CS161 - Homework 4
DUE Wednesday, June 5th, 2019 IN CLASS!

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];
```

1.1 How many 32-bit integers can be stored in a 16-byte cache block?

1.2 References to which variables exhibit temporal locality?

1.3 References to which variables exhibit spatial locality?

2. Caches are important to providing a high-performance memory hierarchy to processors. Below is a list of 10-bit memory address references in a Byte-addressable system. Assume each word is a 32-bit integer.

12, 720, 172, 8, 764, 352, 760, 56, 724, 176, 744, 1012

2.1 For each of these references, identify 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.

2.2 For each of these references, identify 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.

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 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?

There are many different design parameters that are important to a cache's overall performance. Below are listed parameters for a direct-mapped cache design:

Cache Data Size: 32 KiB
Cache Block Size: 2 words
Cache Access Time: 1 cycle

2.4 Assume we have a byte-addressable memory. 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.

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

3.1 What is the cache block size (in words)?

3.2 How many entries does the cache have?

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

3.3 How many blocks are replaced?

3.4 What is the hit ratio?

3.5 List the final state of the cache, with each valid entry represented as a record of <index, tag, data>.

4. This exercise examines the impact of different cache designs, specifically comparing associative caches to the direct-mapped caches. For these exercises, refer to the address stream shown in Exercise 2.

12, 720, 172, 8, 764, 352, 760, 56, 724, 176, 744, 1012

4.1 Using the sequence of references from Exercise 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.

4.2 Using the references from Exercise 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.

4.3 Using the references from Exercise 2, what is the miss rate for a fully associative cache with two-word blocks and a total size of 8 entries, 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?

5. In this exercise, we will examine space/time optimizations for page tables. The following list provides parameters of a virtual memory system.

Virtual Address (bits)	Physical DRAM Installed	Page Size	PTE Size (byte)
43	16 GiB	4 KiB	4

5.1 For a single-level page table, how many page table entries (PTEs) are needed? How much physical memory is needed for storing the page table?

5.2 Using a multilevel page table can reduce the physical memory consumption of page tables, by only keeping active PTEs in physical memory. How many levels of page tables will be needed in this case? And how many memory references are needed for address translation if missing in TLB?

The following table shows the contents of a 4-entry TLB (in the protection field, R means read, W means write, and X means execute).

Entry-ID	Valid	VA Page	Modified	Protection	PA Page
1	1	140	1	RW	30
2	0	40	0	RX	34
3	1	200	1	RO	32
4	1	280	0	RW	31

5.3 Under what scenarios would entry 3's valid bit be set to zero?

5.4 What happens when an instruction writes to VA page 30?

5.5 What happens when an instruction writes to VA page 200?