15213 - Lecture 12 - POGIL Activity (Caching)

Introduction

In this activity you will learn about caches. Some of this material is based on ideas from peerinstruction4cs.org.

Before you begin, take a minute to assign roles to each member in your group. Try to switch up the roles as much as possible: please don't pick a role for yourself that you have already done more than once. Below is a summary of the four roles; write the name of the person taking that role next to the summary.

If your group only has three members, combine the roles of Facilitator and Process Analyst.

For this and all future activities, the Facilitator should also take the role of the reader and read the questions aloud to the group.

- **Facilitator**: Reads question aloud; keeps track of time and makes sure everyone contributes appropriately.
- **Quality Control**: Records all answers & questions, and provides team reflection to team & instructor.
- **Spokesperson**: Talks to the instructor and other teams. Compiles and runs programs when applicable.
- **Process Analyst**: Considers how the team could work and learn more effectively.

Fill in the following table showing which group member is performing each role:

Role	**Person**
Facilitator	
Quality Control	
Spokesperson	
Process Analyst	

Model 1: Caches

As you proceed through college, you will collect more and more notes and textbooks. You can store them in three places: your backpack, your dorm room, and your parent's house (or other home away from CMU).

1. When you are in class today, which place is the most convienent? Where is the least convienent?
2. Which place can hold the most notes and books?
3. When you leave your dorm room, how do you pick what books and notes to take with you?

4. At the end of the semester, how might you change your storage of your notes and textbooks?

Model 2: Lookup

When a program tries to access memory, before the request goes onto the bus. It is passed through one or more caches first. Each cache must quickly answer the question: does it have the data?

1. If the `array` below starts at (32-bit) address `0x00 00 FF 00`, what is the address of the last element?

   ```
   int array[20];
   ```

2. Given an address, which bits best capture spatial locality?

The following chart shows the diversity of address bit values as recorded in several benchmarks. If the bit is always 0 or always 1, then its diversity value is 0.

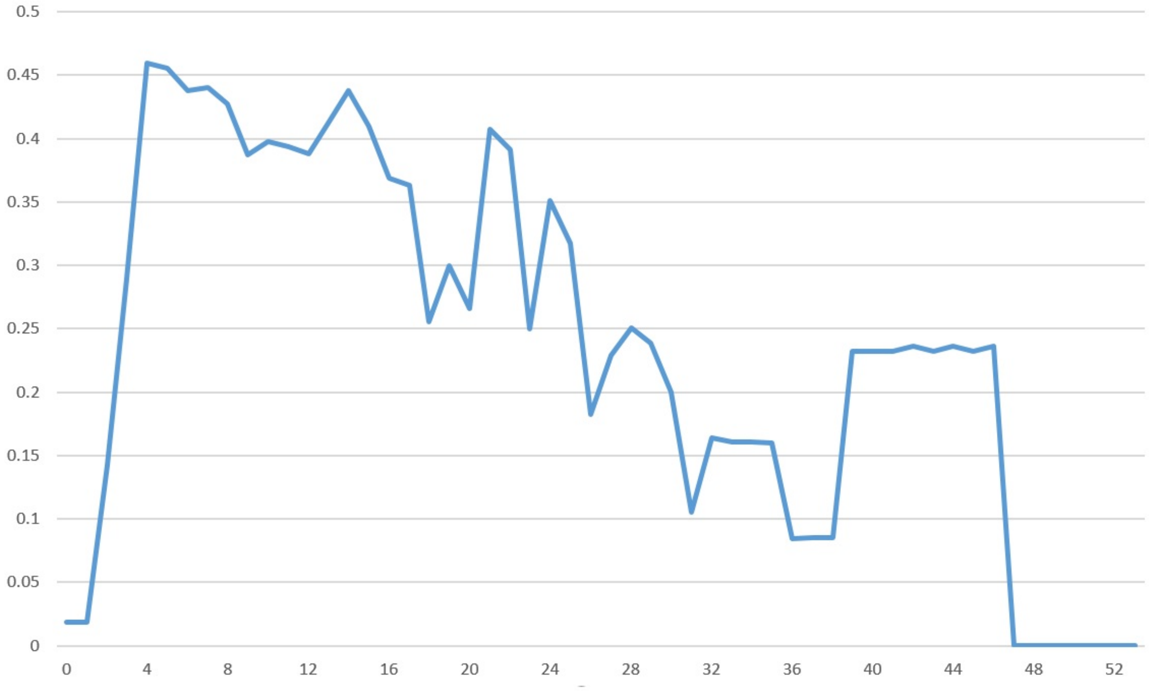

Figure 1: Bit Diversity

3. Approximately which bit has the most diversity? Excluding the left and right edges of the plot, which bit has the least?

4. Draw a line from the most to the least. How does this seem to represent the trend in bit diversity?

5. Which bits in this plot capture spatial locality?

6. A cache is very similar to a hash table. As we have learned (in Sec 5.14 among other places), we want hash values to be across a large range. Caches use a very simple hashing algorithm of selecting a subset of the address bits. Discuss which range(s) you might use.

Model 3: Hardware

We can describe a cache using three numbers: S, E, B. The following figure shows how the first two relate to the cache organization.

1. The hashing bits from the previous model are actually the set index bits. How many bits are required for this purpose?

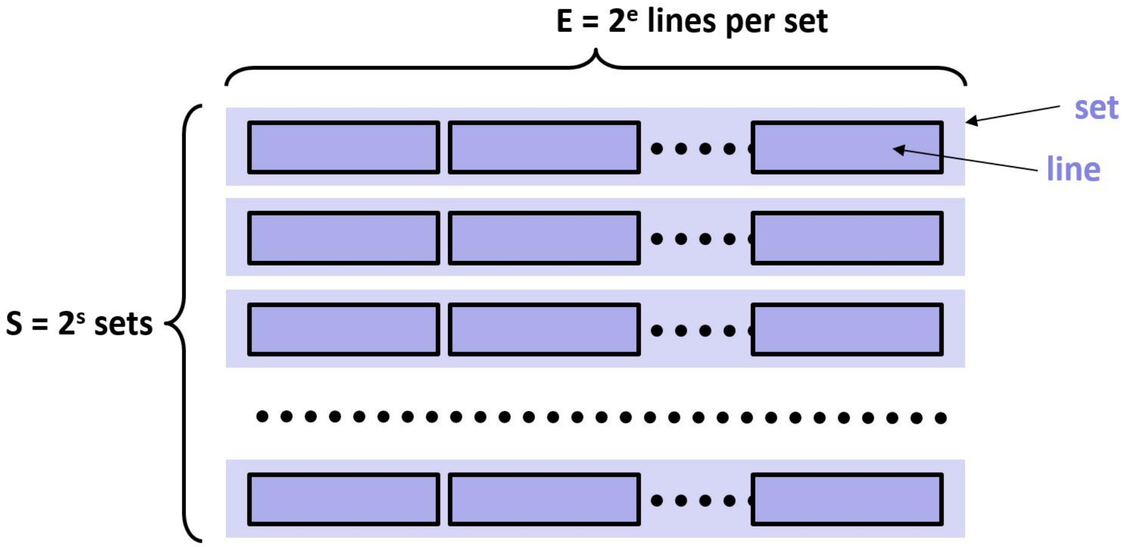

Figure 2: Cache

2. Cache lines contain blocks of data. The spatial locality bits are the block offset, which select the byte(s) from the block to return. How many bits are required for this purpose?

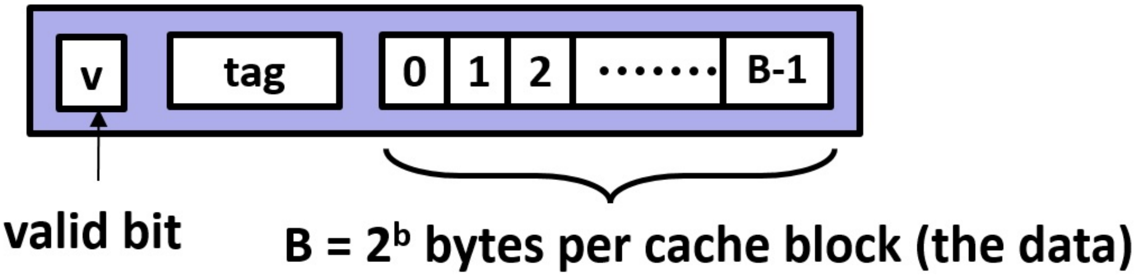

Figure 3: Line

3. On selecting a cache using the index bits, the cache must search (simultaneuously as this is hardware) all of its lines to see if any match the address. Each check is done using the remaining bits of the address. These bits are the tag. If m is the number of address bits, how many bits are in the tag (t)?

Cache lines also contain a valid bit to store if this line is valid.

Sets in caches will have one or more lines. If there is only one line, that cache is termed 'direct mapped'. If there is only one set, then the cache is termed 'fully associative'. Otherwise, the cache is considered E-way associative.

4. Associativity has trade-offs. Remember that hardware costs space and energy. Between direct mapped and fully associative. Select the one that:
   - Requires the most hardware
   - Can determine if a cache line is present fastest
   - Is more likely to have the data present
5. There are three basic types of cache misses. We also rate caches for their miss rates, which are $misses/accesses$. When an address is accessed for the very first time, where is it?
6. The program accesses `array[0]`, `array[1]`, `array[2]`, and `array[3]` (see Model 2). How large should B be for there to be (ideally) only 1 miss?
7. We discussed before that associativity has trade-offs in cache miss rates. If changing the associativity improves the miss rate, then the negated misses are conflict misses. With m = 8, B = 4, S = 2, compute whether the following accesses hit or miss in the two caches.

Address	E = 1	E = 2
0x80	miss	
0x40	miss	
0x20	miss	
0x40		
0x80		

8. Would increasing E further have any change on the cache miss rate?
9. Suppose there is a simple program and cache. The cache only holds 1 line. The program alternates between accessing two blocks. From 0, 1/4, 1/2, 3/4, and 1, what is the likely miss rate?

Model 4: Replacement

Althought not all college students use them, and one of my roommates certainly did not, but let's consider shirts hanging in a closet. In this scenario, your closet has limited space and you also have a box under your bed to store additional clothes.

1. Your closet is full and you have received a new shirt. How do you make room in your closet for this shirt?

- Pick randomly
- Box up the newest shirt
- Box up your oldest shirt
- Box up a shirt you haven't worn in a while
- I use a different policy to replace my shirts (seasonal, etc)

2. For the previous question, what additional information would you need to:

- Make the decision
- Determine whether the decision is best

When a cache access misses, the new adress and its associated line is added to the appropriate set. Caches do not "rehash" addresses on a collision, instead the cache must select a line to replace in that set. Similarly, every cache line also maintains additional data in order to determine which cache line to replace.

3. As caches are trying to exploit temporal and spatial locality to mitigate the memory wall (i.e., the vast difference in CPU cycles and memory access time), Here is the access stream going to a set, where E = 2. Using the common replacement policy, least recently used, mark which accesses will hit and which will miss:

   ```
   A, B, A, C, B, C, A
   ```

4. Could a different replacement algorithm have done better? If so, how?
5. When a cache line is replaced, it is termed evicted. What should the cache do with the evicted line?

Model 5: Writing to Cache Lines

1. When a processor writes to a memory location, from where does the data come?
2. When a processor writes to a cache line present, the access hits in the cache. However, there is the question of what to do with the written data. Should the cache immediately update memory? Suppose that `i` is stored in memory in the following code, is there locality in the writes?

   ```
   for (int i = 0; i < (1000 * 1000); i++)
   ```

3. If the cache caches the written values, then the cache is write back (WB). If it immediately updates memory, then it is write through (WT). In a write back cache, what must the cache do in when a cache line is evicted? Does the cache line need additional information?
4. If the cache line was not already present for an address, shoud the cache store the written data and allocate a cache line for this address?
5. Given the two common pairings: WBWA and WTWNA, when might it be advantageous to use each?