CS773 (Jan-April 2025), 120 minutes, March 2, 2025

Q1 to 10 [20 minutes]:
True/False with proper justifications will lead to full marks
True/False without justifications will lead to ZERO marks
Justifications without clearly specifying True/False will lead to ZERO marks

1. Cache-based timing channel attacks can be mitigated if an ISA provides secure instructions to access a cache line where a particular cache line can be accessed by a particular process only.
   False

2. A fully associative cache mitigates cache occupancy attacks.
   False

3. KPTI mitigates Spectre variant II and uses out-of-order execution.
   False

4. Flush-based attacks can be mitigated if we disable the flushing of shared but writable data.
   False

5. Eviction set creation is a necessary evil for cache occupancy-based attacks.
   False

6. Set-based cache partitioning is more scalable than way-based partitioning techniques.
   True

7. Meltdown attack exploits speculative execution to dump the kernel memory into the cache.
   False

8. ASLR and KASLR prevent Spectre variant-I but not variant-II.
   False

9. Delay-based approaches need changes in the processor pipeline, whereas invisible speculation-based approaches do not change the processor pipeline while mitigating Spectre-like attacks.
   True

10. A transient instruction will fail to transmit speculative information during a Spectre attack if the branch predictor accuracy is 100% and the BTB hit rate is 100%.
    True

Take a break. Take a kitkat (sprite maybe). Clflush your stress as the real exam begins in a few nanoseconds.

The devil is in the details, so answer in detail. Answers without sufficient details will lead to zero marks.

Bouncer from Shoaib [20 minutes]

----

11. Shoaib is designing a single-core processor that uses the GhostMinion Cache hierarchy. It has 1024 entries in ROB. The processor runs only one program in a data center processor and generates only one kind of LOAD from a for loop that accesses a large integer array sequentially. One integer is four bytes. As per GhostMinion, an instruction is safe once it leaves/retires from ROB. What should be the size of GhostMinion Cache if, on average, an instruction takes 128 cycles to retire after it is done with its execution? You can assume only 25% of the instructions are LOADs. STOREs perform lazy writes to the cache hierarchy on commit/retire. You can also assume the cache line size of GhostMinion Cache is 64 bytes. Please provide an average size and worst-case size of the GhostMinion data cache. **[5 marks]**

As discussed in the class 1024/4 loads = 256 entries X 64 bytes in the worst case
One cache line can store 16 integer loads, so 1024/64 = 16 entries X 64 bytes in the average case

What a catch (cache) by Sweta [30 minutes]

----

12. As a student of CS773, Sweta wants to reverse engineer a new processor designed by Mriganko and Ishan. The goal is reverse engineering the cache hierarchy with the following two codes. The array data contains 32-bit unsigned integer values. For simplicity, we consider that accesses to the array latency bypass all caches (i.e., latency is not cached). timer() returns a timestamp in cycles.
    Code-I

```
-----------------
j = 0;
for (i=0; i<size; i+=stride){
start = timer();
d = data[i];
stop = timer();
latency[j++] = stop - start;
}
```

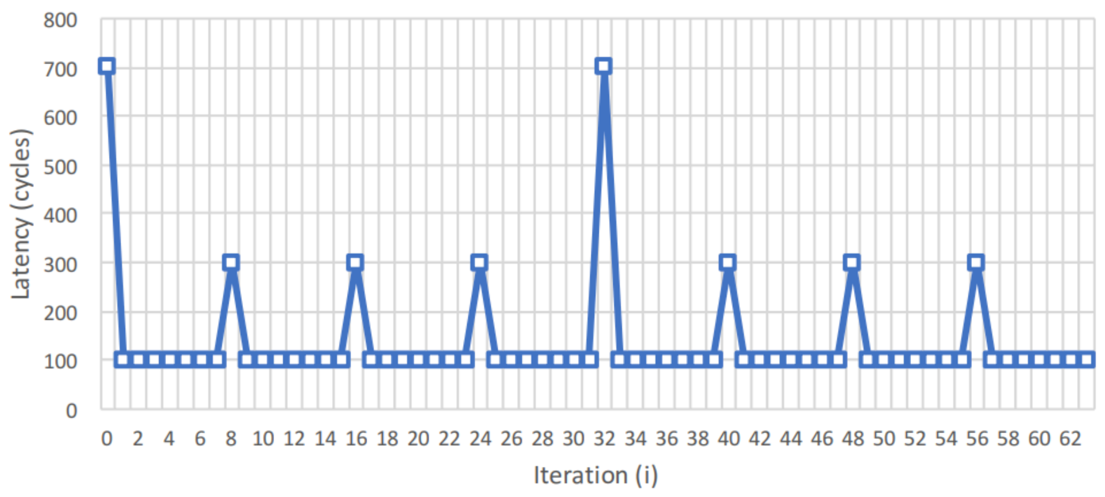
    Code-II

```
-------------------
for (i=0; i<size1; i+=stride1){
d = data[i];
}
j = 0;
for (i=0; i<size2; i+=stride2){
start = timer();
d = data[i];
stop = timer();
latency[j++] = stop - start;
}
```

Mriganko did a lscpu to realize that the cache hierarchy has two levels. L1 is a 4kB set associative cache.

(a) When we run code-I, we obtain the latency values in the following chart for the first 64 reads to the array data (in the first 64 iterations of the loop) with a stride equal to 1. What are the cache block sizes in L1 and L2? **[3 marks]**

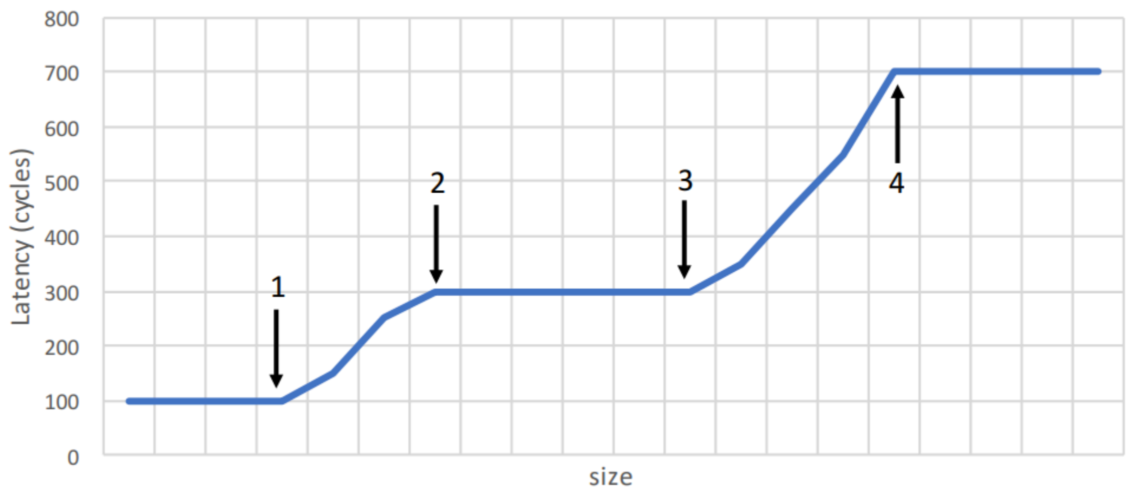
L1 cache block size is 32 bytes, and L2 cache block size is 128 bytes.



(b) Using code-II with stride1 = stride2 = 32, size1 = 1056, and size2 = 1024, we observe latency[0]= 300 cycles. However, if size1 = 1024, latency[0] = 100 cycles. What is the maximum number of ways in L1? (Note: The replacement policy can be either FIFO or LRU) **[3 marks]**

The maximum number of ways is 32.

(c) Now, we carry out two consecutive runs of code-I for different size values. In the first run, stride is equal to 1. In the second run, stride is equal to 16. We ignore the latency results of the first run and average the latency results of the second run. We obtain the following graph. What do the four parts shown with the arrows represent? **[4 marks]**



Before arrow 1:
In arrow 1 the size of the array is the same as the size of L1 (4kB).
Between arrow one and arrow two:
Some accesses in the second run hit in L1 and other accesses hit in L2.
Between arrow two and arrow three:
All accesses in the second run hit in L2.
Between arrow three and arrow four:
Still some accesses in the second run hit in L2. Arrow 3 marks the size of the L2 cache.
After arrow four:
The accesses of the second run always miss in L2, and it is necessary to access main memory.

Kalind goes down the ground, Kalind goes out of the ground [20 minutes]
----
13. Kalind wants to design a cache attack detector capable of performing conflict-based cache attacks and Spectre attacks through cache conflicts. His detector should be lightweight and agile and not use ML/AI techniques. Please specify the *necessary and sufficient conditions* to detect attacks assuming two application domains and then *argue how exactly you fulfill the necessary and sufficient conditions.* You can take the help of OS and microarchitecture events, but you are not supposed to assume any support from the compiler or runtime system.
[5 marks]

A simple detector should be tracking cross-core conflicts between two processes, applications, etc. One way conflict wont be enough. There should be cyclic interference between process a to b and then again from b to a.

CS773-midterm finish in style [30 minutes]

----

14. After attending the first ten lectures of CS773, a bunch of UGs and PGs decided to design a secure cache hierarchy that secures Spectre, Meltdown, and all three kinds of cache attacks at the L1, L2, and L3 caches. The L1 and L2 caches are shared by two SMT threads of an SMT-based processor. The L3 cache is a 16-way 32 MB cache shared by 32 cores. Anubhav met the UGs and PGs and suggested that the best way to design a secure cache hierarchy is to design a heterogeneous solution with different mitigation techniques at different levels of cache. The mitigation techniques discussed in the class so far are cache partitioning (set, way, coloring) techniques, randomization-based techniques, and delay and invisible speculation-based techniques. Please clearly specify the design choices you will make while designing a secure L1 cache, L2 cache, and L3 cache, keeping all the above-mentioned attacks in mind and keeping performance and security tradeoffs in mind. The key is the performance-security tradeoff. Clearly explain how you mitigate all the attacks at each level with your design choice. **[10 marks]**

L1: partitioned as only two threads share it, randomization will be costly in terms of performance as it will affect hit latency
L2: partitioned as only two threads share it, randomization will be costly in terms of performance as it will affect hit latency
L3: partitioning not scalable as 32 cores sharing the L3. so a randomized cache is better for mitigating all except occupancy attacks. For occupancy attack you can use core id along with the randomized cache to create isolated and randomized regions.

On top of these ideas, we need ideas like GhostMinion at all the levels to make sure Spectre and Meltdown does not happen.

If you have liked the course, draw a small heart on the chalkboard before leaving the exam hall [0 marks, 2 minutes, only gratitude no CPI effect :)]