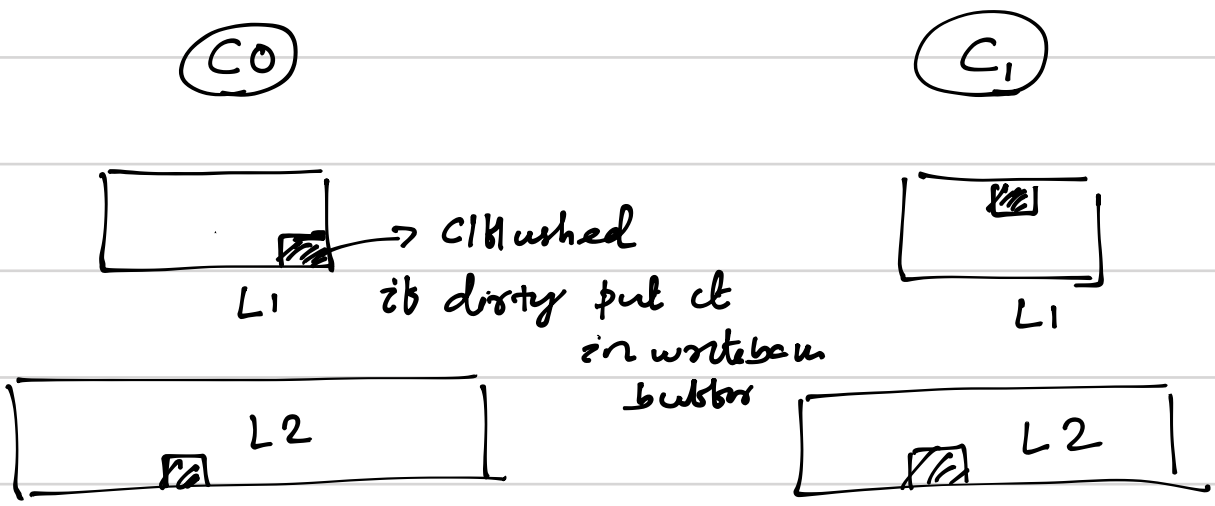
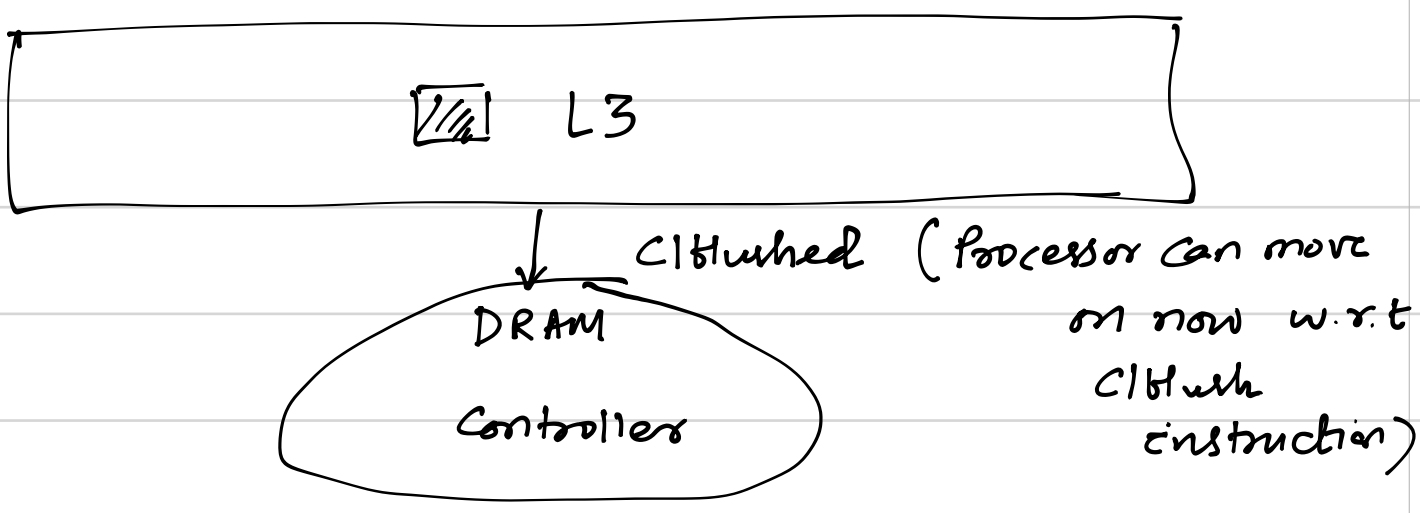
flush based cache attacks

* Usage of clflush. clflush (virtual address)
* clflush flushes (invalidates) cache line corresponding to the virtual address from all levels of caches and from all the cores.
*





* Strengths:-
  - highly precise & fine-grained
    - (i) Works at all the levels of caches
    - (ii) Independent of inclusivity, cache size, block size, replacement policy
    - (iii) clflush is not a privileged instruction.
* Weakness

(i) demands page sharing. clflush won't be effective if the address space is not shared

by Gogo & Golle.

(ii) Limited applicability & limited scope.

* Flush + Reload protocol for covert & side channel attacks

Covert channel protocol:-

	Gogo	Golle	
T1	Clflush(x)		
T2		LOAD(x)	Higher load latency "1" (Cache Miss)
T3	.....		
T4		LOAD(x)	Lower load latency "0" (Cache hit)

T1 to T4: Temporal order

Side channel Protocol

	Gogo	Golle
T1:	Clflush(x)	
T2:	Reload(x)	

Reload(x) → hit, lower access latency, Golle has accessed it in between T1 & T2

Reload(x) → Miss, higher access latency, Golle has not accessed it.

Mitigation of Flush+Reload Attack

Still an open problem. Want to explore the same for your RA1 ☺

Some of the points of interest

(i) Time gap between flush & Reload (0 cycles, ∞ cycles)

(ii) how to find the actual/ideal time gap? If you can't find it, repeat the steps without gap.

(iii) What if there are multiple processes/threads running that are sharing the same address space (e.g. library)?

(iv) How do you find the effectiveness of an attack?

covert channel: Accuracy & Bandwidth

20 bits out of 100 bits at a rate of 2 KB/sec.

Side channel: TPR (True positive rate)

$$\frac{\#\text{ Attacker thinks victim has accessed}}{\#\text{ victim has accessed}}$$

(Higher the better)