Malicious Management Unit

Why Stopping Cache Attacks in Software is Harder Than You Think

Stephan van Schaik

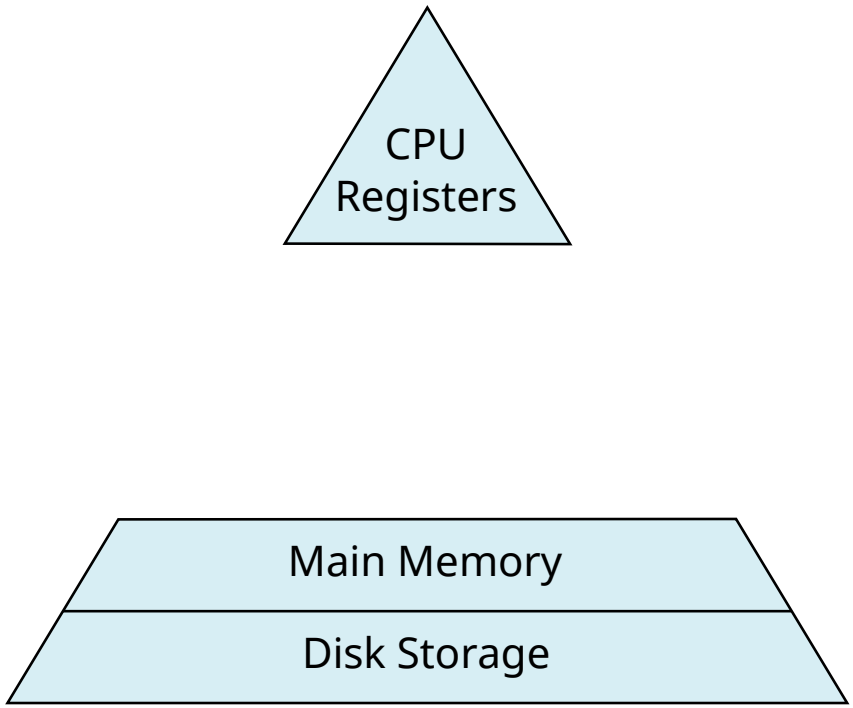
Cristiano Giuffrida Herbert Bos Kaveh Razavi

VU VRIJE UNIVERSITEIT AMSTERDAM

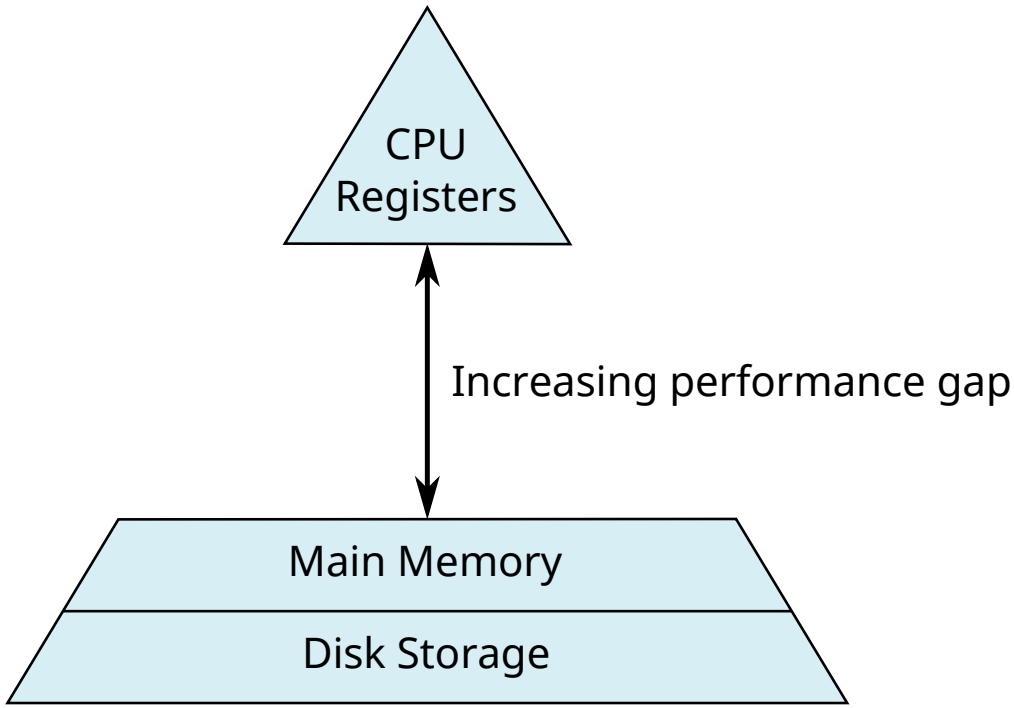
Motivation

Why should you care about cache attacks?

Motivation

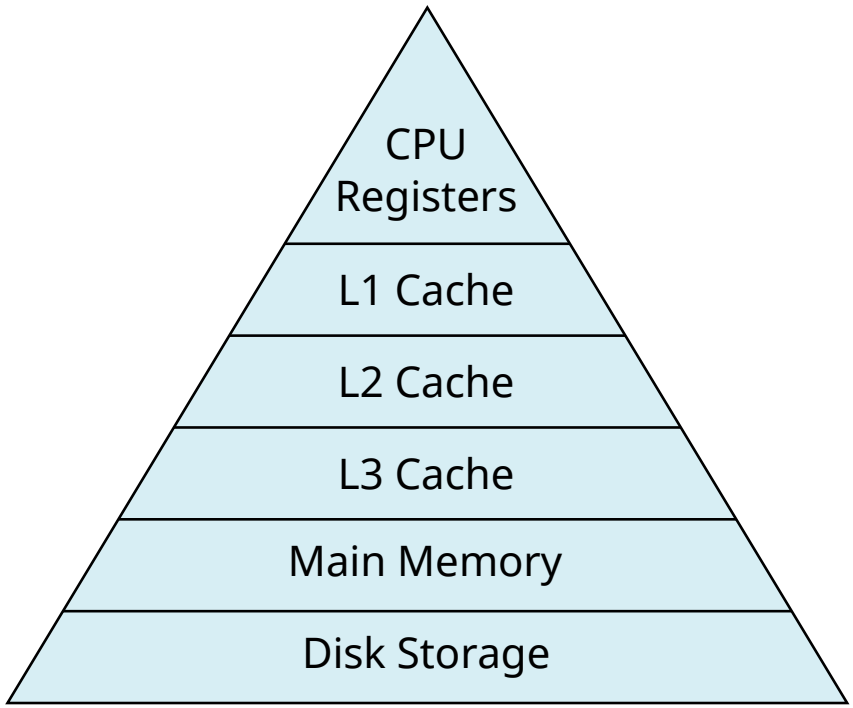


Motivation

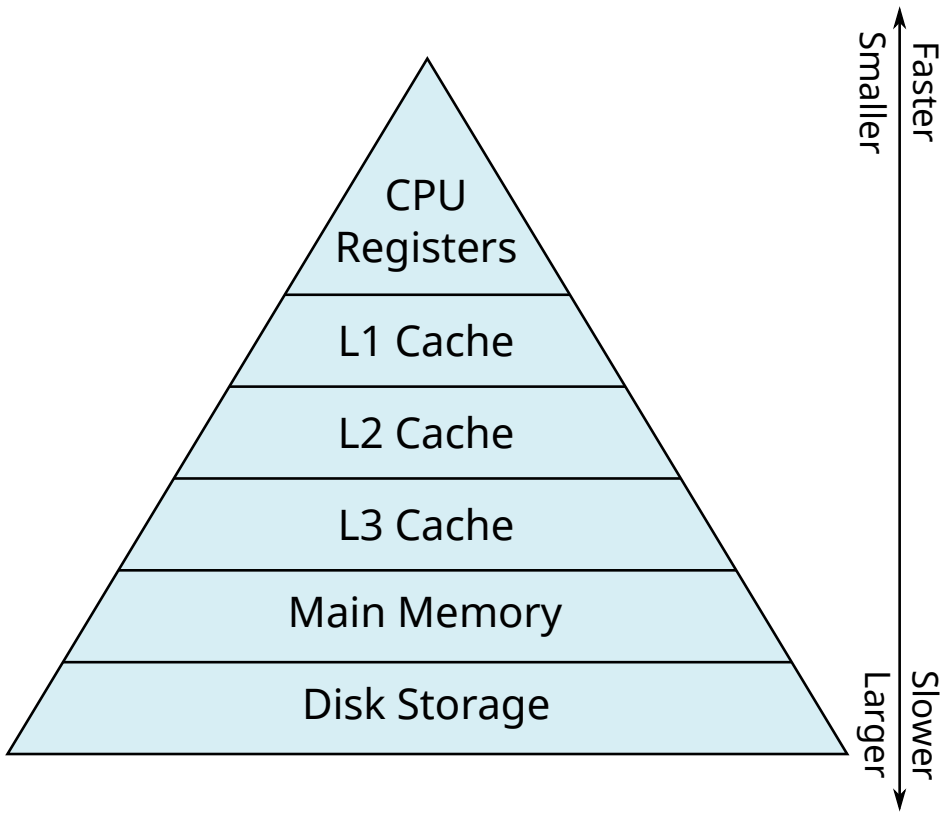


Processors advance faster than memory

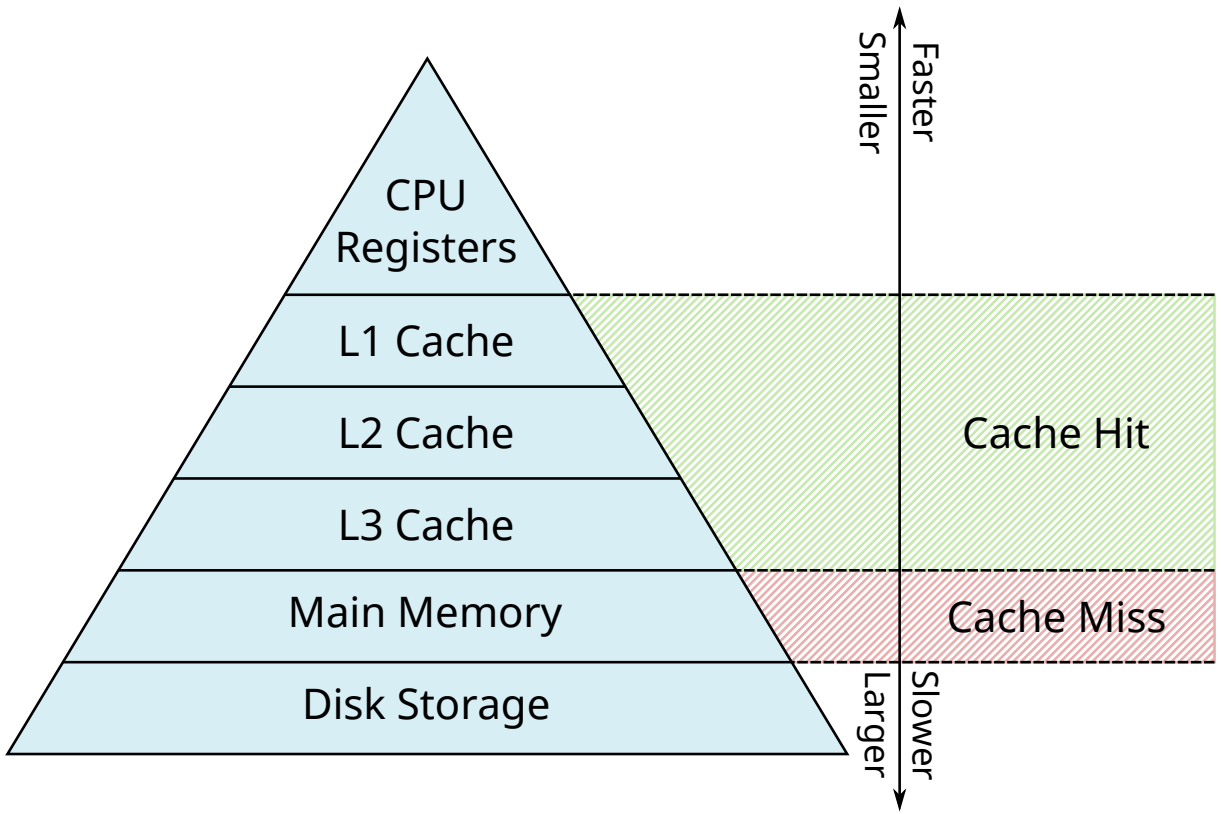
Motivation


CPU
Registers
L1 Cache
L2 Cache
L3 Cache
Main Memory
Disk Storage

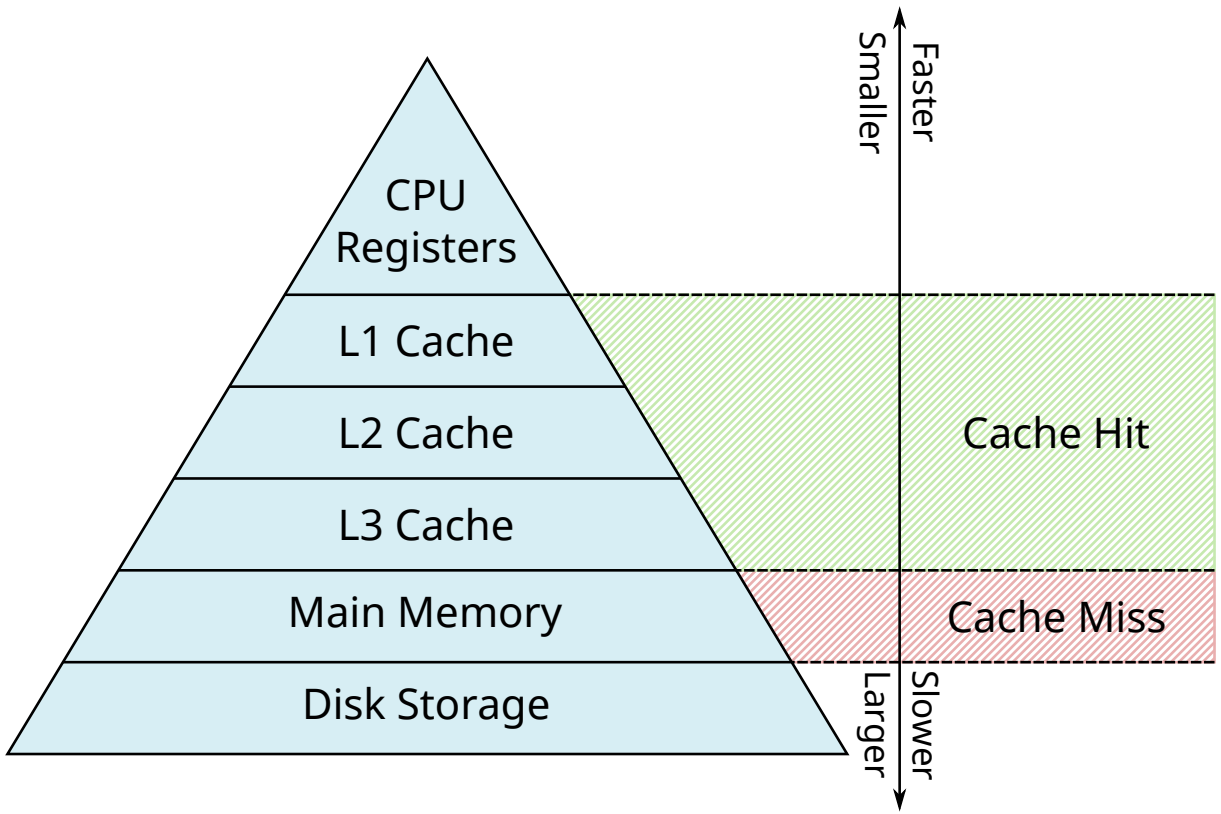
Motivation


CPU
Registers
L1 Cache
L2 Cache
L3 Cache
Main Memory
Disk Storage
Faster
Smaller
Slower
Larger

Motivation


CPU
Registers
L1 Cache
L2 Cache
L3 Cache
Main Memory
Disk Storage
Smaller
Faster
Larger
Slower
Cache Hit
Cache Miss

Motivation

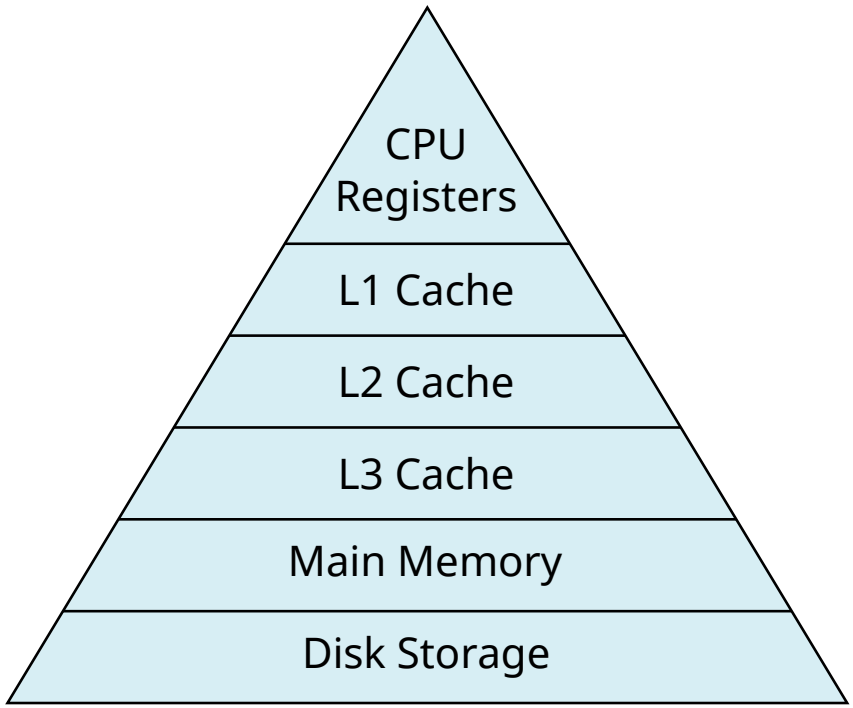


Memory accesses are not performed in constant time

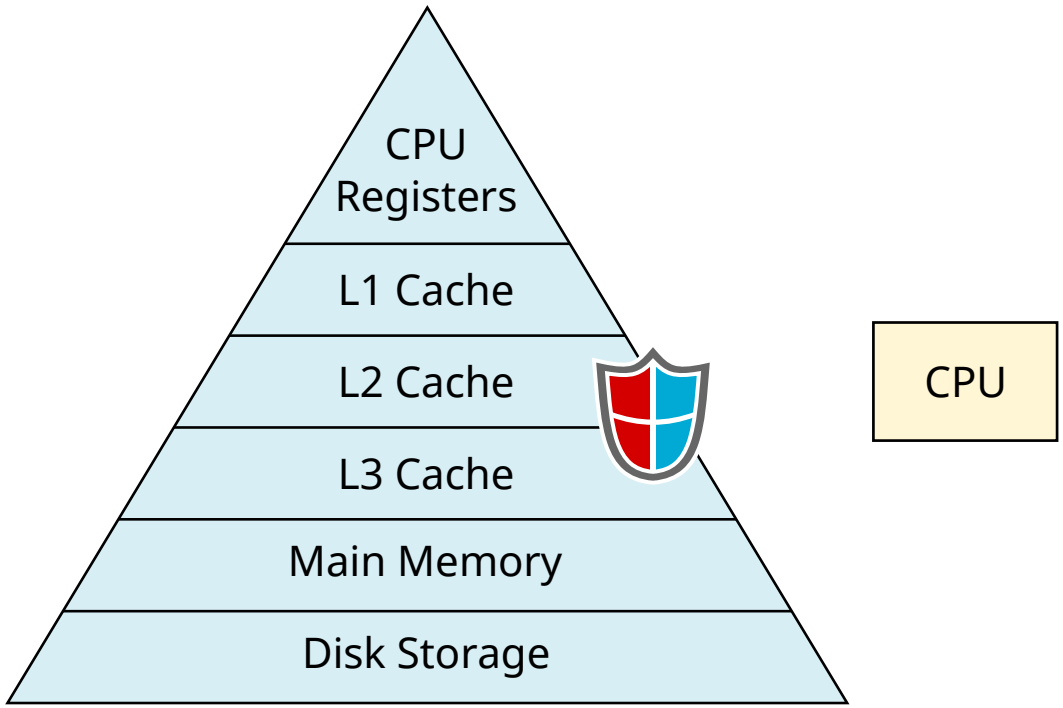
Motivation

- Caches are shared resources
- Caches can be manipulated
- Spy on other processes
- Input events
- Leak sensitive data

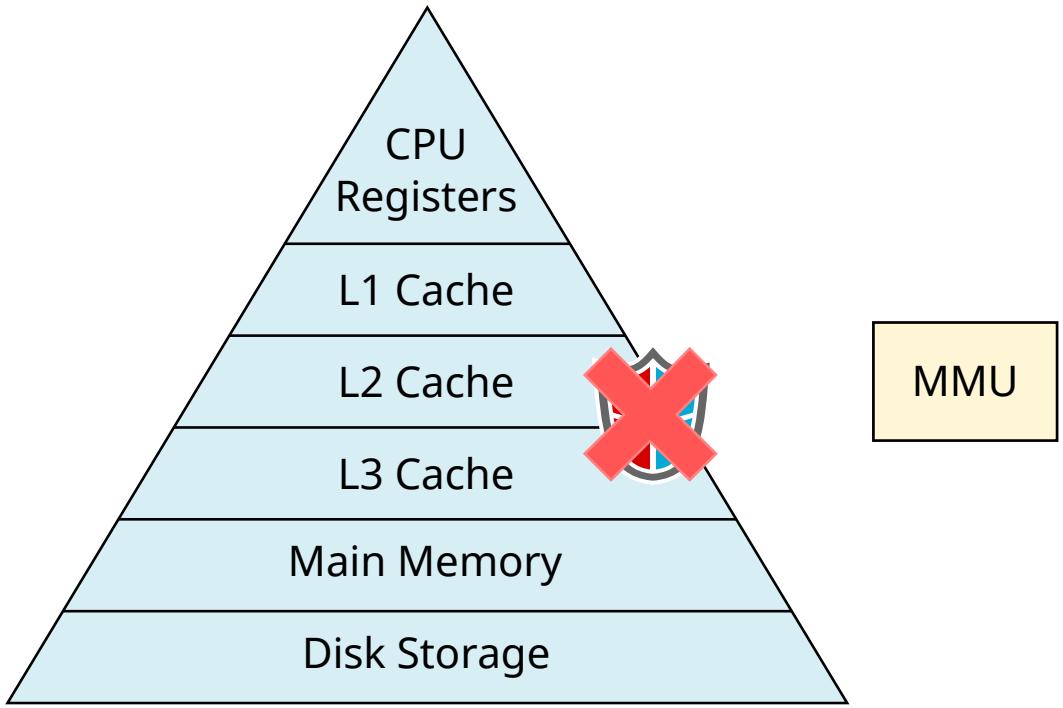
Motivation


CPU
Registers
L1 Cache
L2 Cache
L3 Cache
Main Memory
Disk Storage

Motivation


CPU
Registers
L1 Cache
L2 Cache
L3 Cache
Main Memory
Disk Storage
CPU

Motivation


CPU
Registers
L1 Cache
L2 Cache
L3 Cache
Main Memory
Disk Storage
MMU

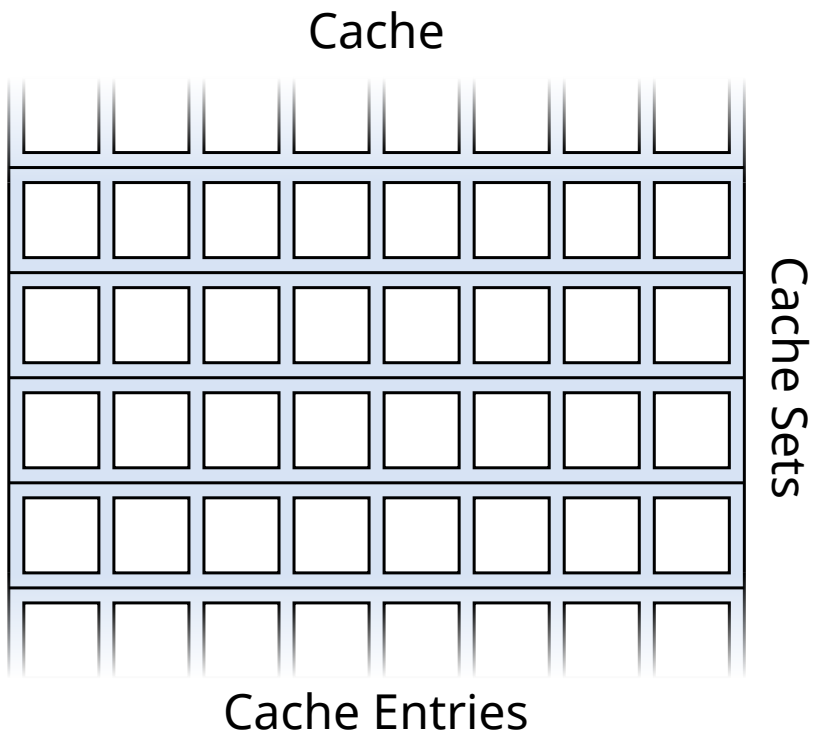
AES

- Advanced Encryption Standard
- Software implementations use T-tables
- $T[p_i \oplus k_i]$
- Indices are key-dependent
- Elements may be in main memory or the cache

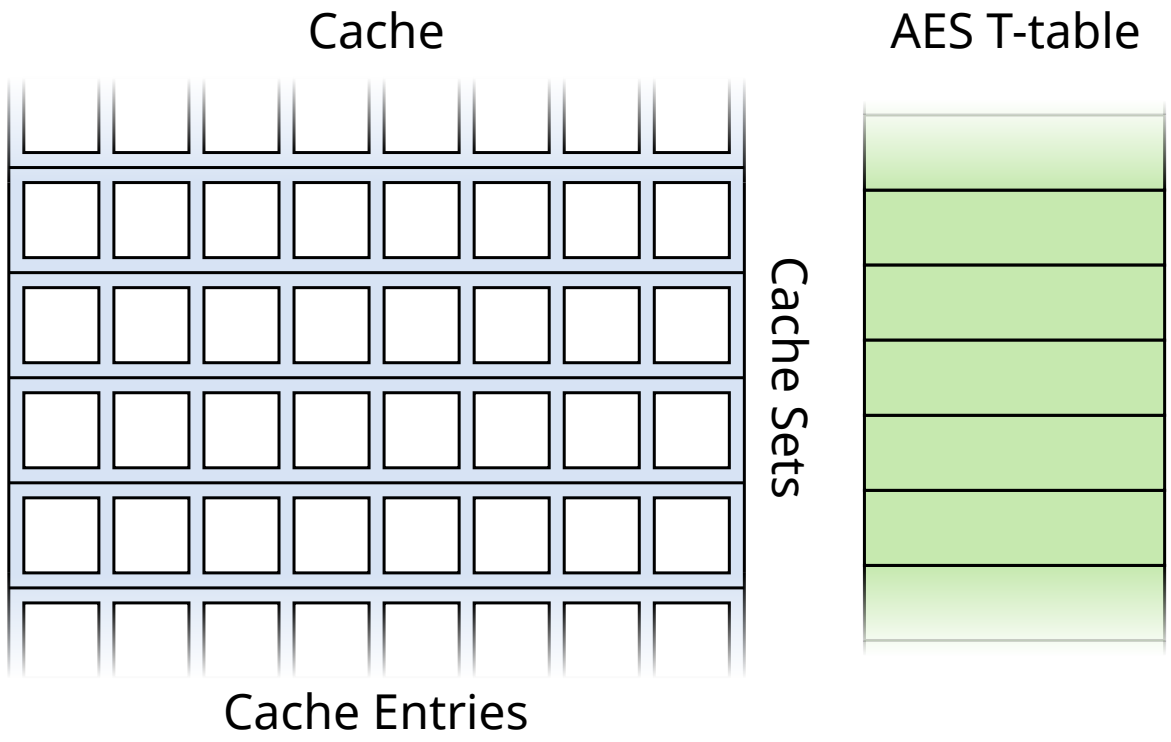
PRIME + PROBE

An example of PRIME + PROBE against AES encryption

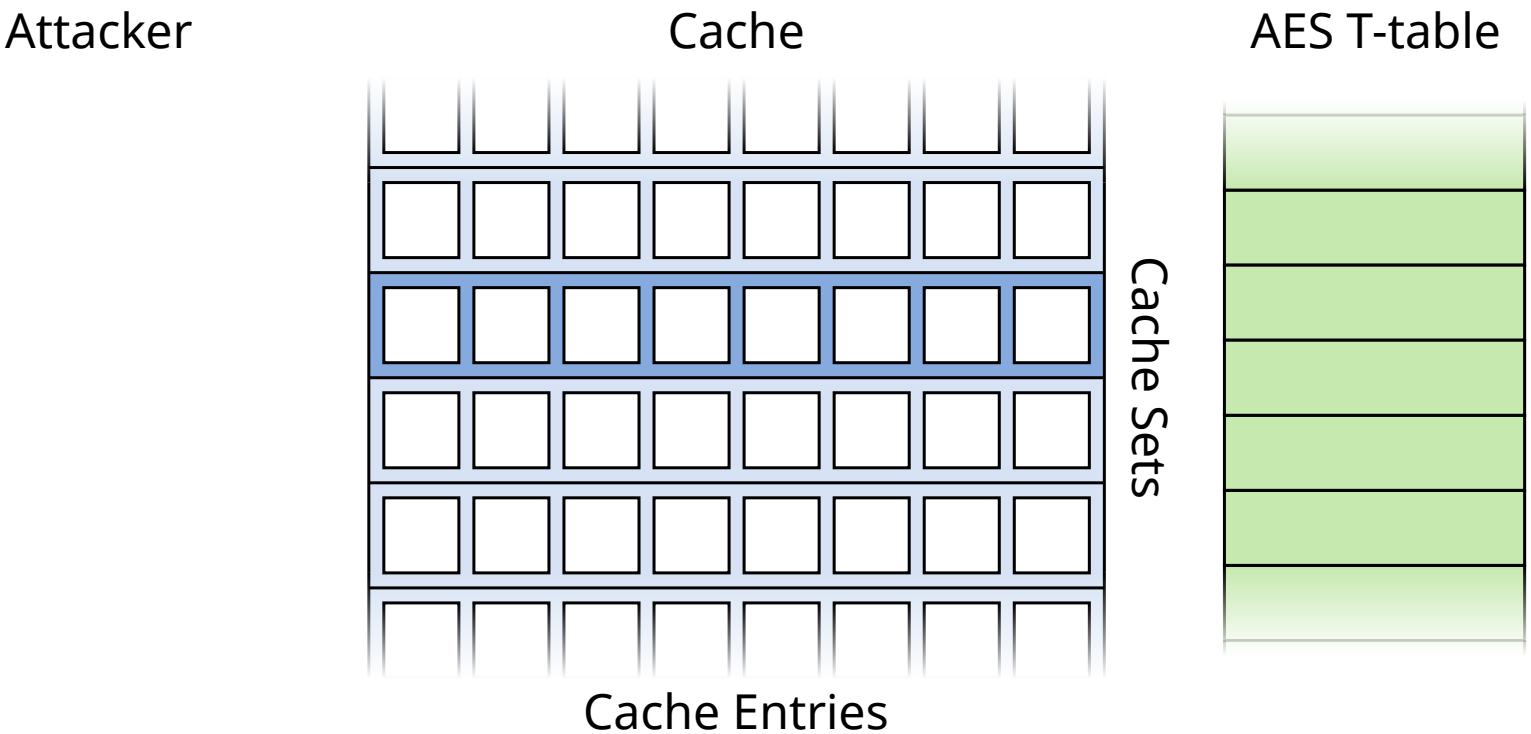
PRIME + PROBE



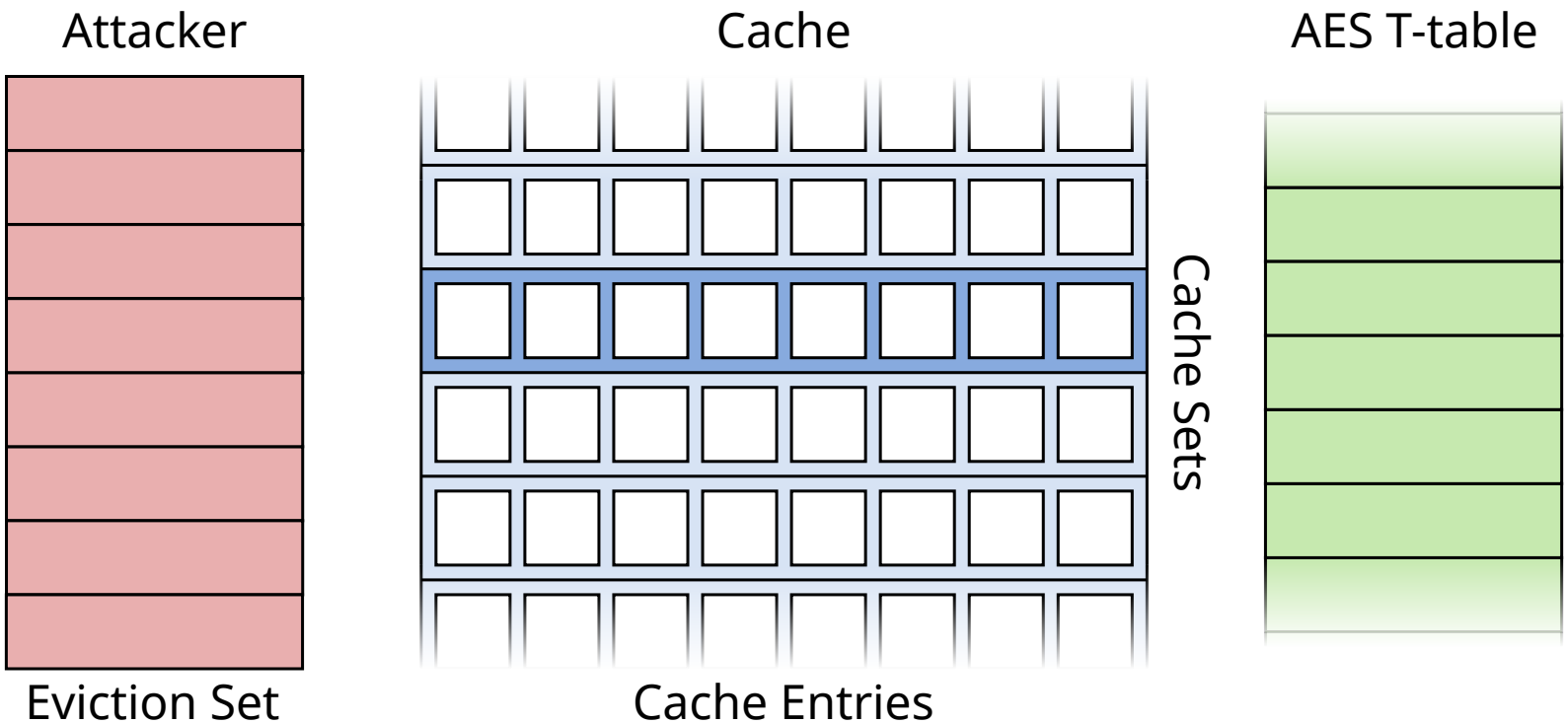
PRIME + PROBE



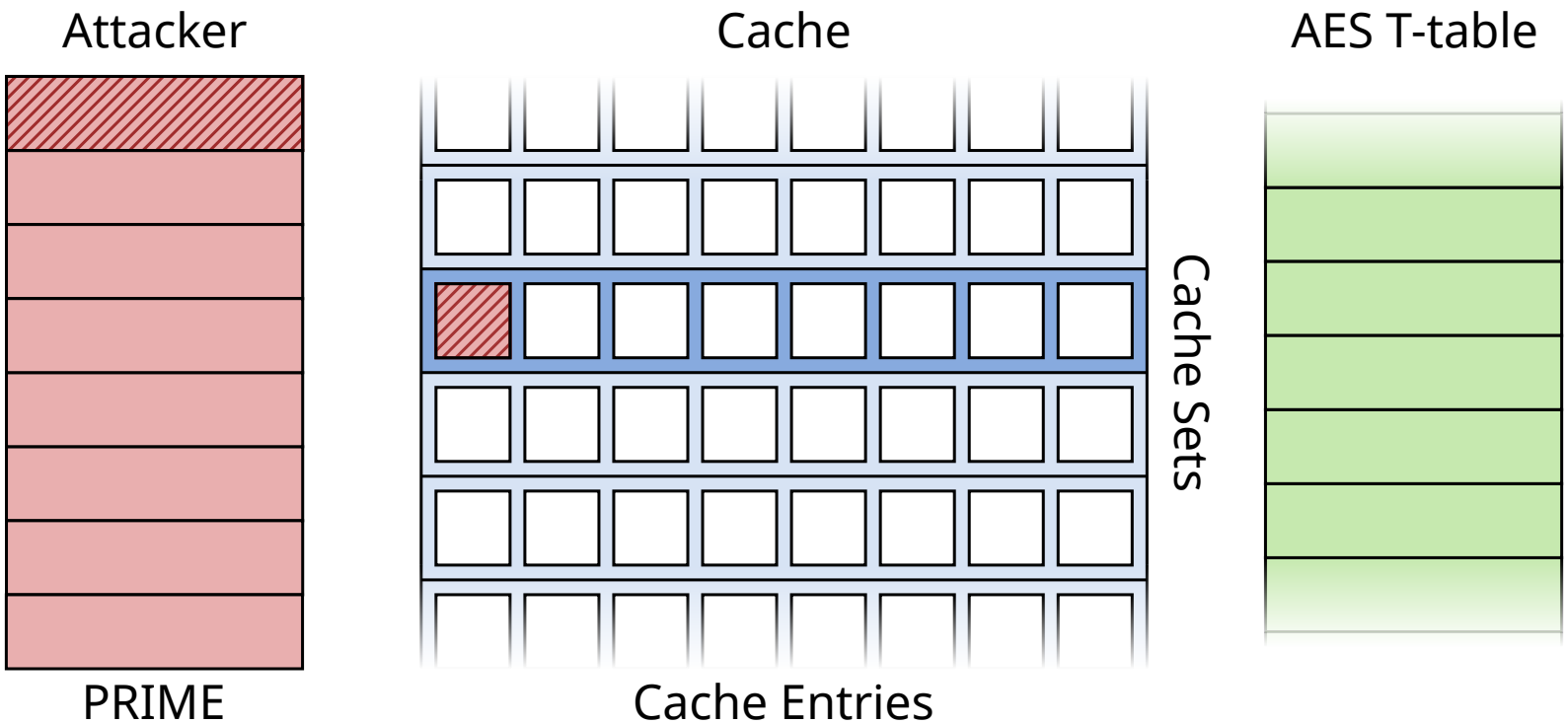
PRIME + PROBE



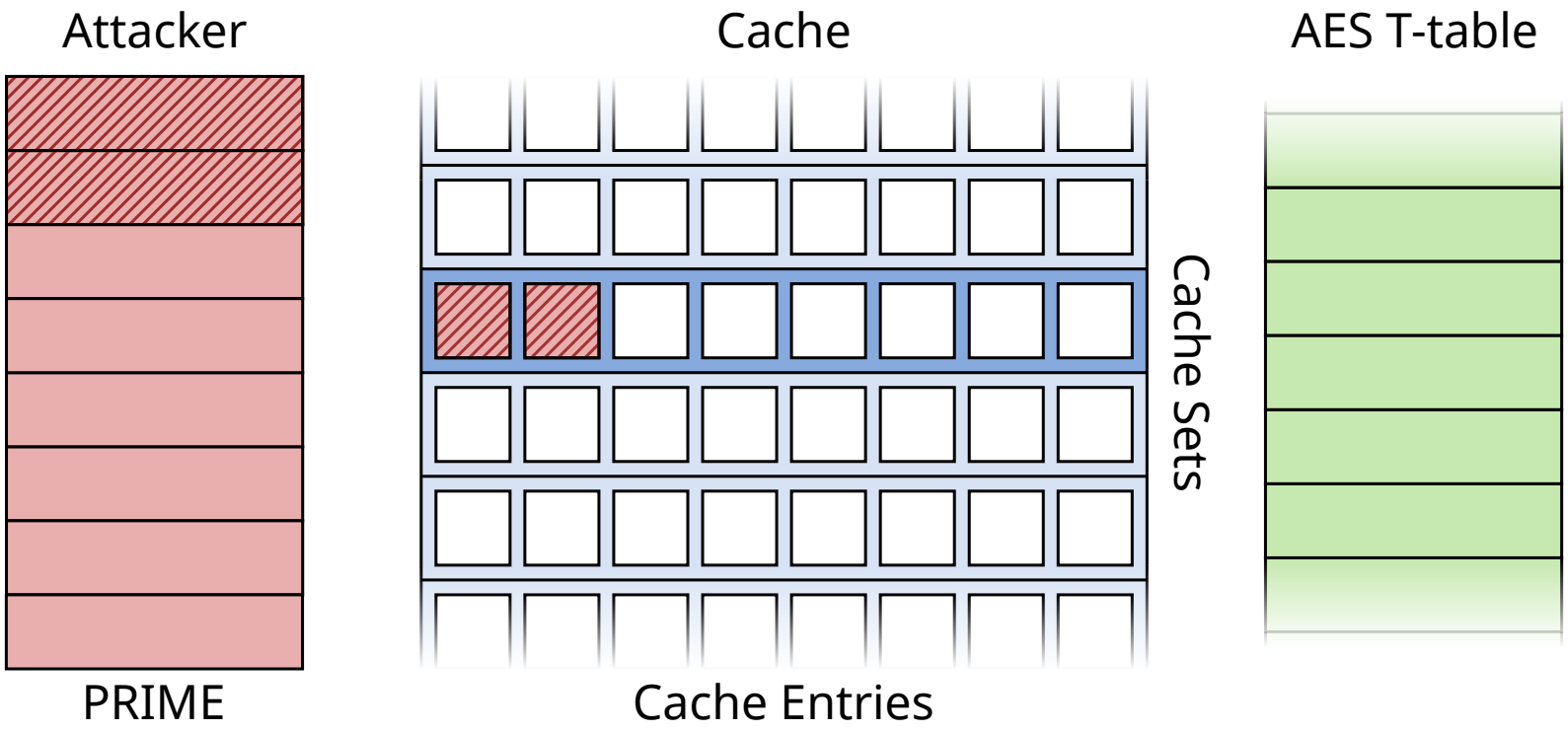
PRIME + PROBE


Attacker
Cache
AES T-table
Cache Sets
Eviction Set
Cache Entries

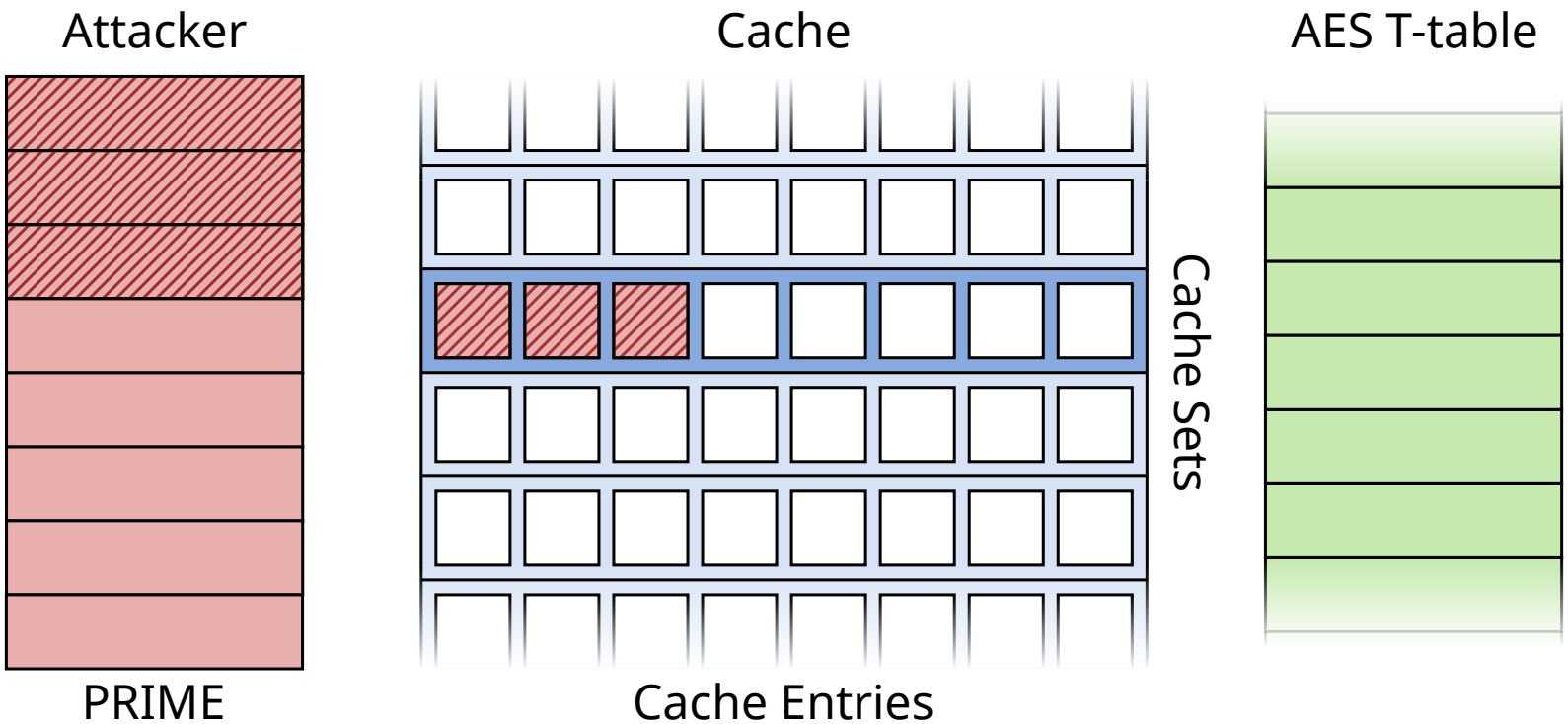
PRIME + PROBE



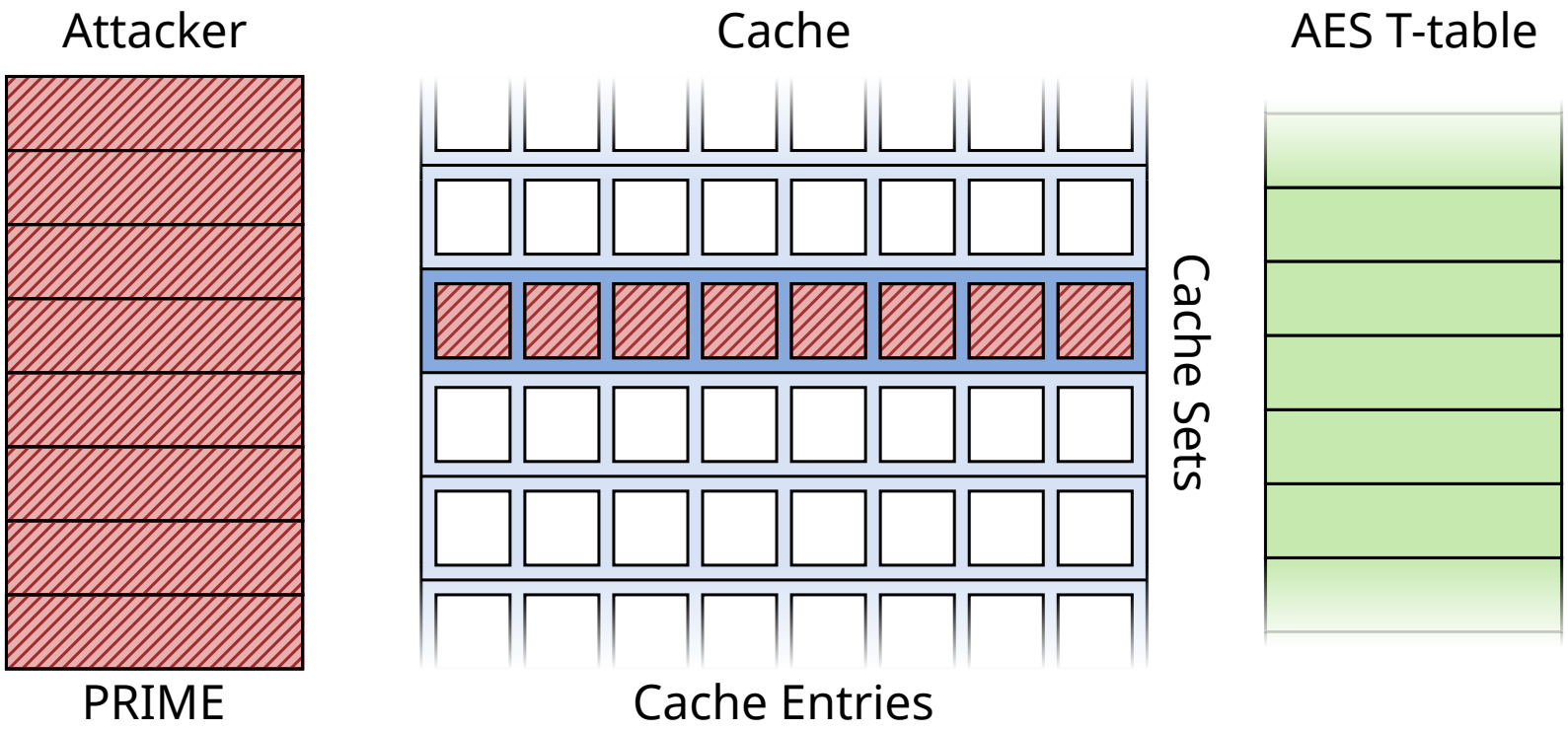
PRIME + PROBE



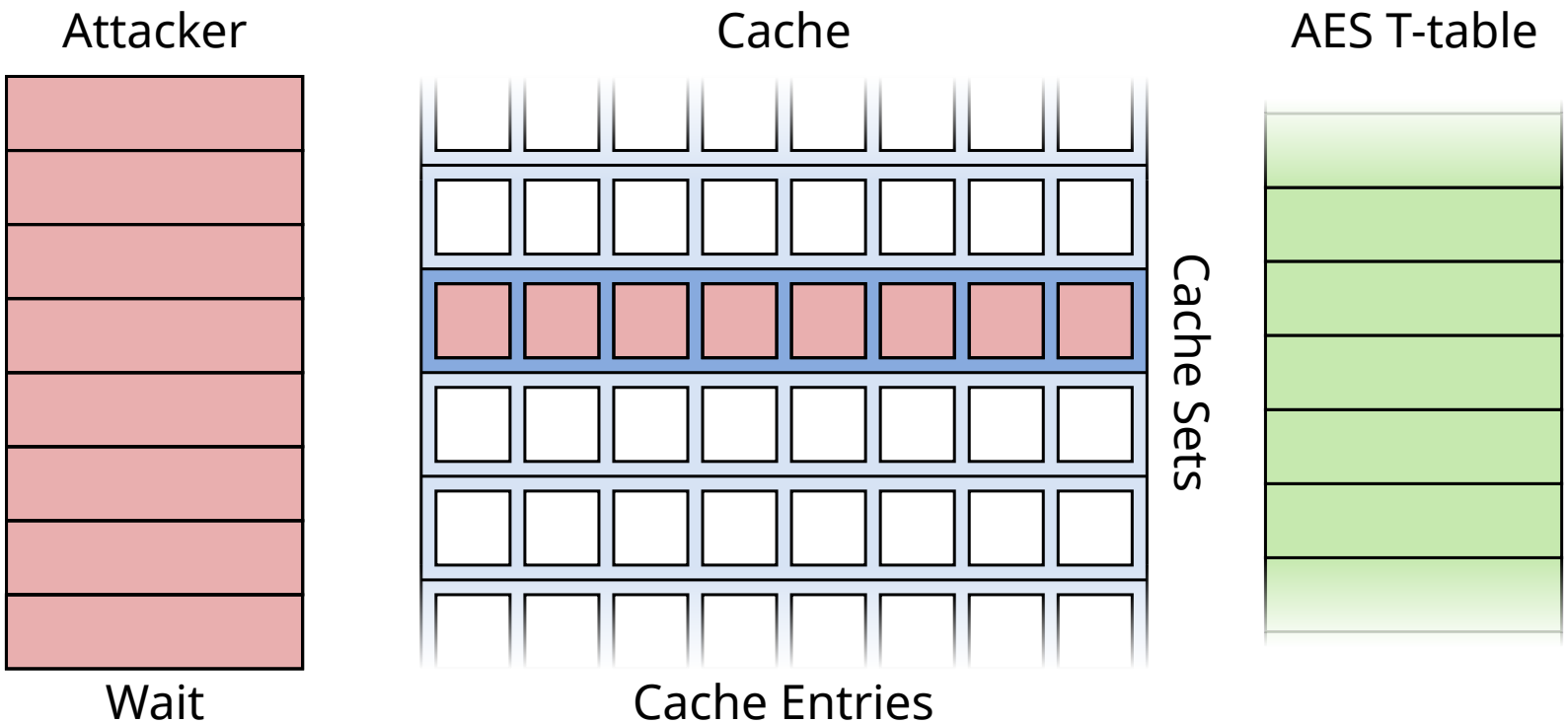
PRIME + PROBE



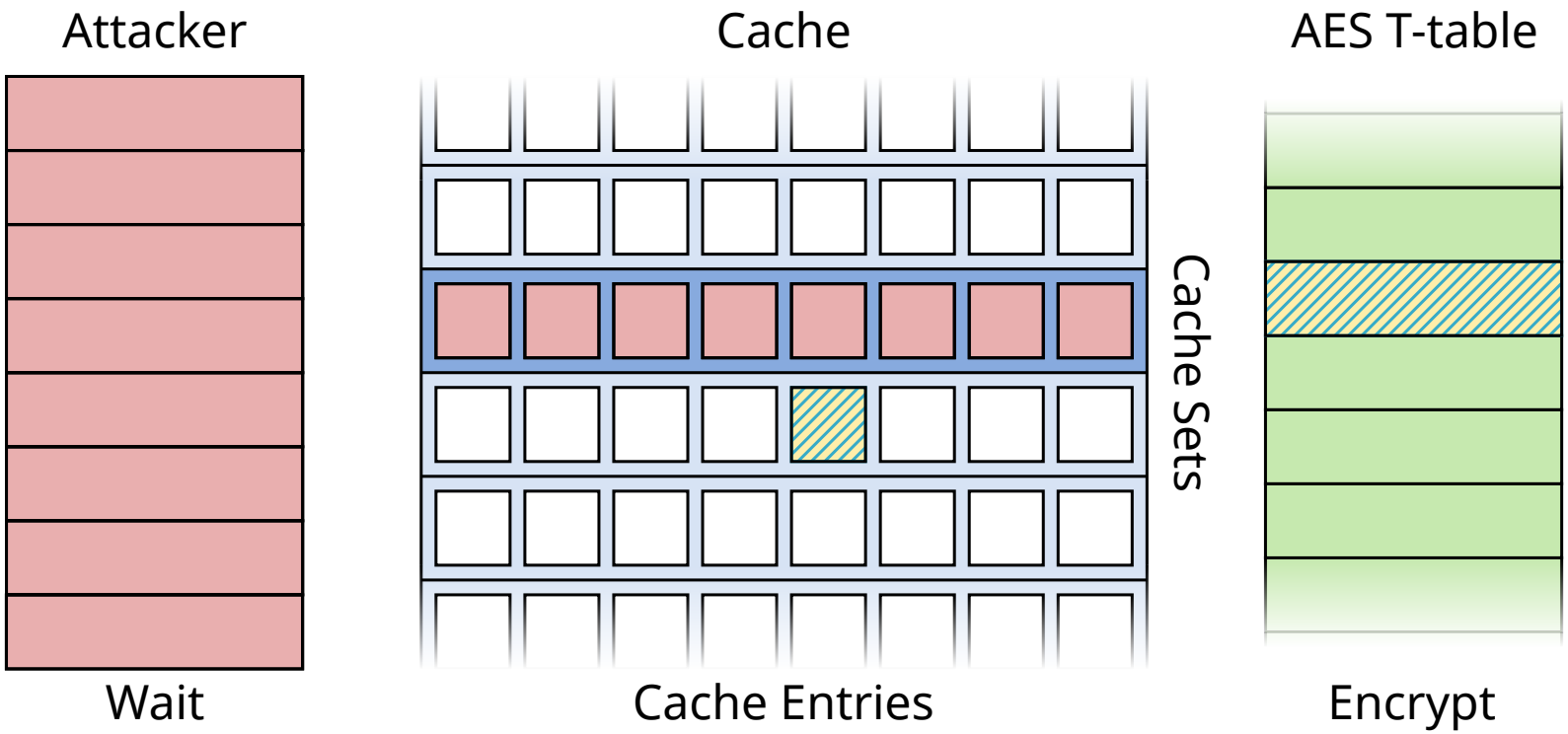
PRIME + PROBE



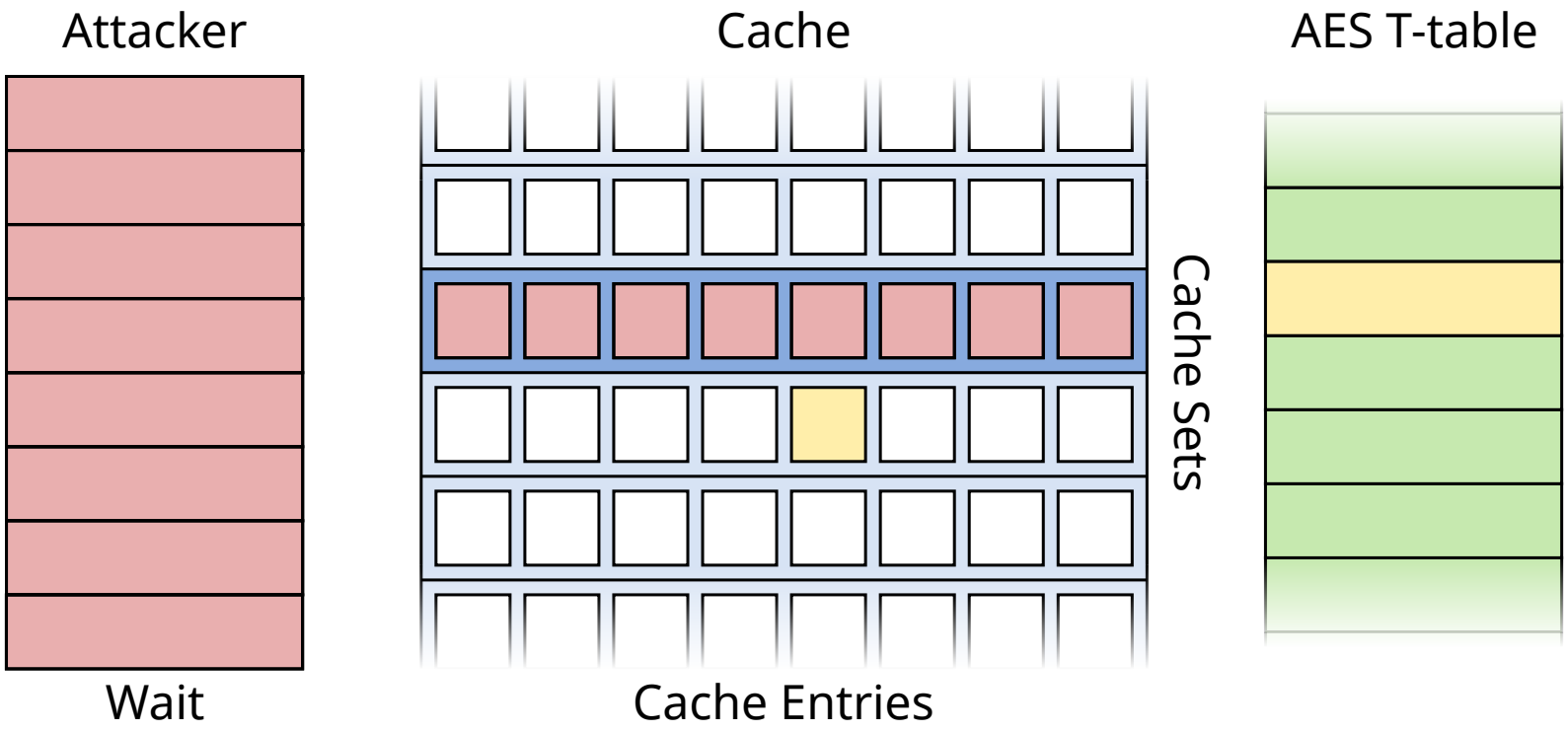
PRIME + PROBE



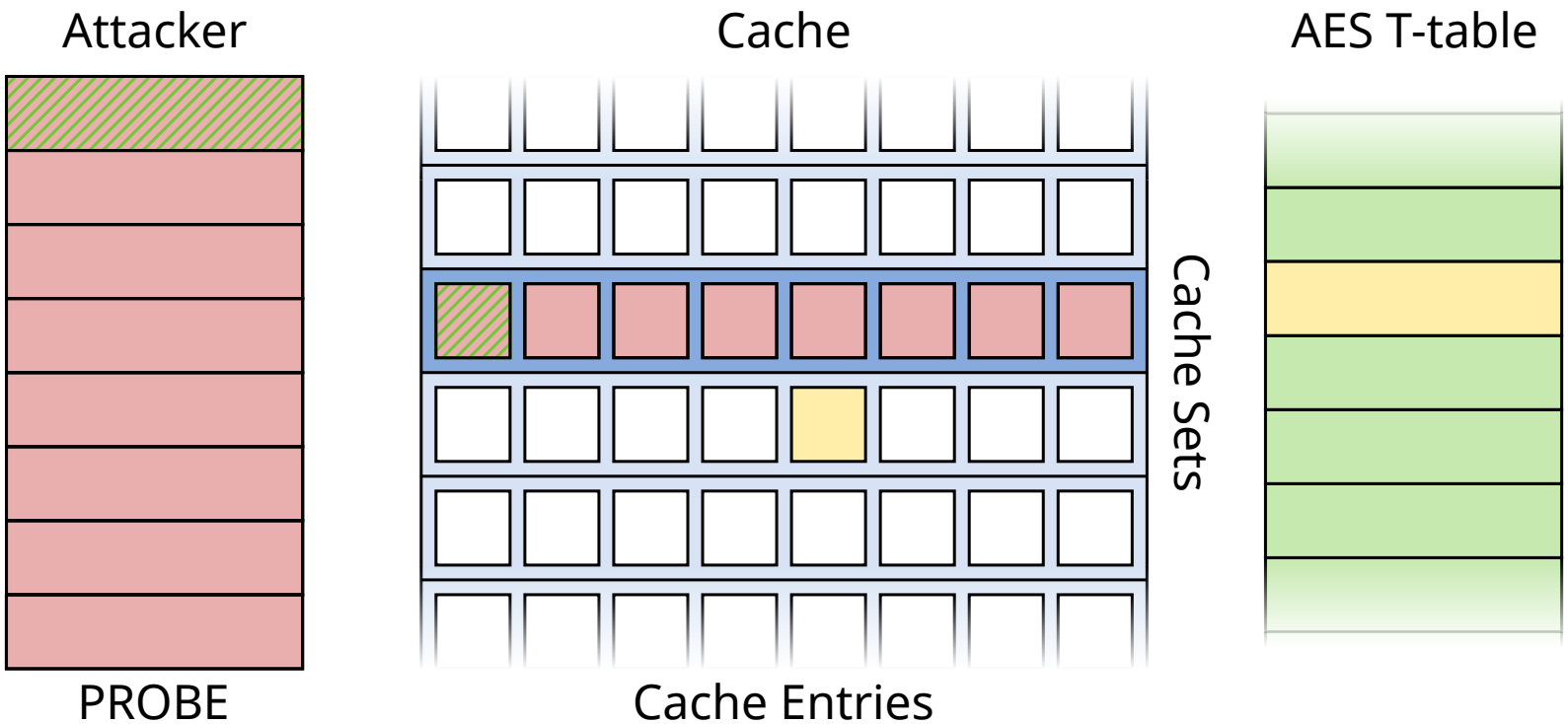
PRIME + PROBE


Attacker
Cache
AES T-table
Cache Sets
Wait
Cache Entries
Encrypt

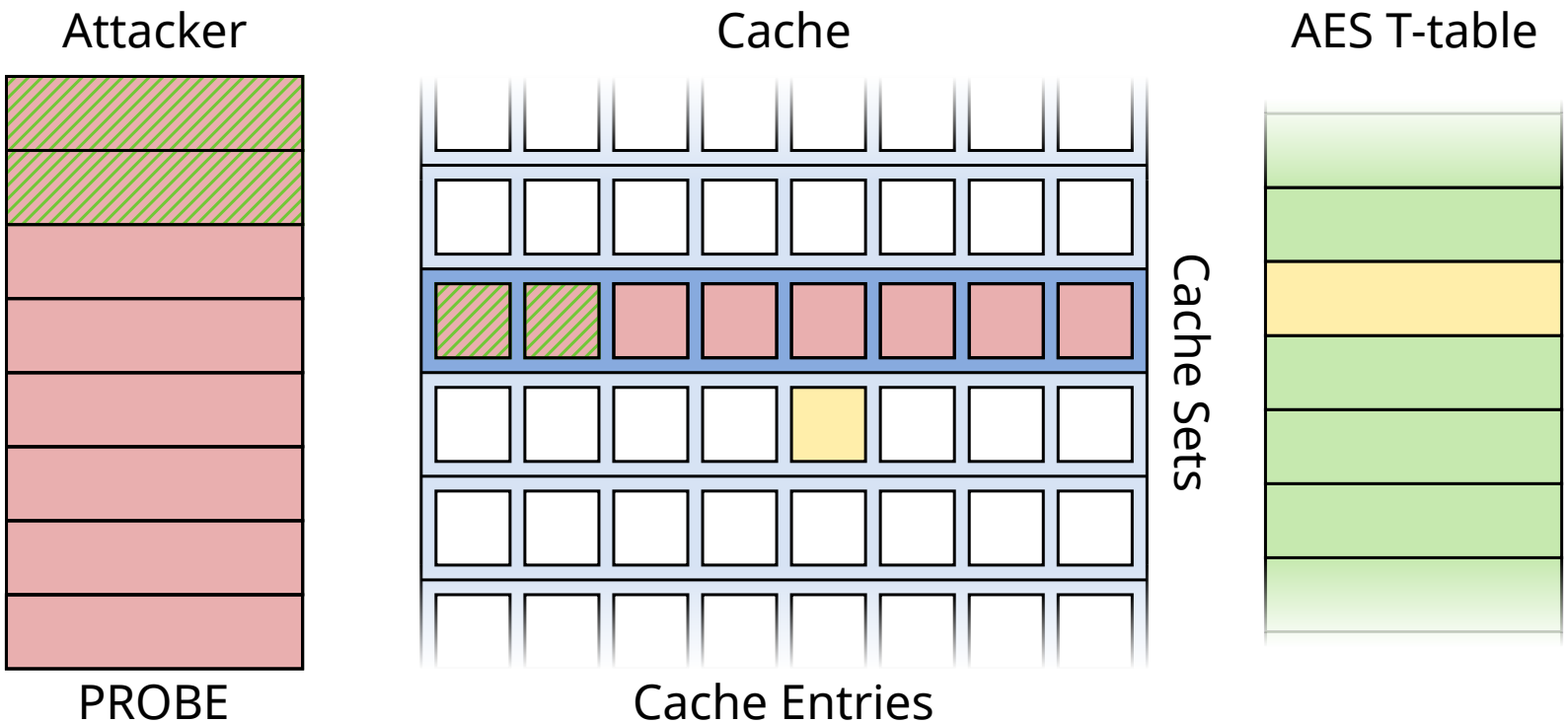
PRIME + PROBE



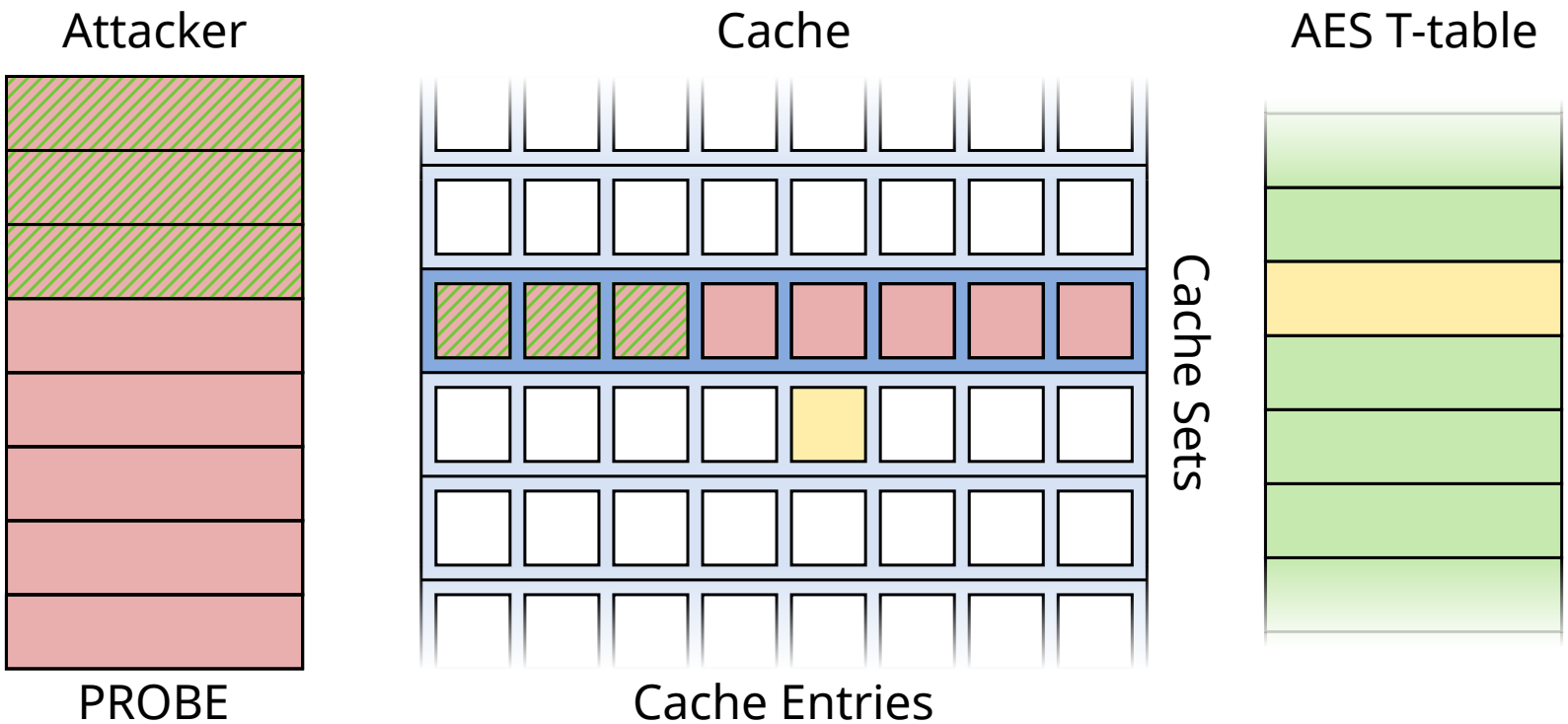
PRIME + PROBE



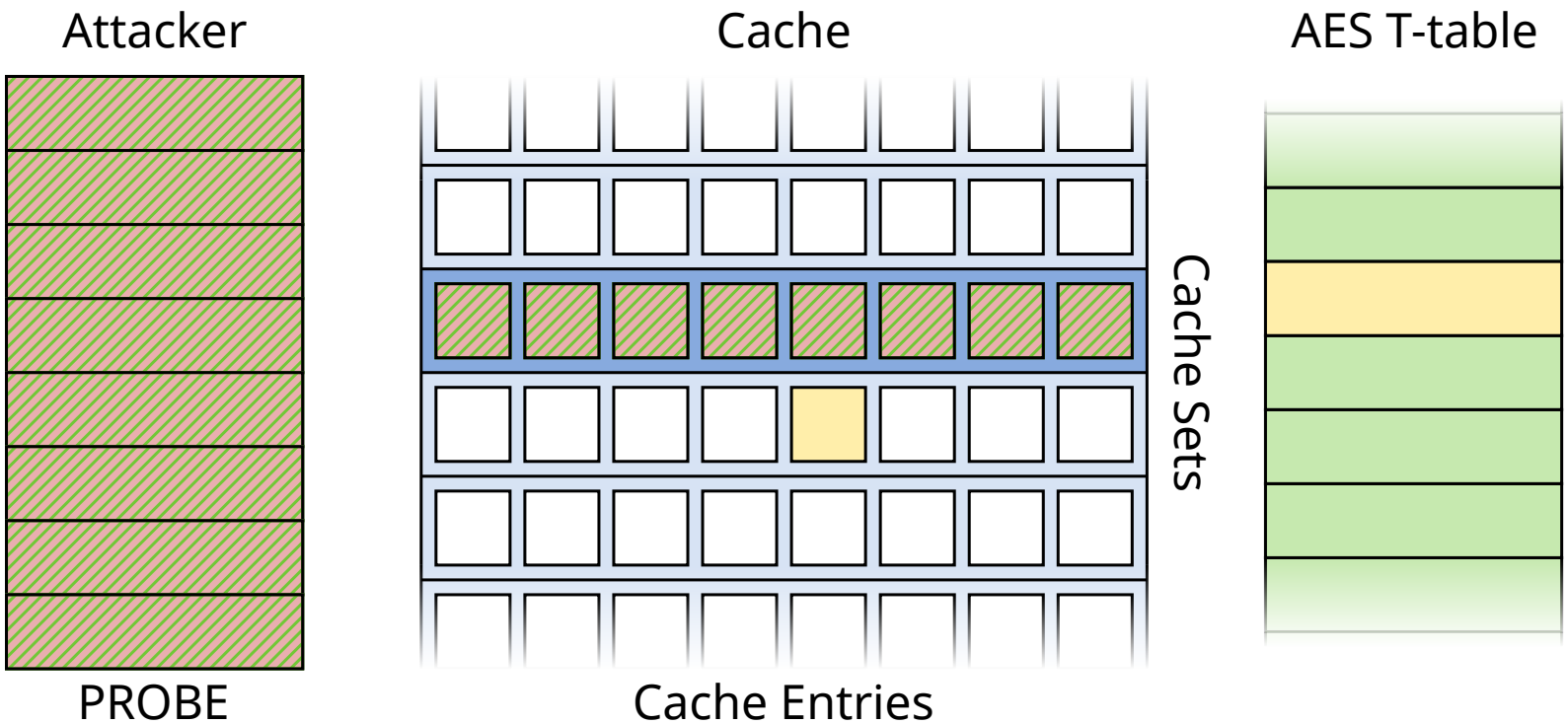
PRIME + PROBE



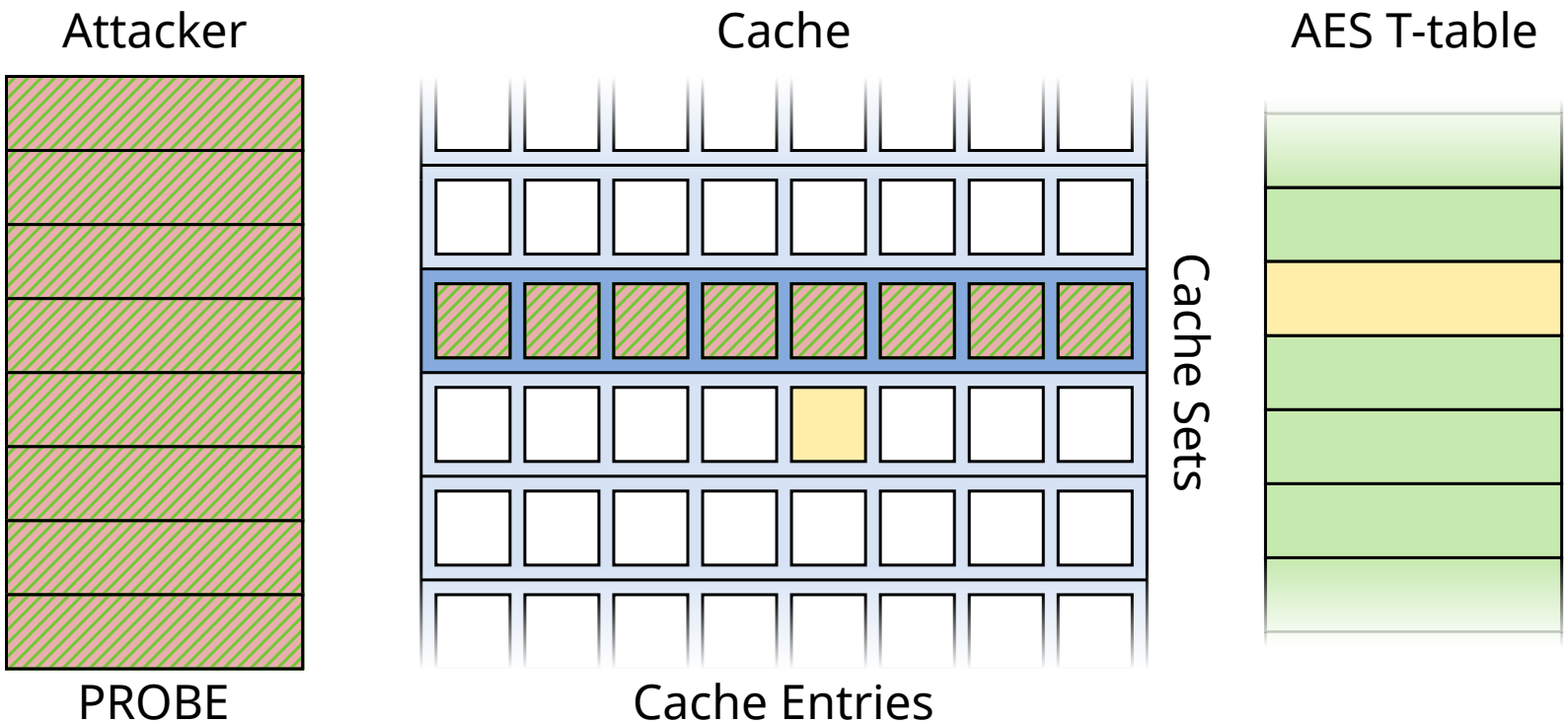
PRIME + PROBE



PRIME + PROBE

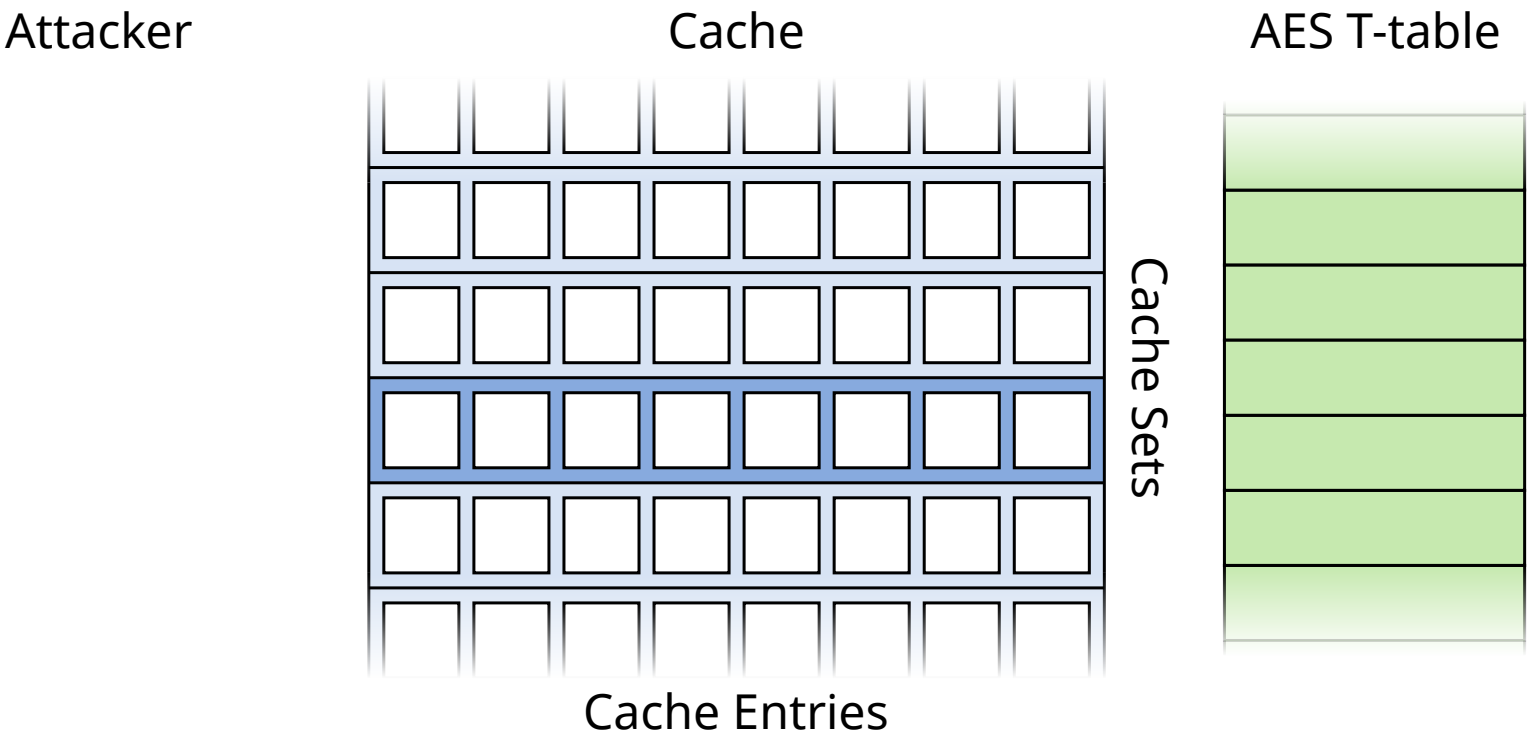


PRIME + PROBE

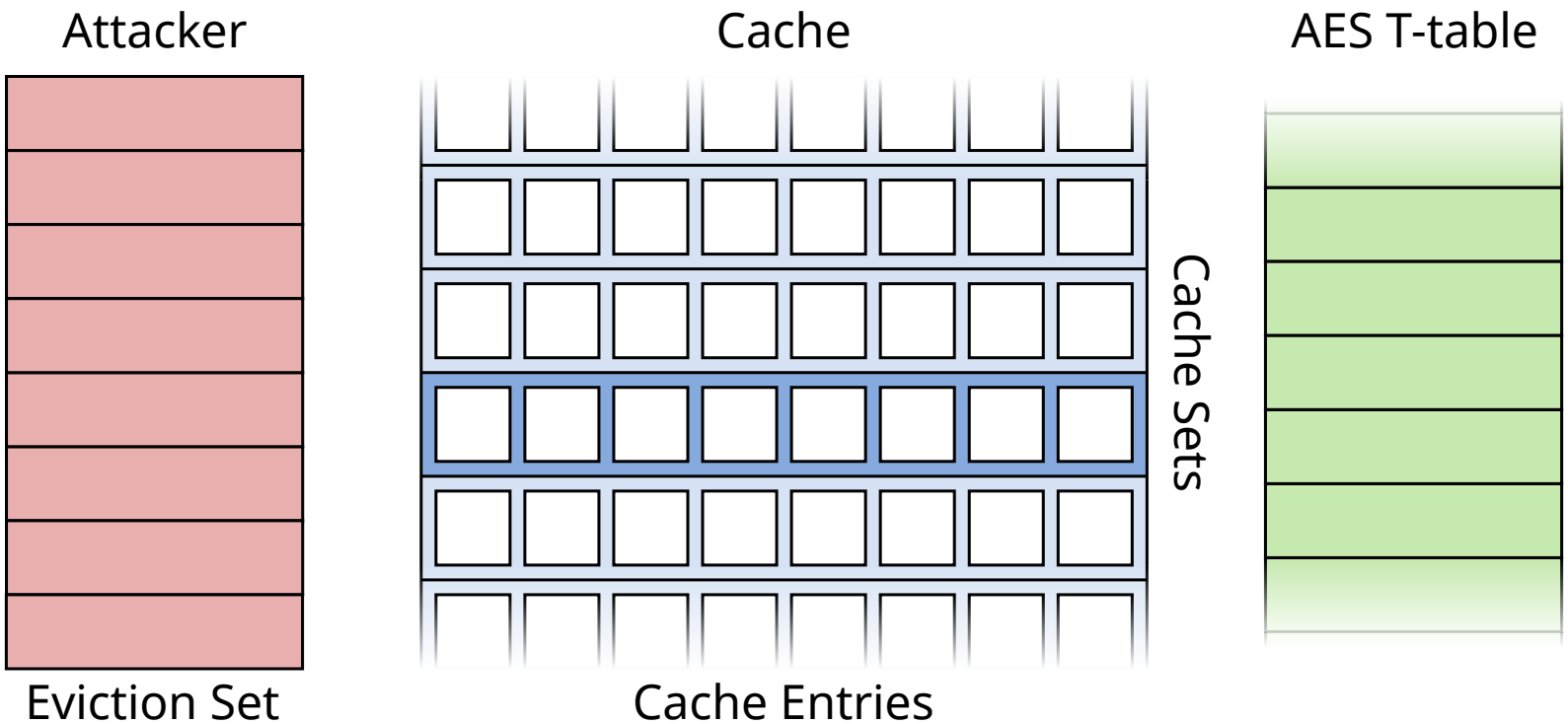


AES encrypt used another cache set

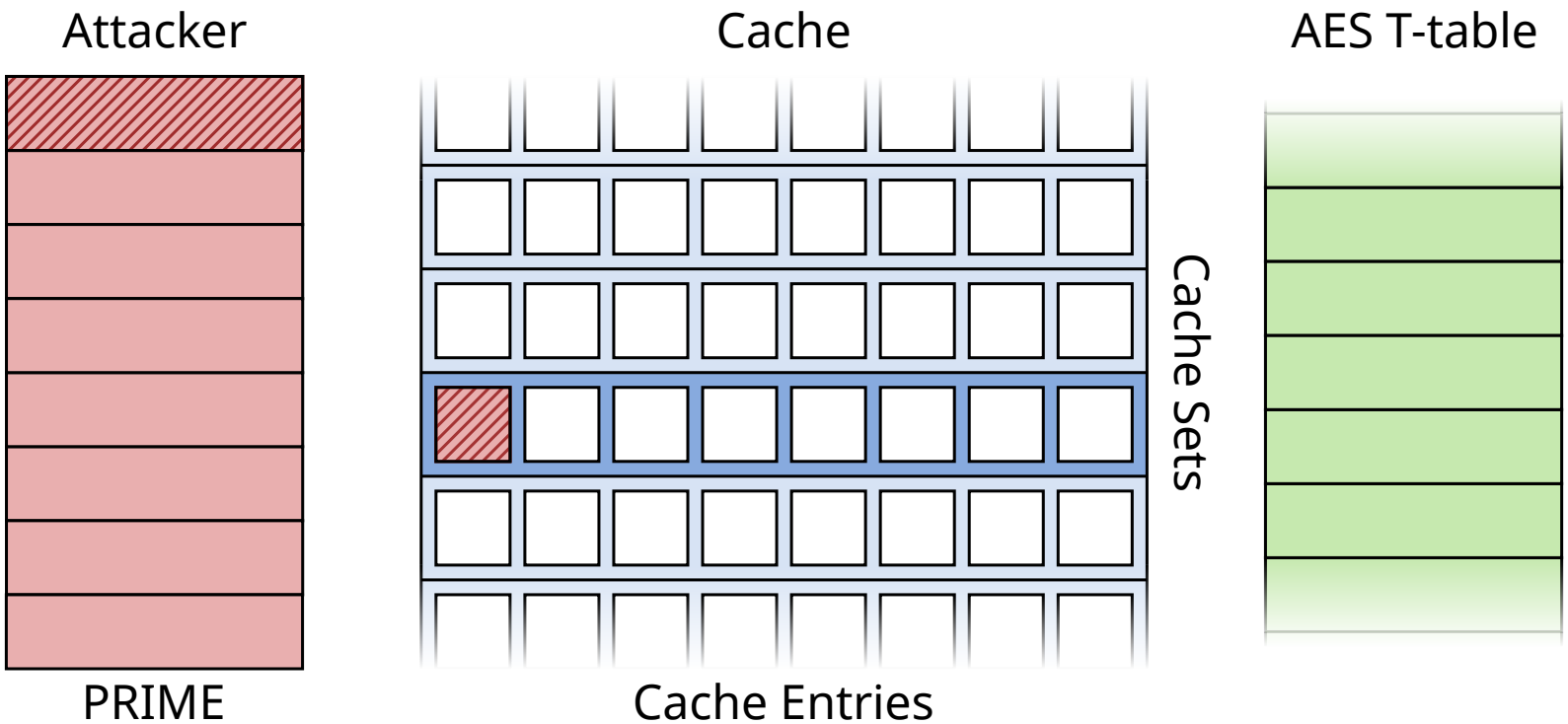
PRIME + PROBE



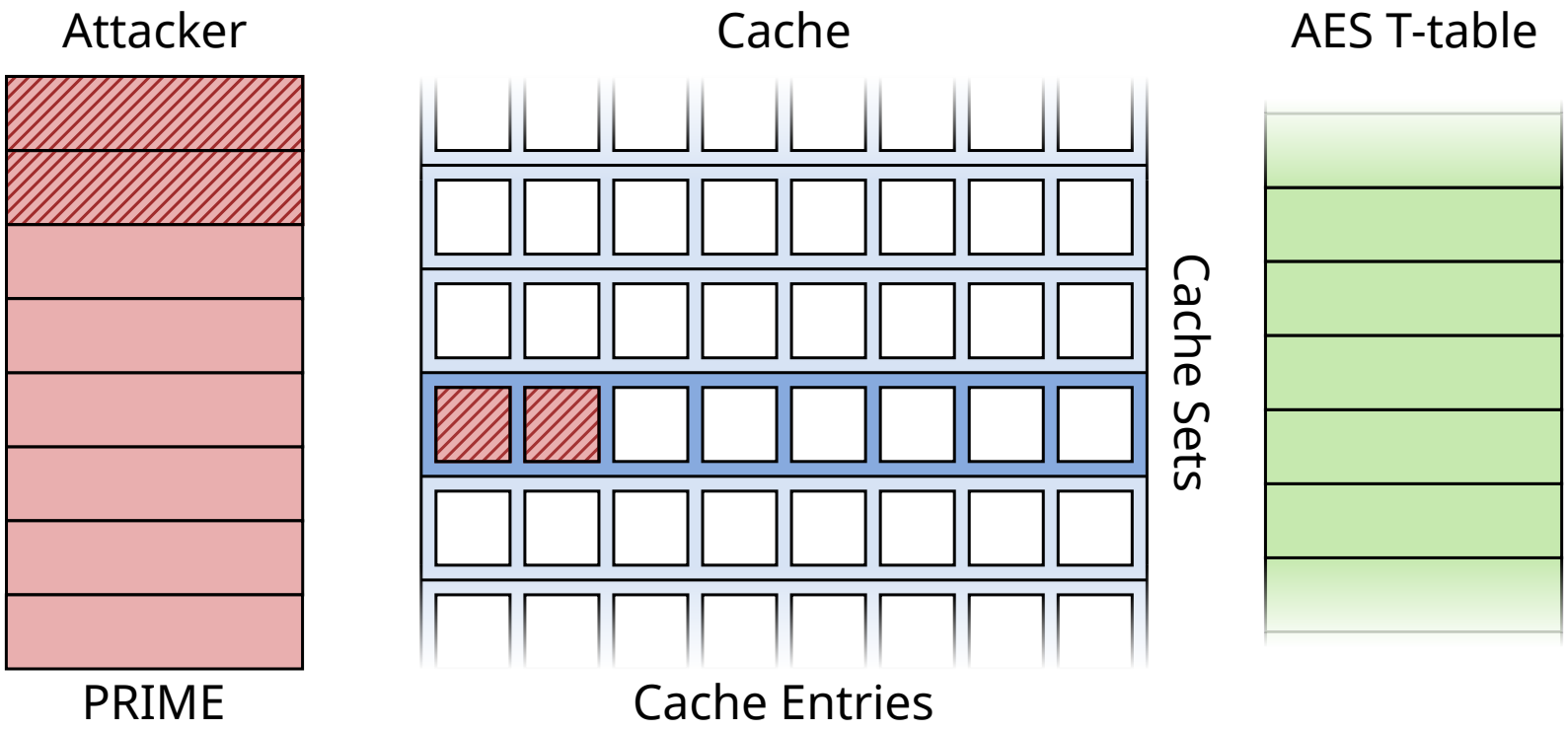
PRIME + PROBE


Attacker
Cache
AES T-table
Cache Sets
Eviction Set
Cache Entries

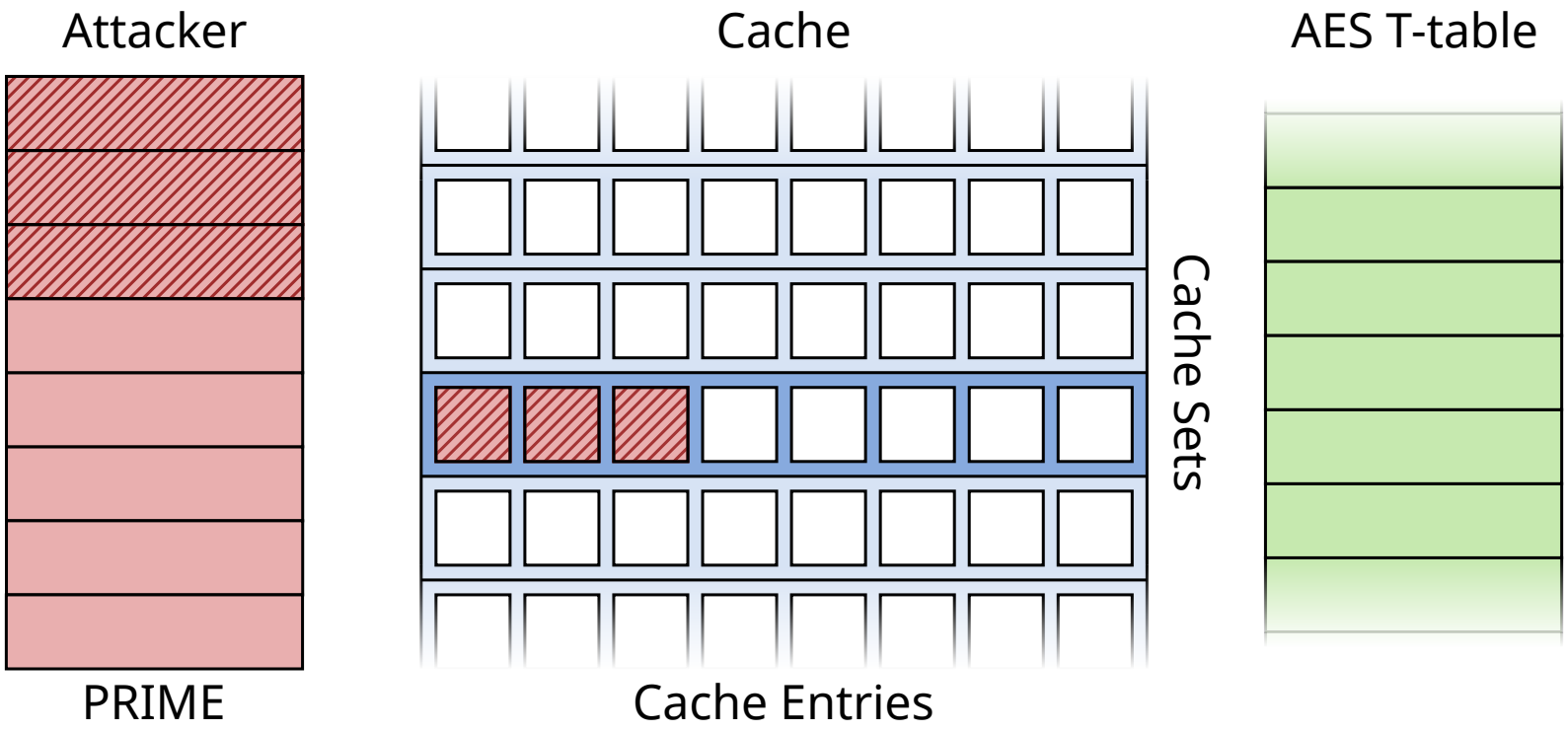
PRIME + PROBE



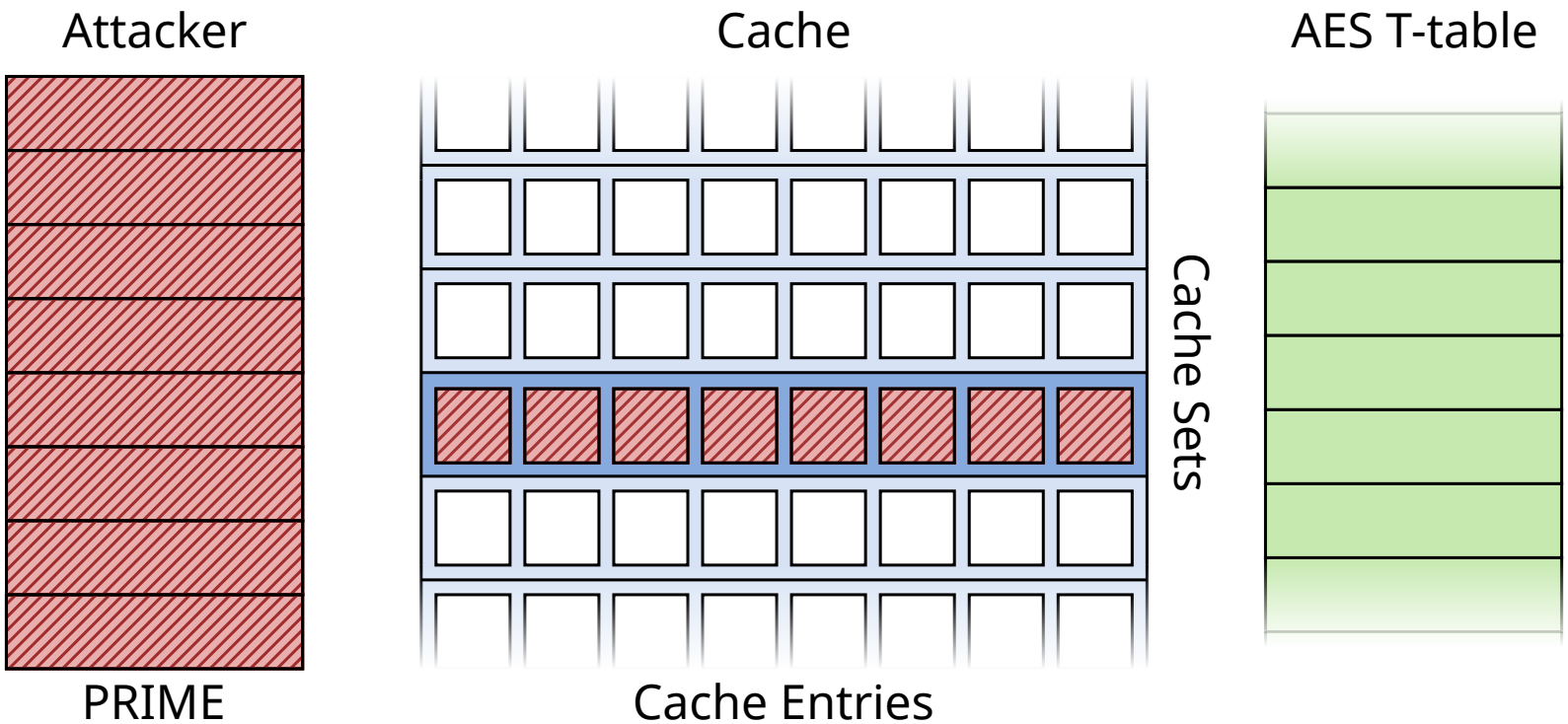
PRIME + PROBE



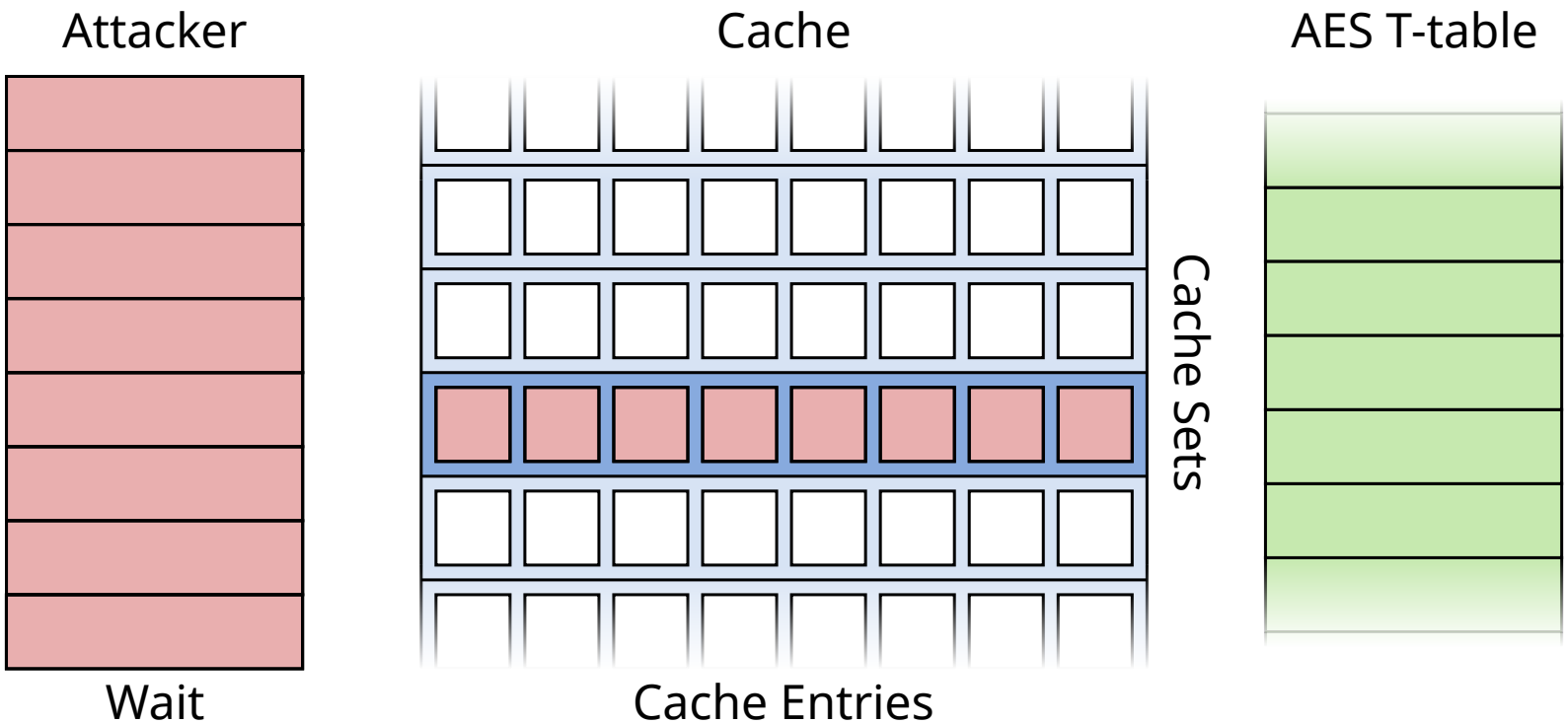
PRIME + PROBE



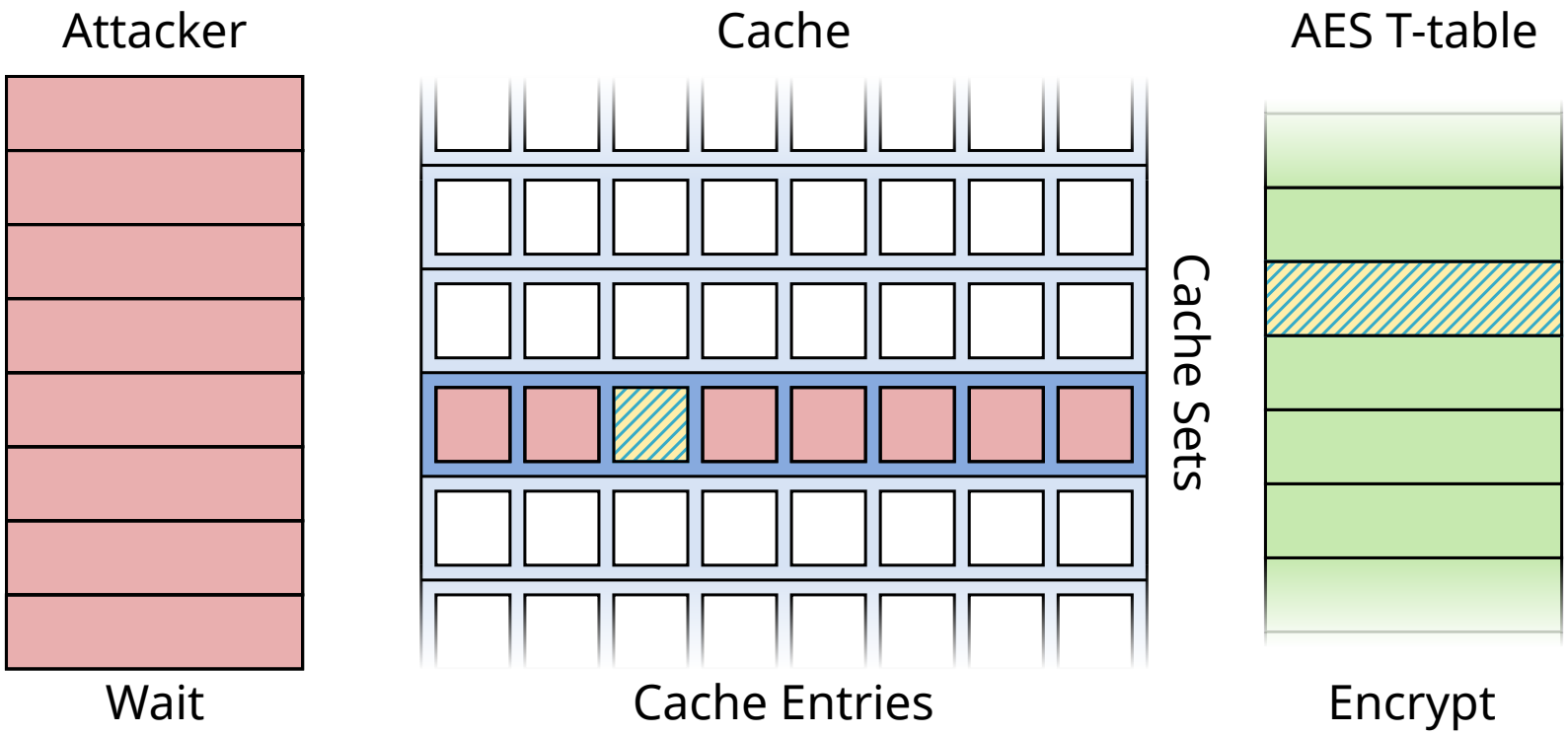
PRIME + PROBE



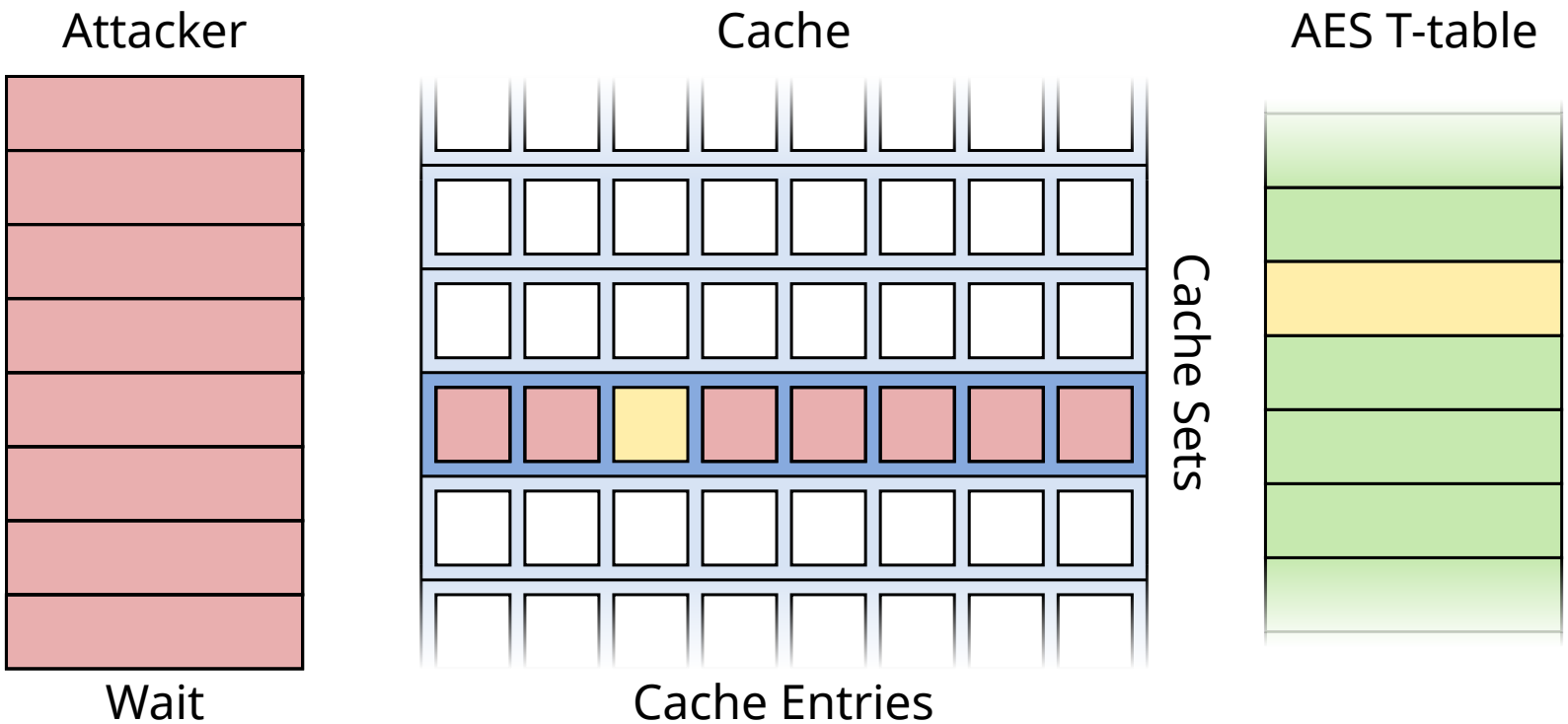
PRIME + PROBE



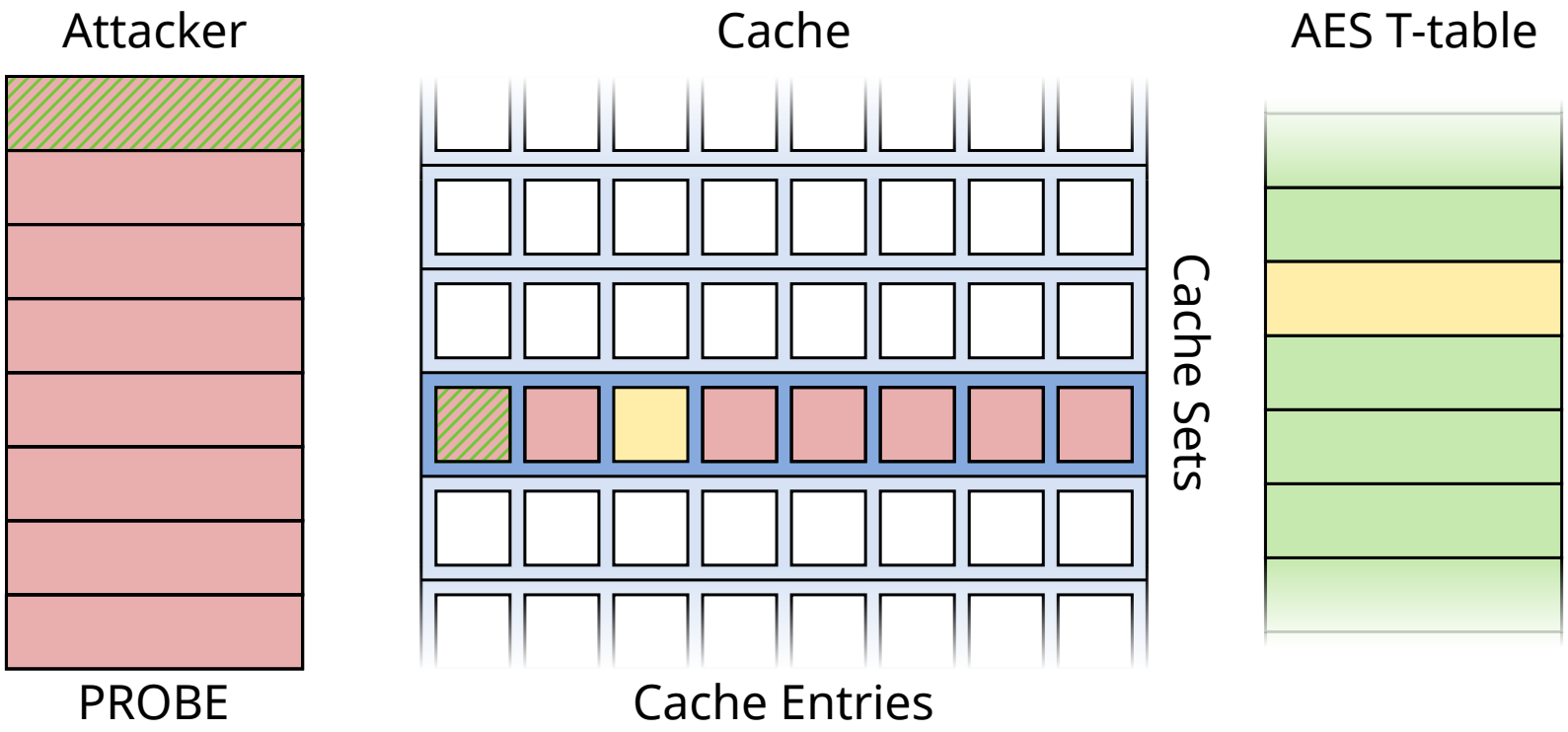
PRIME + PROBE


Attacker
Cache
AES T-table
Cache Sets
Wait
Cache Entries
Encrypt

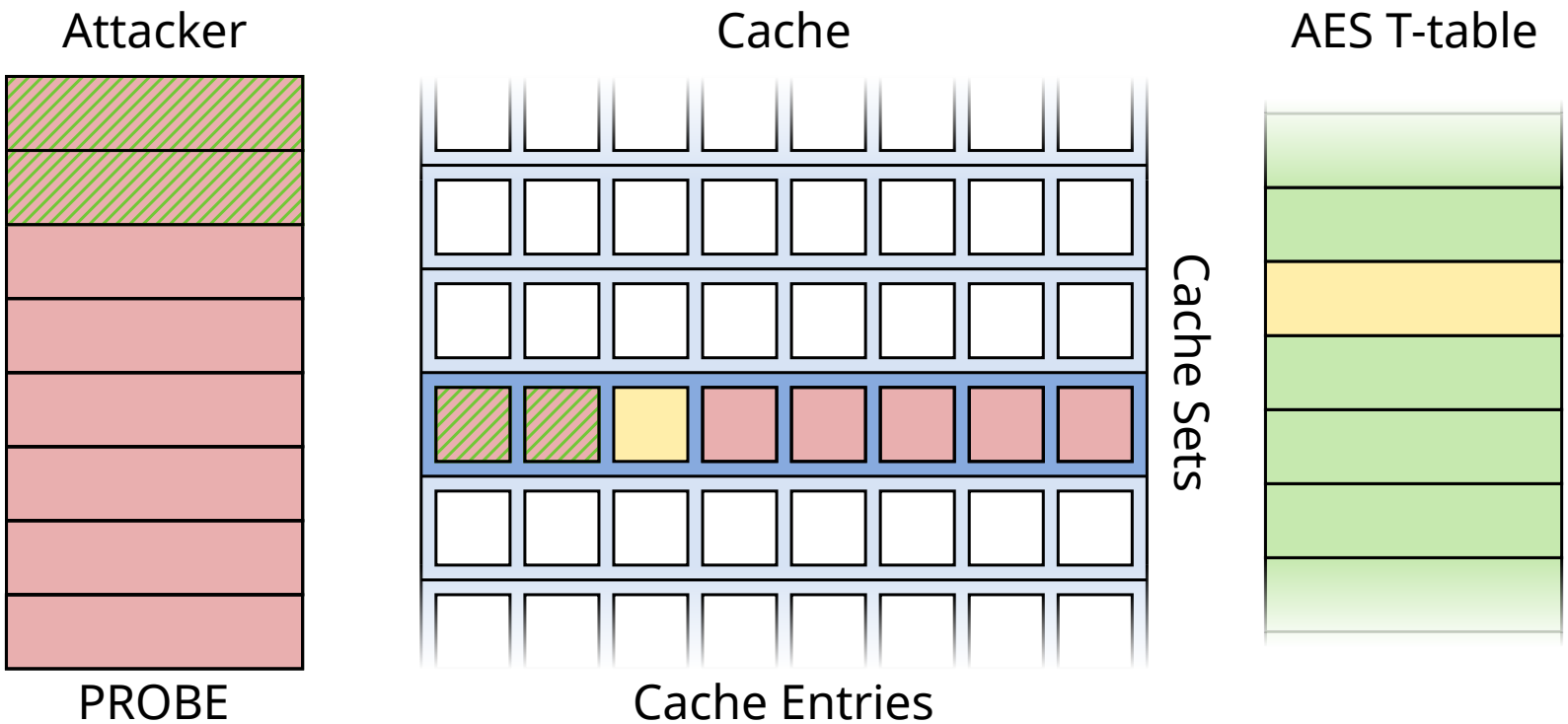
PRIME + PROBE



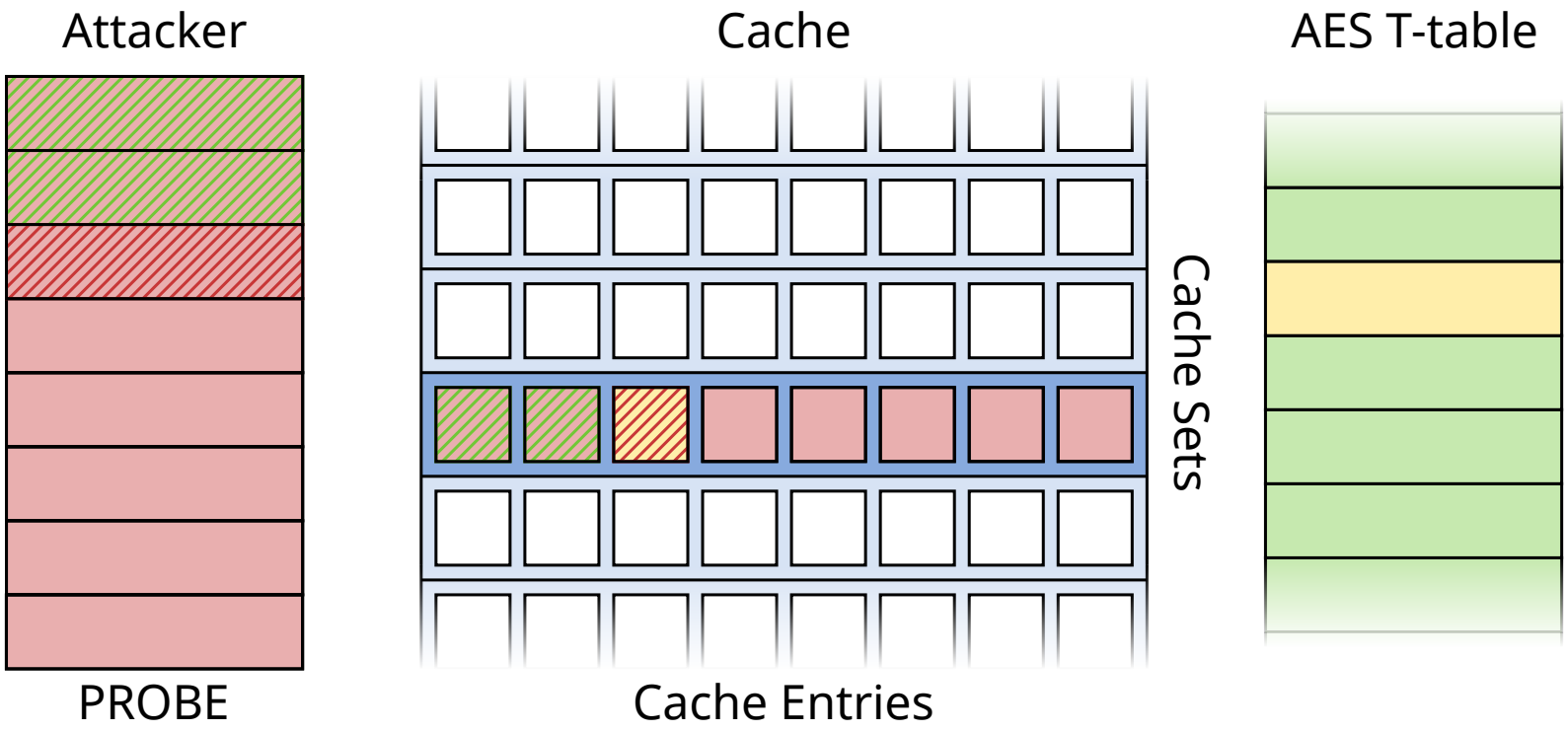
PRIME + PROBE



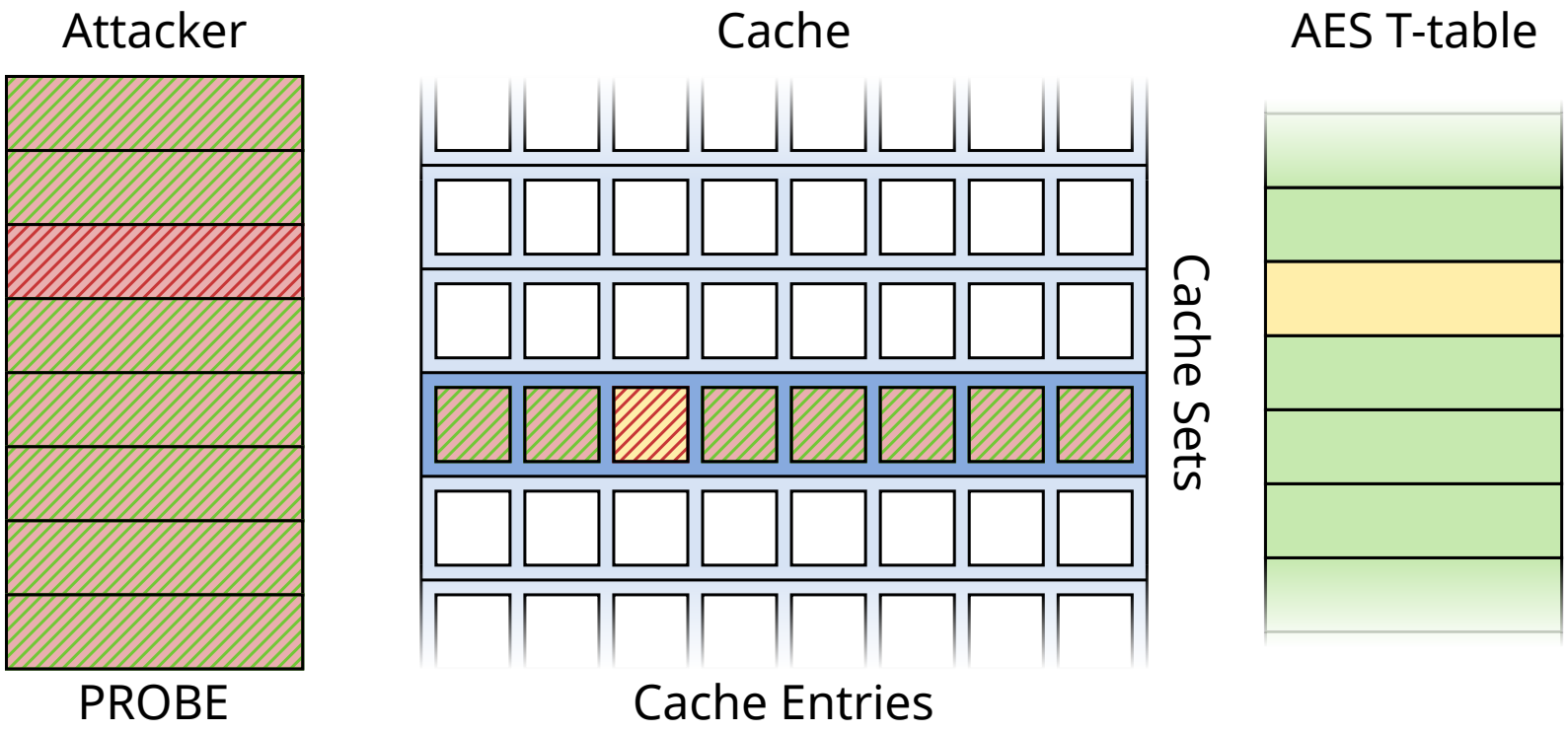
PRIME + PROBE



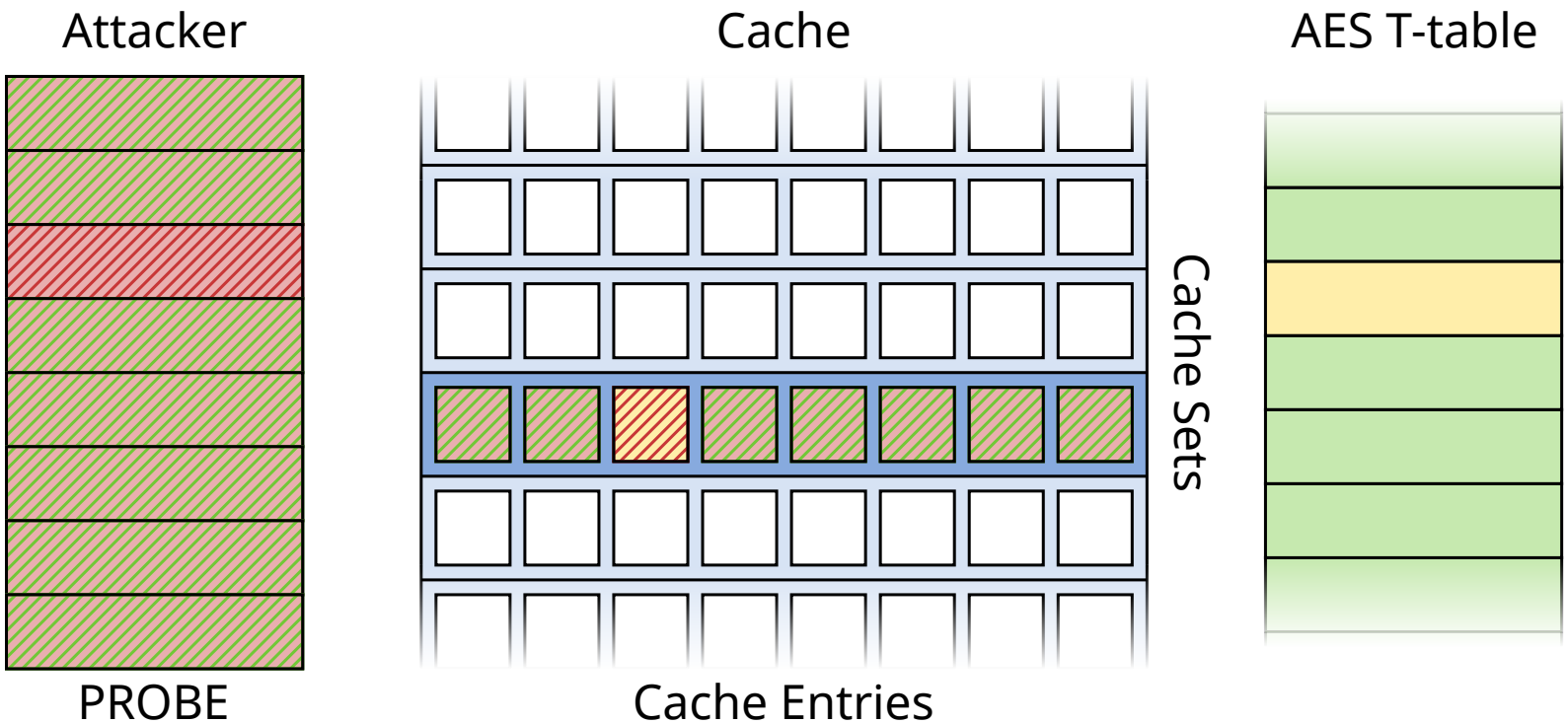
PRIME + PROBE



PRIME + PROBE



PRIME + PROBE

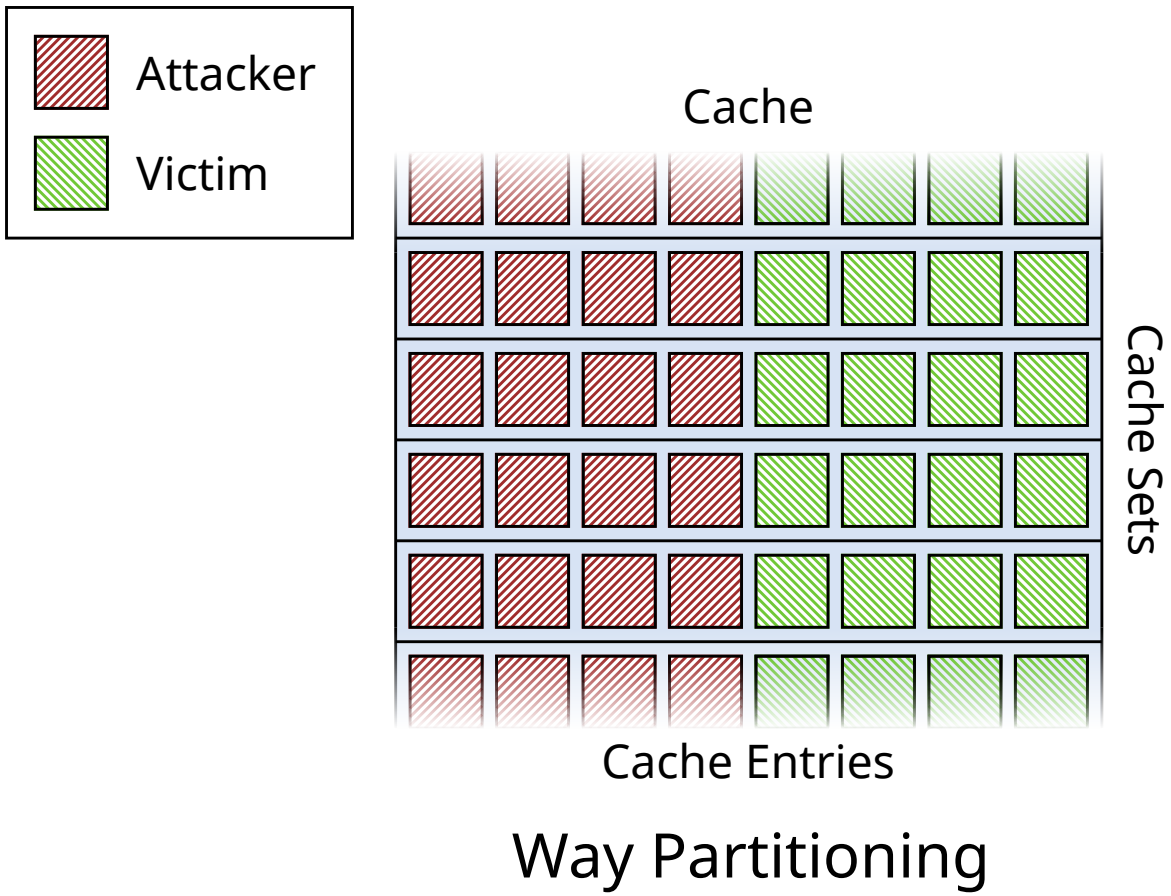


AES encrypt used the same cache set

Defenses

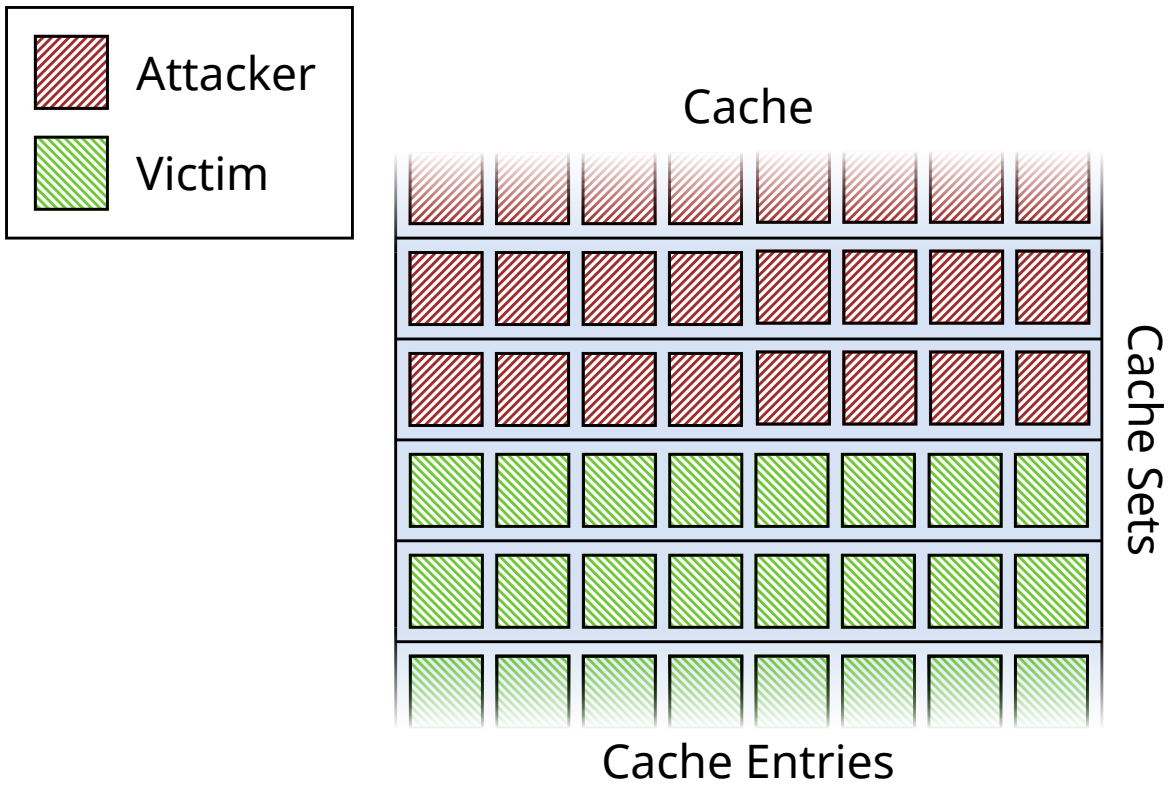
Can we defend against cache attacks?

Defenses



Way Partitioning

Defenses

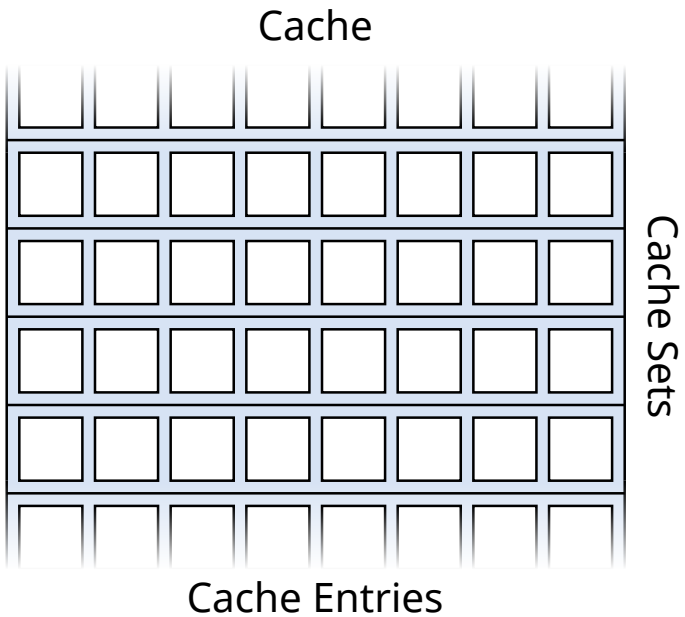


Set Partitioning

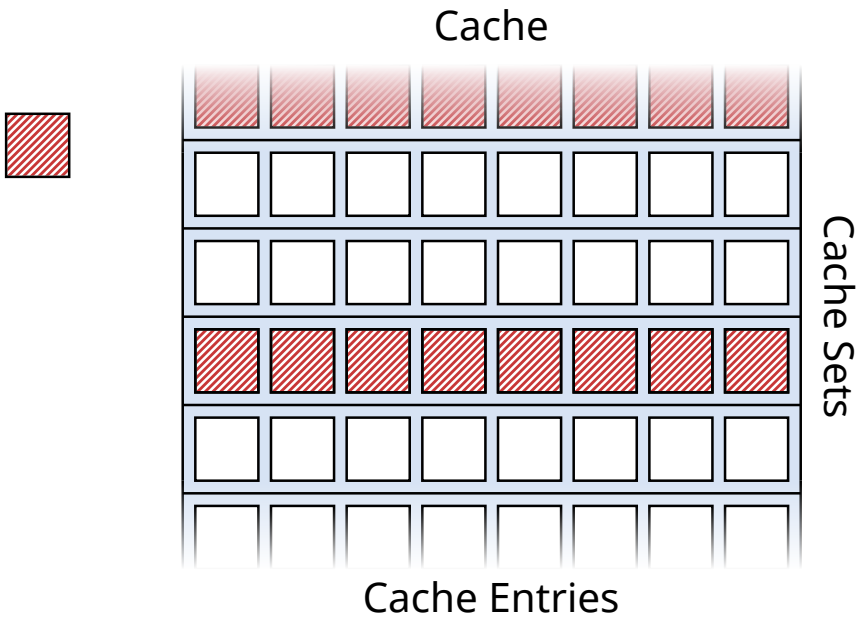
Page coloring

How does page coloring work?

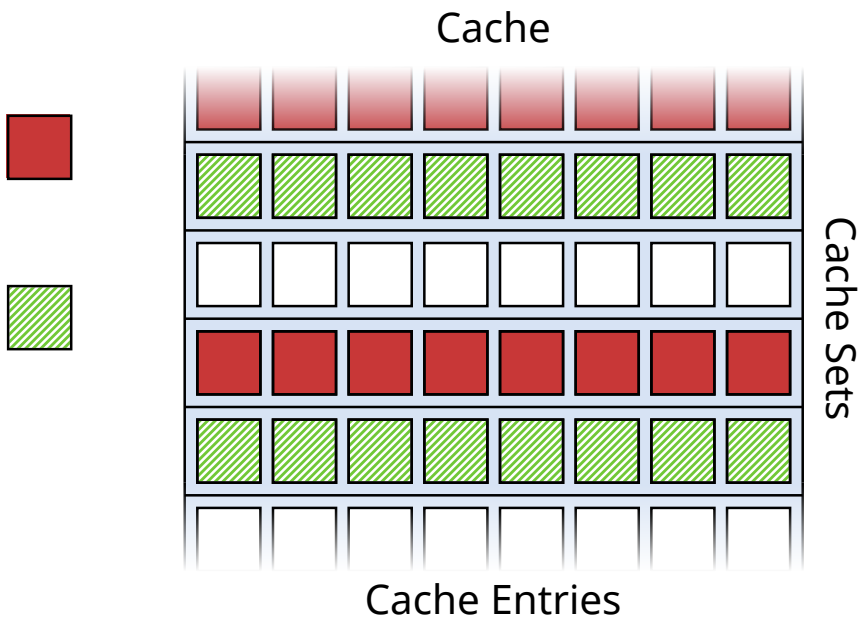
Page coloring



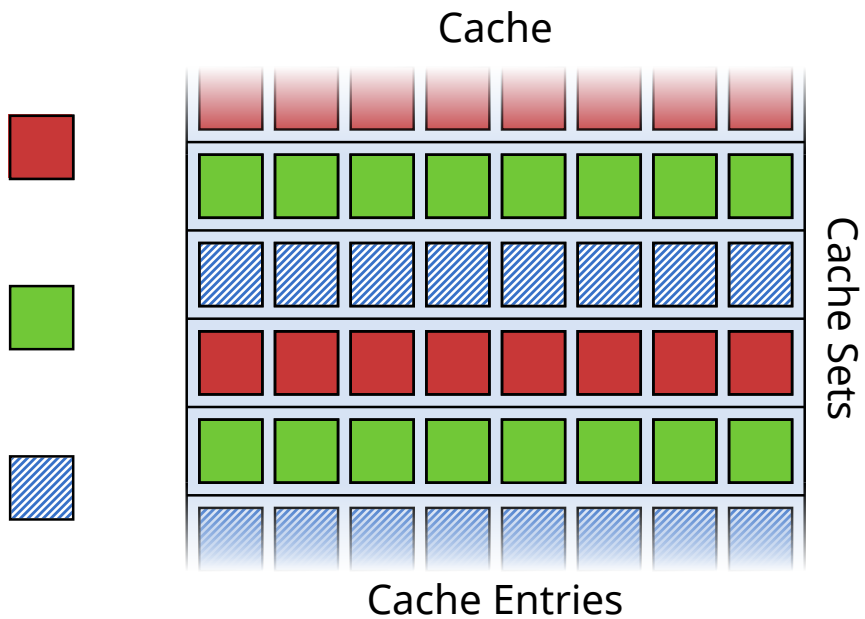
Page coloring



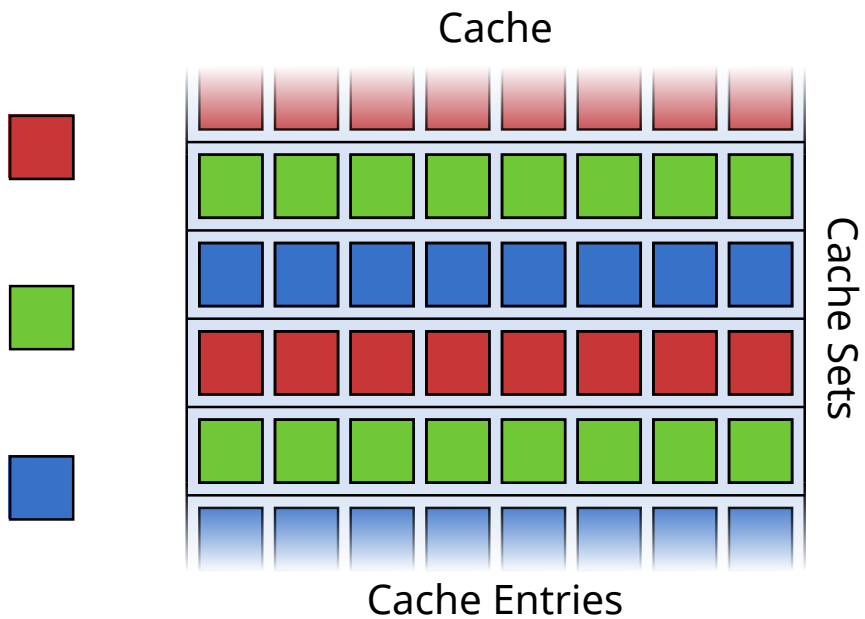
Page coloring



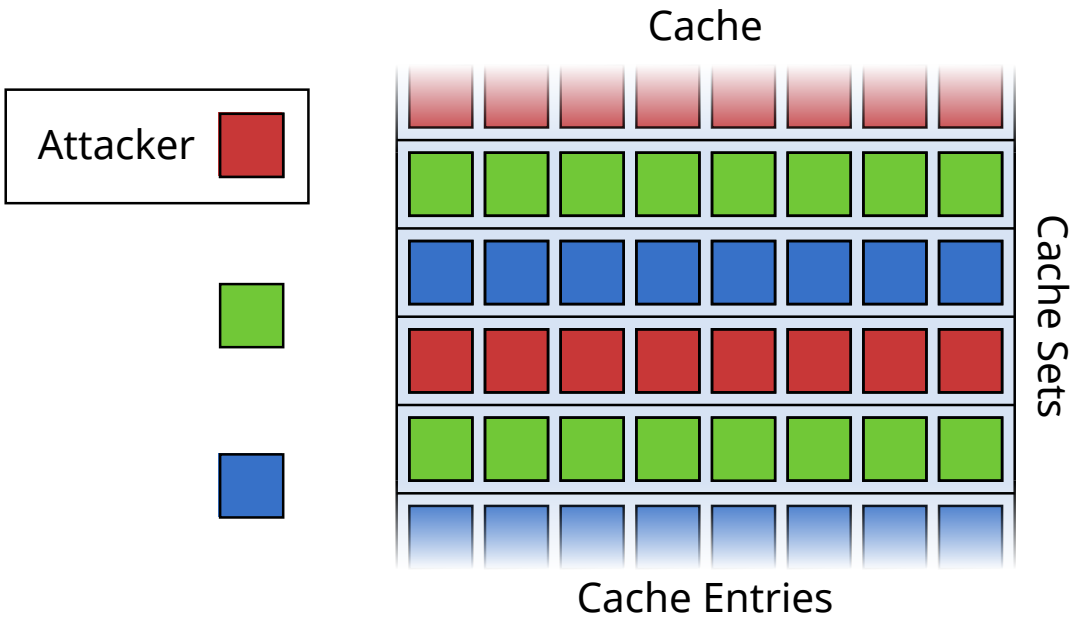
Page coloring



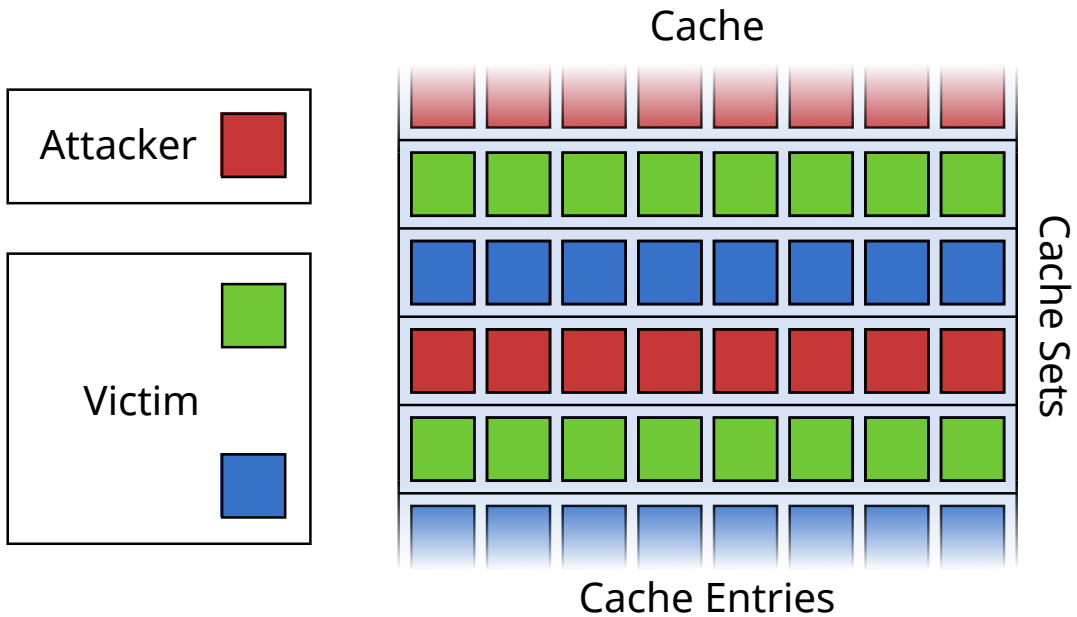
Page coloring



Page coloring



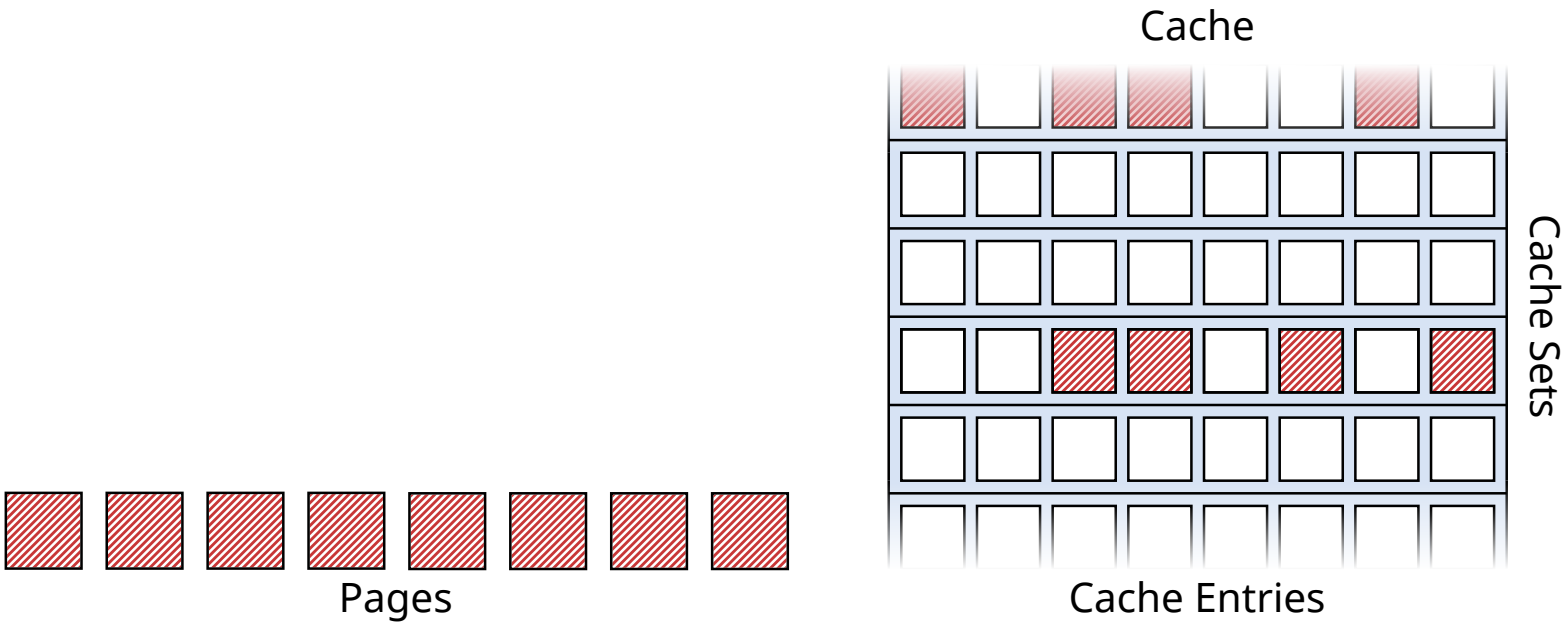
Page coloring



Page coloring

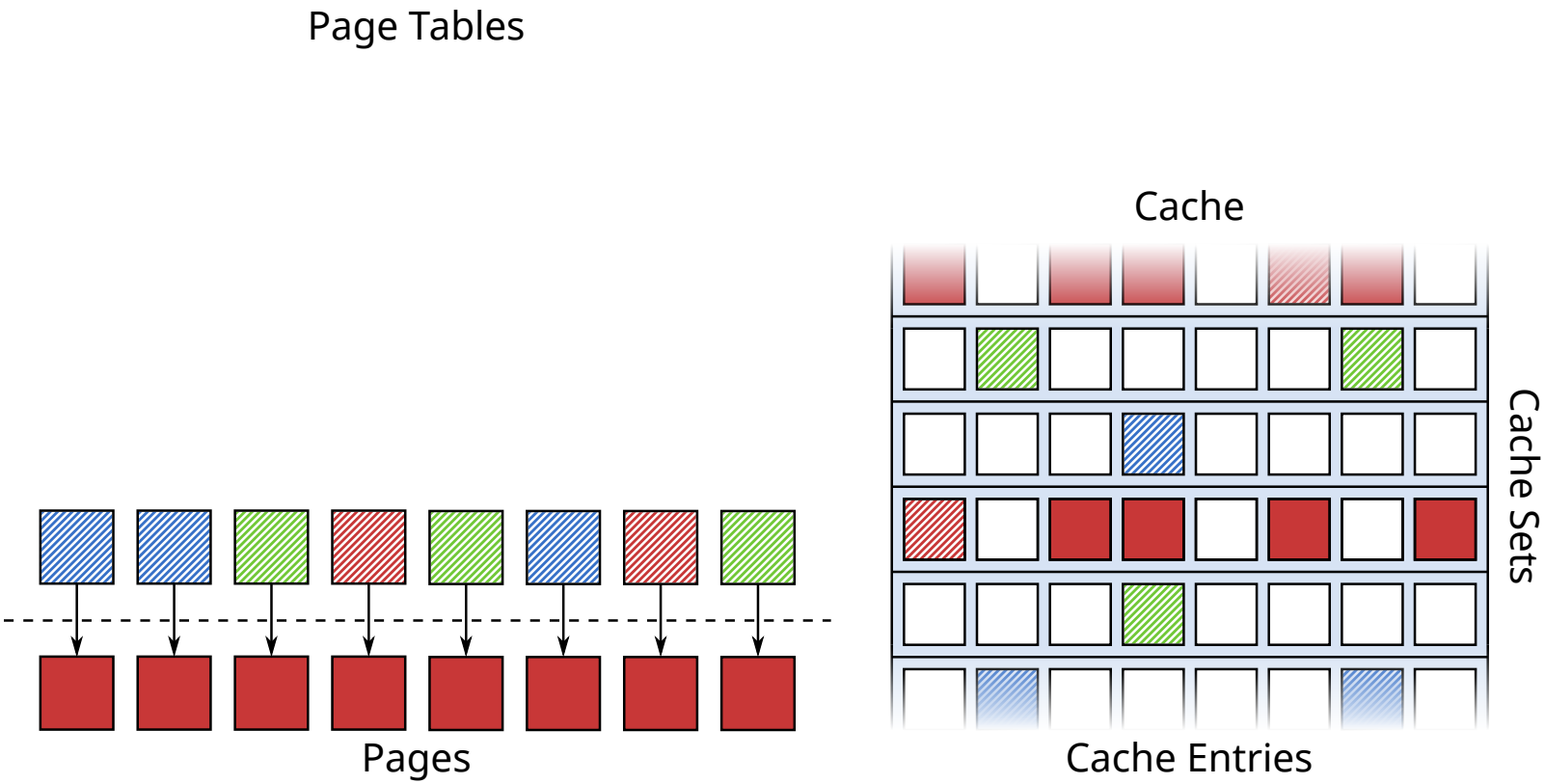
The victim and the attacker are nicely isolated.

Page coloring



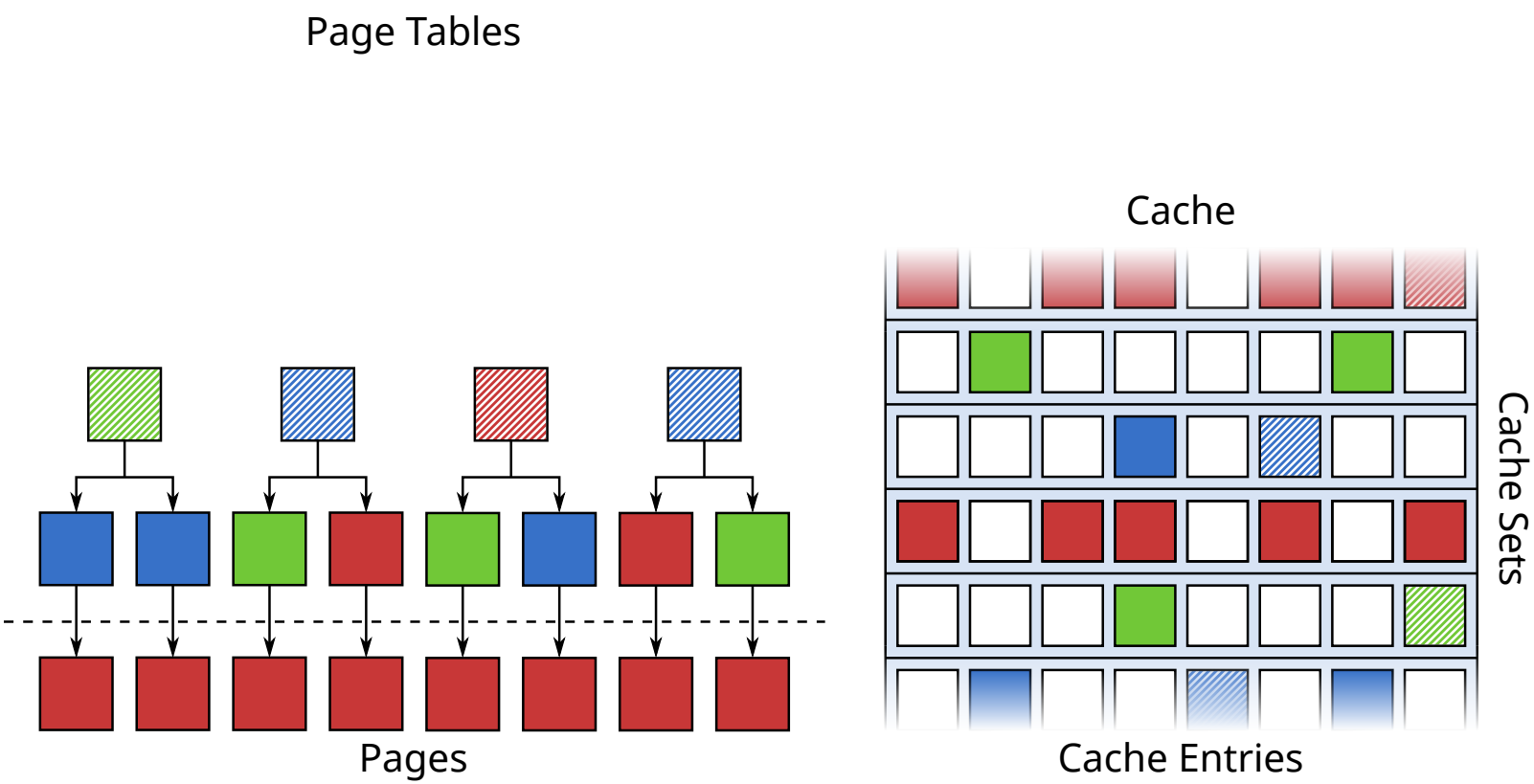
The attacker can only allocate red pages

Page coloring



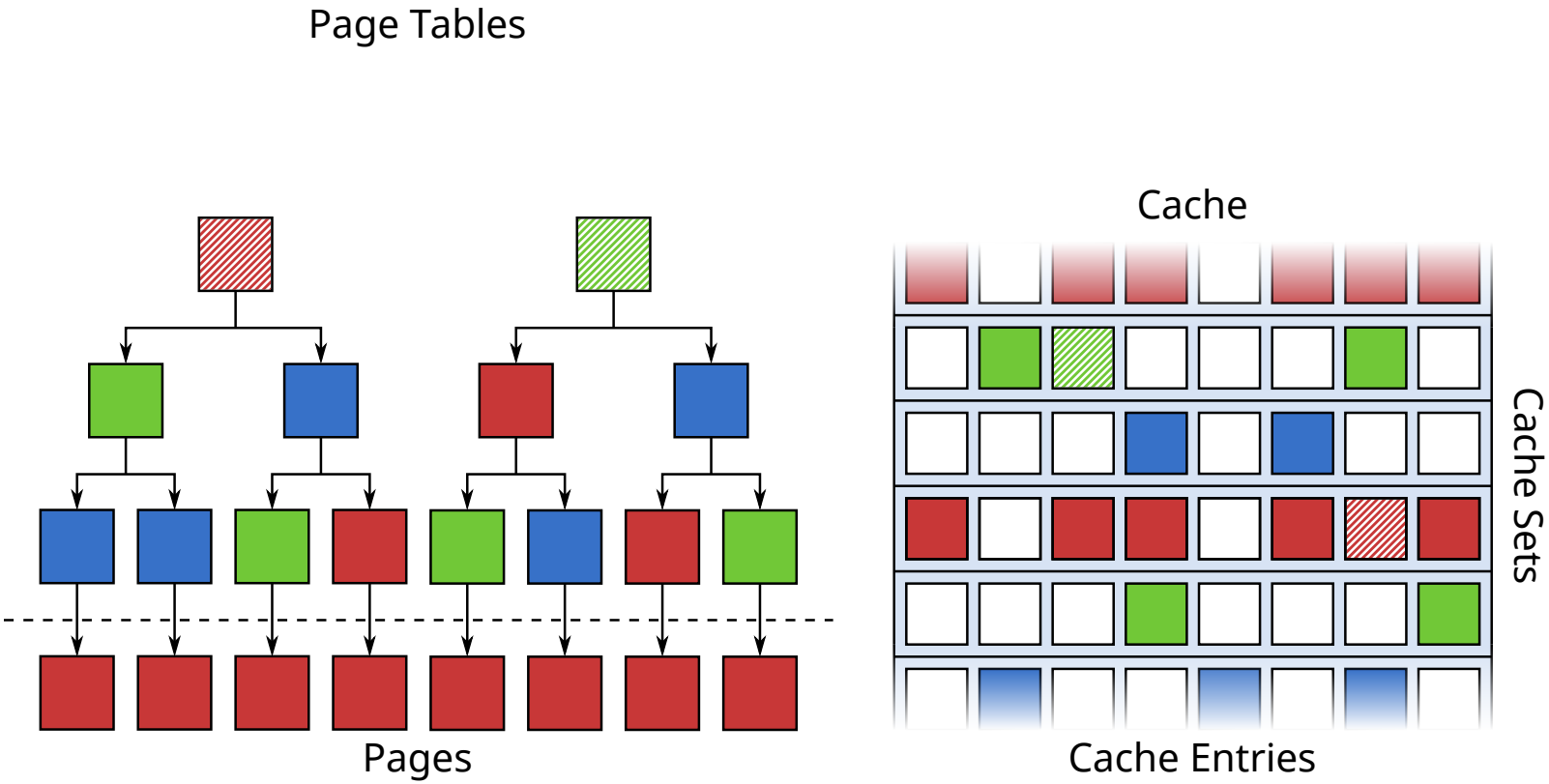
However, the page tables aren't colored

Page coloring



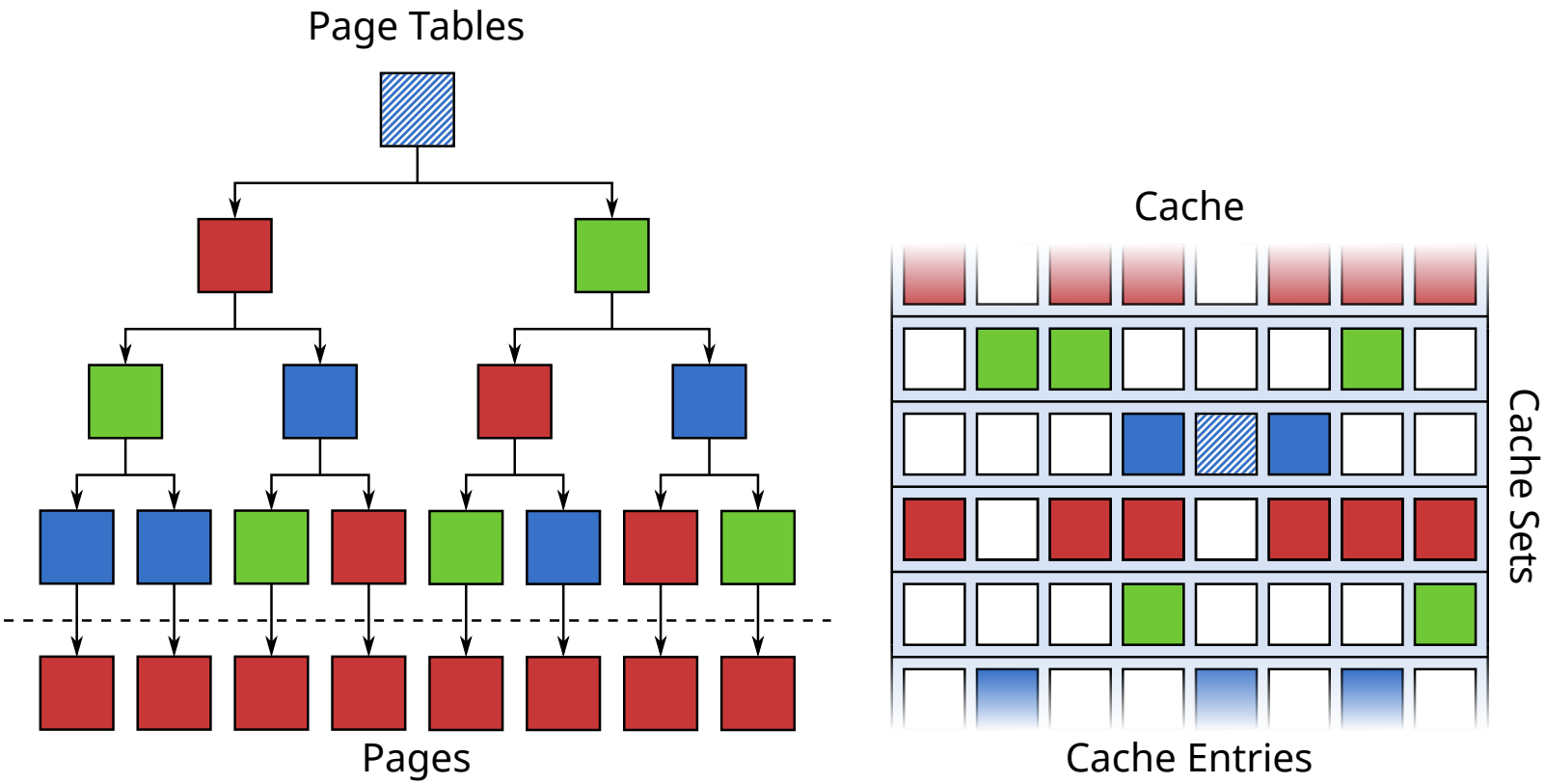
However, the page tables aren't colored

Page coloring



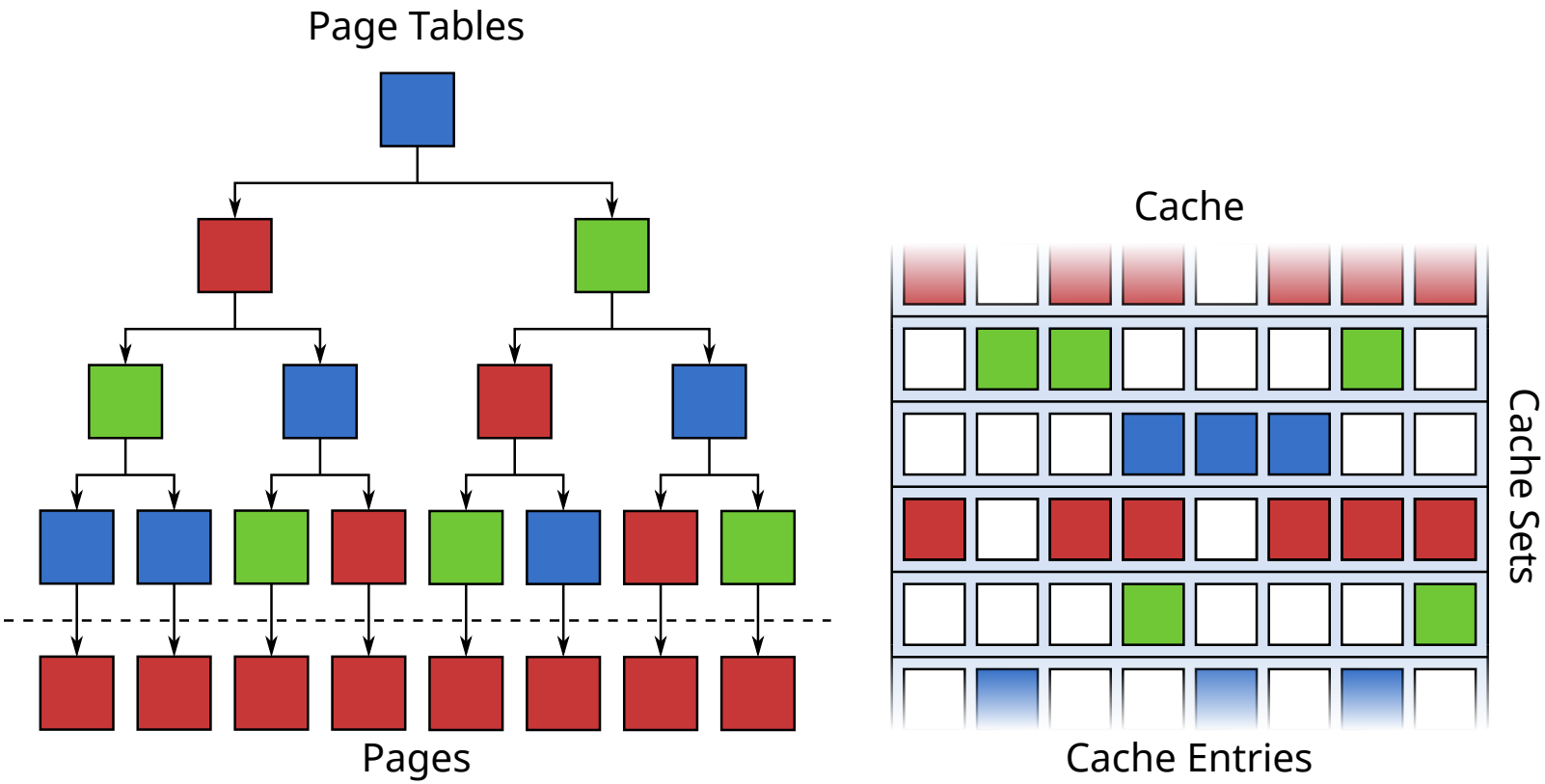
However, the page tables aren't colored

Page coloring



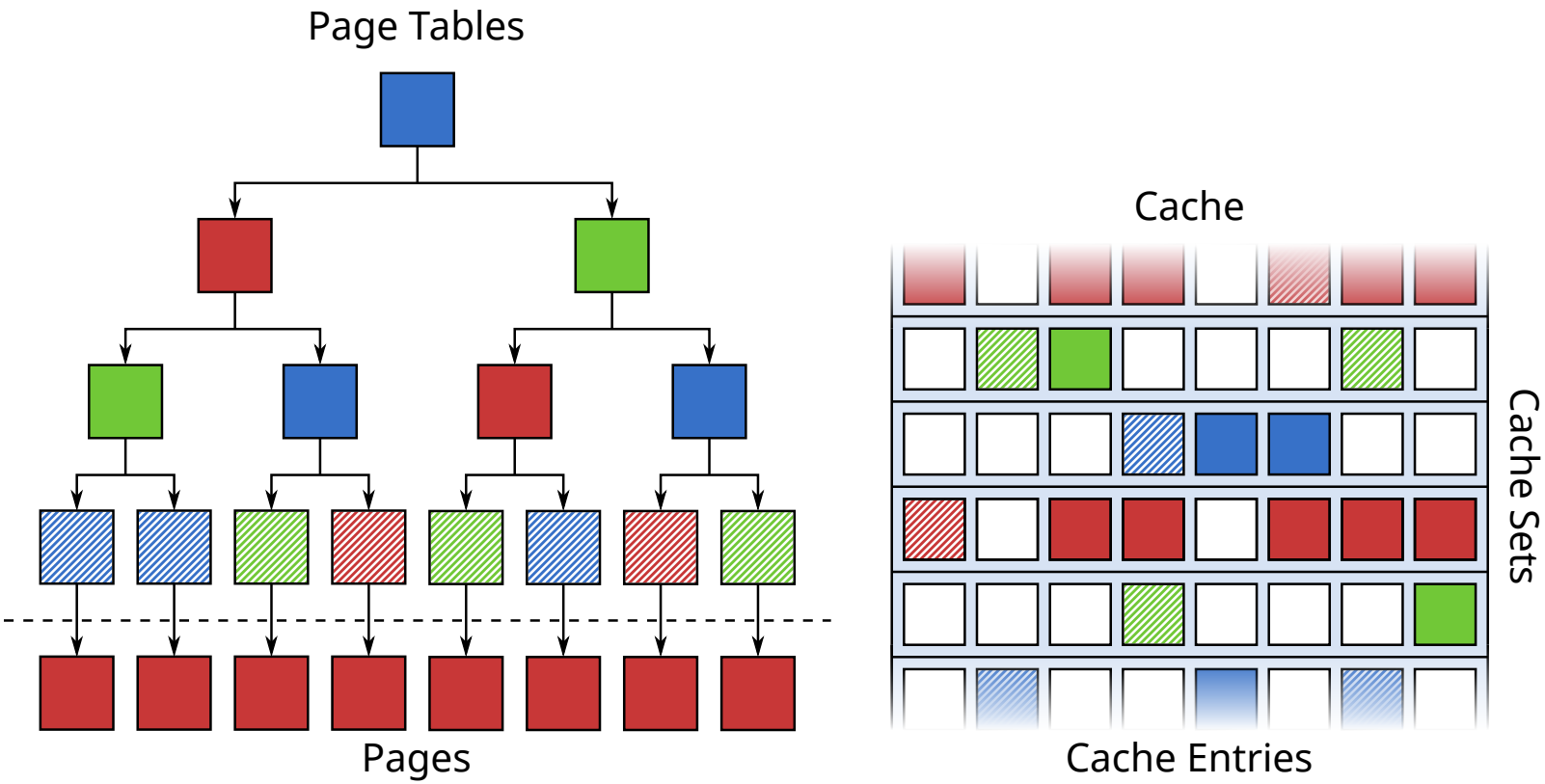
However, the page tables aren't colored

Page coloring



However, the page tables aren't colored

Page coloring



Can we control the page tables for cache attacks?

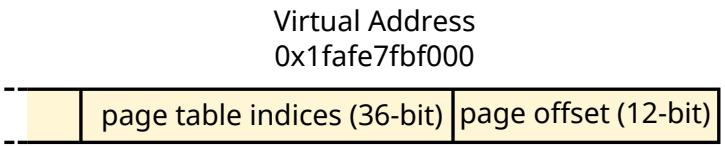
XLATE attacks

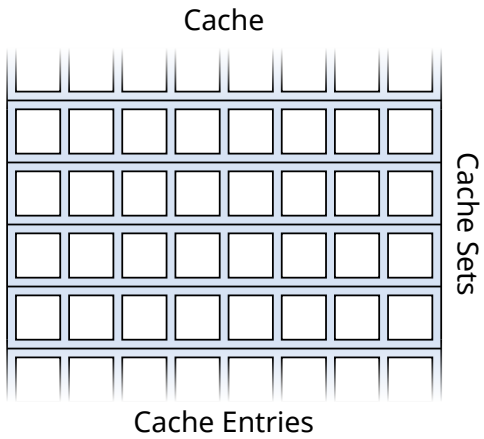
- Memory Management Unit (MMU)
- Translates virtual addresses into their physical counterparts
- Hence translate or XLATE attacks
- XLATE + PROBE caches page tables instead of pages

Page table walks

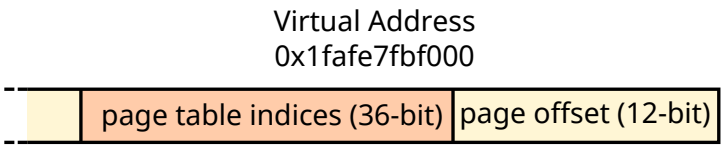
How does the MMU perform page table walks?

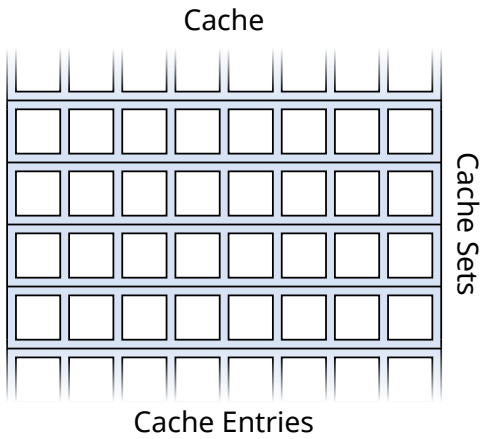
Page table walks



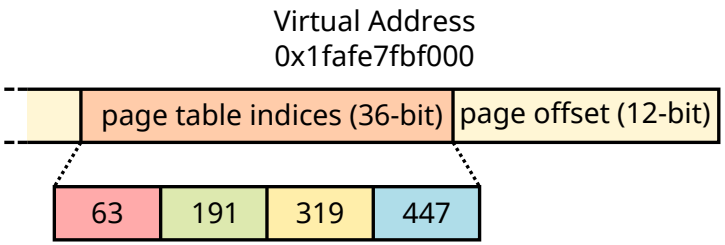


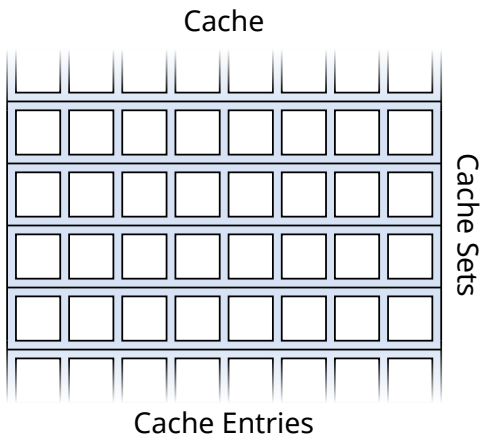
Page table walks



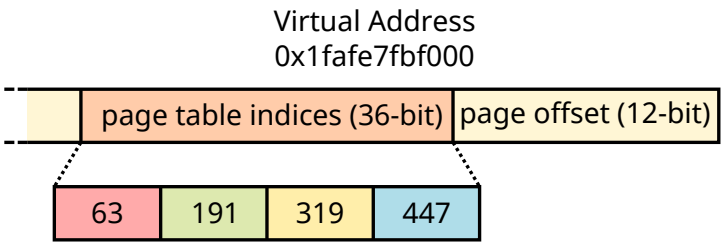


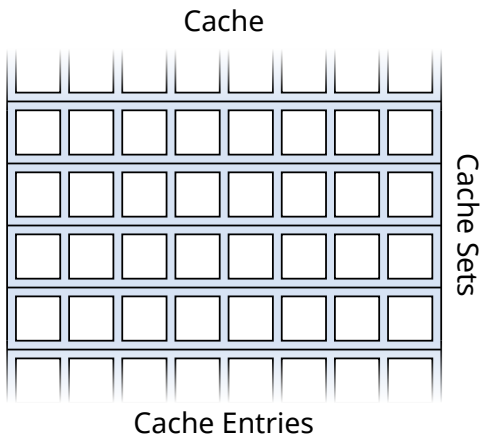
Page table walks



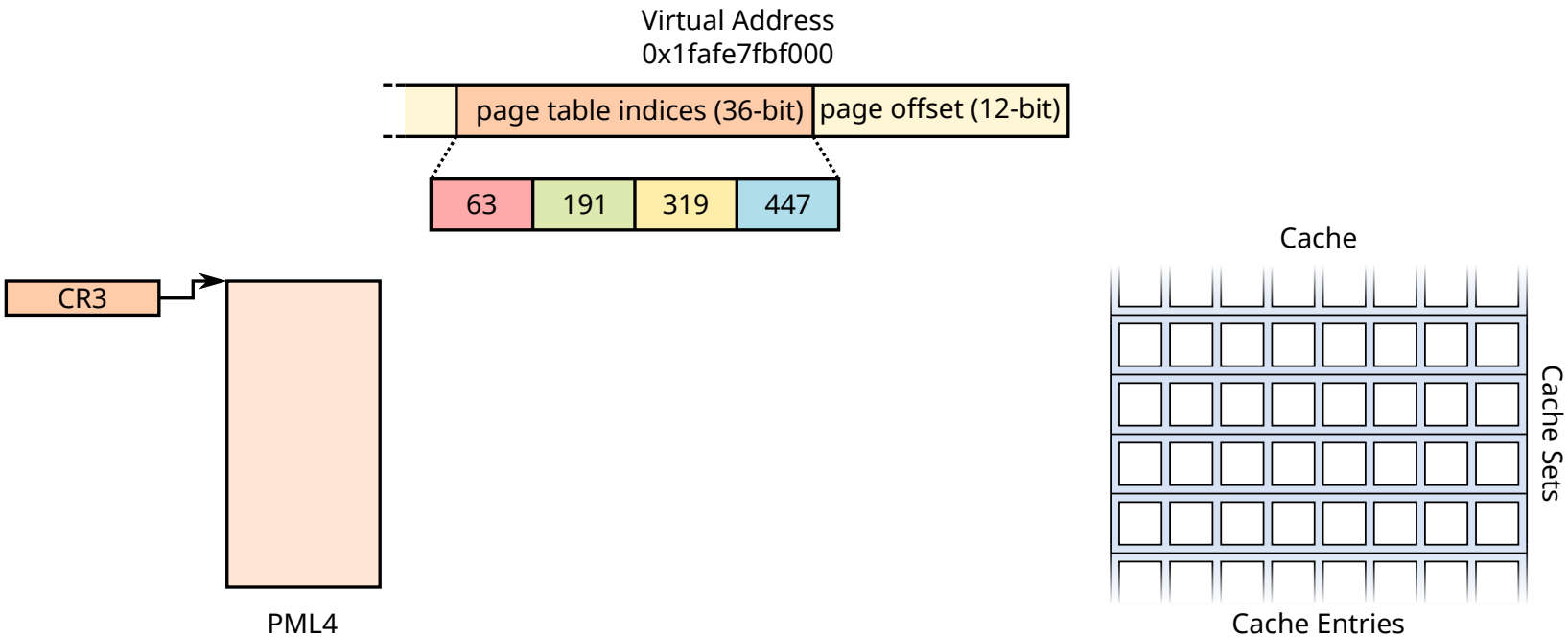


Page table walks

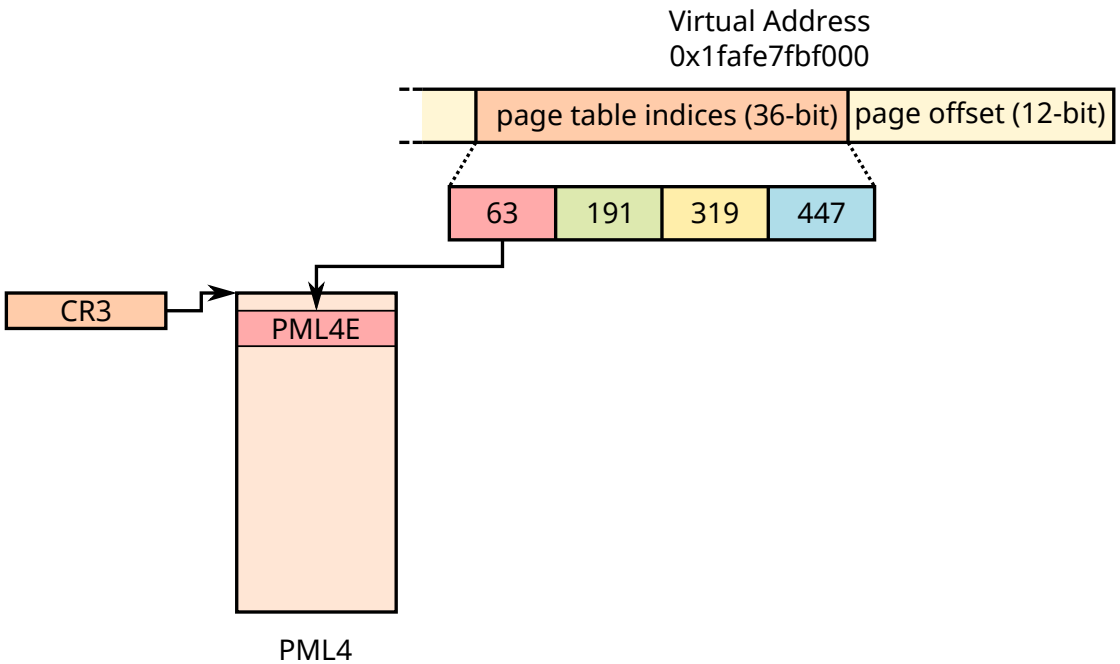


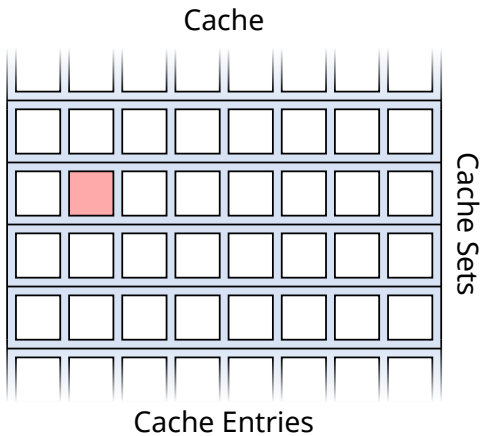


Page table walks

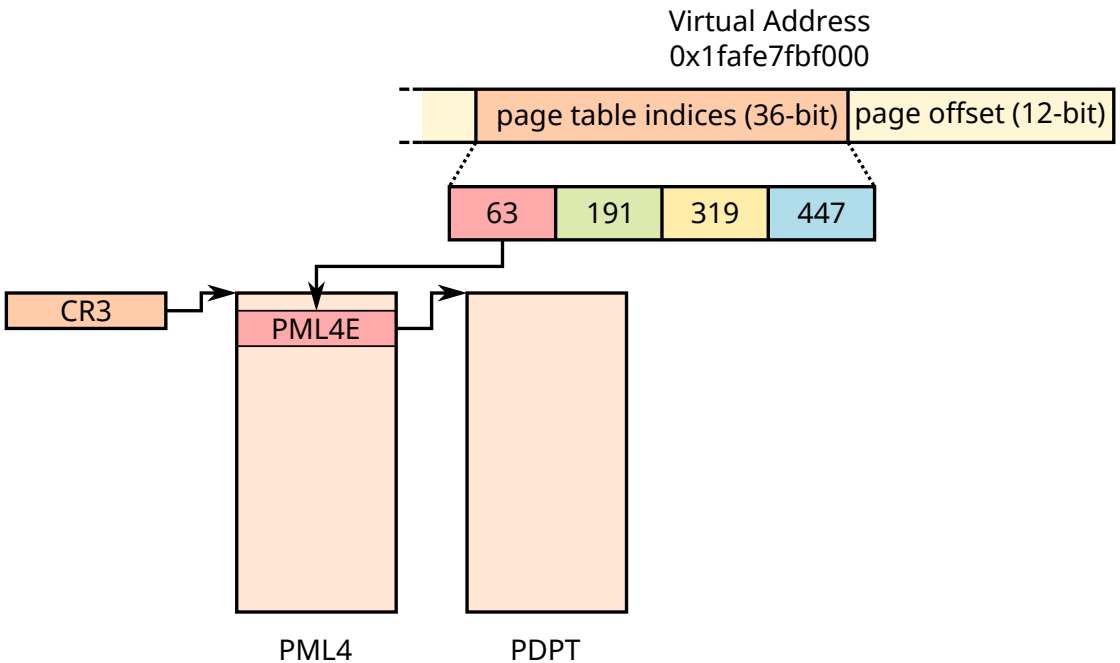


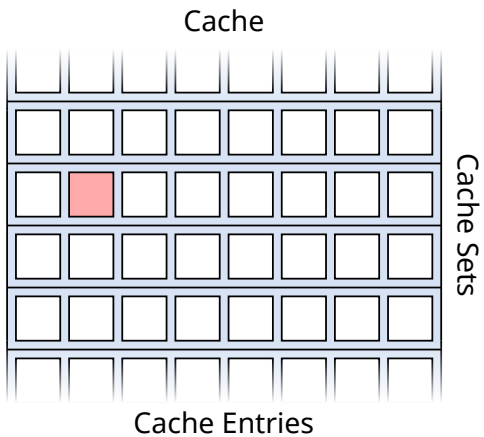
Page table walks



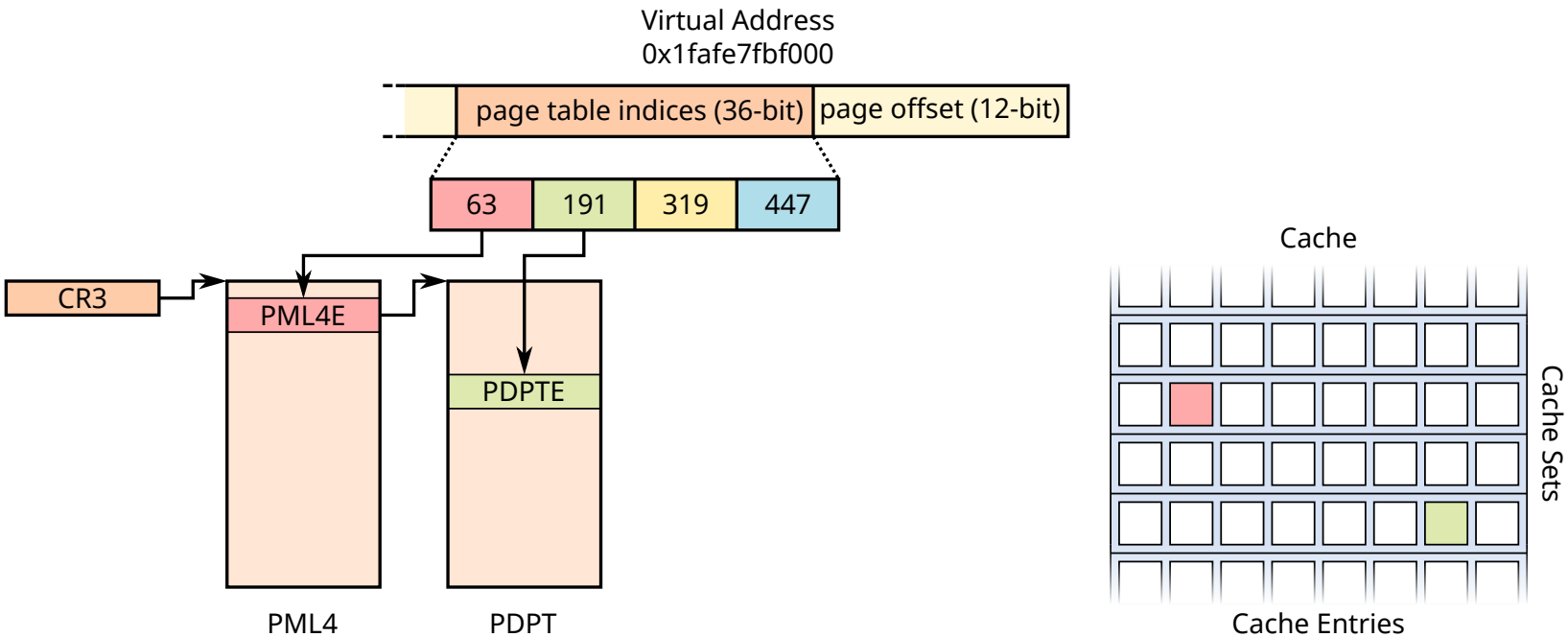


Page table walks

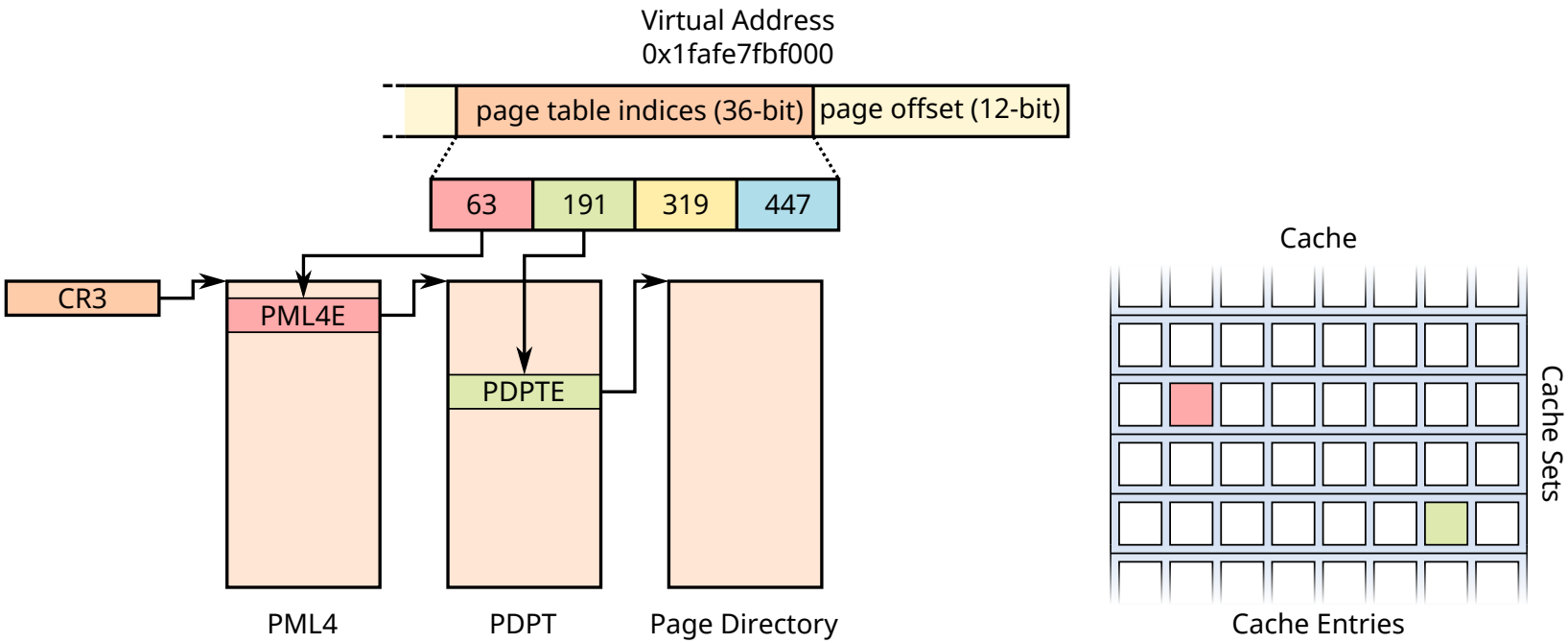




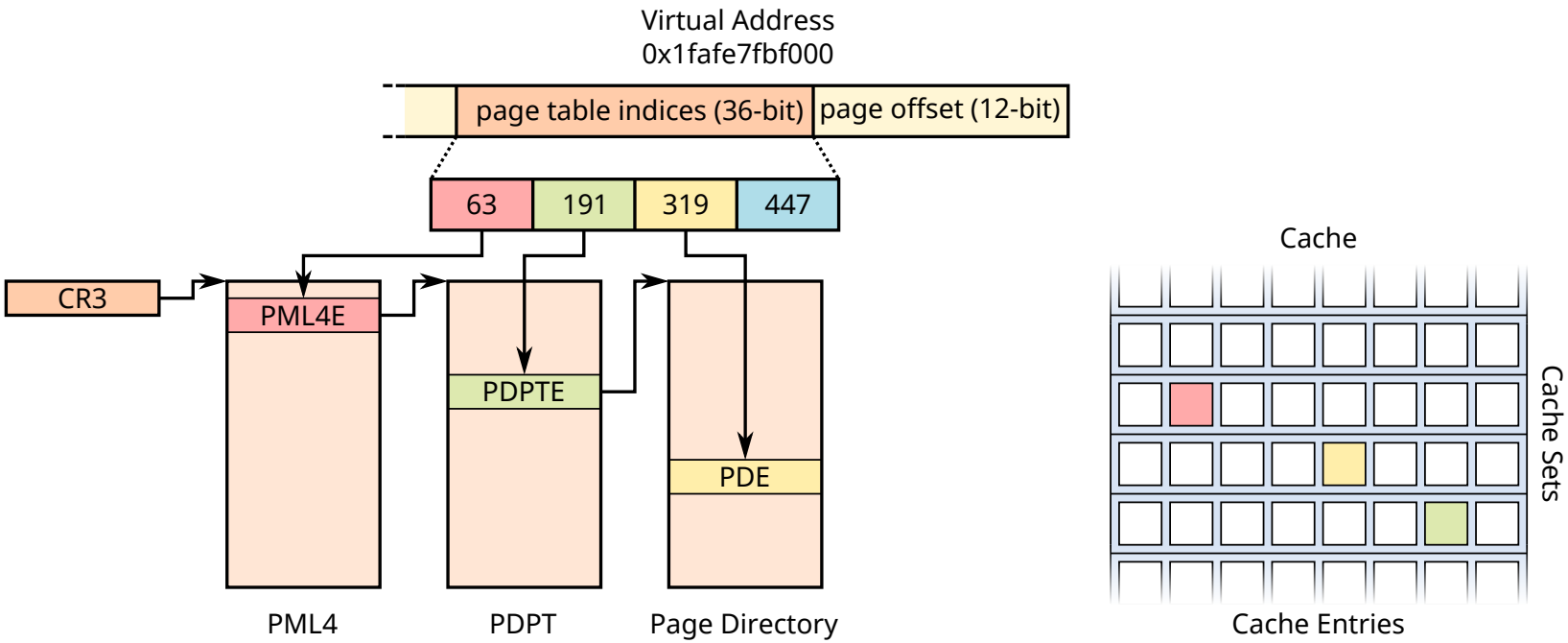
Page table walks



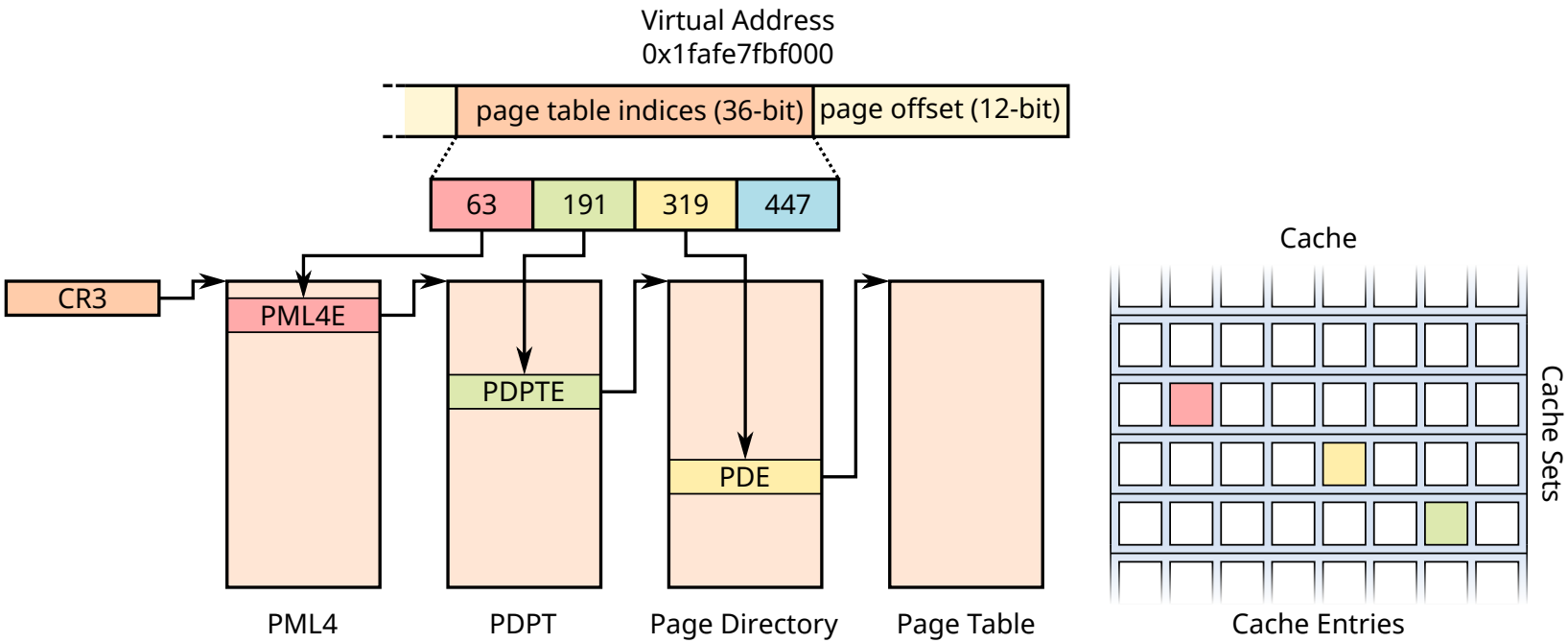
Page table walks



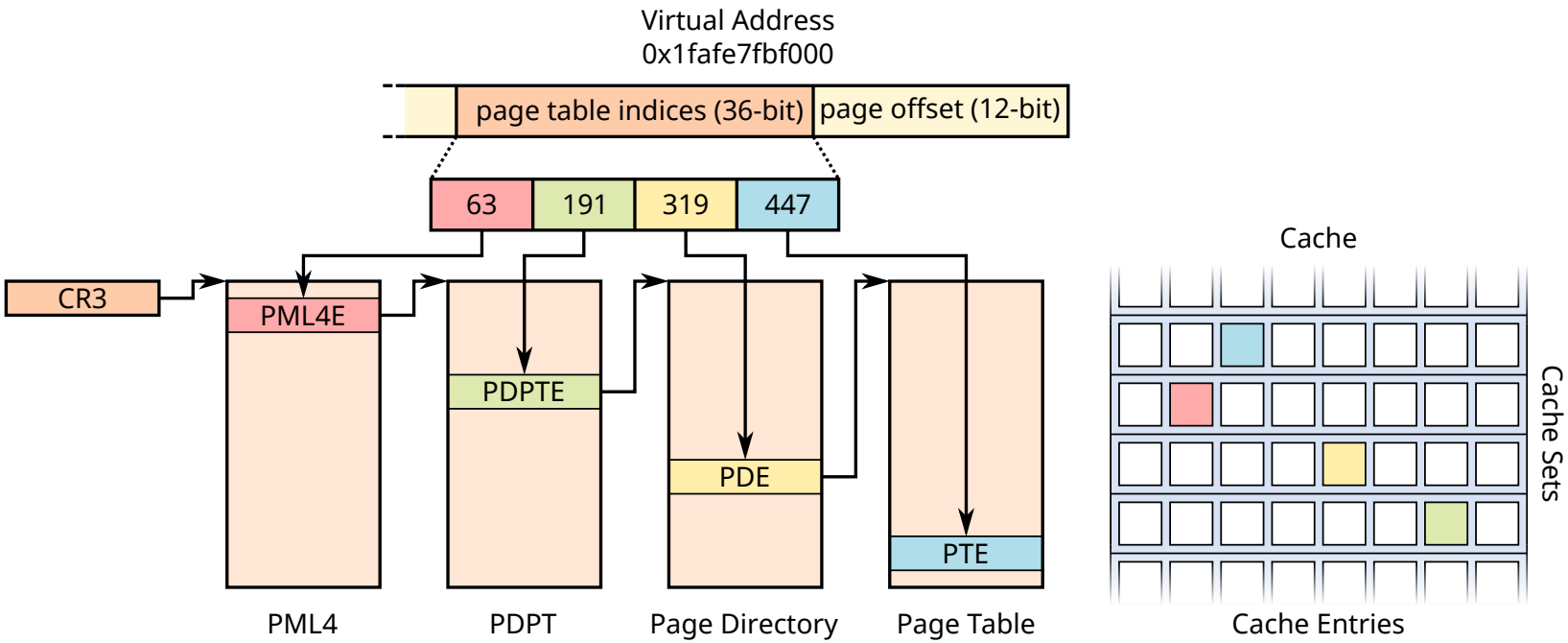
Page table walks



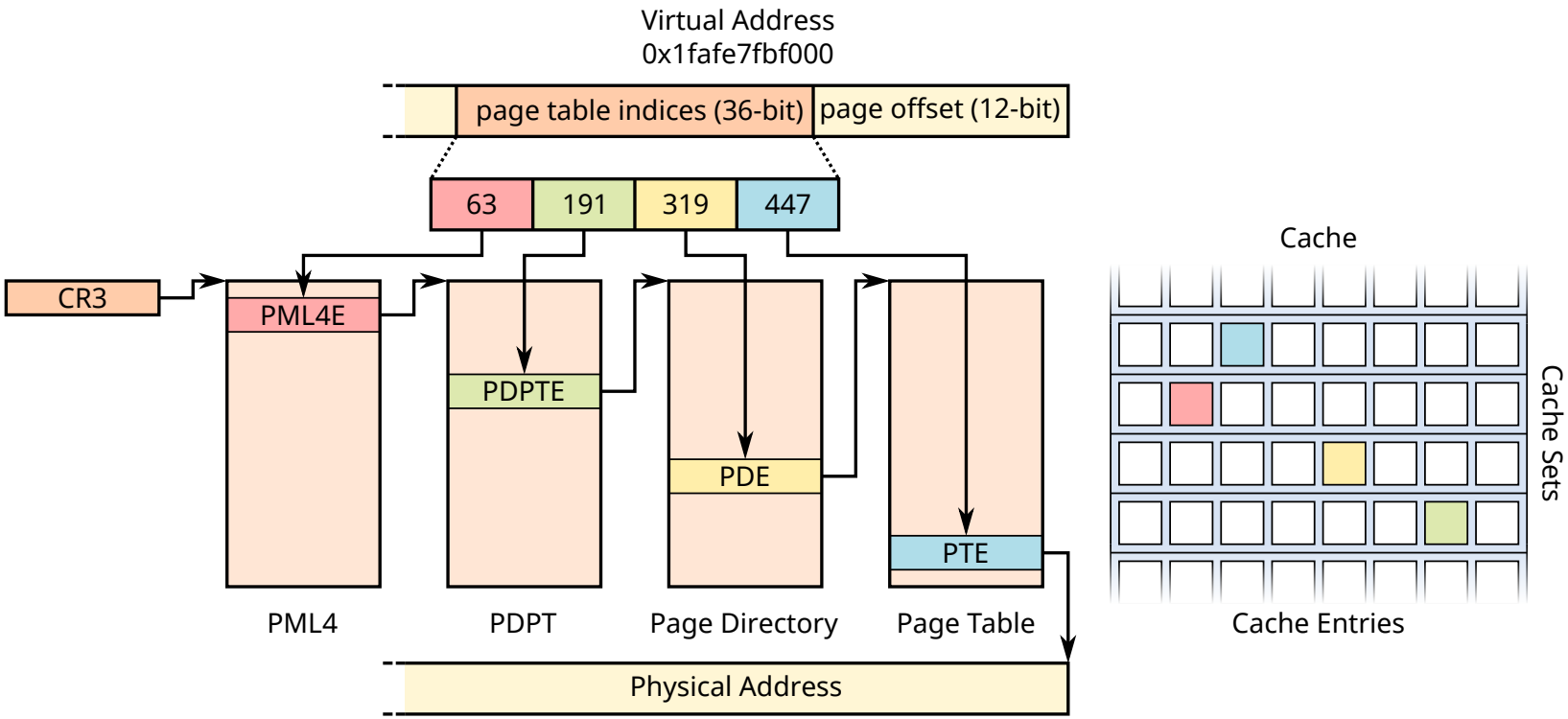
Page table walks



Page table walks



Page table walks



Page table walks

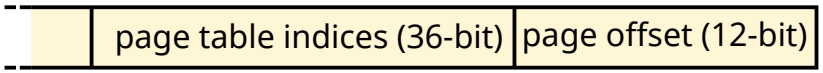
What do we need for XLATE + PROBE?

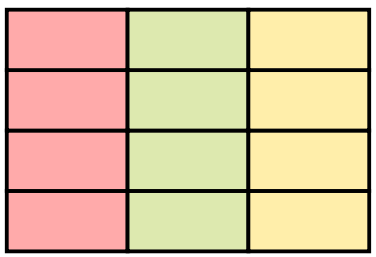
Challenges

- ✗ Avoid noise from high-level page tables
- ✗ Avoid noise from pages
- ✗ Build eviction sets

Translation Caches

Virtual Address
0x1fafe7fbf000

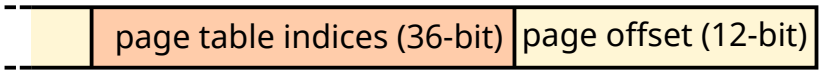


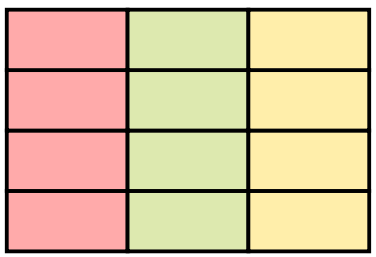


Translation Cache

Translation Caches

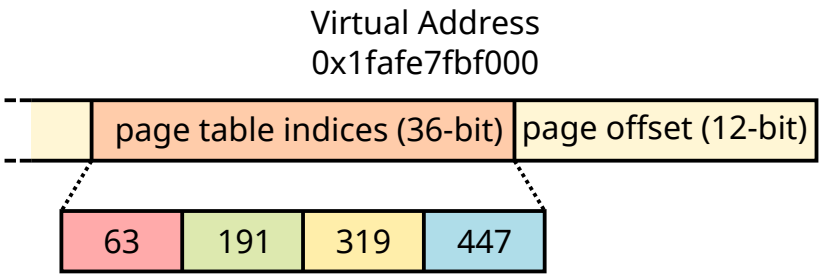
Virtual Address
0x1fafe7fbf000

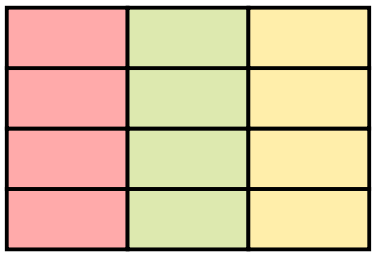




Translation Cache

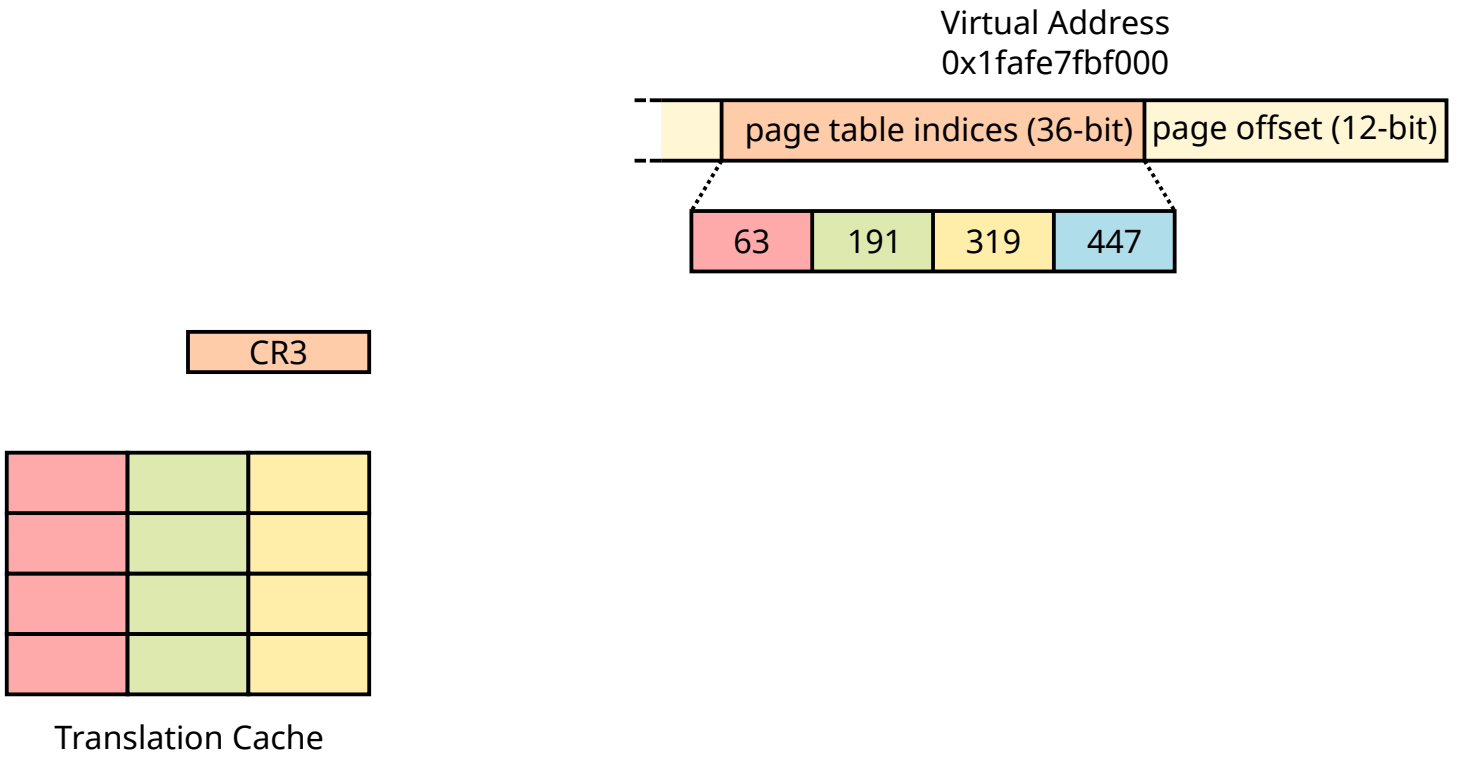
Translation Caches



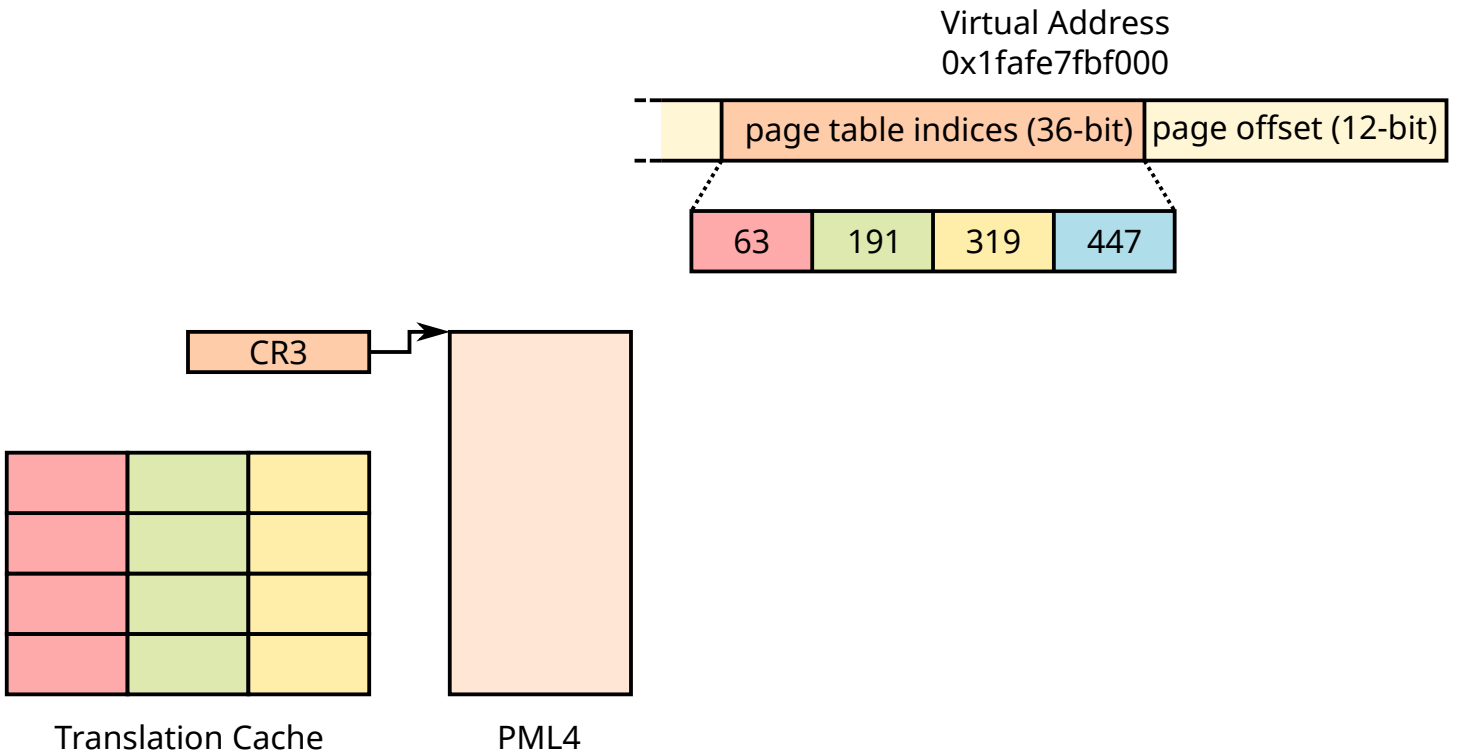


Translation Cache

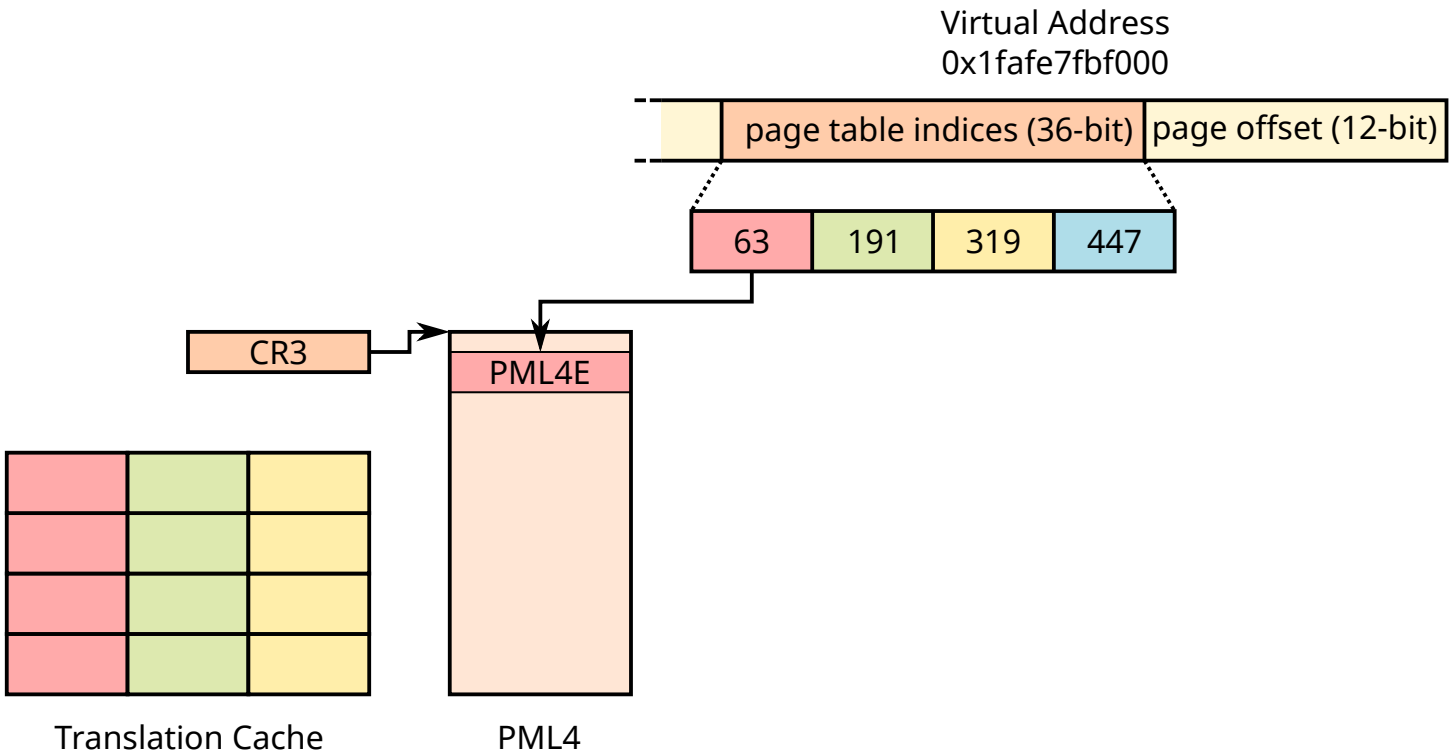
Translation Caches



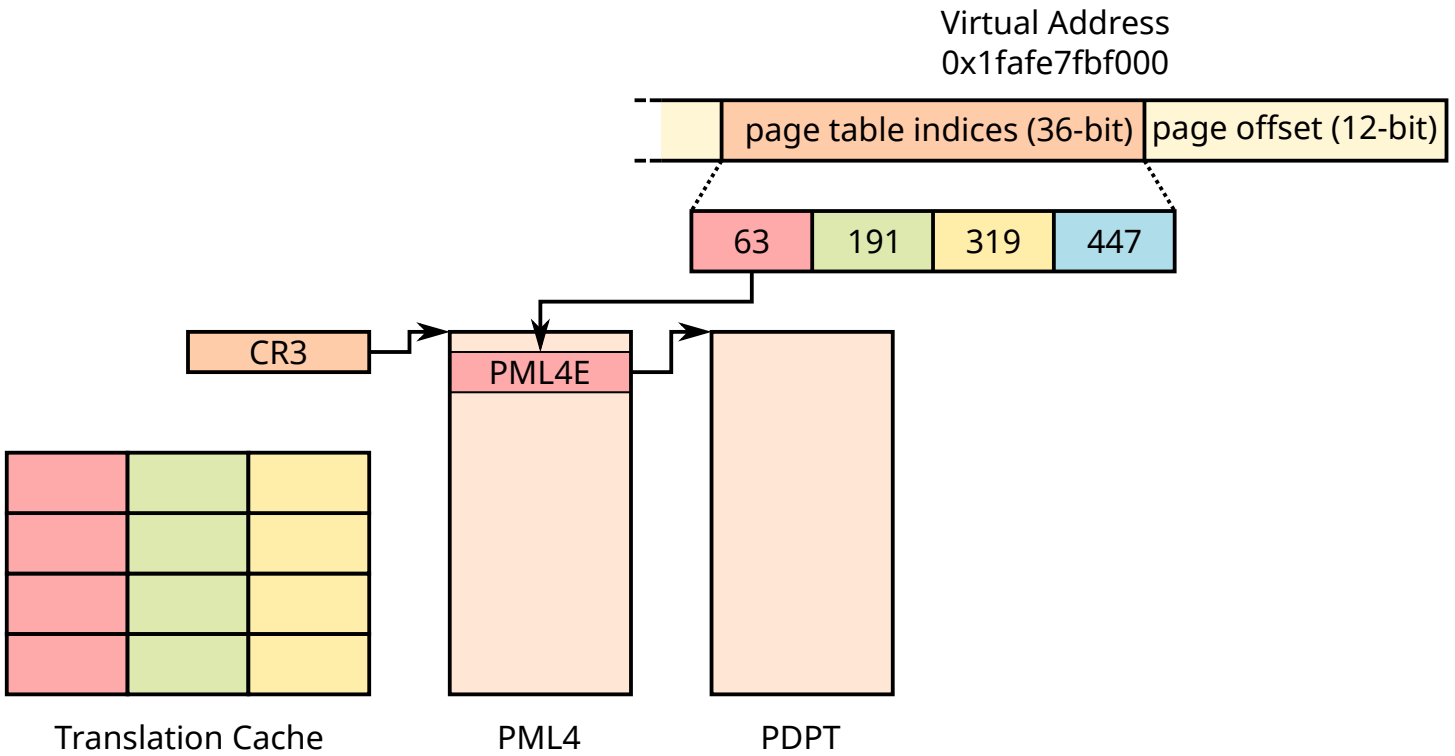
Translation Caches



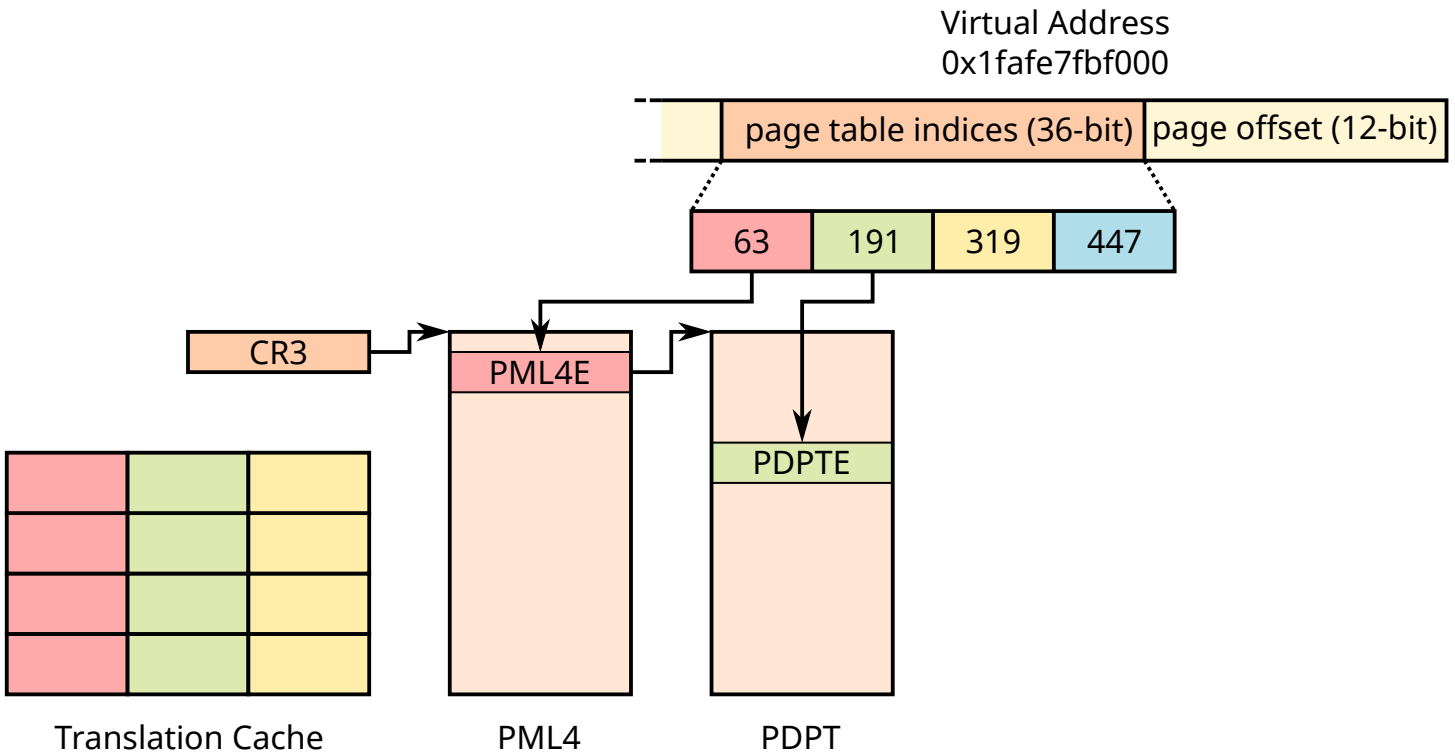
Translation Caches



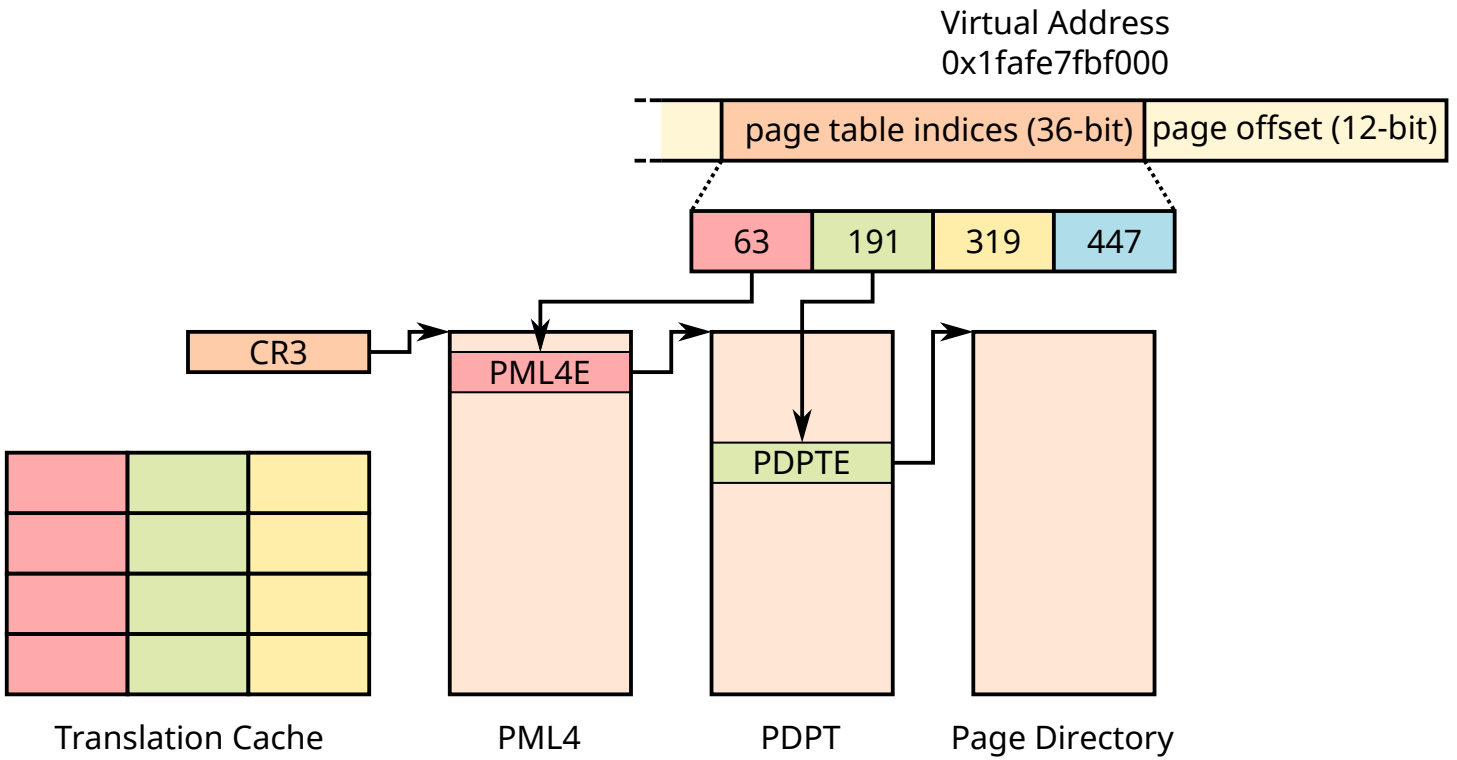
Translation Caches



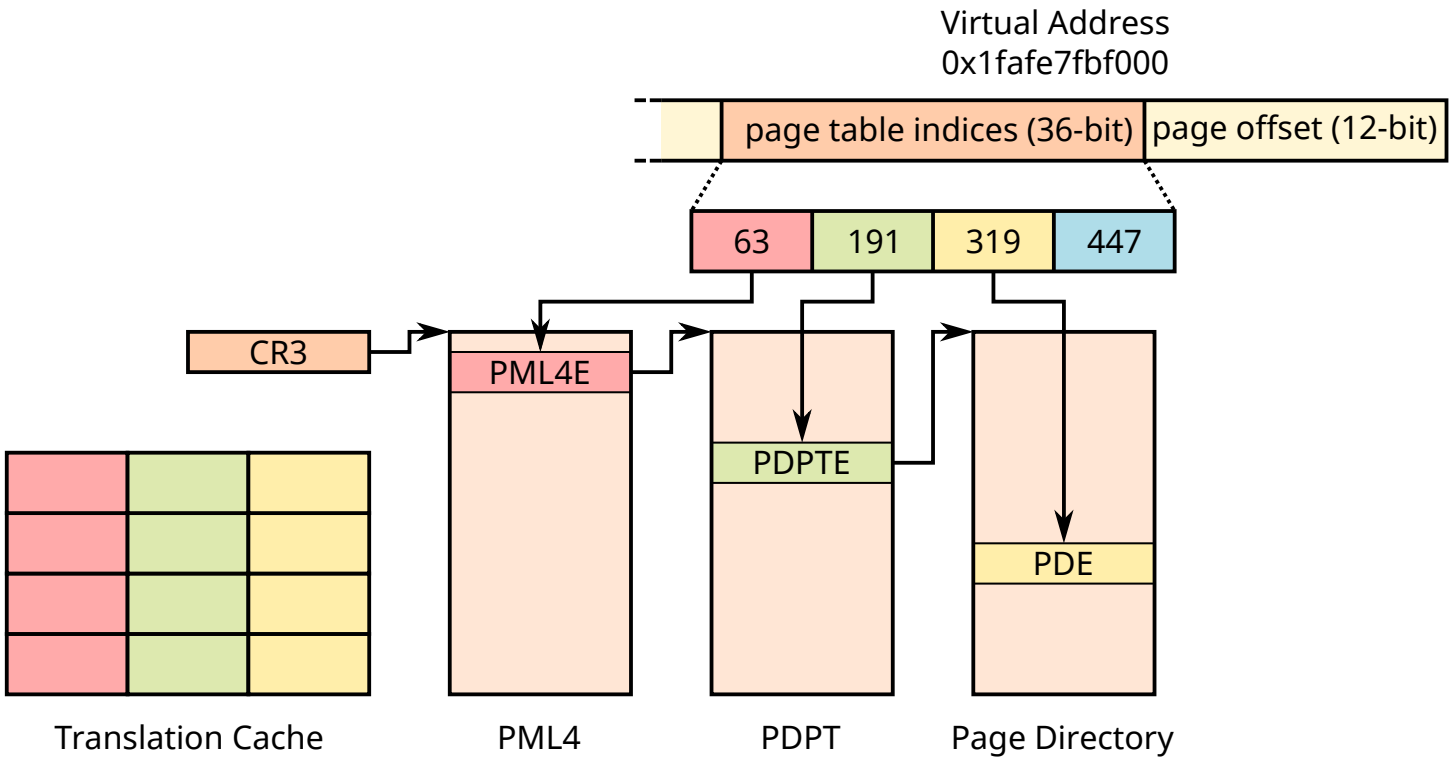
Translation Caches



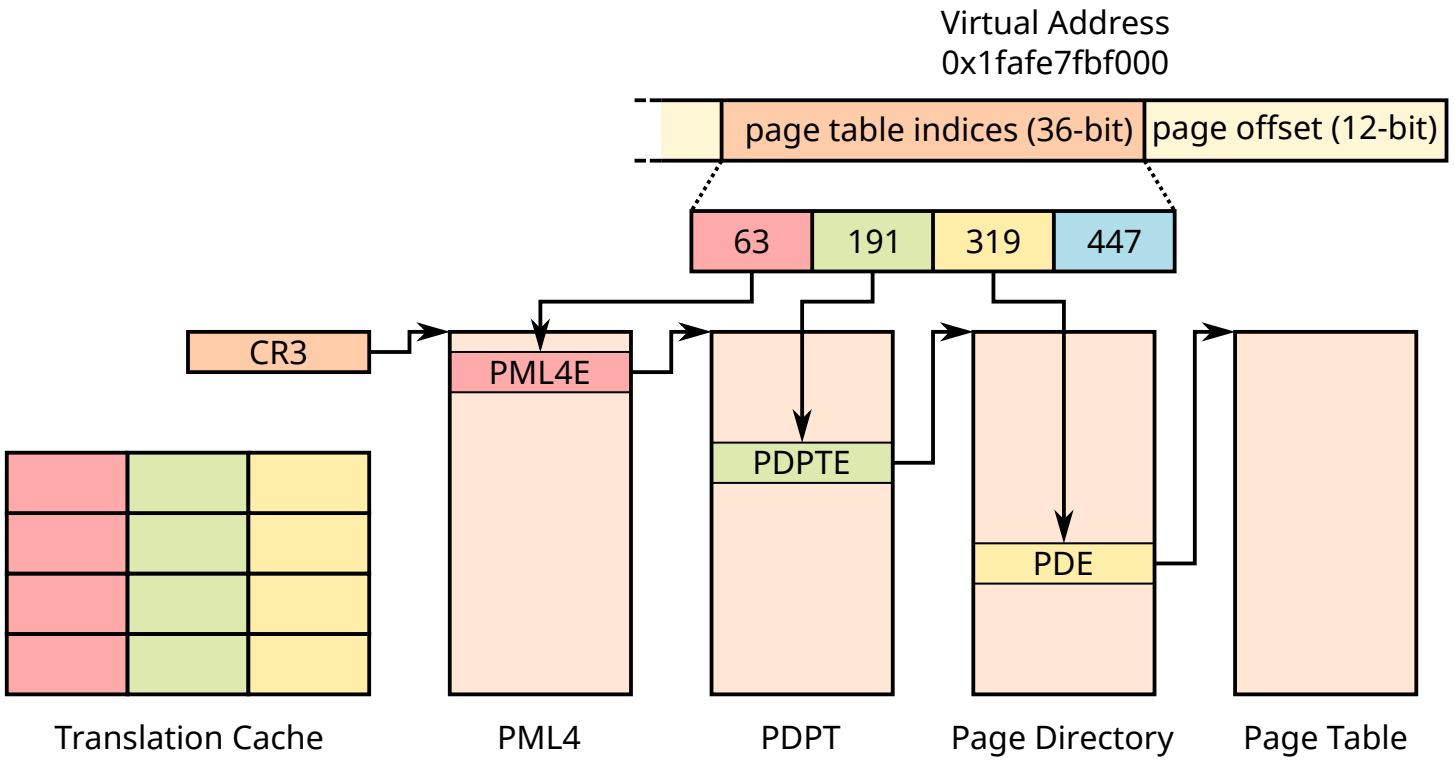
Translation Caches



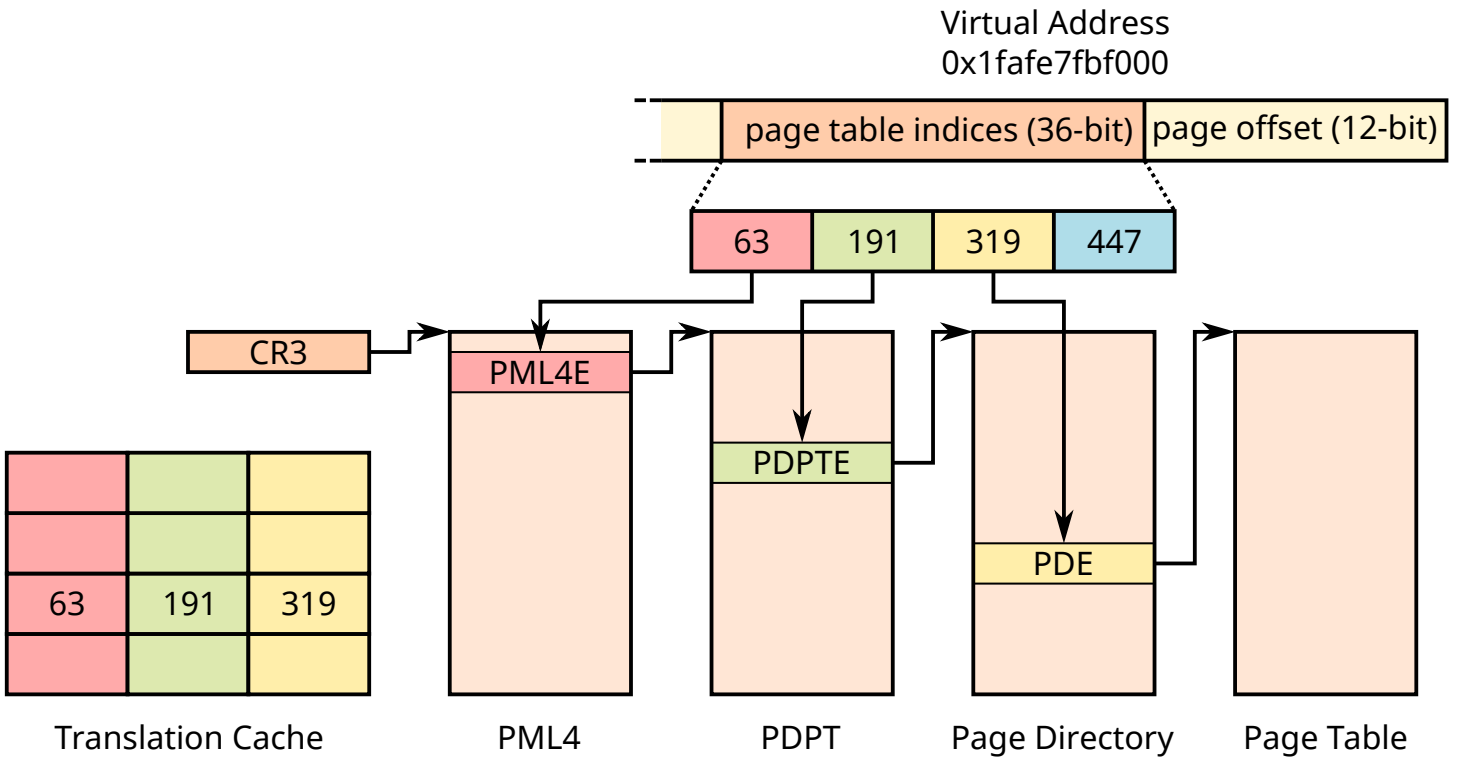
Translation Caches



Translation Caches

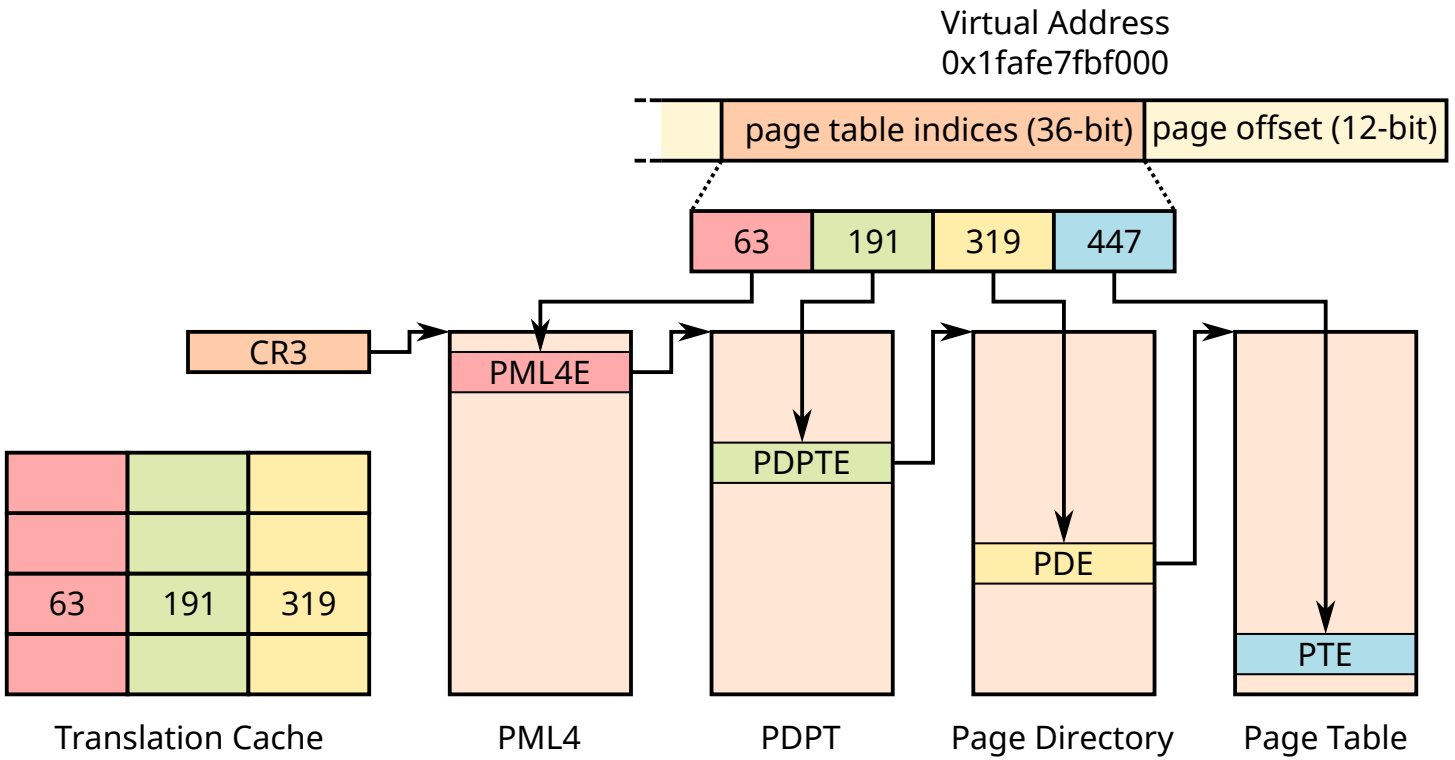


Translation Caches

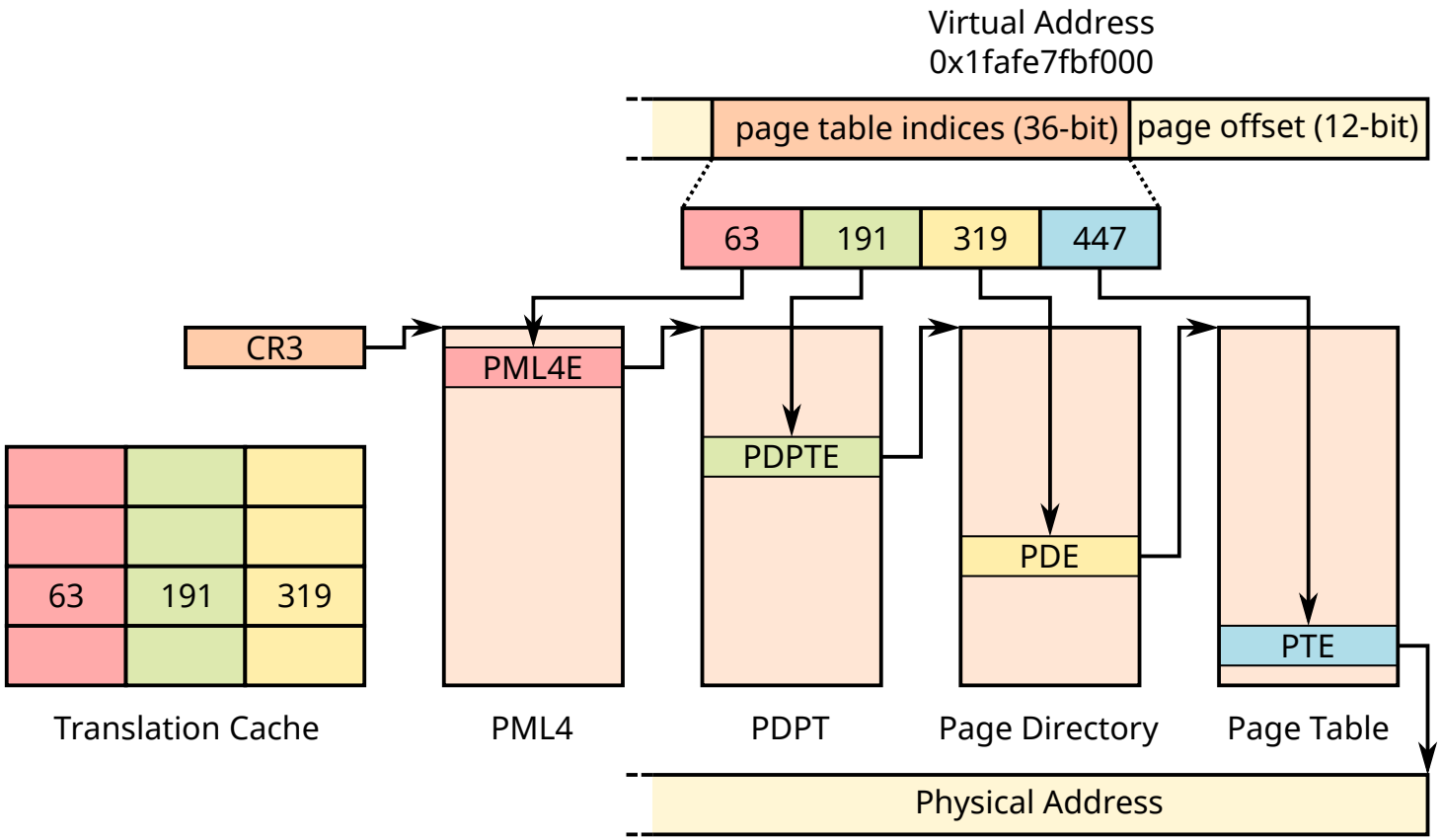


Translation caches cache intermediate page tables

Translation Caches

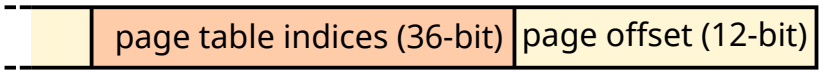


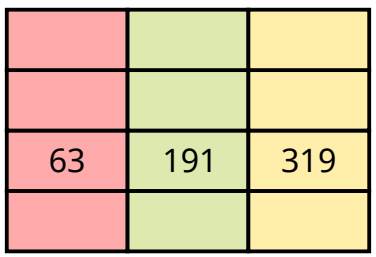
Translation Caches



Translation Caches

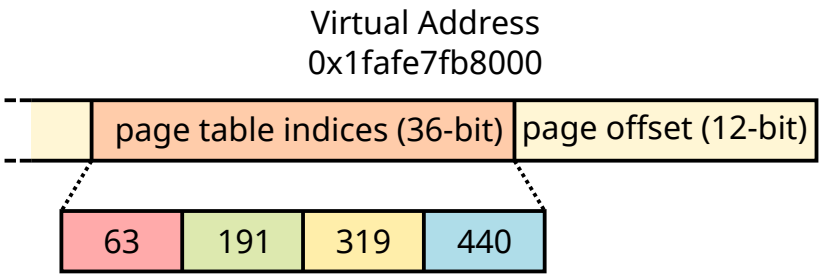
Virtual Address
0x1fafe7fb8000

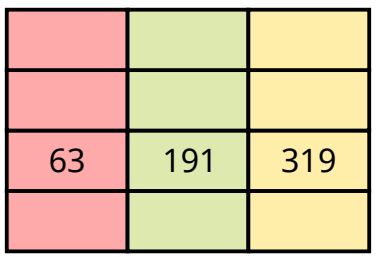




Translation Cache

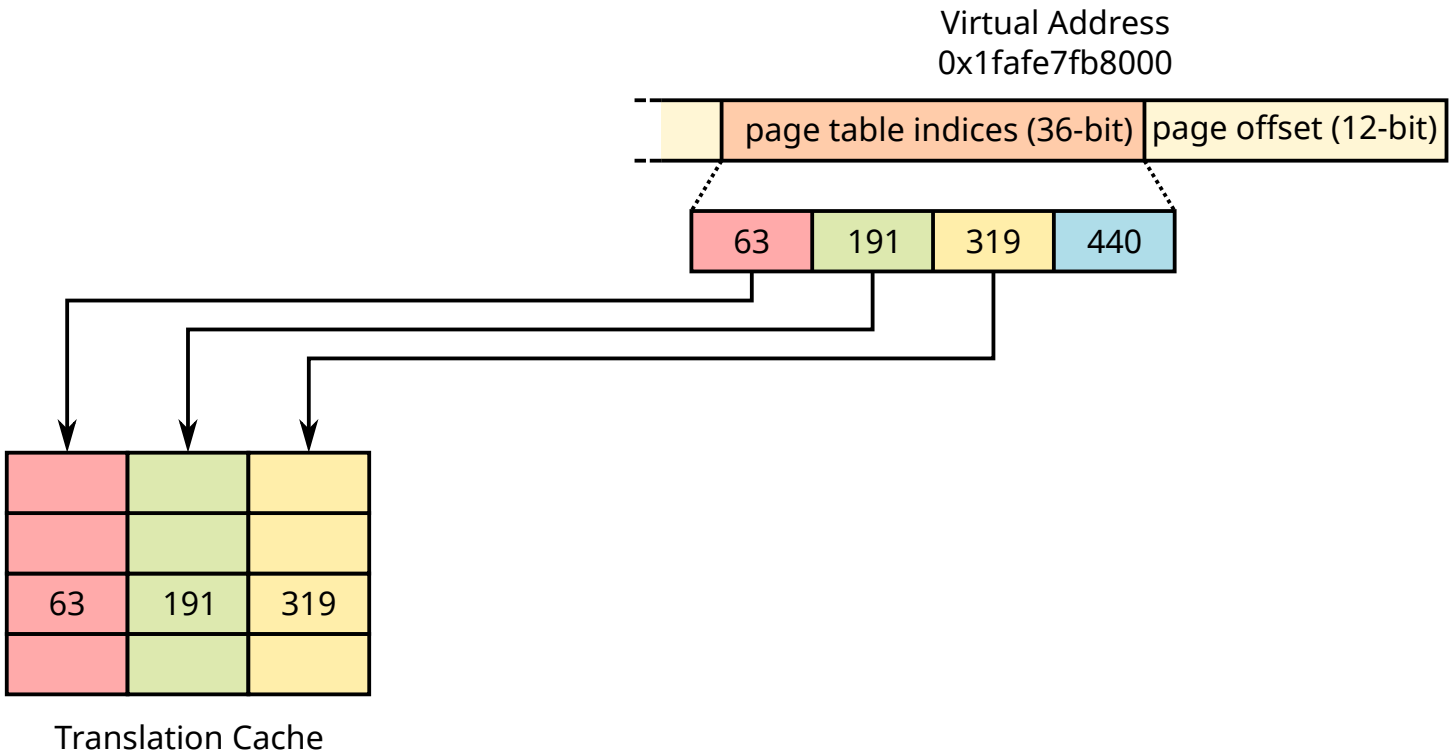
Translation Caches





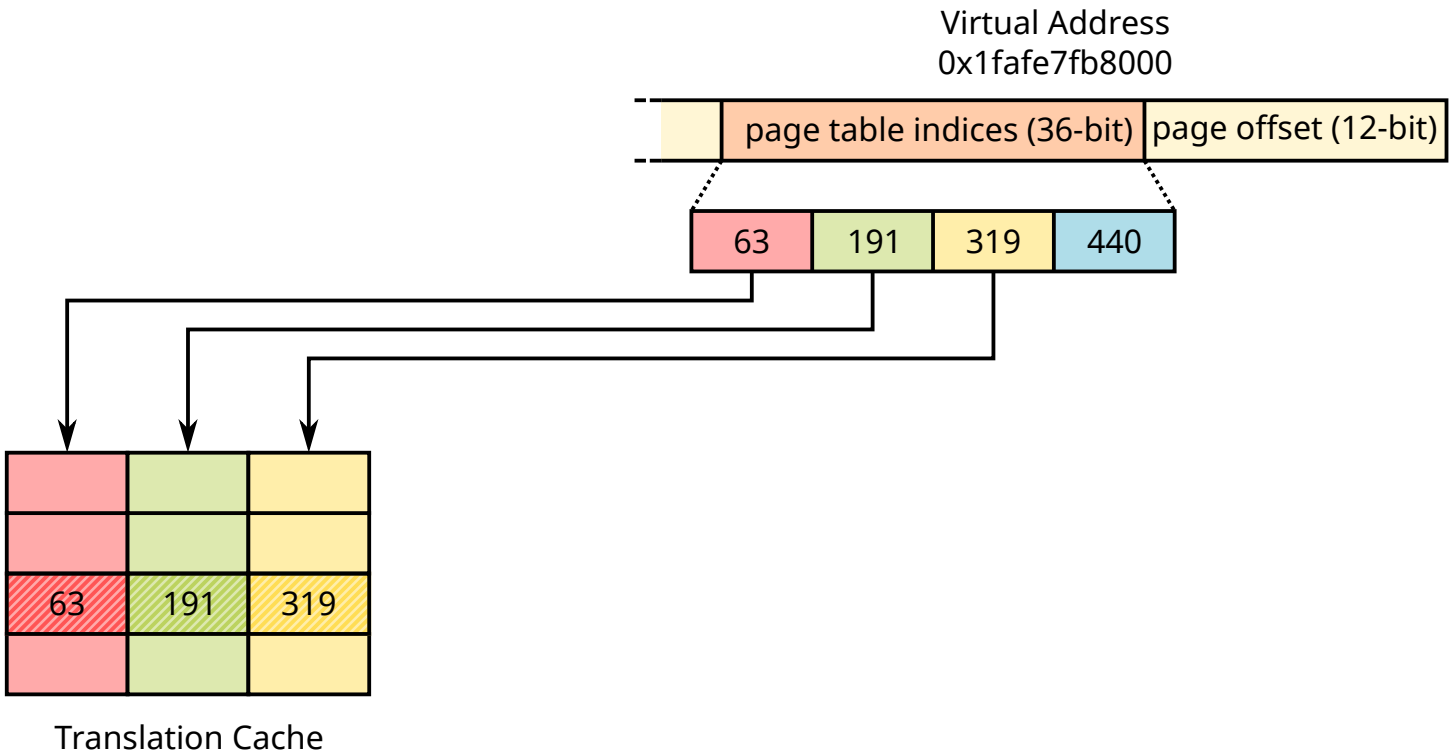
Translation Cache

Translation Caches



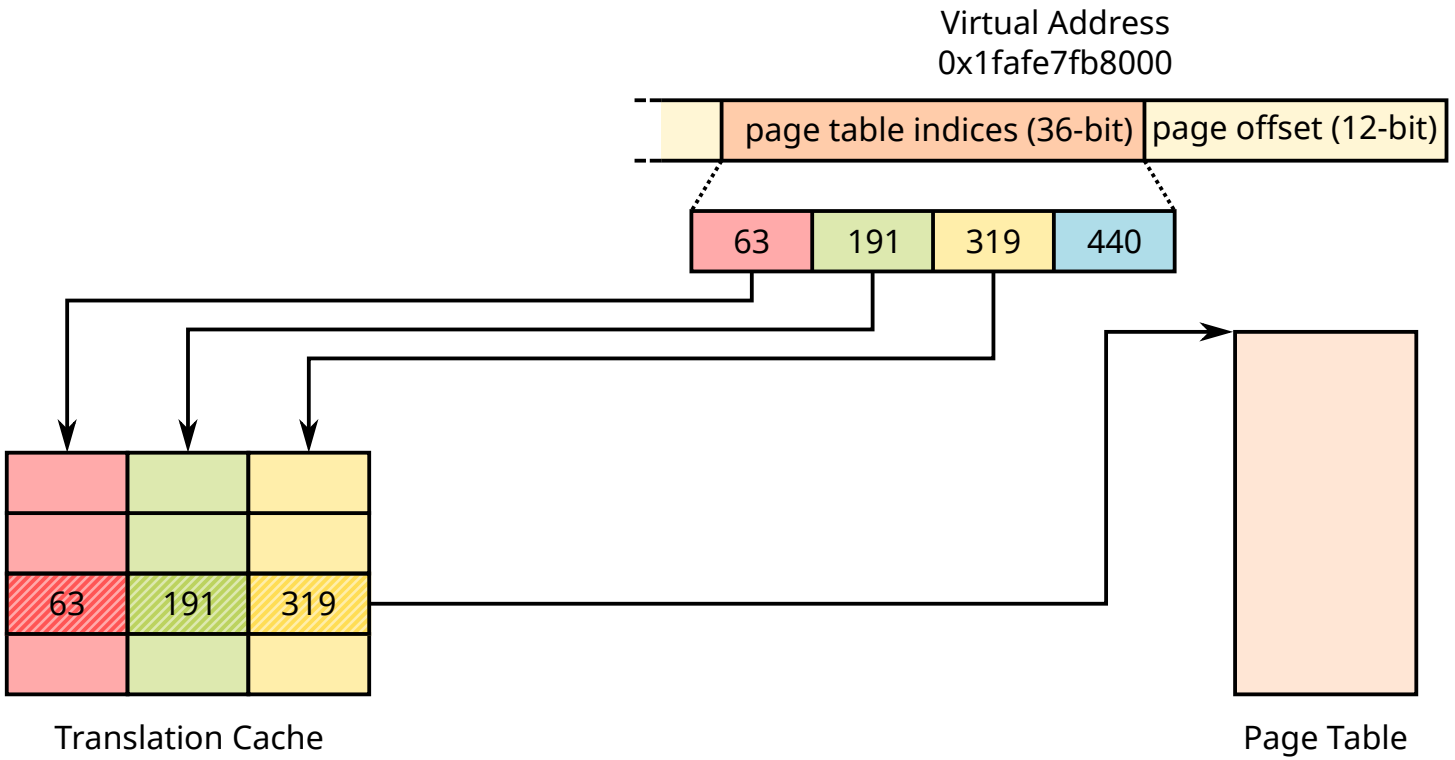
Translation caches cache intermediate page tables

Translation Caches



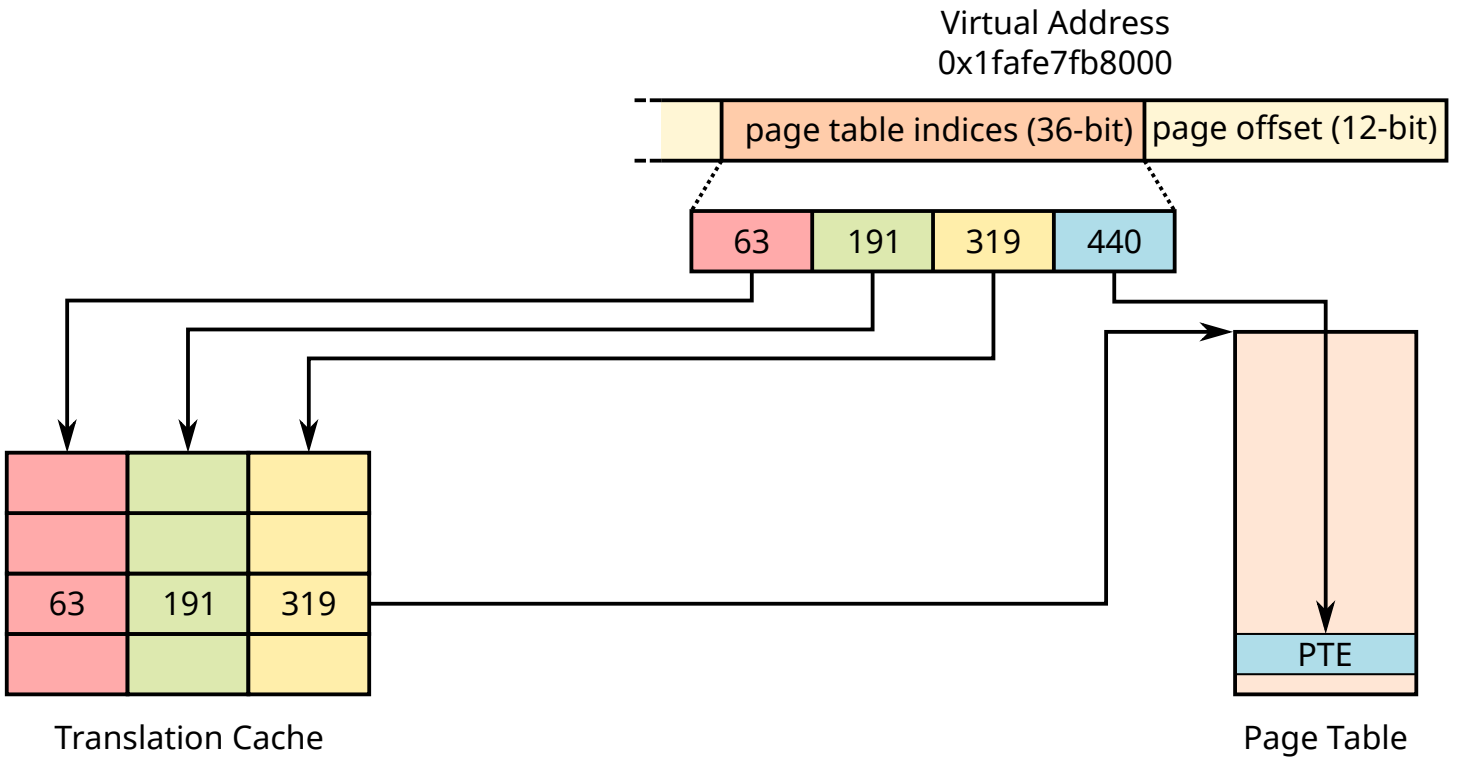
Translation caches cache intermediate page tables

Translation Caches

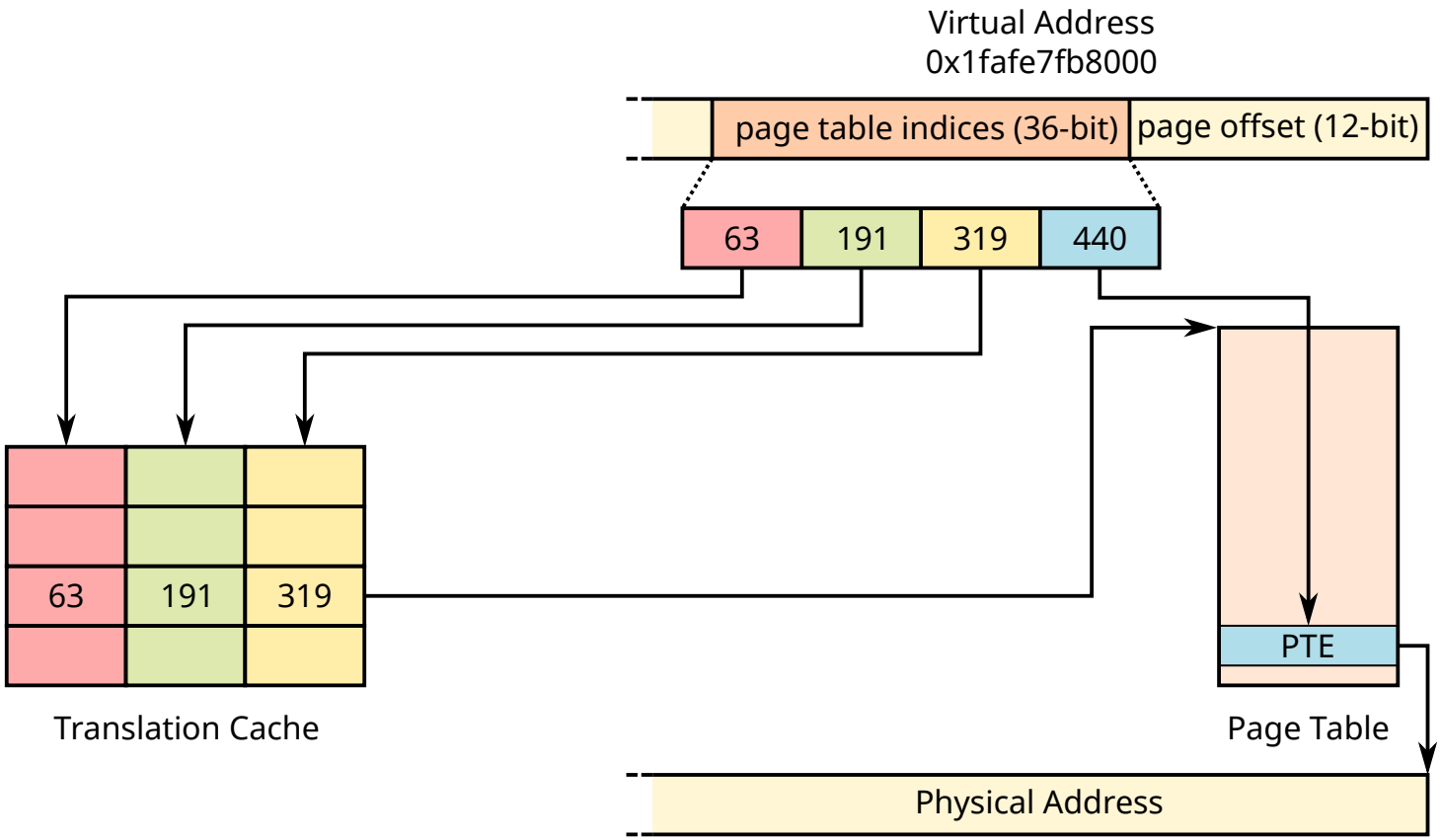


Translation caches cache intermediate page tables

Translation Caches



Translation Caches



Translation Caches

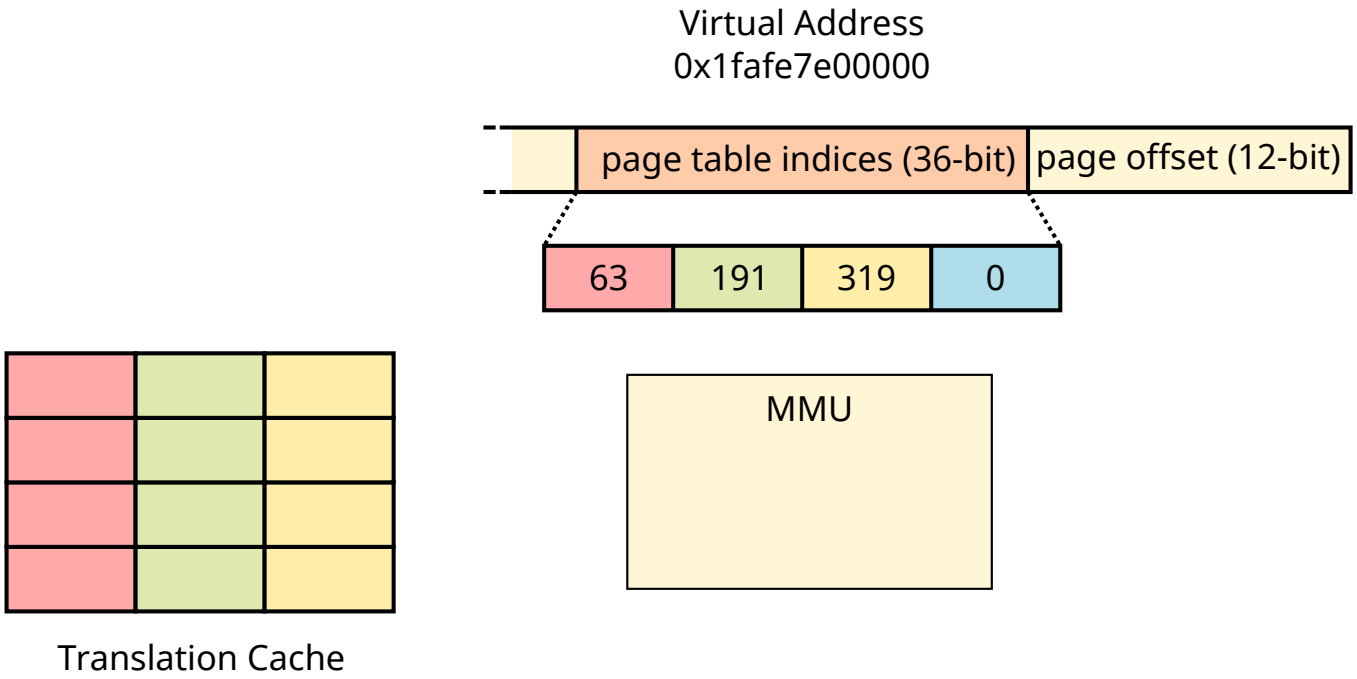
Some properties of translation caches are undocumented

Translation Caches

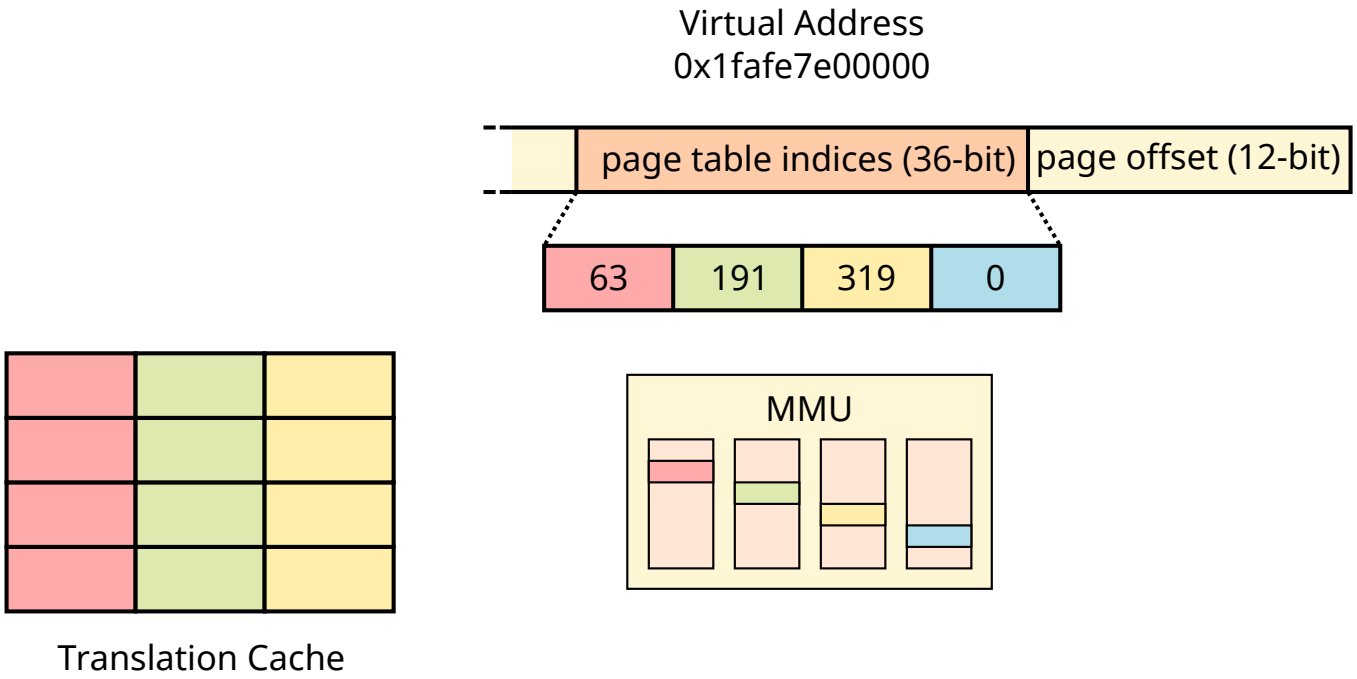
Some properties of translation caches are undocumented

How do we reverse engineer them?

Translation Caches

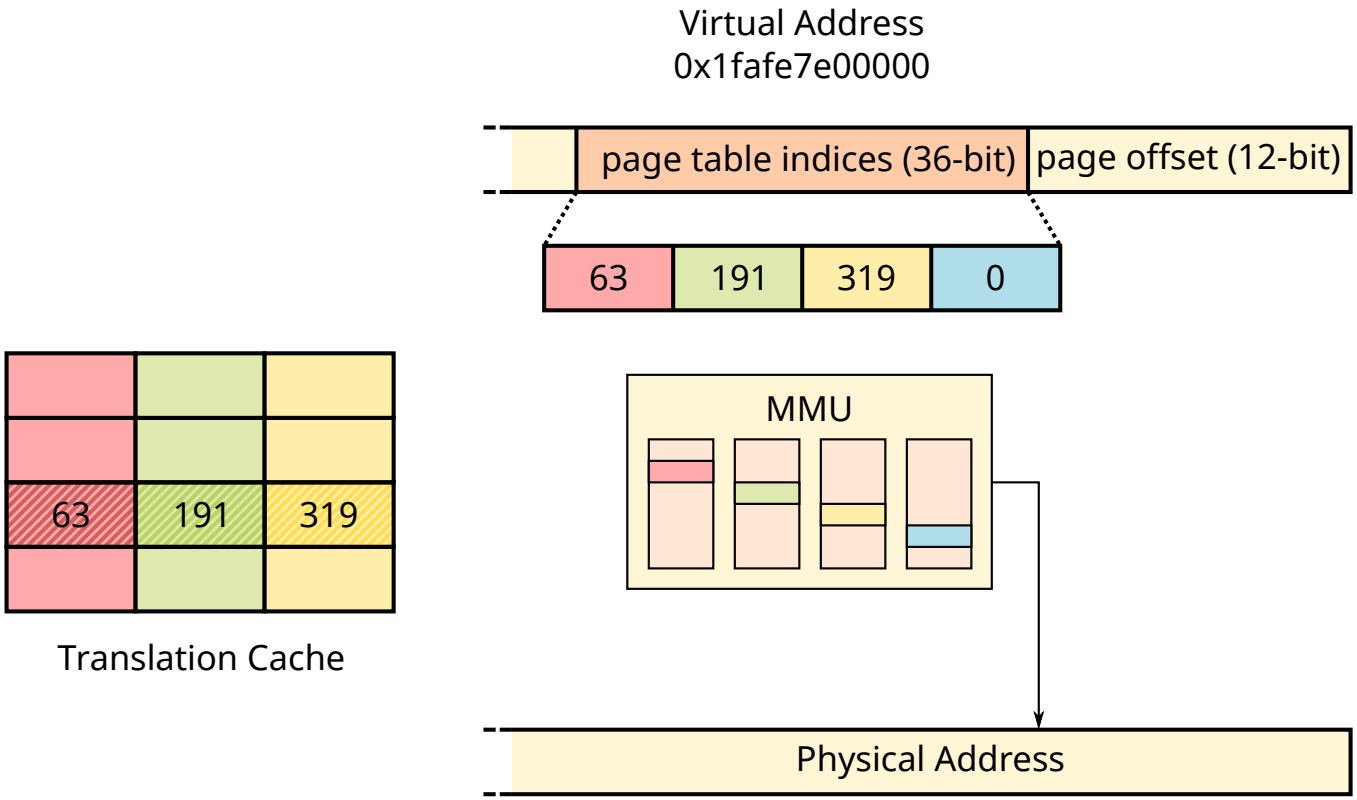


Translation Caches



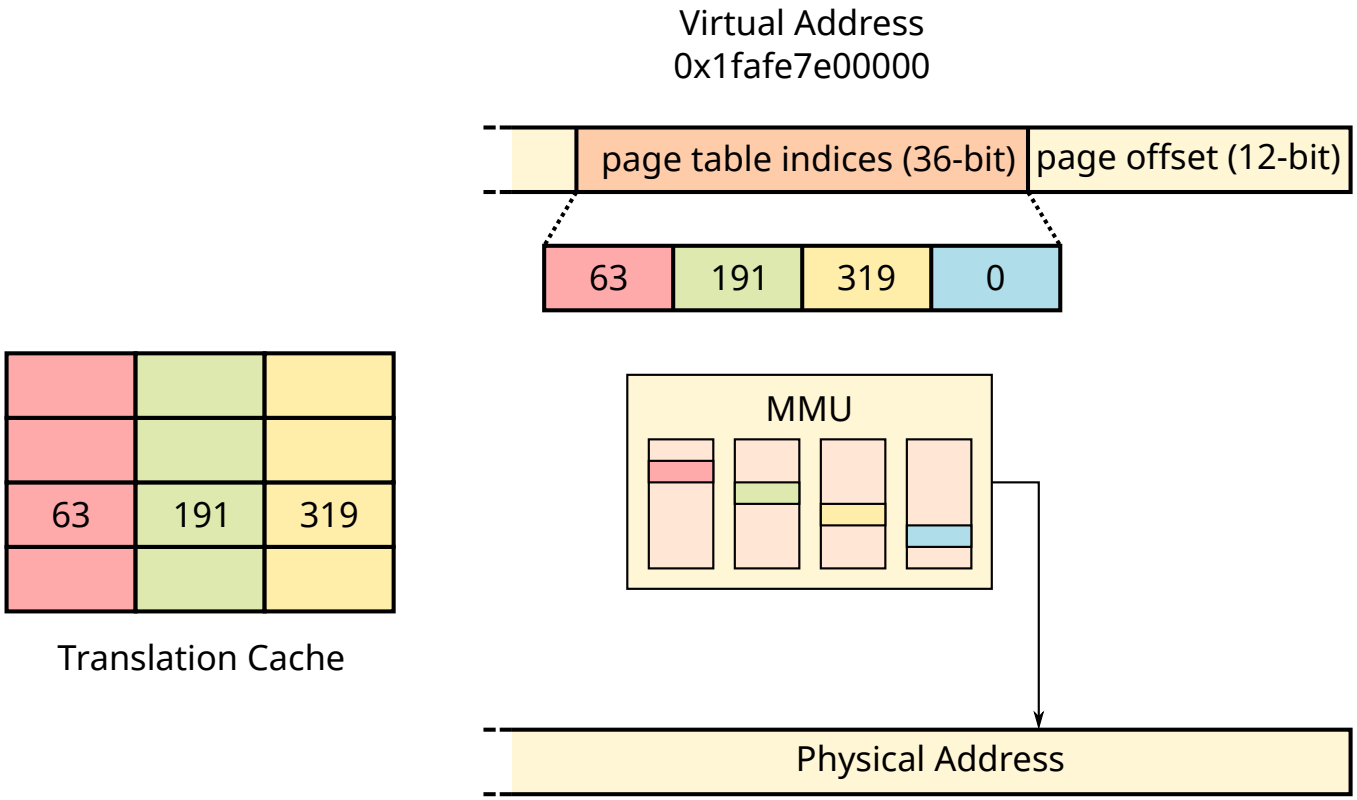
Load into translation cache

Translation Caches



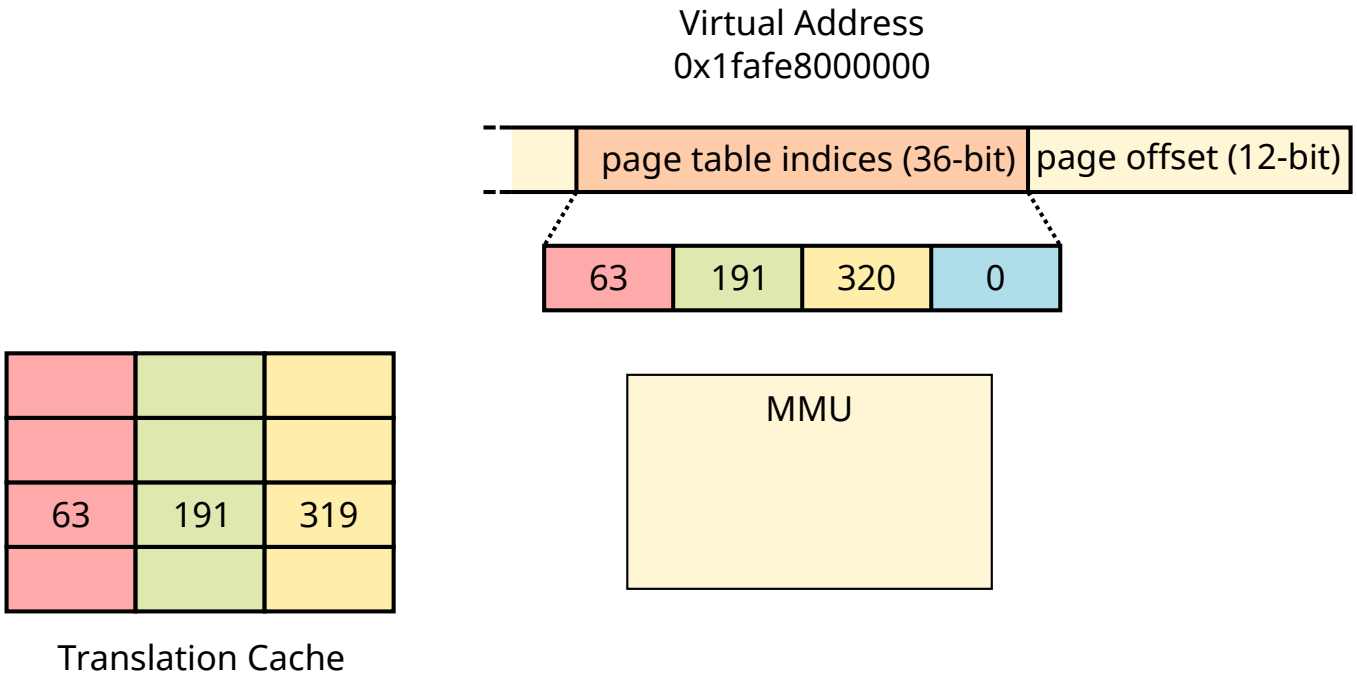
Load into translation cache

Translation Caches



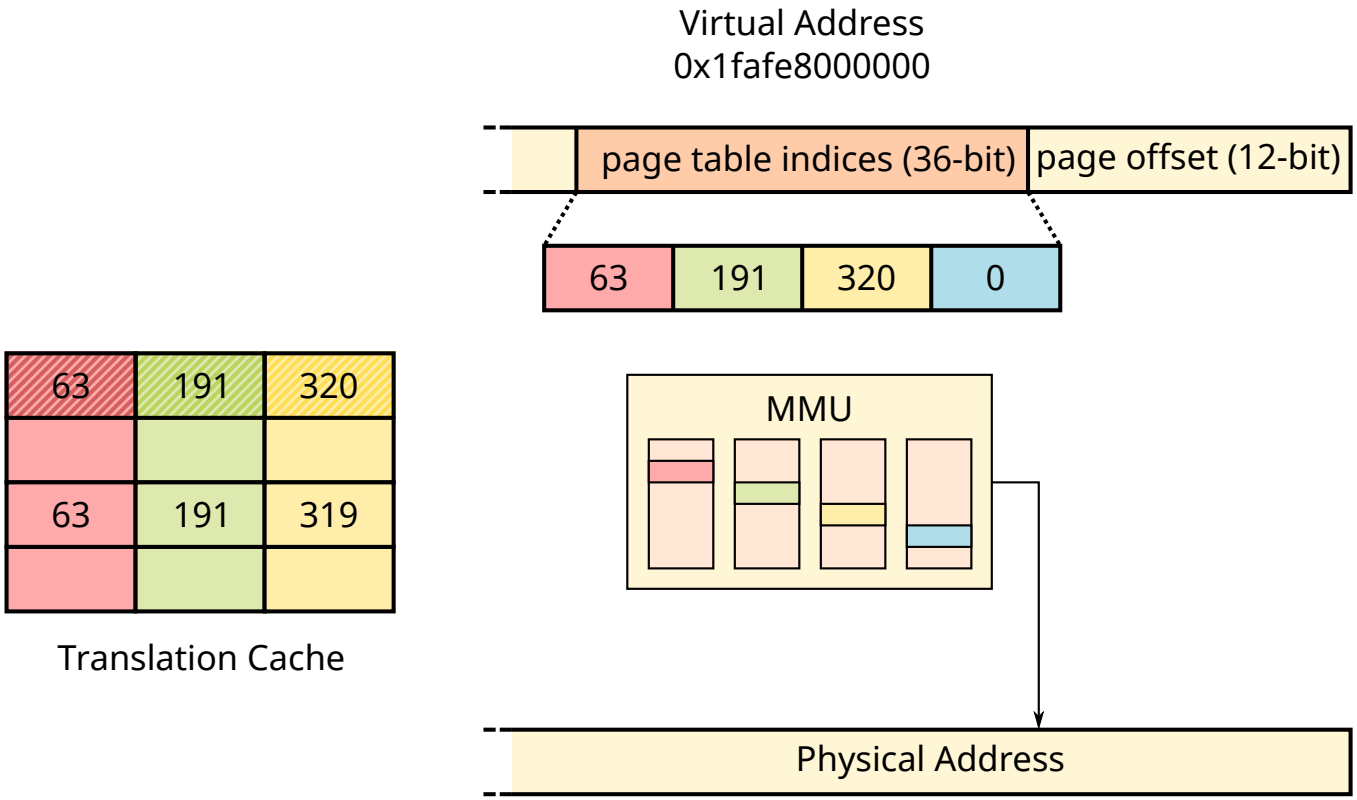
Perform n page table walks

Translation Caches



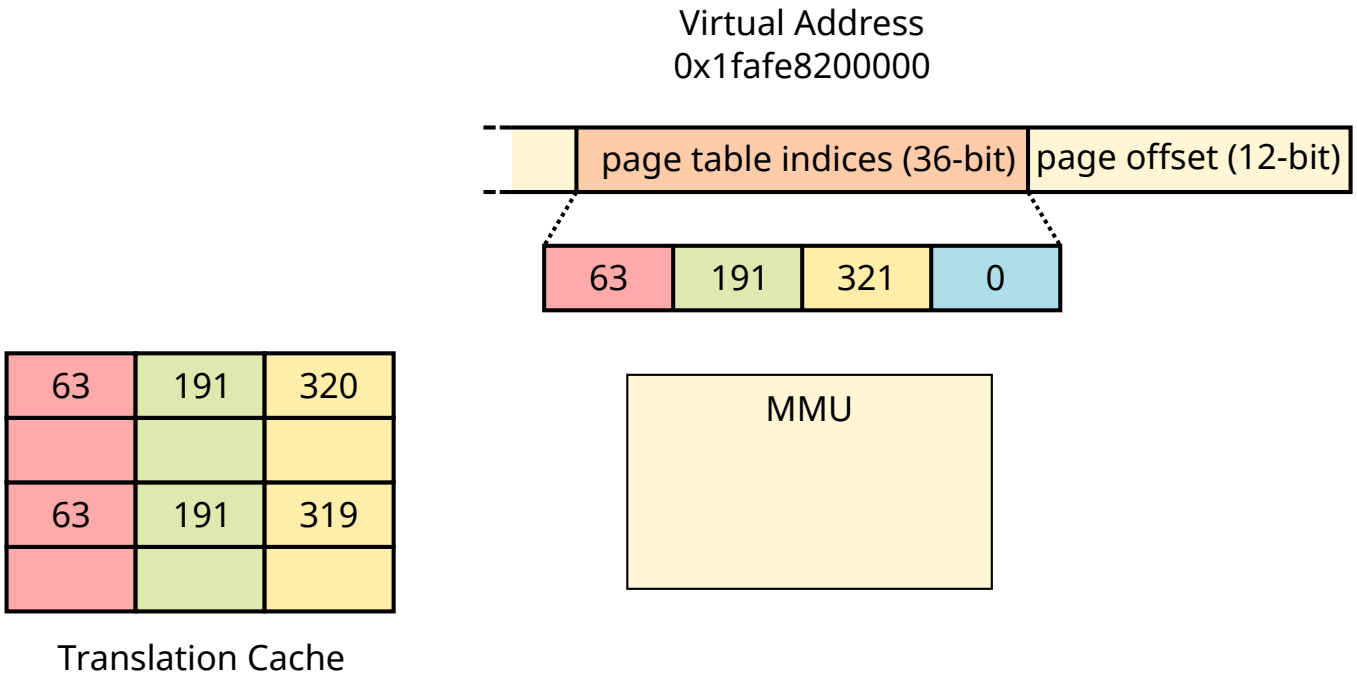
Perform n page table walks

Translation Caches



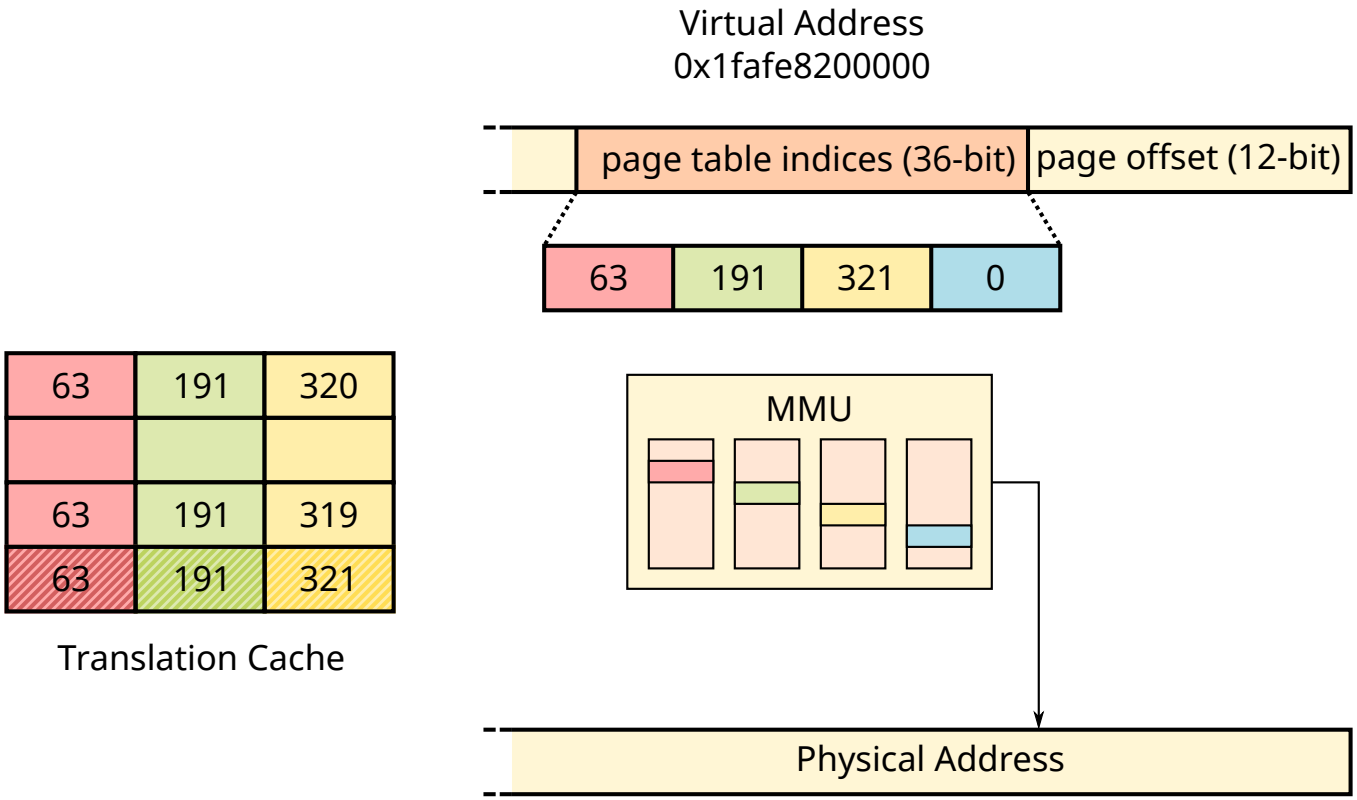
Perform n page table walks

Translation Caches



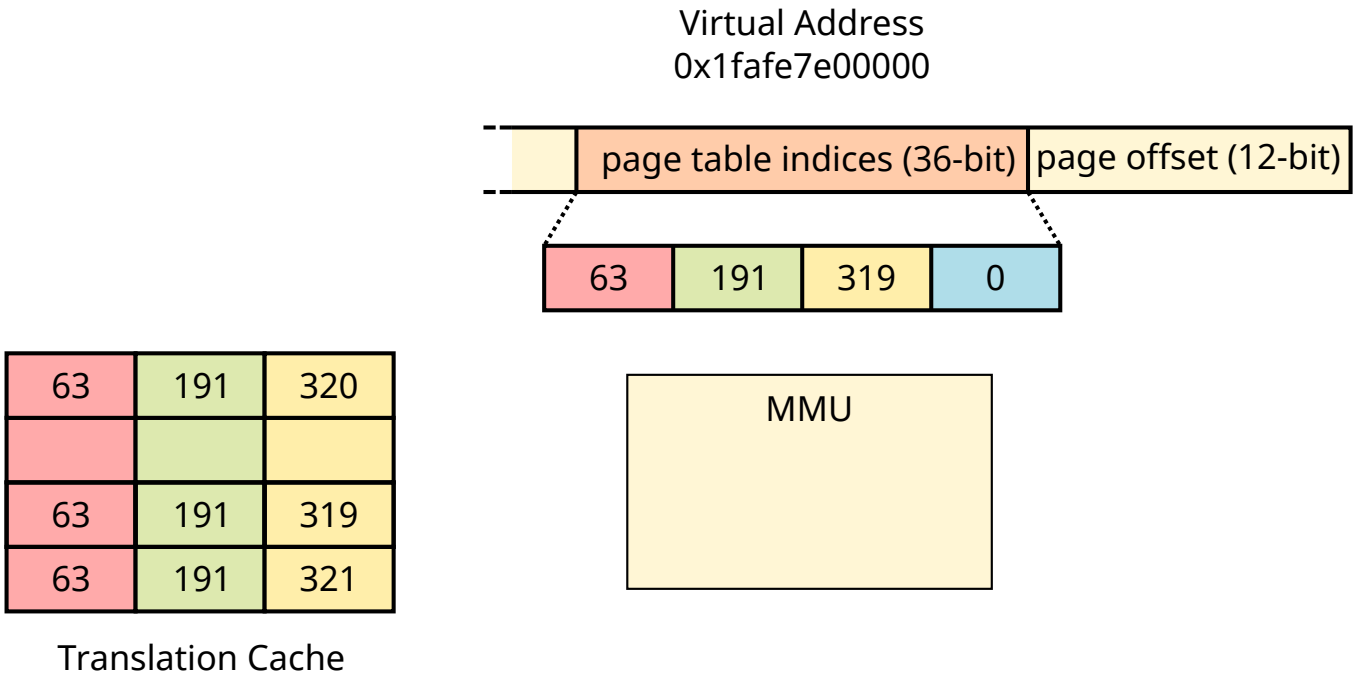
Perform n page table walks

Translation Caches



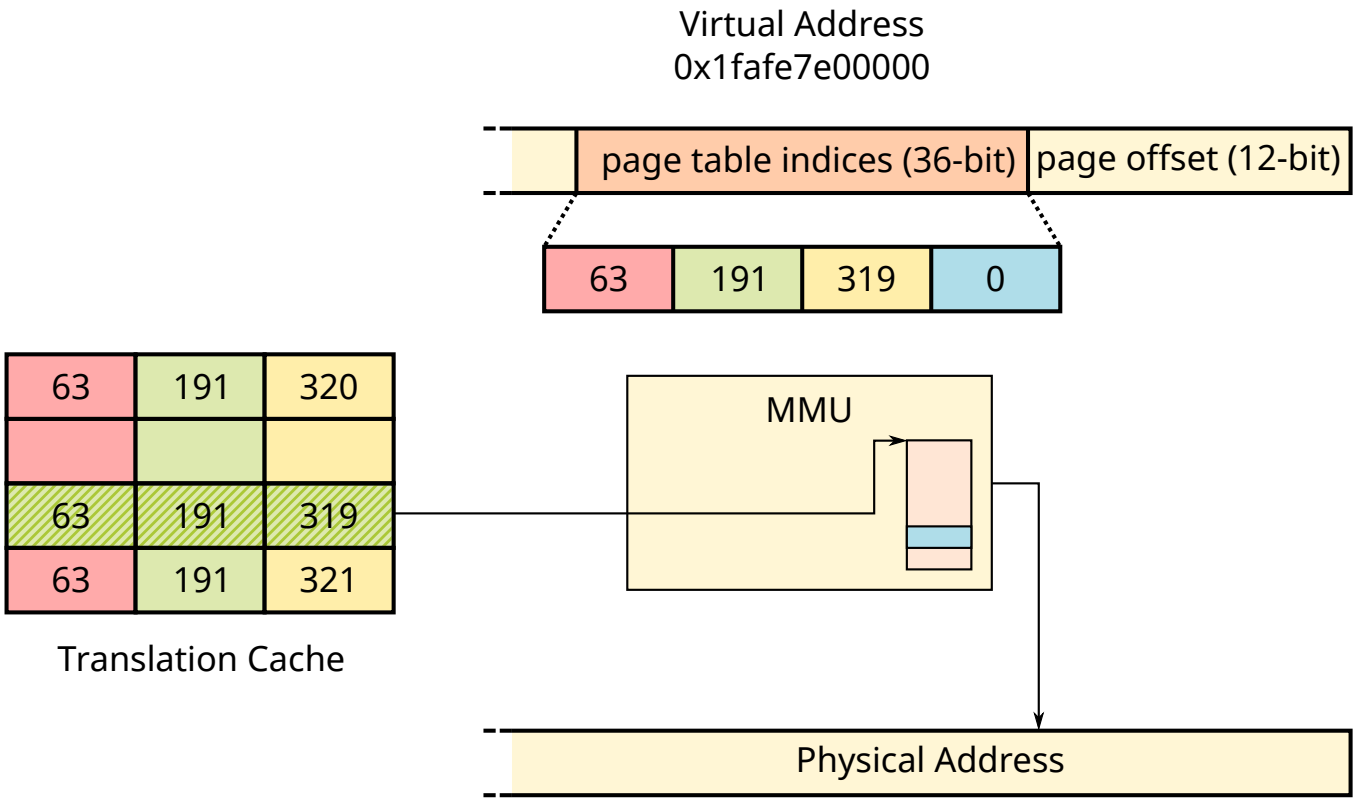
Perform n page table walks

Translation Caches



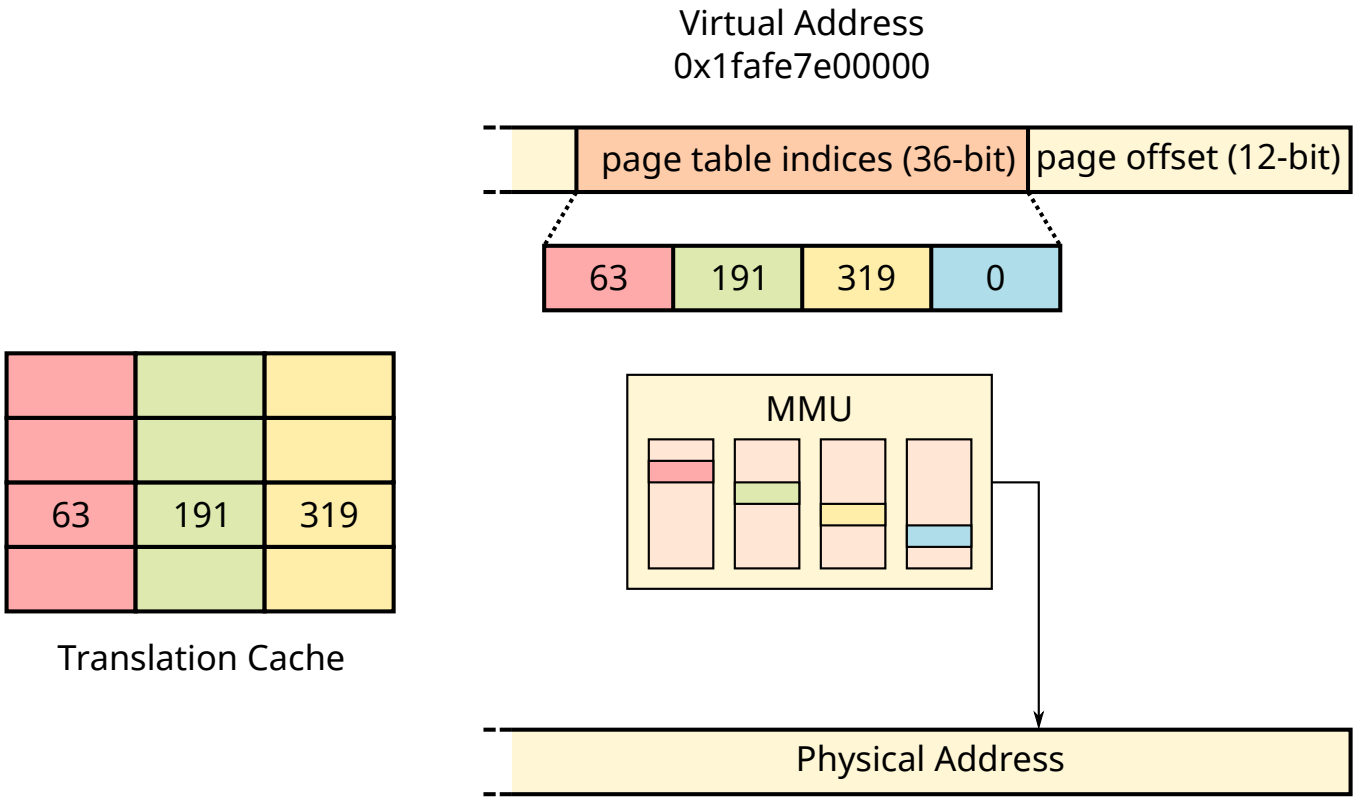
Reload the target

Translation Caches



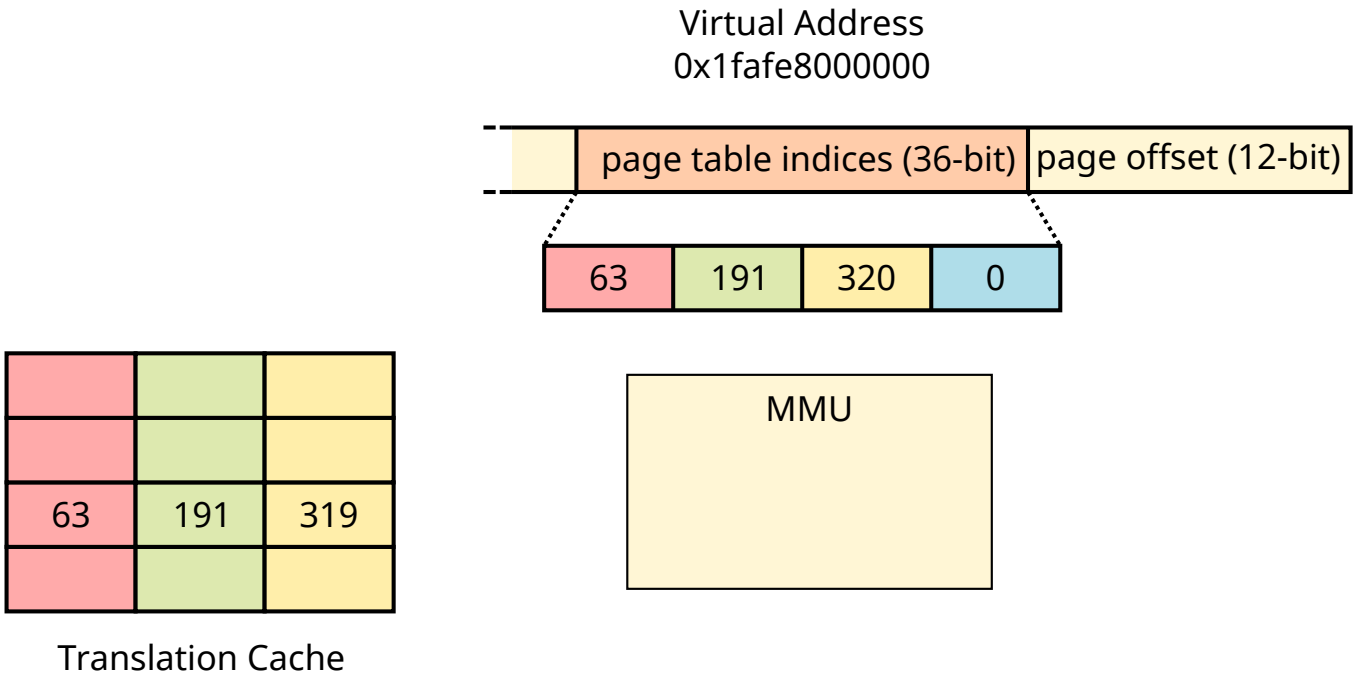
The page table entry is still cached

Translation Caches



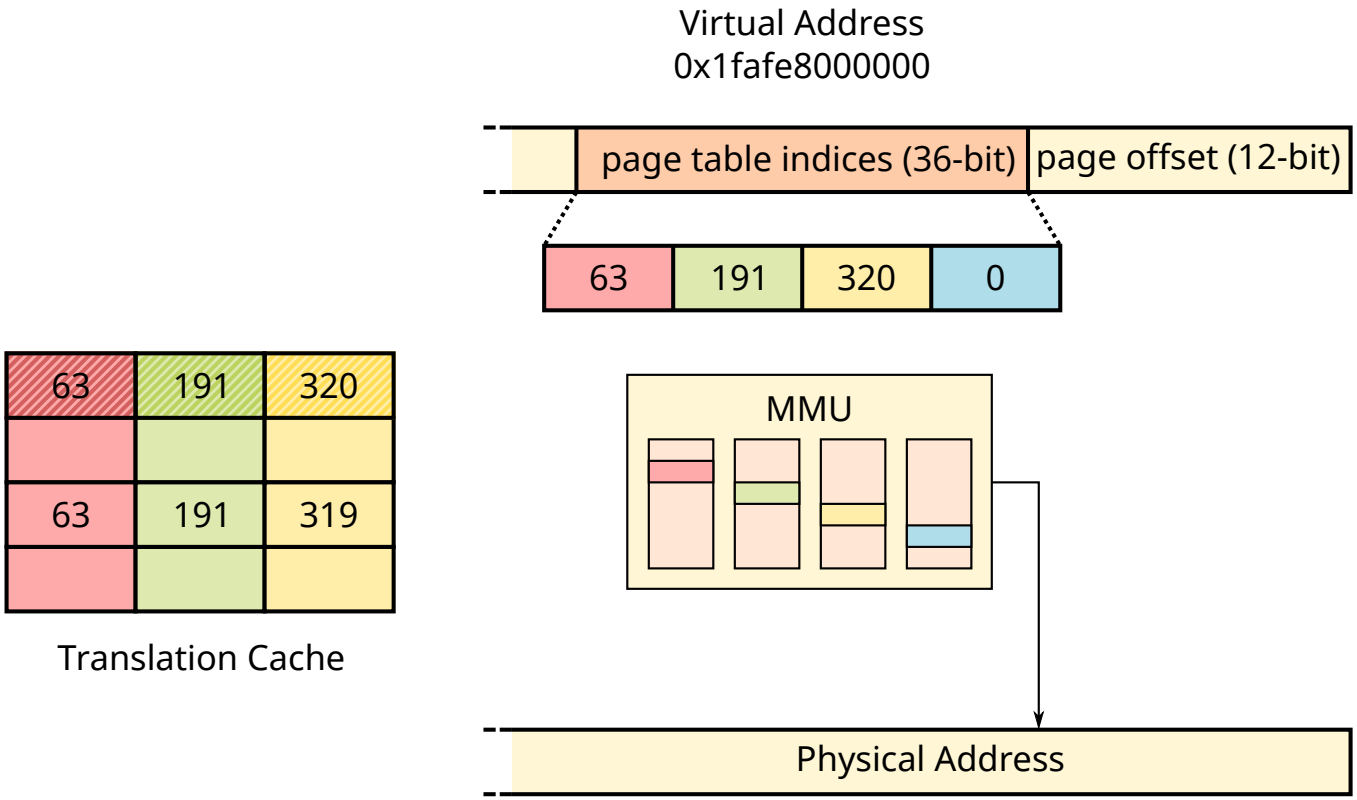
Perform n page table walks

Translation Caches



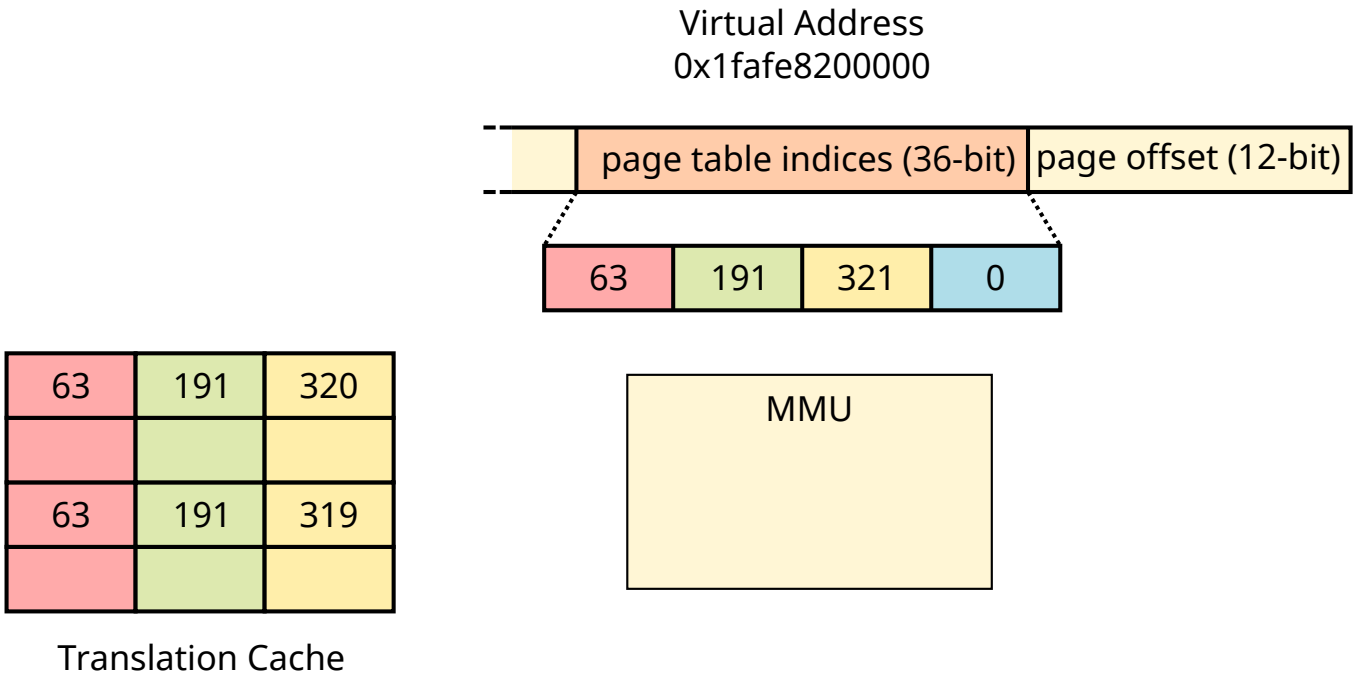
Perform n page table walks

Translation Caches



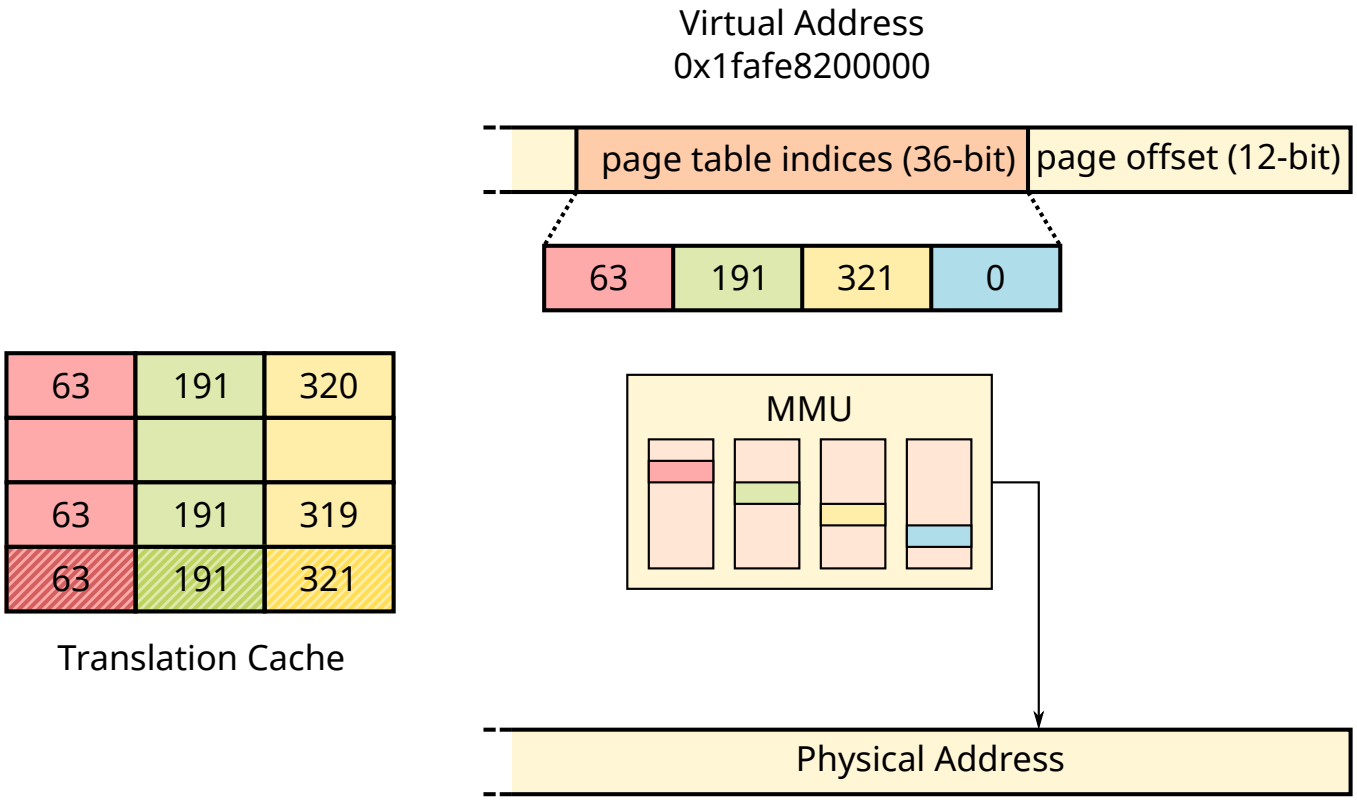
Perform n page table walks

Translation Caches



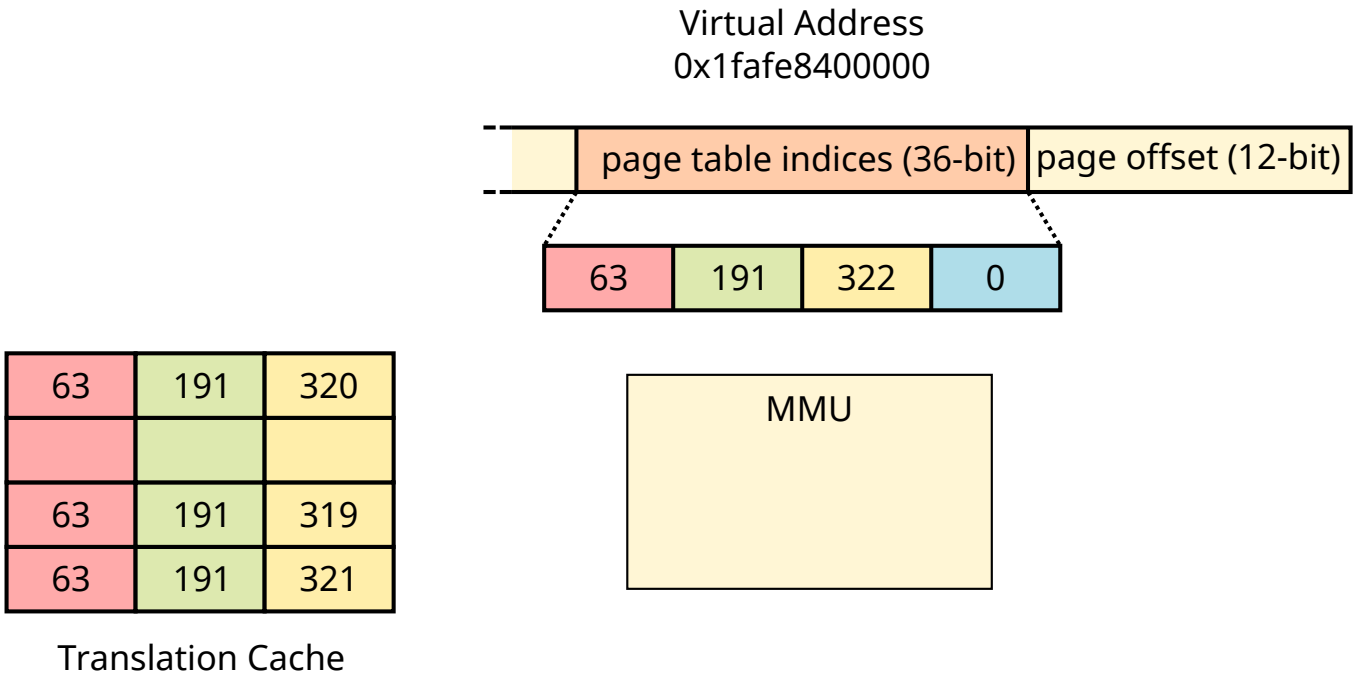
Perform n page table walks

Translation Caches



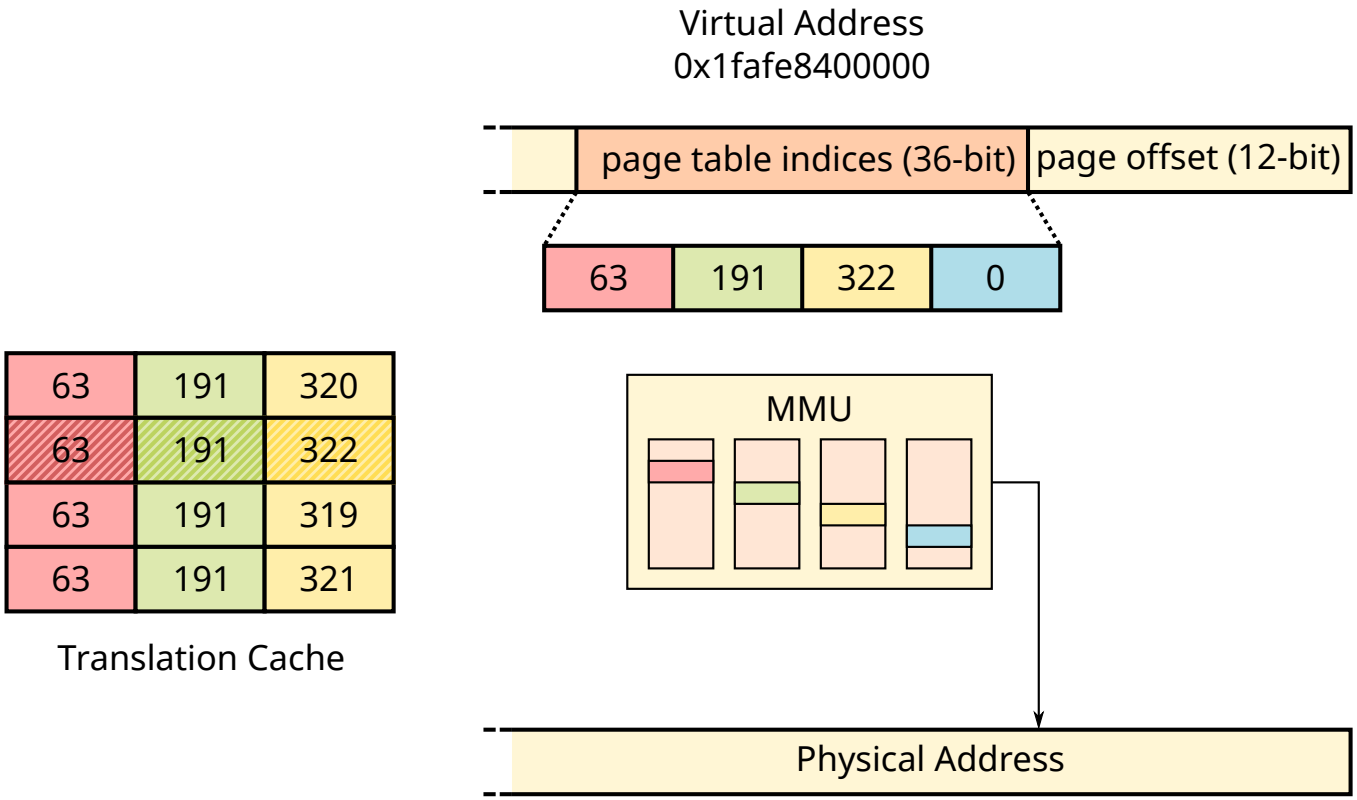
Perform n page table walks

Translation Caches



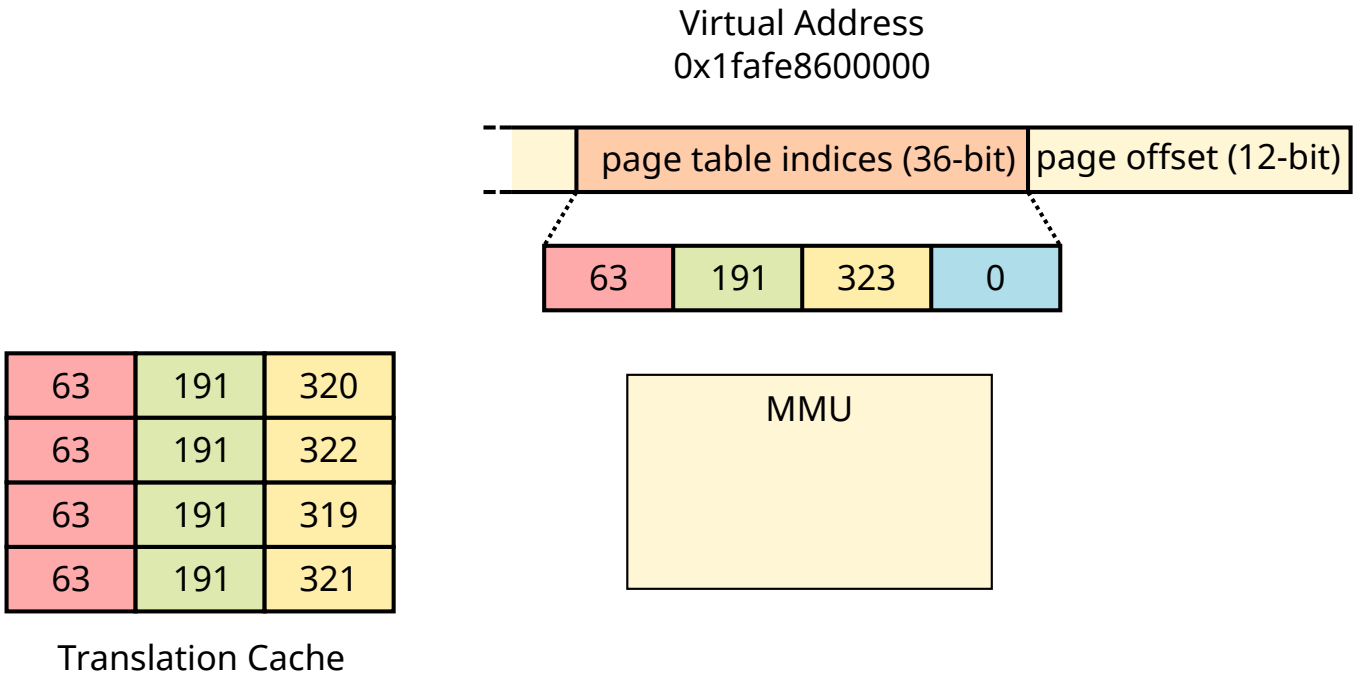
Perform n page table walks

Translation Caches



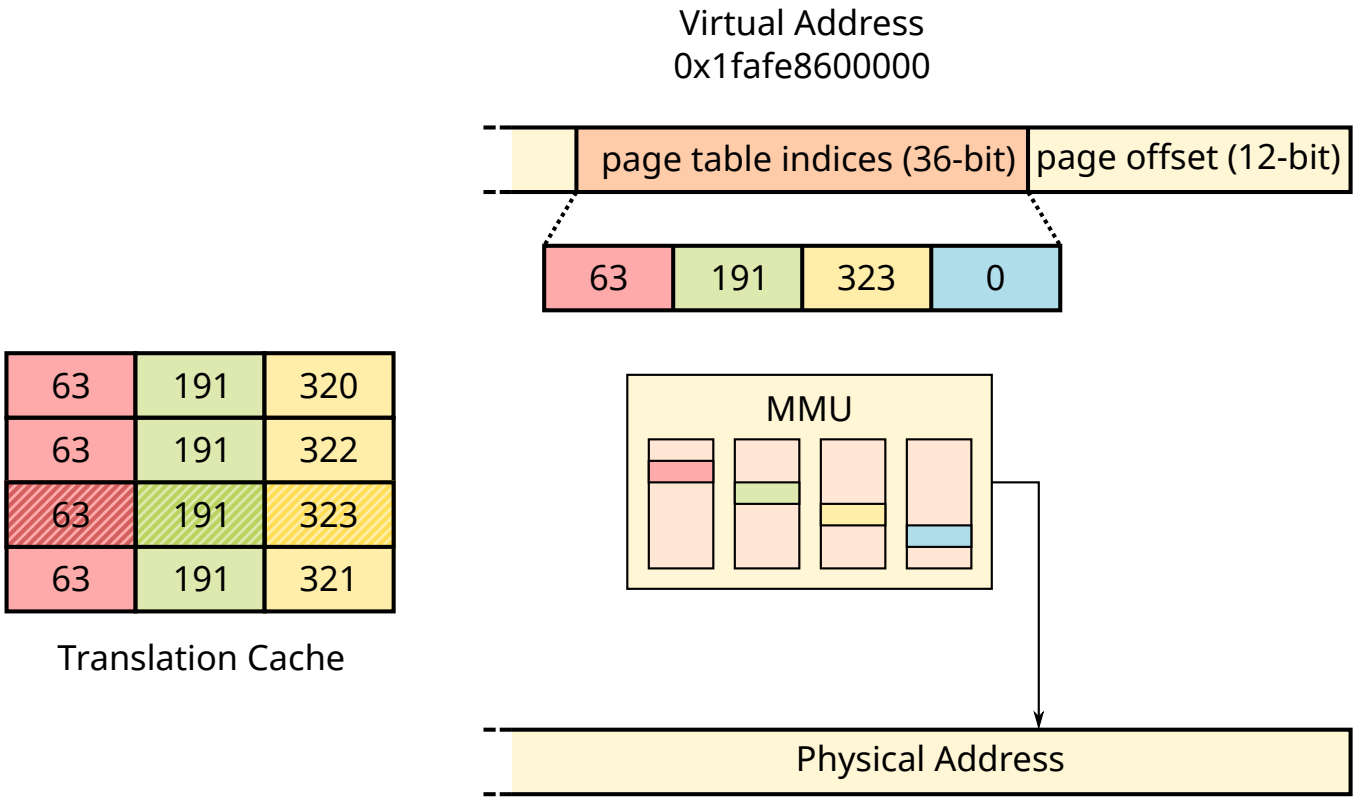
Perform n page table walks

Translation Caches



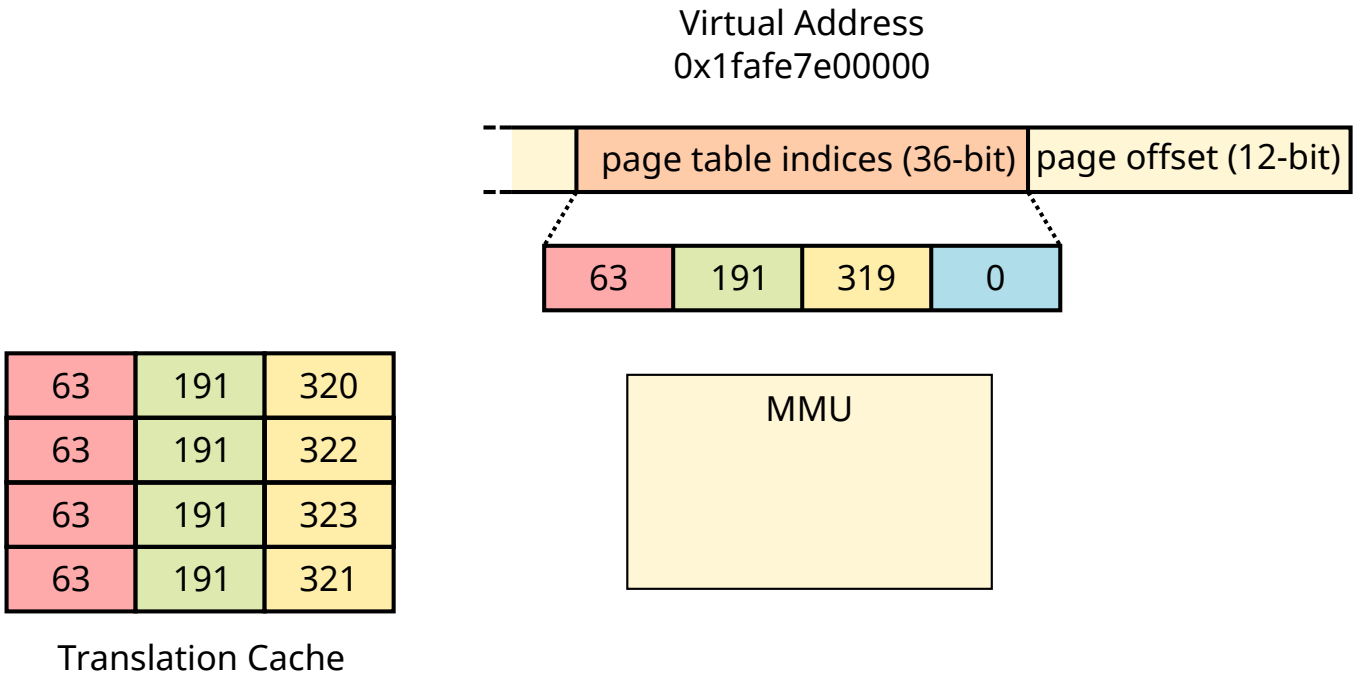
Perform n page table walks

Translation Caches



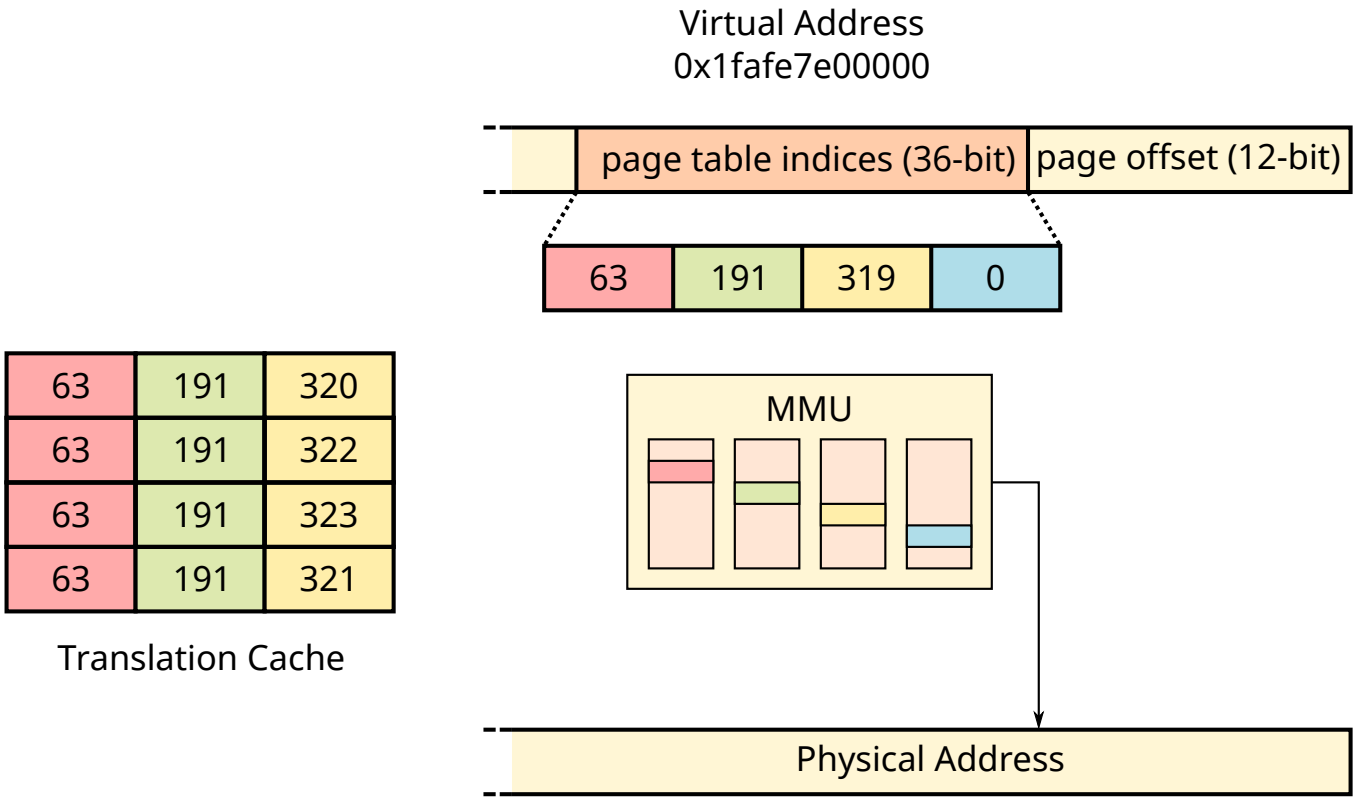
Perform n page table walks

Translation Caches



Reload the target

Translation Caches



Perform full page table walk

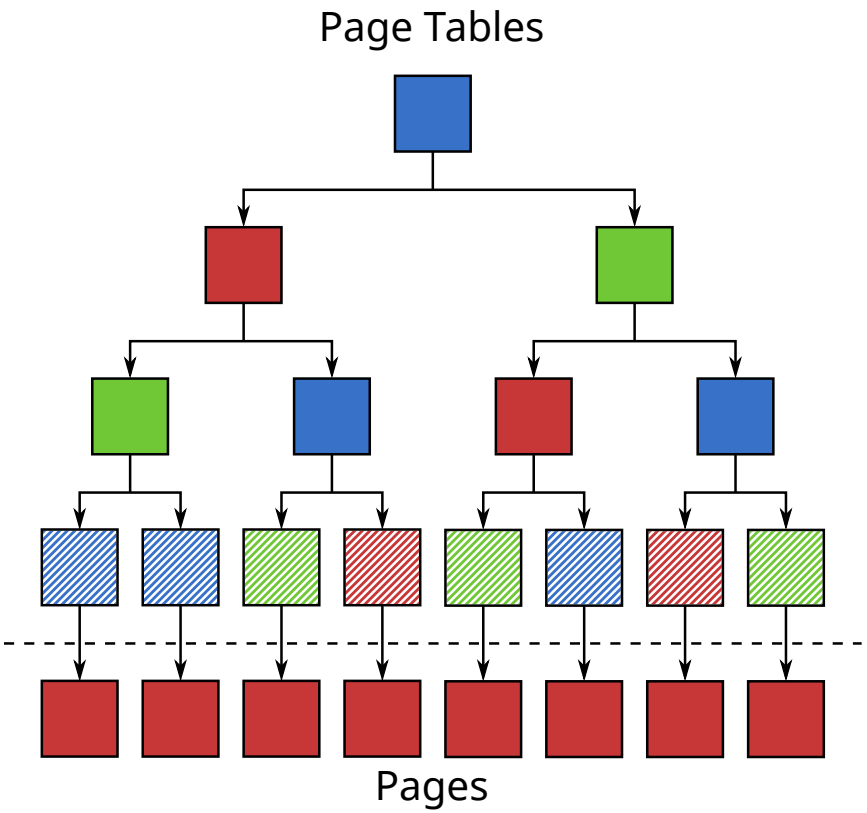
Translation Caches

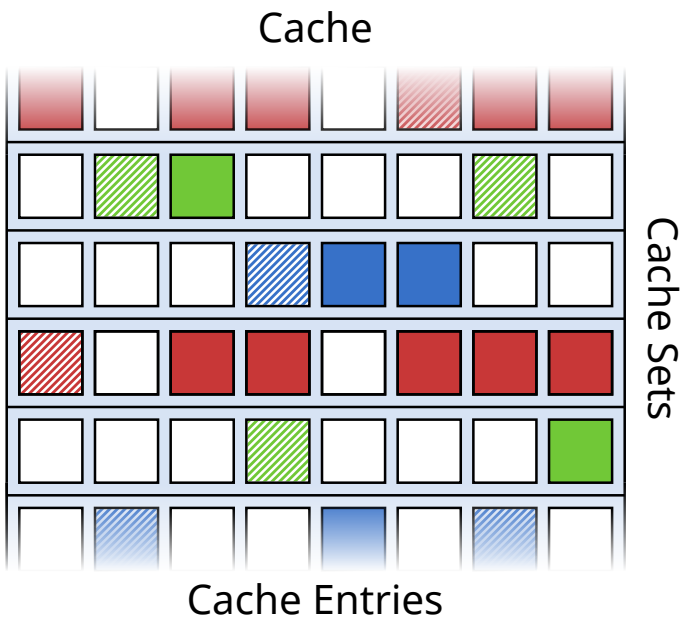
CPU	Year	Caches			TLBs			Translation Caches			Time
		L1d	L2	L3	4K pages	2M pages	1G pages	PML2E	PML3E	PML4E	
Intel Core i7-7500U (Kaby Lake) @ 2.70GHz	2016	32K	256K	4M	1600	32	20	24-32	3-4	0	5m49s
Intel Core m3-6Y30 (Skylake) @ 0.90GHz	2015	32K	256K	4M	1600	32	20	24	3-4	0	6m01s
Intel Xeon E3-1240 v5 (Skylake) @ 3.50GHz	2015	32K	256K	8M	1600	32	20	24	3-4	0	3m08s
Intel Core i7-6700K (Skylake) @ 4.00GHz	2015	32K	256K	8M	1600	32	20	24	3-4	0	3m41s
Intel Celeron N2840 (Silvermont) @ 2.16GHz	2014	24K	1M	N/A	128	16	N/A	12-16	0	0	52s
Intel Core i7-4500U (Haswell) @ 1.80GHz	2013	32K	256K	4M	1088	32	4	24	3-4	0	2m53
Intel Core i7-3632QM (Ivy Bridge) @ 2.20GHz	2012	32K	256K	6M	576	32	4	24-32	3	0	3m05s
Intel Core i7-2620QM (Sandy Bridge) @ 2.00GHz	2011	32K	256K	6M	576	32	4	24	2-4	0	3m11s
Intel Core i5 M480 (Westmere) @ 2.67GHz	2010	32K	256K	3M	576	32	N/A	24-32	2-6	0	2m44s
Intel Core i7 920 (Nehalem) @ 2.67GHz	2008	32K	256K	8M	576	32	N/A	24-32	3	0	4m26s
AMD Ryzen 7 1700 8-Core (Zen) @ 3.3GHz	2017	32K	512K	16M	1600	1600	64	0	64	0	13m16s
AMD Ryzen 5 1600X 6-Core (Zen) @ 3.6GHz	2017	32K	512K	16M	1600	1600	64	0	64	16	30m50s
AMD FX-8350 8-Core (Piledriver) @ 4.0GHz	2012	64K	2M	8M	1088	1088	1088	0	0	0	2m50s
AMD FX-8320 8-Core (Piledriver) @ 3.5GHz	2012	64K	2M	8M	1088	1088	1088	0	0	0	2m47s
AMD FX-8120 8-Core (Bulldozer) @ 3.4GHz	2011	16K	2M	8M	1056	1056	1056	0	0	0	2m33s
AMD Athlon II 640 X4 (K10) @ 3.0GHz	2010	64K	512K	N/A	560	176	N/A	24	0	0	7m50s
AMD E-350 (Bobcat) @ 1.6GHz	2010	32K	512K	N/A	552	8-12	N/A	8-12	0	0	5m38s
AMD Phenom 9550 4-Core (K10) @ 2.2GHz	2008	64K	512K	2M	560	176	48	24	0	0	6m52s
Rockchip RK3399 (ARM Cortex A72) @ 2.0GHz	2017	32K	1M	N/A	544	512	N/A	16	6	N/A	17m49s
Rockchip RK3399 (ARM Cortex A53) @ 1.4GHz	2017	32K	512K	N/A	522	512	N/A	64	0	N/A	7m06s
Allwinner A64 (ARM Cortex A53) @ 1.2GHz	2016	32K	512K	N/A	522	512	N/A	64	0	N/A	52m26s
Samsung Exynos 5800 (ARM Cortex A15) @ 2.1GHz	2014	32K	2M	N/A	544	512	N/A	16	0	N/A	13m28s
Nvidia Tegra K1 CD580M-A1 (ARM Cortex A15) @ 2.3GHz	2014	32K	2M	N/A	544	512	N/A	16	0	N/A	24m19s
Nvidia Tegra K1 CD570M-A1 (ARM Cortex A15; LPAE) @ 2.1GHz	2014	32K	2M	N/A	544	512	N/A	16	0	N/A	6m35s
Samsung Exynos 5800 (ARM Cortex A7) @ 1.3GHz	2014	32K	512K	N/A	266	256	N/A	64	0	N/A	17m42s
Samsung Exynos 5250 (ARM Cortex A15) @ 1.7GHz	2012	32K	1M	N/A	544	512	N/A	16	0	N/A	6m46s

Translation Caches

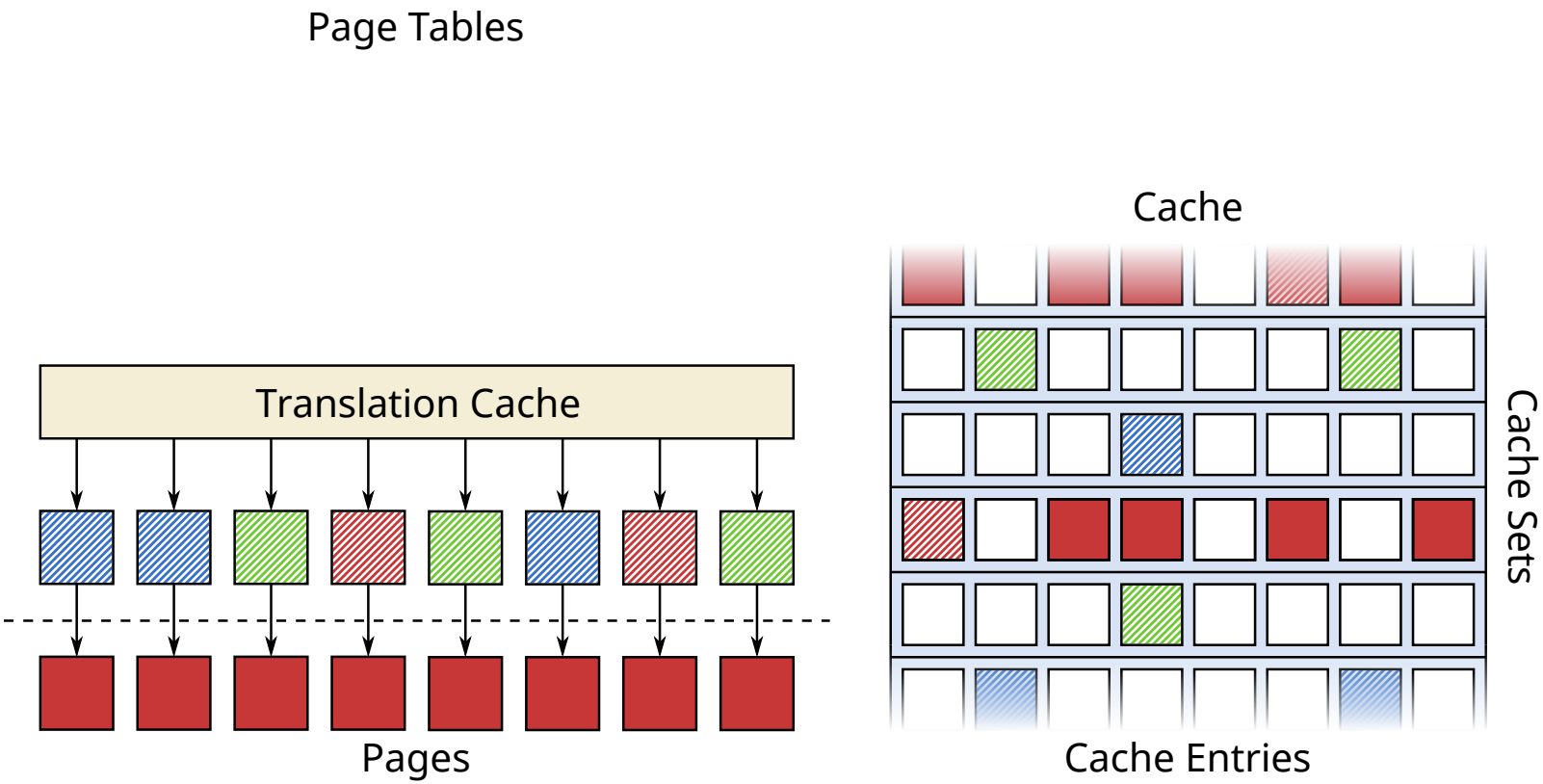
Translation caches are widely available on Intel, AMD and ARM

Translation Caches





Translation Caches

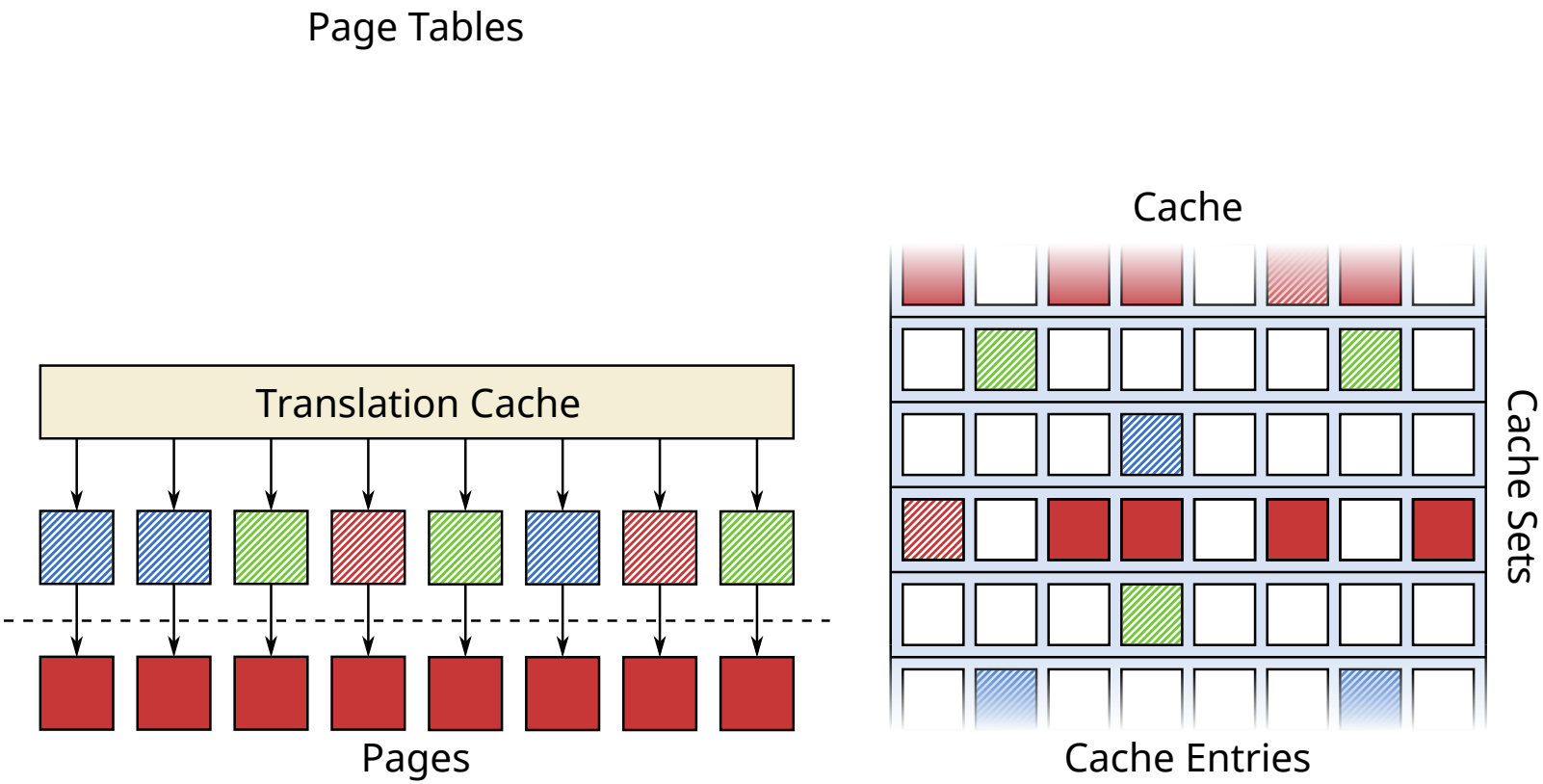


Translation caches skip page table walks

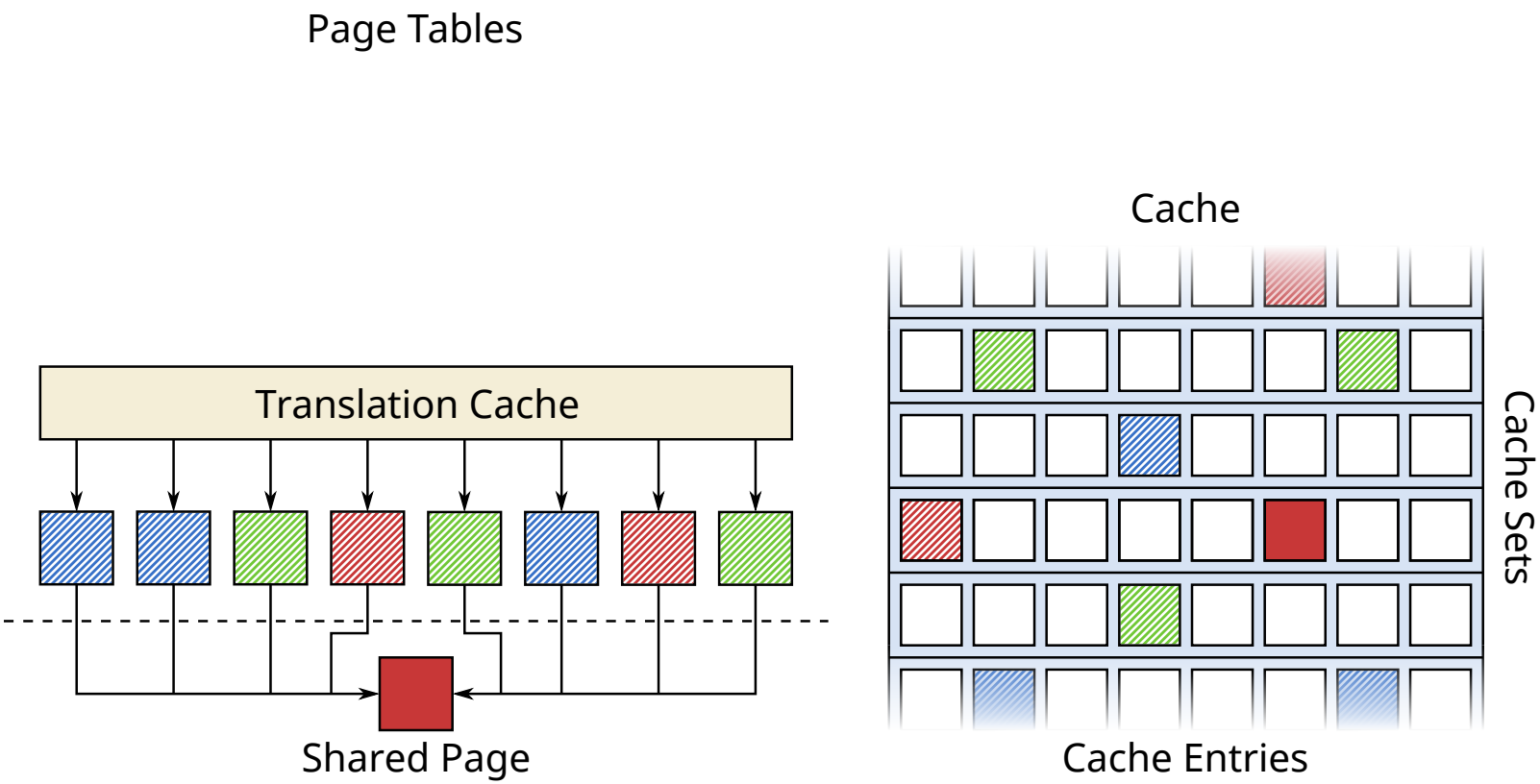
Challenges

- ✓ Avoid noise from high-level page tables
- ✗ Avoid noise from pages
- ✗ Build eviction sets

Shared Memory



Shared Memory



Use shared memory to reduce noise

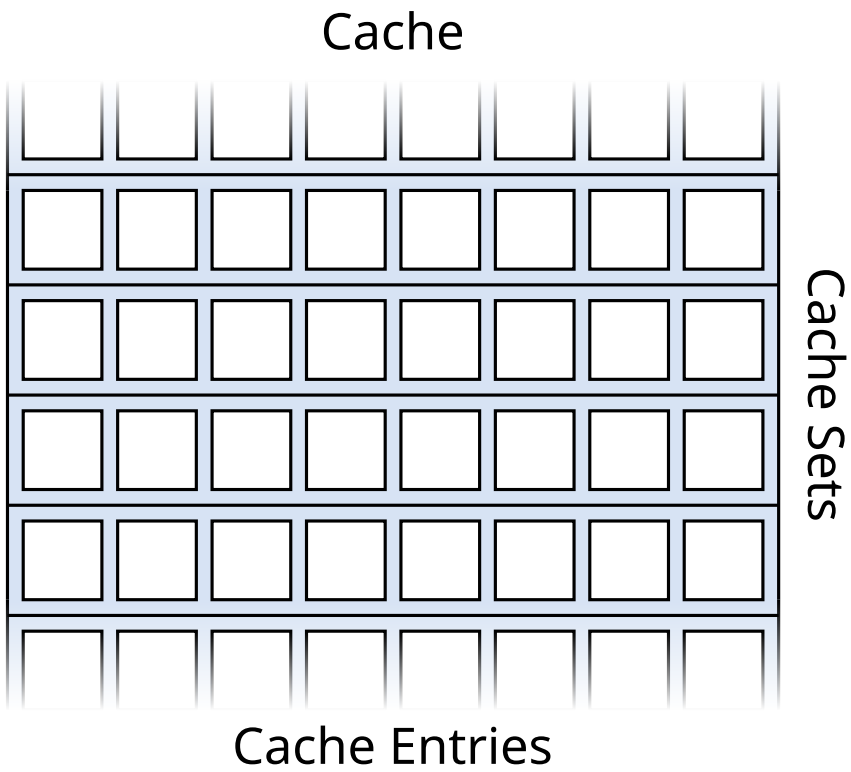
Challenges

- ✓ Avoid noise from high-level page tables
- ✓ Avoid noise from pages
- ✗ Build eviction sets

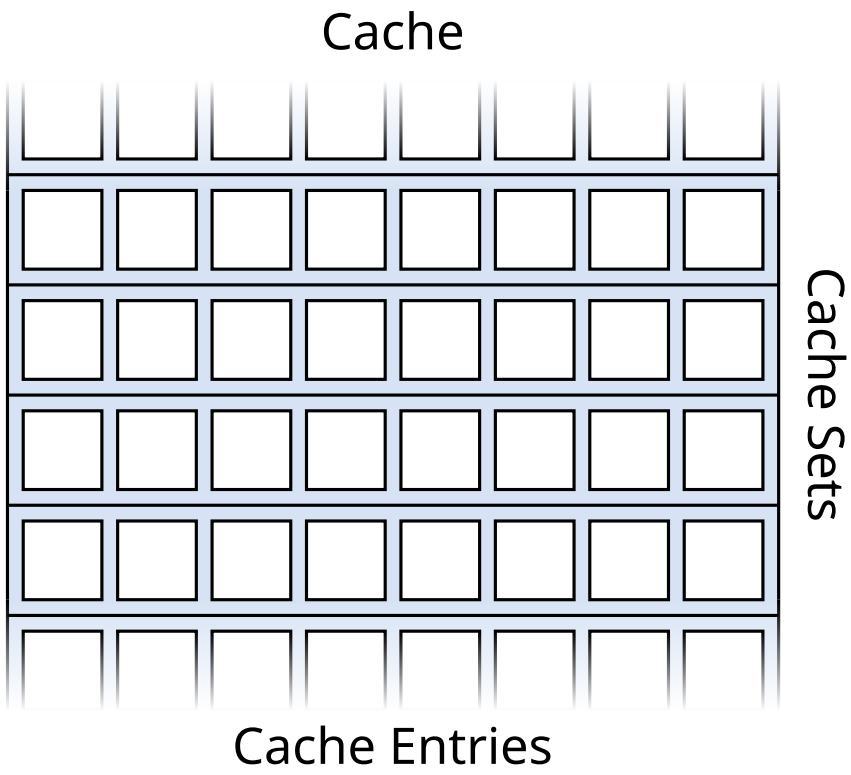
Building Eviction Sets

”The Spy in the Sandbox: Practical Cache Attacks in JavaScript and their Implications” - Oren et al.

Building Eviction Sets

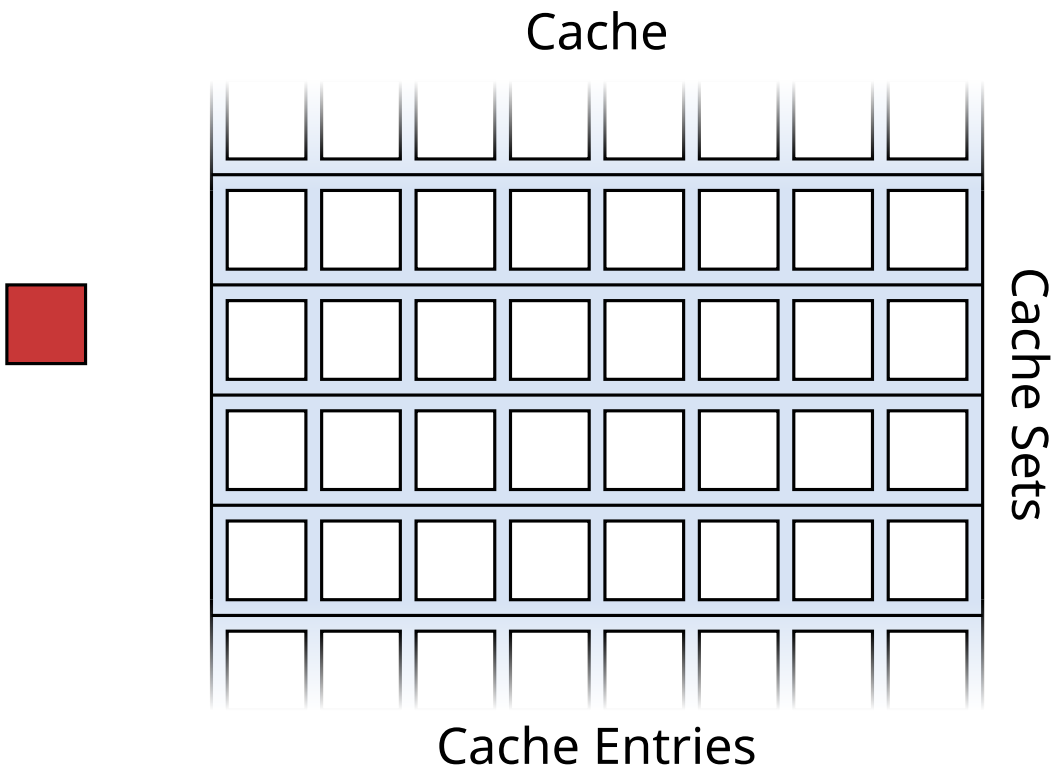


Building Eviction Sets



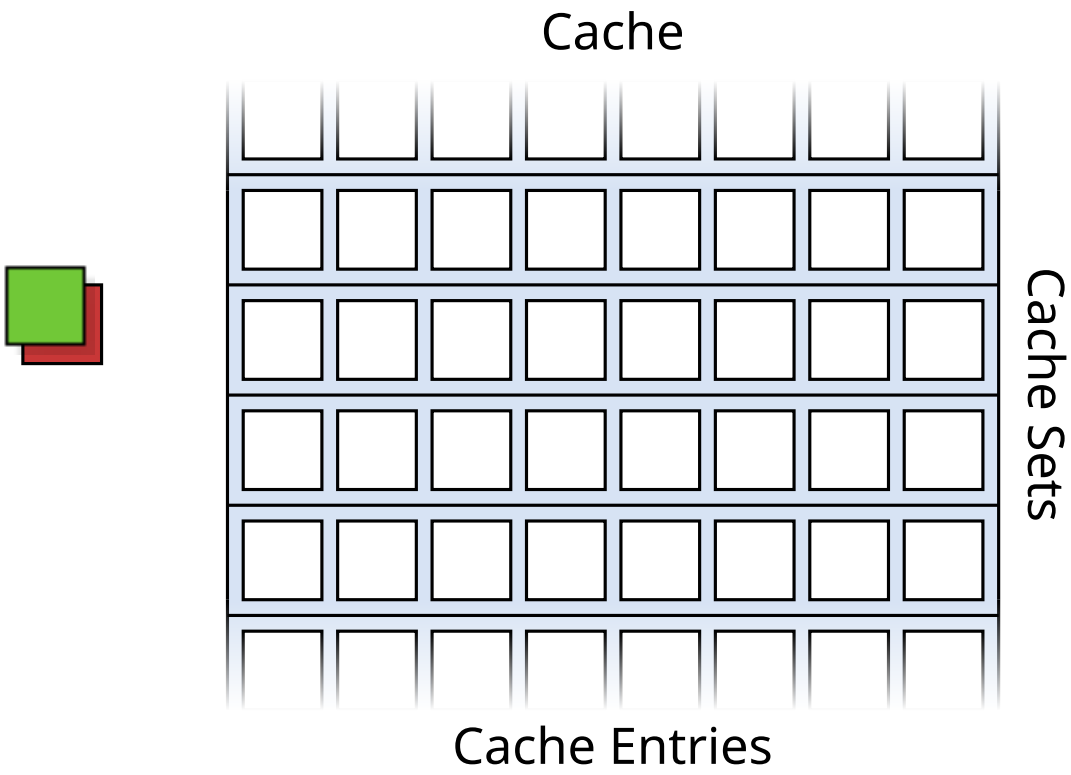
Allocate pages

Building Eviction Sets



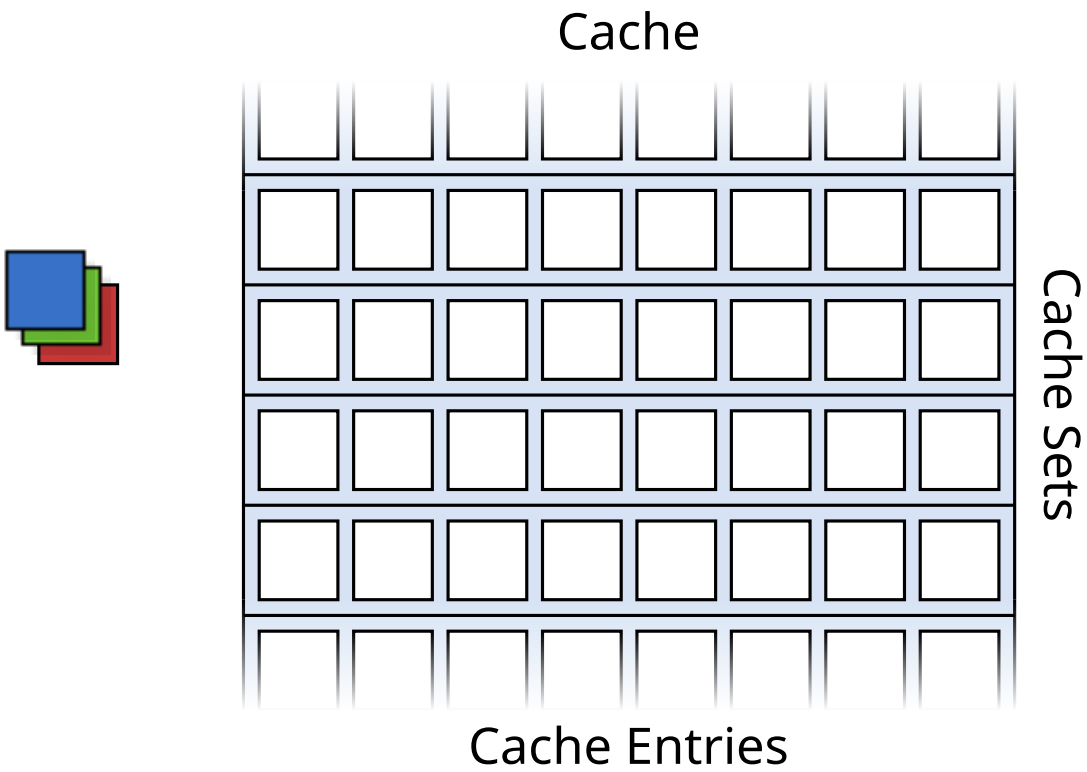
Allocate pages

Building Eviction Sets



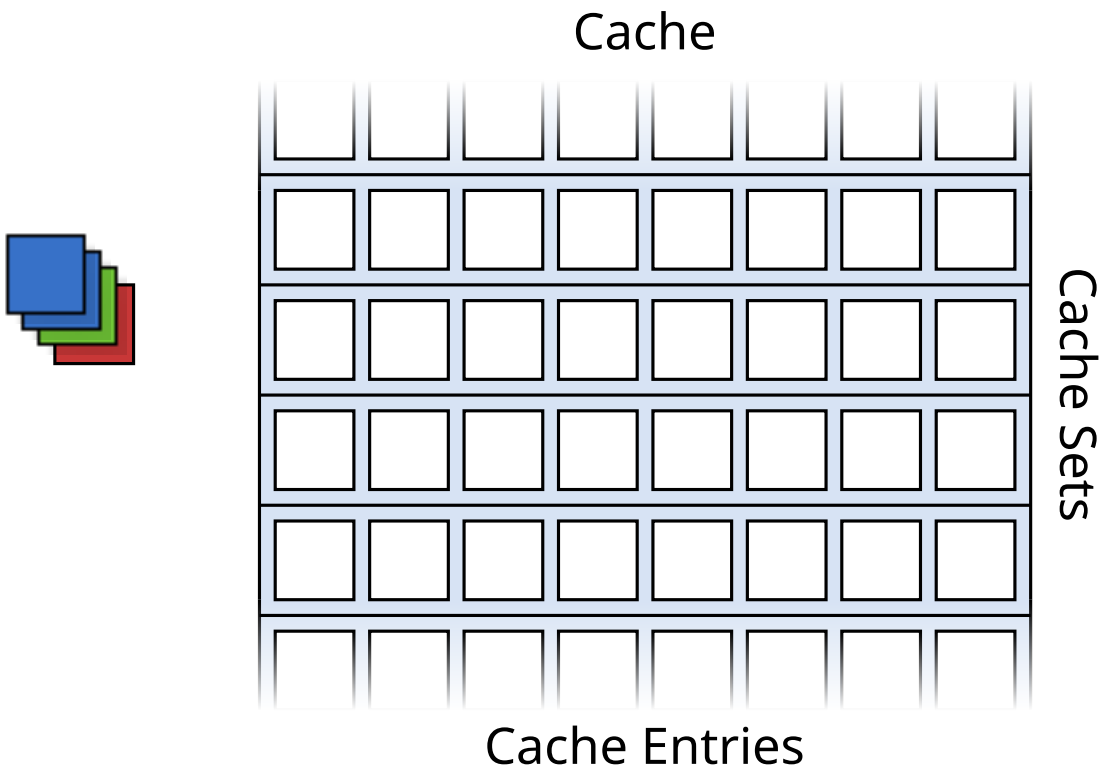
Allocate pages

Building Eviction Sets



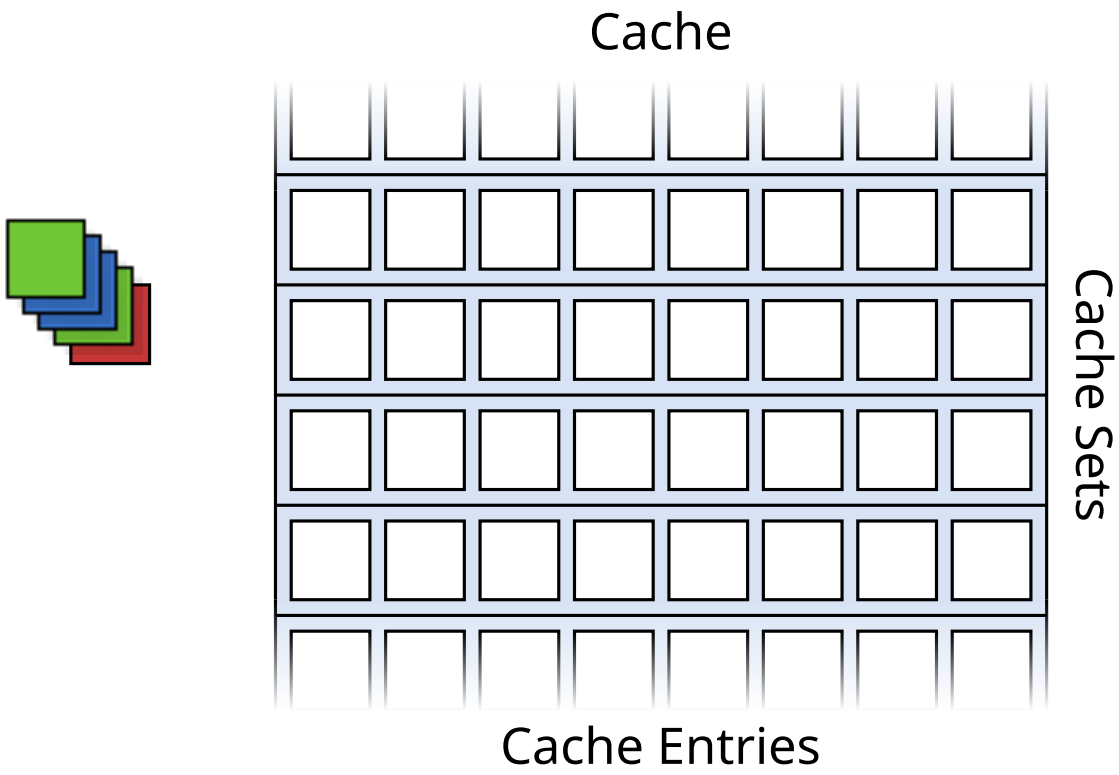
Allocate pages

Building Eviction Sets



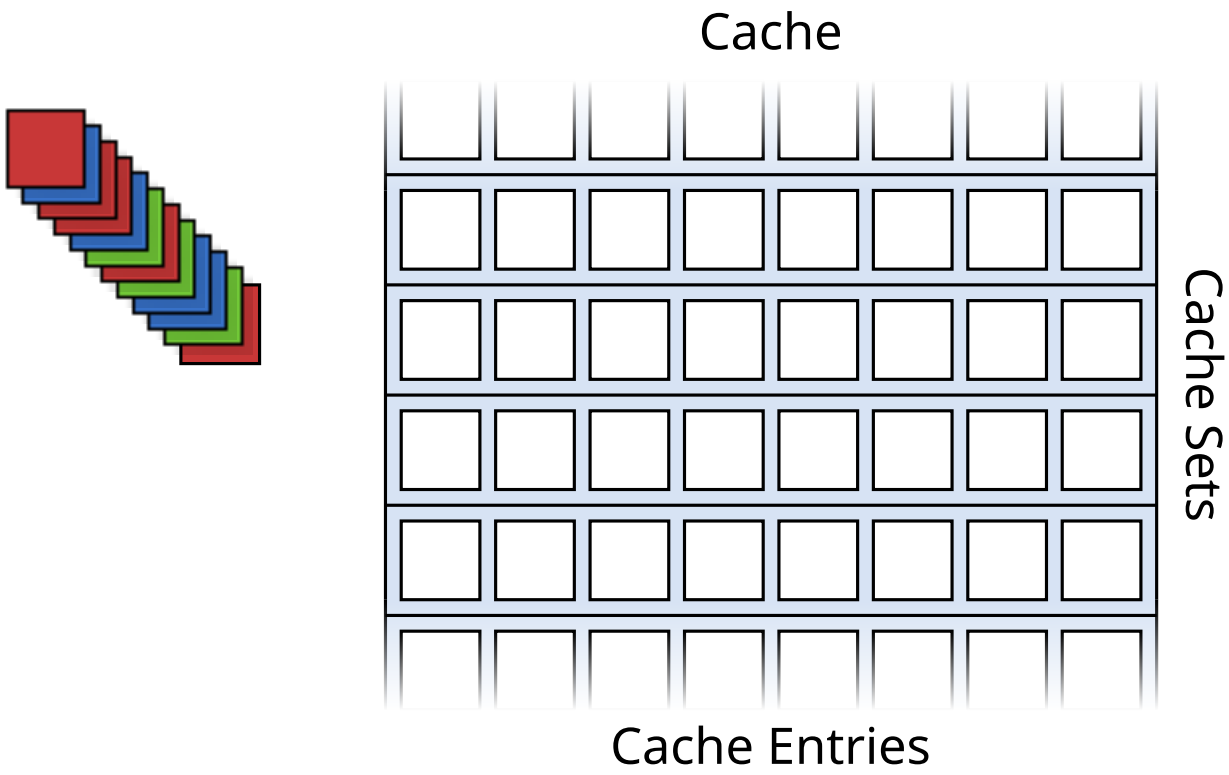
Allocate pages

Building Eviction Sets



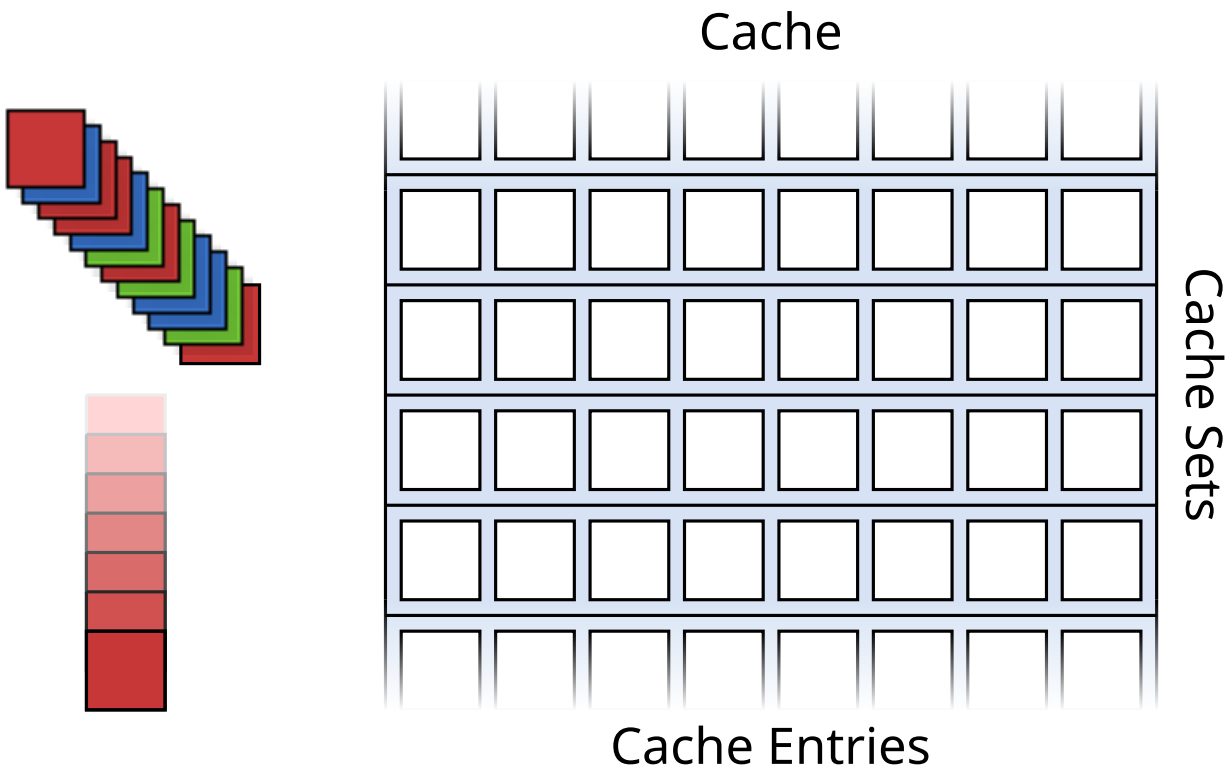
Allocate pages

Building Eviction Sets

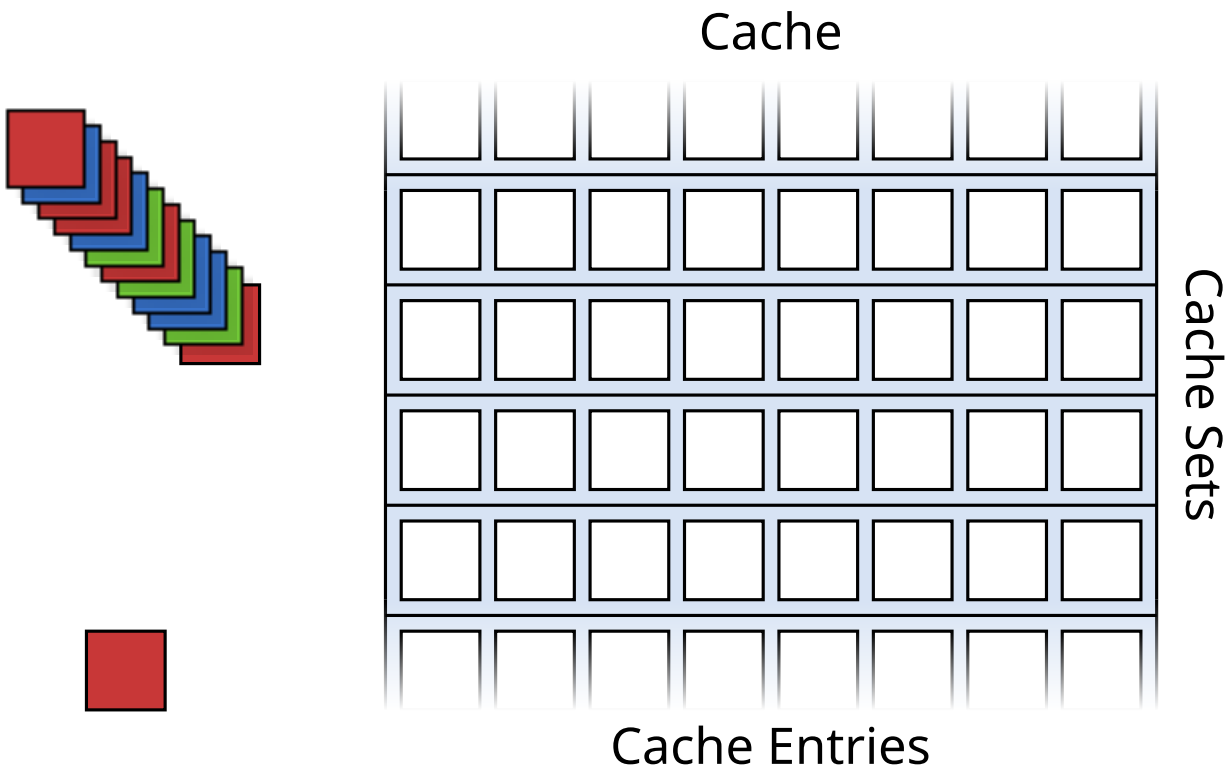


Allocate pages

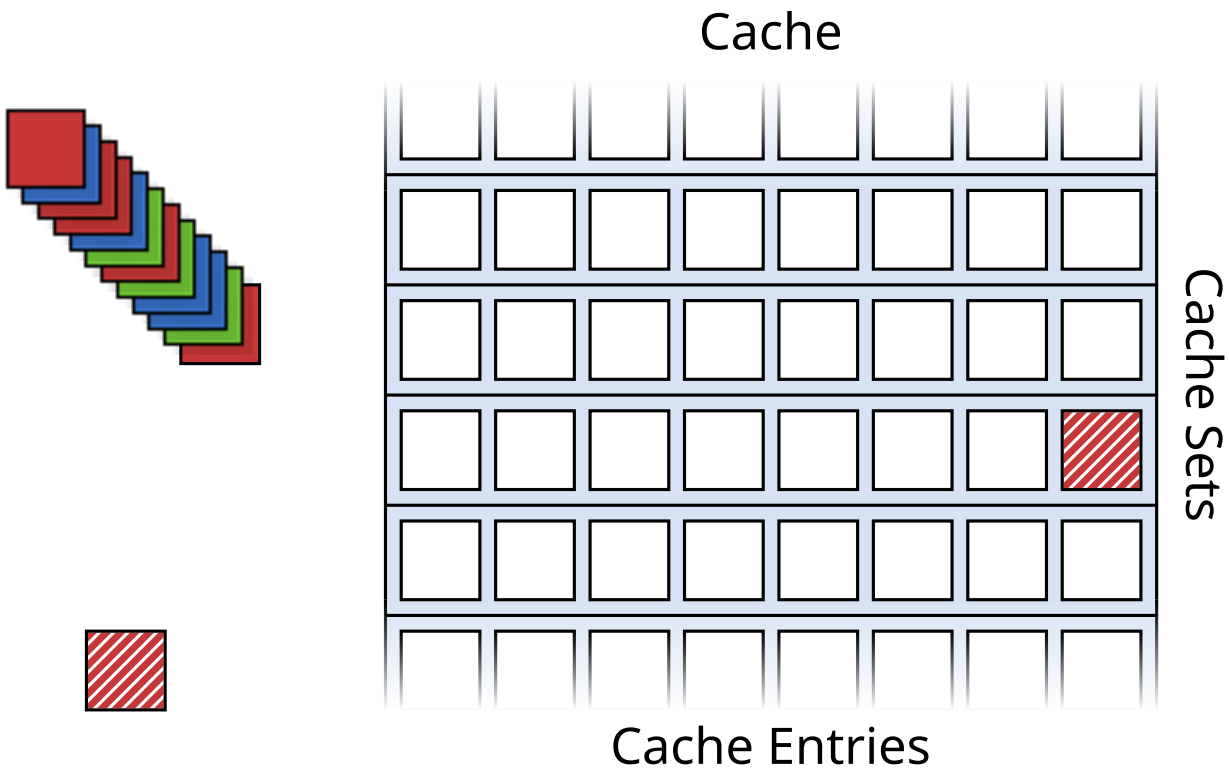
Building Eviction Sets



Building Eviction Sets

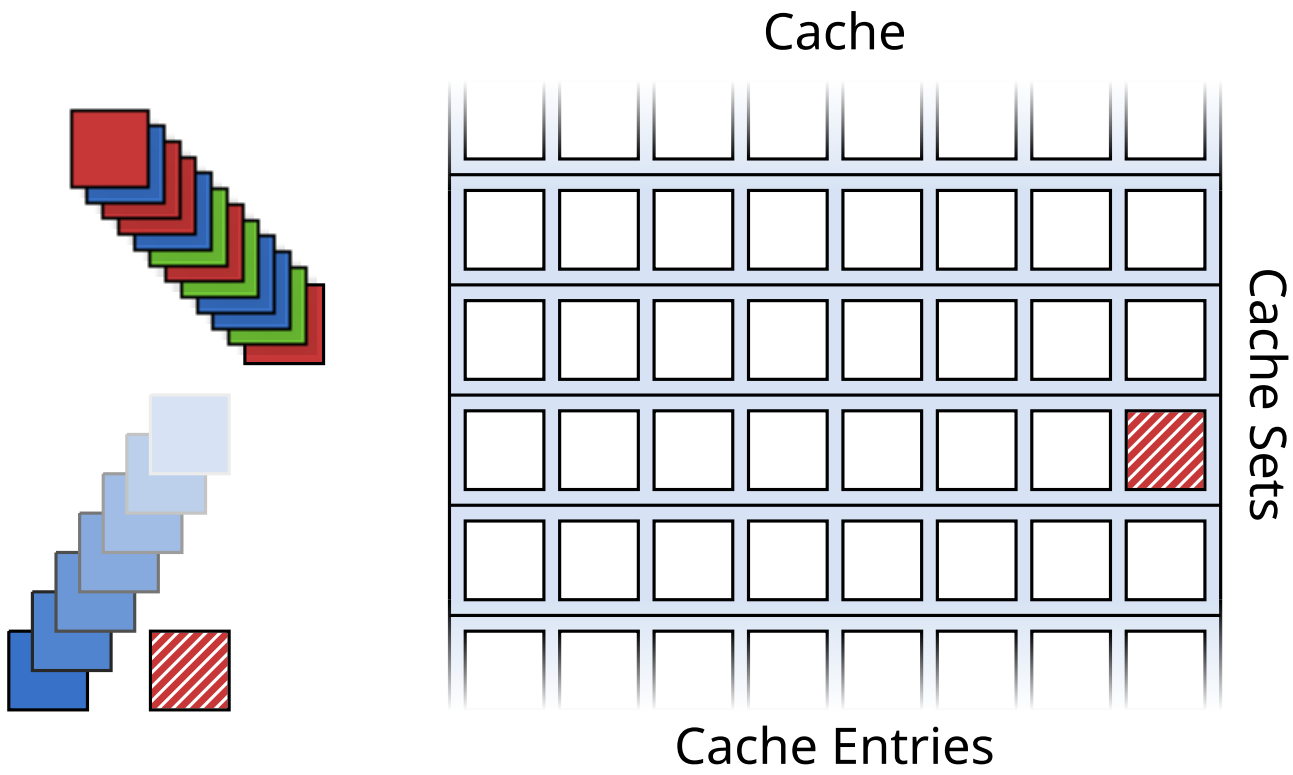


Building Eviction Sets



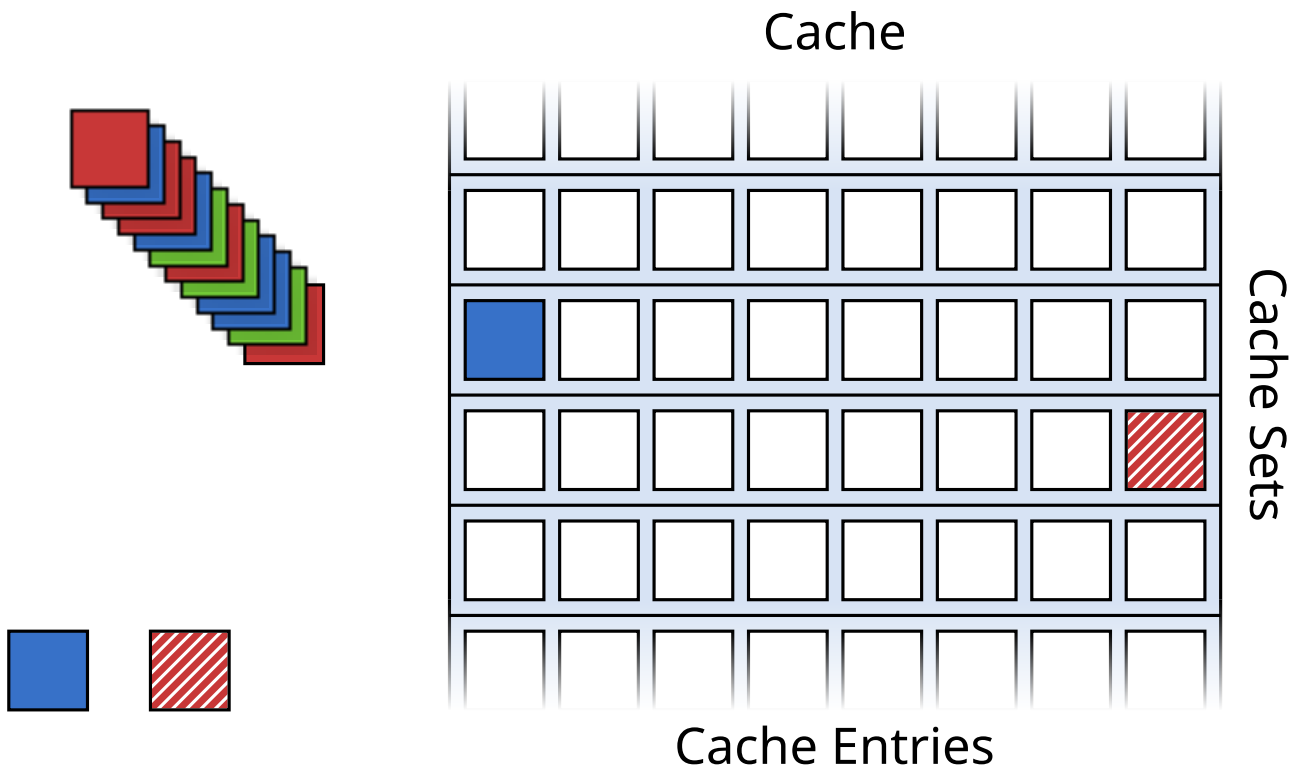
Load target into cache

Building Eviction Sets



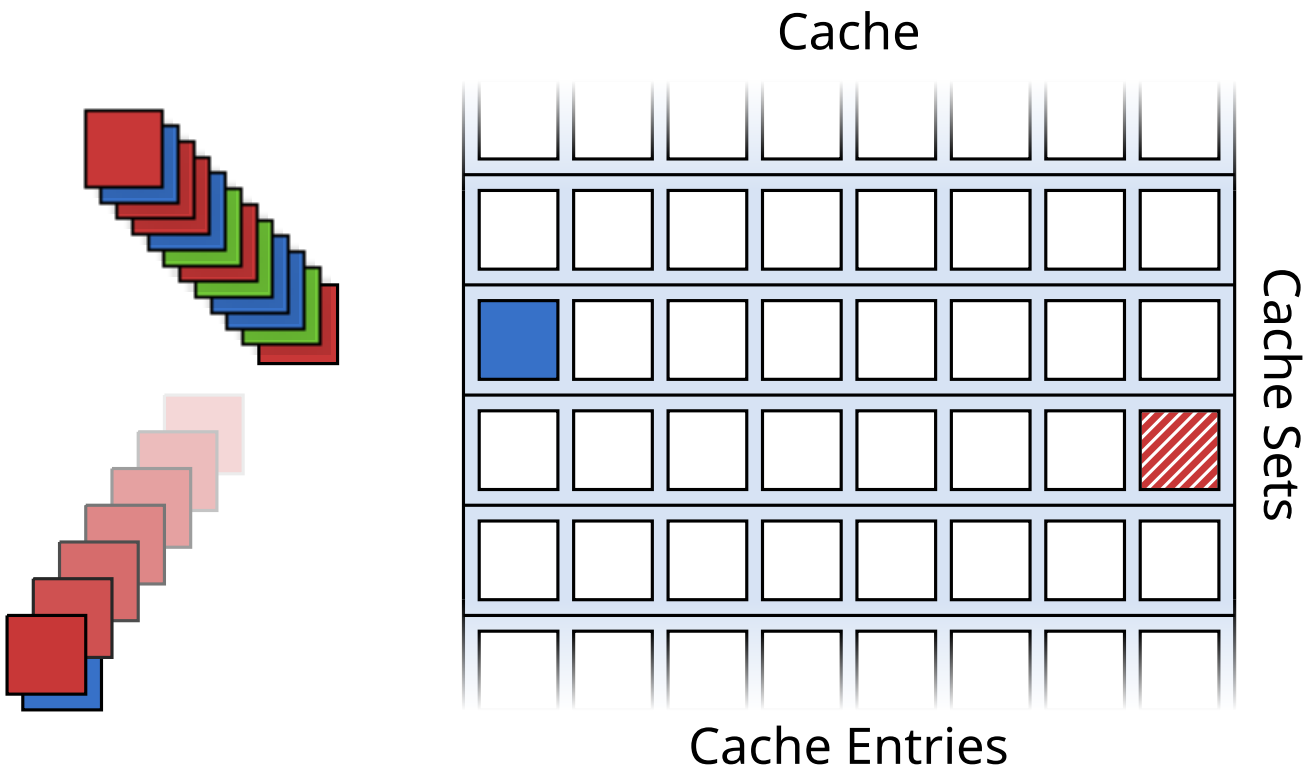
Draw pages and try to evict the target

Building Eviction Sets



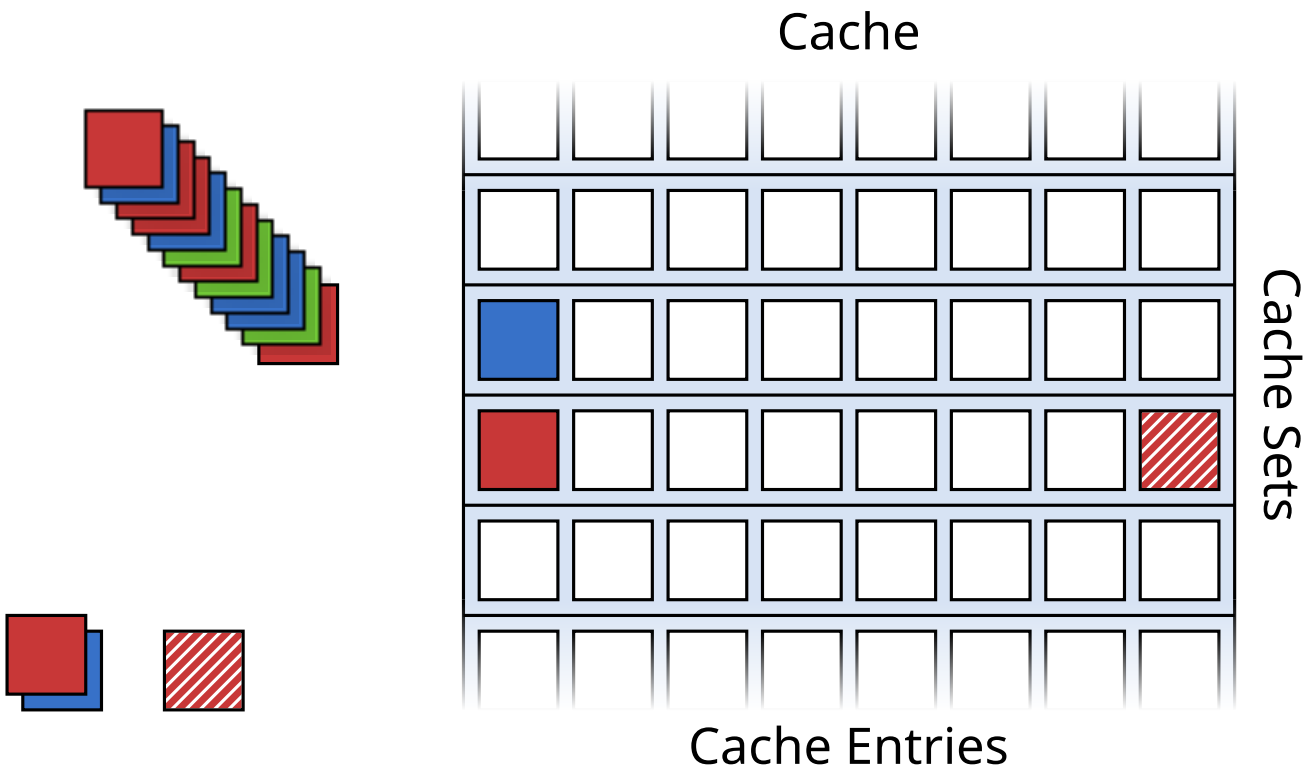
Draw pages and try to evict the target

Building Eviction Sets



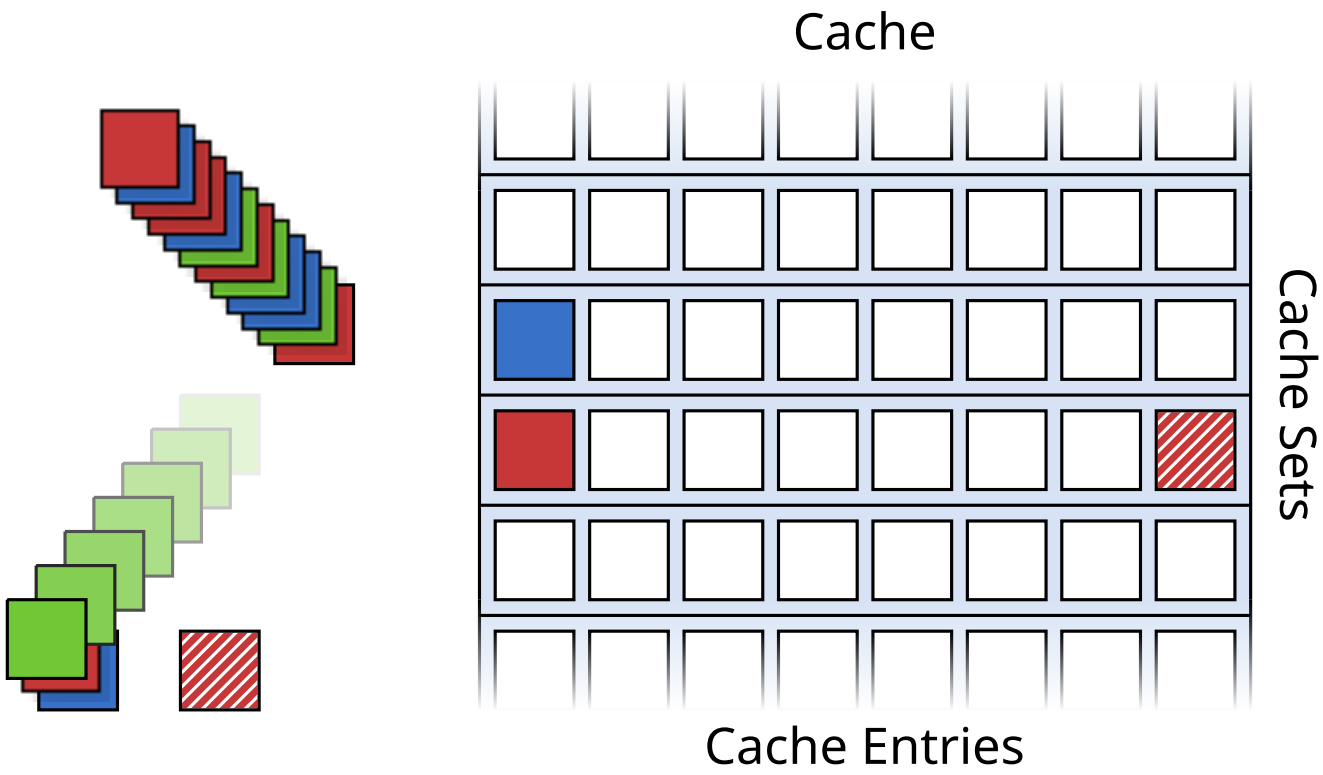
Draw pages and try to evict the target

Building Eviction Sets



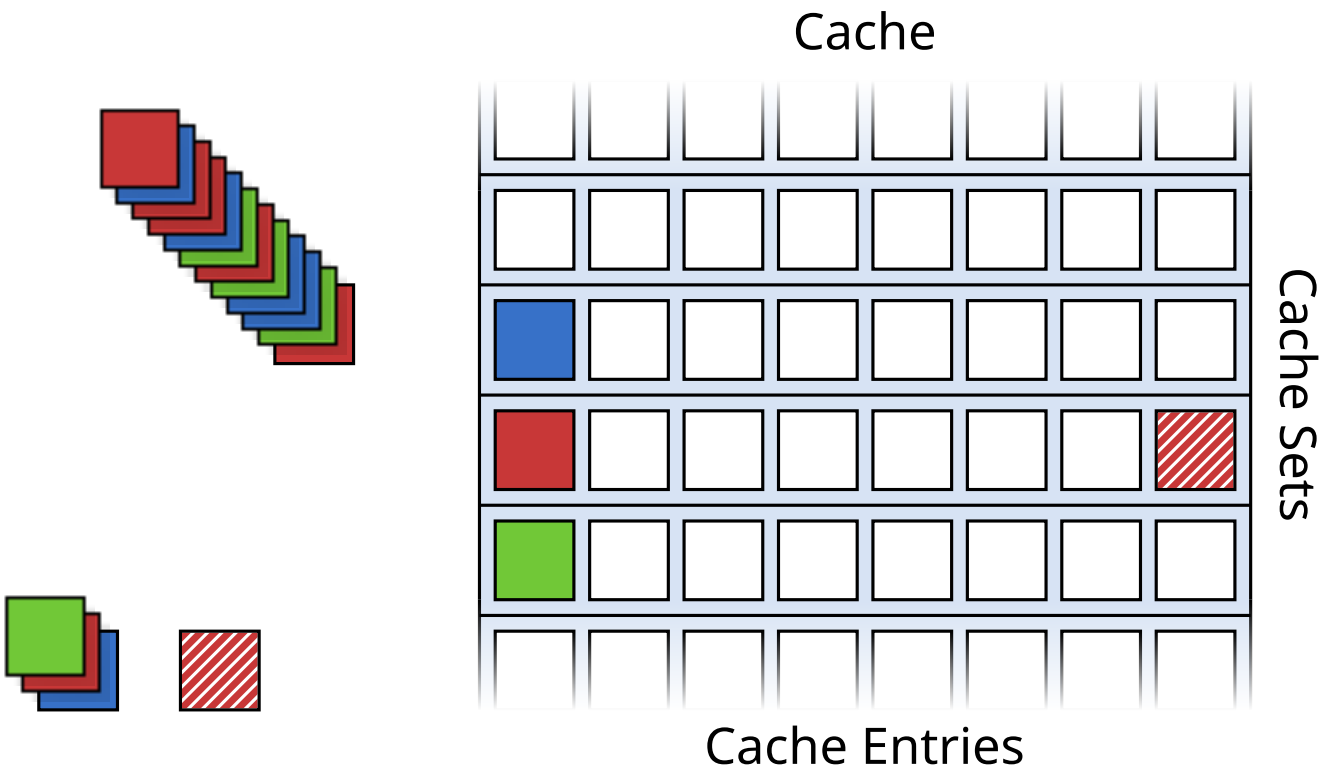
Draw pages and try to evict the target

Building Eviction Sets



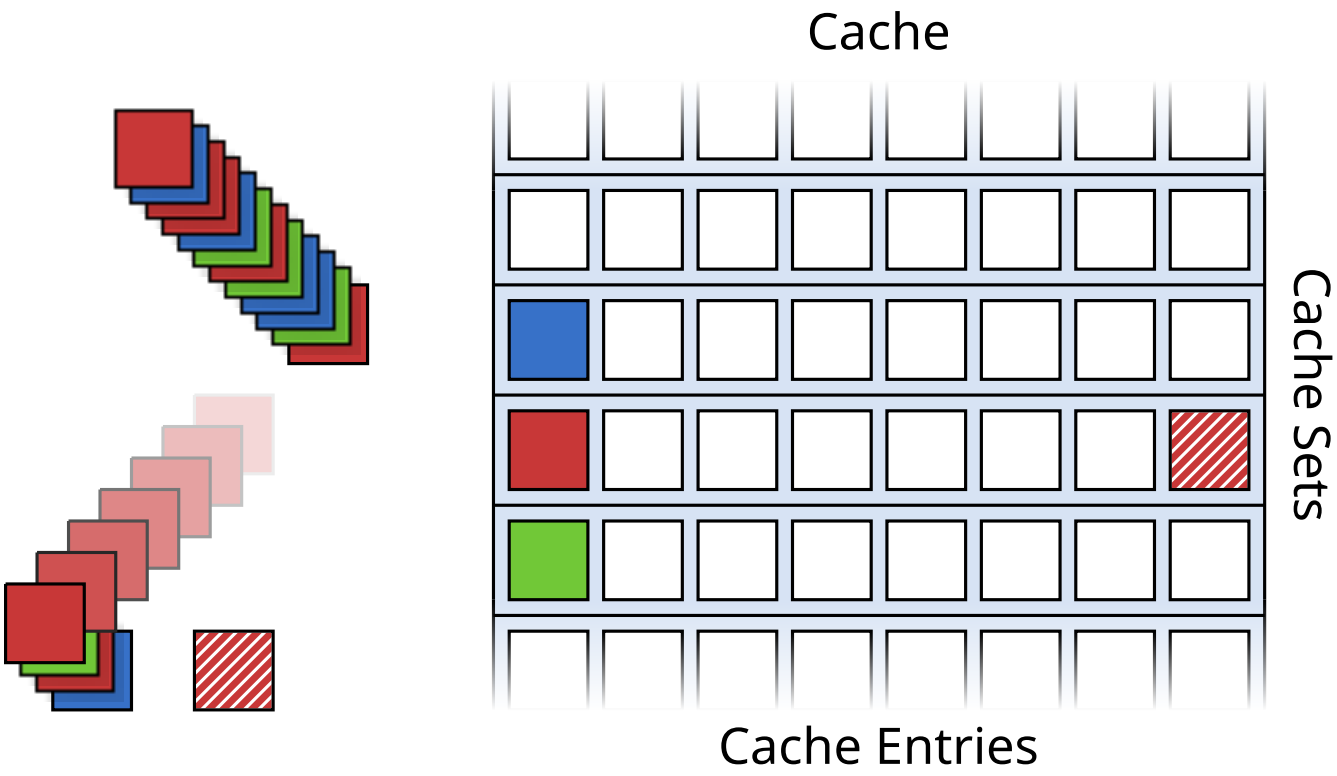
Draw pages and try to evict the target

Building Eviction Sets



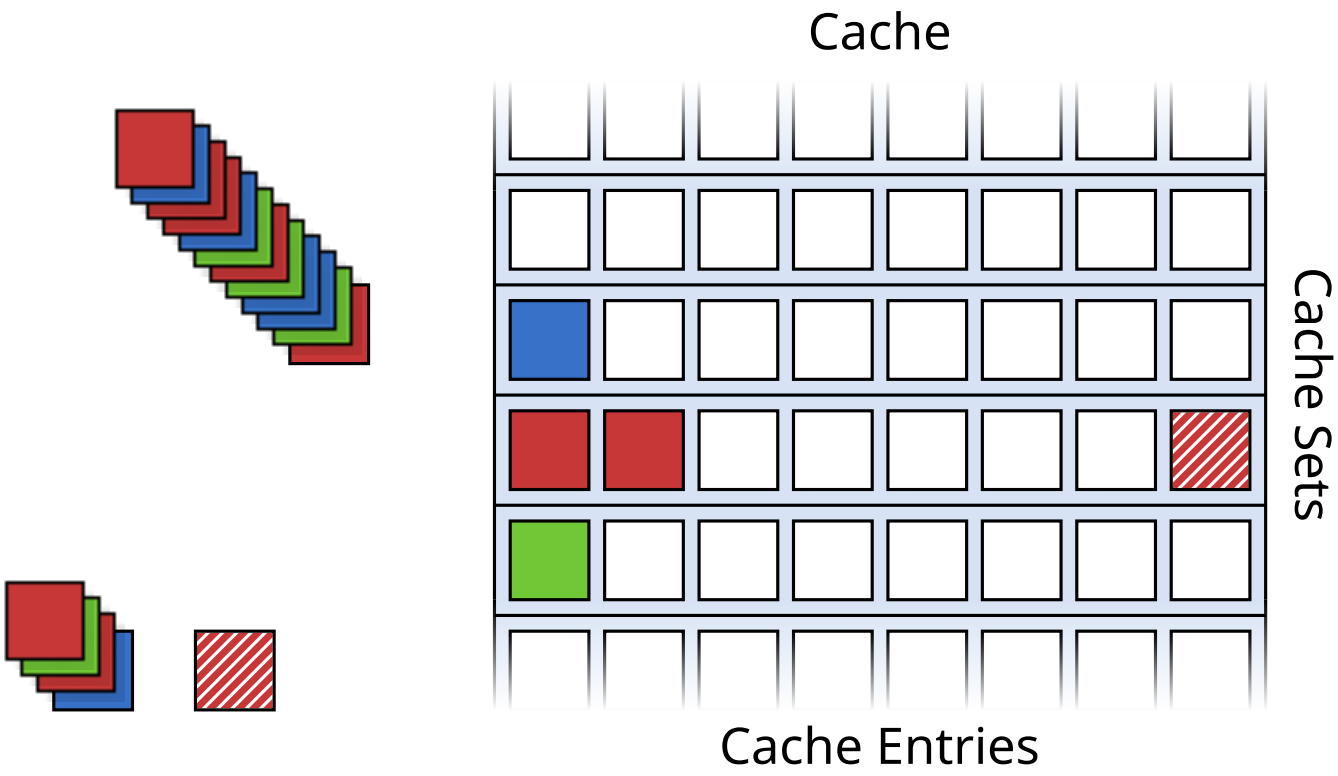
Draw pages and try to evict the target

Building Eviction Sets



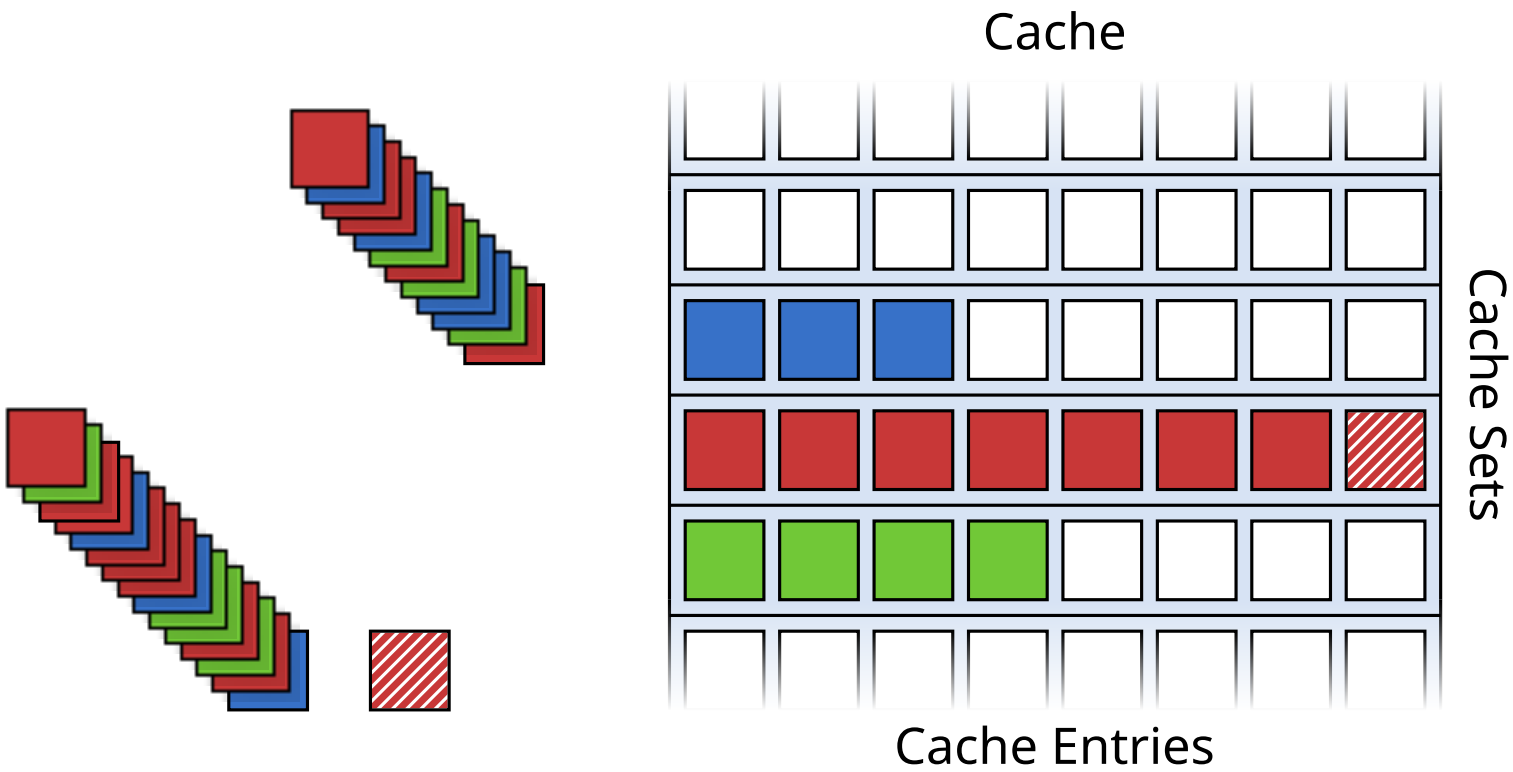
Draw pages and try to evict the target

Building Eviction Sets



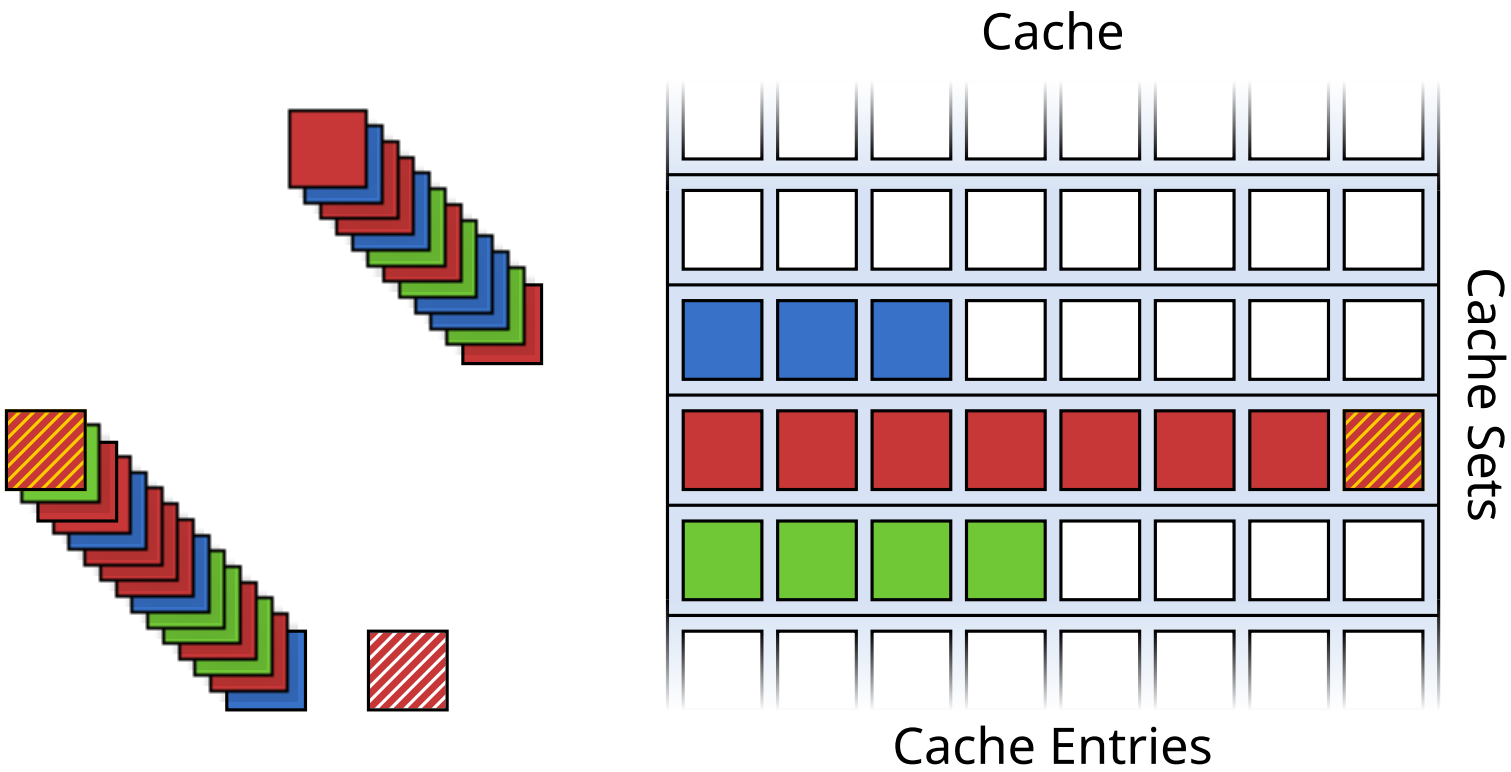
Draw pages and try to evict the target

Building Eviction Sets



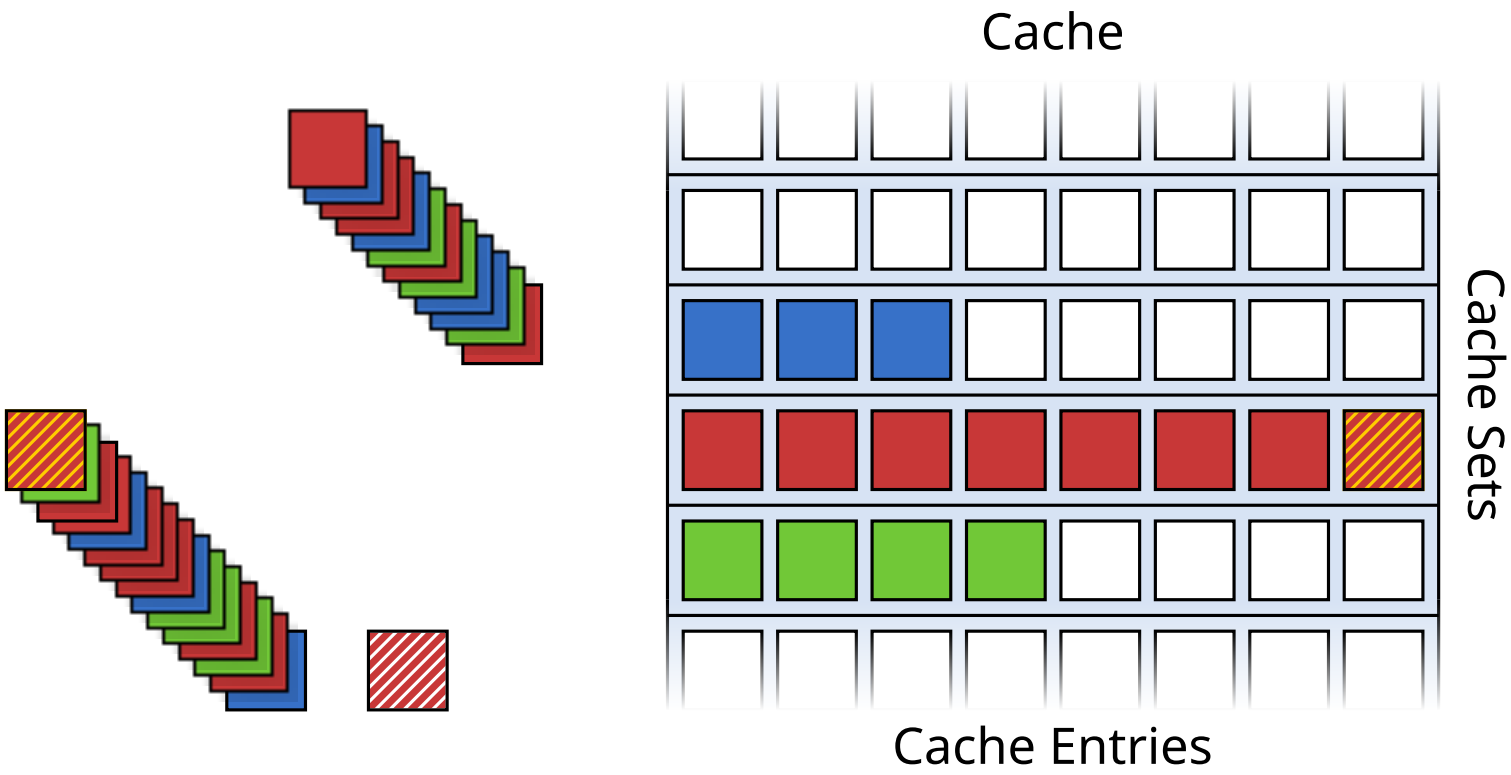
Draw pages and try to evict the target

Building Eviction Sets



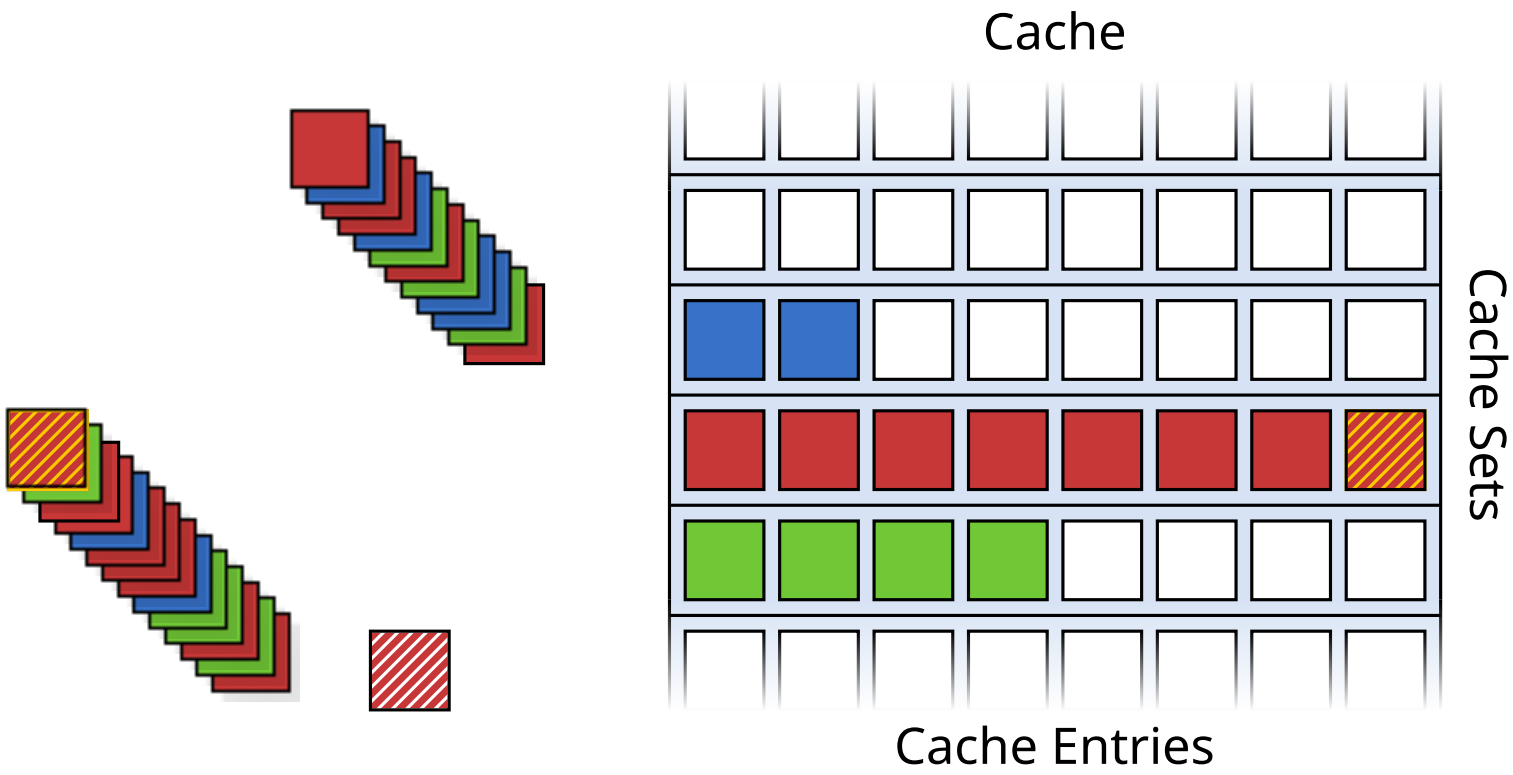
Found an eviction set

Building Eviction Sets



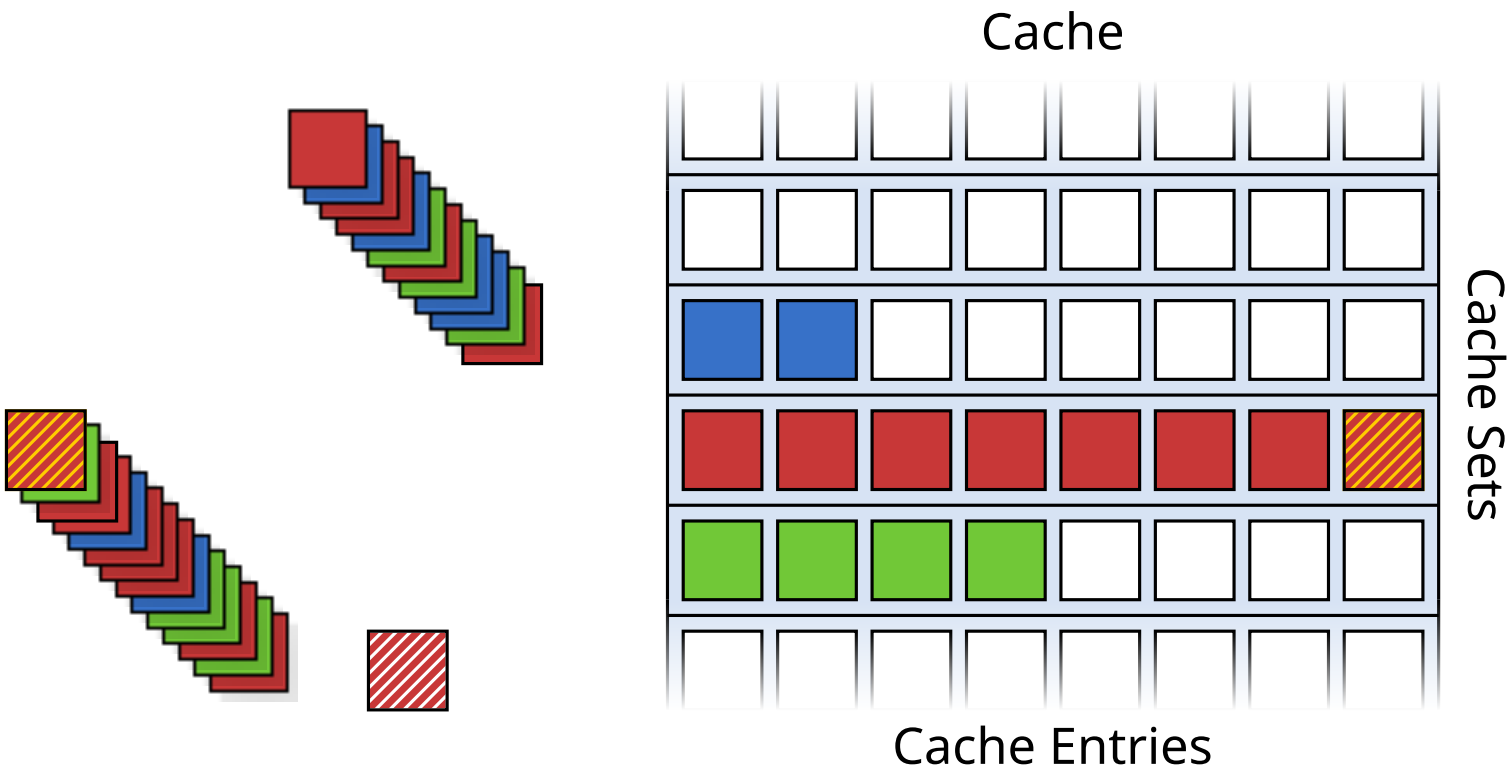
Optimize the eviction set

Building Eviction Sets



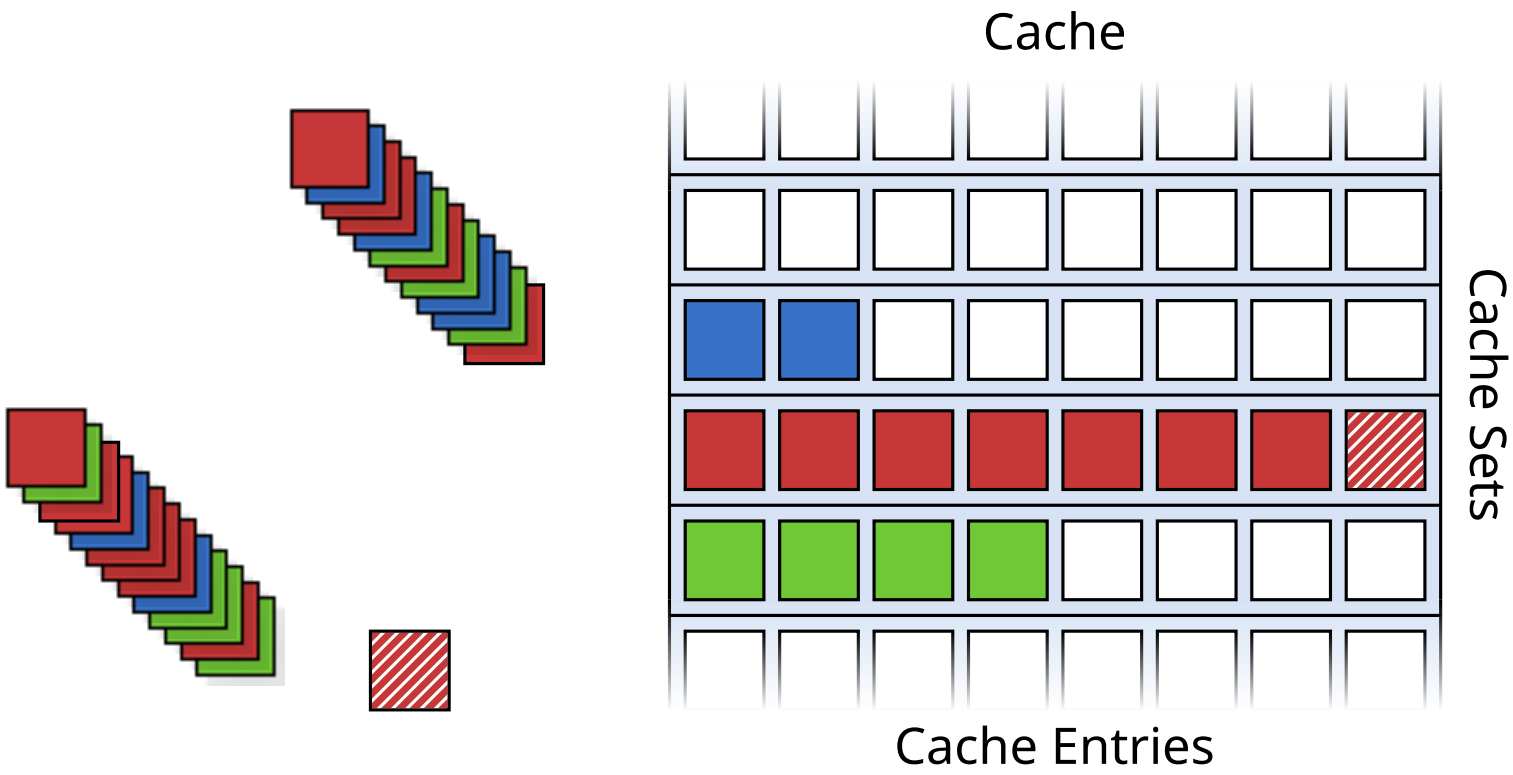
Optimize the eviction set

Building Eviction Sets



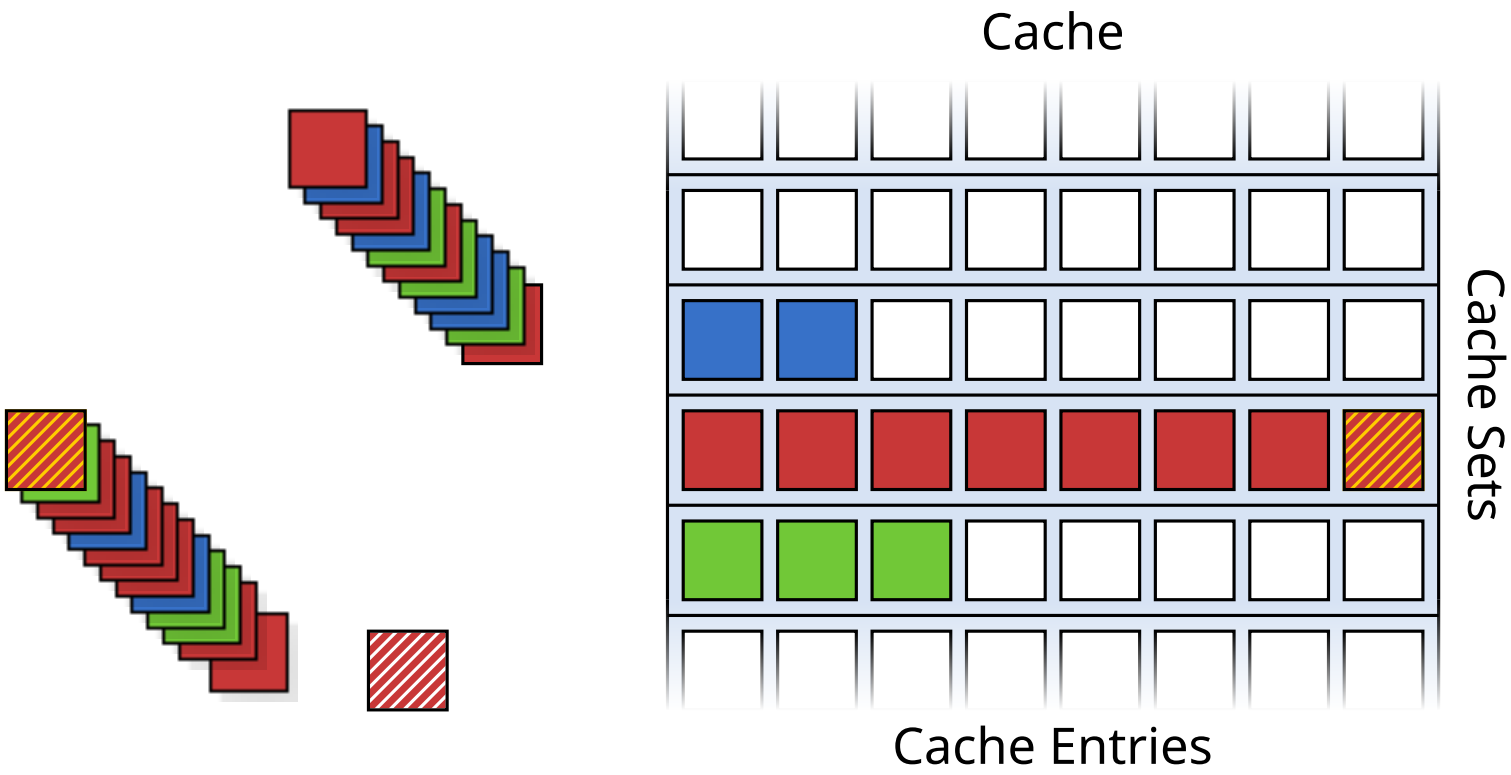
Optimize the eviction set

Building Eviction Sets



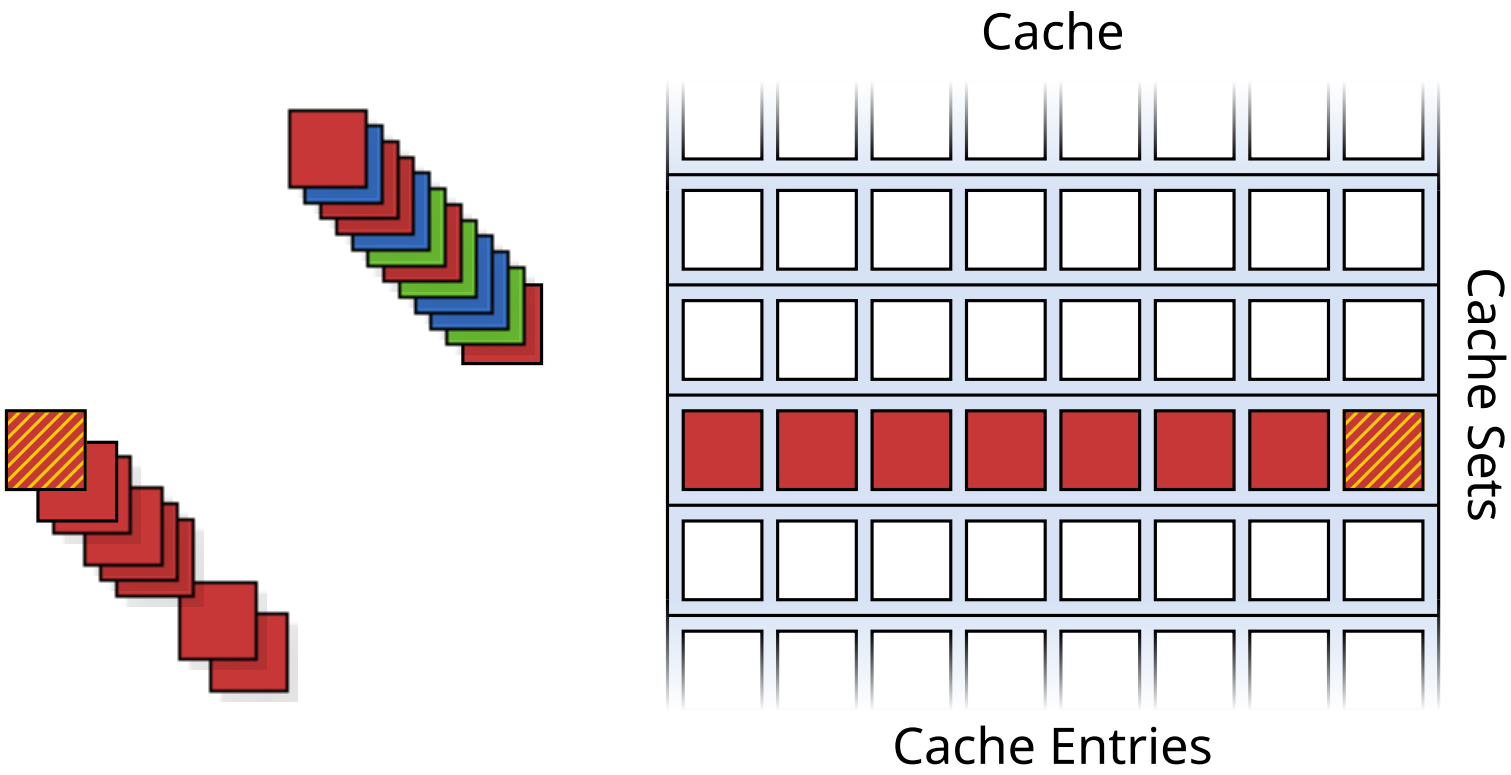
Optimize the eviction set

Building Eviction Sets



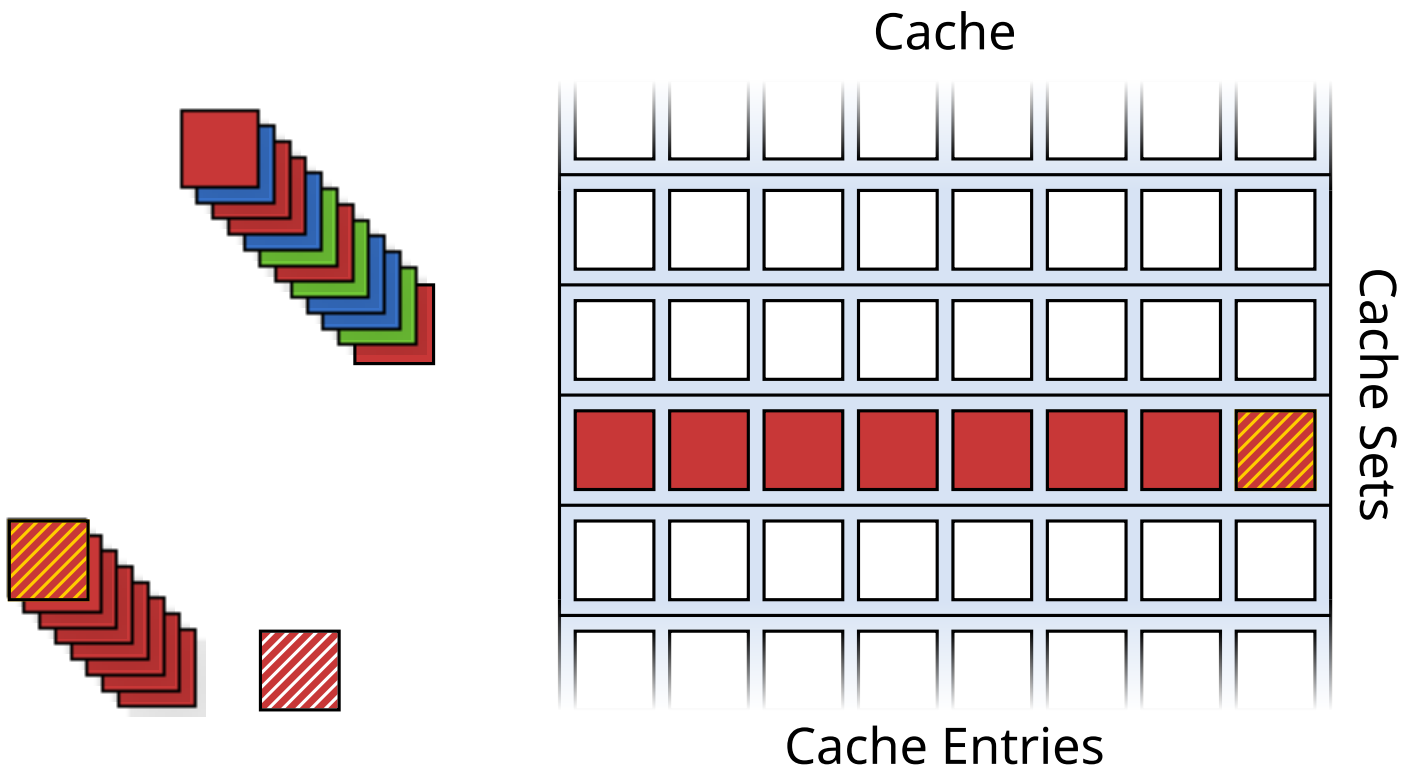
Optimize the eviction set

Building Eviction Sets



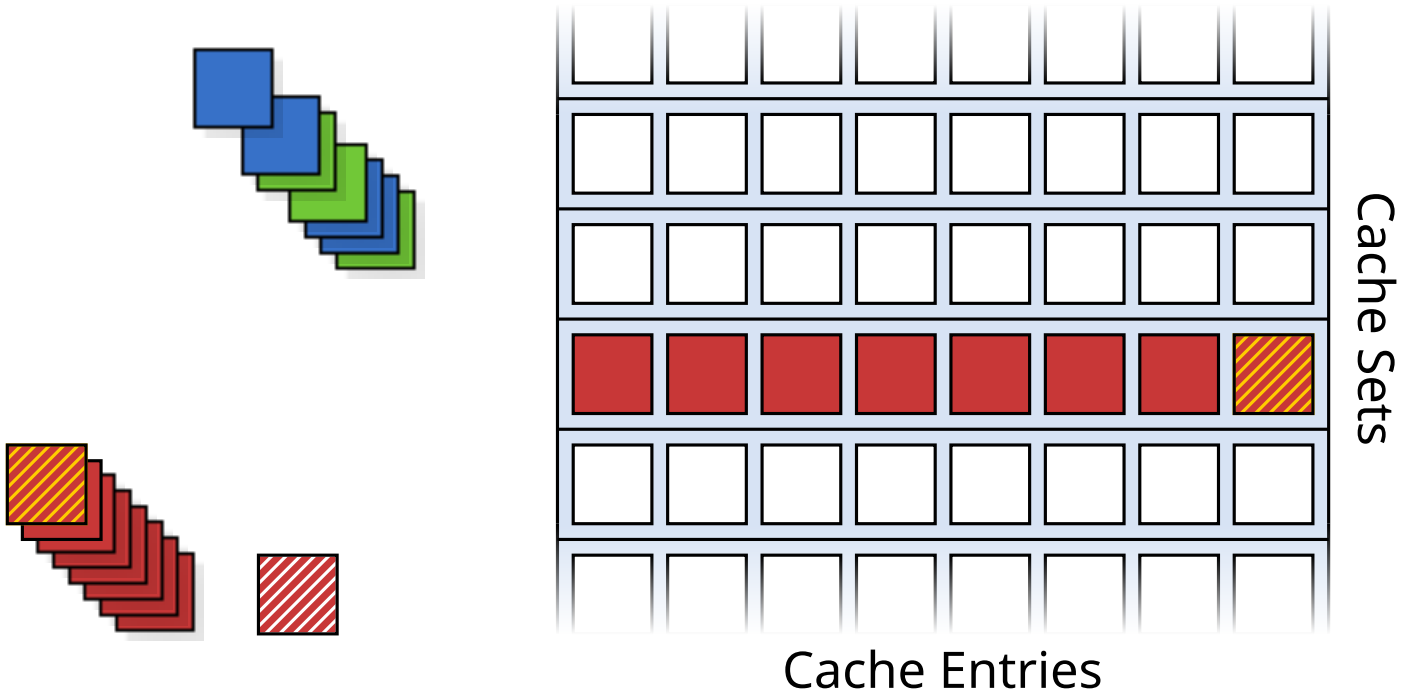
Optimize the eviction set

Building Eviction Sets



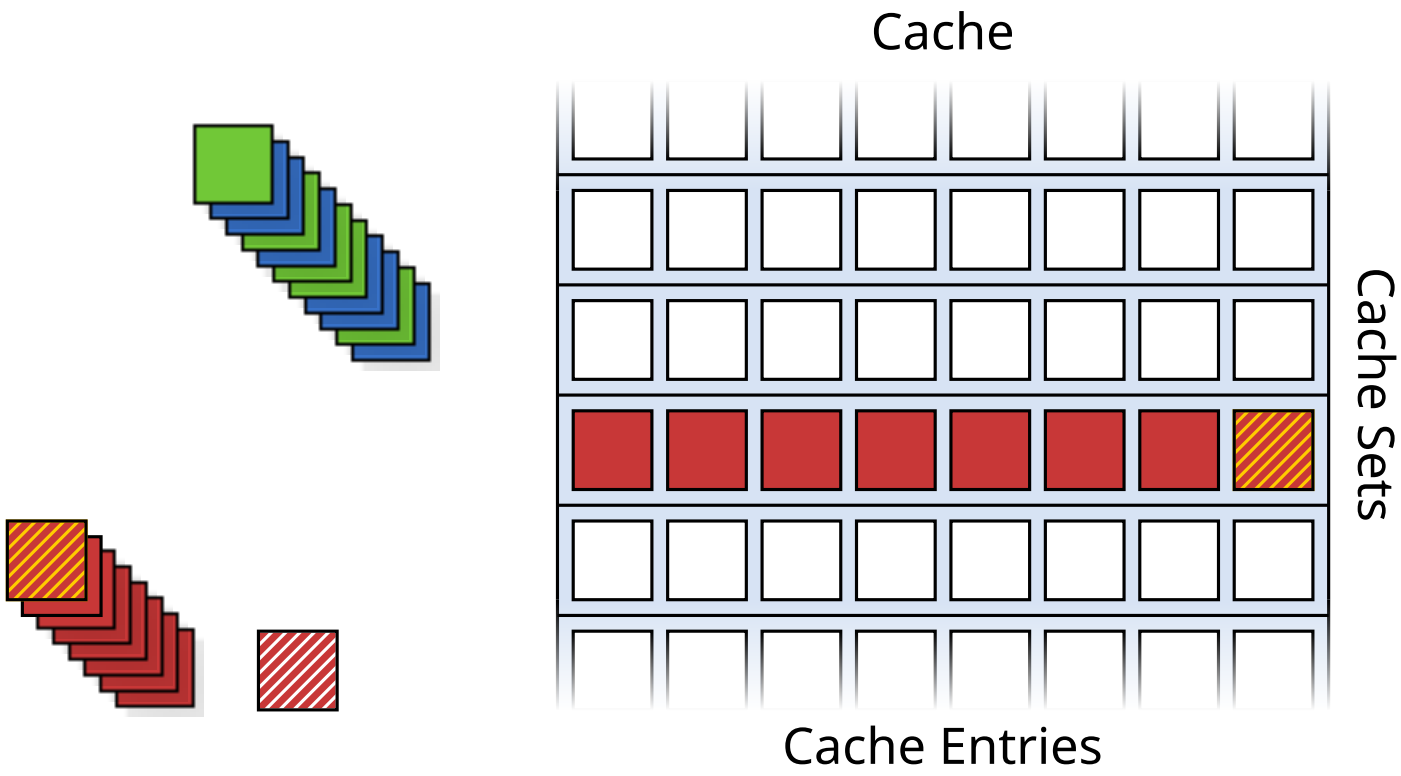
Optimal eviction set found

Building Eviction Sets



Filter red pages

Building Eviction Sets



Filter red pages

Building Eviction Sets

This technique can also be applied to page tables

Challenges

- ✓ Avoid noise from high-level page tables
- ✓ Avoid noise from pages
- ✓ Build eviction sets

XLATE + PROBE

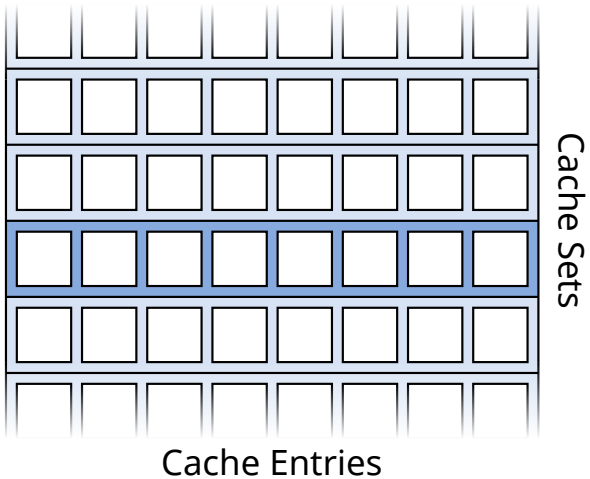
It’s time for the big picture.

XLATE + PROBE

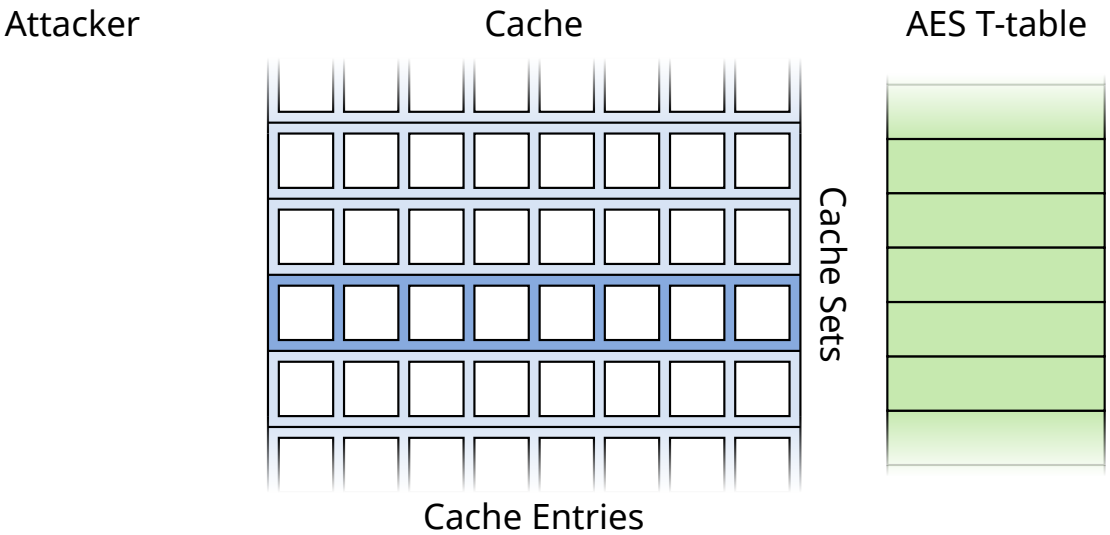
Attacker

Cache

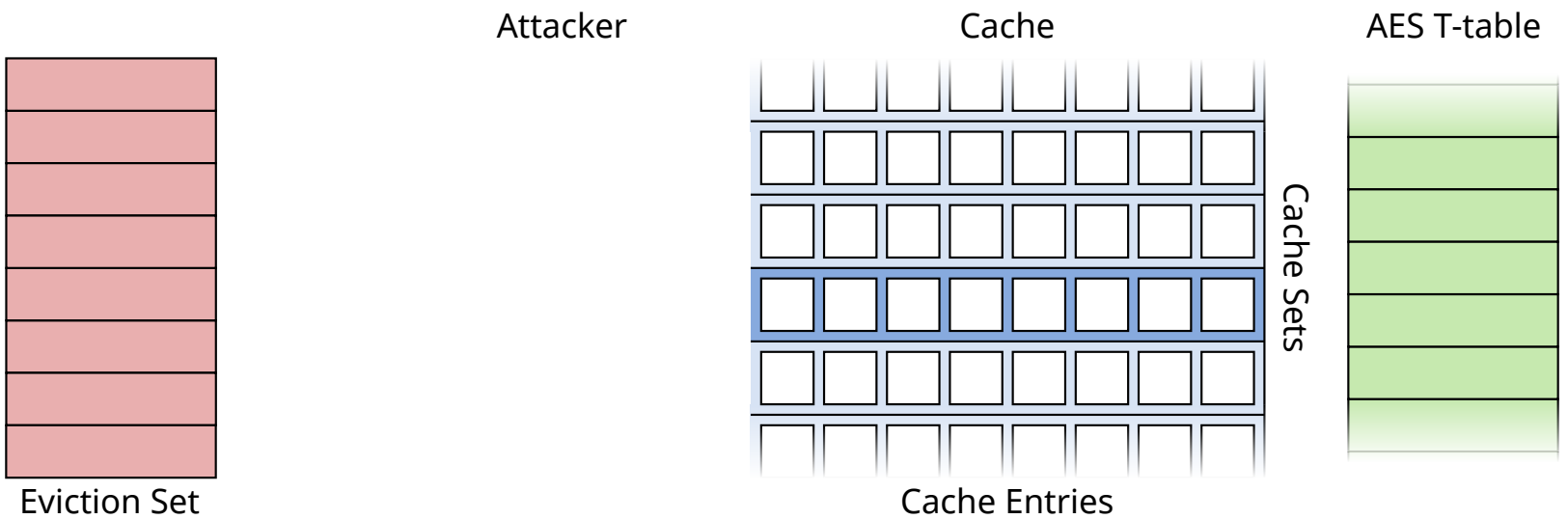
AES T-table



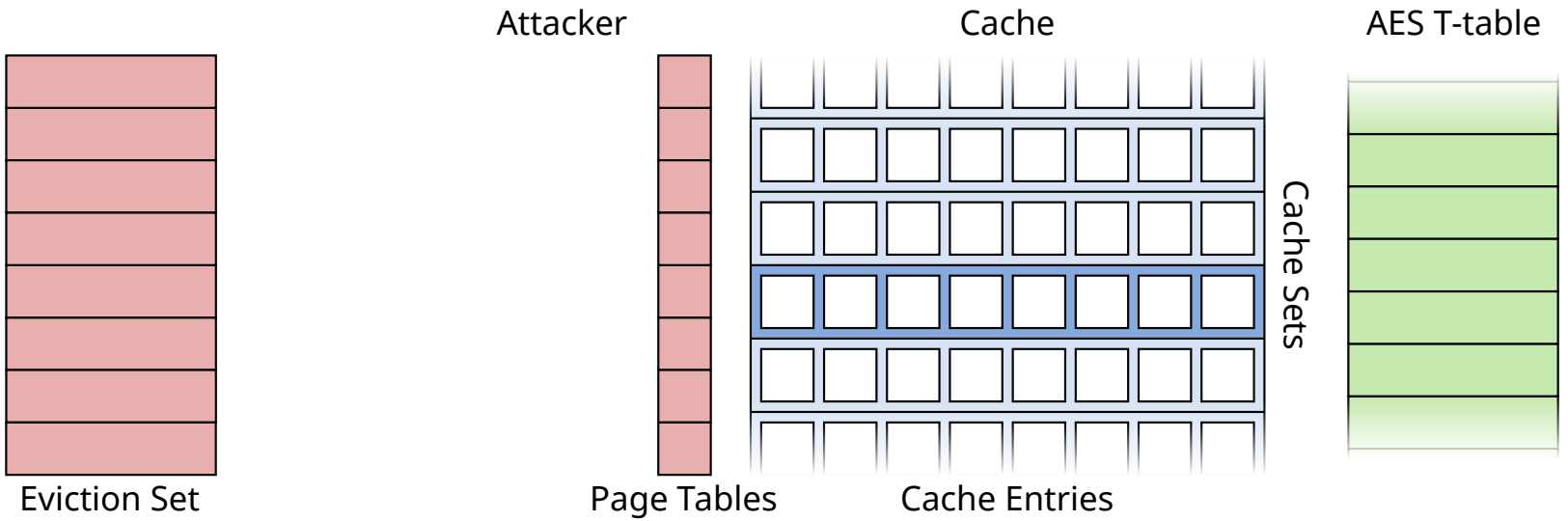
XLATE + PROBE



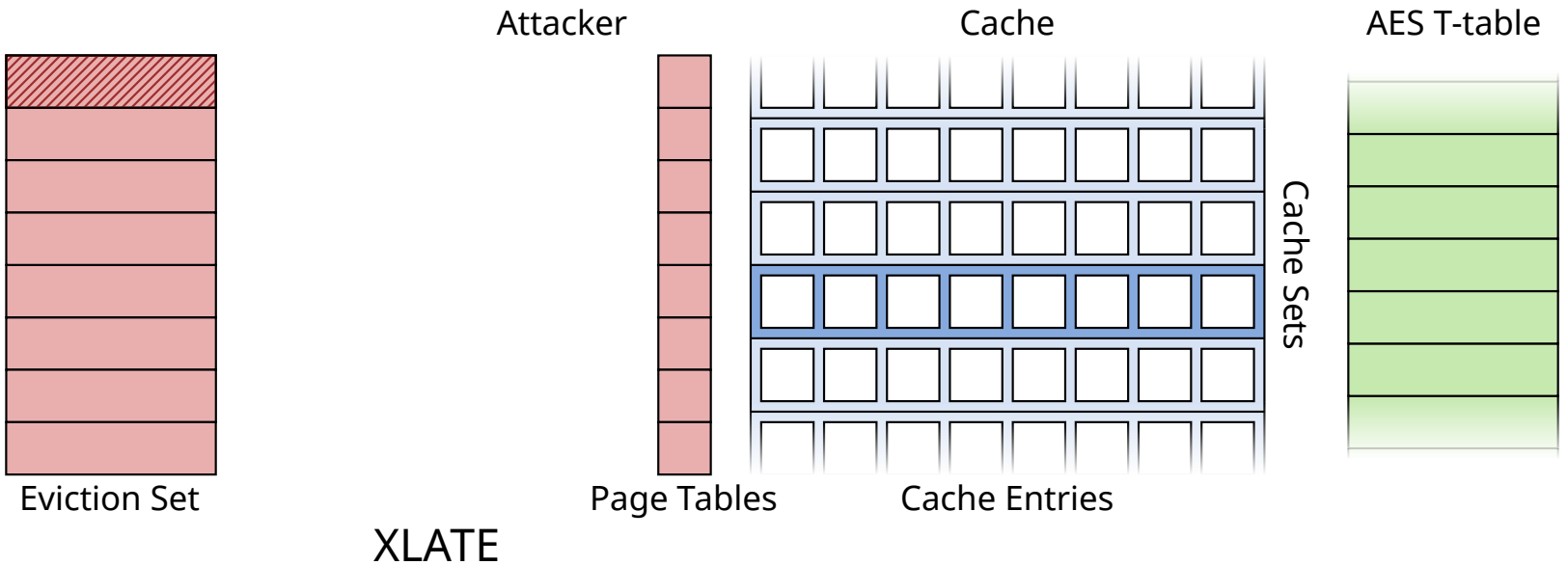
XLATE + PROBE


Attacker
Cache
AES T-table
Cache Sets
Eviction Set
Cache Entries

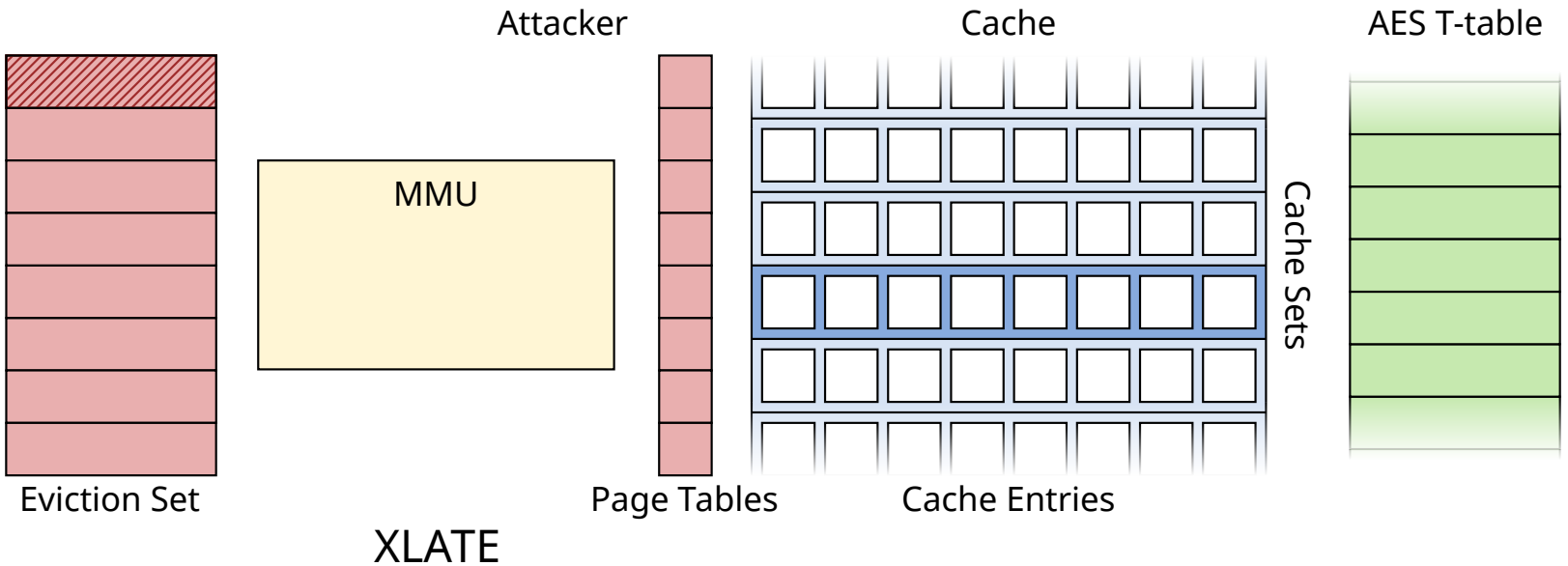
XLATE + PROBE


Attacker
Cache
AES T-table
Cache Sets
Eviction Set
Page Tables
Cache Entries

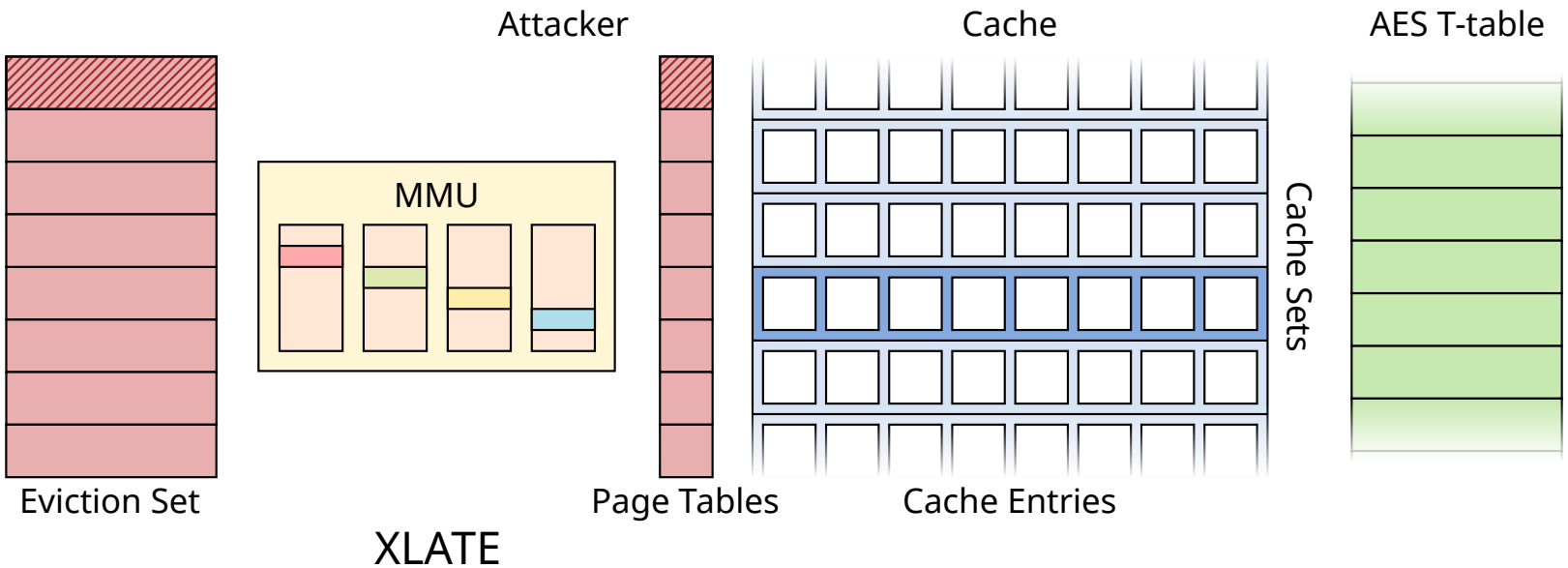
XLATE + PROBE


Attacker
Cache
AES T-table
Cache Sets
Eviction Set
Page Tables
Cache Entries
XLATE

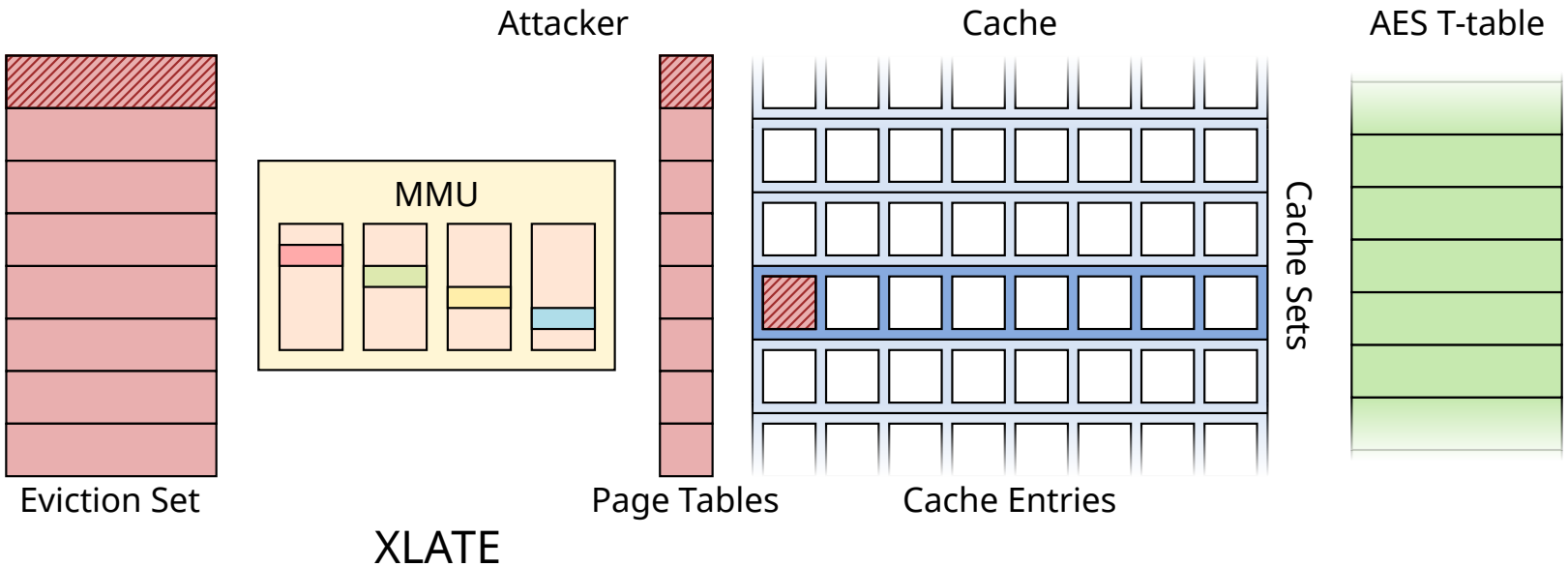
XLATE + PROBE


Attacker
Cache
AES T-table
MMU
Cache Sets
Eviction Set
Page Tables
Cache Entries
XLATE

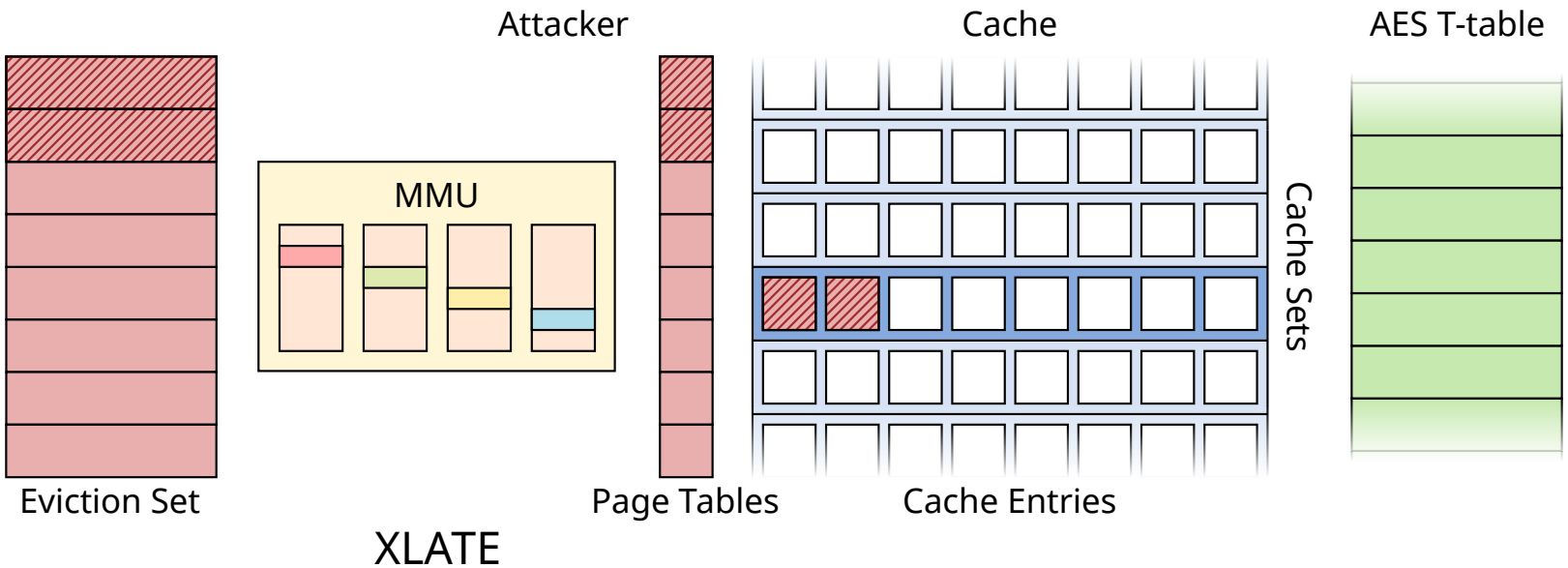
XLATE + PROBE


Attacker
Cache
AES T-table
MMU
Cache Sets
Eviction Set
Page Tables
Cache Entries
XLATE

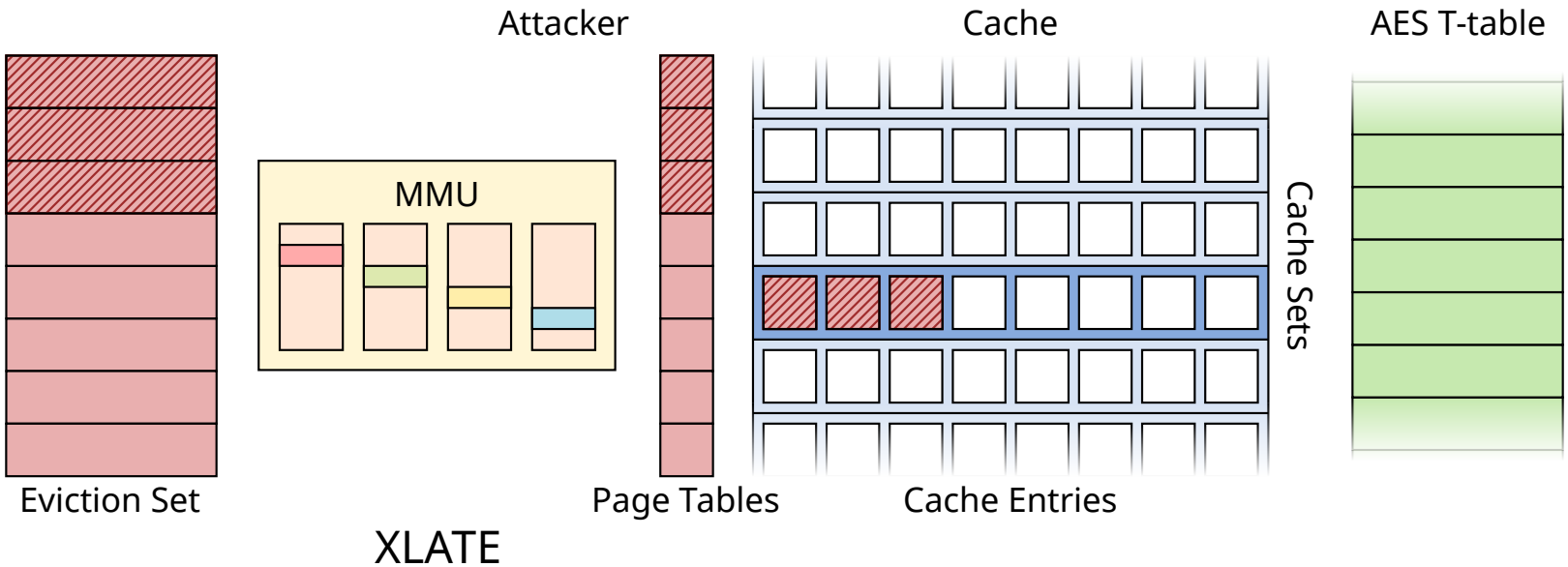
XLATE + PROBE


Attacker
Cache
AES T-table
MMU
Cache Sets
Eviction Set
Page Tables
Cache Entries
XLATE

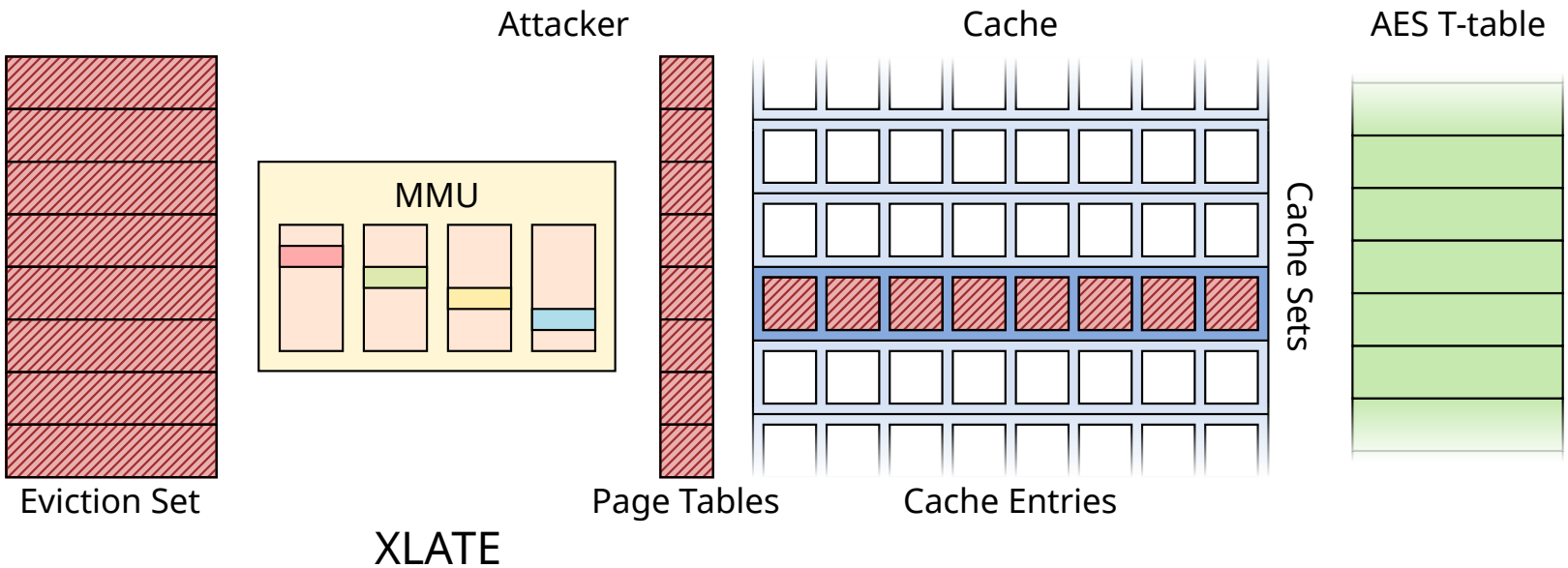
XLATE + PROBE


Attacker
Cache
AES T-table
MMU
Cache Sets
Eviction Set
Page Tables
Cache Entries
XLATE

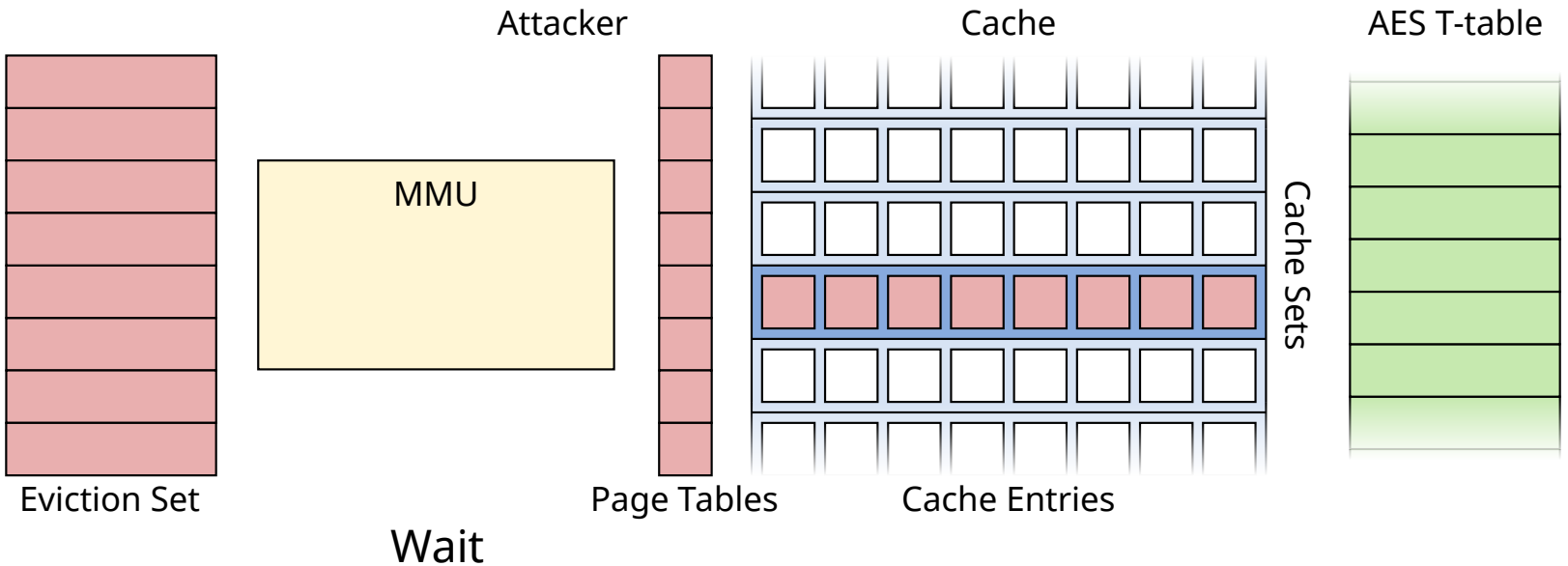
XLATE + PROBE


Attacker
Cache
AES T-table
MMU
Cache Sets
Eviction Set
Page Tables
Cache Entries
XLATE

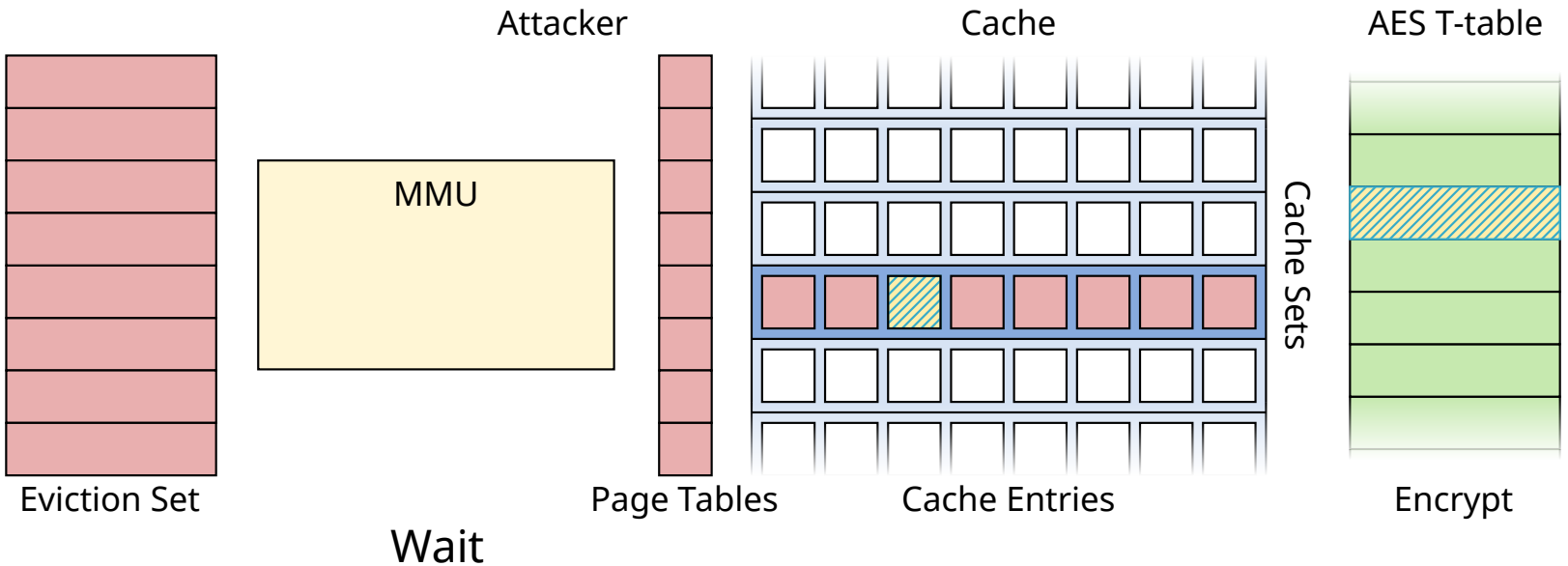
XLATE + PROBE


Attacker
Cache
AES T-table
MMU
Cache Sets
Eviction Set
Page Tables
Cache Entries
XLATE

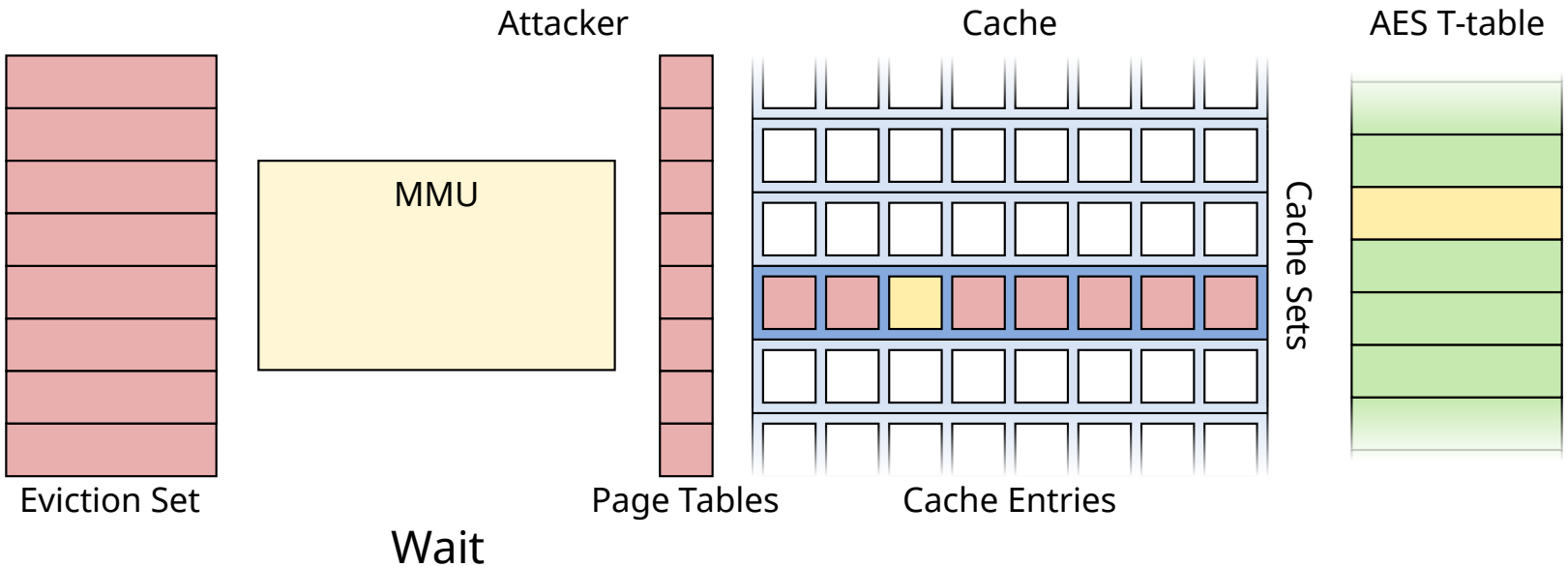
XLATE + PROBE


Attacker
Cache
AES T-table
MMU
Cache Sets
Eviction Set
Page Tables
Cache Entries
Wait

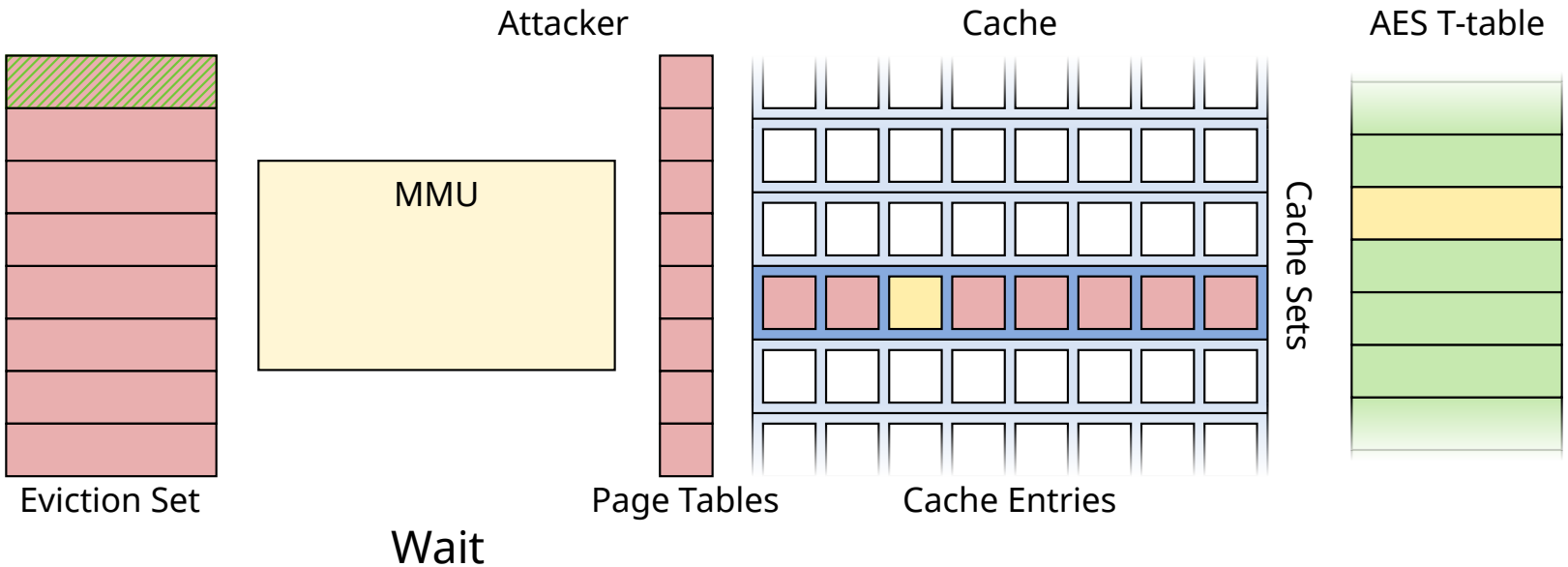
XLATE + PROBE


Attacker
Cache
AES T-table
MMU
Cache Sets
Eviction Set
Page Tables
Cache Entries
Encrypt
Wait

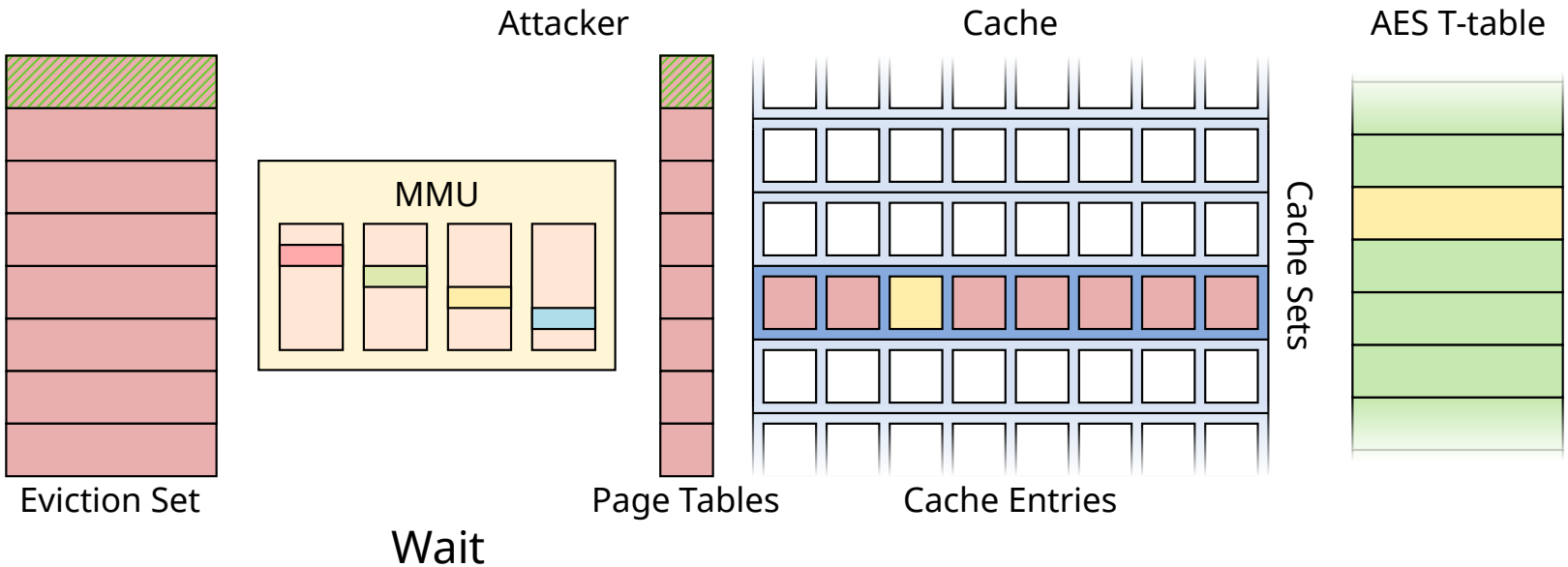
XLATE + PROBE


Attacker
Cache
AES T-table
MMU
Cache Sets
Eviction Set
Page Tables
Cache Entries
Wait

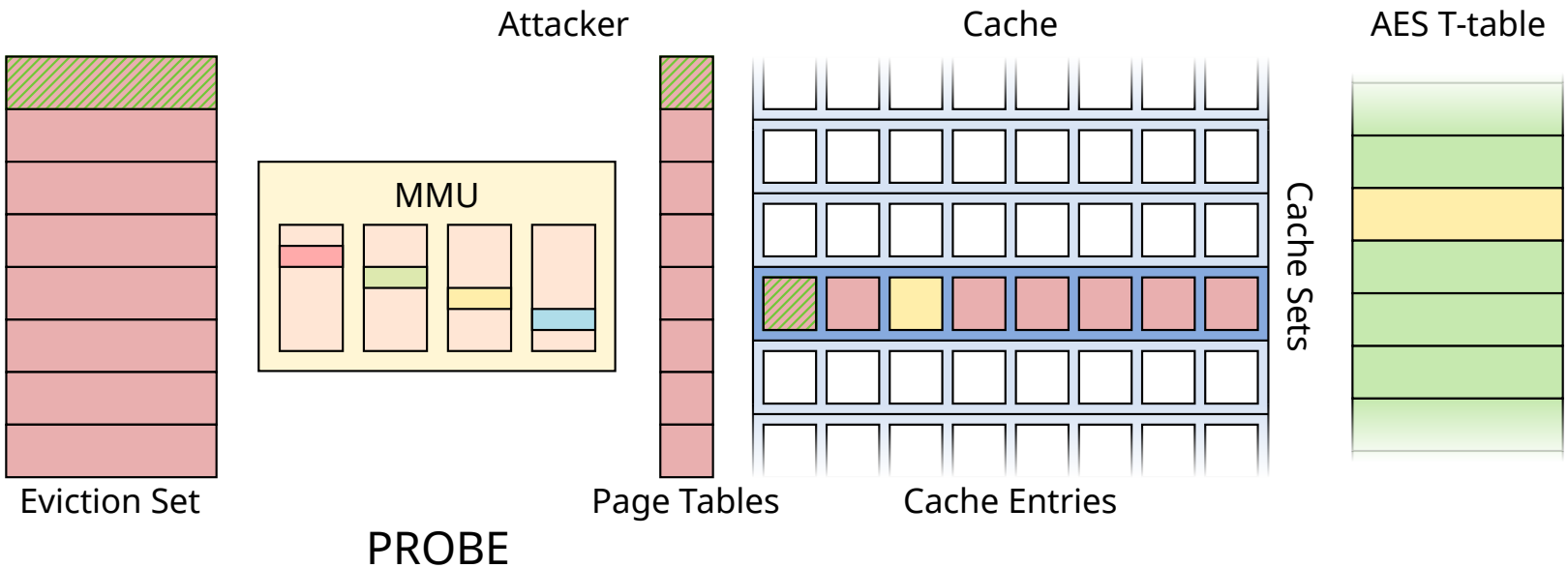
XLATE + PROBE


Attacker
Cache
AES T-table
MMU
Cache Sets
Eviction Set
Page Tables
Cache Entries
Wait

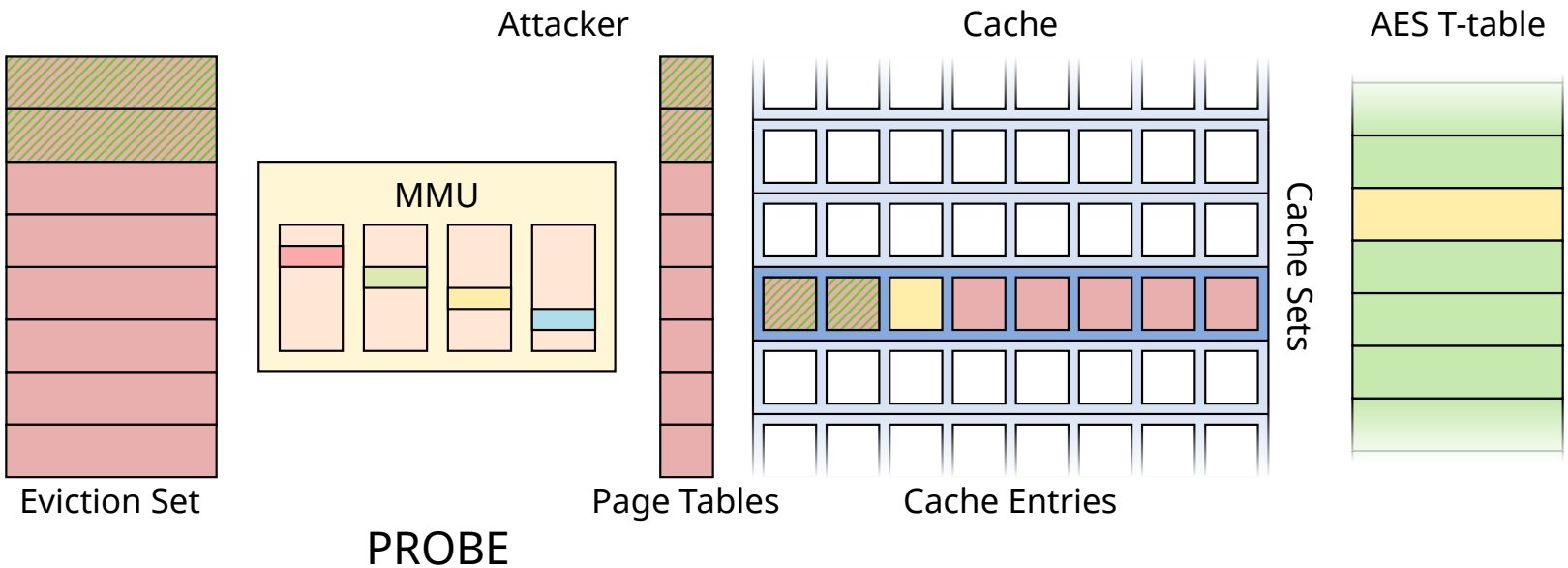
XLATE + PROBE


Attacker
Cache
AES T-table
MMU
Cache Sets
Eviction Set
Page Tables
Cache Entries
Wait

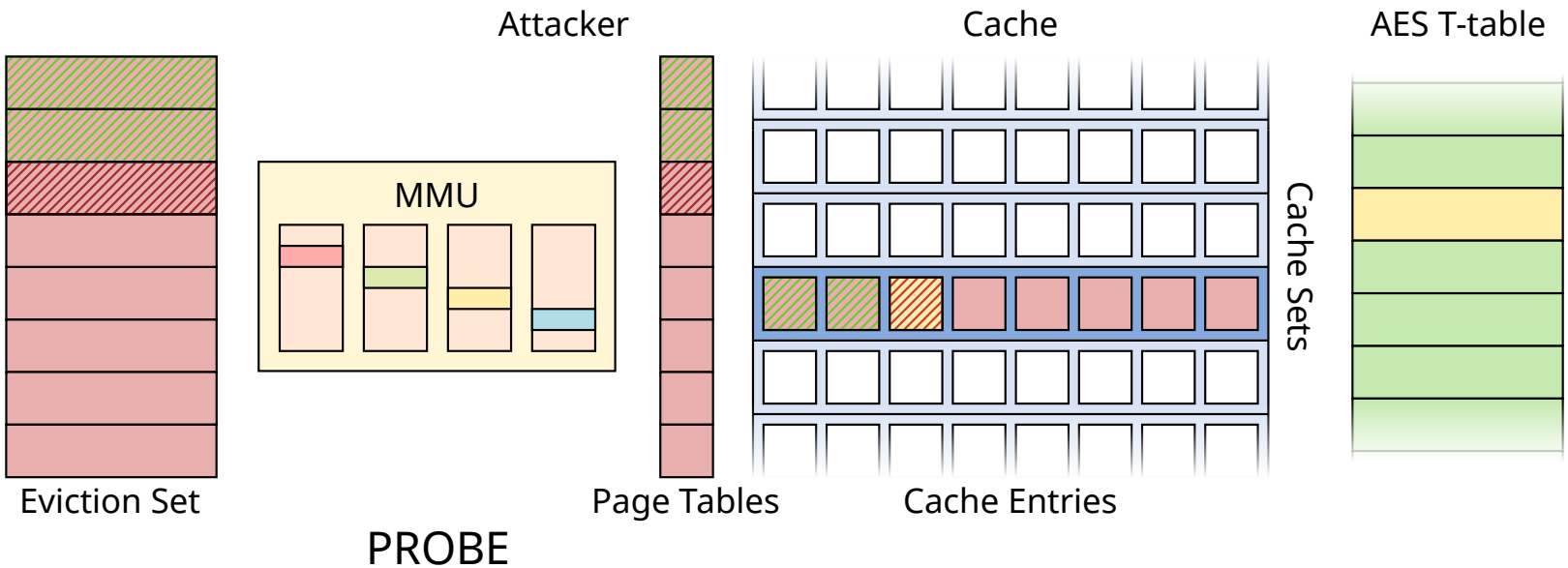
XLATE + PROBE


Attacker
Cache
AES T-table
MMU
Cache Sets
Eviction Set
Page Tables
Cache Entries
PROBE

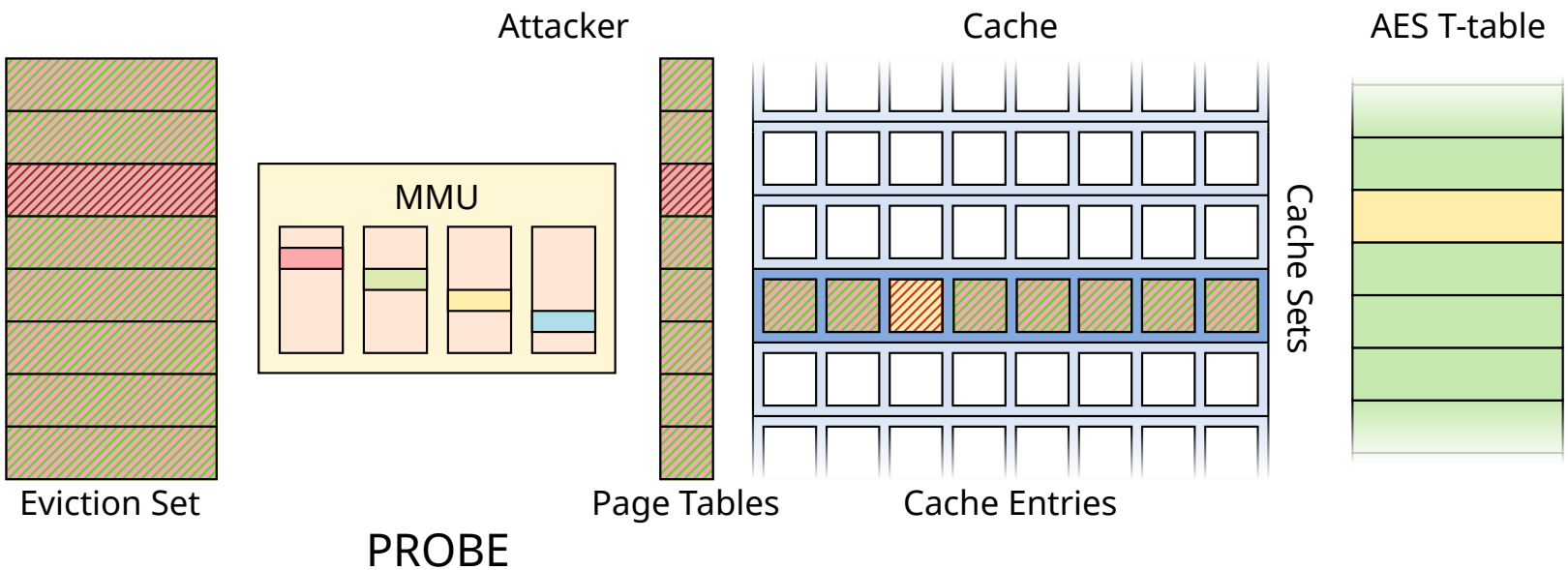
XLATE + PROBE


Attacker
Cache
AES T-table
MMU
Cache Sets
Eviction Set
Page Tables
Cache Entries
PROBE

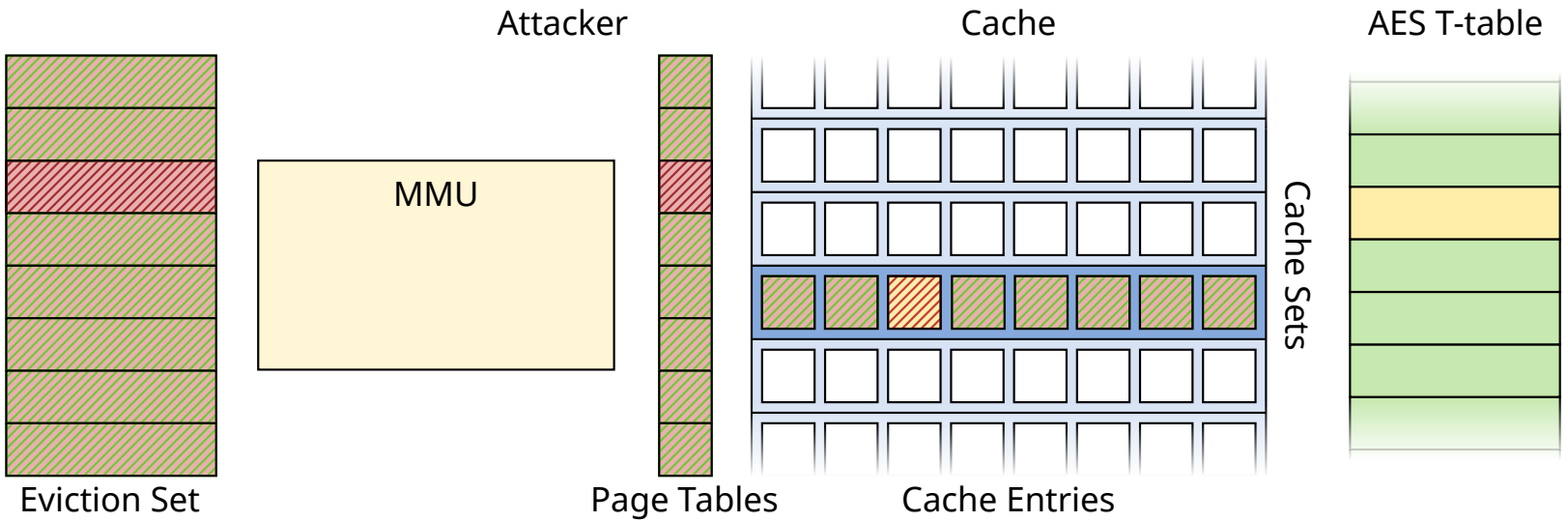
XLATE + PROBE


Attacker
Cache
AES T-table
MMU
Cache Sets
Eviction Set
Page Tables
Cache Entries
PROBE

XLATE + PROBE


Attacker
Cache
AES T-table
MMU
Cache Sets
Eviction Set
Page Tables
Cache Entries
PROBE

XLATE + PROBE



PROBE

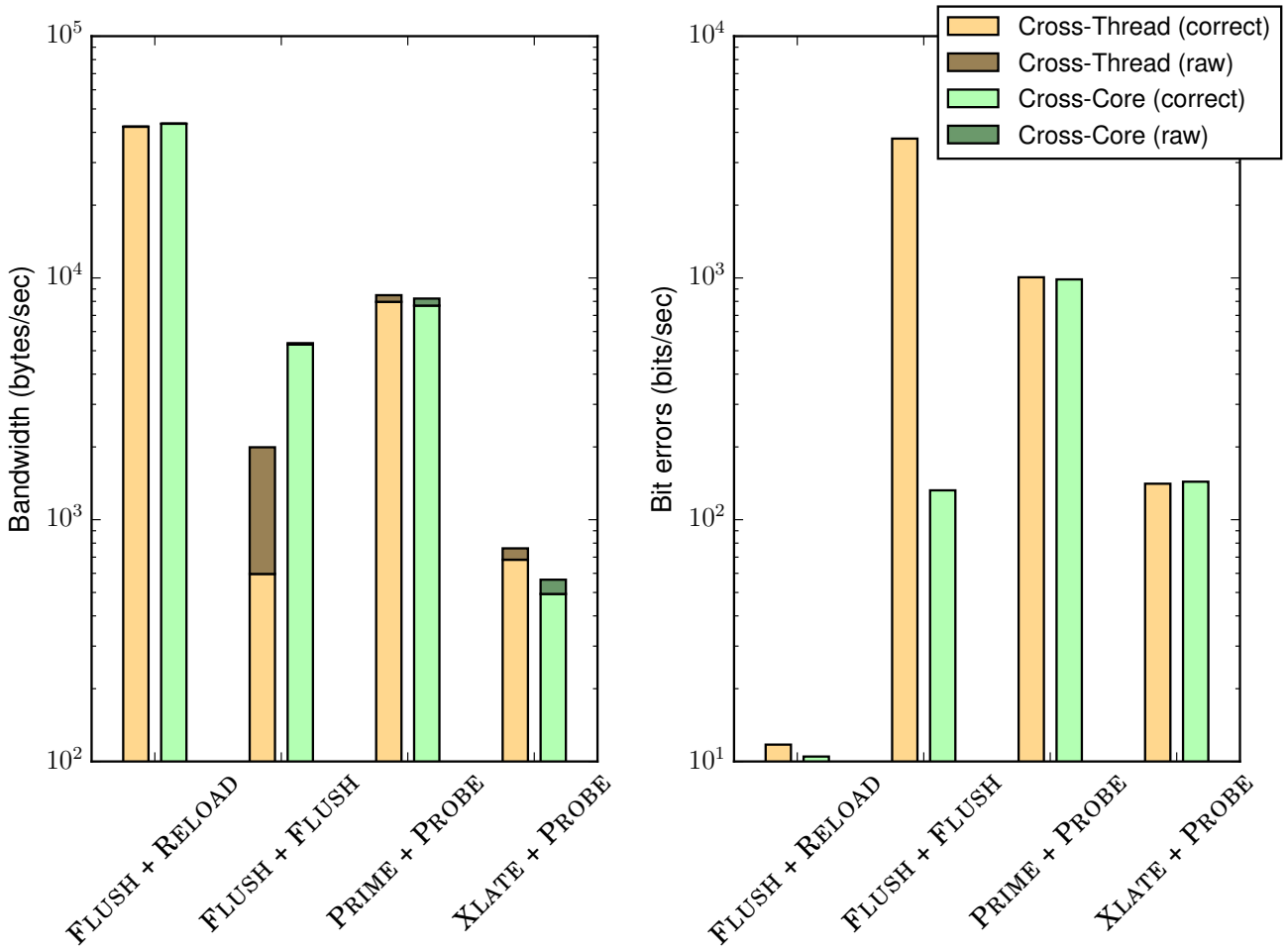
AES encrypt used the same cache set

Evaluation

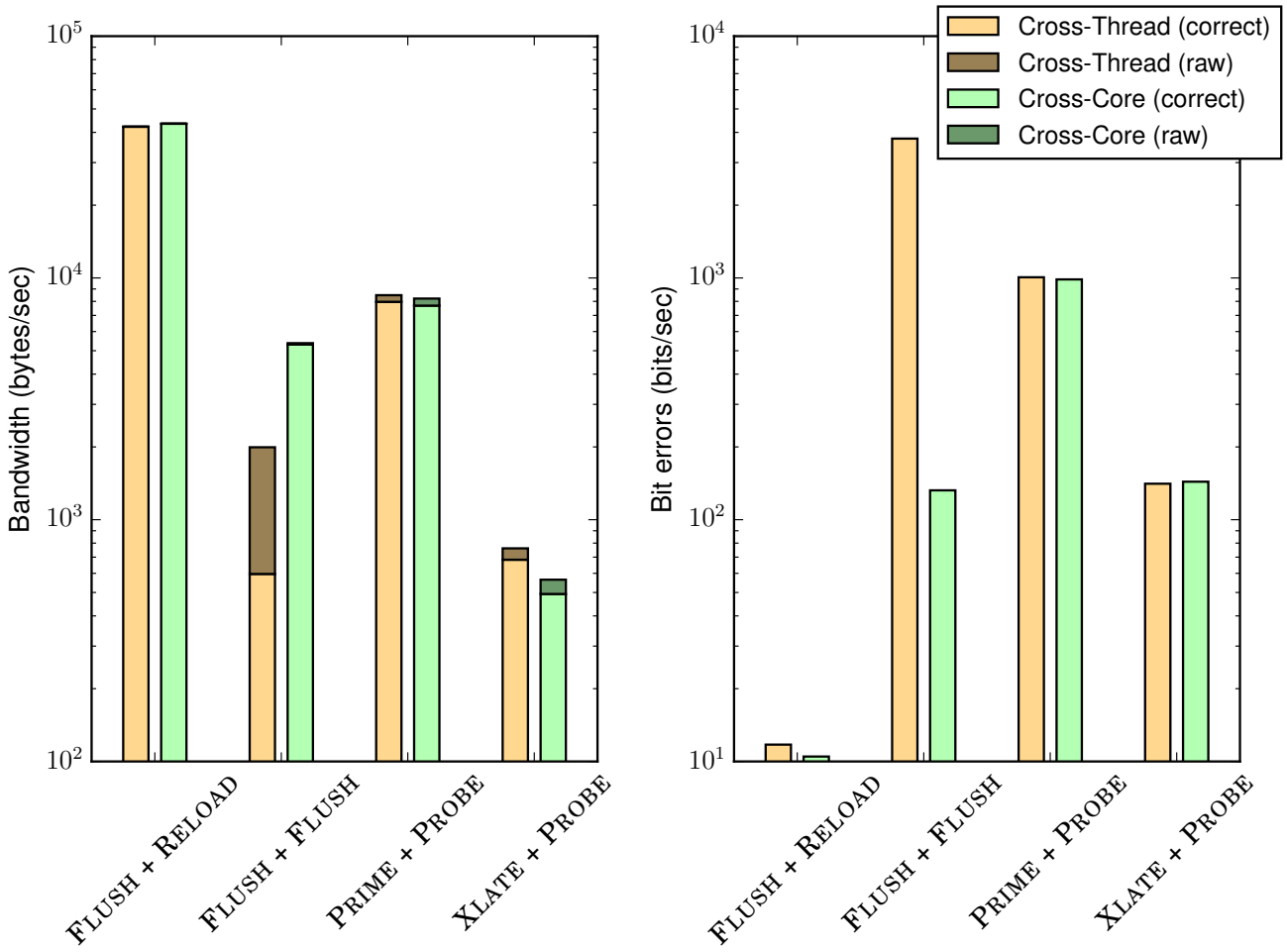
Evaluation

- Reliability
- Effectiveness
- Cache defenses

Reliability

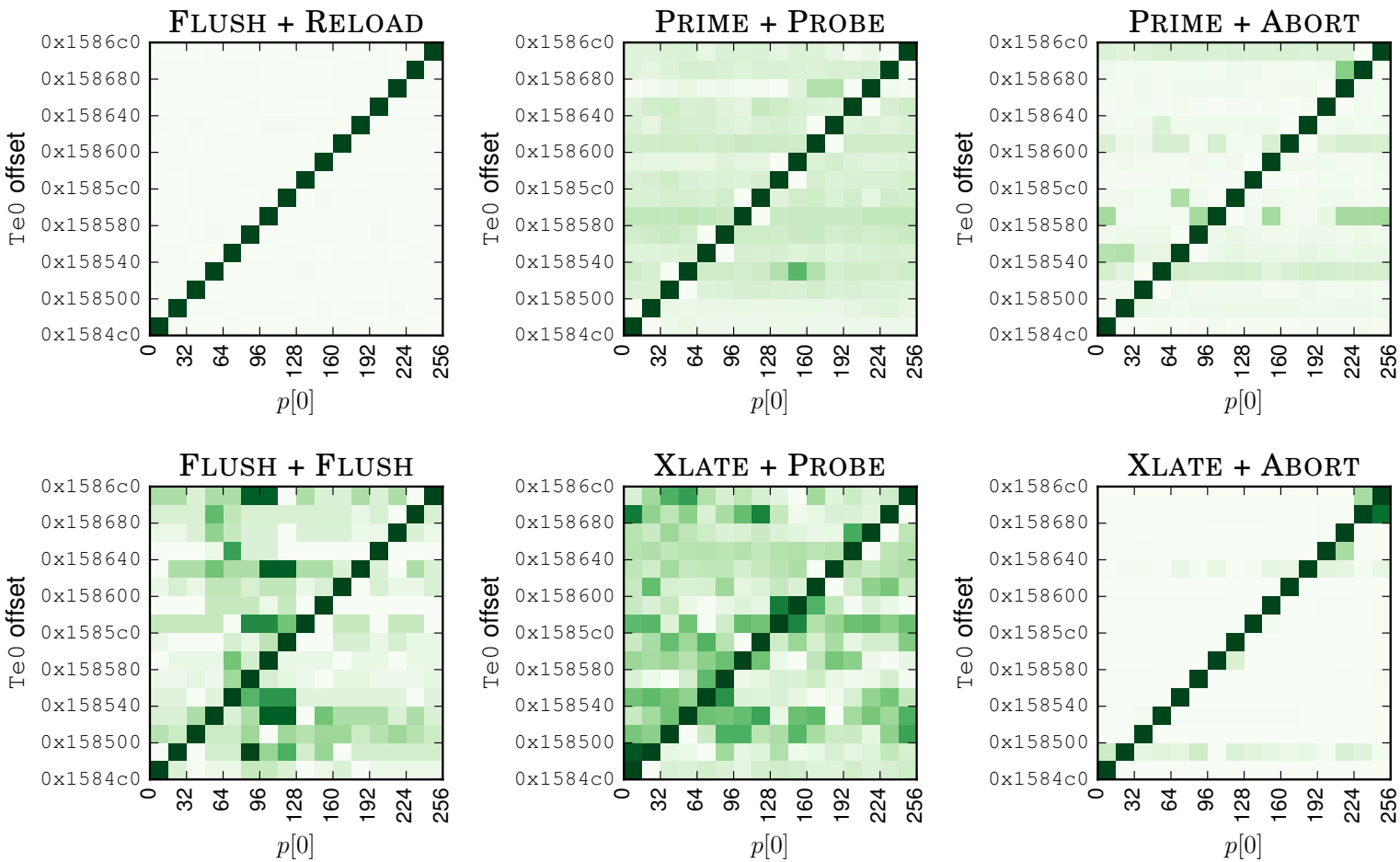


Reliability

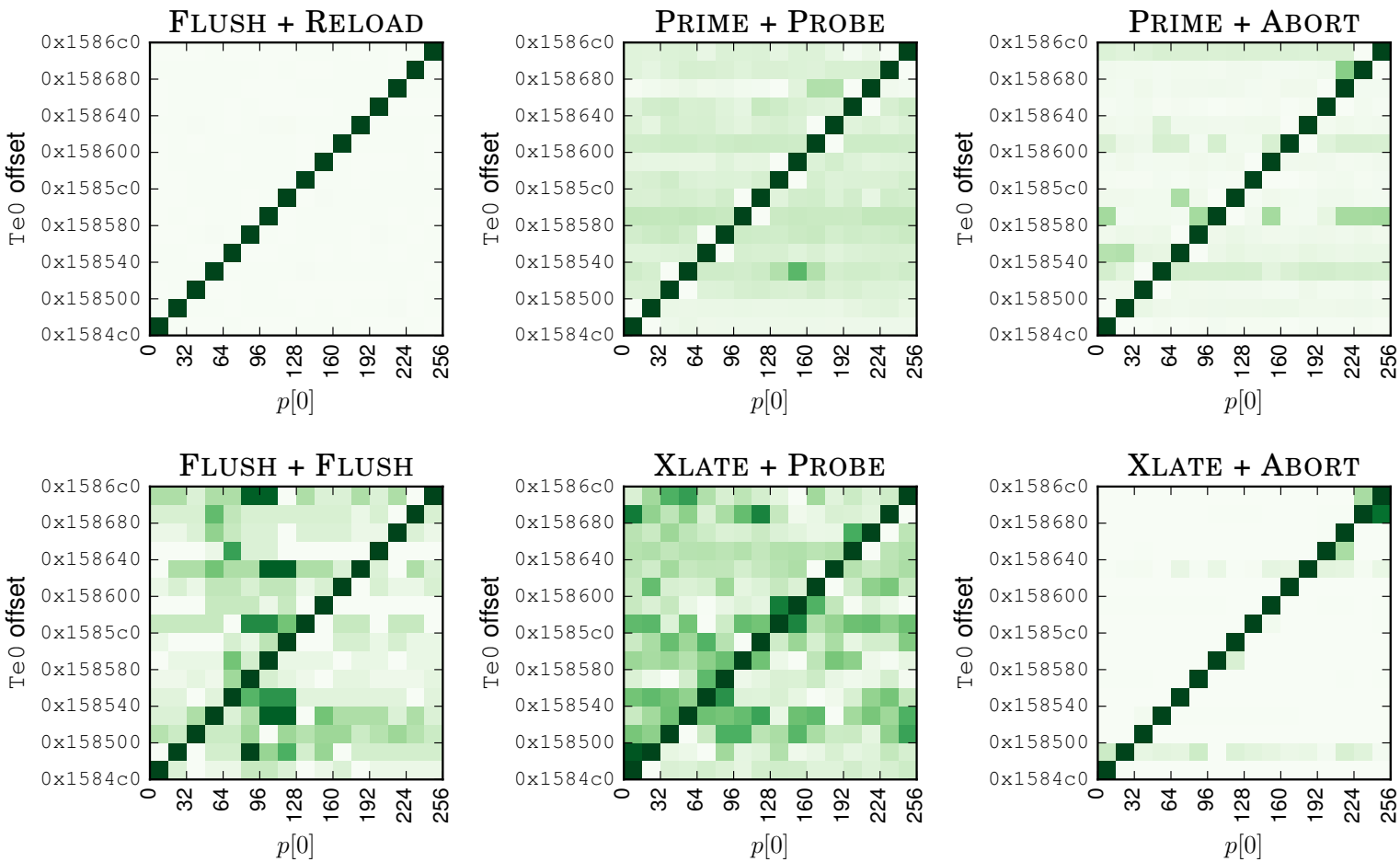


XLATE attacks are practical

Effectiveness



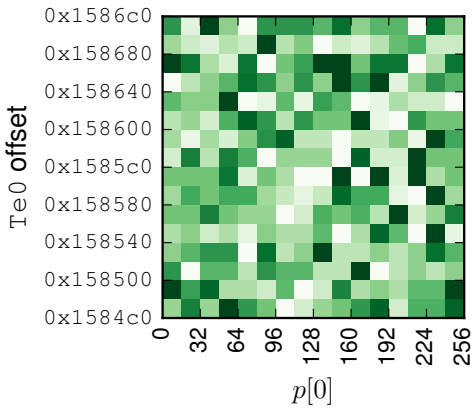
Effectiveness



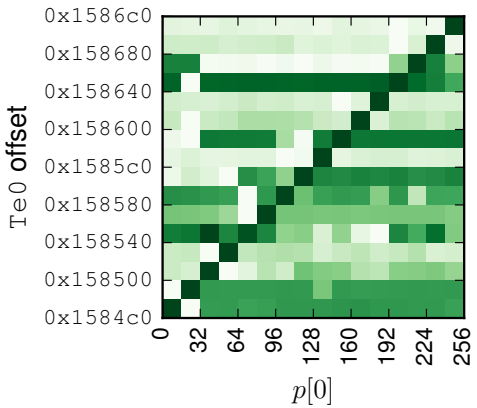
XLATE + PROBE is effective against AES T-tables

Cache Defenses

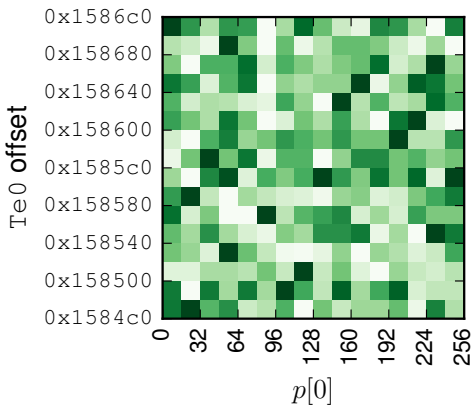
PRIME + PROBE (coloring)



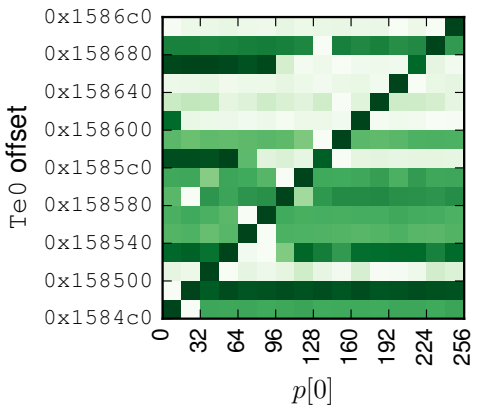
XLATE + PROBE (coloring)



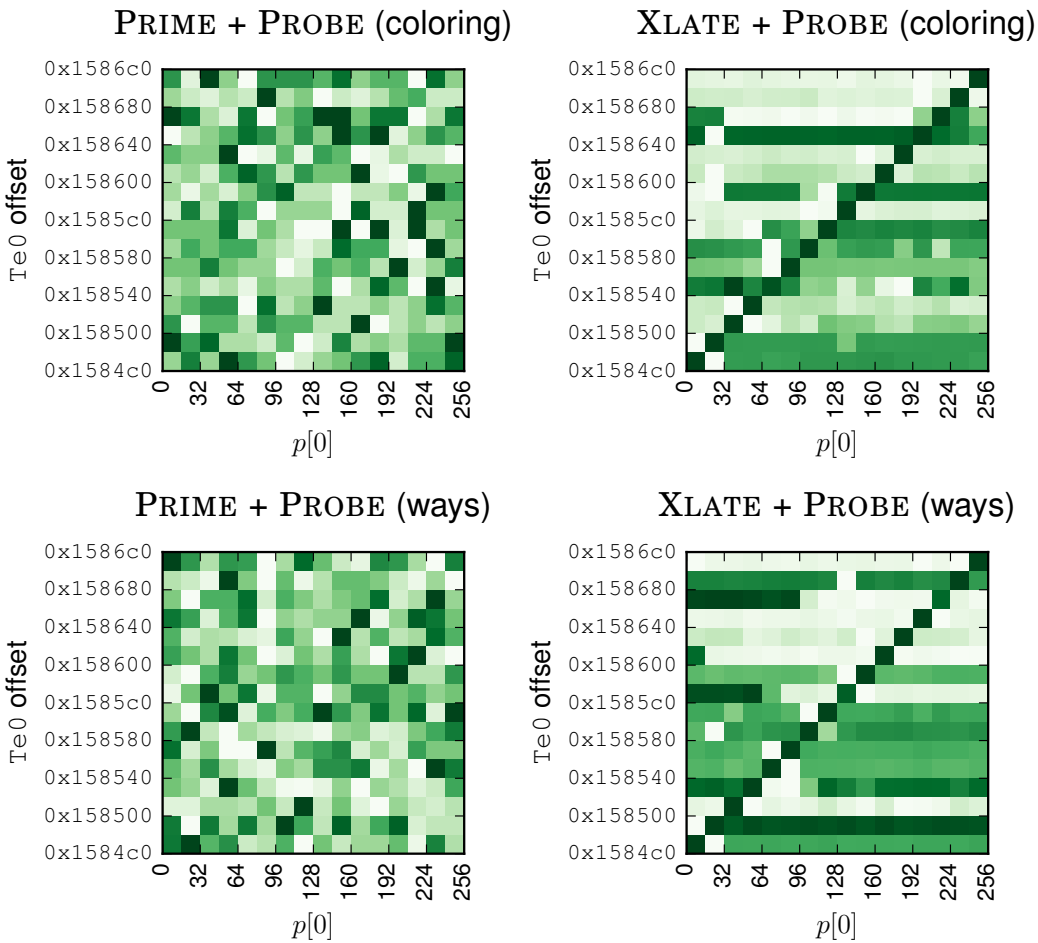
PRIME + PROBE (ways)



XLATE + PROBE (ways)



Cache Defenses



XLATE + PROBE bypasses set and way partitioning

Conclusions

- New family of cache attacks: XLATE
- Indirect cache attacks are practical
- Reconsider existing cache defenses
- `https://vusec.net/projects/xlate`