15-213: Introduction to Computer Systems
Written Assignment #4

This written homework covers the Memory Hierarchy and Cache Memories.

Directions

Complete the question(s) on the following pages with single paragraph answers. These questions are not meant to be particularly long! Once you are done, submit this assignment on Canvas.

Below is an example question and answer.

Q: Please describe benefits of two's-complement signed integers versus other approaches.

A: Other representations of signed integers (ones-complement and sign-and-magnitude) have two representations of zero (+0 and −0), which makes testing for a zero result more difficult. Also, addition and subtraction of two's complement signed numbers are done exactly the same as addition and subtraction of unsigned numbers (with wraparound on overflow), which means a CPU can use the same hardware and machine instructions for both.

Grading

Assignments are graded via *peer review:*

1. Three other students will each provide short, constructive feedback on your assignment, and a score on a scale of 1 to . You will receive the maximum of the three scores.
1. You, in turn, will provide feedback and a score for three other students (not the same ones as in part 1). We will provide a *rubric*, a document describing what good answers to each question look like, to assist you. You receive five additional points for completing all of your peer reviews.

Due Date

This assignment is due on July 6 by 11:59pm Pittsburgh time (currently UTC−4). Remember to convert this time to the timezone you currently reside in.

Peer reviews will be assigned roughly 12 hours later, and are due a week after that.

Question 1

a) Explain one thing that cache can do but main memory can't and one thing that main memory can do but a cache can't. If comparing speed or size, be sure to give approximate quantities.

b) Explain one thing that main memory can do but local disks can't and one thing that local disks can do but main memory can't. If comparing speed or size, be sure to give approximate quantities.

If a student only mentions the big idea above the bullet points, that is partial credit. If a student uses one of the bullet points (or any other valid example of an advantage), they receive the full points for the problem

The main benefit of cache over main memory is that you can access data much quicker.

- Caches are built to be small and easy to access
- They are often on or very close to the CPU chip
- They take in the tens of cycles
- Sizes on the order of kilobytes.

The main benefit of main memory over cache is that it is much cheaper and can store more data.

- It is about 100th the cost as cache memory.
- This leads it to have sizes in the tens of GB.
- Furthermore, main memory does not have to deal with concurrency issues caused by different cores having different threads.
- Lastly, main memory is often upgradable whereas caches are mostly a part of whatever chip they are used in.

The main benefit of main memory over local disks is that you can access data much quicker.

- It is on the order of 100 cycles (~10 ms) as opposed to 100,000 (~100 ms).
- Furthermore, programs only run from main memory so files need to be loaded from the local disk

The main benefit of the local disk is that it can store more data than main memory.

- Local disks are on the scale of terabytes.
- Furthermore, main memory is only available while the computer is on. When the computer is turned off, all information stored in main memory is lost. Therefore, to save work, it must be stored on the local disk (or some local disk on a remote system).

Question 2

How does a write-back cache handle a write hit differently than a write-through cache? What are the pros and cons of a write-back cache? What are the pros and cons of a write-through cache?

For a write-hit and write-back implementation, a cache does not update the value at the address provided for lower levels of the cache until the element is evicted from the upper-level cache.

For a write-hit and write-through implementation, a cache will immediately change the value stored at the provided address space for lower levels of the cache.

A pro of write-back is that it reduces the number of accesses to lower levels of memory, which makes the program faster and more efficient. A con is that there is more complexity with an additional dirty bit that has to be checked whenever an element is evicted.

A pro of write-through is all values in the cache are correct and up to date. A con of write-through is that it takes time to change the lower values of the cache for every write-hit.

Question 3

Consider a cache of the following configurations

Total Cache Size = 1024 bytes
Block size = 32 bytes
Number of sets = 8
16 bit address space

A. How many block, set, and tag bits are there?
B. How many lines are there per set?
C. Memory is accessed at the address 0x1A23. Write out the tag, set and block bits.
D. Next, memory is accessed at 0x1B32. Is this a hit or a miss? If it is a miss, what kind, and does it evict the other line from the cache? If it is a hit, what kind of locality does it share with the other line?
E. Following this, the next byte of data is accessed with address 0x1B33. Is this a hit or a miss? If it is a miss, what kind, and does it evict the other line from the cache? If it is a hit, what kind of locality does it share with the other line?
F. Finally, the programmer begins a program which loops, accessing 33 distinct places in memory, none of which share a cache block. What kind of misses does this pattern create? How can you differentiate this from any other kinds of misses discussed in (D) or (E)?

Solution:

a. Block = log_2 32 = 5 bits, set = log_2 8 = 3, tag = 16 - (5 + 3) = 8
b. There are 4 lines per set, since 1024 / (8 * 32) = 4
c. Block = 0x3, Set = 0x1, tag = 0x1A
d. This is a miss - the breakdown for this address is Block = 0x12, Set = 0x1, tag = 0x1B. Since the tag and set bits are not the same, this is a miss; on top of that, it is a cold (compulsory) miss since this is the first time this line is being accessed. Since there are four lines per set, even though these two addresses map to the same set, the first one will **not** be evicted since there is still space.
e. At a glance, the set and tag bits will be exactly the same; this is in the same block as the piece of memory accessed in part (d), so this is a hit, as that entire block will still be in the cache. This hit leverages spatial locality - even though this particular block has not been accessed previously, a nearby one was.
f. This creates capacity misses, since a 1024 byte cache with 32 byte blocks can only hold 1024/32 = 32 blocks in the cache. You can differentiate this from the cold miss in part (d) by noticing that these are blocks which have been in the cache before, but were evicted because the cache was full - in other words, this was a "preventable" miss.