement? What is the miss rate using MRU (most recently used) replacement? Finally what is the best possible miss rate for this cache, given any replacement policy?

Multilevel caching is an important technique to overcome the limited amount of space that a first-level cache can provide while still maintaining its speed. Consider a processor with the following parameters:

©Kevin B Long. Portions ©Elsevier Inc. All rights reserved.

Base CPI, No Memory Stalls	Processor Speed	Main Memory Access Time	First-Level Cache Miss Rate per Instruction	Second-Level Cache, Direct-Mapped Speed	Global Miss Rate with Second-Level Cache, Direct-Mapped	Second-Level Cache, Eight-Way Set Associative Speed	Global Miss Rate with Second Level Cache, Eight-Way Set Associative
1.5	2 GHz	100 ns	7%	12 cycles	3.50%	28 cycles	1.50%

5.7.4 [10] <§5.4> Calculate the CPI for the processor in the table using: 1) only a first-level cache, 2) a second-level direct-mapped cache, and 3) a second-level eight-way set-associative cache. How do these numbers change if main memory access time is doubled? If it is cut in half?

5.7.5 [10] <§5.4> It is possible to have an even greater cache hierarchy than two levels. Given the processor above with a second-level, direct-mapped cache, a designer wants to add a third-level cache that takes 50 cycles to access and will reduce the global miss rate to 1.3%. Would this provide better performance? In general, what are the advantages and disadvantages of adding a third-level cache?

5.7.6 [20] <§5.4> In older processors such as the Intel Pentium or Alpha 21264, the second level of cache was external (located on a different chip) from the main processor and the first-level cache. While this allowed for large second-level caches, the latency to access the cache was much higher, and the bandwidth was typically lower because the second-level cache ran at a lower frequency. Assume a 512 KiB off-chip second-level cache has a global miss rate of 4%. If each additional 512 KiB of cache lowered global miss rates by 0.7%, and the cache had a total access time a 50 cycles, how big would the cache have to be to match the performance of the second-level direct-mapped cache listed above? Of the eight-way set-associative cache?

5.9 This Exercise examines the single-error-correcting, double-error-detecting (SEC/DED) Hamming code.

5.9.1 [5] <§5.5> What is the minimum number of parity bits required to protect a 128-bit word using the SEC/DED code?

5.9.2 [5] <§5.5> Section 5.5 states that modern server memory modules (DIMMs) employ SEC/DED ECC to protect each 4 bits with 8 parity bits. Compute the cost/performance ratio of this code to the code from 5.9.1. In this case, cost is the relative number of parity bits needed while performance is the relative number of errors that can be corrected. Which is better?

©Kevin B Long. Portions ©Elsevier Inc. All rights reserved.

5.9.3 Consider an SEC code that protects 8-bit words with 4 parity bits. If we read the value 0x375 is there an error? If so, correct it.

Hint: use the bit numbering from section 5.5, where you number bit from the left starting at 1 (unlike everything else we've learned). Use this table to fill out the bits so you can tell the parity bits apart from the data bits. You'll need to follow the steps in the example problem on page 420-21 of the book.

Bit Position		1	2	3	4	5	6	7	8	9	10	11	12
Encoded data bits		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8
Parity bit coverate	p1												
	p2												
	p3												
	p4												

Extra problem:
Read the extra reading for HW #7 that is alongside the homework assignment in Blackboard, entitled "ErrorCorrectionAndDetectionSupplement.pdf".

Do the following problems from that reading:
1. A 12-bit Hamming code word containing 8 bits of data and 4 parity bits is read from memory. What was the original 8-bit data word that was written into memory if the 12-bit word read out is:

(a) 010011111000 (b) 011101010010 (c) 010000000101

4. A modified single-error-correcting, double-error-detecting Hamming code for four bits of data D3 , D5 , D6 , and D7 has the following parity bit equations:

$$P1 = D3 \oplus D5 \oplus D6$$
$$P2 = D3 \oplus D5 \oplus D7$$
$$P4 = D3 \oplus D6 \oplus D7$$
$$P8 = D5 \oplus D6 \oplus D7$$

(a) Find the binary values of the four check bits for a single error in each of the eight bit positions of the code.

©Kevin B Long. Portions ©Elsevier Inc. All rights reserved.