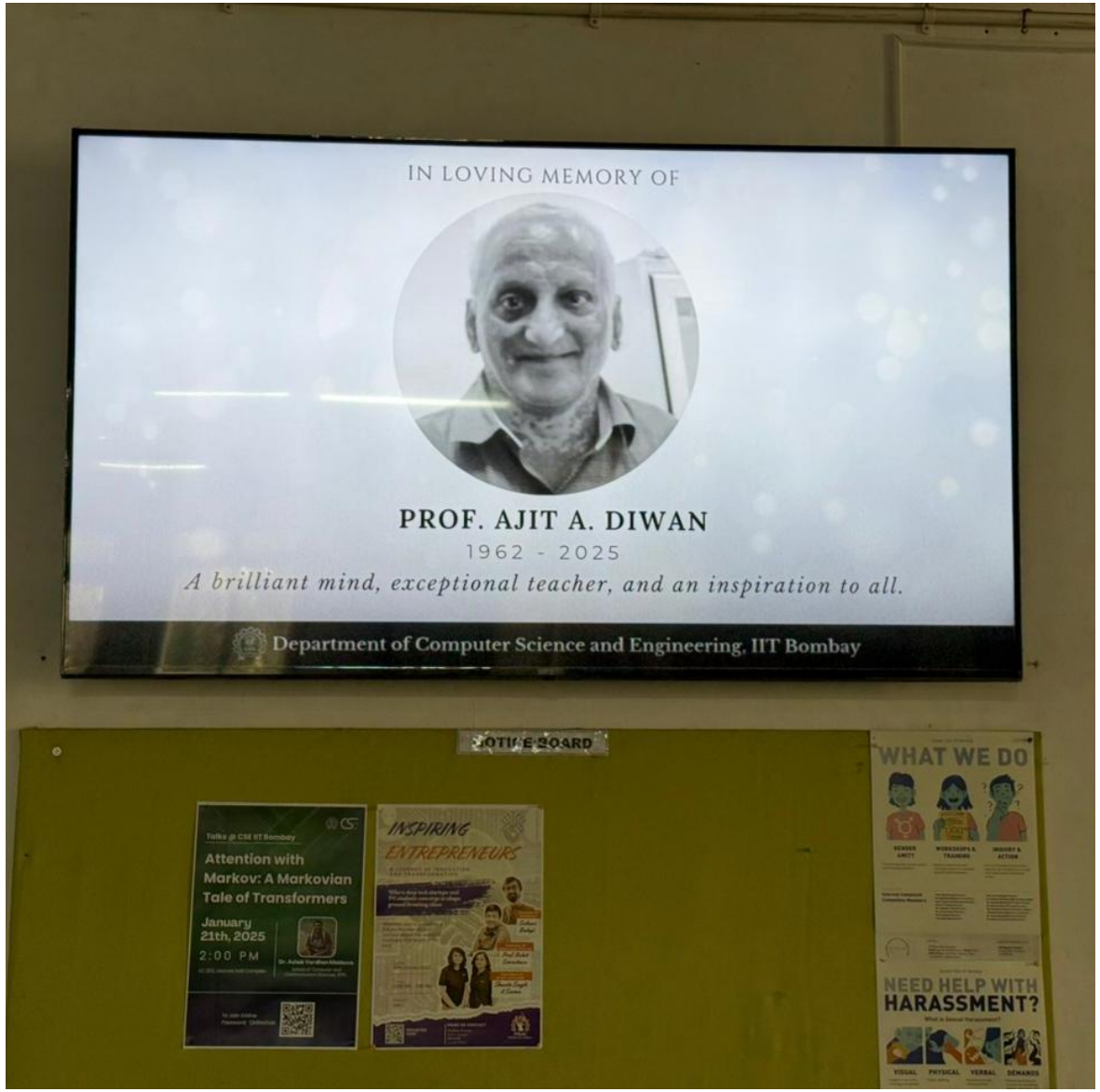
CASPER

CS773-2025-Spring: Computer Architecture for Performance and Security

Lecture 4: Flush it Again

https://www.cse.iitb.ac.in/~biswa/

Prof. Ajit Diwan



CASPER

Logistics

- Plan A to Plan B and vice versa is still possible after assignment-I or during assignment-I
- Once you move to plan B, we will stop grading for plan A and nullify all that you have done as part of plan A.
- Spend some time in the next few weeks thinking about what you can do. Meet me, and discuss this in detail.

More Logistics

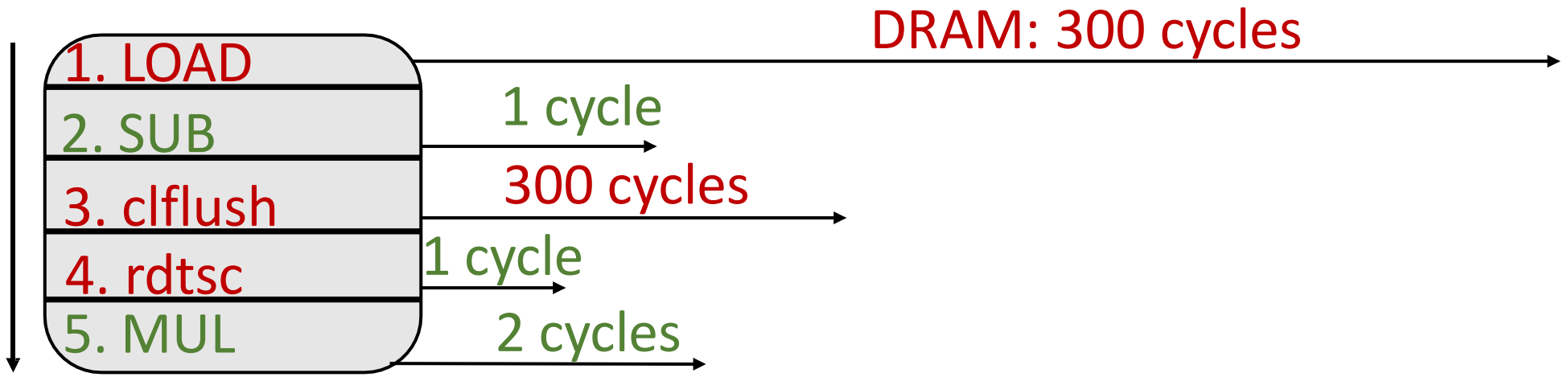
- Hands-on Help Session on 27th Jan: 7PM to 8:30 PM
- Protocol: We will release Assignment 1.2 this Thursday. Go through it completely and come with your queries/hiccups/questions for TAs and me. Attendance compulsory. You have to come with your laptop. We will float a Google form asking for your signature. We will evaluate your assignments only if you have submitted the Google form.
- Post 27th, if you ask basic questions that are resolved, then we may/may not reply to all the queries.
- Quiz-1: Feb 3rd, 7 PM to 8:30 PM

Security vs Privacy

- Security: Person A communicates (sends data) to Person B. Person C should not be able to see it. This is the confidentiality aspect of Security.

- Privacy: Hiding the information about who is A and who is B is privacy.

The O3 effect (See the F+R code)



Out-of-order execution ☹
(Multiple fetch in one cycle)

rdtsc

```
uint64_t rdtsc_nofence() {
uint64_t a, d;
asm volatile ("rdtsc" : "=a" (a), "=d" (d));
a = (d<<32) | a;
return a;
}
```

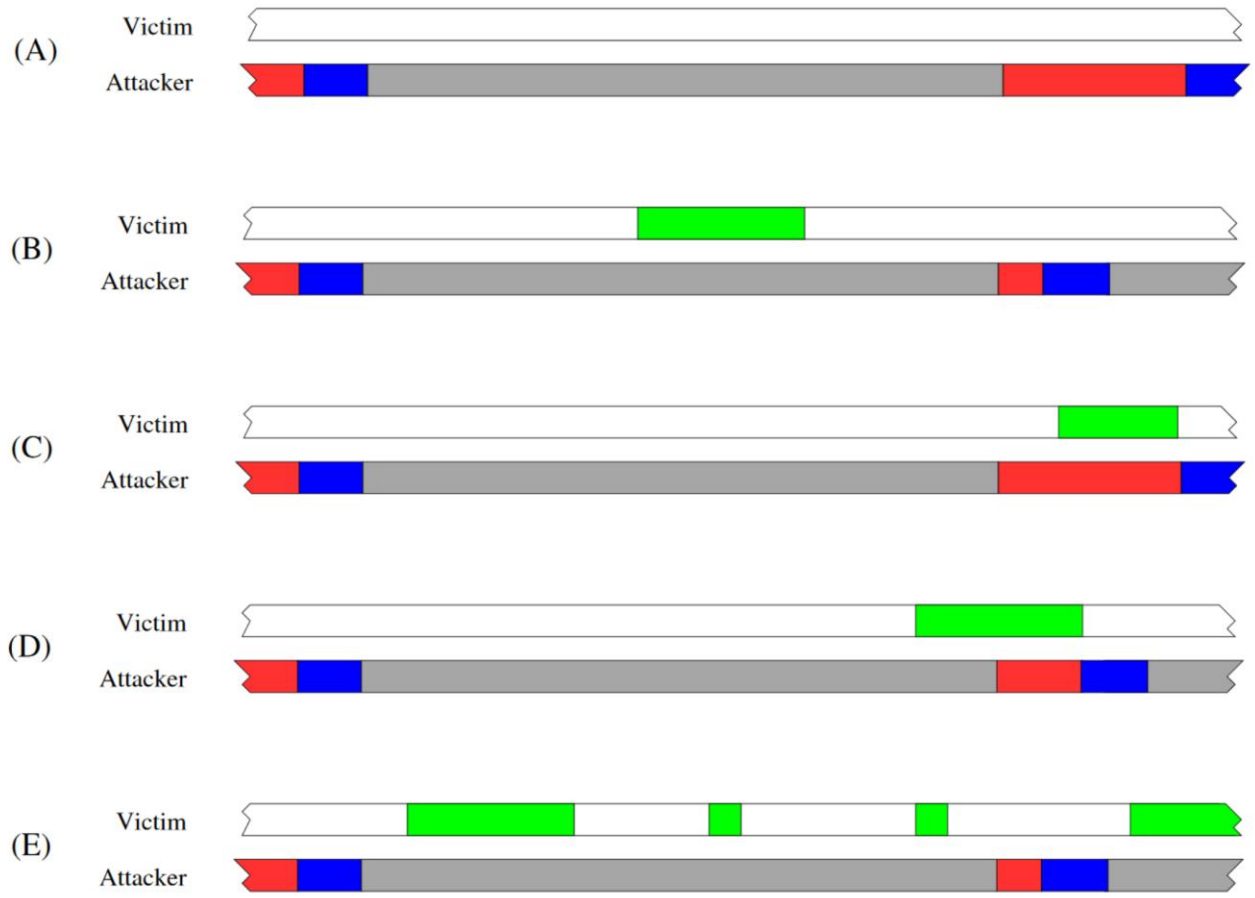
asm: To include assembly code in C/C++
rdtsc registers the clock ticks passed since the last reset, 64 bit values stored as 32-bits in EDX (upper half) and EAX (lower half)
volatile: makes sure compiler does not do anything stupid ☺

rdtsc with fence

```
uint64_t rdtsc() {
uint64_t a, d;
asm volatile ("mfence");
asm volatile ("rdtsc" : "=a" (a), "=d" (d));
a = (d<<32) | a;
asm volatile ("mfence");
return a;
}
```

There are lfence, sfence too.

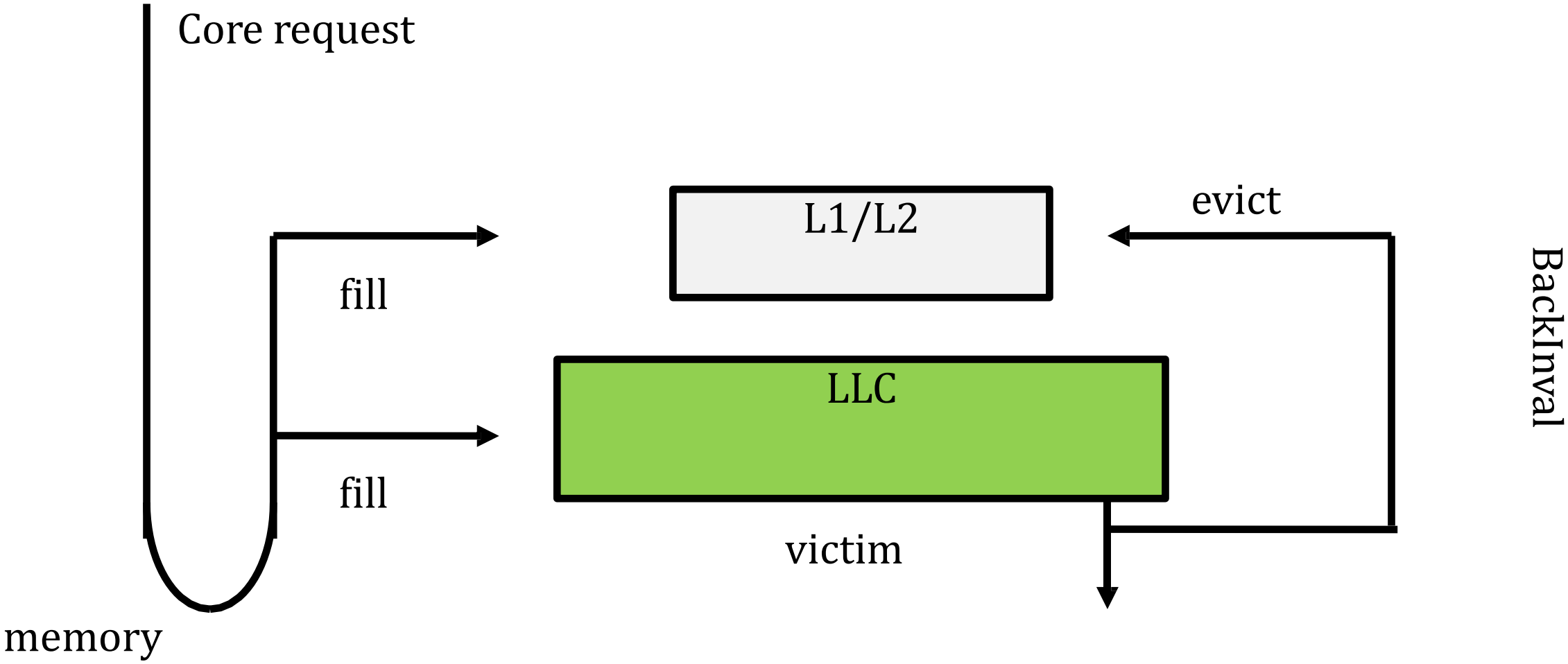
The notion of Time (an agreement in covert channel)



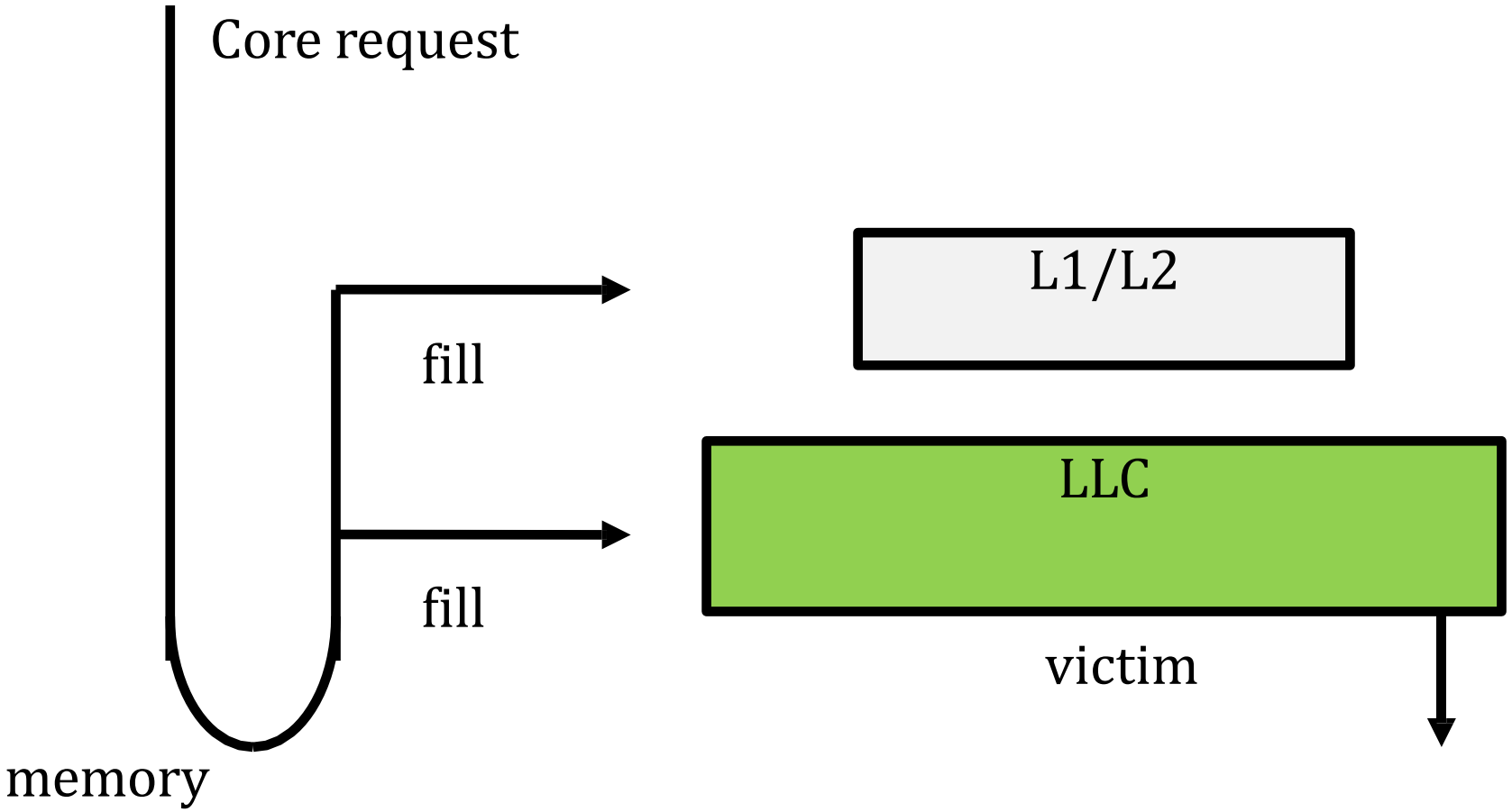
Flush attacks are unaffected by cache hierarchies.

What about Cache hierarchy effects?

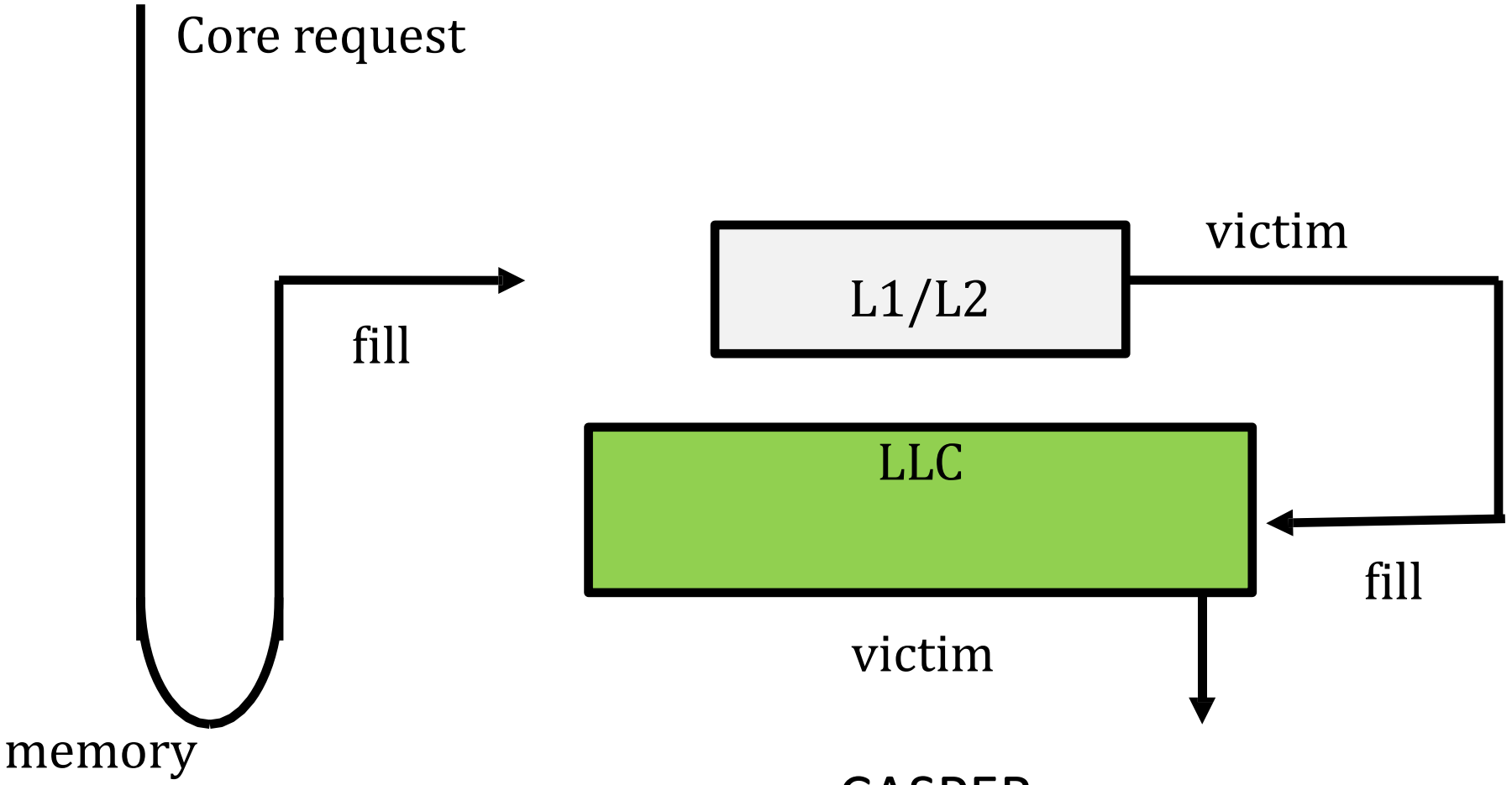
Inclusive Cache Hierarchy



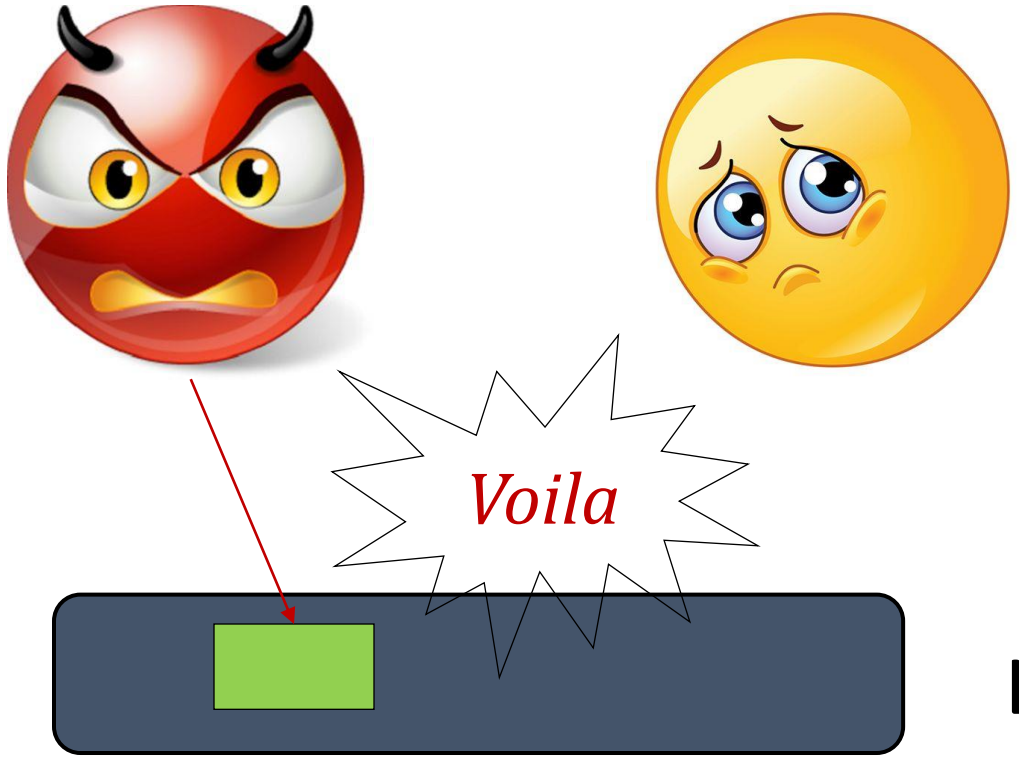
Non-inclusive (many commercial machines)



Exclusive hierarchy



Flush&Flush attack [how does it work]



Step 0:Spy *maps* the shared library, shared in the cache

Step 1:Spy *flushes* the cache block

Step 2: Victim *reloads* the cache block

Step 3: Spy *flushes* the cache block again

Confused?

CASPER

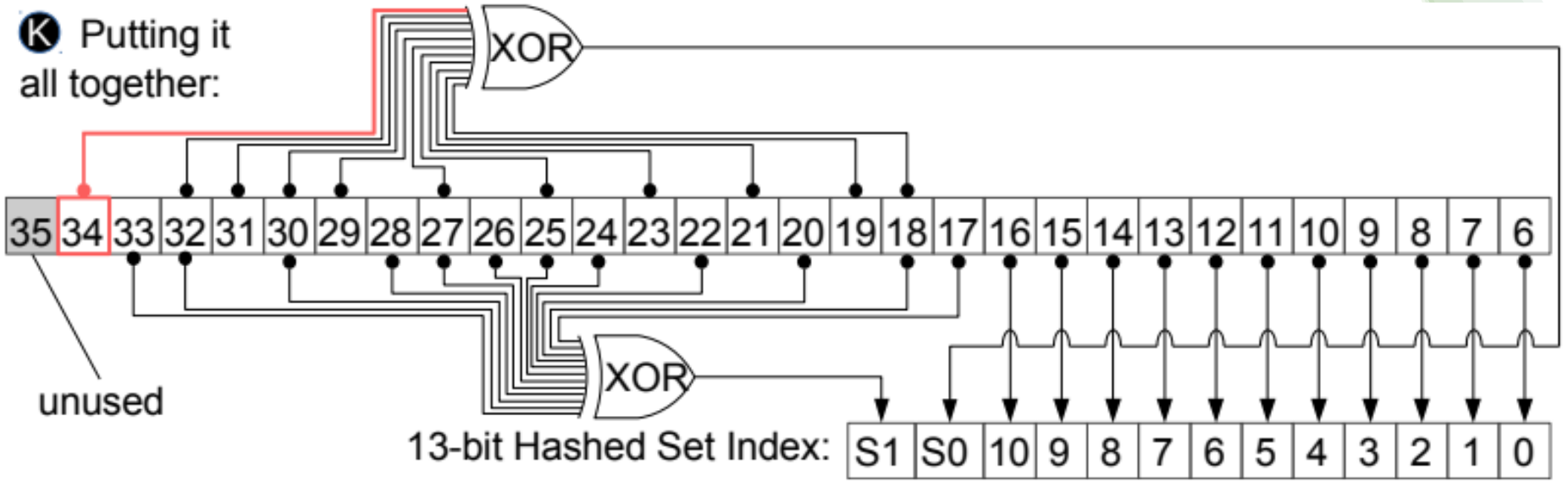
Clflush

On a hit at the LLC, clflush has to flush L1+L2 of the victim too

On a miss at the LLC, do nothing (faster than hits)

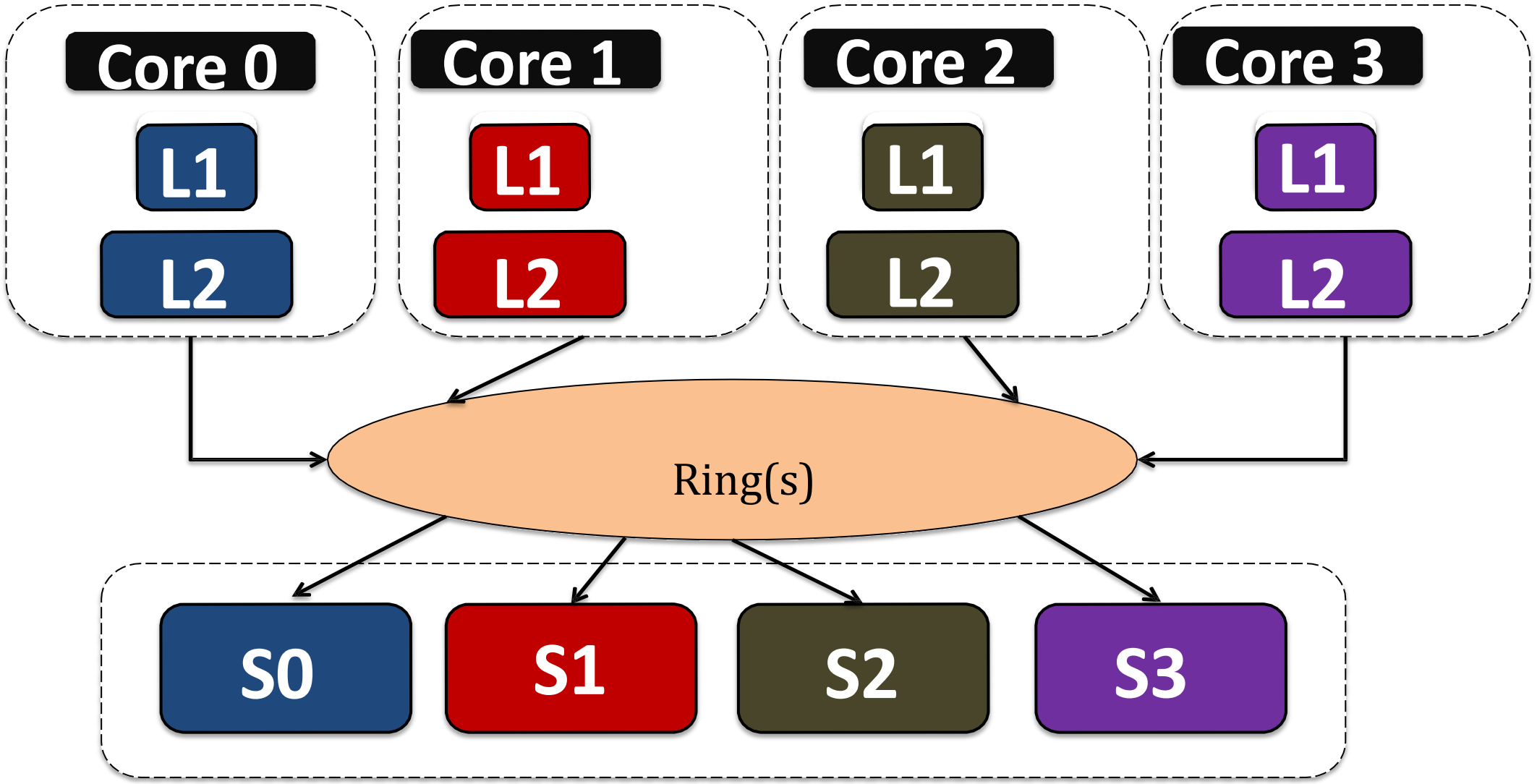
What if no clflush but still sharing is ON?

- Evict+Reload attack
- Evict the cache line address(es) from a cache set.
- Reload the set again with the set of addresses that you sent in the previous step.
- Difference in latency will reveal key
- Challenges: Need to know the cache set number at which the location is mapped.
- Need to reverse-engineer LLC indexing (address to set mapping)



Intel uses the following Hashing (Indexing)

World of Sliced Caches make it even more difficult



CASPER

Job of an attacker

CALIBRATION; FOR LATENCY THRESHOLD

FIND OUT ADDRESSES OF INTEREST

BITS OF INTEREST

CASPER

How good is the attacker

Bandwidth
synchronous/asynchronous

Accuracy (error rate)

Stealthy and agility

CASPER