CSE291 Lab 1

Discussion 1:
1: Caches in modern processors are generally set-associative. Use the commands above and what you learned in “Computer Architecture Fundamentals” to fill in the blanks in the following table. The L1 line size has been filled in for you.

	Cache Line Size	Total Size	Associativity	Number of Sets	Raw Latency
L1	64	32KB	8	64	4 cycles
L2	64	256KB	8	512	12 cycles
L3	64	8MB	16	8192	~30 cycles(avg)

Latency sources:
https://www.7-cpu.com/cpu/IvyBridge.html
https://uops.info/cache.html

Discussion 2:
How many cache lines do you need to access to fill up the entire L2 cache and L3 cache respectively? You can derive the answer from Table 1.

L2: 4096 Cache Lines
L3: 131072 Cache Lines

Discussion 3:
Based on the generated histogram, report two thresholds, one to distinguish between L2 and L3 latency and the other to distinguish between L3 and DRAM latency.

L2/L3 Threshold: ~50 cycles

L3/DRAM Threshold: ~200 cycles

Eviction Set Construction:
This is a discussion on how to construct an eviction set in the L2 cache. Since the same mechanism is used in single-bit as well as 8 bit channels, this is stated here and will be applicable for the next 2 questions.

To construct an eviction set, we need to find addresses which map to a particular set in the number of the associativity. In our L2 cache, the associativity is 8, so we need to construct a set of 8 addresses for a particular set number. Since the caches are virtually indexed, we need the 9 LSBs after ignoring the first 6 LSBs. We ignore the first 6 as they indicate the block offset. The next 9 indicate the index of the L2 cache. We compare these 9 bits to the set number and decide that such an address would map to this cache set. We then iteratively run through a

buffer which is the size of the L2 cache and find 8 addresses which would map to a single cache set. For the 8 bit channel, we do this same process for 8 different set indexes and compare.

Discussion 4:
Describe your communication protocol to communicate a single bit value.

The code is present in the branch “one_bit_channel”.
Repository link: https://github.com/gdpande97/SecProcsFa22

The logic for one bit is simpler than the 8 bit logic. In this the sender and receiver monitor a particular set. The sender constructs an eviction set of size 8 which is the number of ways in the L2 cache. The addresses mapped here are accessed by the sender when the bit is 1 and not accessed with the bit is 0.

The receiver on its end also creates an eviction set which has 8 addresses mapped to the same set. The receiver primes all the addresses in the cache by accessing them before the sender. Once the sender sends the bit, the receiver tries to access the sets again and check if the access time is higher than before. This then transmits whether the bit was 0 or 1.

Discussion 6:
Describe your communication protocol. Please refer to Section 2.2 for the components involved in a protocol.

The code is present in main branch.
Repository link: https://github.com/gdpande97/SecProcsFa22

Sender:
Since the two processes are running on the same processor, the L2 should be shared, which will be useful to communicate through the covert channel. For an 8-bit communication protocol, I start off by choosing 8 random sets from the L2 cache. I also allocate a buffer the size of the L2 cache. This buffer is then accessed to find addresses in the buffer which would map to the sets chosen before. This creates my eviction set.
After this is done, the sender goes into a while loop, where an input is asked interactively, which will be the message to be sent to the receiver. The sender converts the input character into a binary representation. This binary representation is then sent bit by bit.

To send each bit, the set at the position of the corresponding bit is accessed by the sender if the bit is 1 and not accessed if it is 0. This is accessed for all the ways of that particular set number.

Receiver:

For the receiver, the code runs based on prime+probe. The receiver uses the same sets used by the sender. First the receiver also finds the addresses to construct an eviction set which will

map to those sets and all their ways. Once this is constructed, the receiver accesses all the addresses in that eviction set, priming the cache.

Once this is primed, it waits for the user input to start probing the cache. This user input is given once the sender has sent a character to transmit. Once received, the receiver starts timing his accesses to all the addresses in the eviction set and tries to see if there's a particular pattern in the access times. In case the bit has been 1, it would be accessed by the sender, which would mean that the latency would be more than the access latency of bits 0, which were not accessed by the sender. This can be deciphered into a binary message which can then be reconstructed as a string.