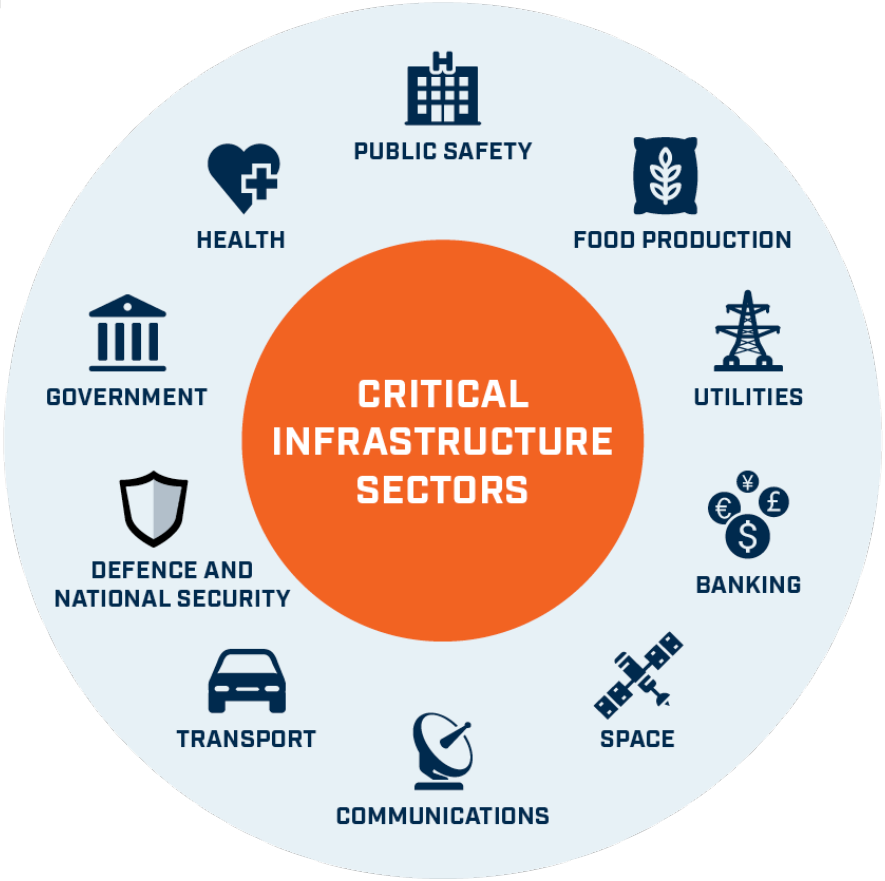
Microarchitectural Attacks

Maria MUSHTAQ
Associate Professor at Télécom Paris

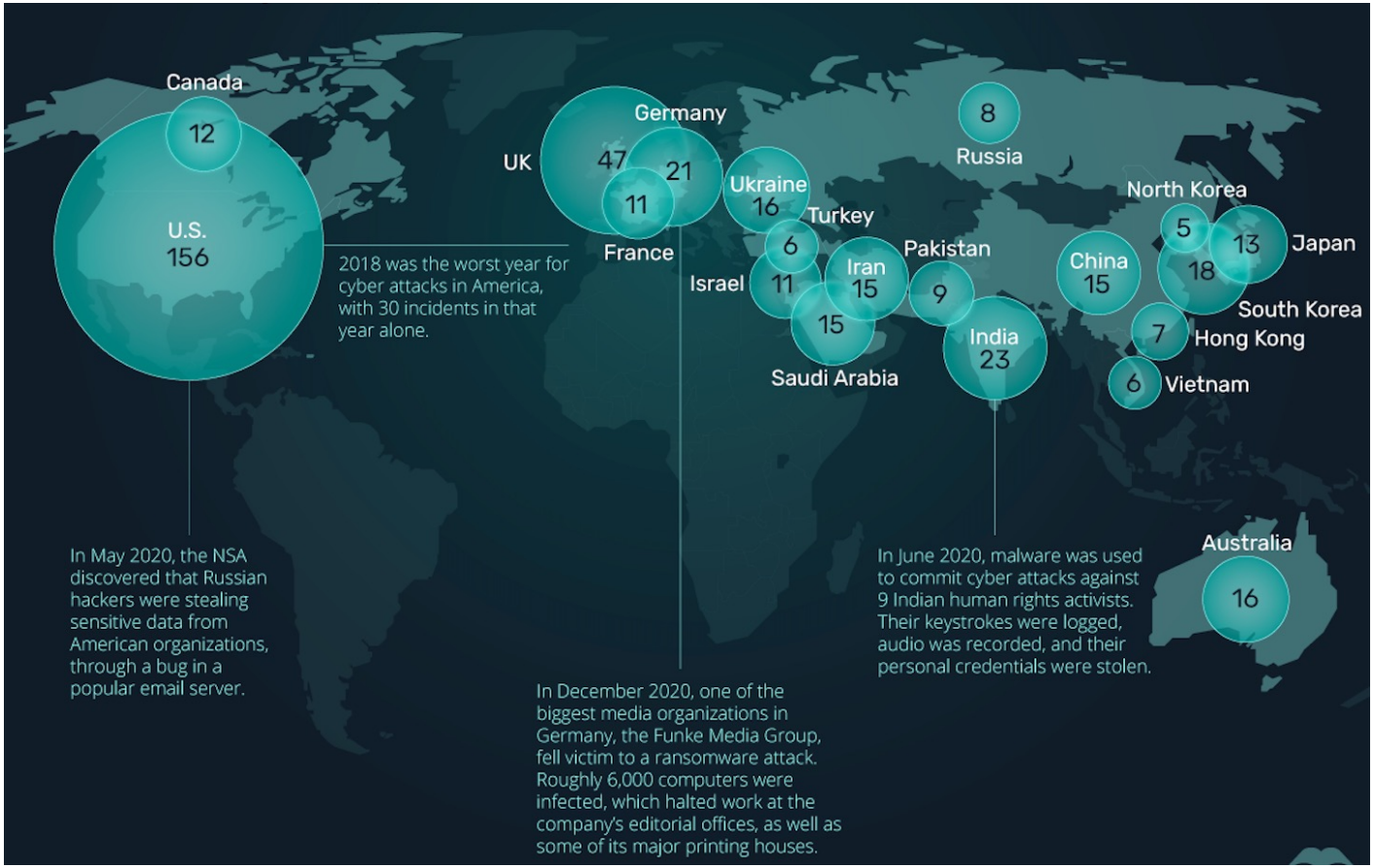
Avec le soutien de la Fondation Mines-Télécom

Information Security -Perspective

- A shared concern by many application domains



Information Security -Perspective



Source: https://www.visualcapitalist.com/cyber-attacks-worldwide-2006-2020/

Information Security -Perspective

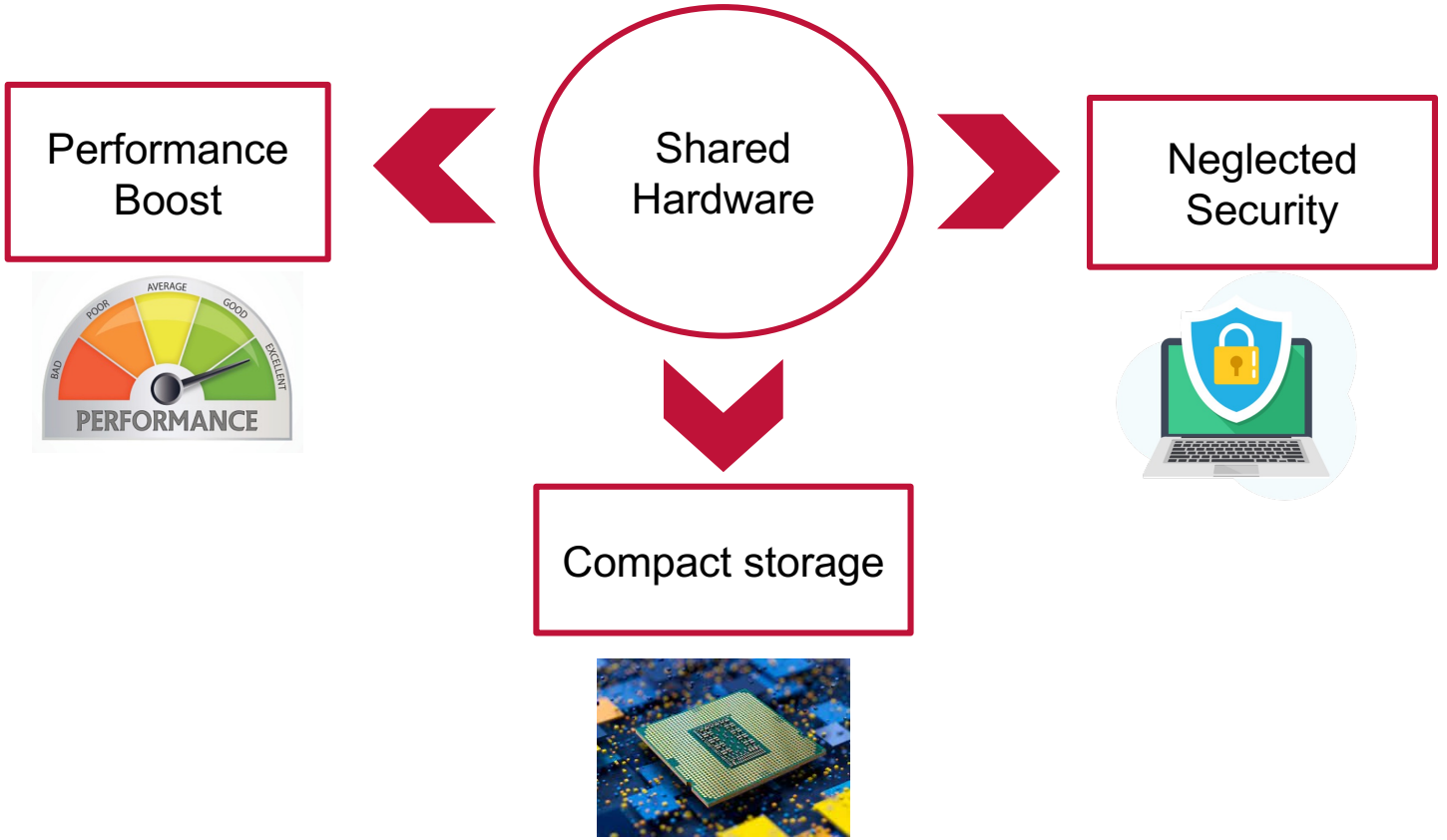
Information Security -Perspective

- Modern Processors - Intel, ARM, AMD are vulnerable…..

Spectre and Meltdown

Information Security -Perspective

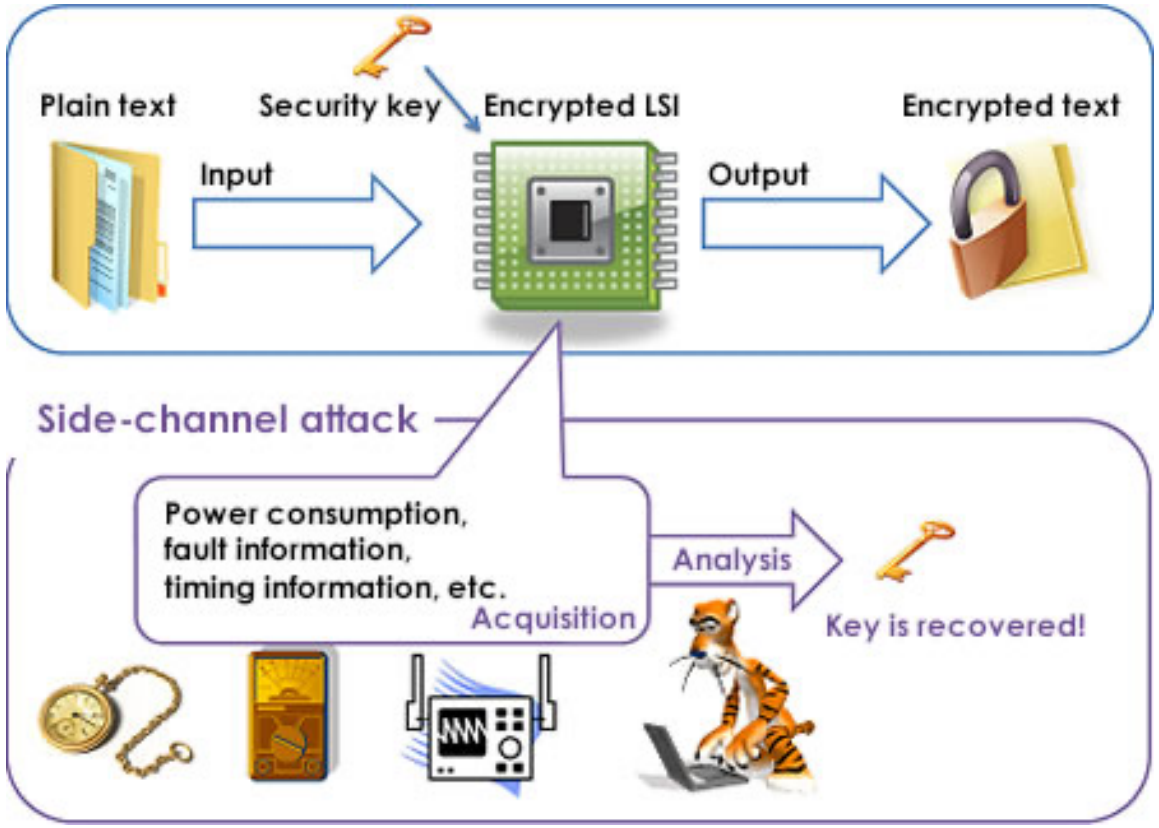
Hardware is Vulnerable!

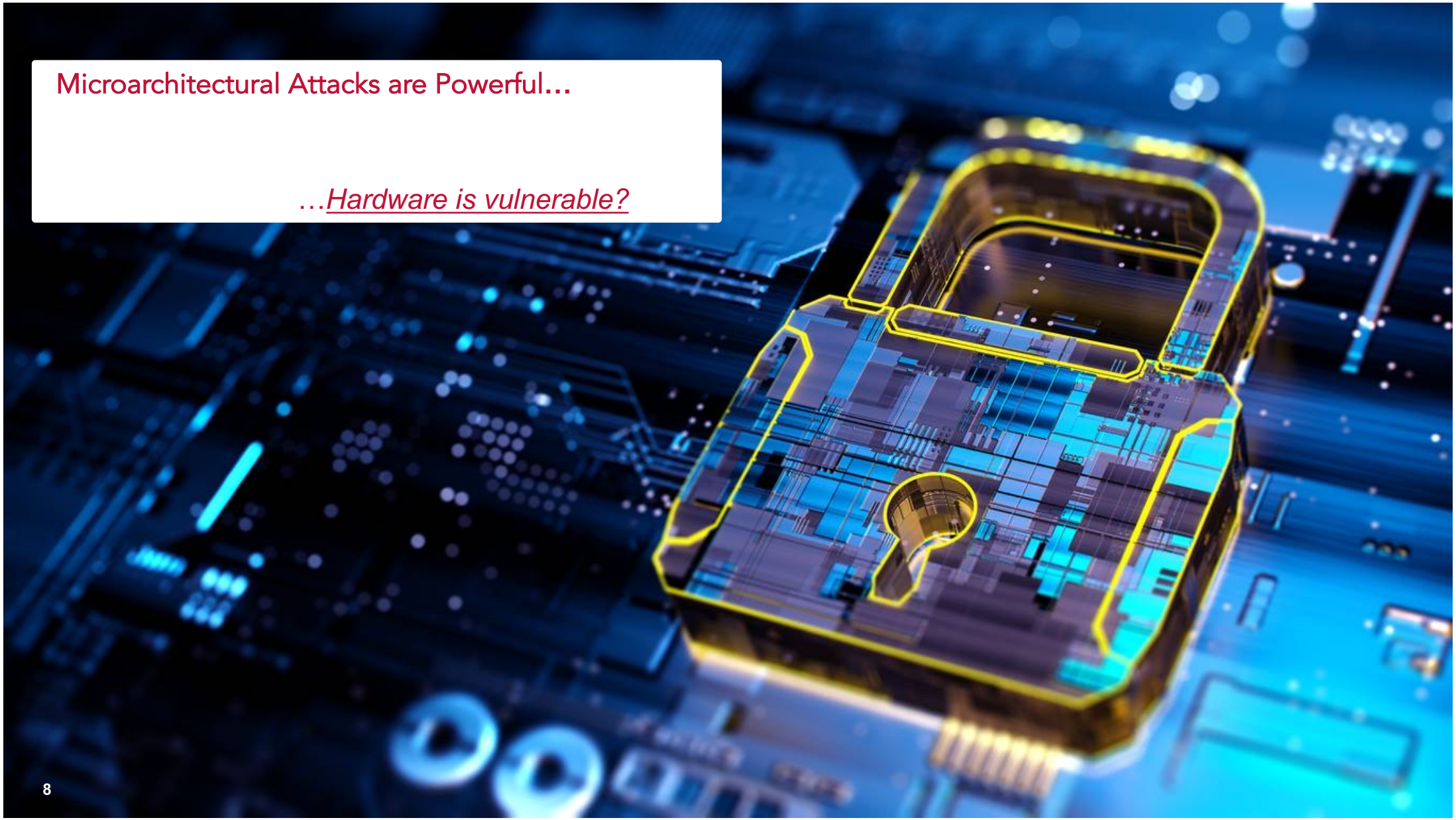


Information Security -Perspective

Side Channel Attacks

- Hardware Attacks
- Software Attacks




Microarchitectural Attacks are Powerful…
…Hardware is vulnerable?

Disclaimer

- Important Background to understand the microarchitectural attacks in detail
- We need to understand how microarchitectural components behave for security reasons i.e., caches

Memory

- Ideal memory: zero latency, zero cost, infinite capacity and bandwidth
- All these ideals oppose each other:

  -Infinite Capacity: bigger takes longer to determine the location

  -Zero Latency: technology i.e., SRAM,DRAM, Disk

  -Zero Cost: require more banks, ports, frequency and faster technology

Memory Technology

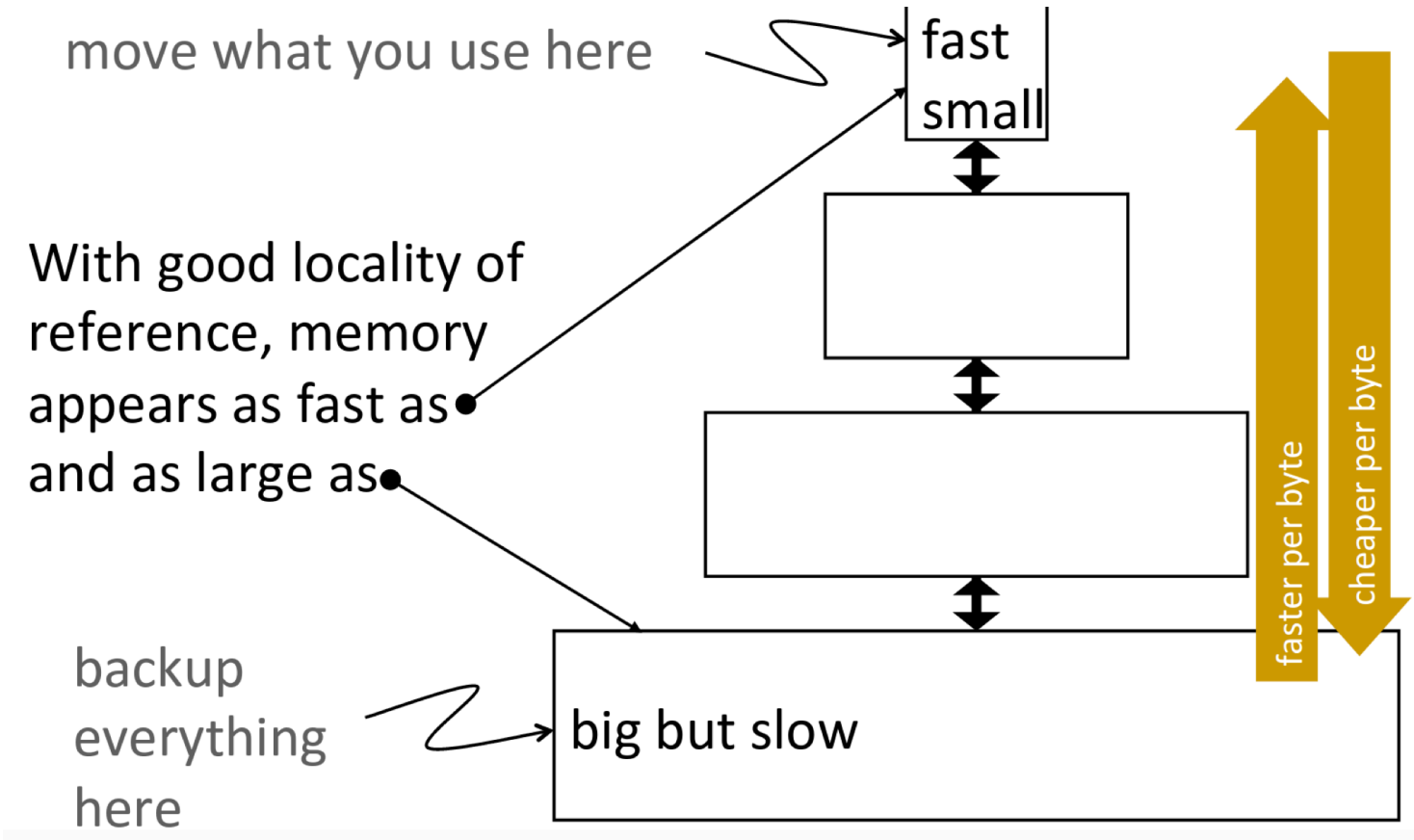
- DRAM VS SRAM

DRAM (Dynamic Random Access Memory)	SRAM (Static Random Access Memory)
Slow acess	Fast Access
High desnsity (1 transistor per cell)	Low densirt (6 transistors per cell)
Low cost	High cost
Require refresh (charge loss over time)	No refresh required

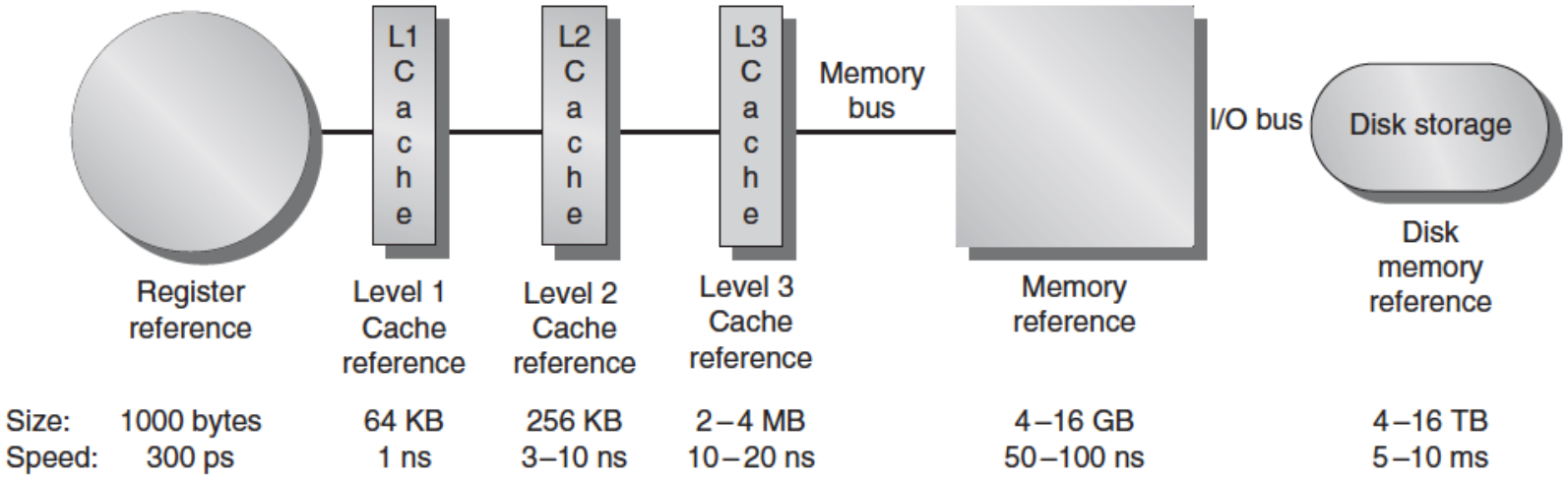
Memory Technology

- Can we have both large and fast memory?
  - No we can not have both large and fast technology with single level of memory
  - Progressively bigger and slower as level go father from processor
  - Ensure most of the data processor needs is kept in the faster levels

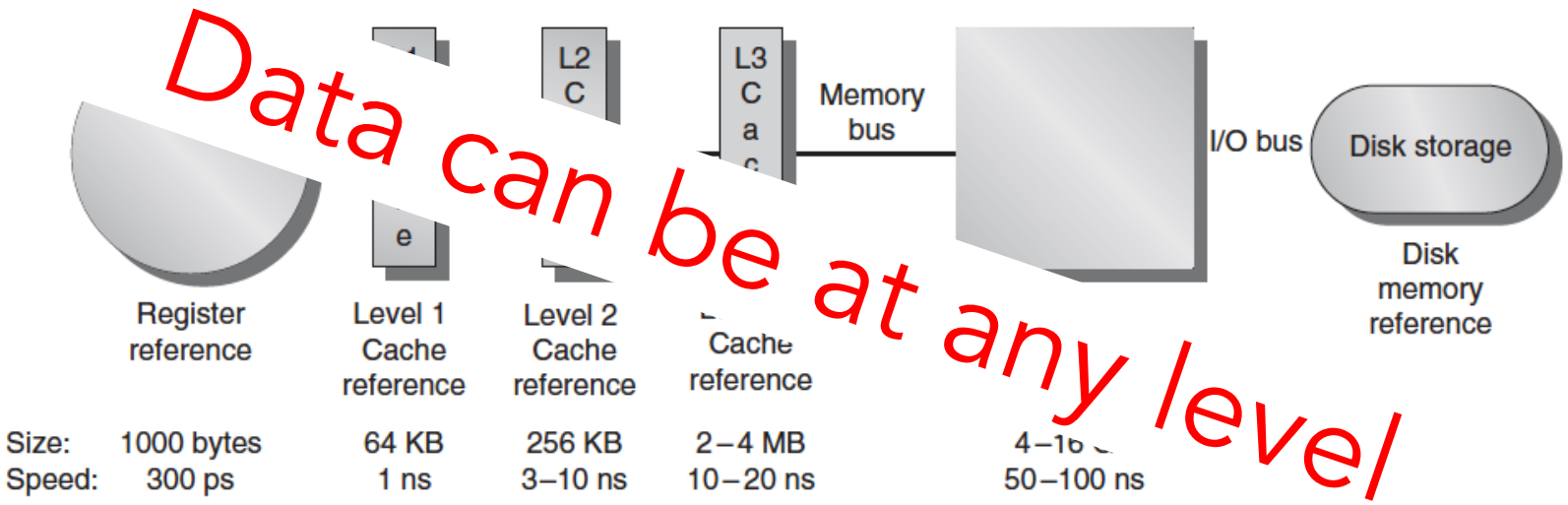
Memory Technology



Memory Hierarchy



Memory Hierarchy



Caching Basics: Temporal VS Spatial

- Memory is organized for *locality*
- Temporal Locality
  - Data/Instructions being referenced are more likely to be referenced again very soon within small time window (i.e., loops)
  - recently accessed data will be accessed again soon
- Spatial Locality
  - A program tends to reference a cluster of memory locations at a time, e.g., sequential instruction access, array traversal
  - Nearby data will be accessed soon

Memory Utilization

- Memory has always been short

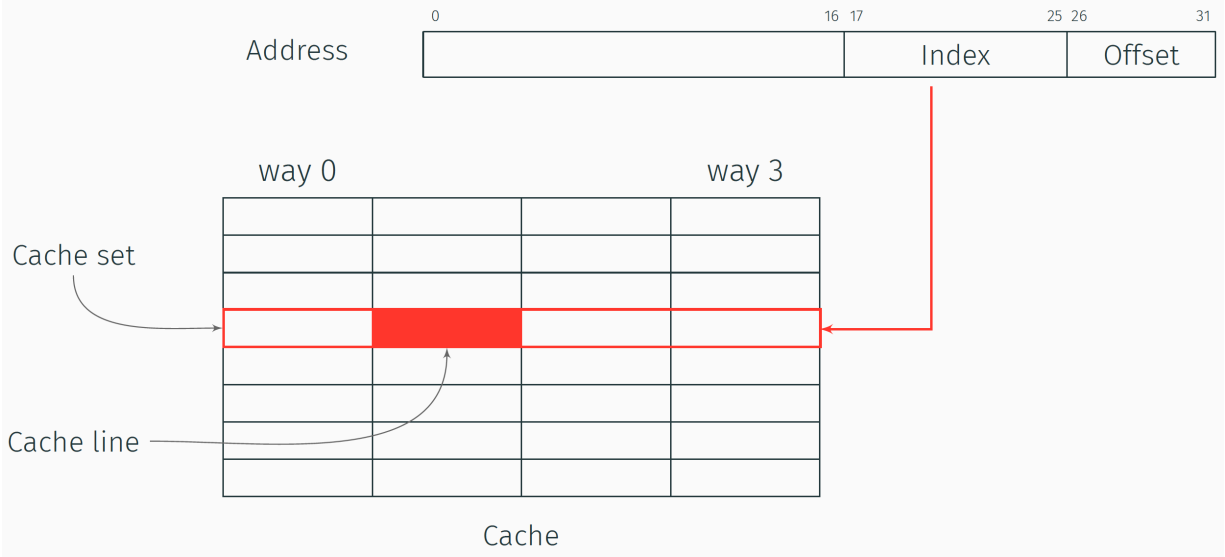
IBM
Model 350 Disk File
5 MB

ScanDisk
1 TB

- Techniques to reduce memory footprint of system
  - Shared libraries
  - Shared data/text segments
  - De-duplication

Set-Associative Cache

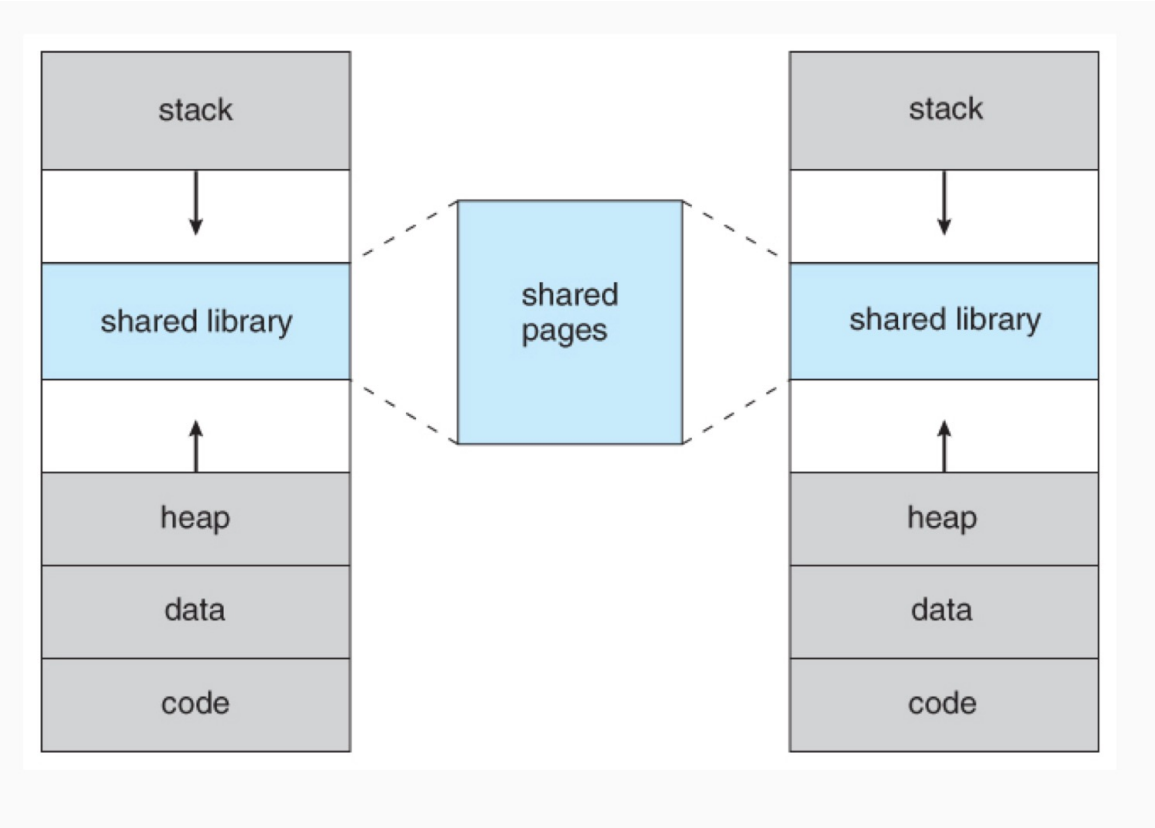
- Memory is organized in a specific way!



  - Data is loaded into specific *set* depending on address
  - Cache line is loaded into a specific *way* depending on replacement policy

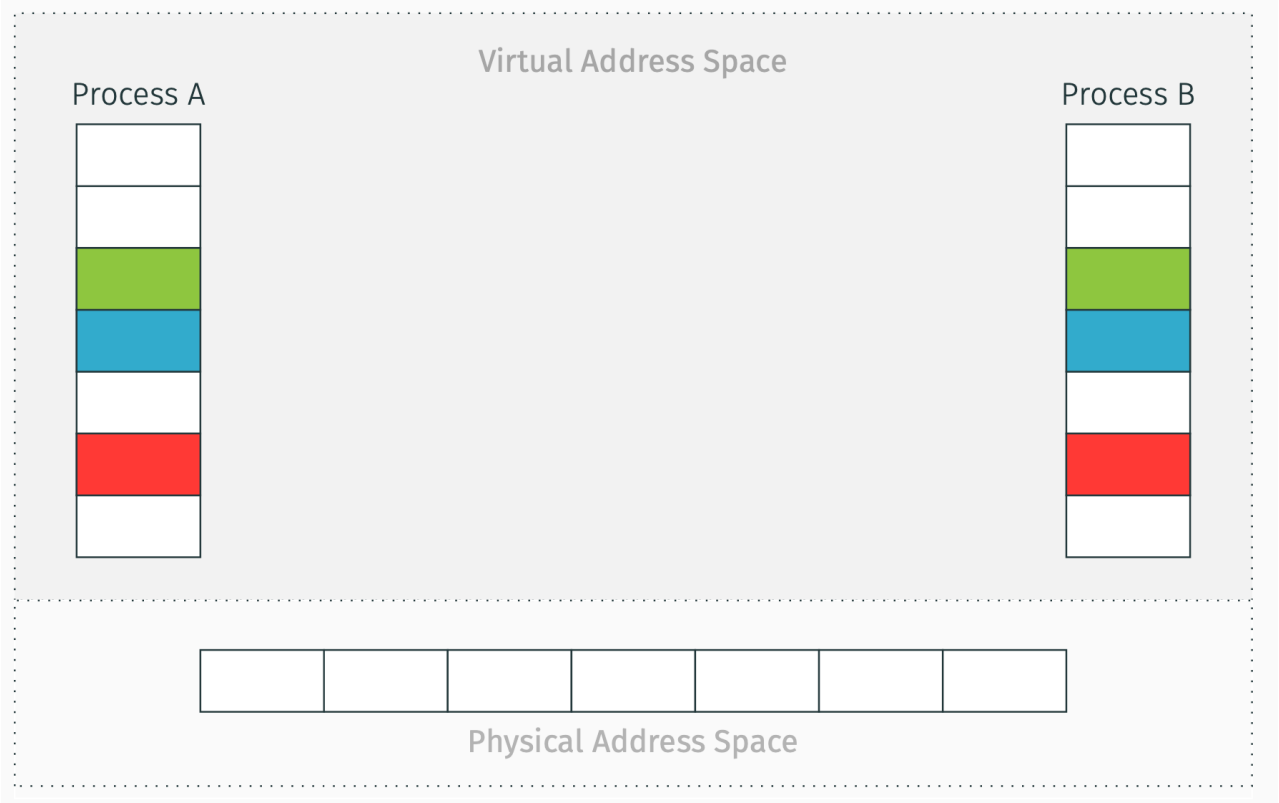
Shared Memory

- Shared Library –Shared in Physical Library



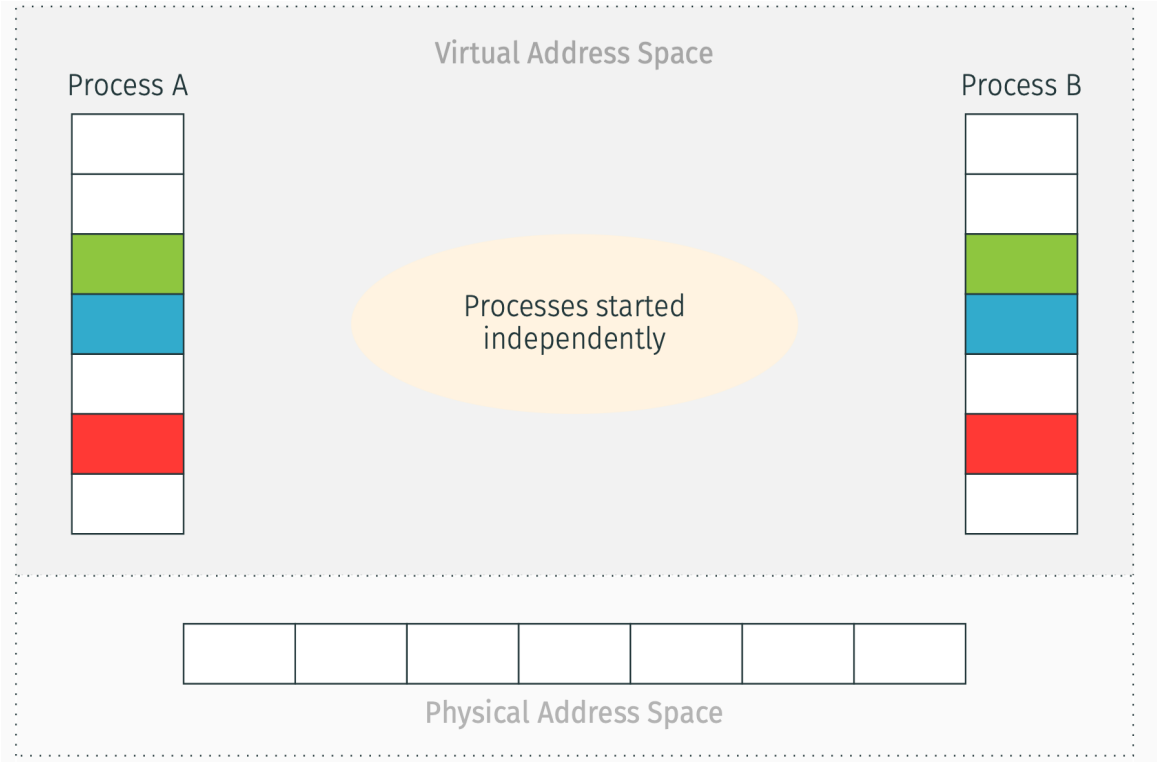
Shared Memory

- Page De-duplication



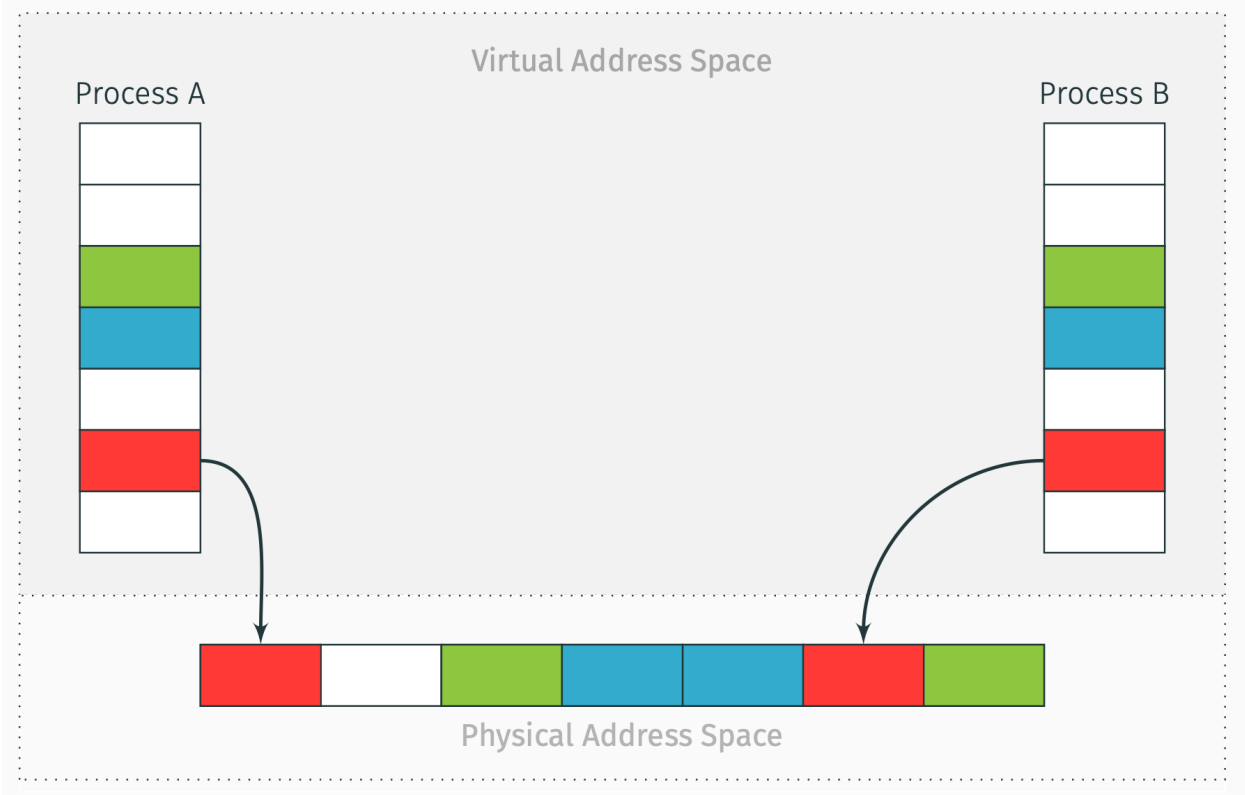
Shared Memory

- Page De-duplication



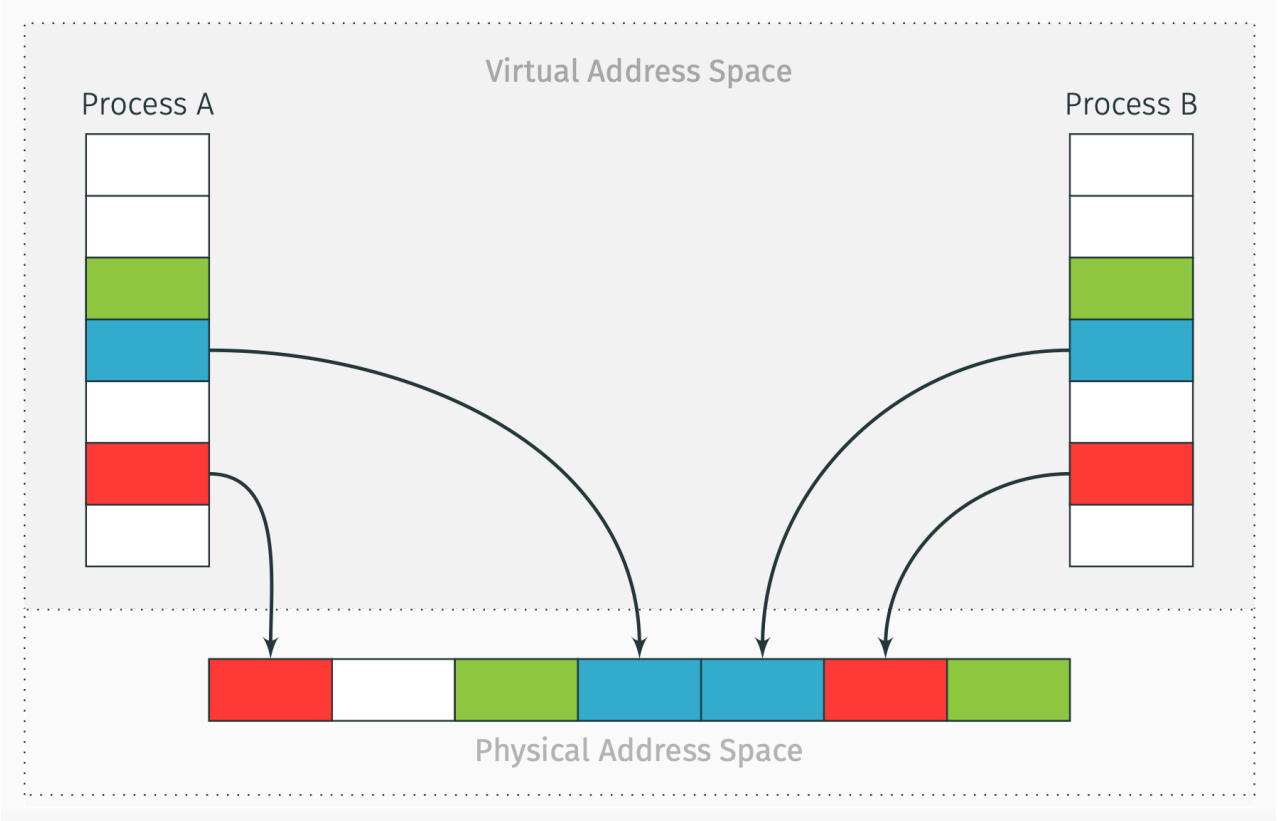
Shared Memory

- Page De-duplication



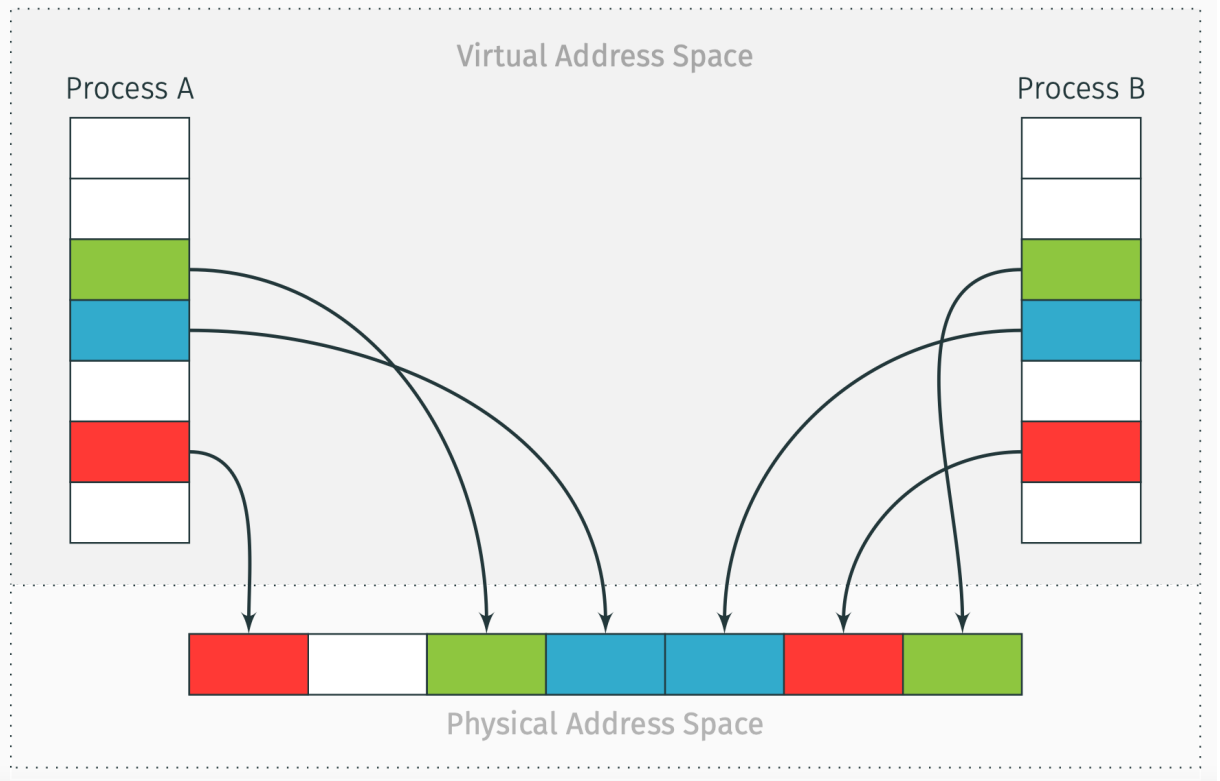
Shared Memory

- Page De-duplication



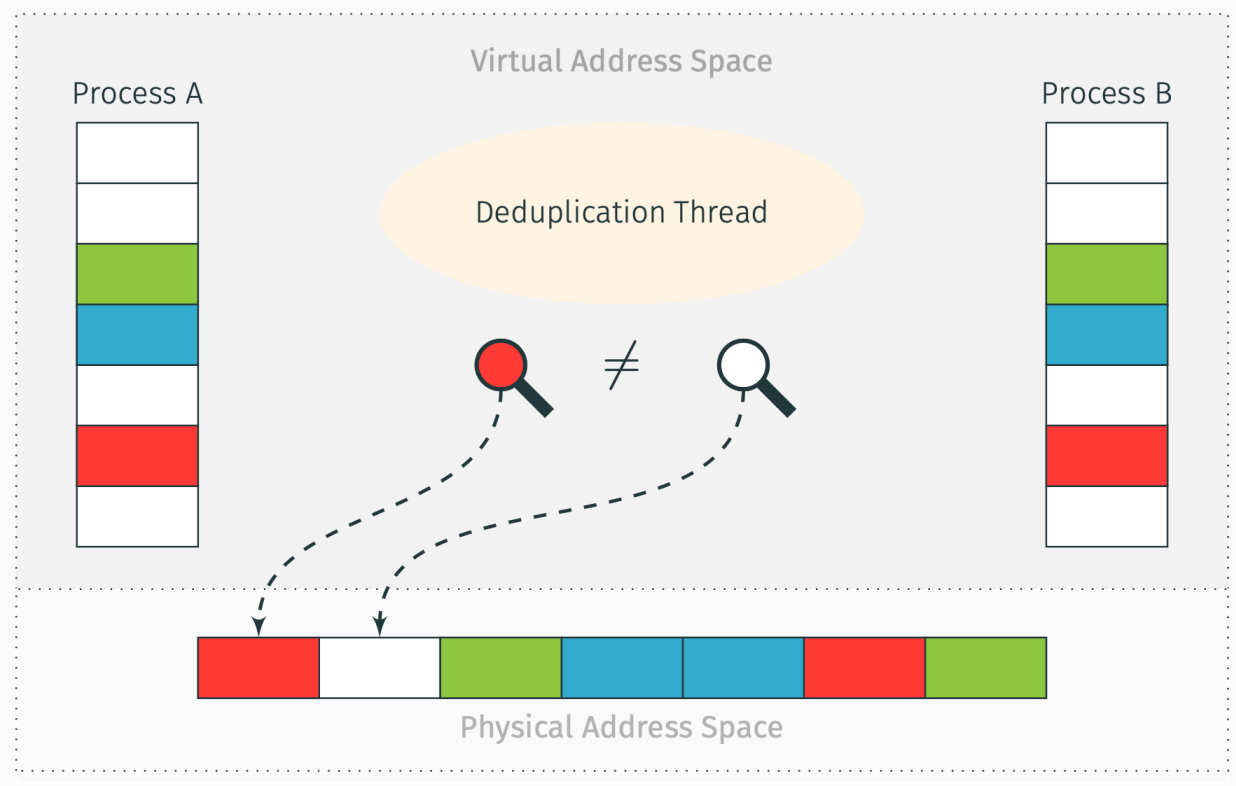
Shared Memory

- Page De-duplication



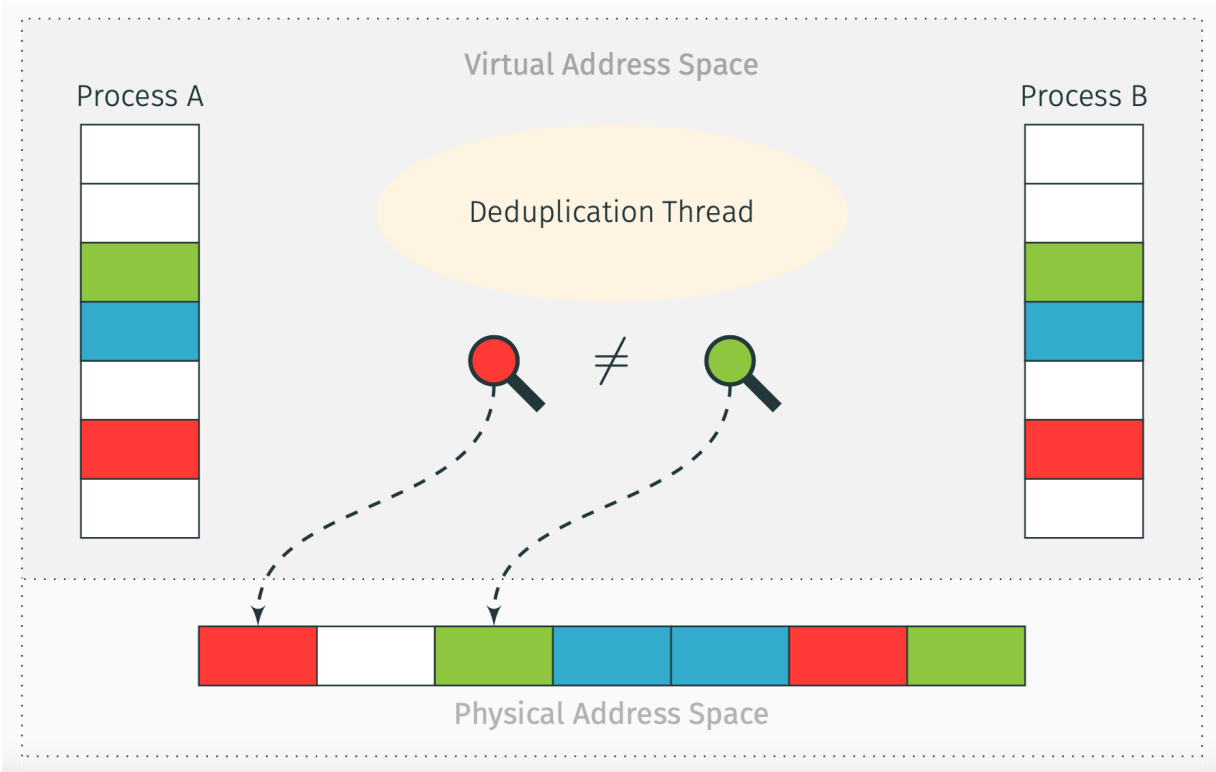
Shared Memory

- Page De-duplication



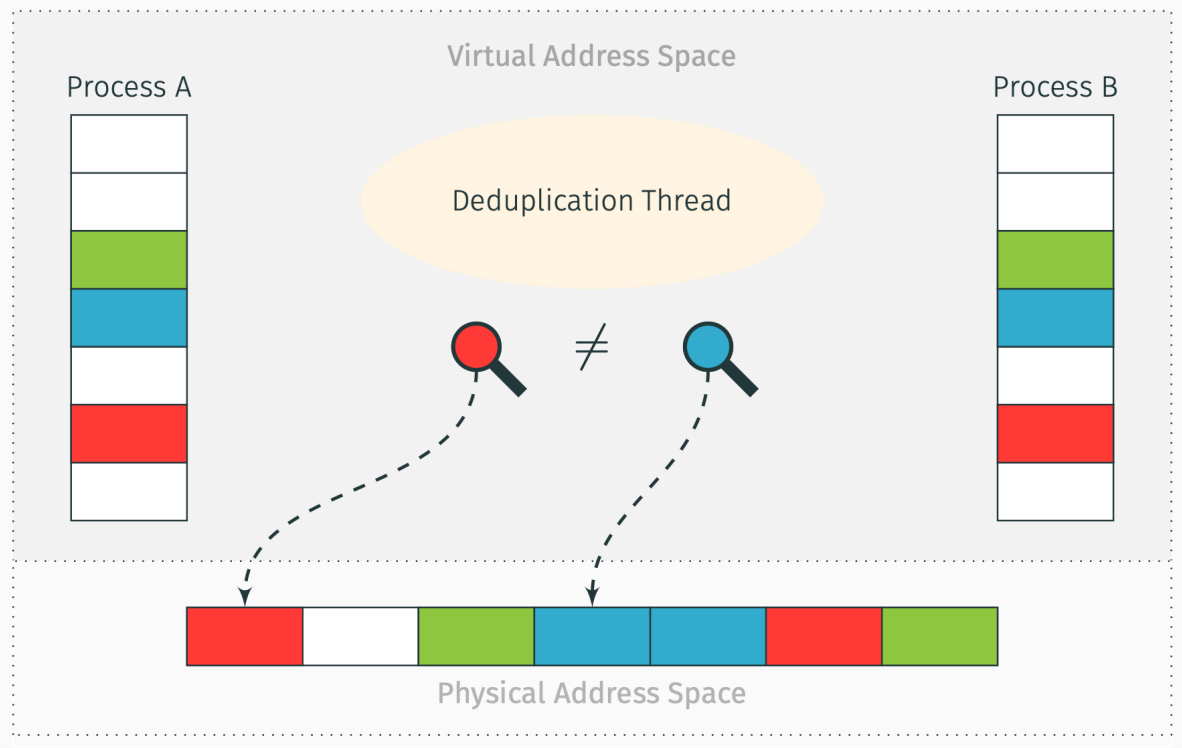
Shared Memory

- Page De-duplication



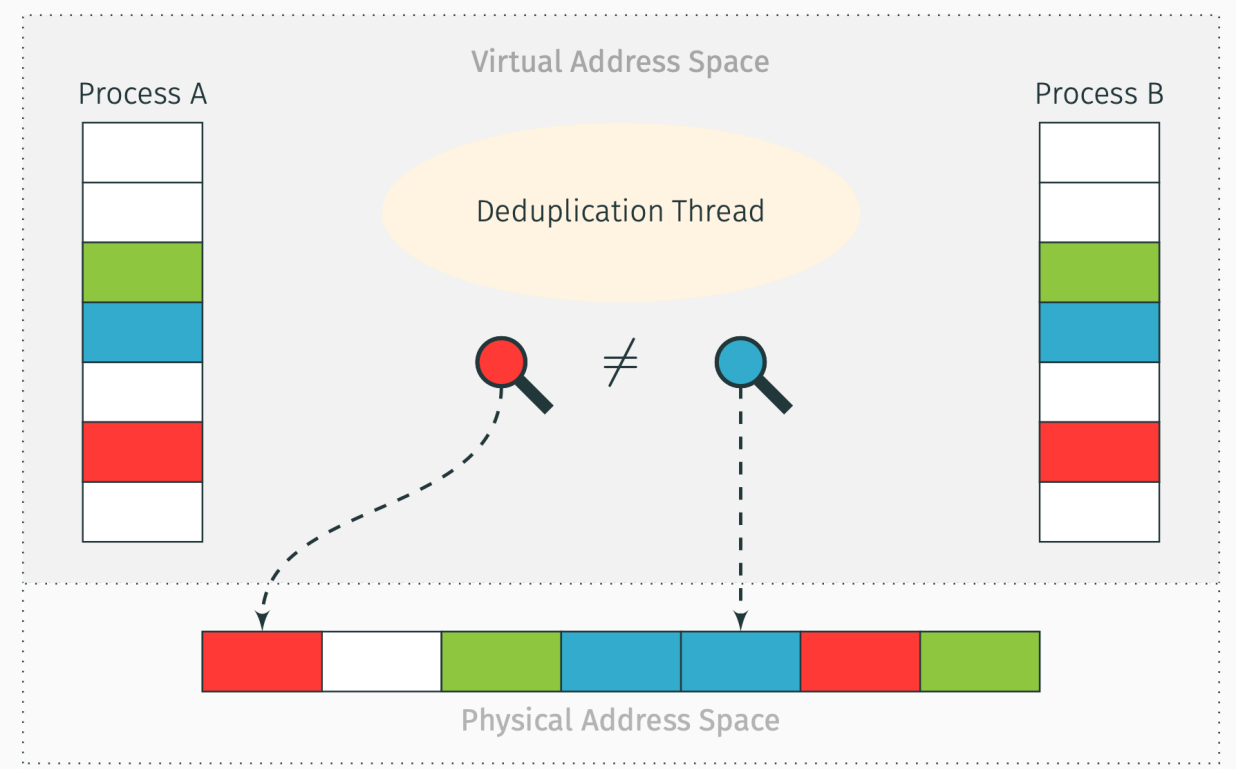
Shared Memory

- Page De-duplication



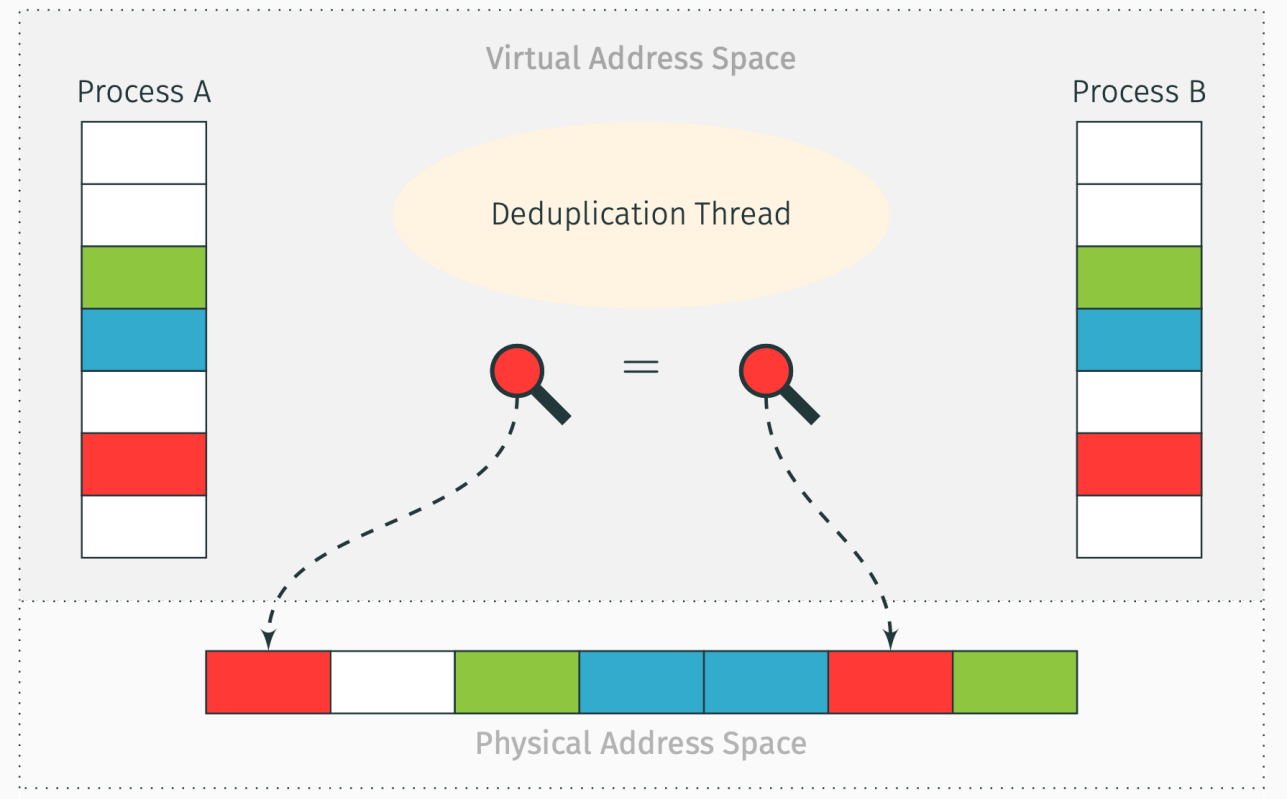
Shared Memory

- Page De-duplication



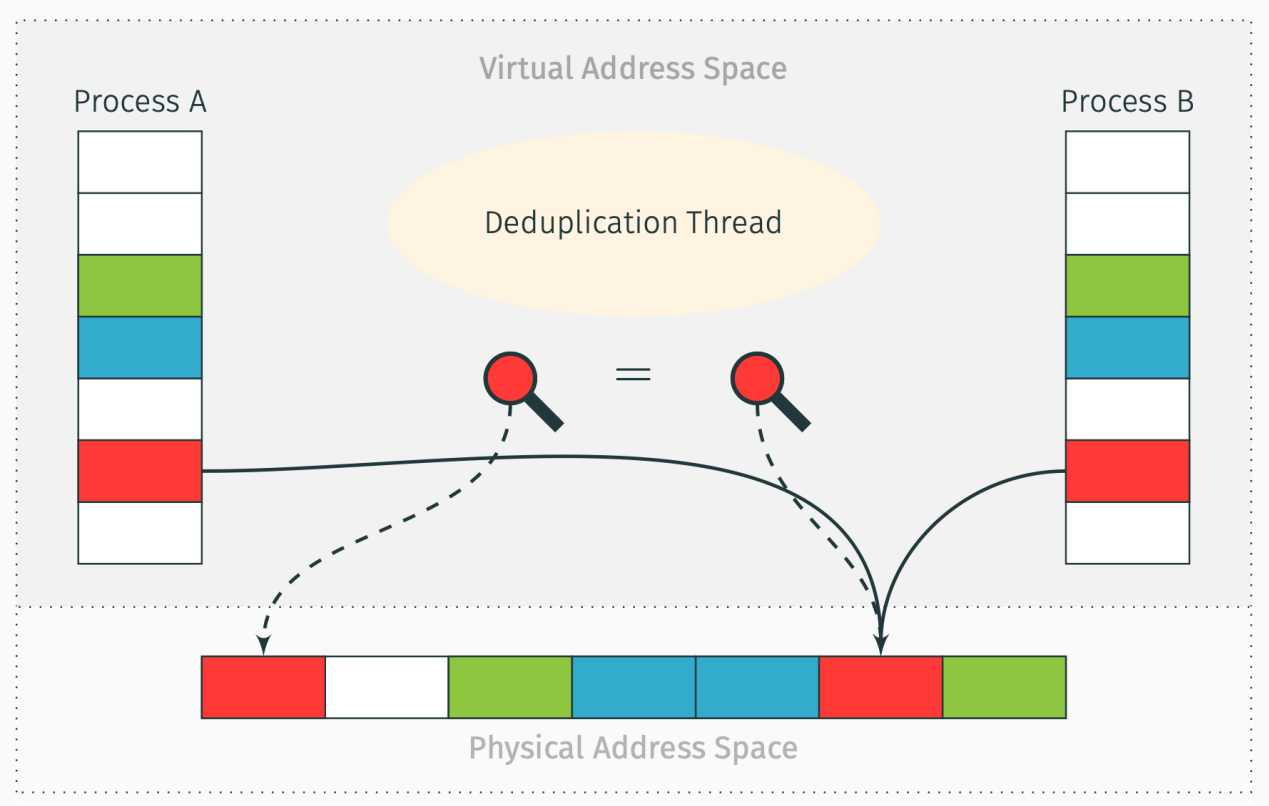
Shared Memory

- Page De-duplication



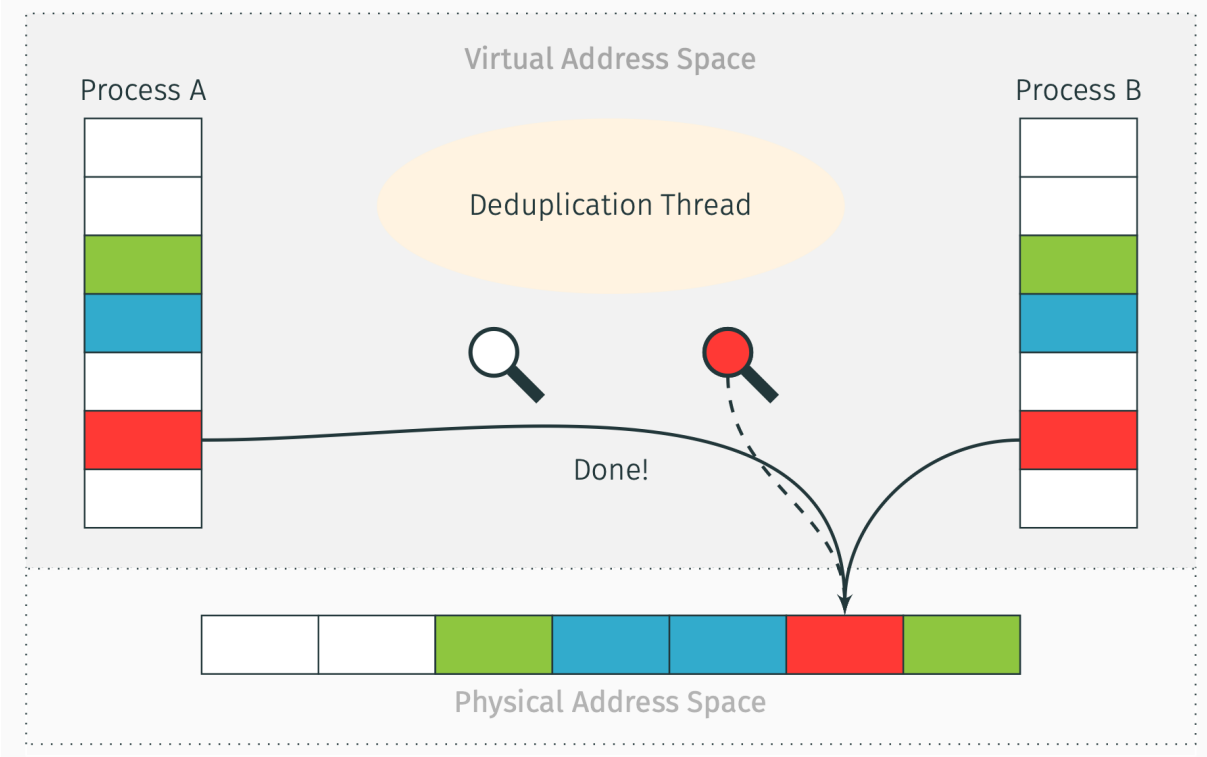
Shared Memory

- Page De-duplication



Shared Memory

- Page De-duplication



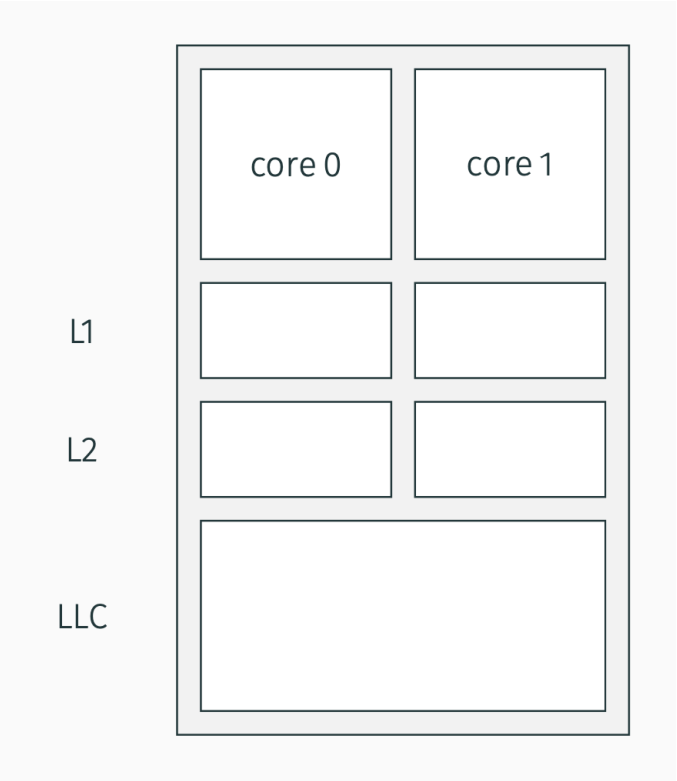
- **What happens when there is no shared memory?**

e.g. there is no memory deduplication on Amazon EC2

Shared Memory

- Inclusive Caches

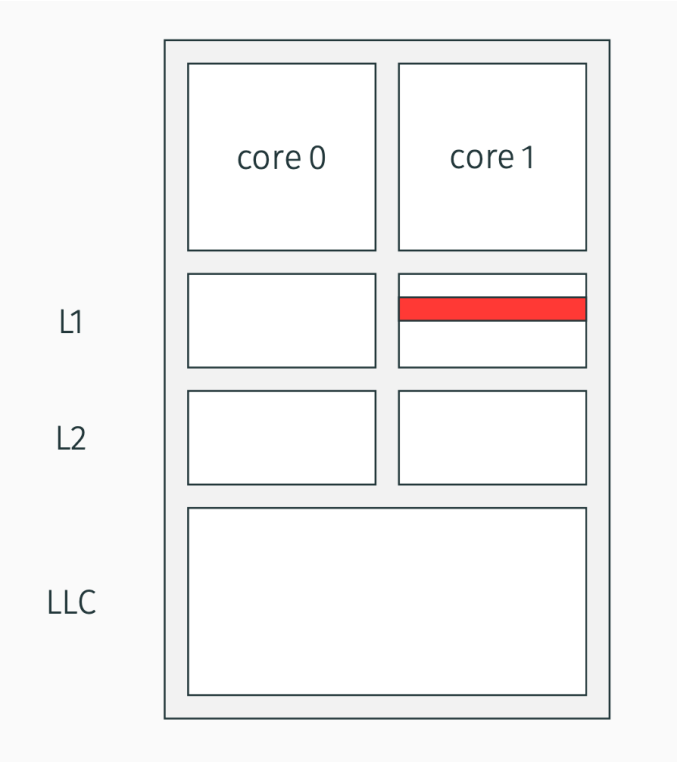
- Inclusive LLC is superset of L1, L2
- Data evicted from LLC is evicted from L1, L2
- A core can evict lines in the private L1 of another core



Shared Memory

- Inclusive Caches

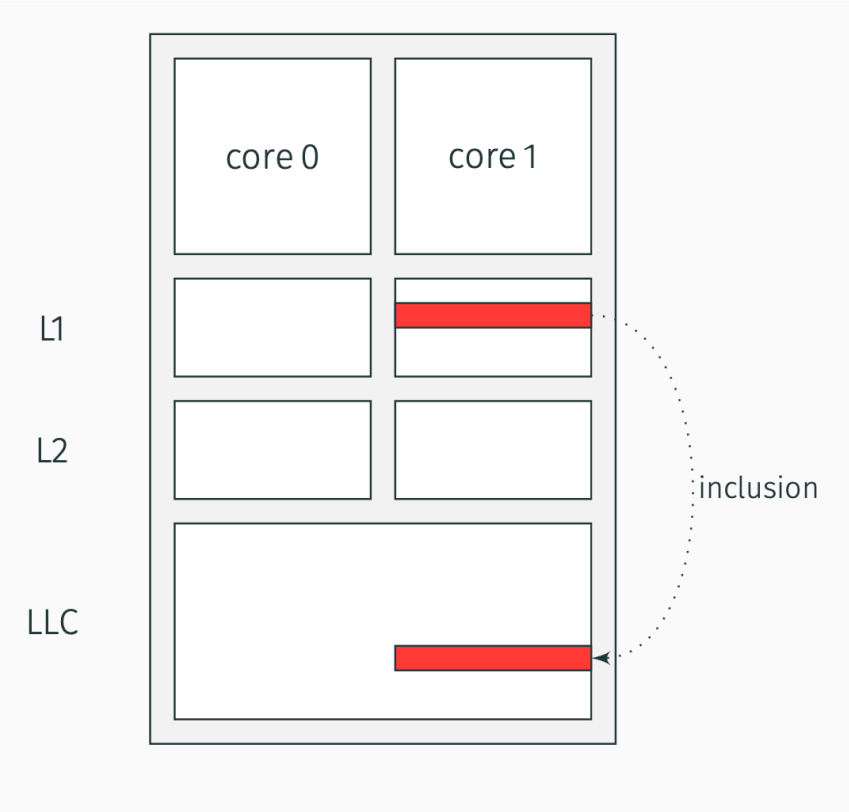
- Inclusive LLC is superset of L1, L2
- Data evicted from LLC is evicted from L1, L2
- A core can evict lines in the private L1 of another core



Shared Memory

- Inclusive Caches

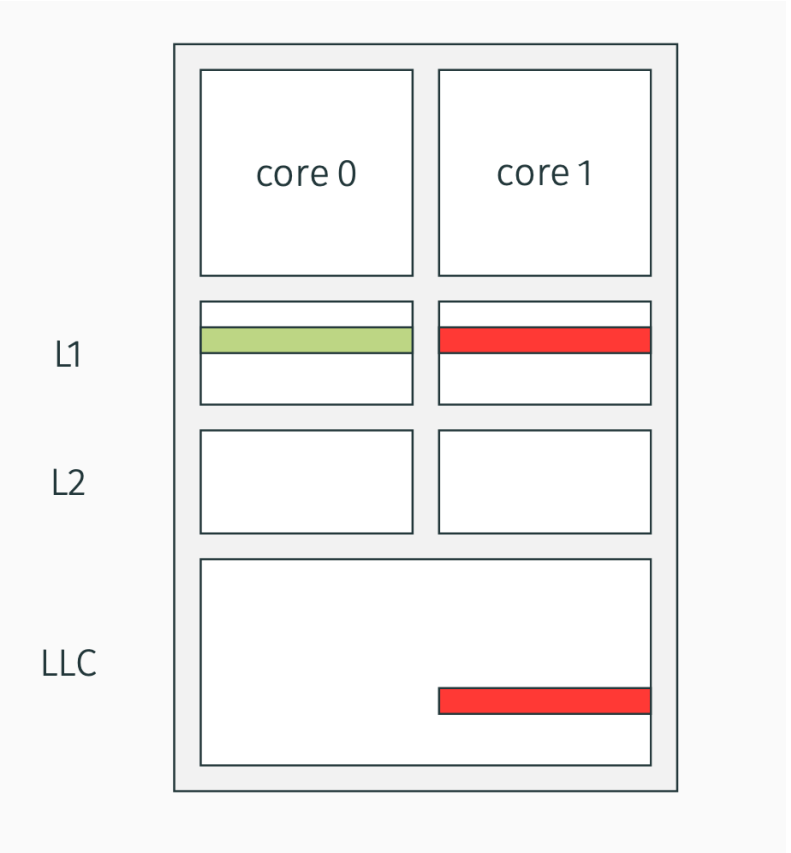
- Inclusive LLC is superset of L1, L2
- Data evicted from LLC is evicted from L1, L2
- A core can evict lines in the private L1 of another core



Shared Memory

- Inclusive Caches

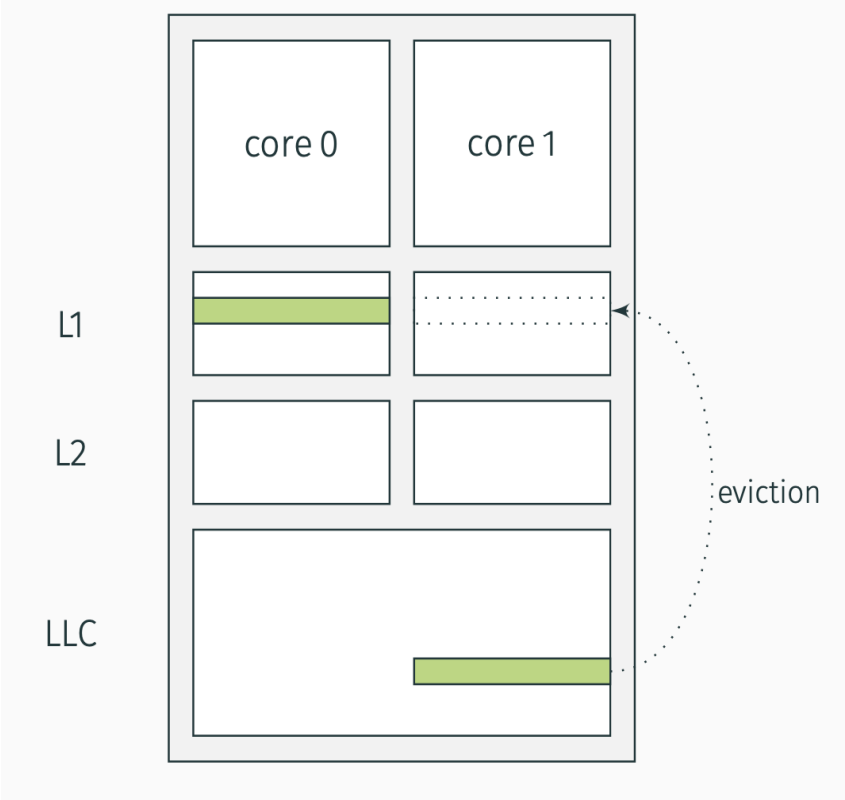
- Inclusive LLC is superset of L1, L2
- Data evicted from LLC is evicted from L1, L2
- A core can evict lines in the private L1 of another core



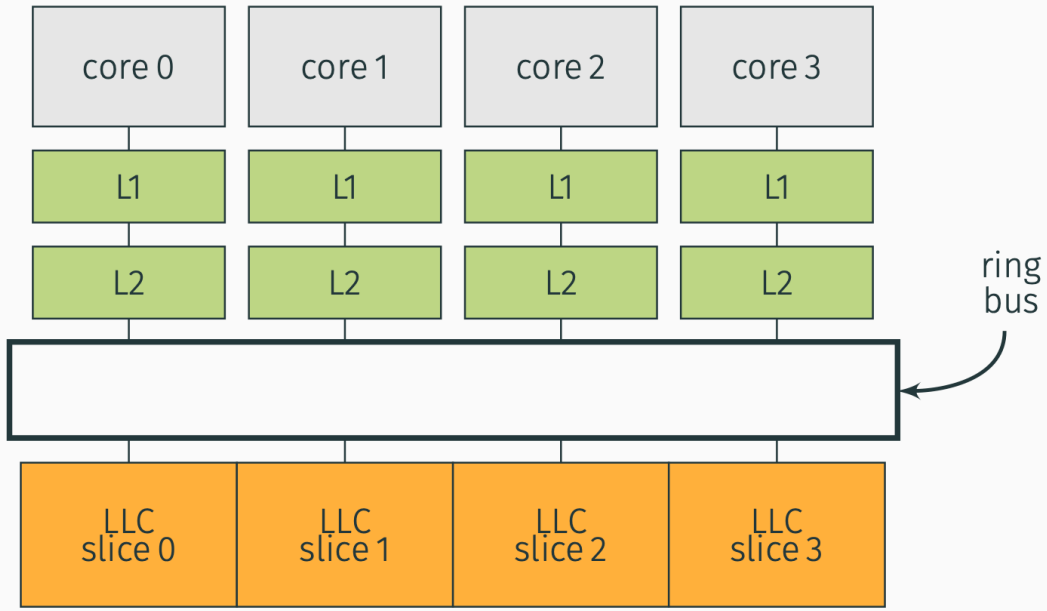
Shared Memory

- Inclusive Caches

- Inclusive LLC is superset of L1, L2
- Data evicted from LLC is evicted from L1, L2
- A core can evict lines in the private L1 of another core



Caches on Intel CPU's



- set-associative
- L1 and L2 are private
- last-level cache
  - divided in slices
  - shared across cores
  - inclusive

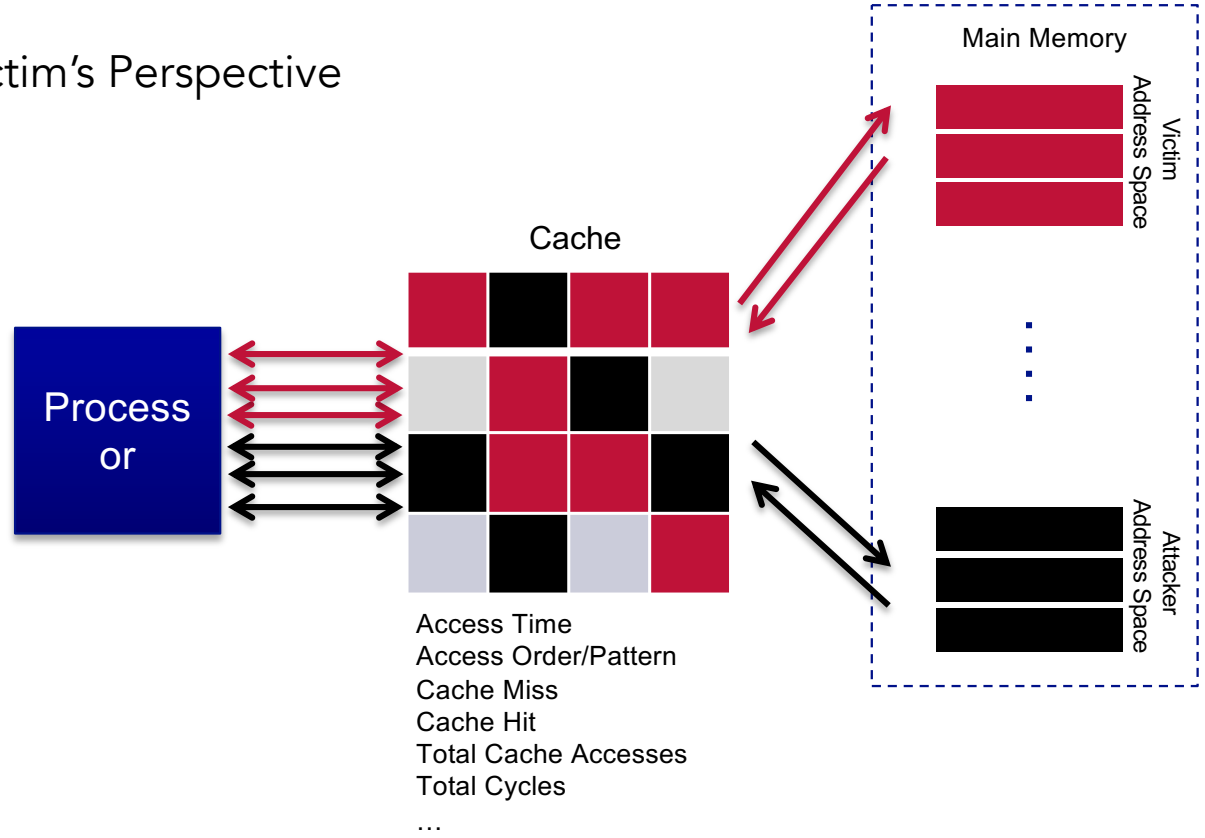
Caches on Intel CPU's

- User program can optimize cache usage in x86:

  - Prefetch: can suggest CPU to load data
  - Clflush: throw out data from all caches

Caches on Intel CPU's

- Cache SCAs affect or alter cache behavior!
  - The Victim's Perspective



CPU Caches

CPU Caches

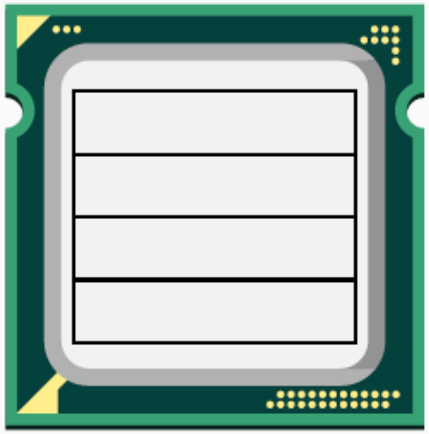
CPU Caches

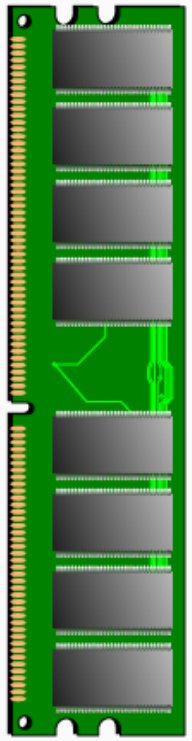

1337 4242
FOOD CACHE
Revolutionary concept!
Store your food at home, never go to the grocery store during cooking.
Can store ALL kinds of food.

Caches Leak Information

- Memory Access Time & Access Pattern

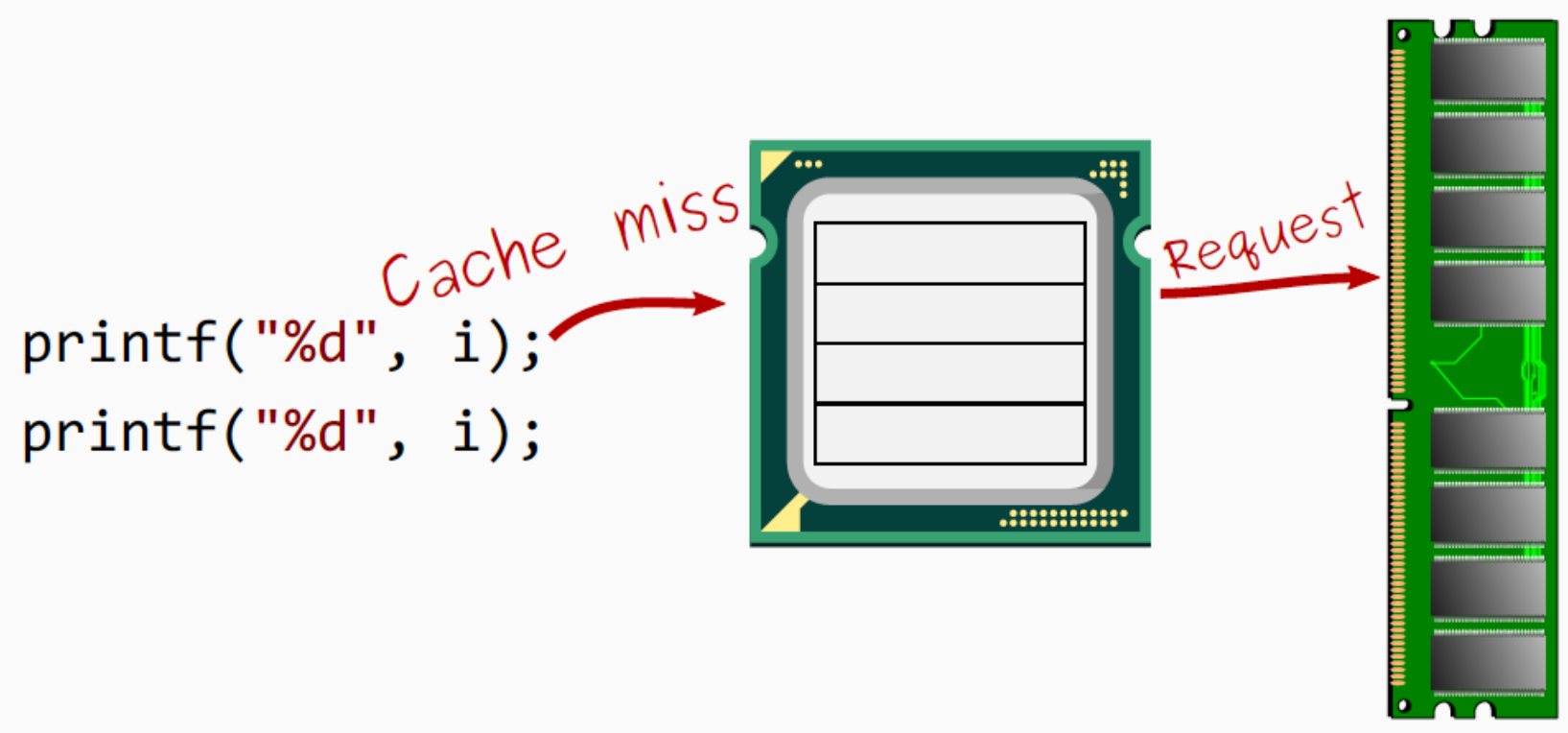
```
printf("%d", i);
printf("%d", i);
```





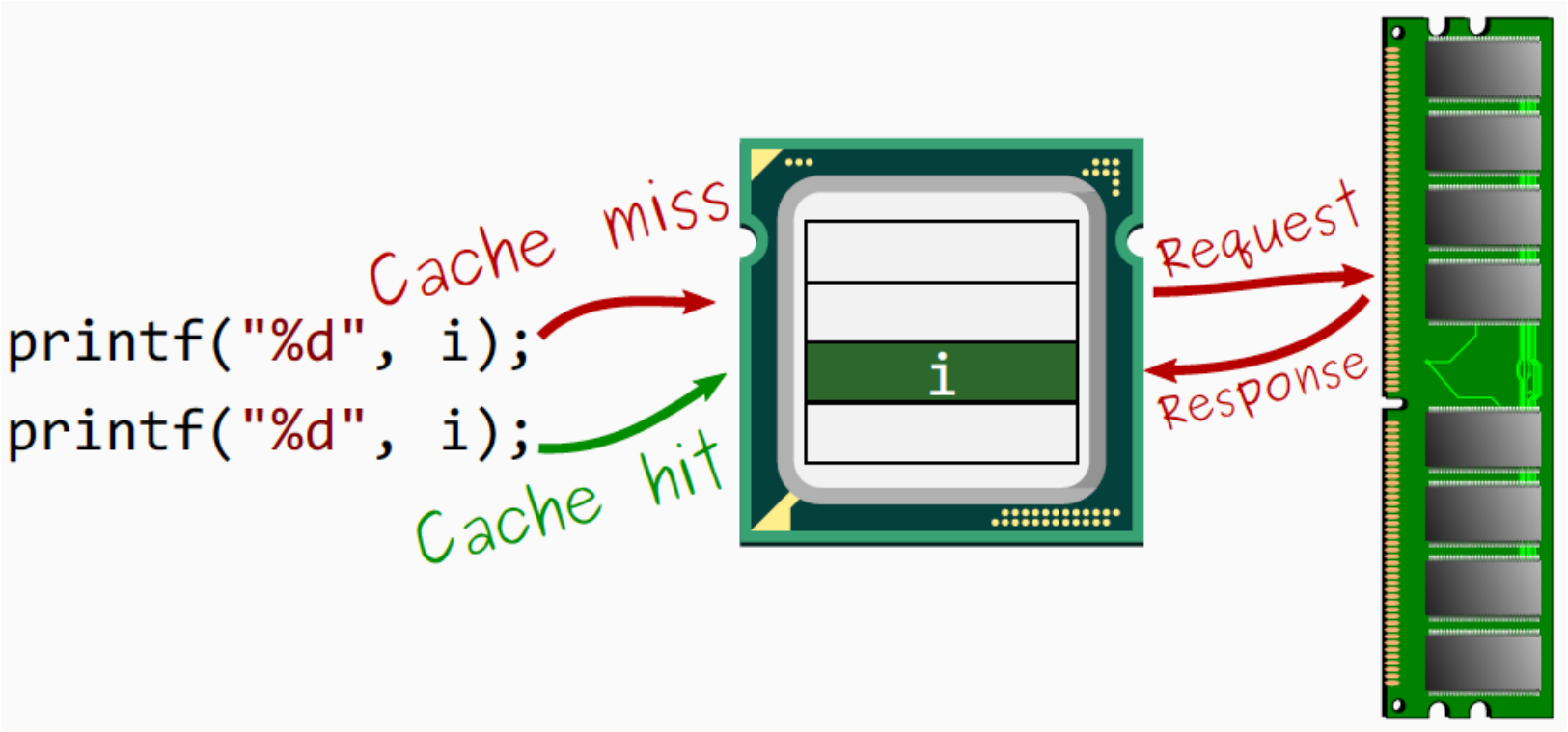
Caches Leak Information

- Memory Access Time & Access Pattern



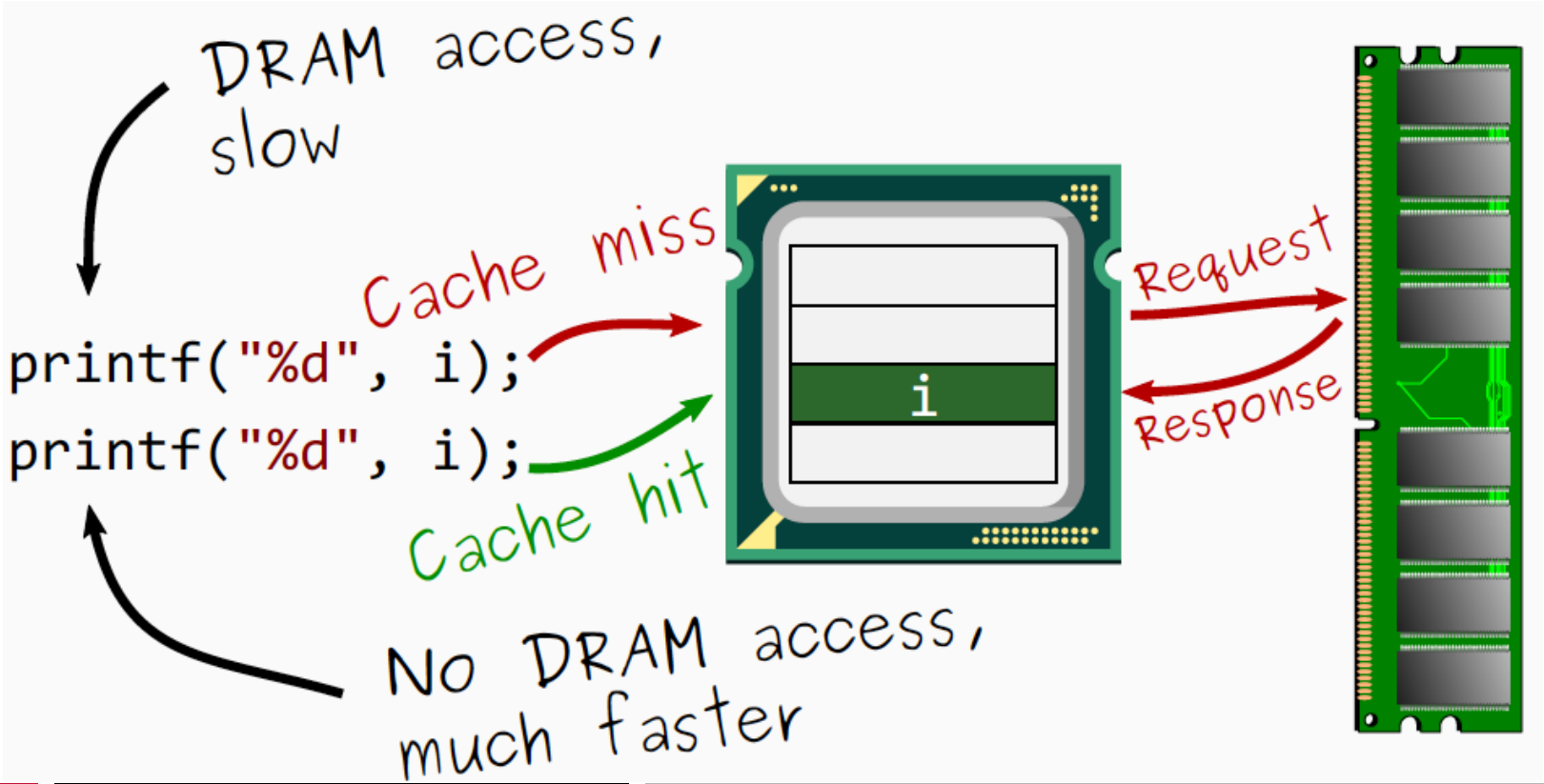
Caches Leak Information

- Memory Access Time & Access Pattern



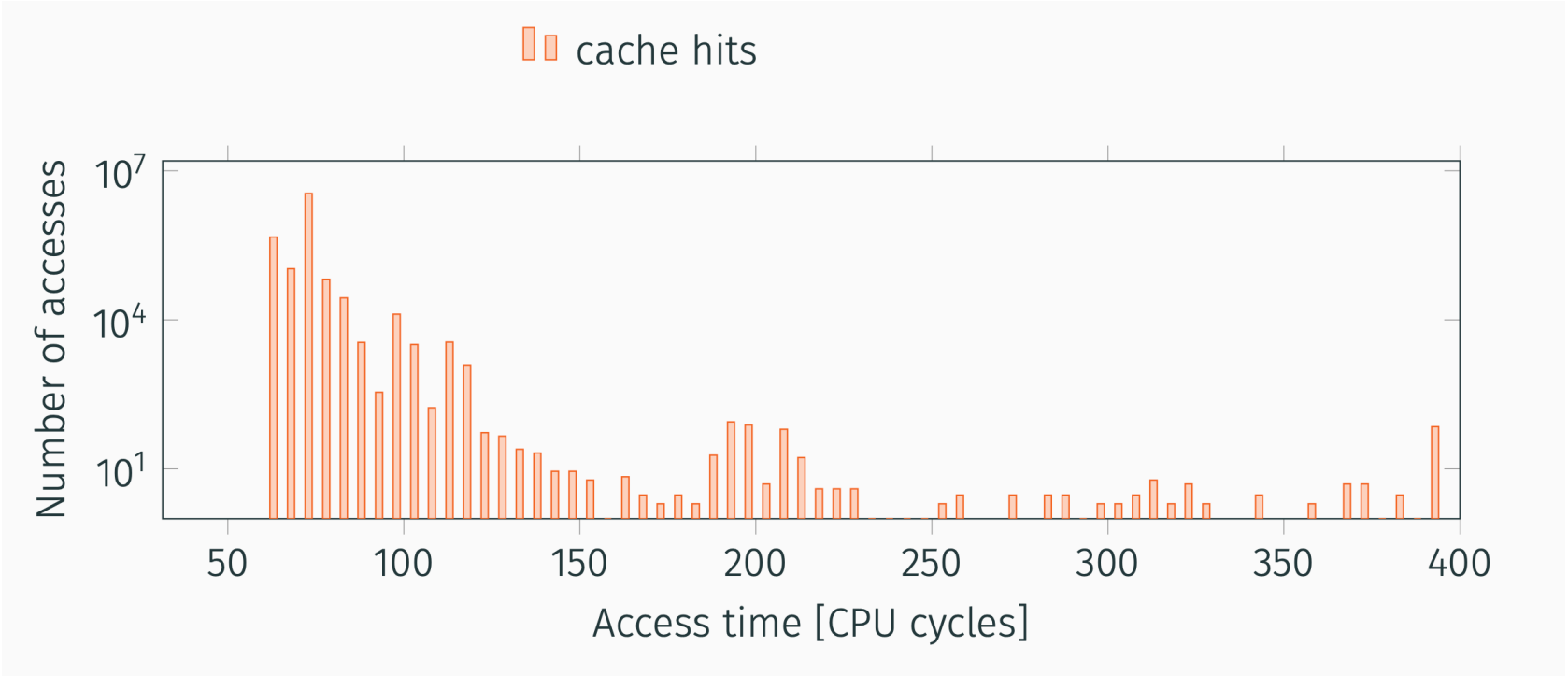
Caches Leak Information

- Memory Access Time & Access Pattern



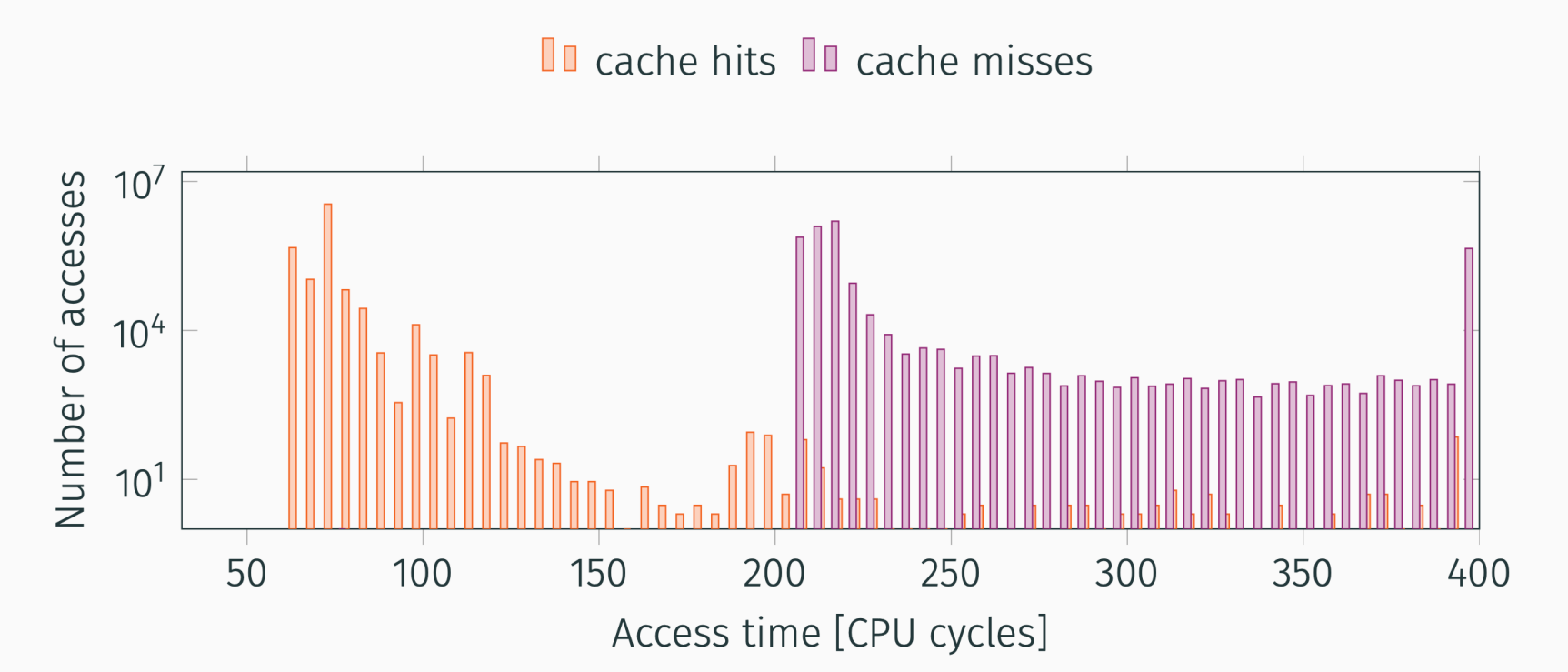
Caches Leak Information

- Memory Access Time & Access Pattern



Caches Leak Information

- Memory Access Time & Access Pattern



Known States in a Processor

	L1d	L1i	L2	L3
level size	32 KB	32 KB	256 KB	3 MB
line size	64 B	64 B	64 B	64 B
# ways	8	8	8	12
# sets	64	64	512	4096
inclusive?	no	no	no	yes

Intel Core i-5 with 2 cores, 4 threads

Known States in a Processor

event	latency
1 CPU cycle	0.3 ns
level 1 cache access	0.9 ns
level 2 cache access	2.8 ns
level 3 cache access	12.9 ns
main memory access	120 ns
solid-state disk I/O	50-150 us
rotational disk I/O	1-10 ms

Wake up call!

Why known state of processor a threat?

Conclusions

- Software execution on underlying hardware is a problem
- Shared Hardware is vulnerable
- Timing information can reveal a lot about a victim program
- Now we will focus on microarchitectural attacks

Quiz –Student Evaluation

QUIZ: Can somebody tell me?

- Benefit of data sharing and disadvantage?
- Inclusive Caches are good for performance but what is the security threat from them?
- What is the fastest to access data; cache hit or cache miss?
- When there is a cache miss, data is accessed from?
- How pre-determined timing information of a processor can be a security threat?
- De-duplication helps to optimise memory locations?
- How many levels a standard Intel CPU cache has?
- What is the difference between a cache and DRAM memory?

Microarchitectural Attacks

- Side-Channel Attacks: a malicious process spies the benign process to steal secret information

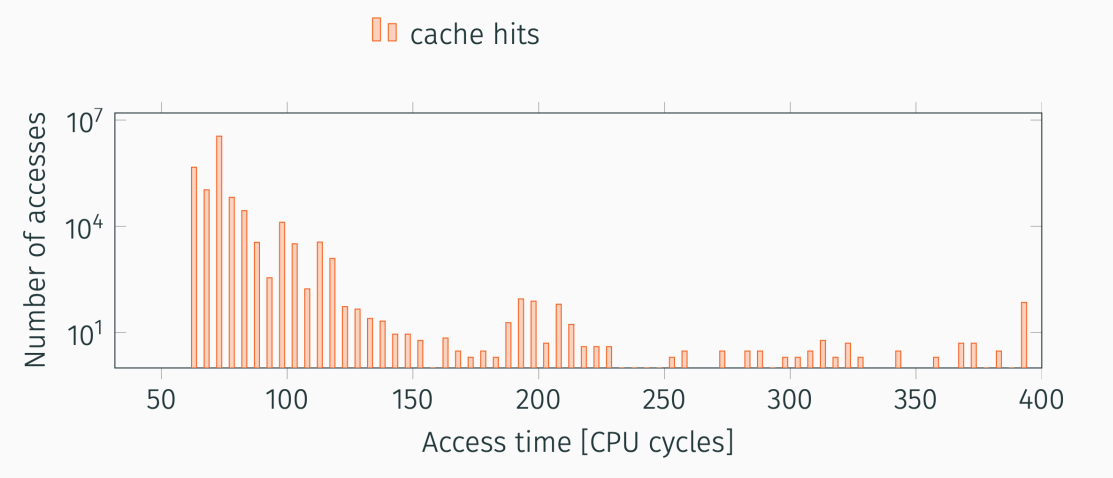
  - exploit timing differences from memory accesses
  - attacker monitors the lines access, not the content
  - learn timing difference by cache hit, cache miss

First Step: Build Histogram

1. Build data for cache hits and cache misses
2. Time each case for multiple samples of time
3. Build histogram
4. Find a threshold to distinguish both cases

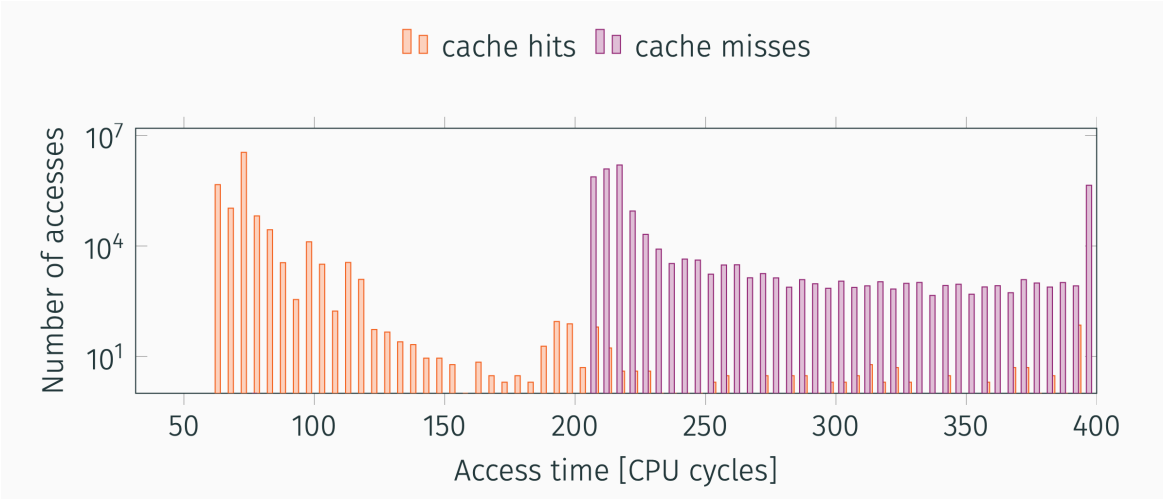
Build Histogram: Cache hits

1. Measure time
2. Acess cache hits
3. Measure time
4. Update histogram



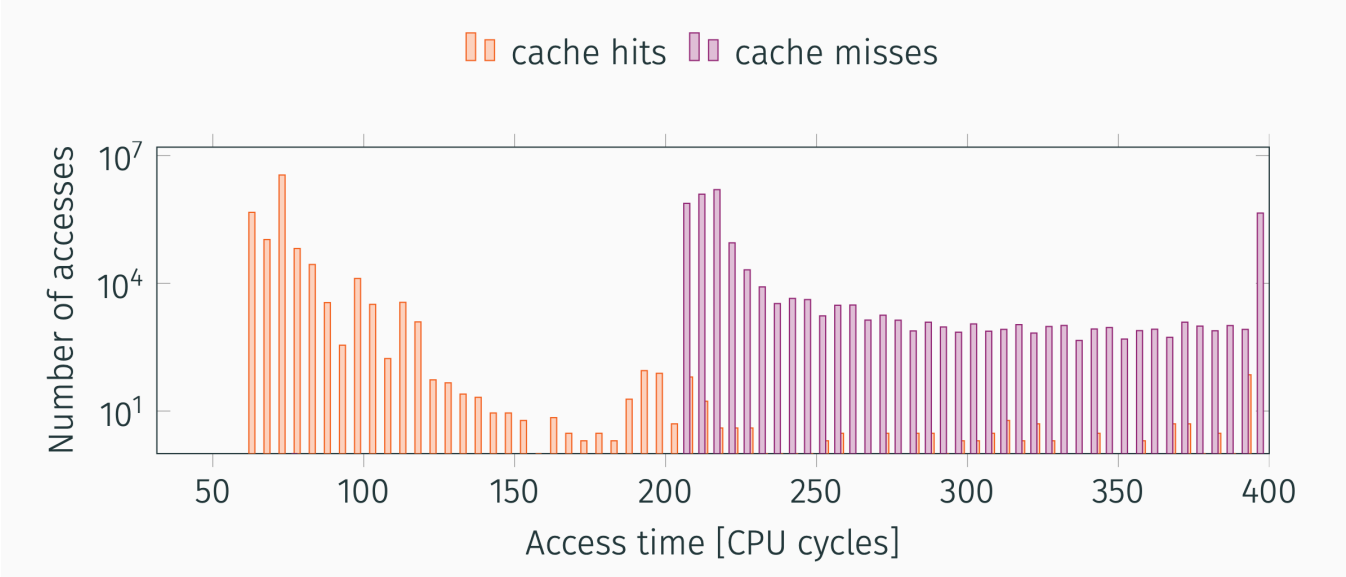
Build Histogram: Cache misses

1. Flush (clflush instructions)
2. Measure time
3. Access cache miss
4. Measure time
5. Update histogram



Determinging Threshold

A mediation point between cache hits and misses



Timing Accuracy

- How to measure very short timings?

1. Rdtsc instruction: cycle accurate timestamps
2. serializing instructions like cupid
3. fences like mfence

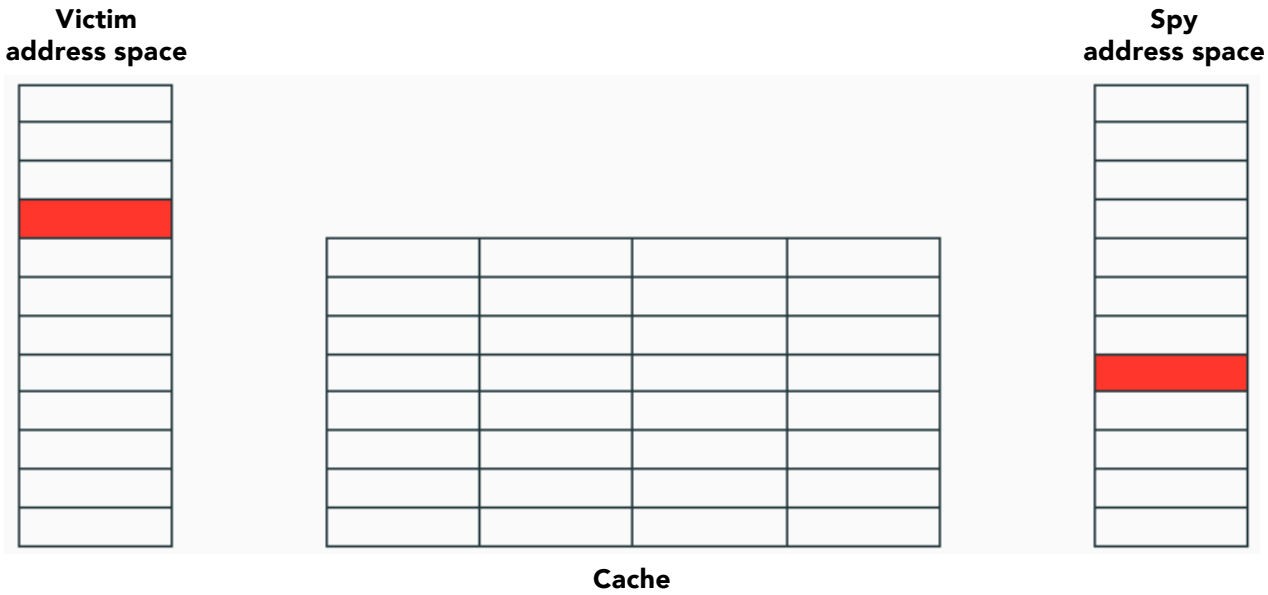
Cache-based SCAs

- Side-Channel Attacks based on Intel's x86 architecture's properties: Sharing & Inclusivity
- Exploitable on x86 and ARM
- Used for side-channel and covert attacks

① FLUSH + RELOAD
② Prime+Probe
③ Flush+Flush

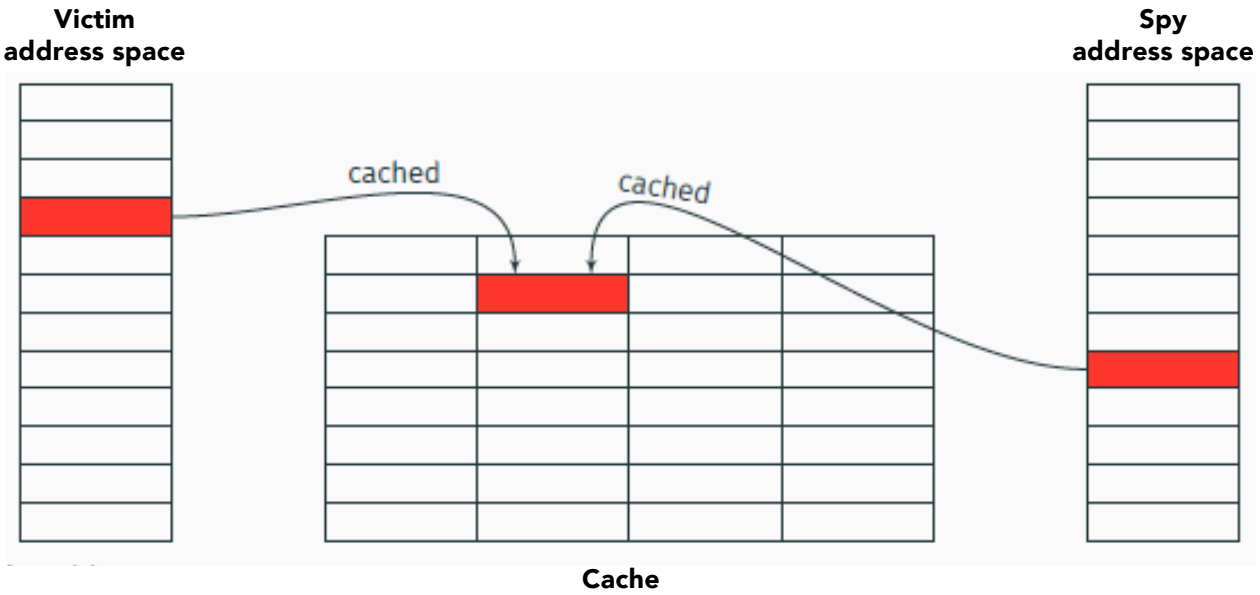
Cache-based SCAs

- Exploit timing differences of memory accesses
  - Attacker (process) monitors which lines are accessed by the Victim (process), and not the content!



Cache-based SCAs

- Intel's x86 sharing property
  - Attacker maps shared library (shared memory in cache)
  - Sharing allows SPY to look at VICTIM's (shared) address space

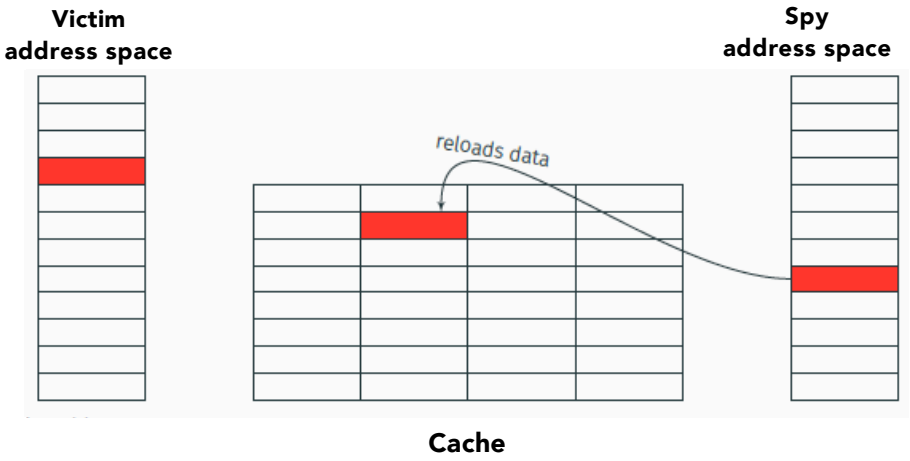


Cache-based SCAs

- Side-Channel Attacks on Intel's x86 architecture

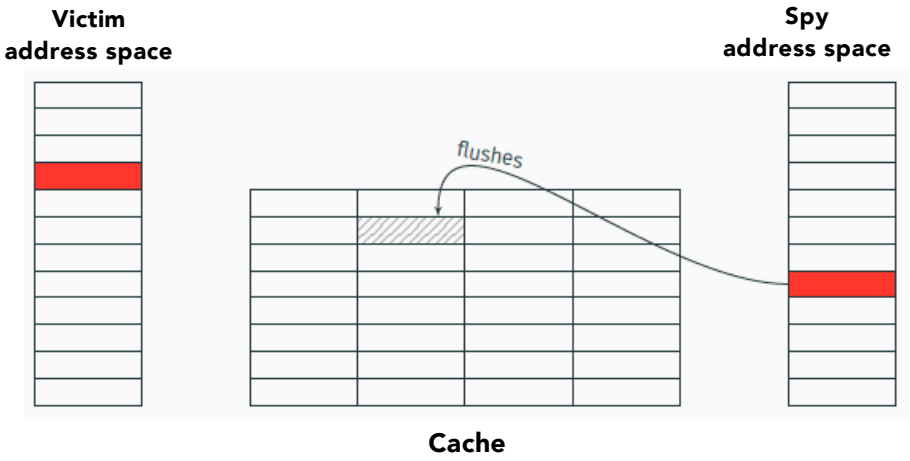
  ① FLUSH+RELOAD

- Spy maps shared library
- Spy flushes shared cache line
- Victim loads data
- Spy reloads data
- Spy measures timing in both cases (with & without cache line)



Cache-based SCAs

- Side-Channel Attacks on Intel's x86 architecture
  - ② FLUSH+FLUSH
    - Spy maps shared library
    - Spy flushes shared cache line
    - Victim loads data
    - Spy flushes the data again
    - Spy measures timing in both cases
      - Cache line hit
      - Cache line miss

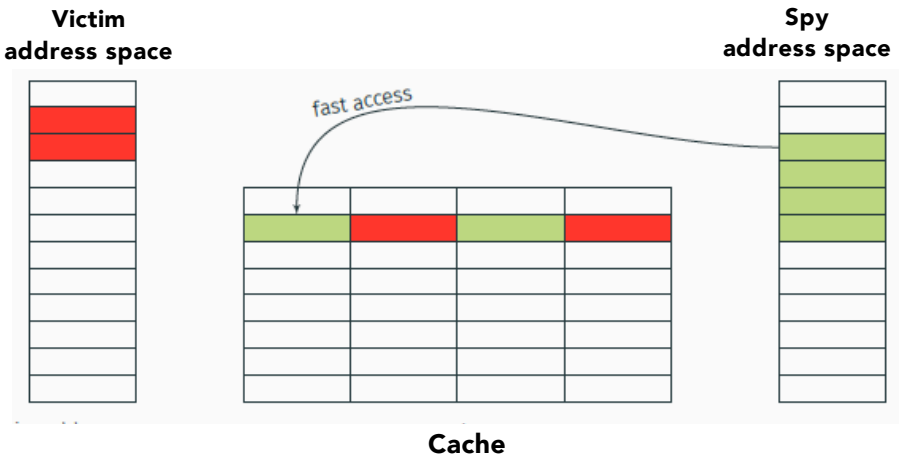


Cache-based SCAs

- Side-Channel Attacks on Intel's x86 architecture

  ③ PRIME+PROBE

  - Spy fills cache lines
  - Victim flushes cache lines while running
  - Spy probes data to determine if set is being accessed or not
  - Spy measures timing in both cases
    - Cache line hit
    - Cache line miss



Computational Attacks

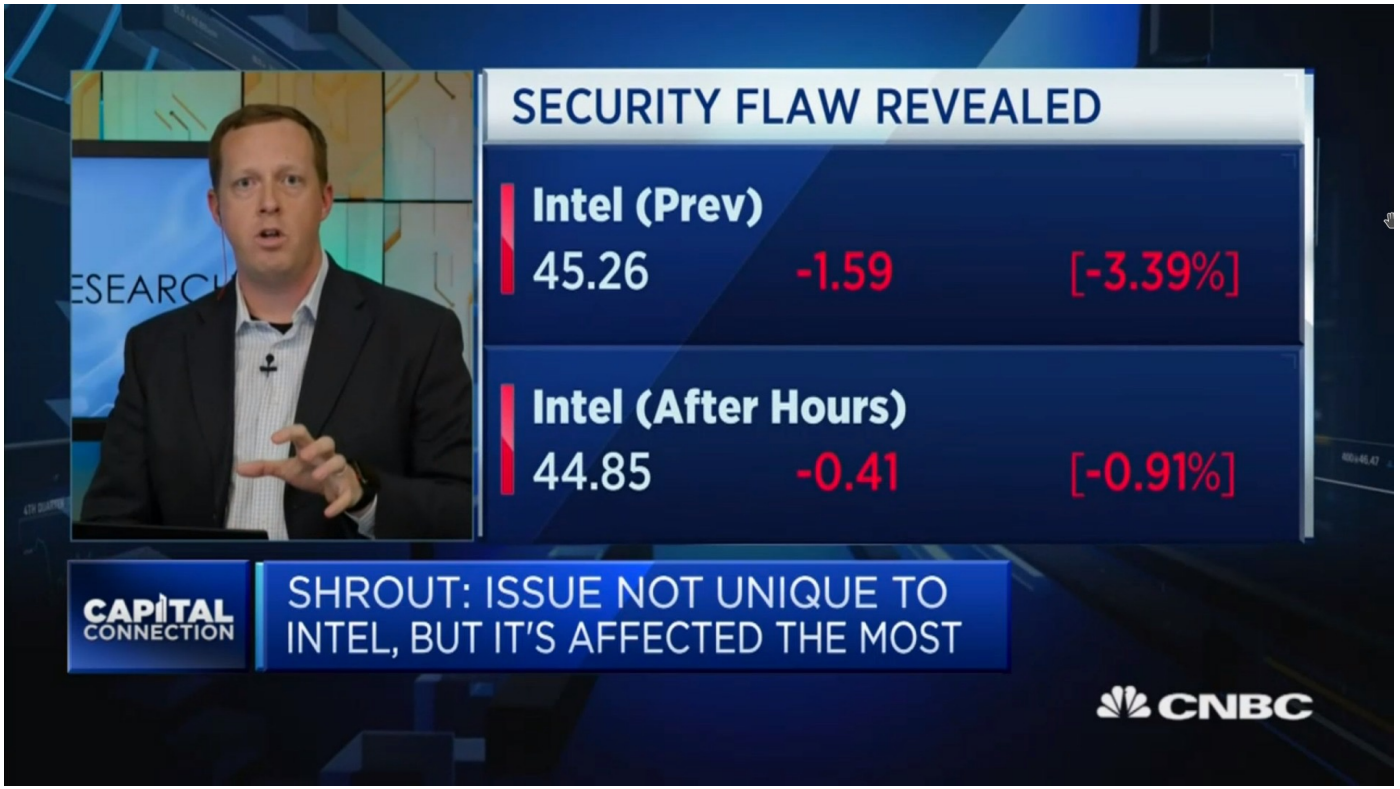
[Spectre & Meltdown Attacks]

Computational Attacks

Computational Attacks

Computational Attacks

Computational Attacks



Computational Attacks

- What Are We Talking About?

Meltdown

- Out-of-Order Execution
- Cache SCAs

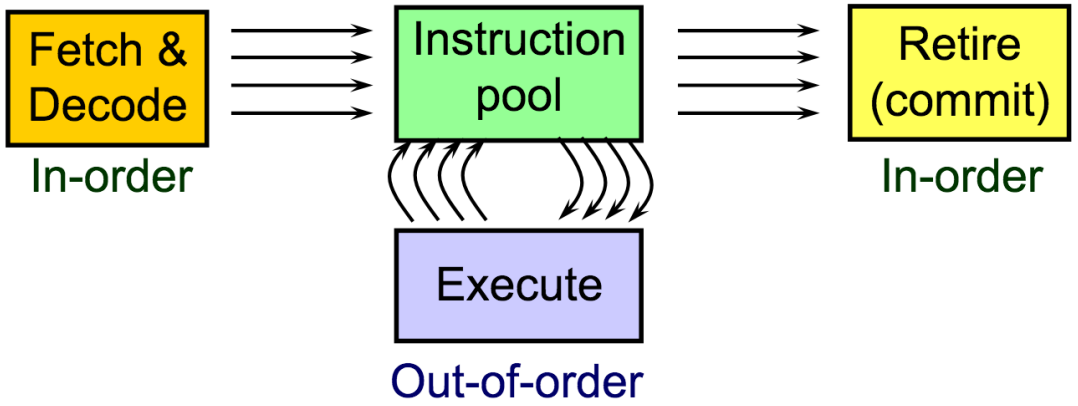
Spectre

- Branch Prediction
- Cache SCAs

- Two CPU vulnerabilities discovered in 2018!
- Both exploit performance enhancement techniques

Computational Attacks

- Meltdown Attack
  - Vulnerability: Permission check for address is done in parallel & out-of-order to the load instruction! Potential Race Condition

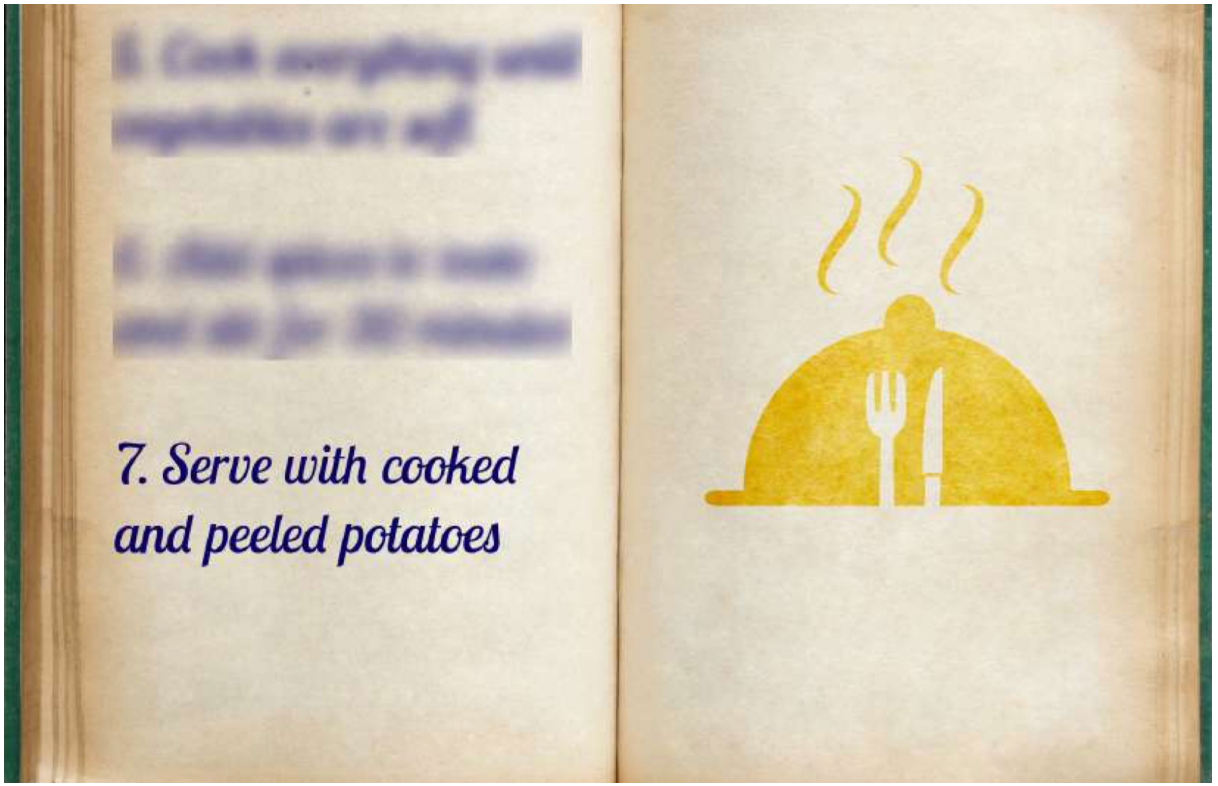


Computational Attacks

- Meltdown Attack

Computational Attacks

- Meltdown Attack



Computational Attacks

- Meltdown Attack

Computational Attacks

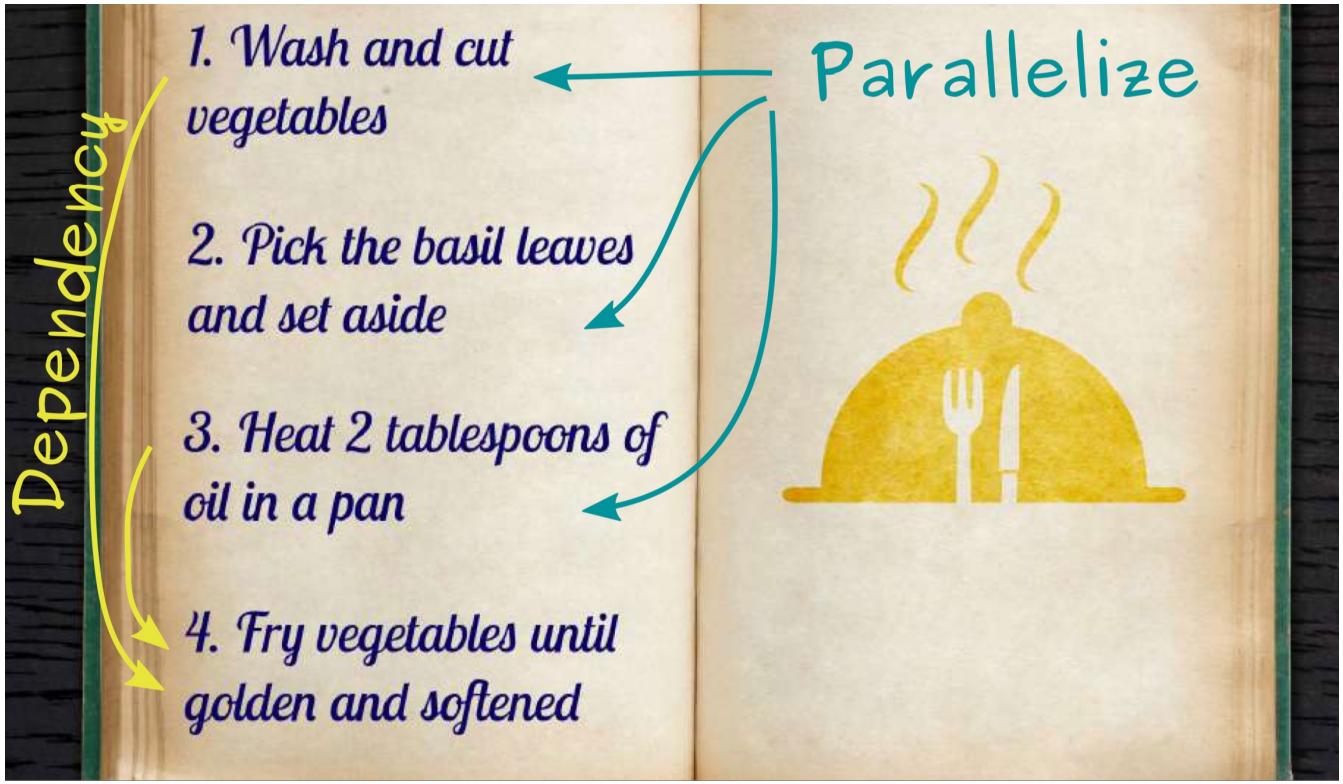
- Meltdown Attack

Computational Attacks

- Meltdown Attack

Computational Attacks

- Meltdown Attack



Computational Attacks

- Meltdown Attack

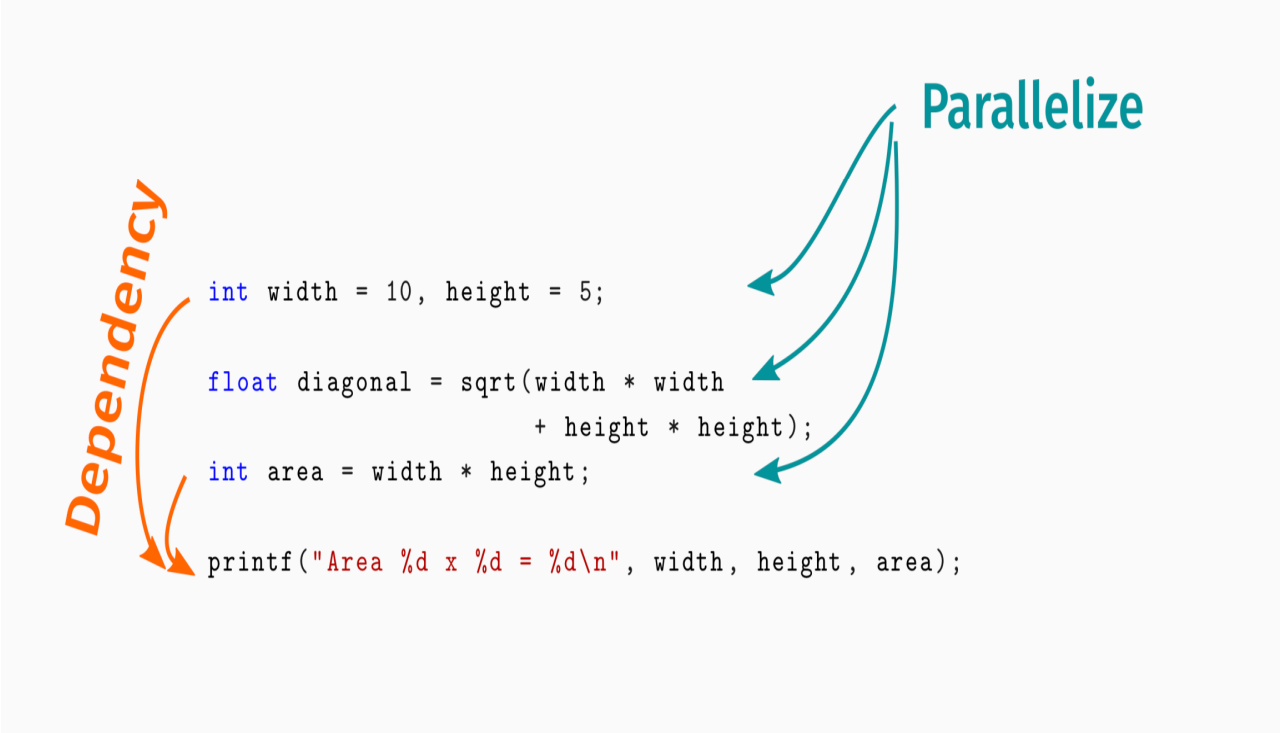


Figure: Courtesy Daniel Gruss et al.

Computational Attacks

Meltdown Attack

- Vulnerability: Permission check for address is done in parallel & out-of-order to the load instruction! Potential Race Condition

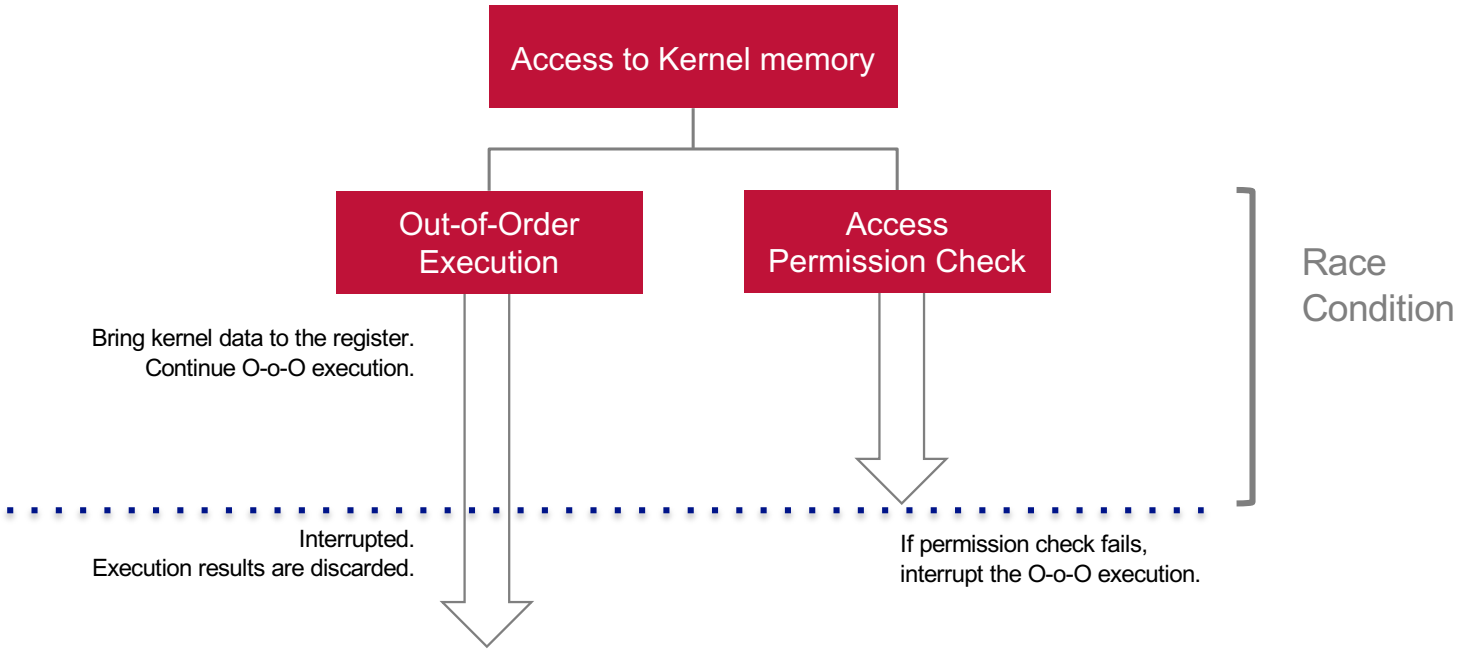


Figure: Courtesy Daniel Gruss et al.

Computational Attacks

- Spectre Attack

Computational Attacks

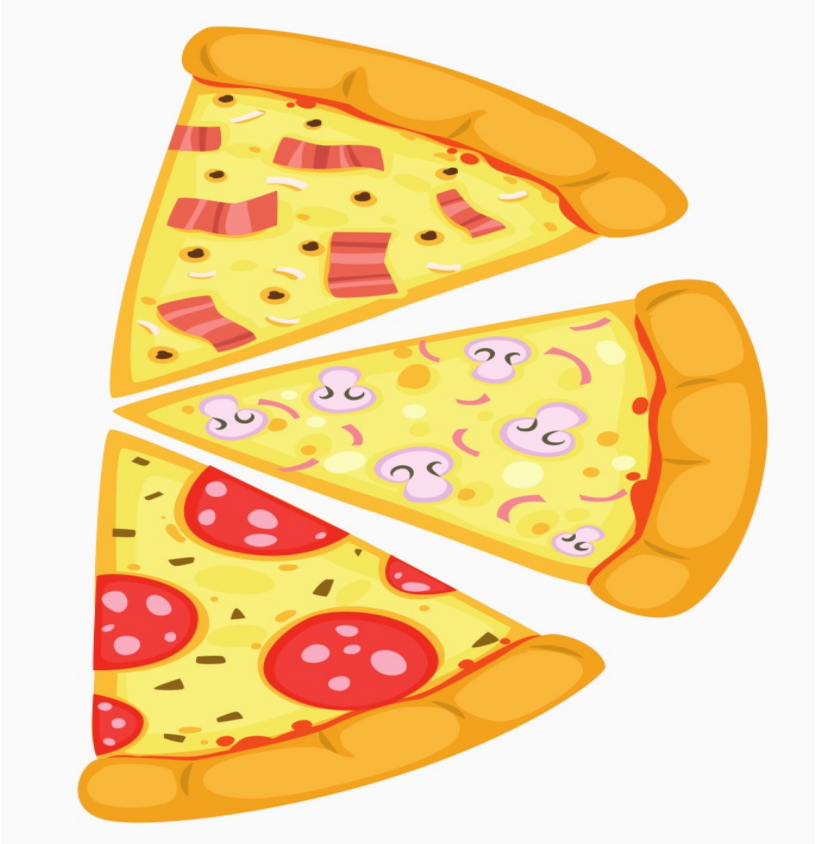
- Spectre Attack

Computational Attacks

- Spectre Attack

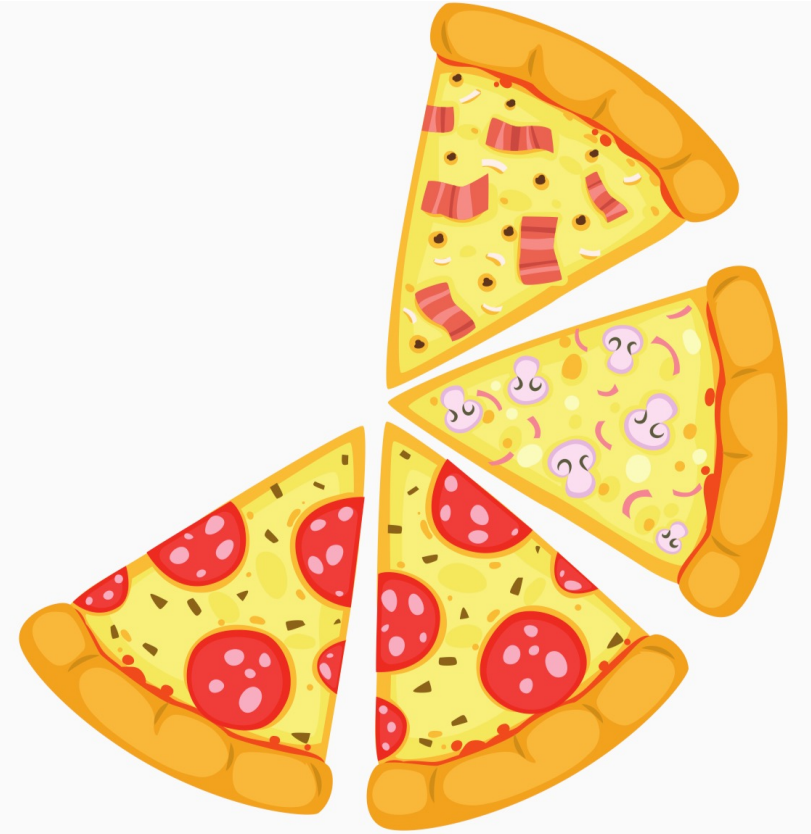
Computational Attacks

- Spectre Attack



Computational Attacks

- Spectre Attack



Computational Attacks

- Spectre Attack

Computational Attacks

- Spectre Attack

Computational Attacks

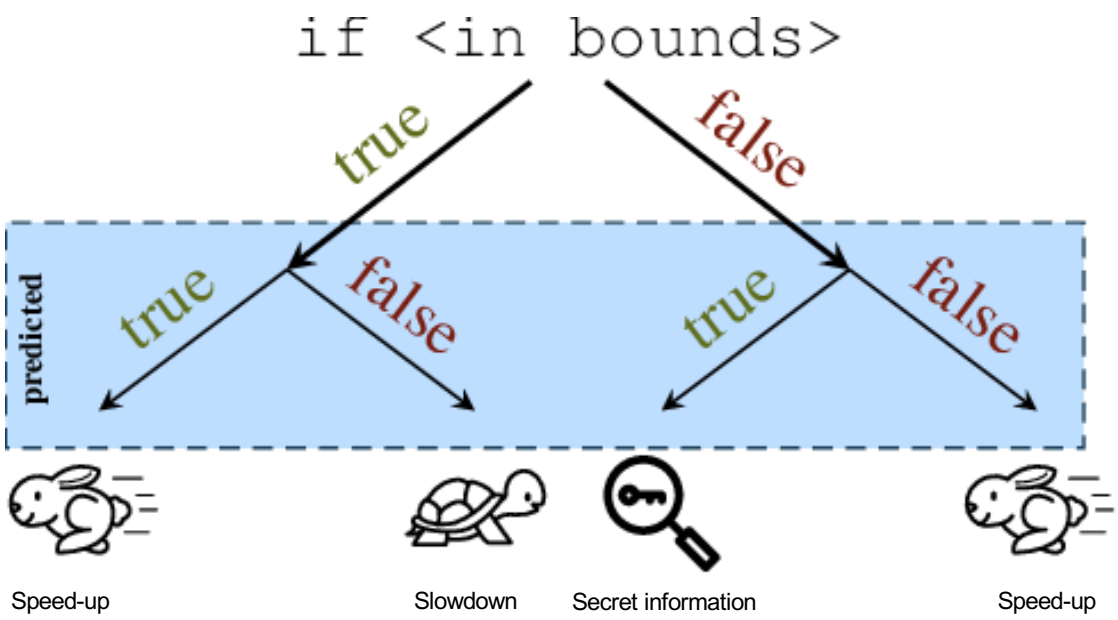
- Spectre Attack

Computational Attacks

- Spectre Attack

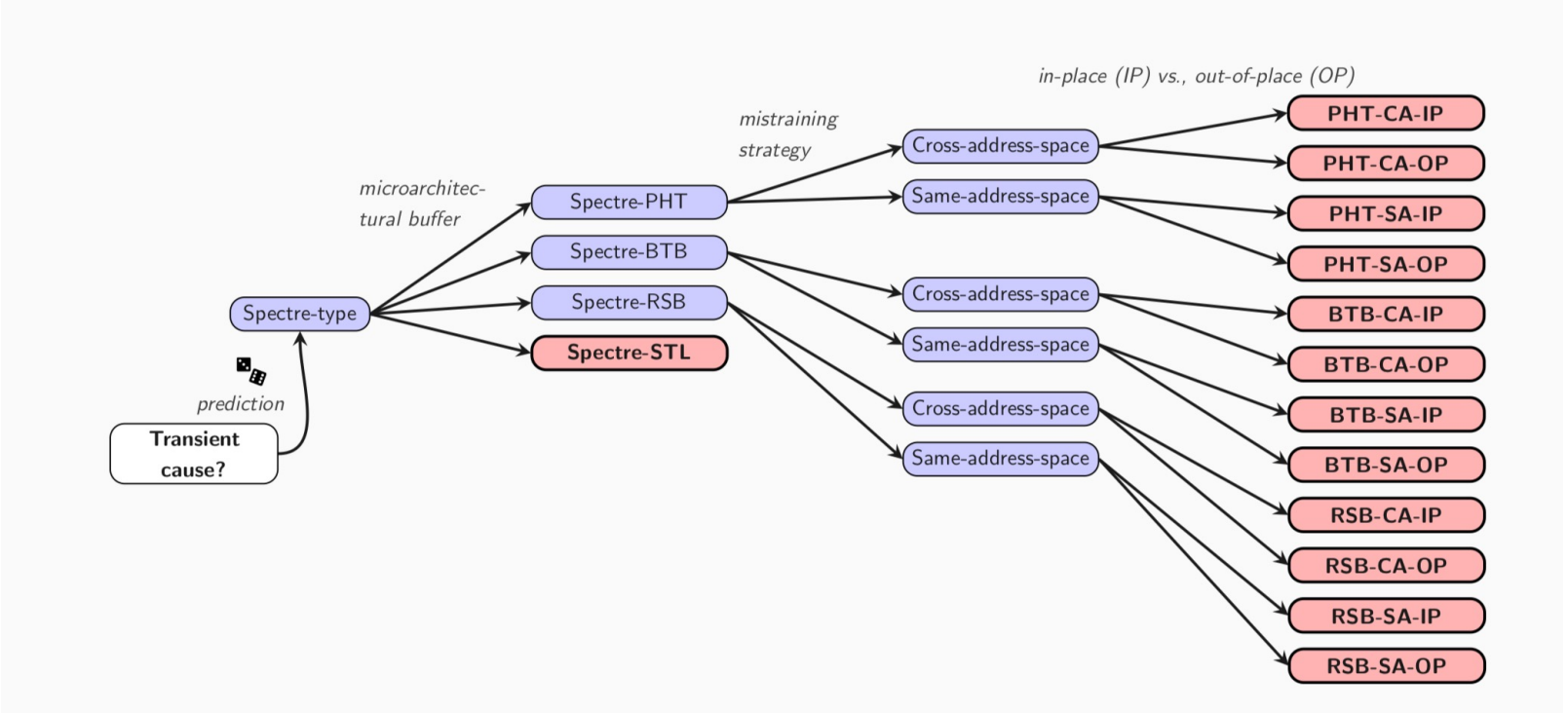
Computational Attacks

- Spectre Attack
  - Vulnerability: Speculative execution of branches
  - Miss-trains Branch Prediction to convince CPU to speculatively execute code that should not be executed



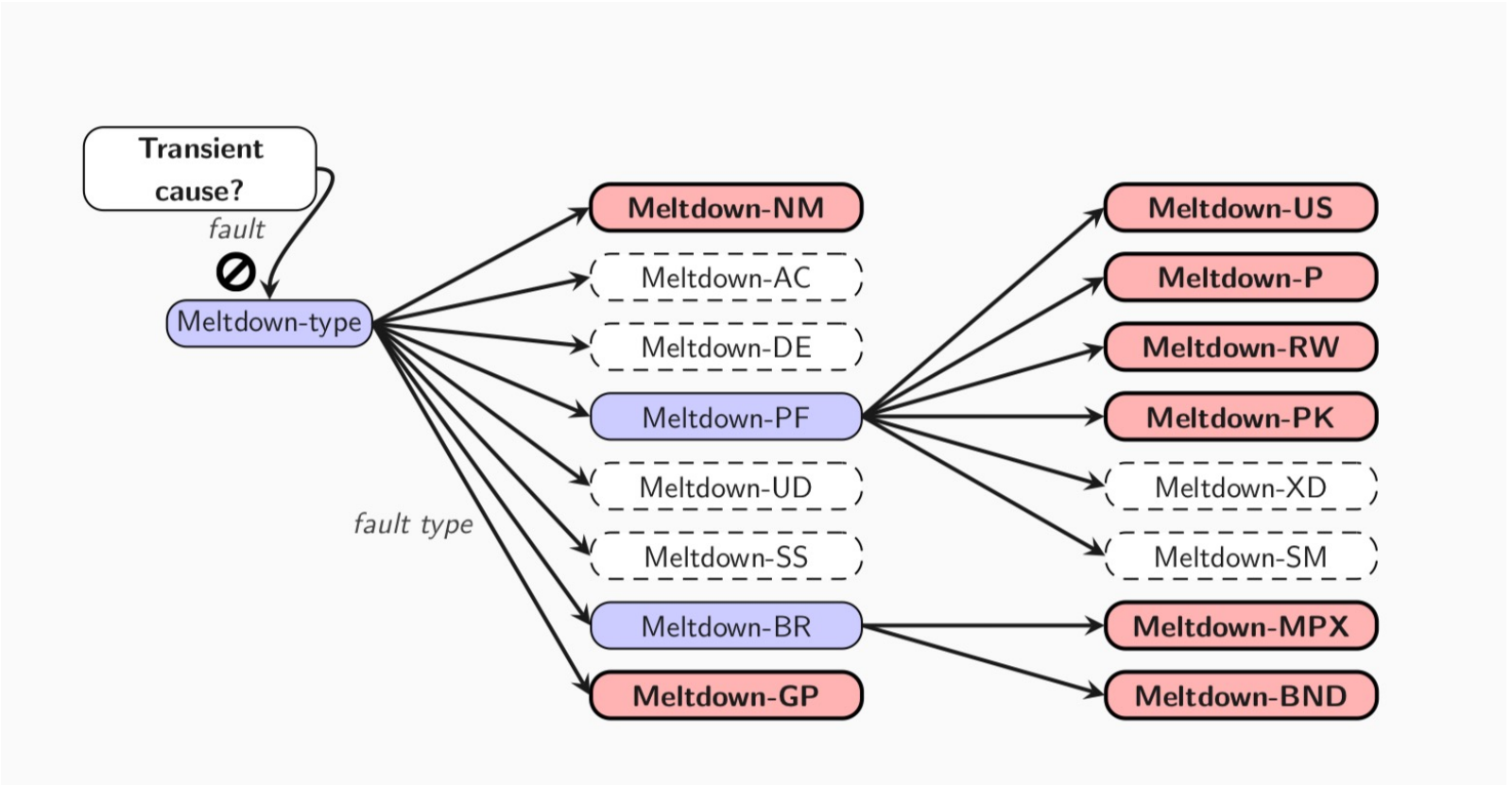
Computational Attacks

- Variants-For our knowledge



Computational Attacks

- Variants-For our knowledge



Conclusions

- Microarchitectural Attacks are a serious threat to computing
  - Crypto and non-crypto applications are under threat
  - RSA and AES implementations can be attacked
  - Does not mean that AES and RSA are broken
- Side Channel Attacks use shared and vulnerable hardware
  - Every memory access should take the same time
  - Hardware components should not be shared
  - Extra microarchitectural states should be cleaned

Quiz –Student Evaluation

Quiz

- Spectre takes benefit of which performance optimization technique?
- What is out-of-order execution?
- How cache hit and cache miss are important for attacker to mount attack?