Cisco CRC - 45 minutes

ASLR^Cache: Practical Cache Attacks on the MMU

Ben Gras
Kaveh Razavi
Erik Bosman
Herbert Bos
Cristiano Giuffrida

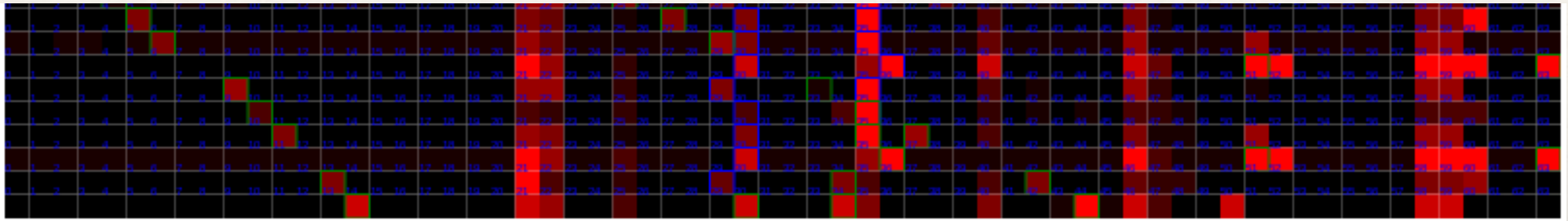
Vrije Universiteit Amsterdam

Teaser

- Compute virtual addresses of data & code
- With microarchitectural MMU side channel, not software
- Thereby breaking ASLR
- On all modern CPU models - Intel, AMD, ARM
- Even from JavaScript

Teaser

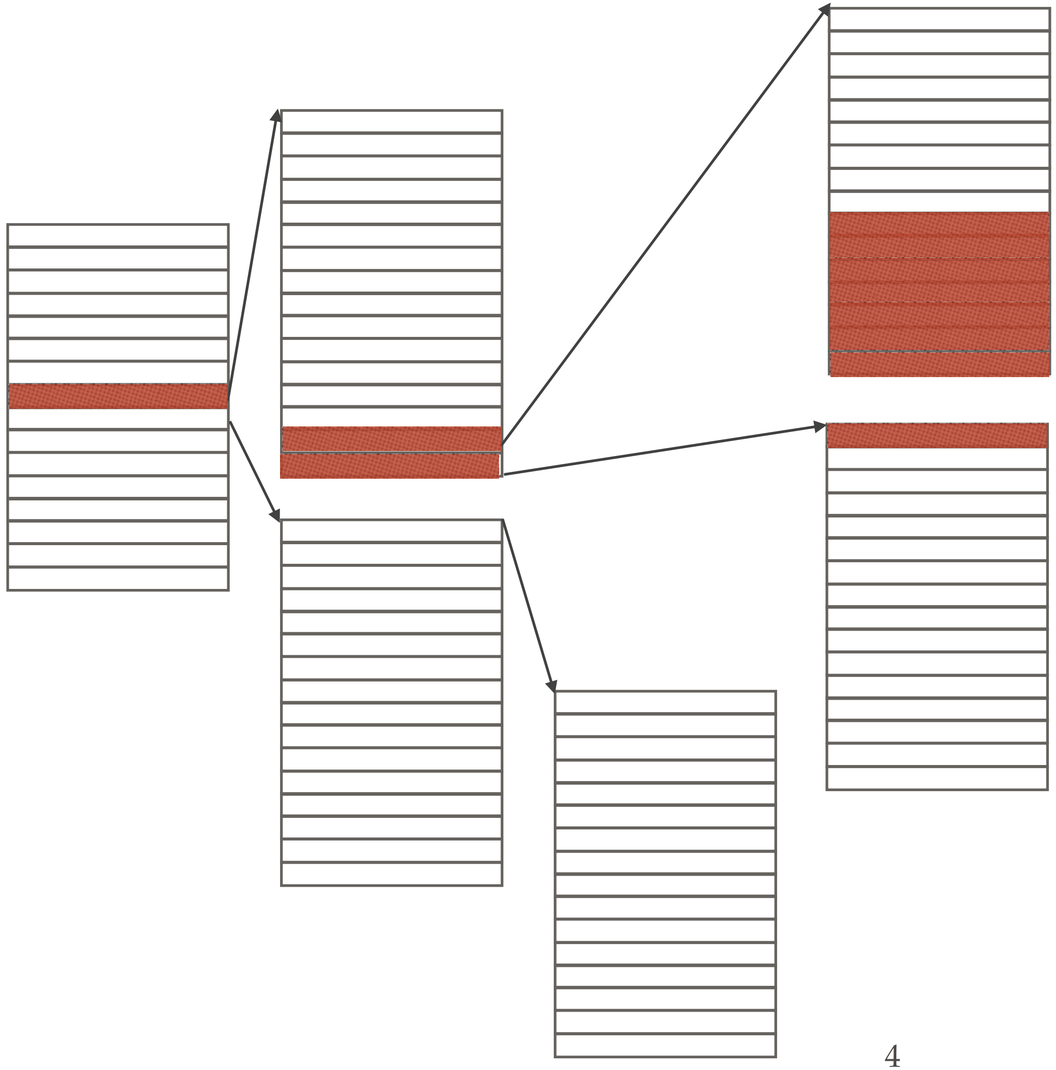
- Visualization - JavaScript - and no software bug

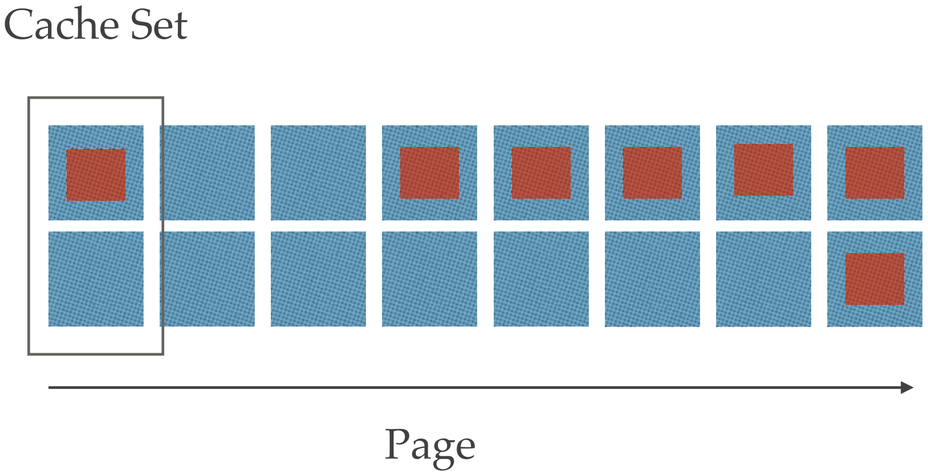


```
24.457 got level 4 - start slot 148, address 0x94000
24.993 got level 3 - start slot 295, address 0x24e94000
24.993 estimated remaining entropy 6 slot solutions: -1,-1,295,148
68.737 got level 4 - start slot 0, address 0x0
69.502 got level 3 - start slot 359, address 0x2ce00000
70.259 got level 2 - start slot 411, address 0x66ece00000
88.041 got level 1 - start slot 238, address 0x7766ece00000
88.041 estimated remaining entropy 0 slot solutions: 238,411,359,0
data: 0x7766ece00000, code slots: -1,-1,295,148, code: 0x7966e4e94000
```

- There will be videos and other refreshments

Big picture: cached page tables





Outline

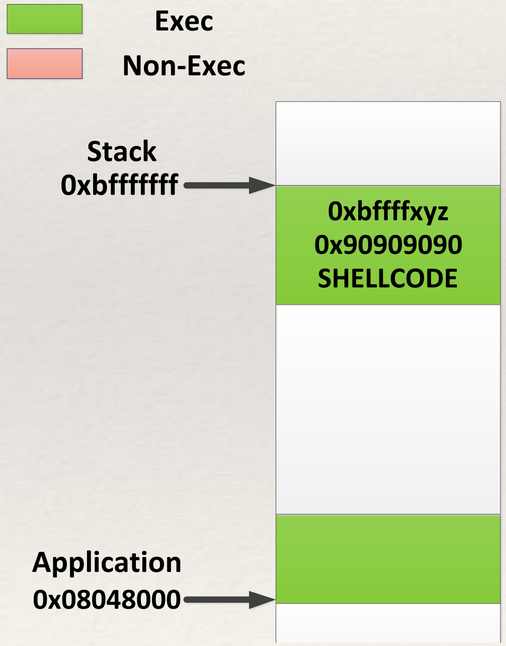
- Justification: ASLR
- Side Channels
- Pagetable walks
- CPU Caches
- EVICT+TIME
- JavaScript
- Results
- Demo

Outline

- **Justification: ASLR**
- Side Channels
- Pagetable walks
- CPU Caches
- EVICT+TIME
- JavaScript
- Results
- Demo

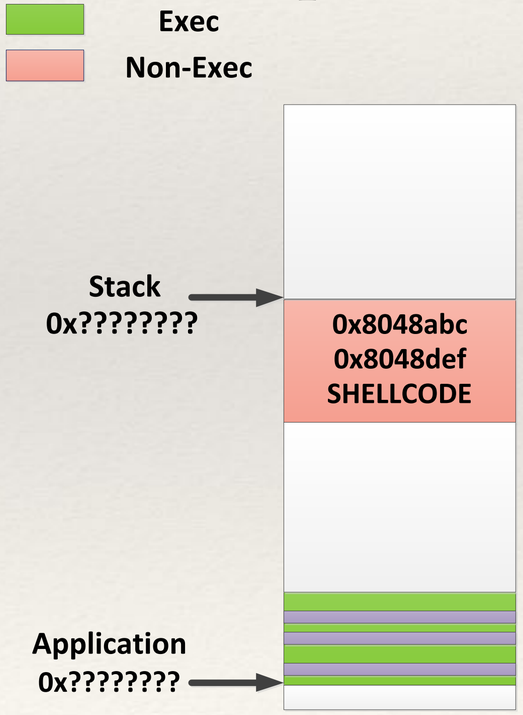
ASLR

- Main justification
- Response to exploitation in the 90s



ASLR

- Let's randomize both areas: ASLR
- Also DEP. So exploitation requires ROP+ASLR bypass

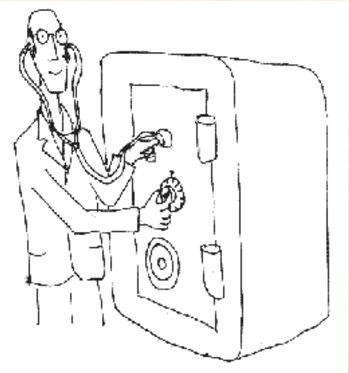


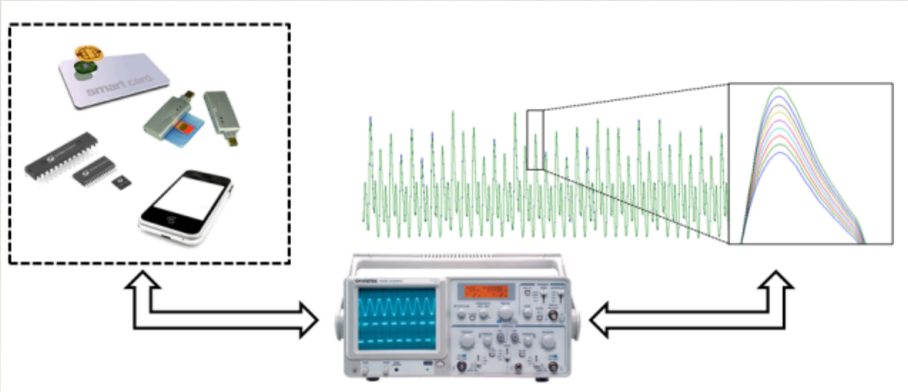
Outline

- Justification: ASLR
- **Side Channels**
- Pagetable walks
- CPU Caches
- EVICT+TIME
- JavaScript
- Results
- Demo

Side Channels

- Get secrets by measuring out of the box
- Side effect outside the system
- e.g. Stethoscopes
- e.g. Power analysis
- e.g. Timing
- e.g. RF

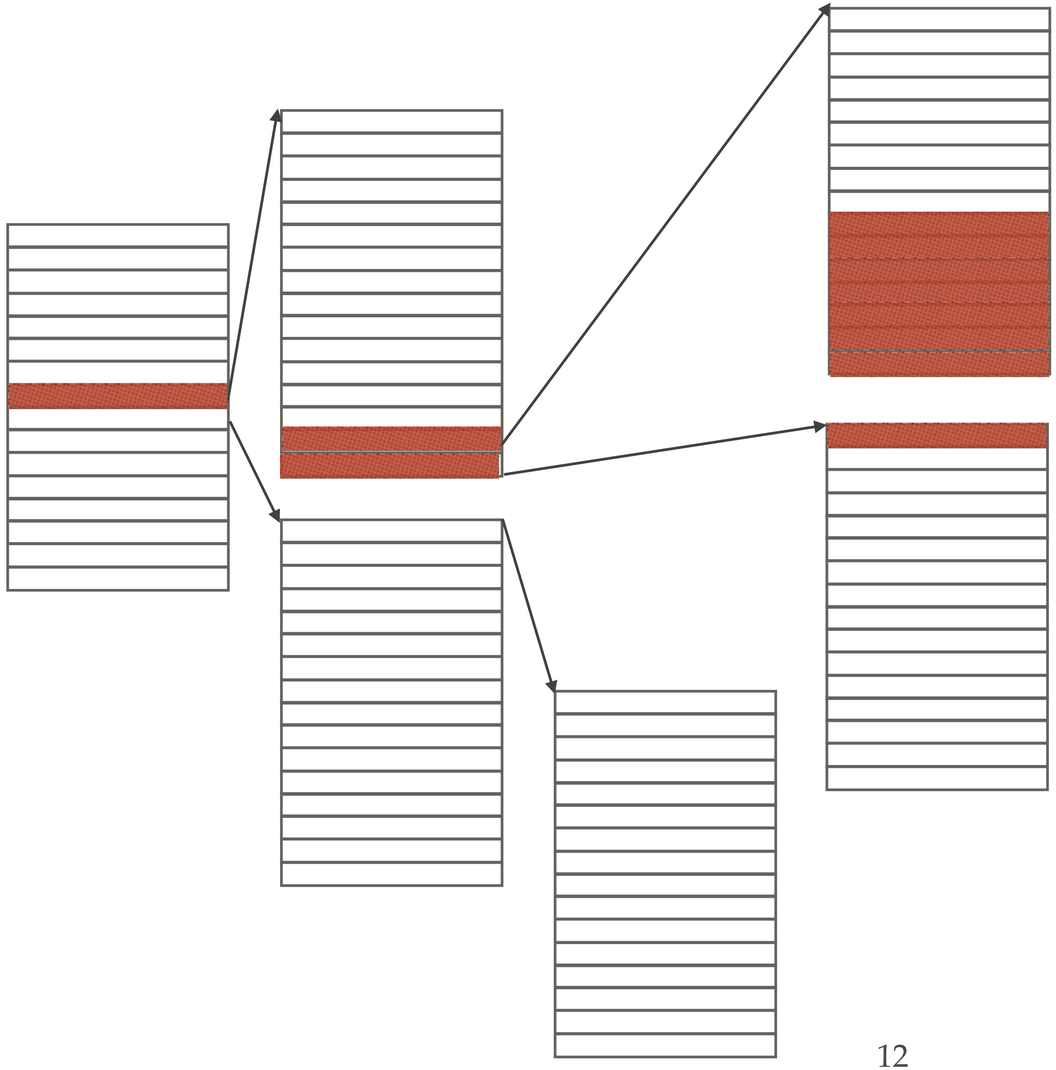


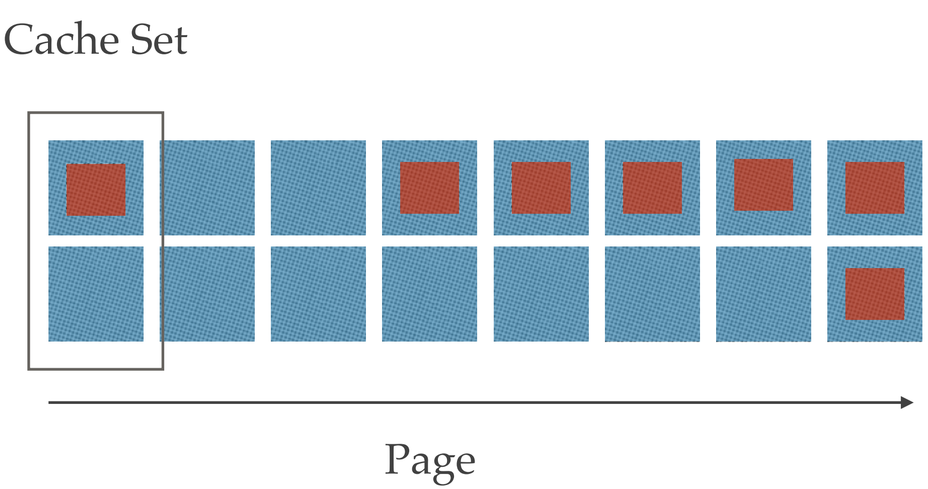


Outline

- Justification: ASLR
- Side Channels
- **Pagetable walks**
- CPU Caches
- EVICT+TIME
- JavaScript
- Results
- Demo

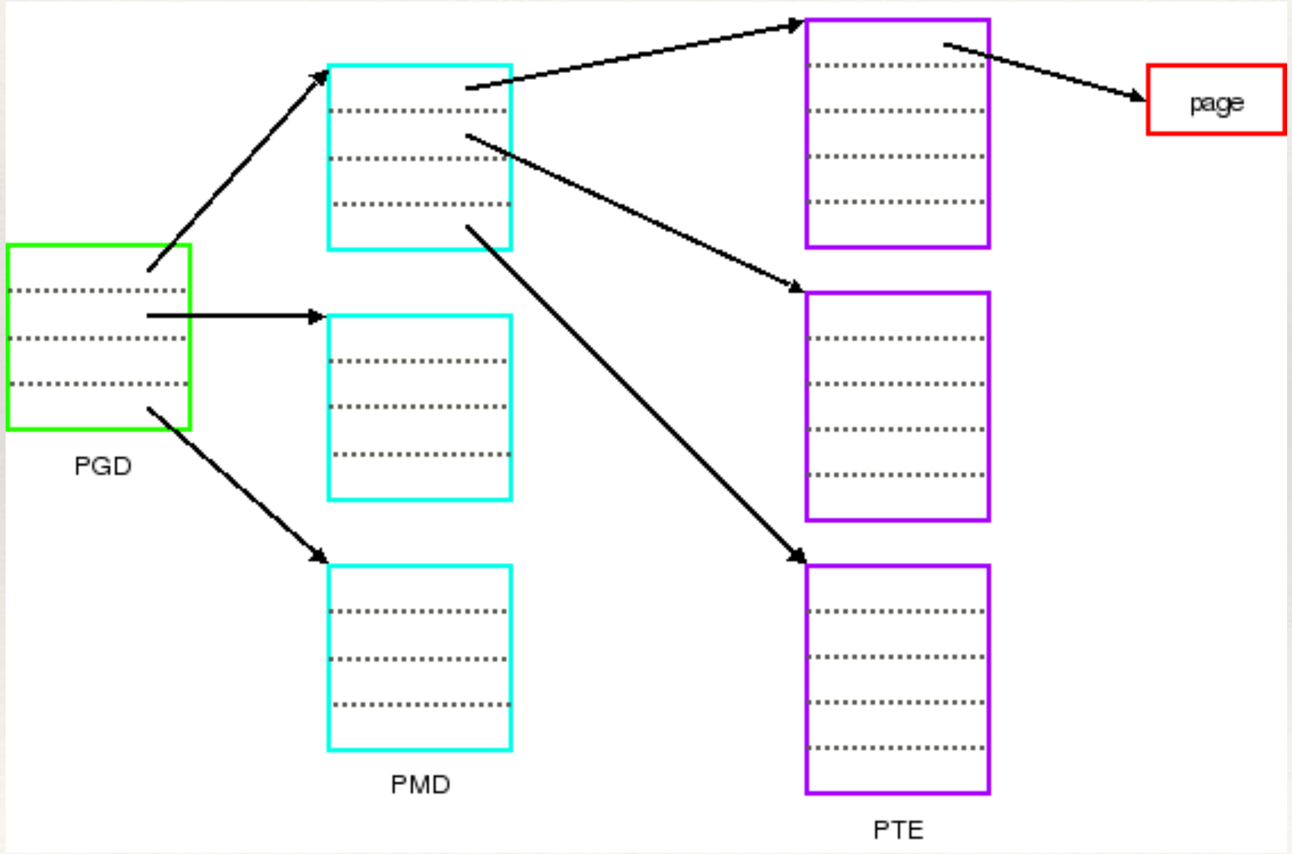
Big picture: cached page tables





Pagetable Walks From DRAM

- Page tables point to the next step in a tree



Pagetable Walks From DRAM

0x644b321f4000

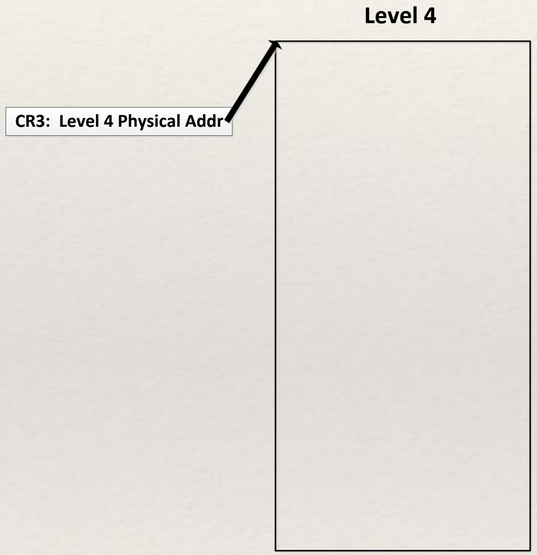
110010001001011001100100001111101000000000000000

CR3: Level 4 Physical Addr

Pagetable Walks From DRAM

0x644b321f4000

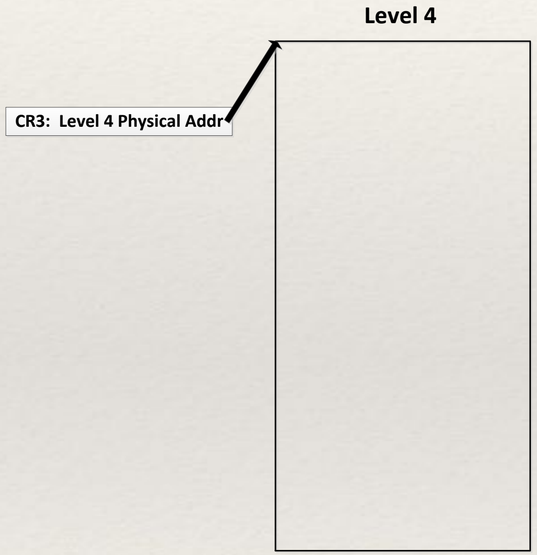
110010001001011001100100001111101000000000000000



Pagetable Walks From DRAM

0x644b321f4000

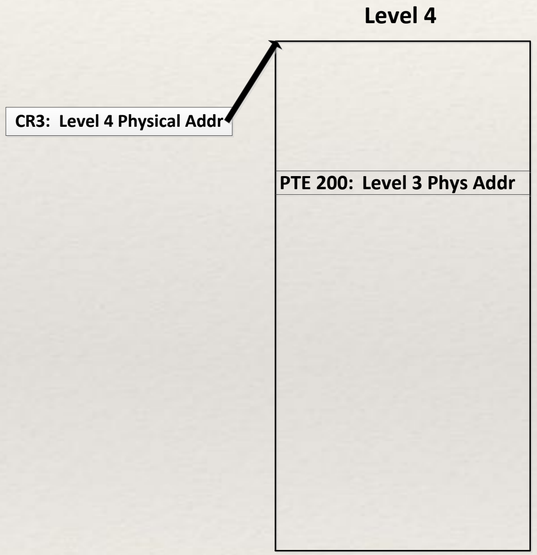
110010001001011001100100001111101000000000000000



Pagetable Walks From DRAM

0x644b321f4000

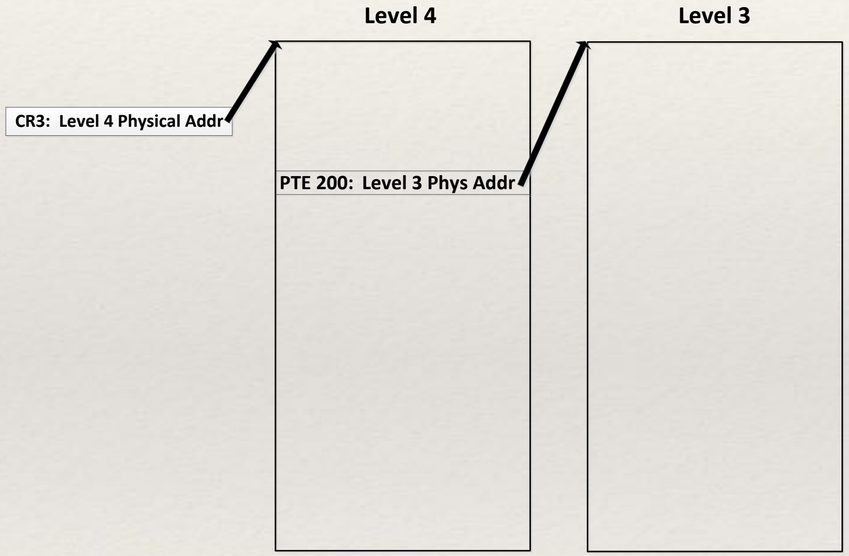
110010001001011001100100001111101000000000000000



Pagetable Walks From DRAM

0x644b321f4000

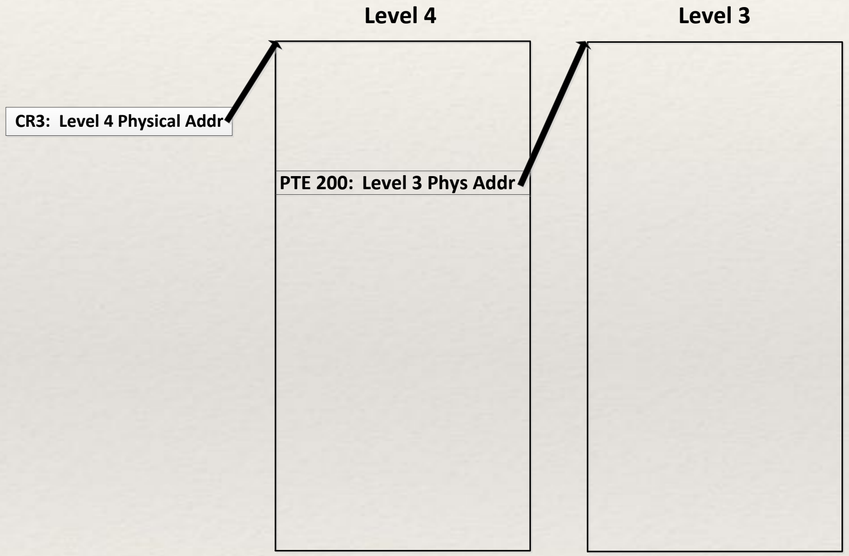
110010001001011001100100001111101000000000000000



Pagetable Walks From DRAM

0x644b321f4000

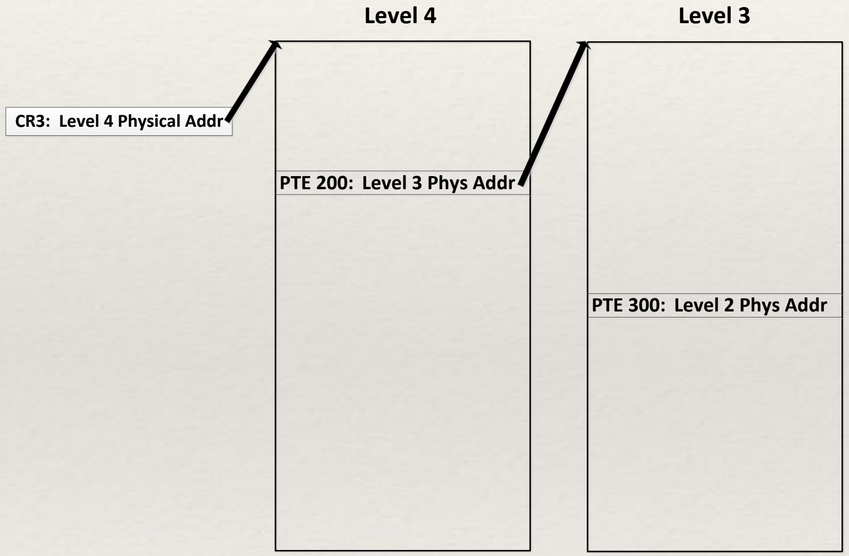
1100100010010110011001000011111010000000000000



Pagetable Walks From DRAM

0x644b321f4000

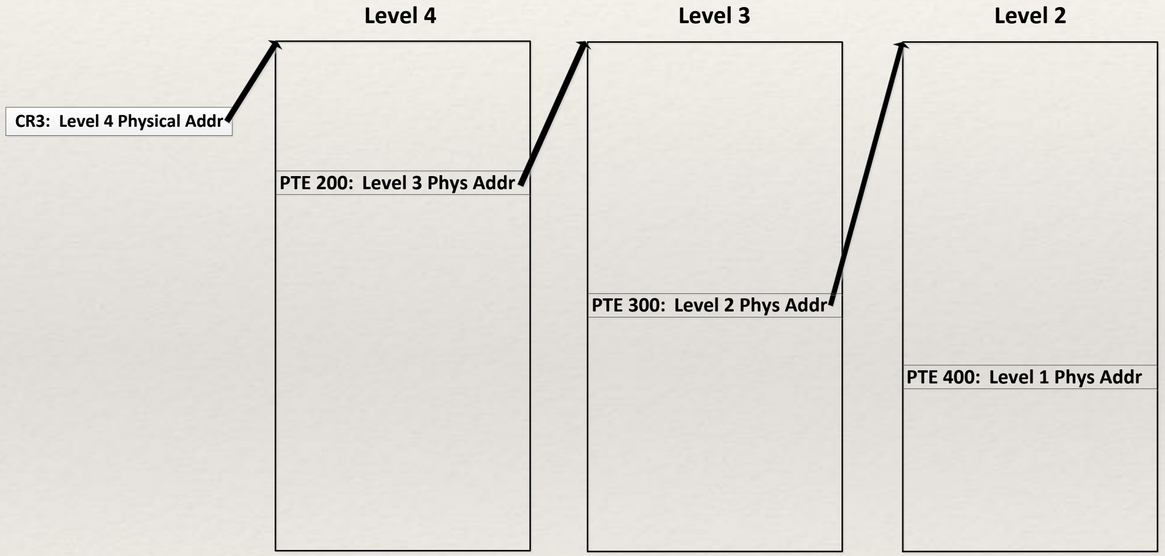
1100100010010110011001000011111010000000000000



Pagetable Walks From DRAM

0x644b321f4000

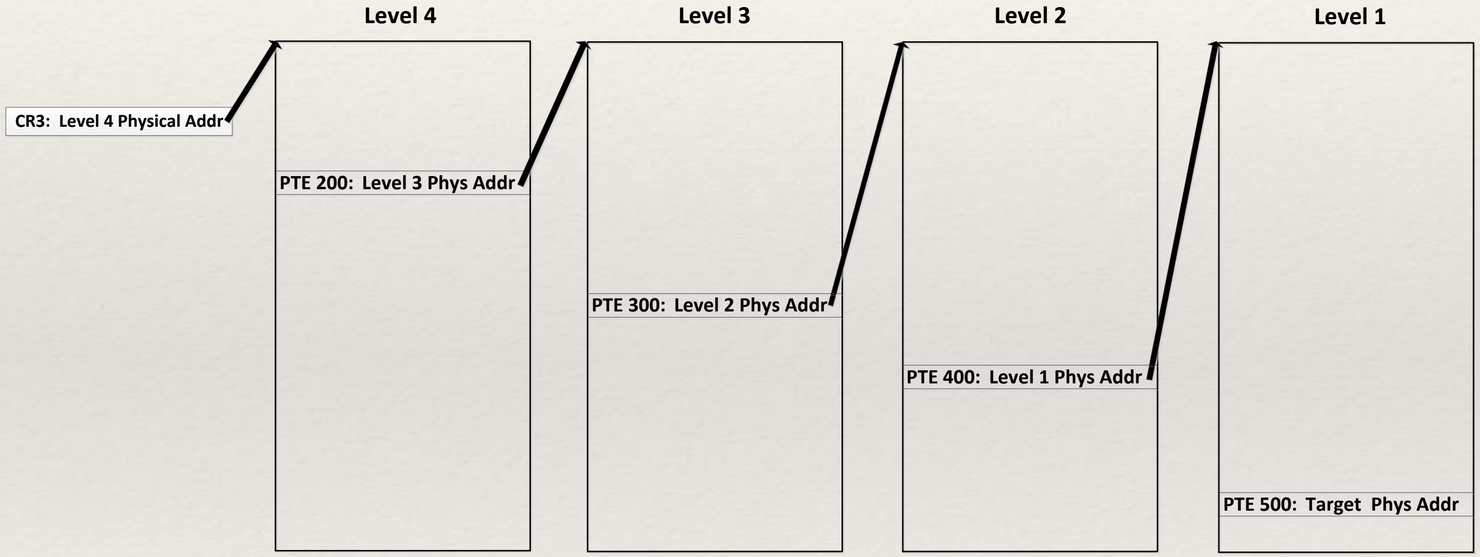
110010001001011001100100001111101000000000000000



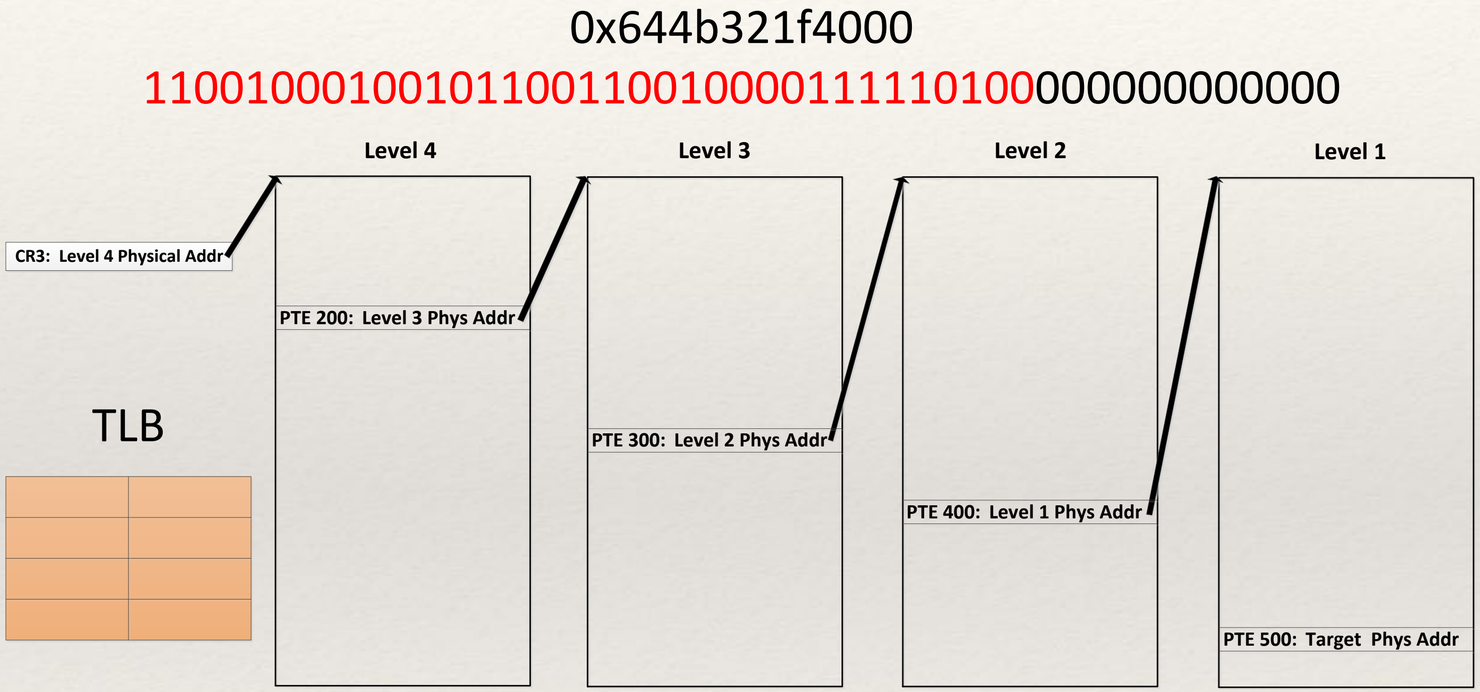
Pagetable Walks From DRAM

0x644b321f4000

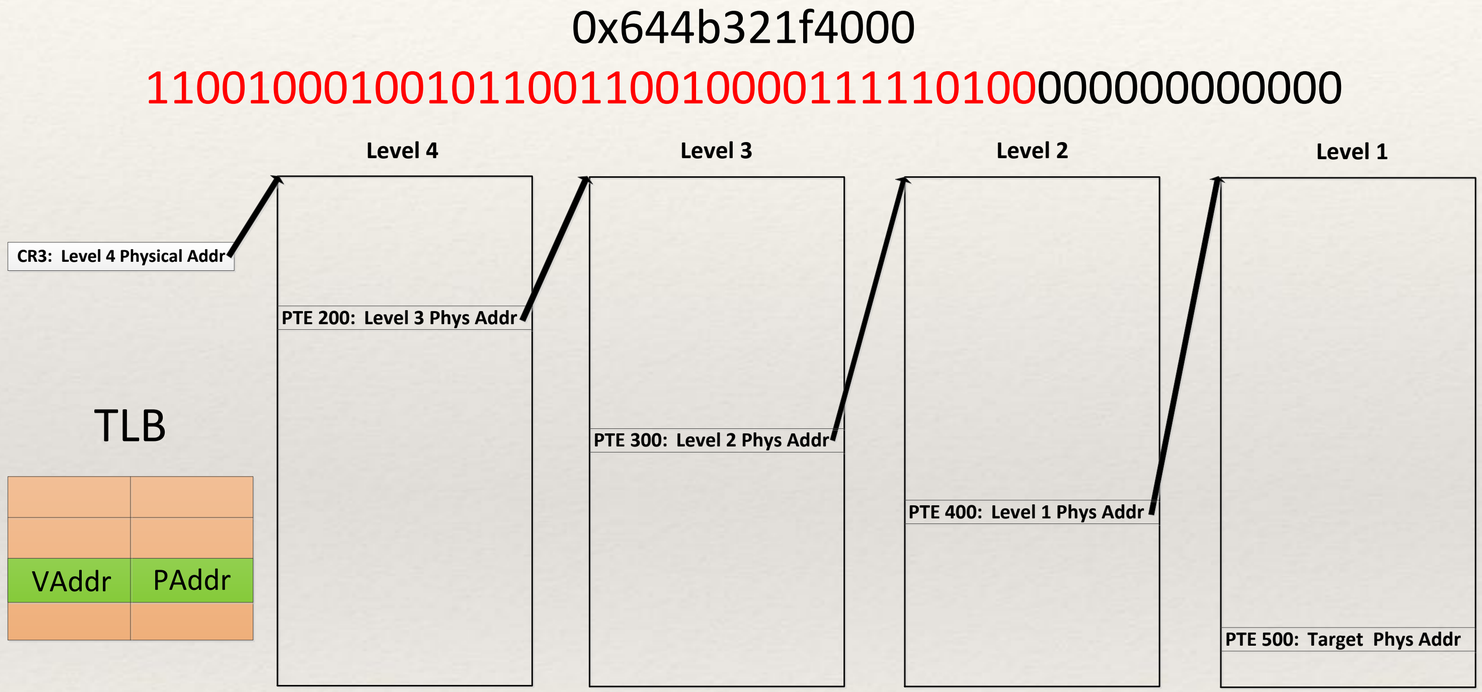
110010001001011001100100001111101000000000000



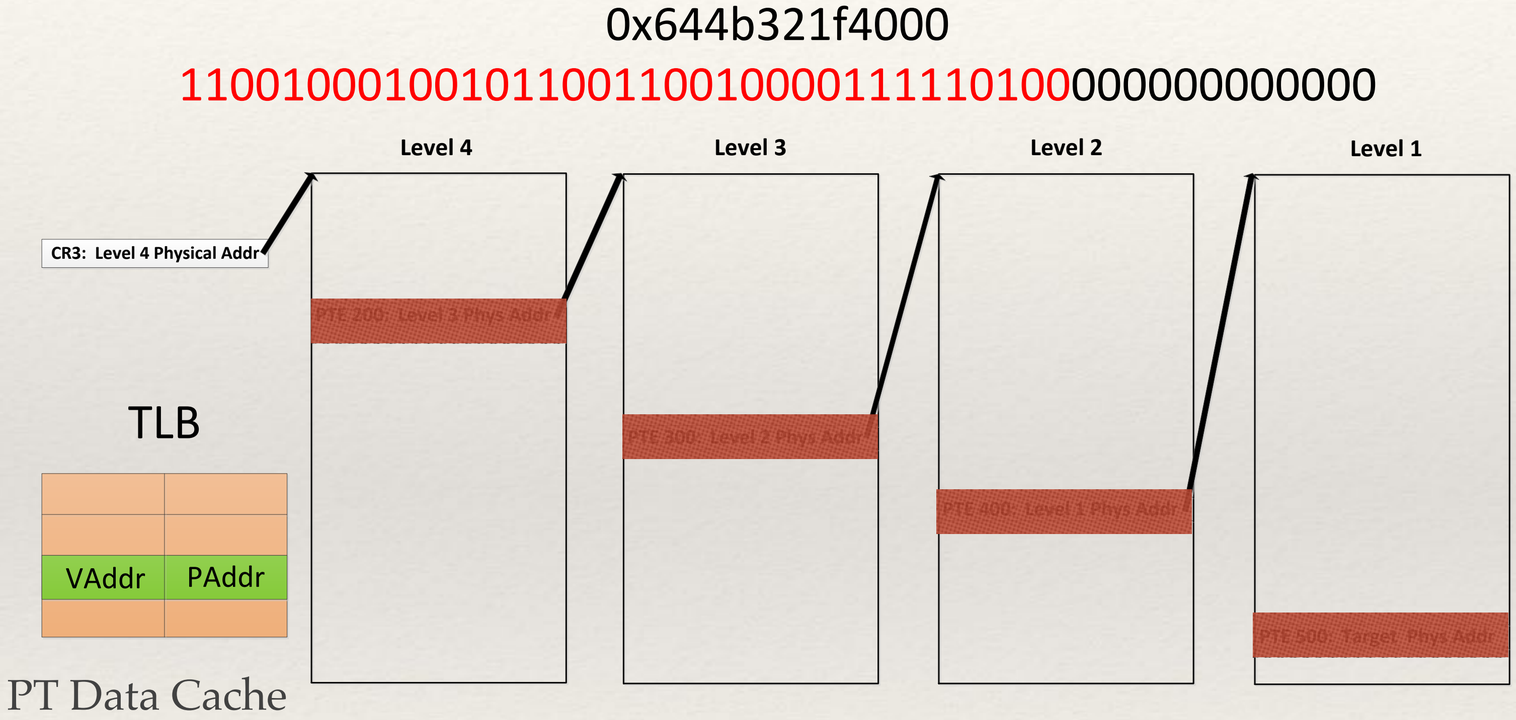
Pagetable Walks From DRAM



Pagetable Walks From DRAM



Pagetable Walks From DRAM

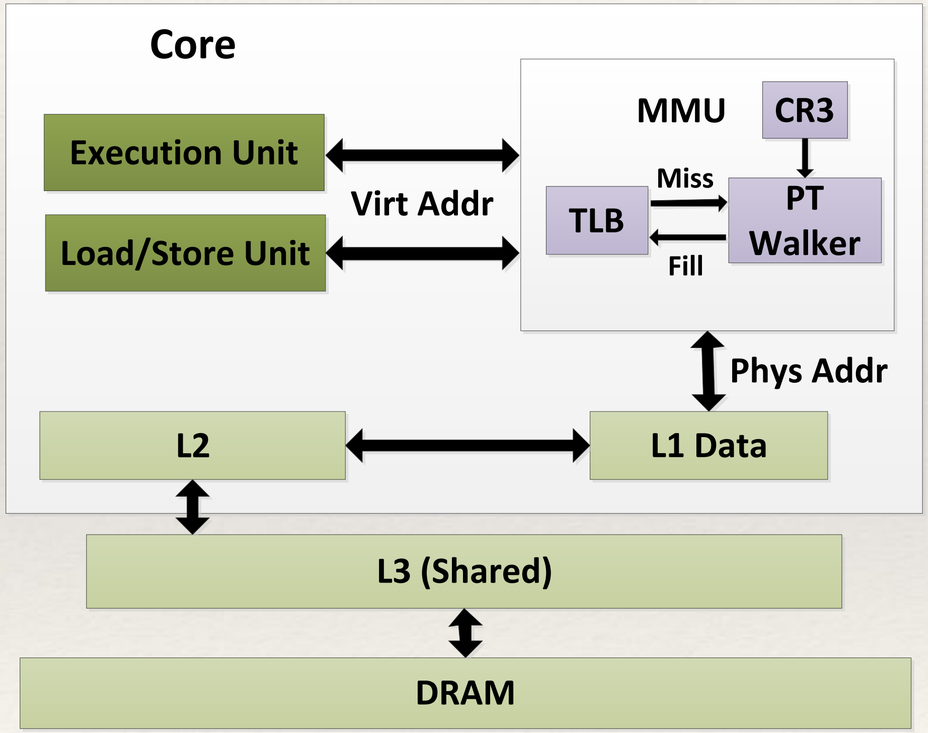


Outline

- Justification: ASLR
- Side Channels
- Pagetable walks
- **CPU Caches**
- EVICT+TIME
- JavaScript
- Results
- Demo

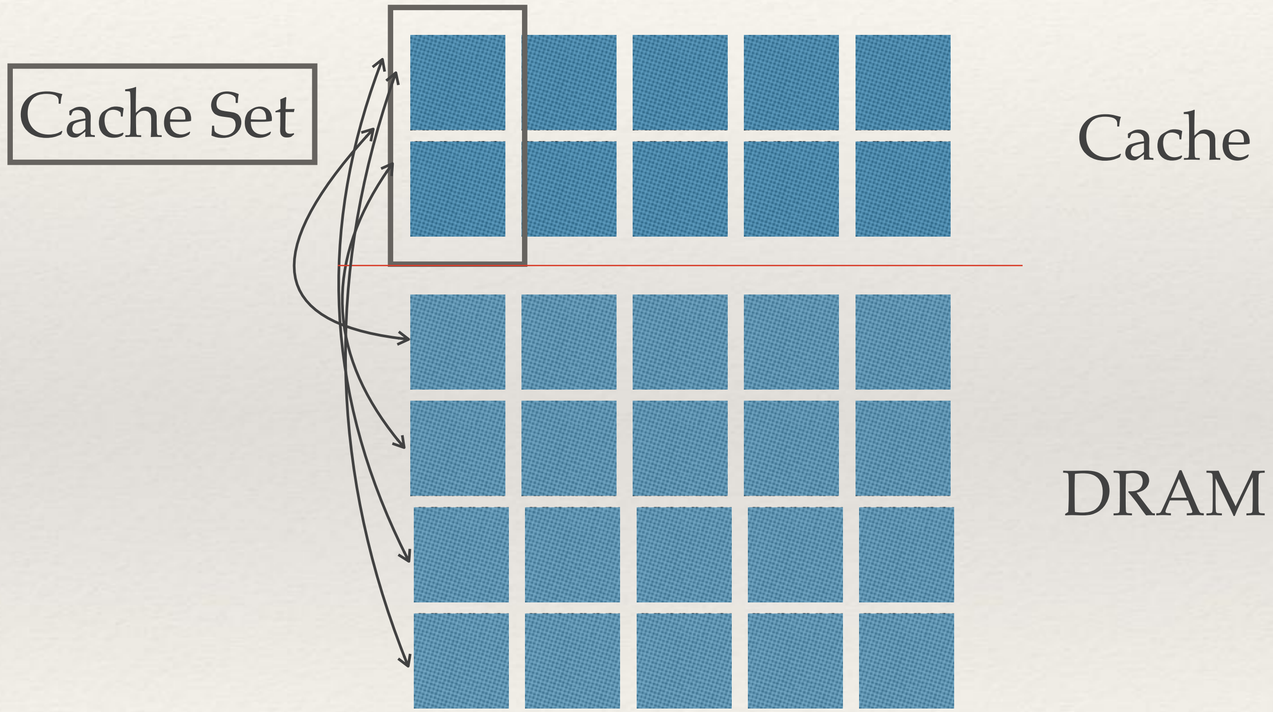
CPU Caches

- Logical layout of a CPU core and connection to DRAM



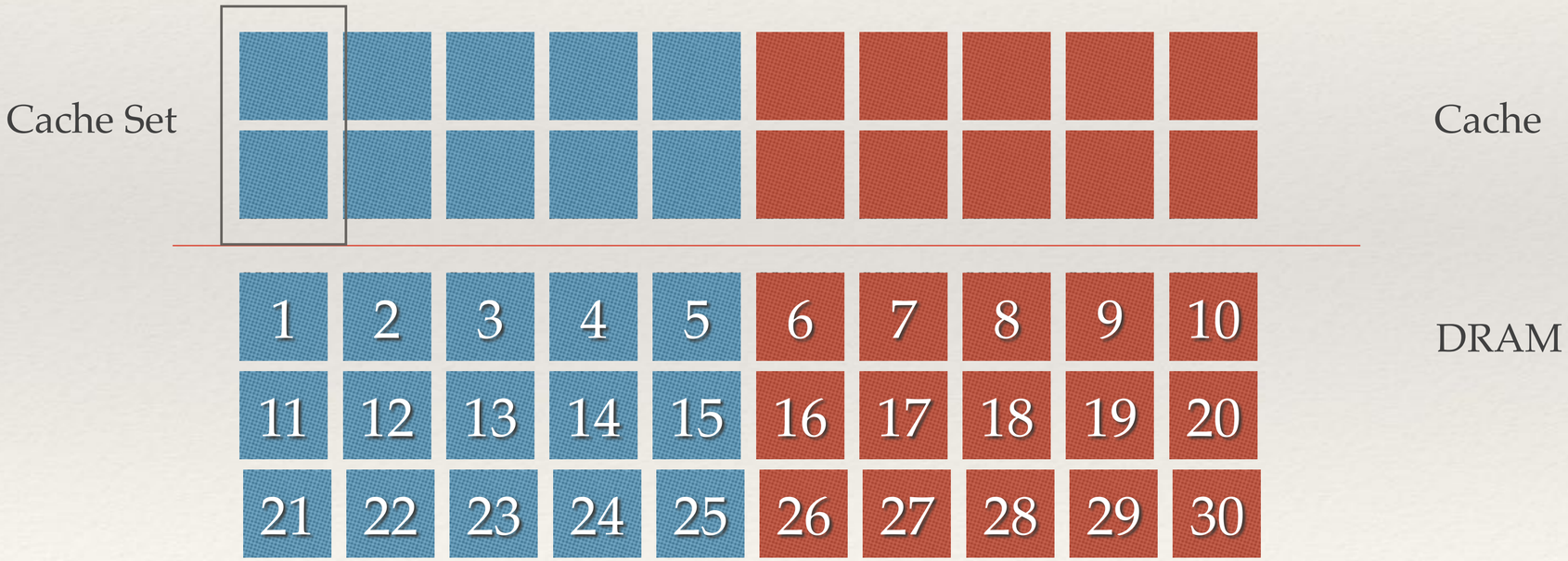
CPU Caches

- Memory cache lines can only go into one small cache set



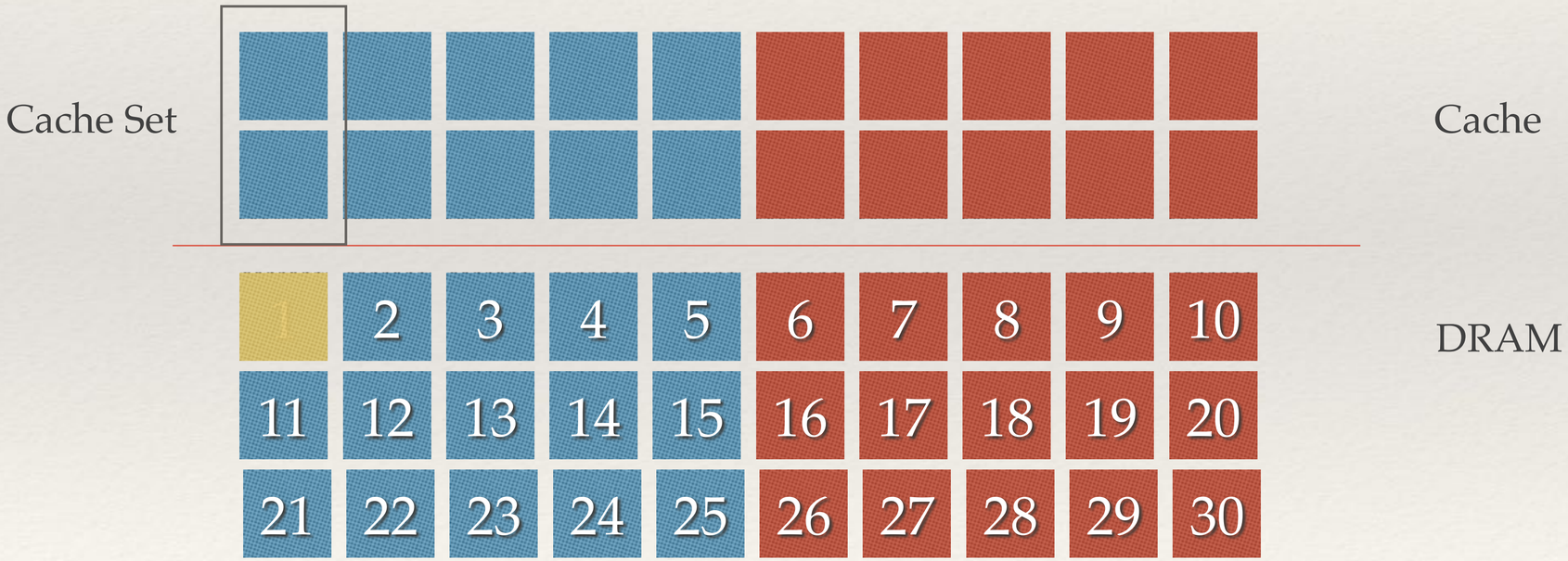
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



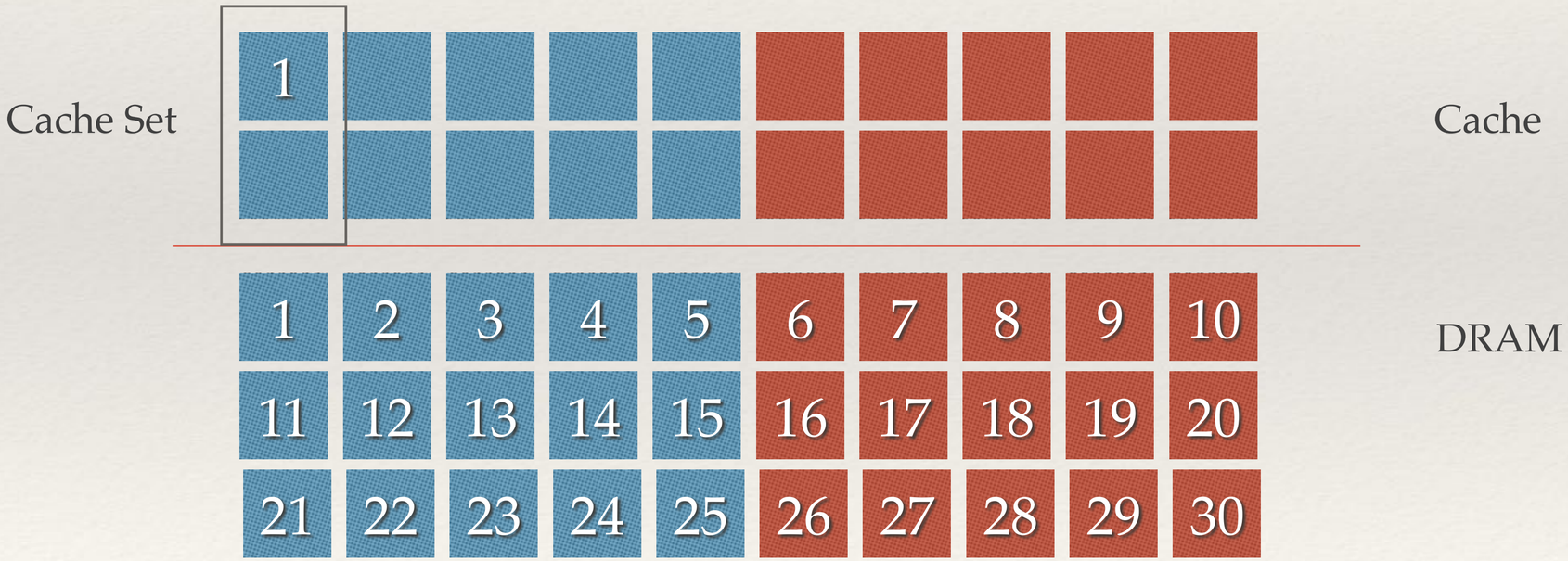
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



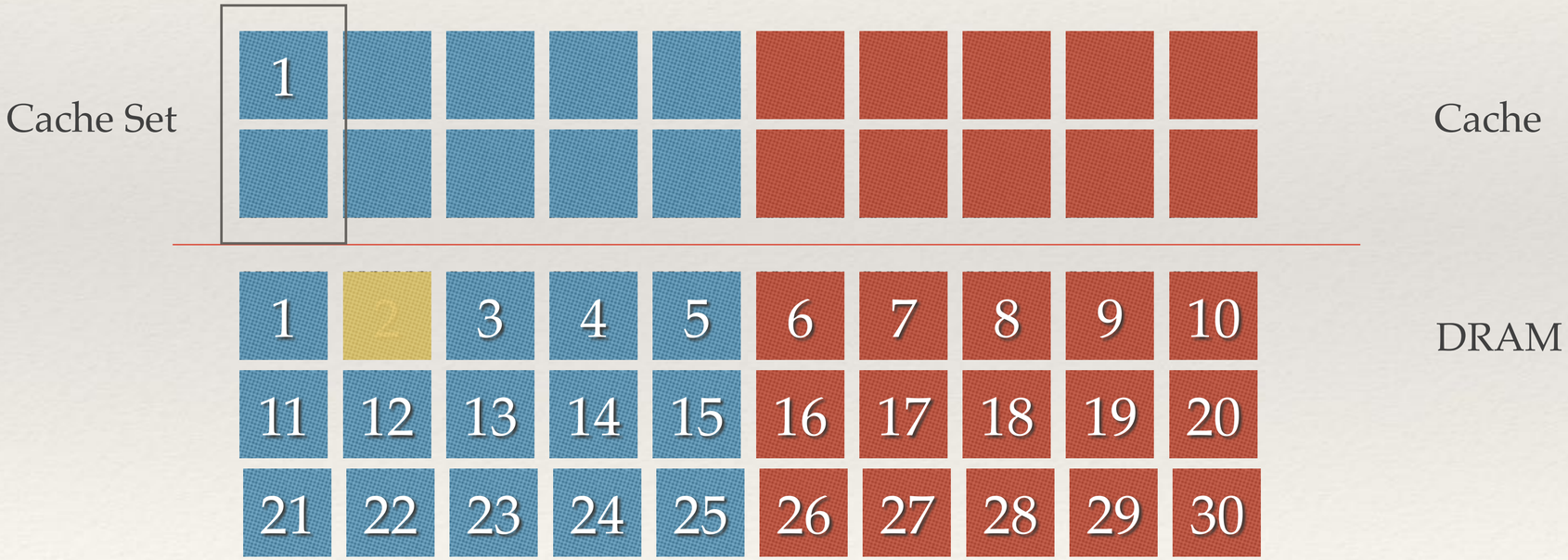
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



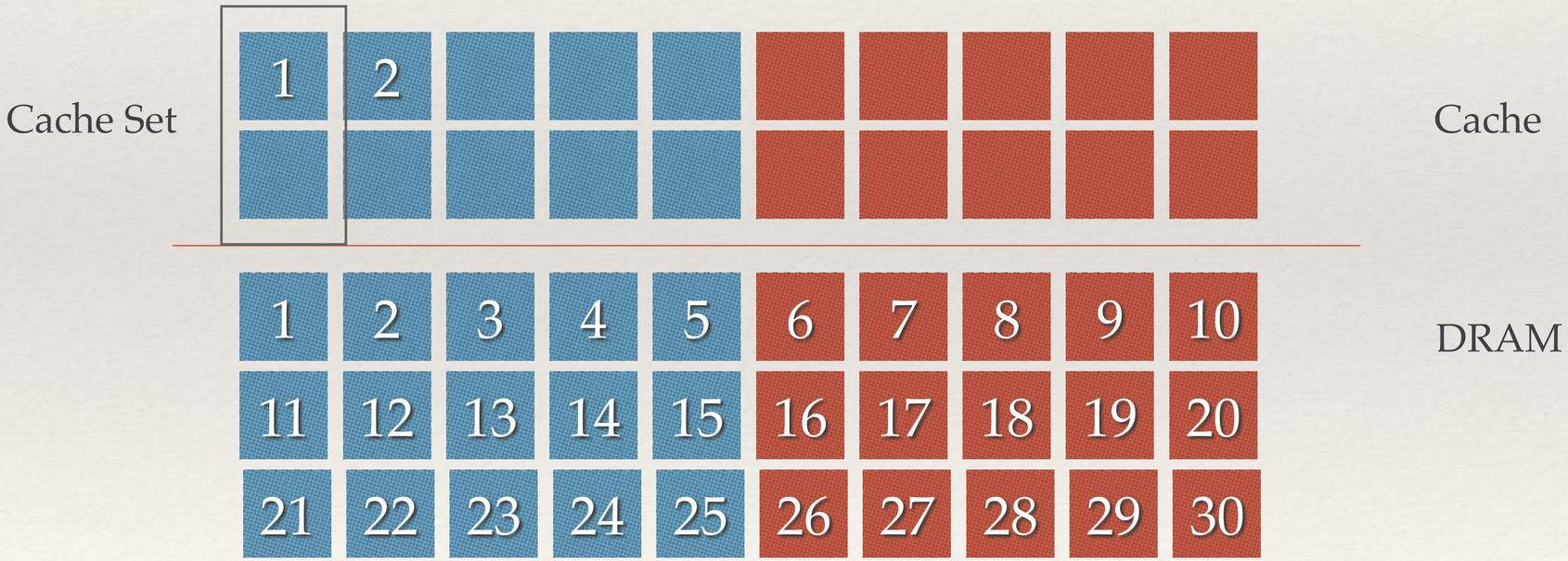
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



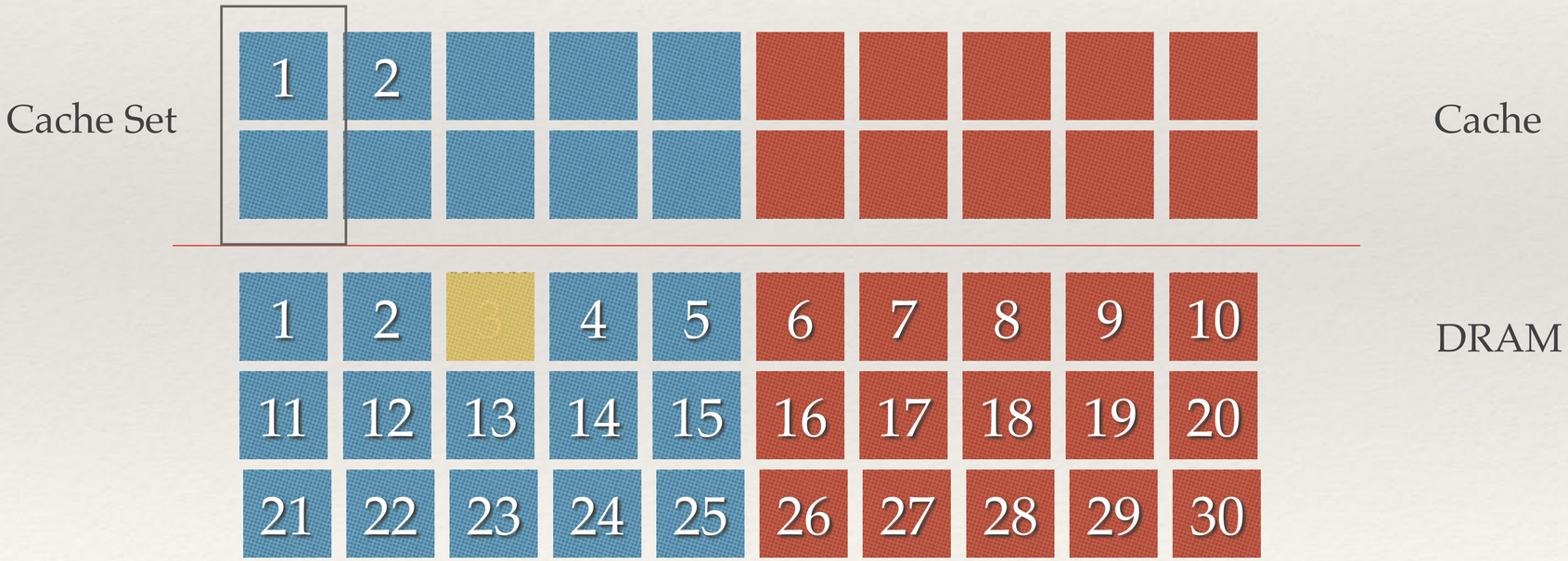
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



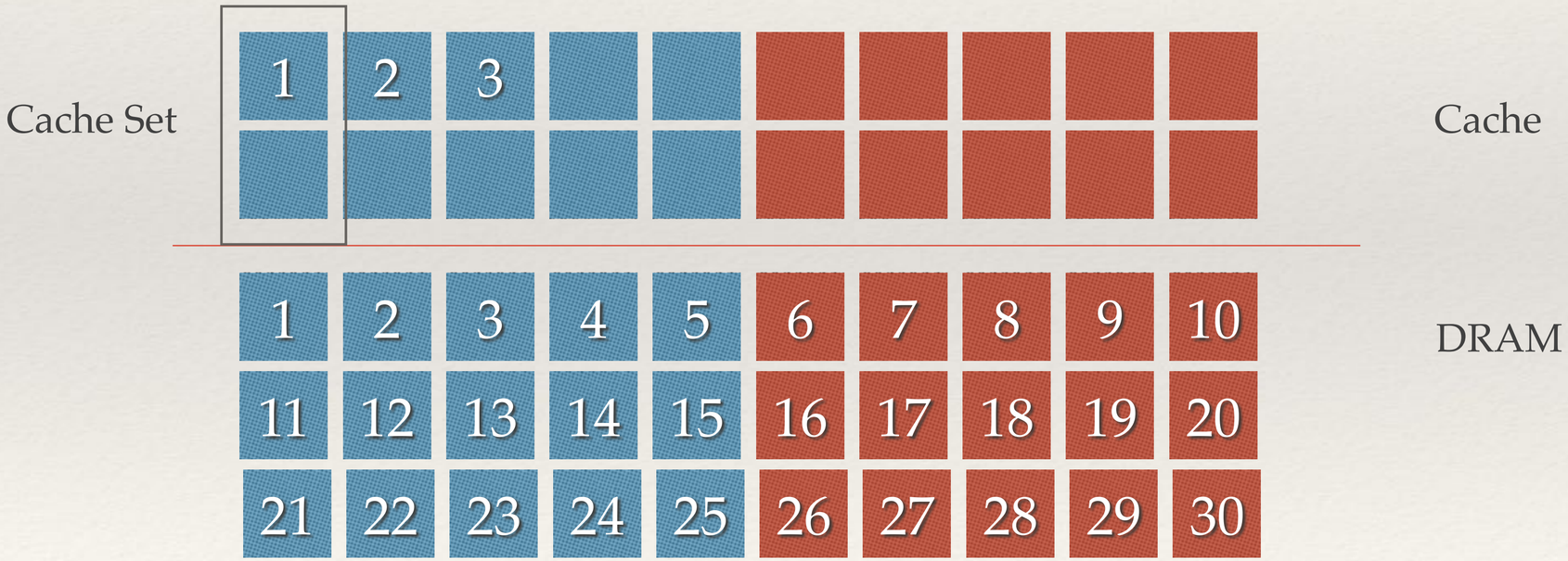
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



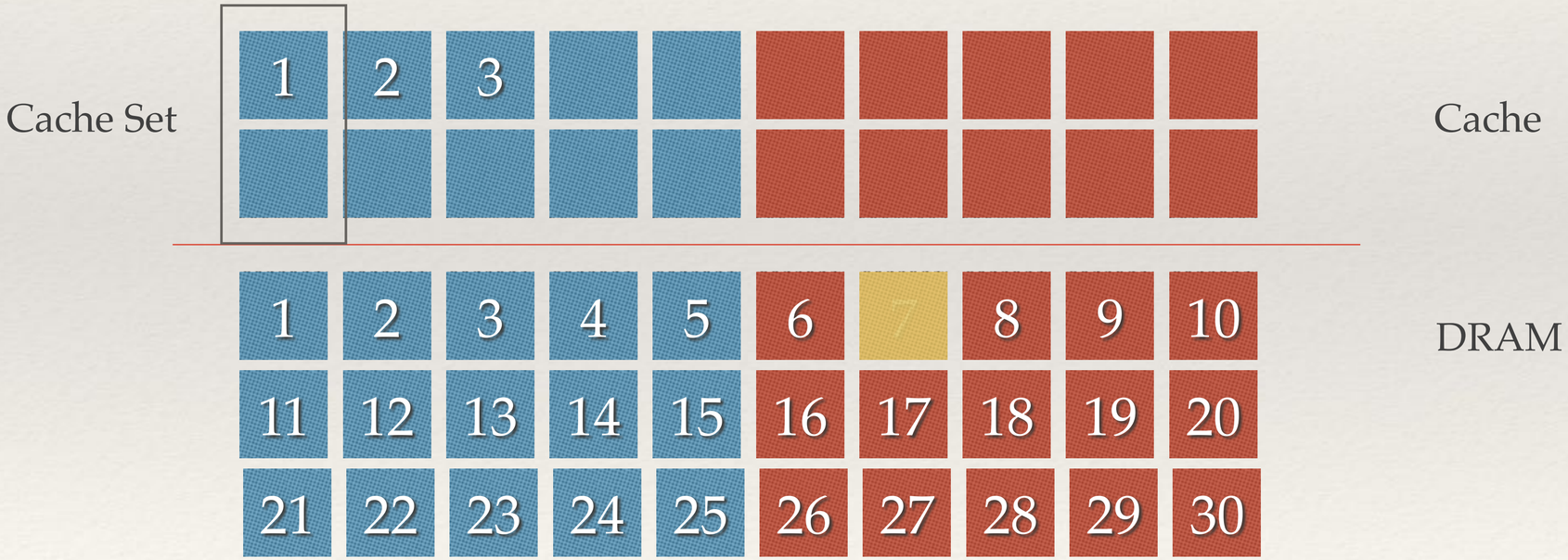
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



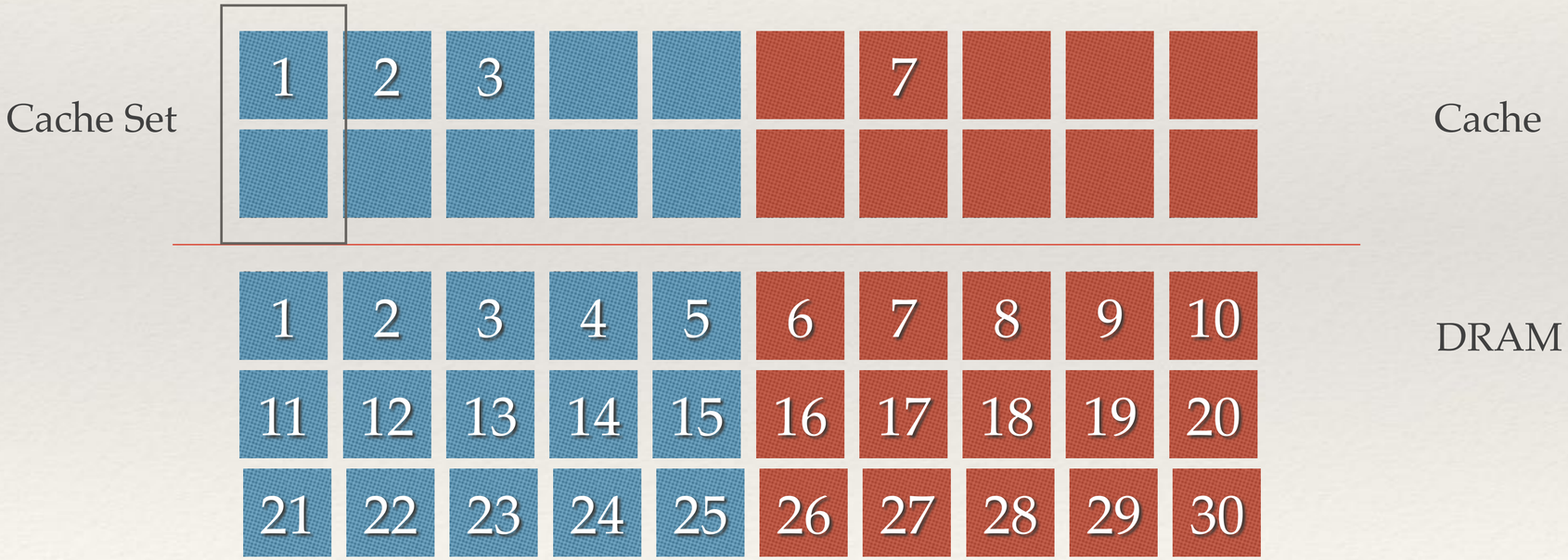
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



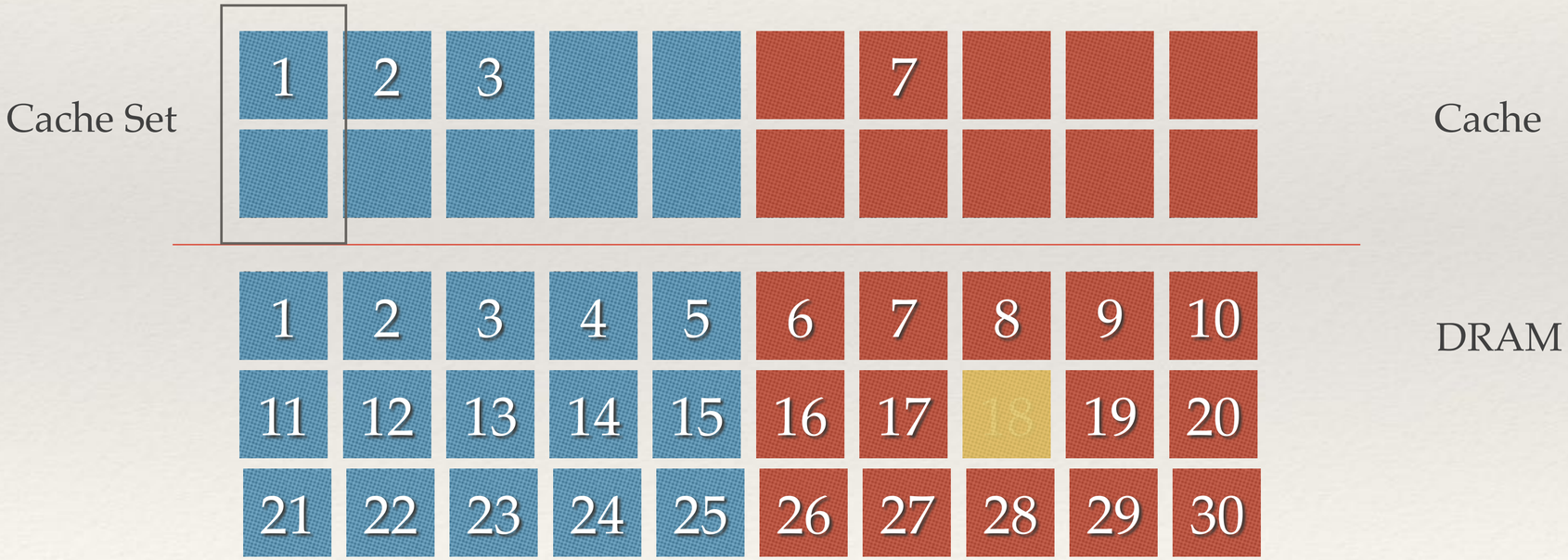
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



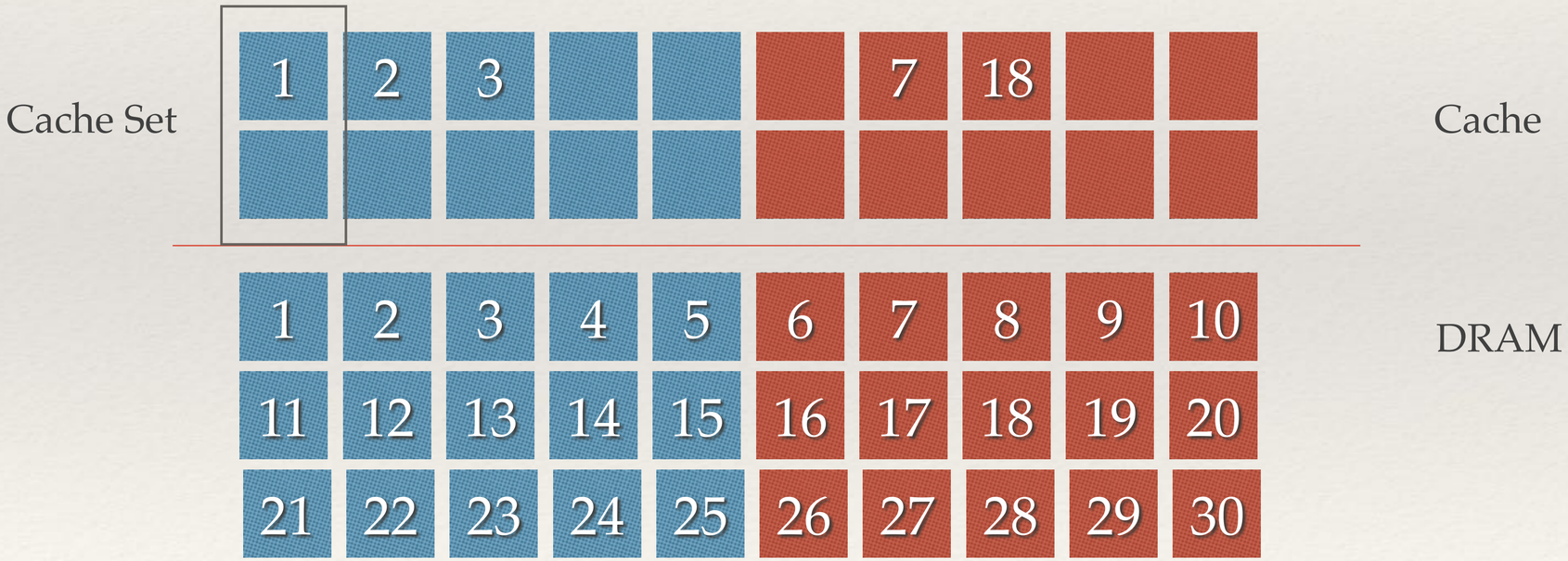
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



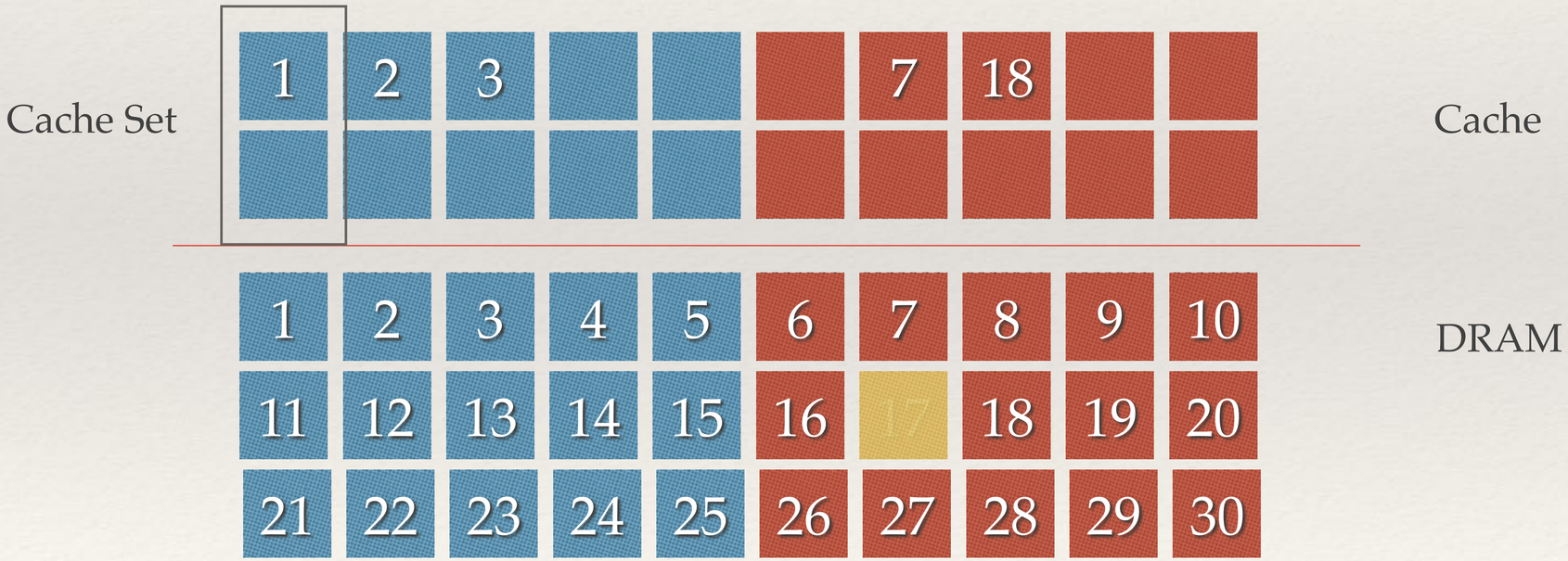
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



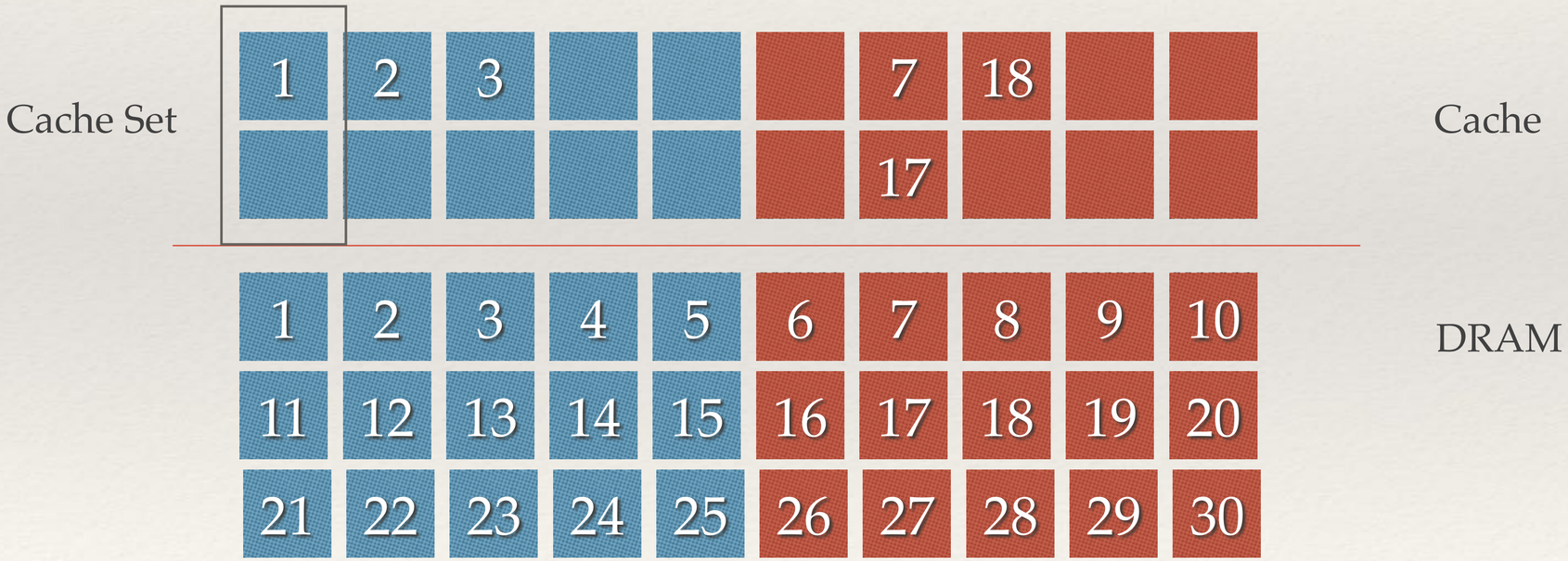
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



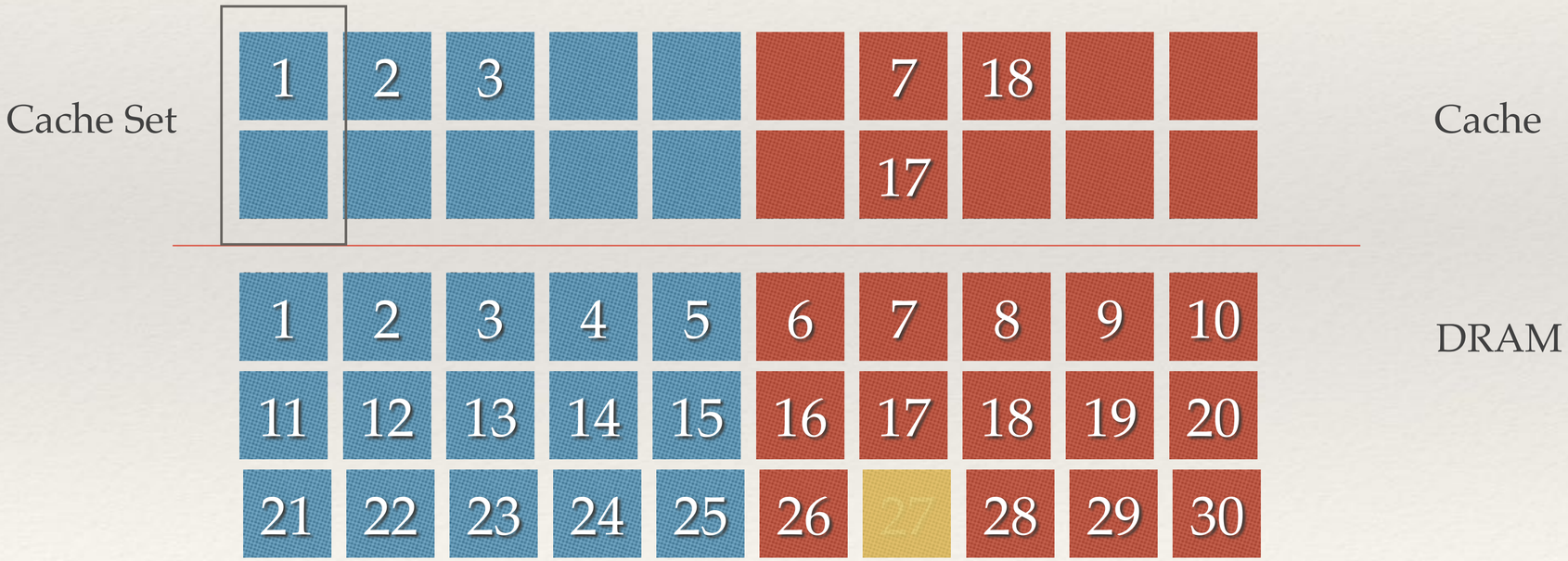
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



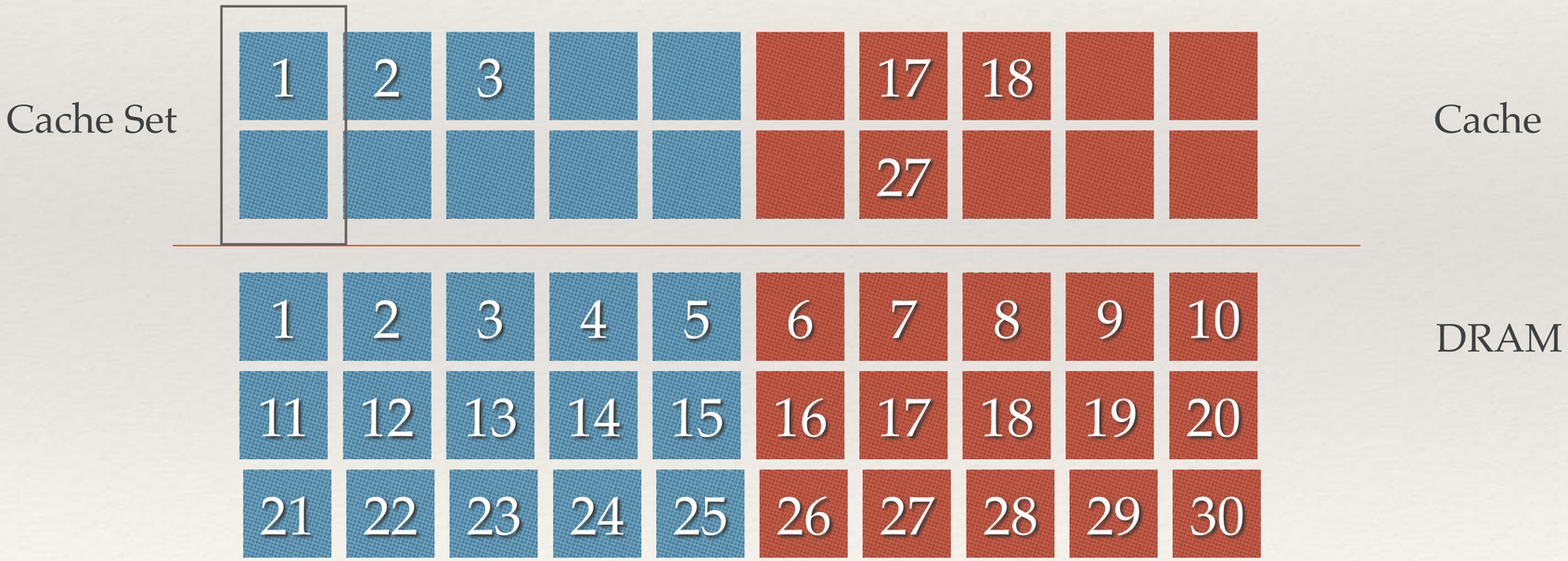
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



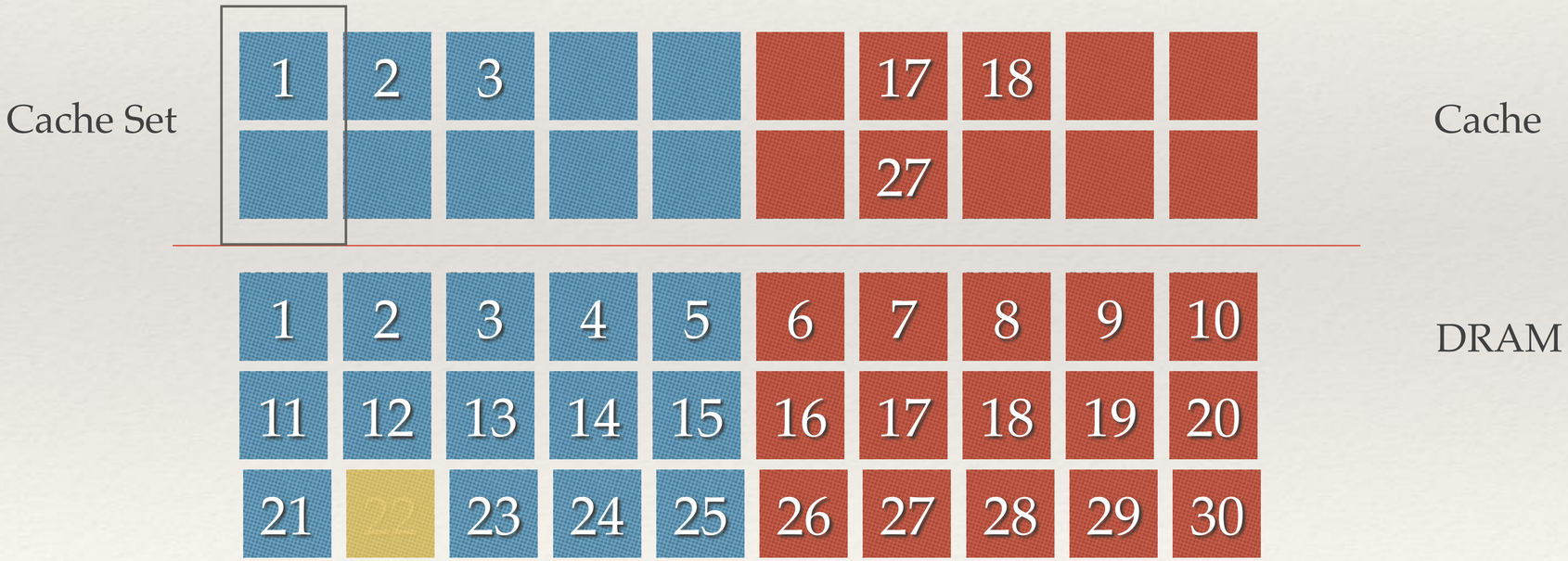
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



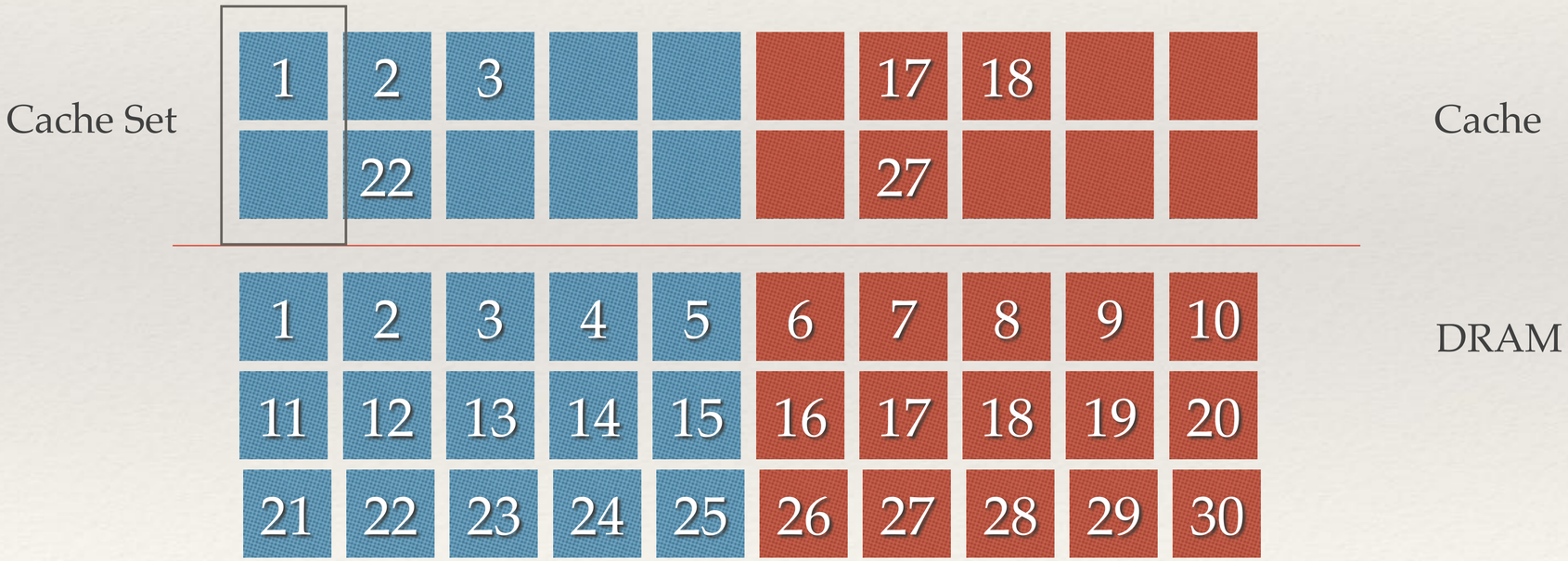
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



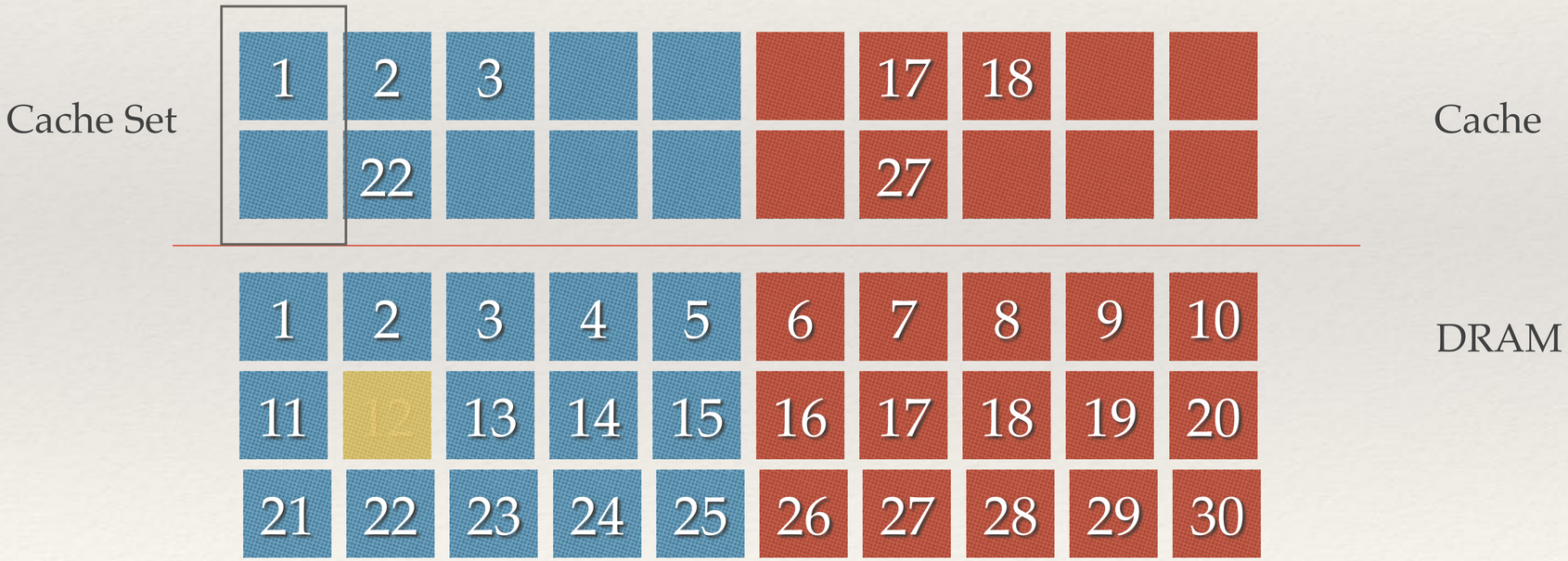
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



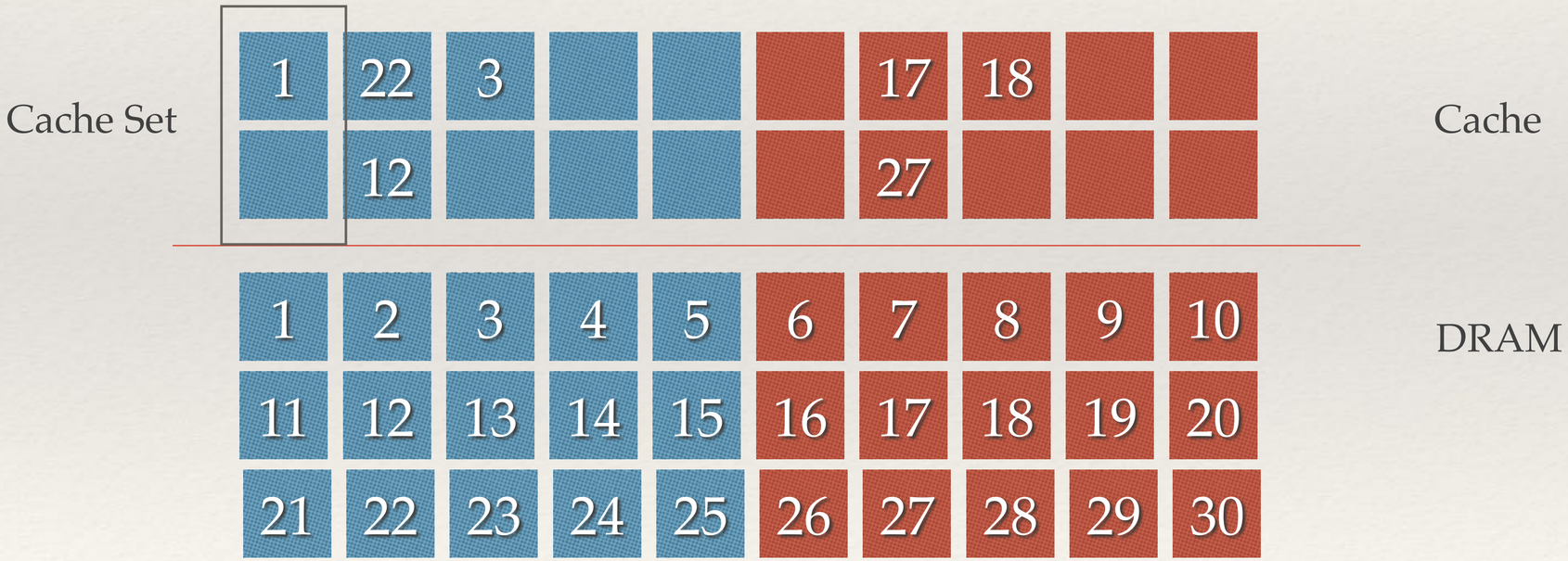
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



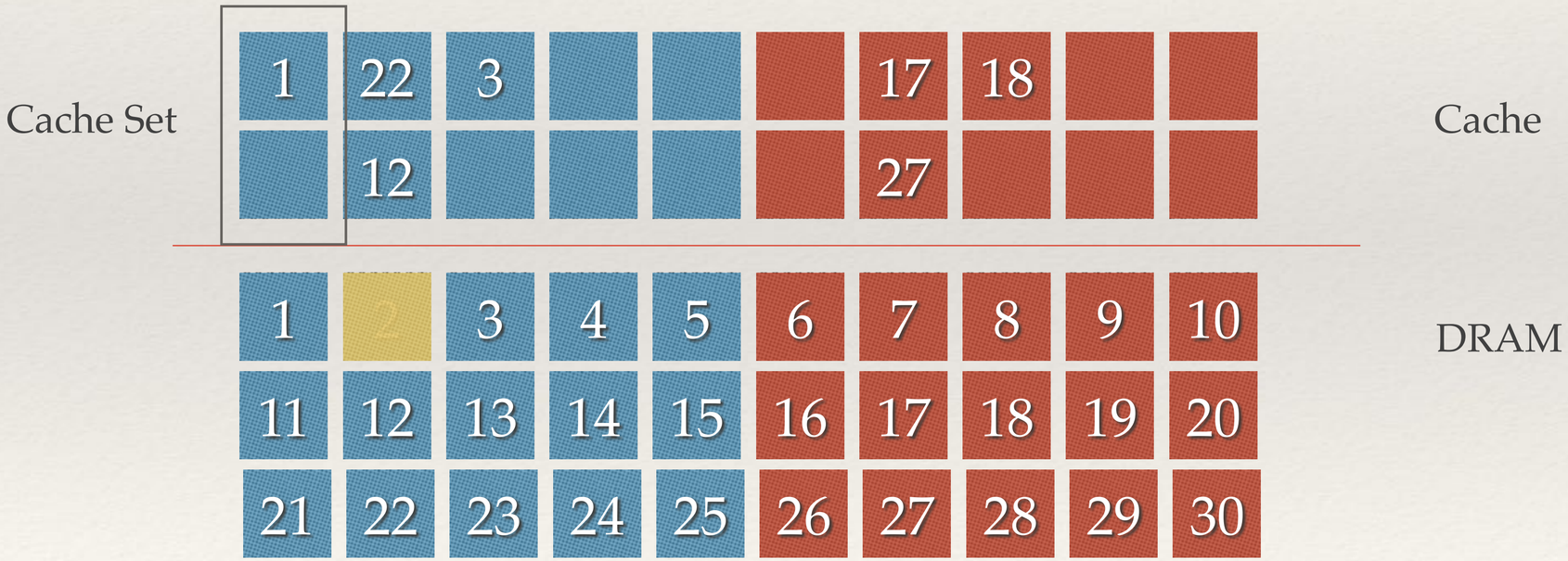
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



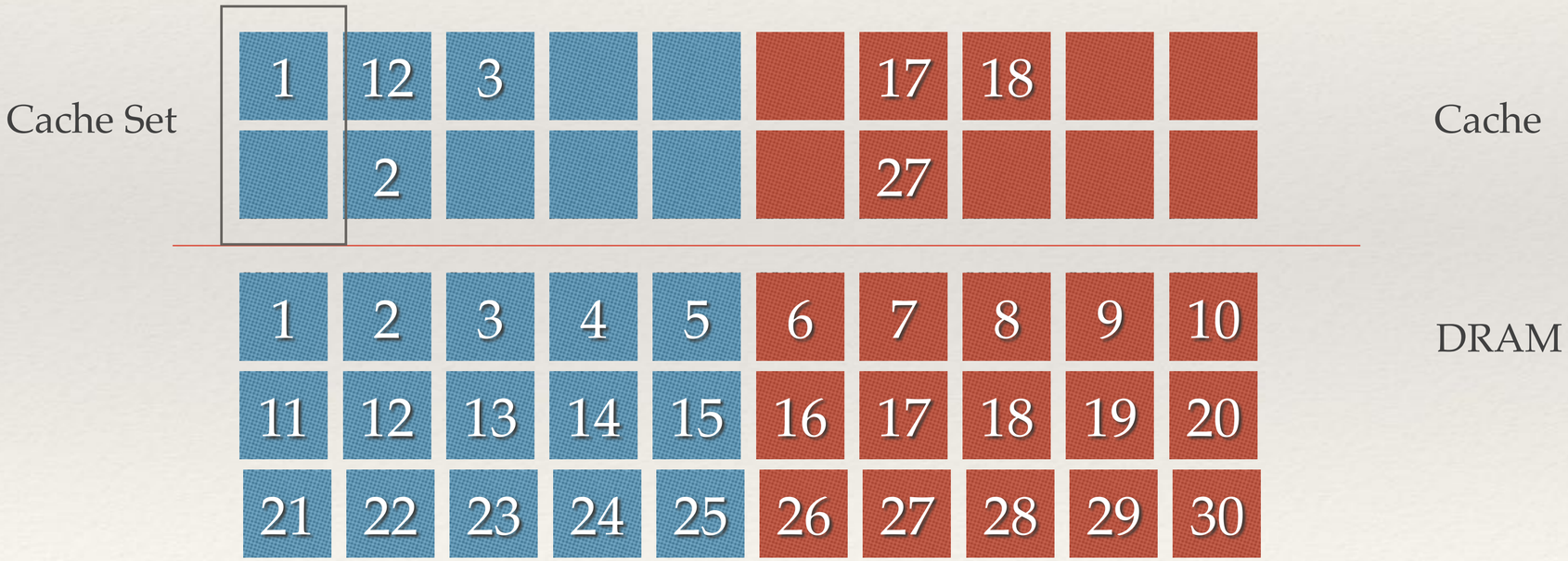
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



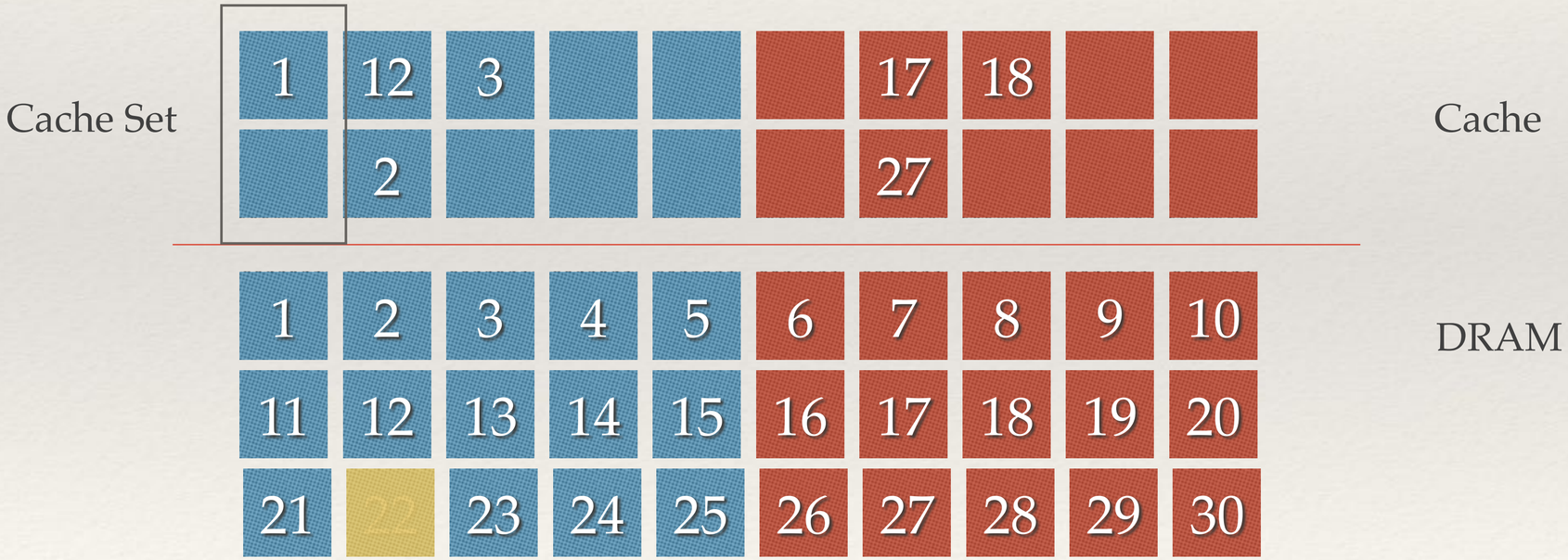
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



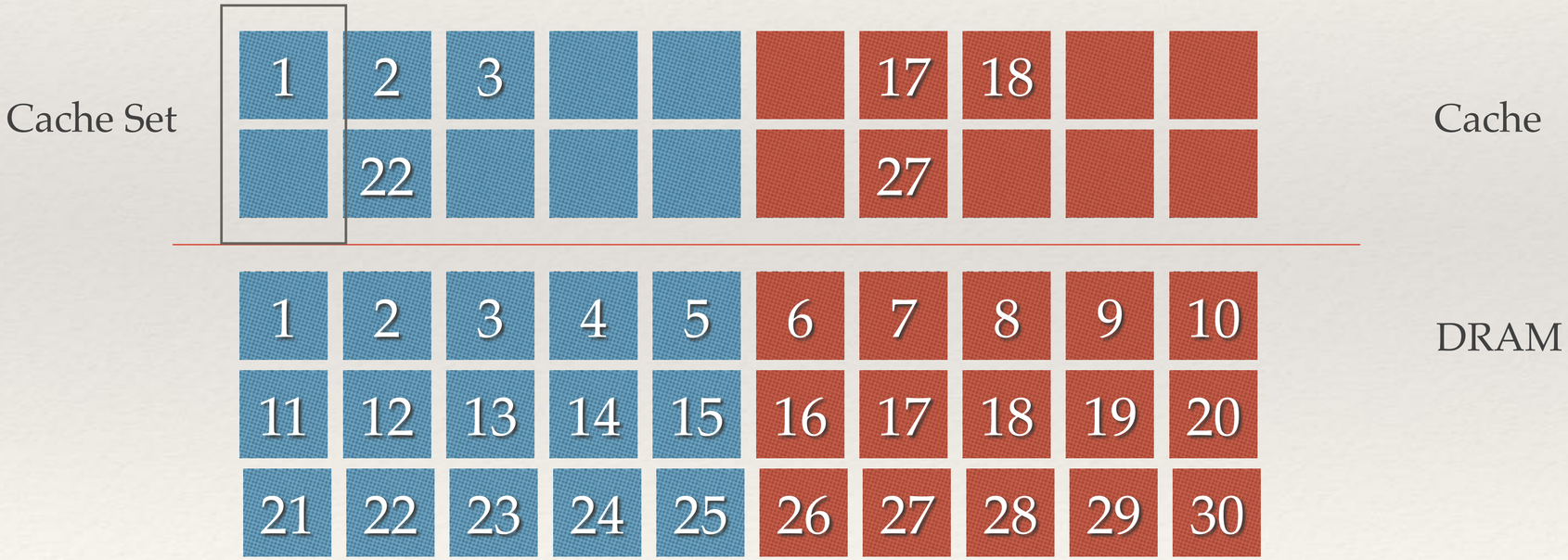
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



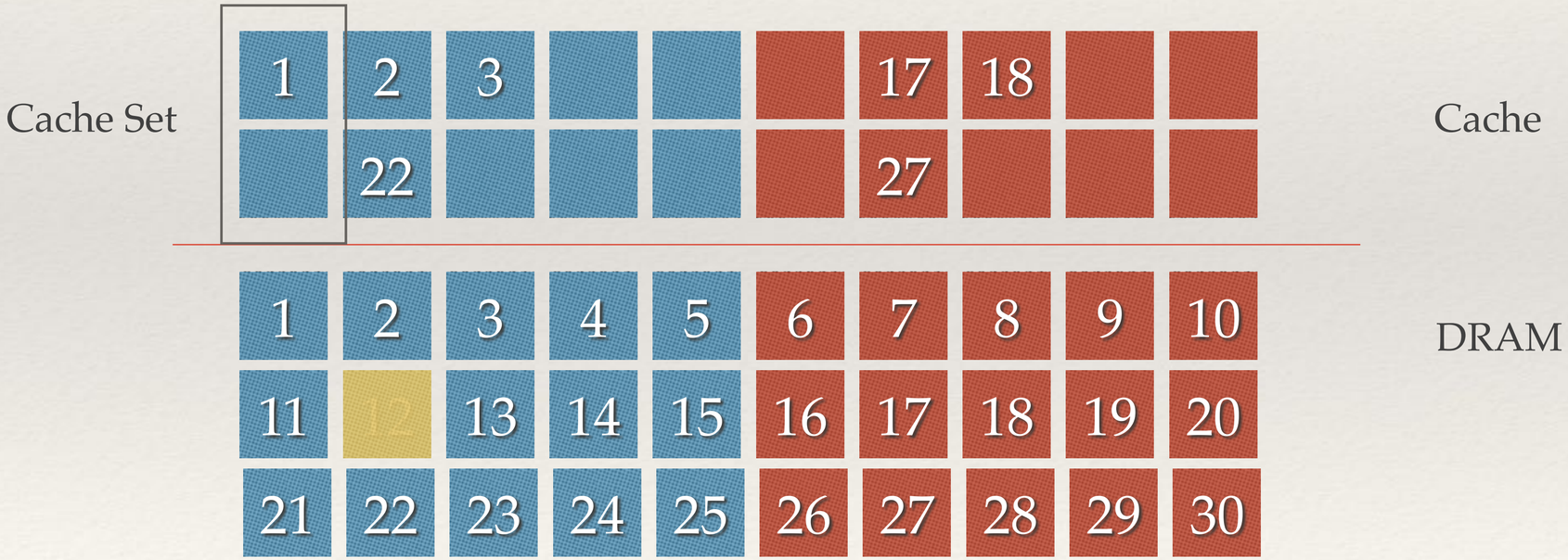
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



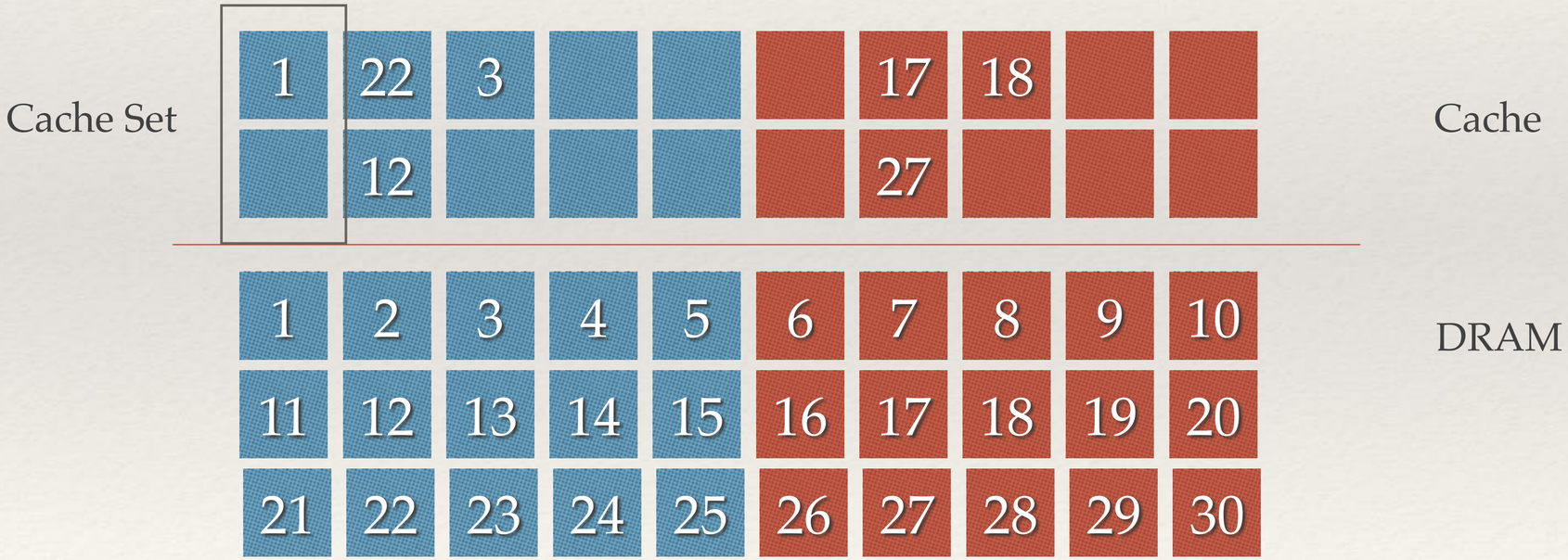
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



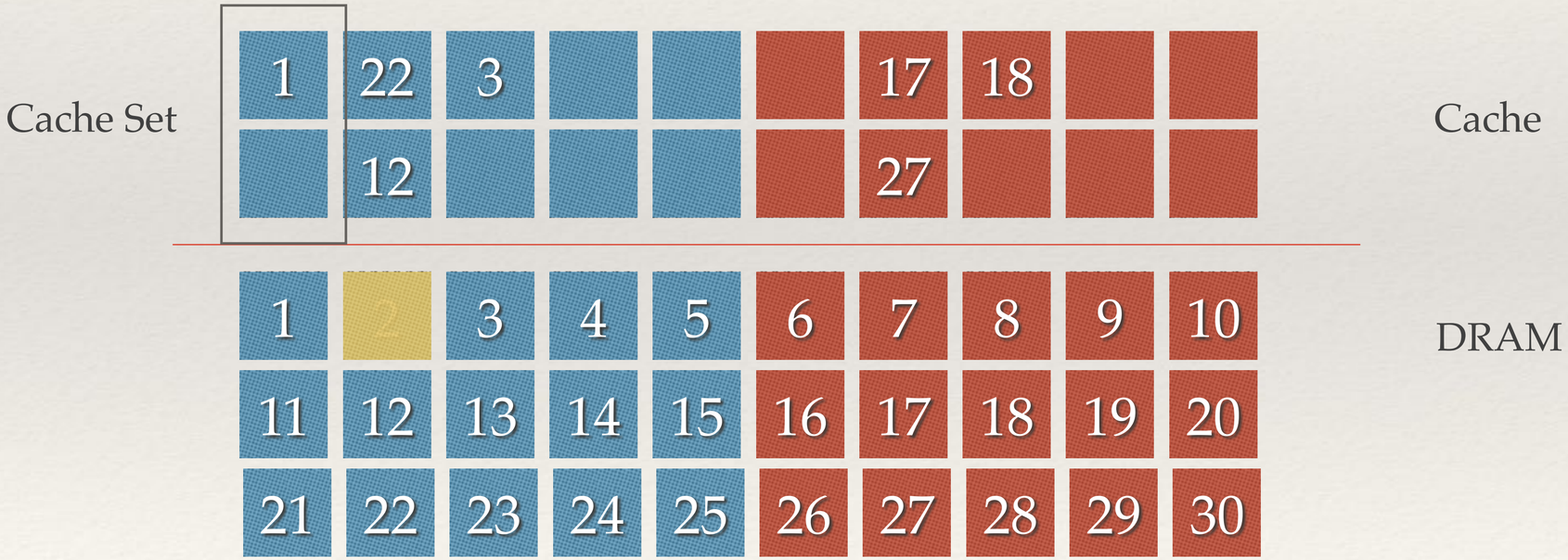
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



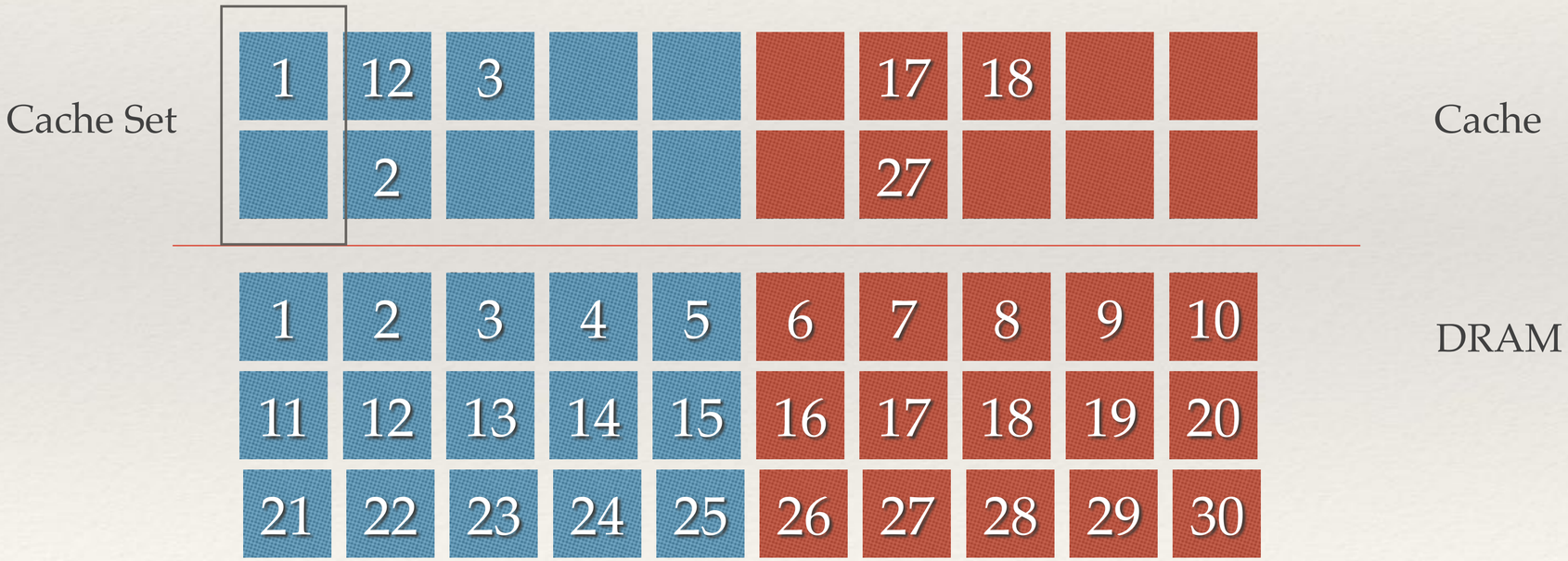
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



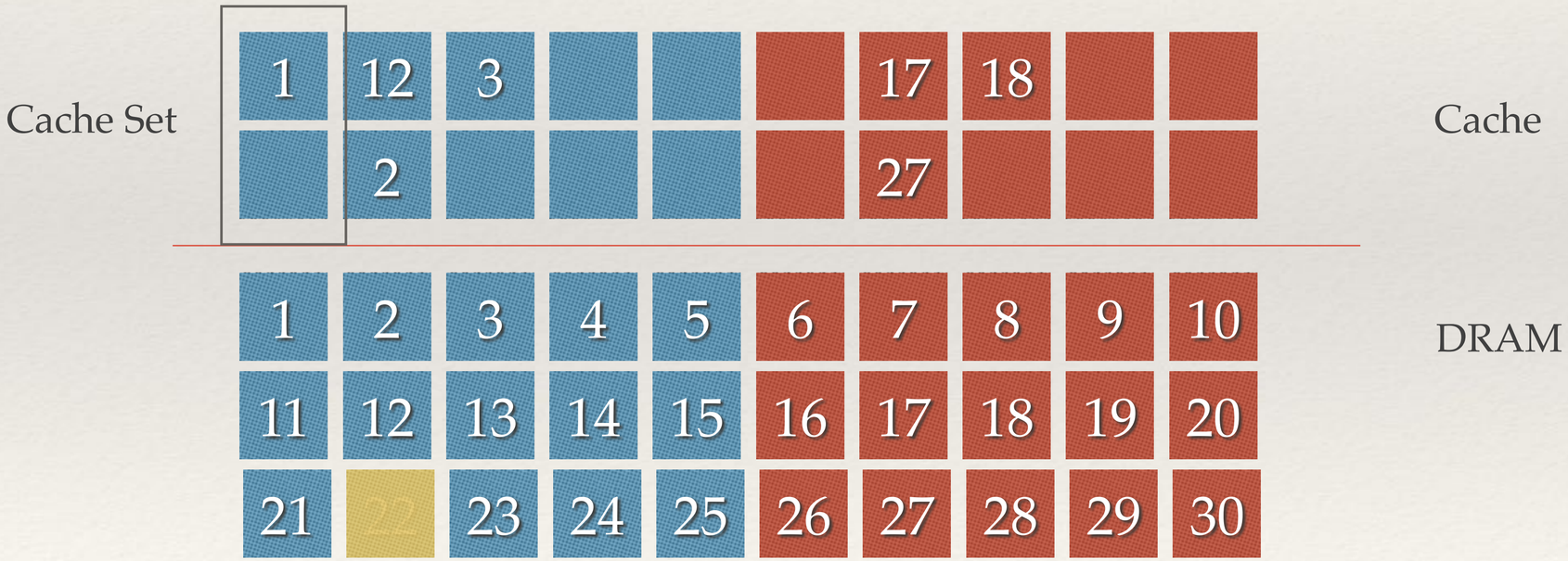
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



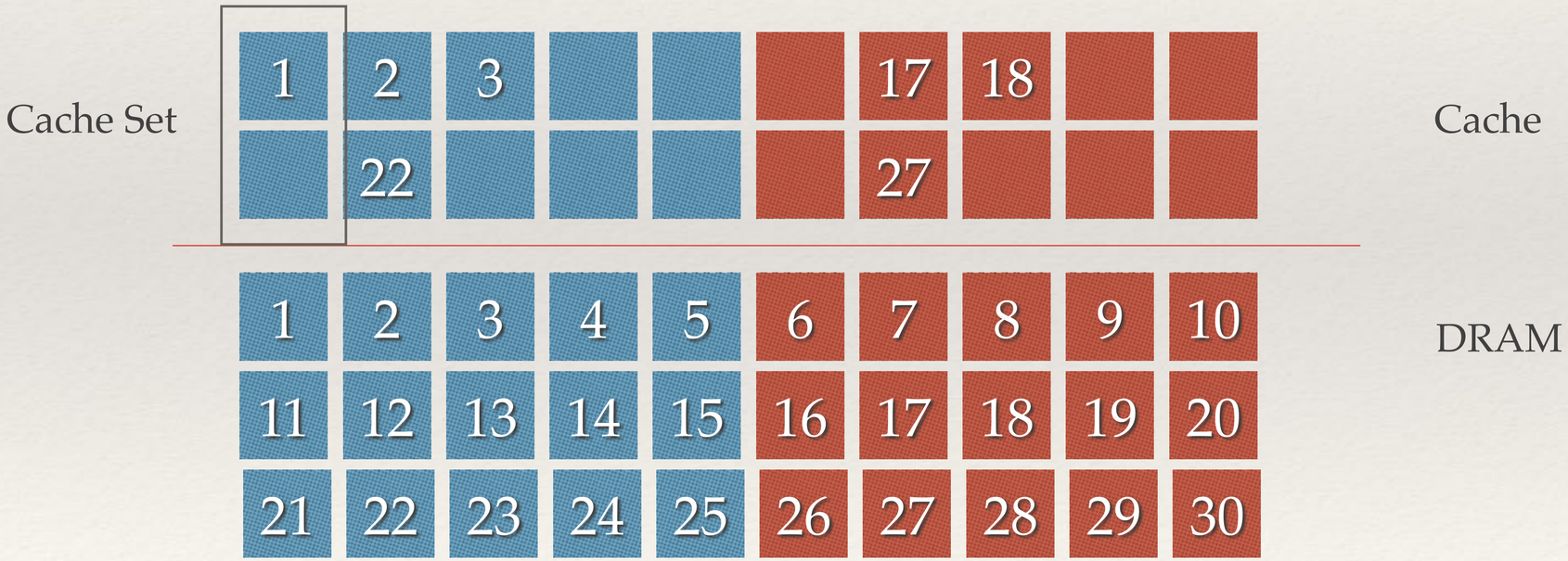
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



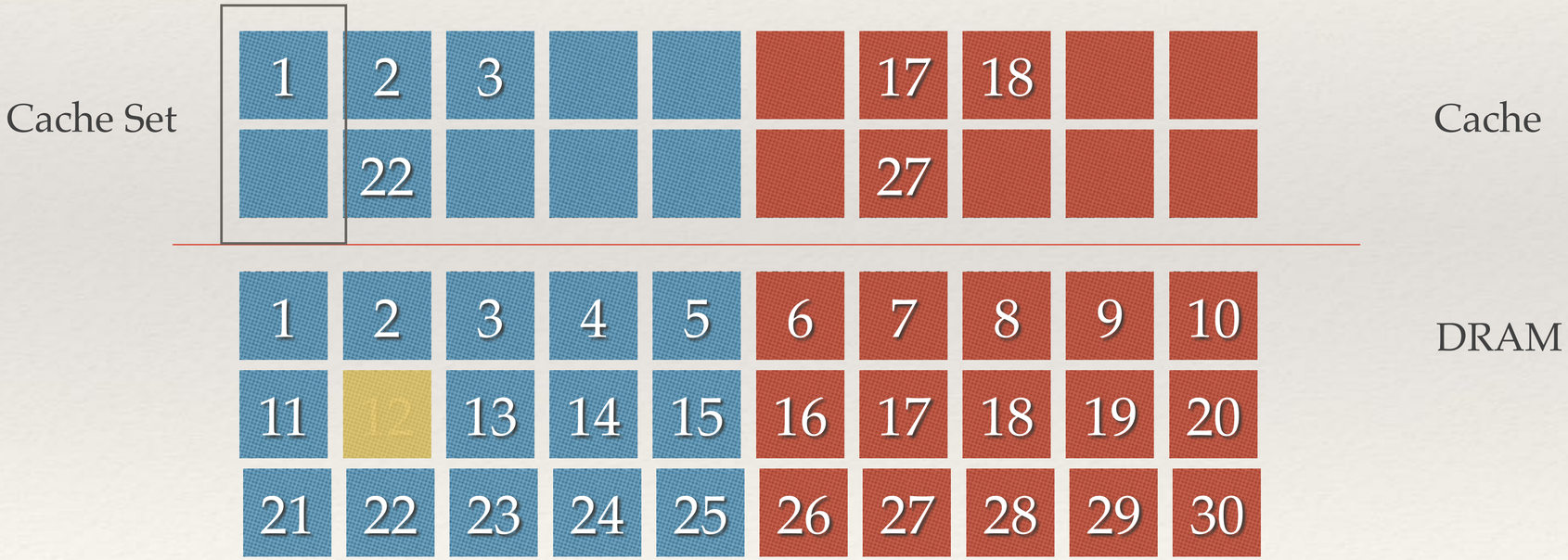
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



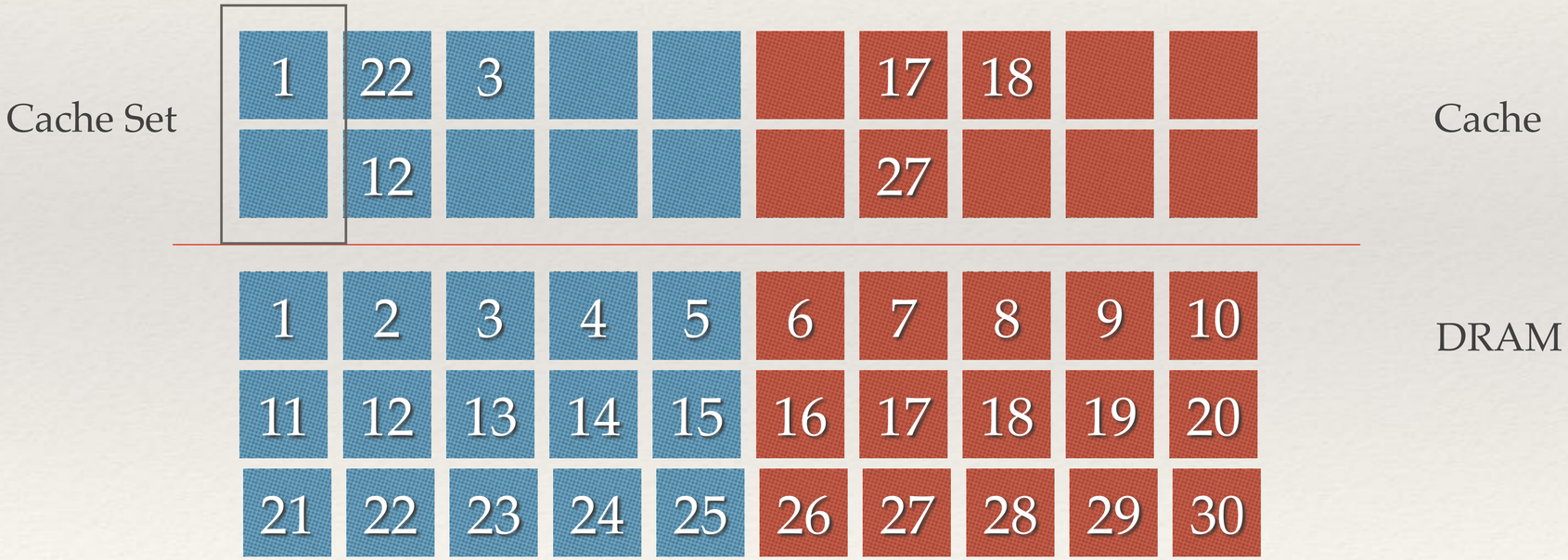
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



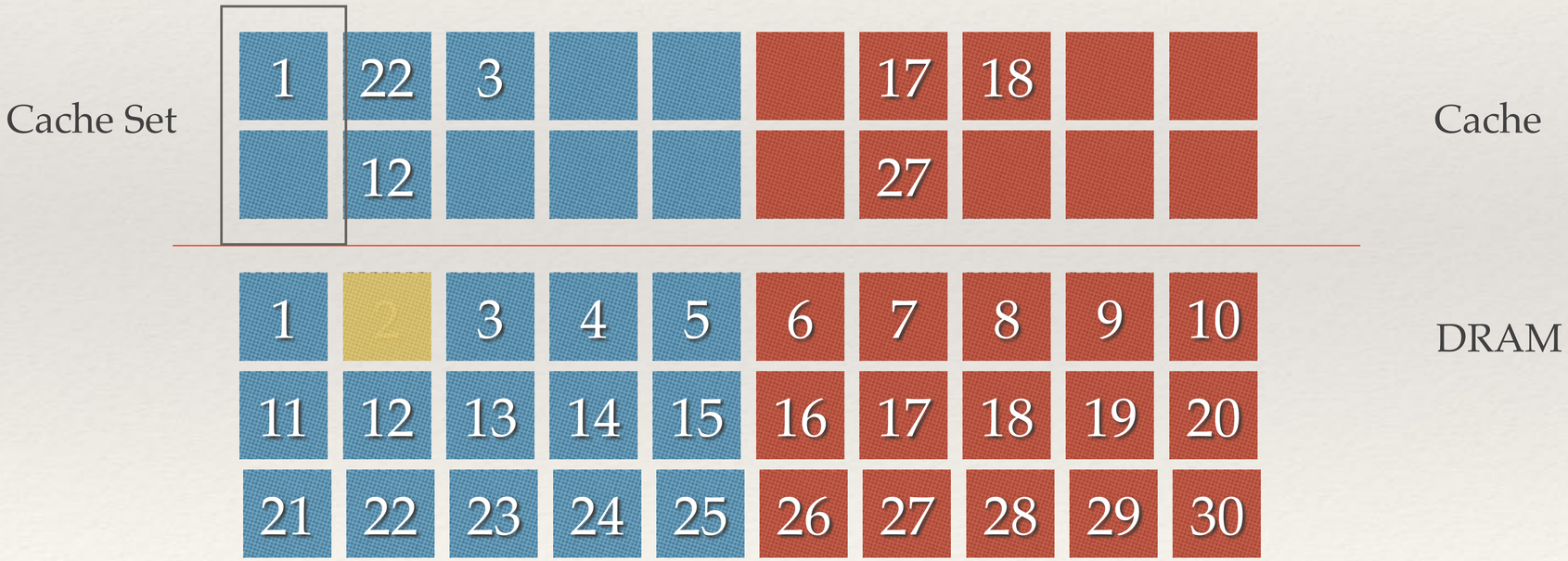
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors



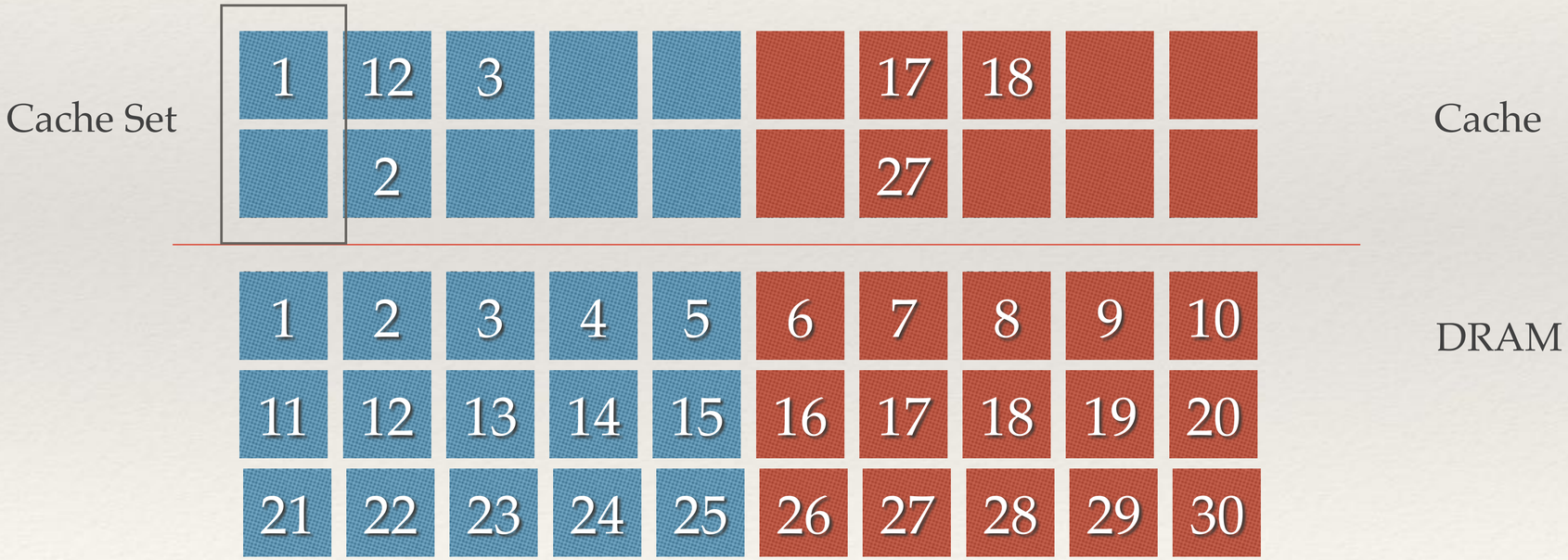
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors

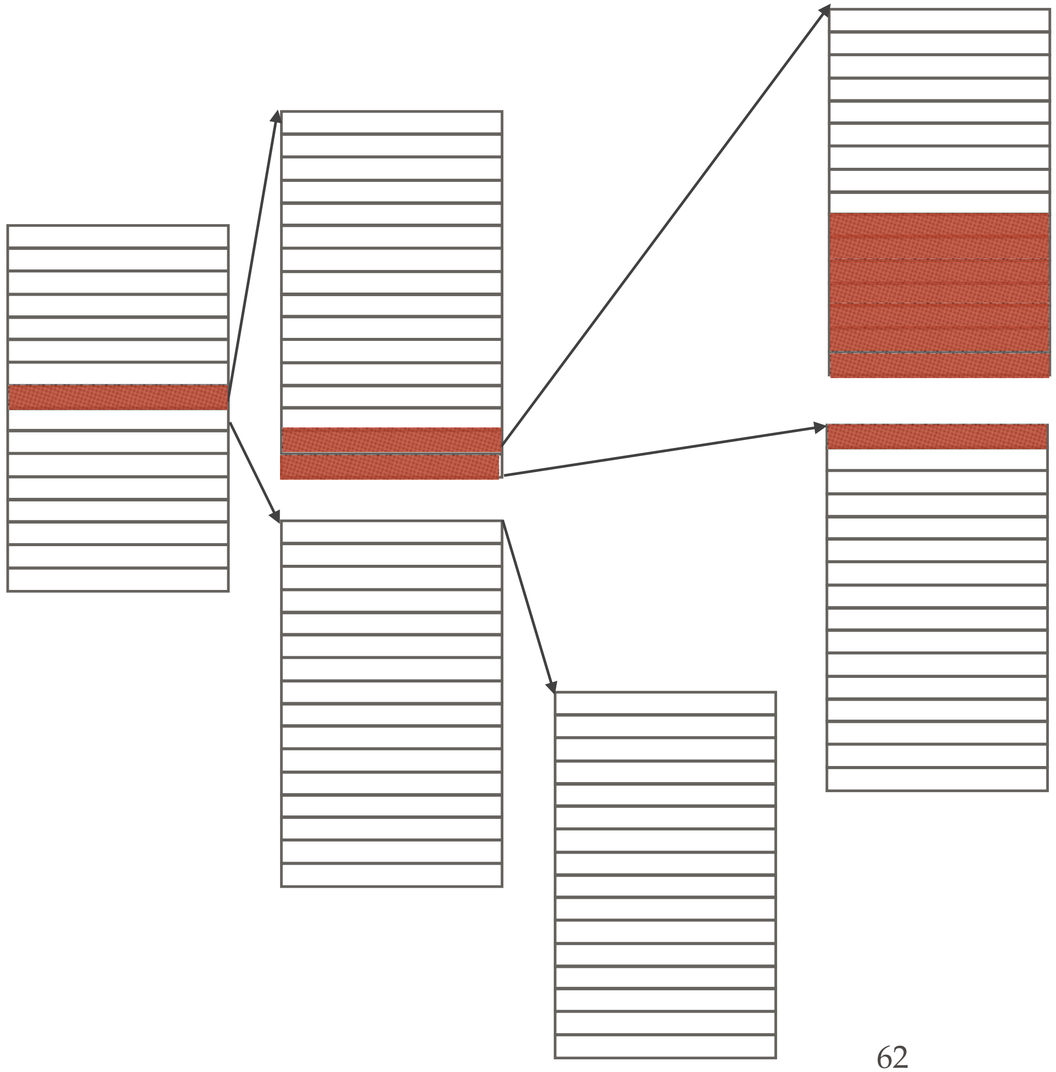


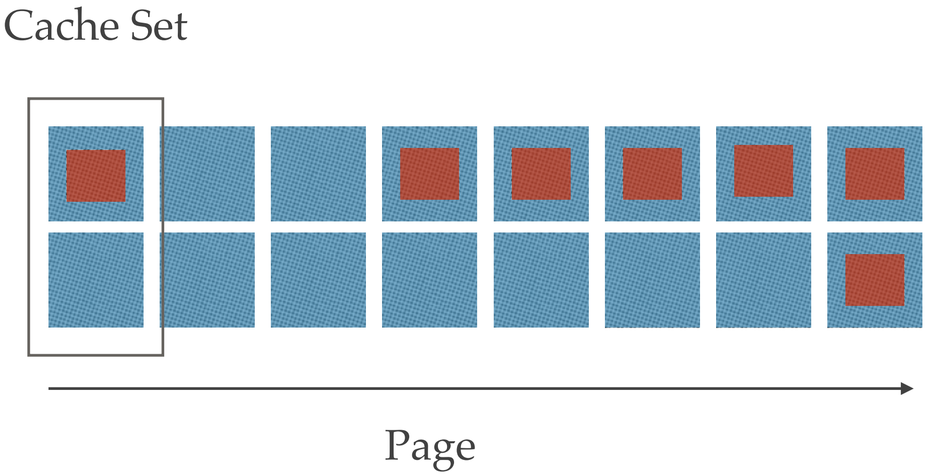
Tiny Cache Example

- 2-way cache, 5 sets per page, showing 2 colors
- Eviction sets follow page offsets



Big picture: cached page tables





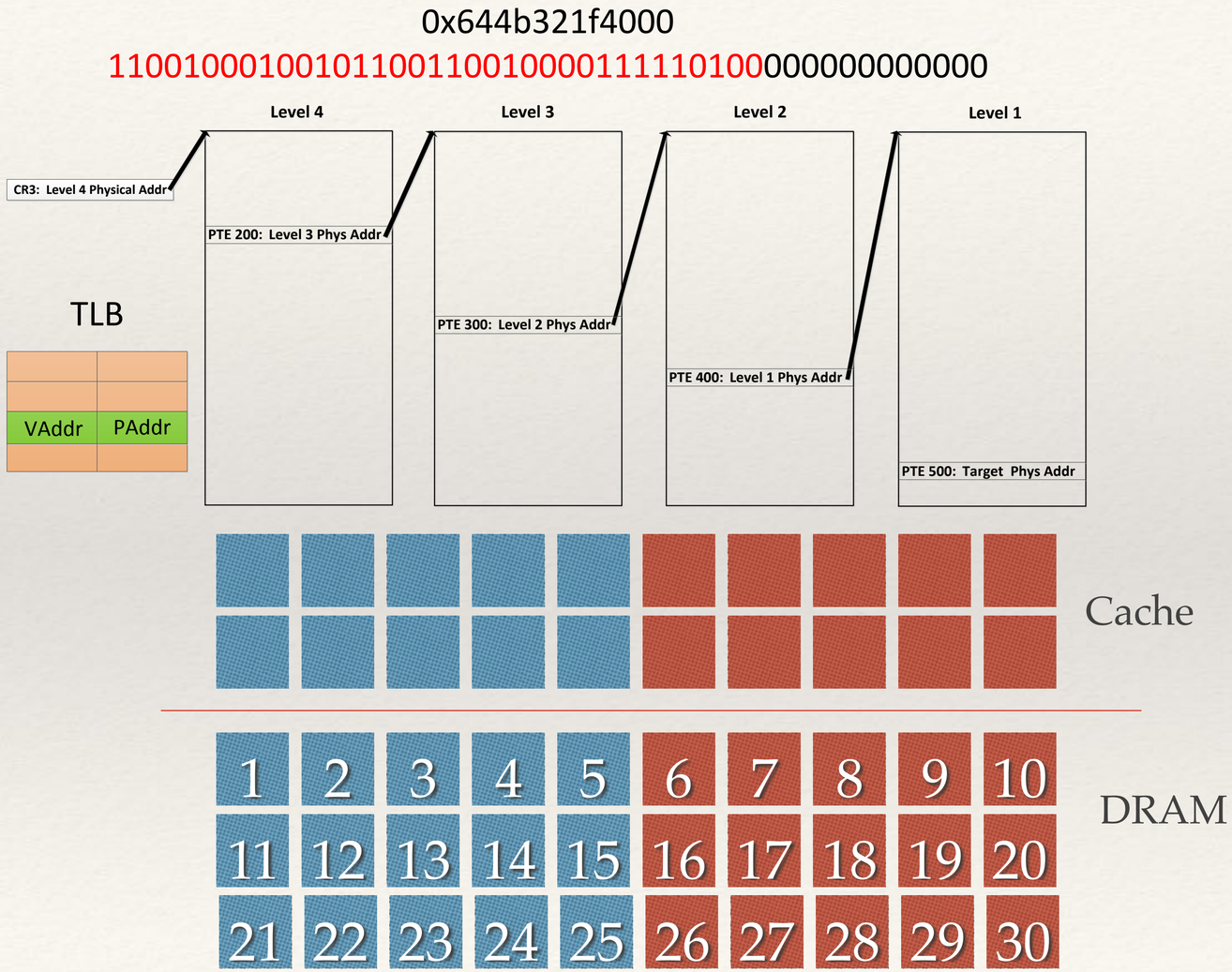
Outline

- Justification: ASLR
- Side Channels
- Pagetable walks
- CPU Caches
- **EVICT+TIME**
- JavaScript
- Results
- Demo

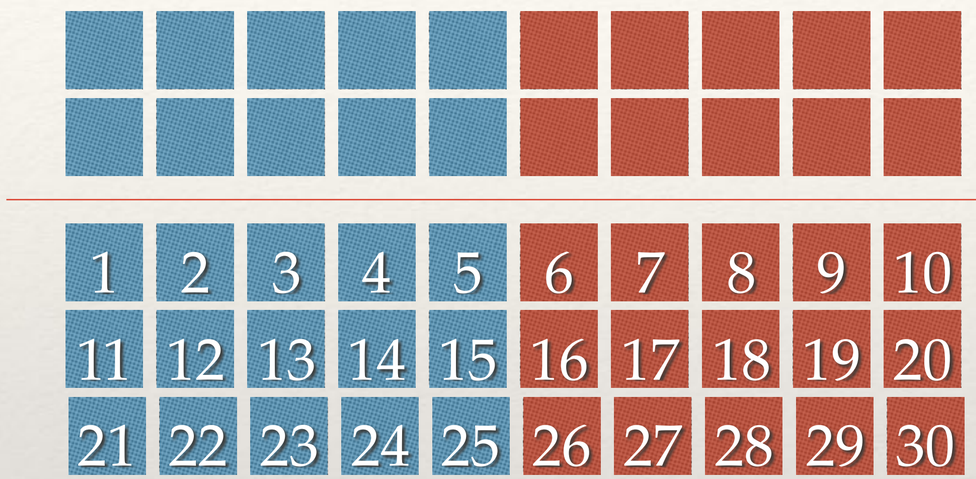
EVICT+TIME

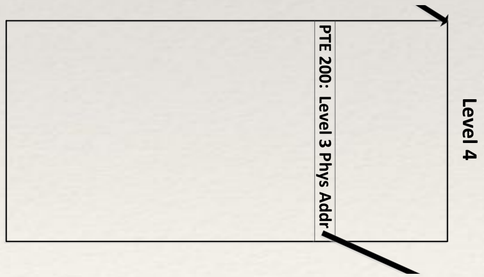
- Flush TLB, forcing pagetable walk
- Evict first cacheline
- Measure lookup time
- Find cacheline dependencies

EVICT+TIME in Cache



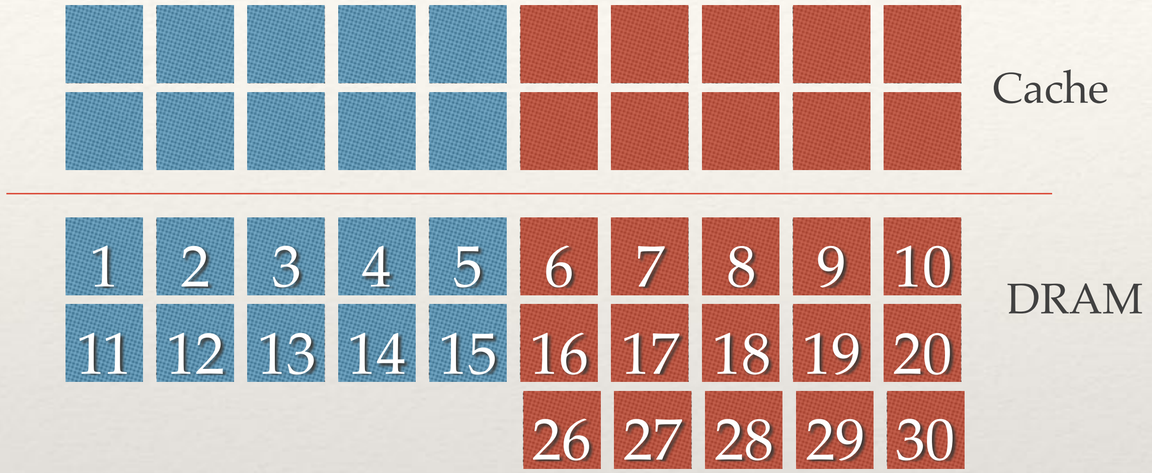
EVICT+TIME in Cache

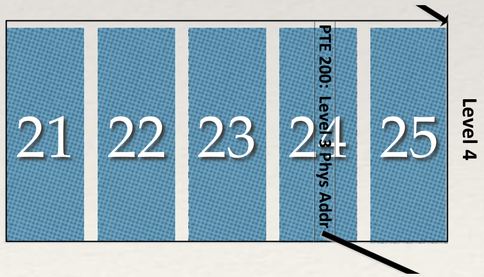




Pagetable in DRAM

EVICT+TIME in Cache

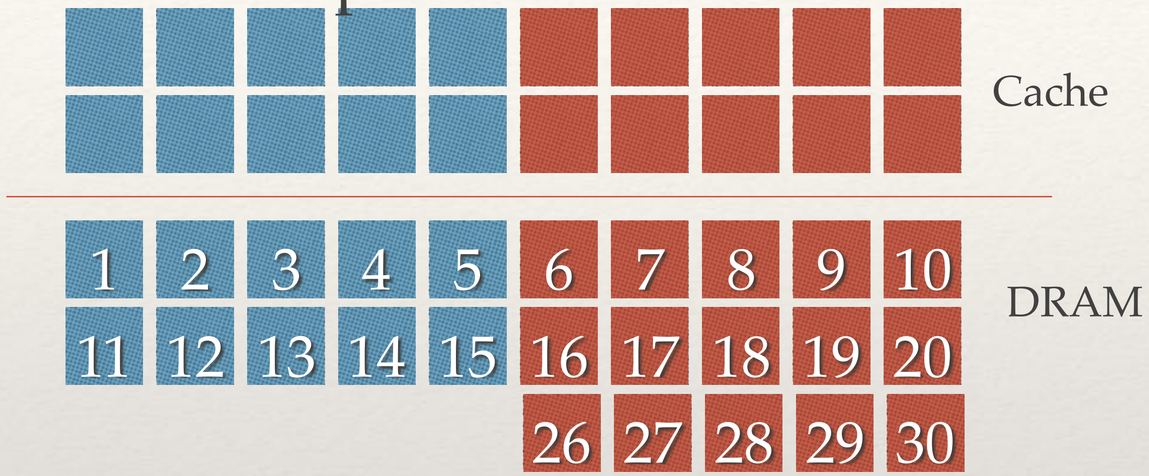


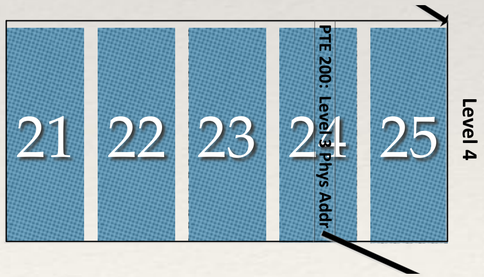


Pagetable in DRAM

EVICT+TIME in Cache

- Do address lookup

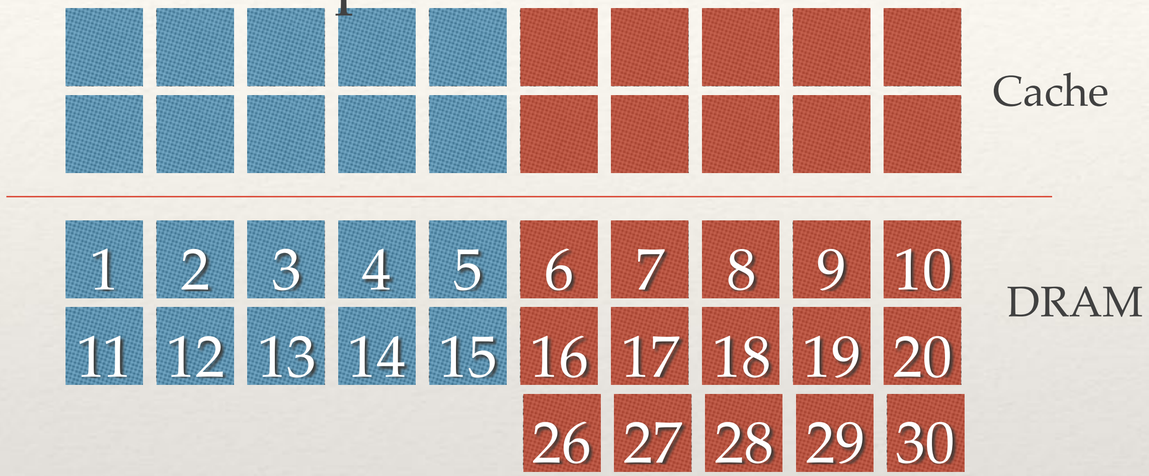




Pagetable in DRAM

EVICT+TIME in Cache

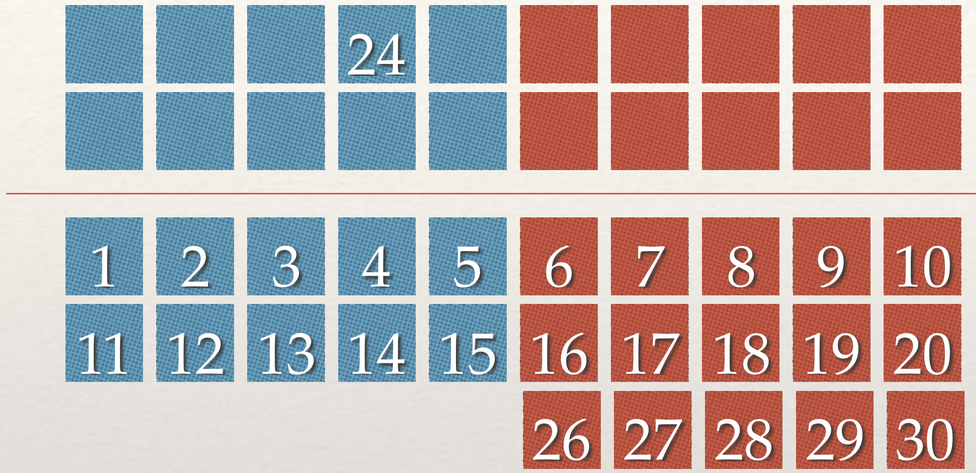
- Do address lookup

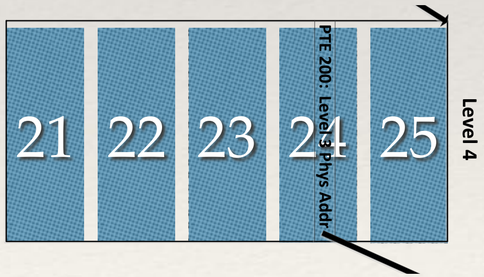


Pagetable in DRAM

EVICT+TIME in Cache

- It was uncached - slow

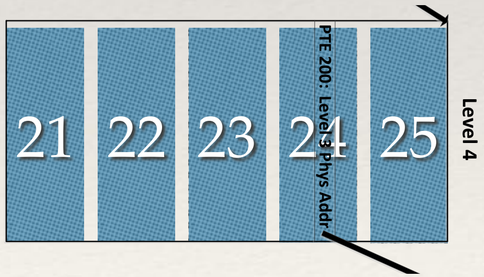




Pagetable in DRAM

EVICT+TIME in Cache

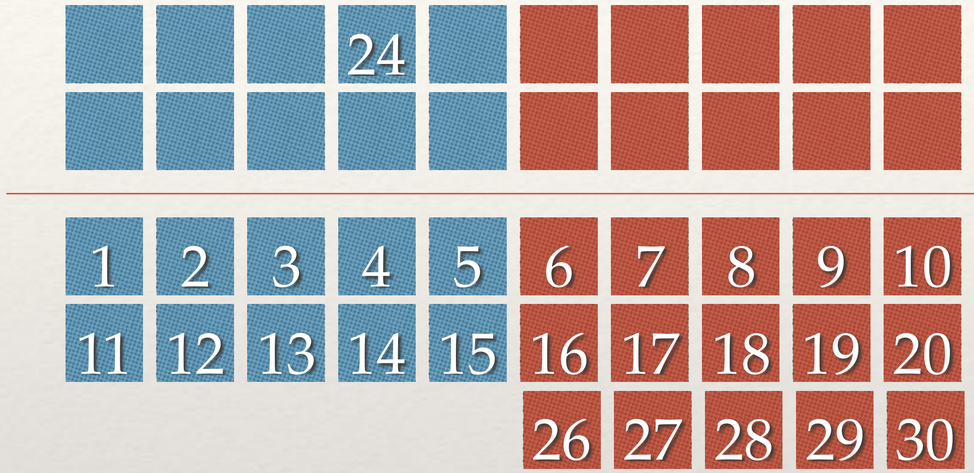
- Let's do it again

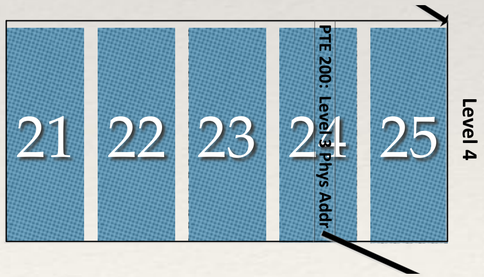


Pagetable in DRAM

EVICT+TIME in Cache

- It was cached - fast

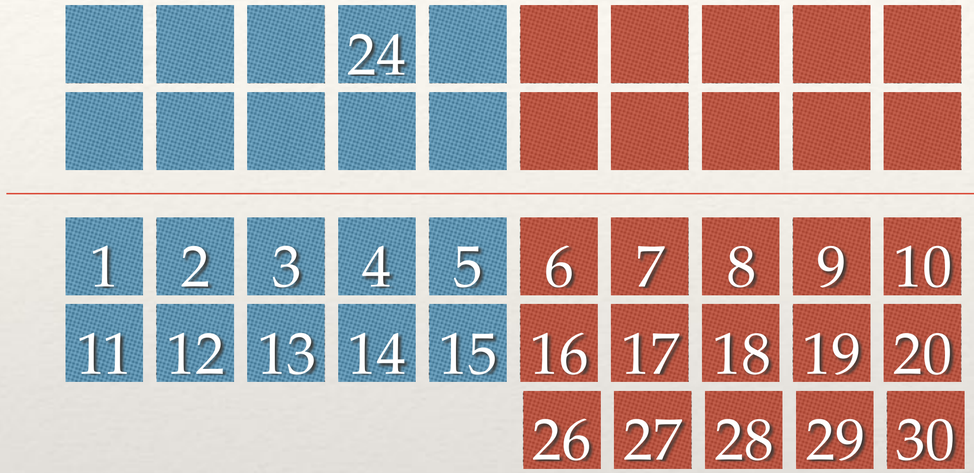


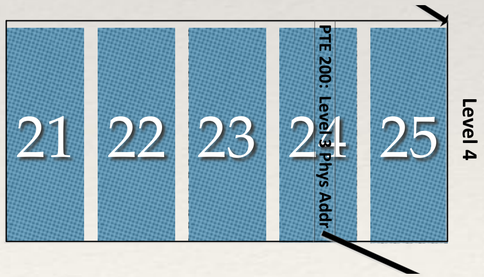


Pagetable in DRAM

EVICT+TIME in Cache

- Let's evict

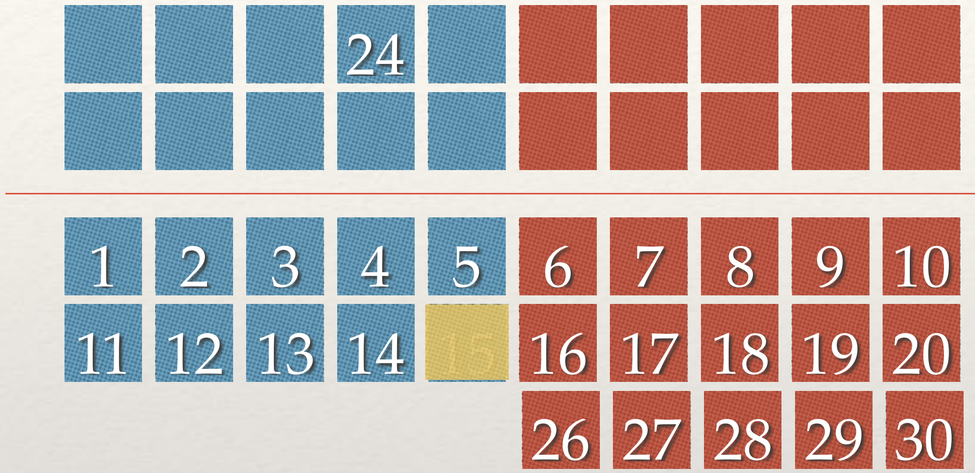


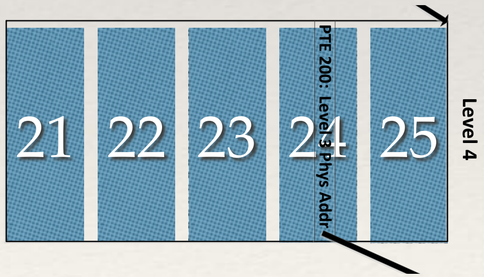


Pagetable in DRAM

EVICT+TIME in Cache

- Let's evict

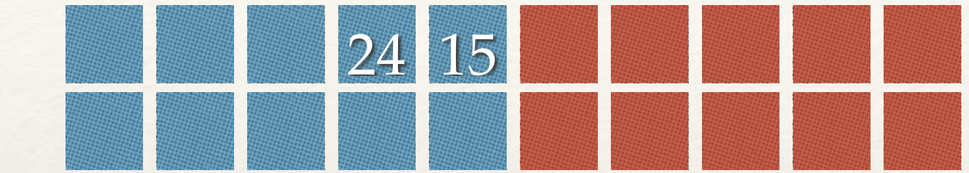




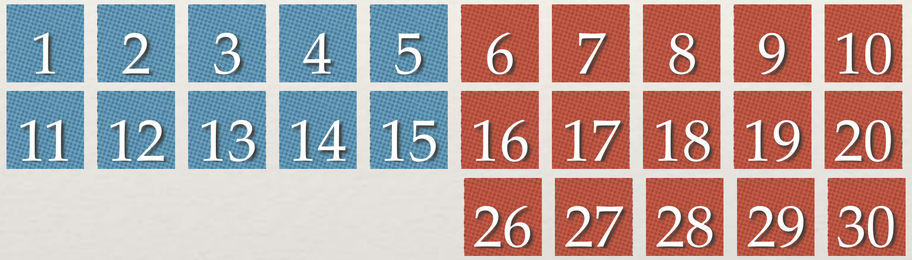
Pagetable in DRAM

EVICT+TIME in Cache

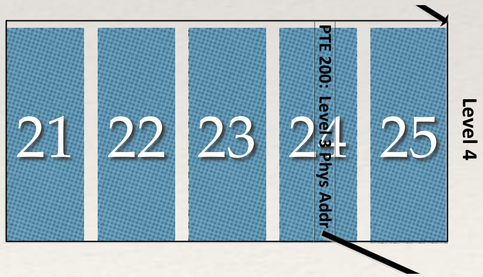
- Let's evict



Cache



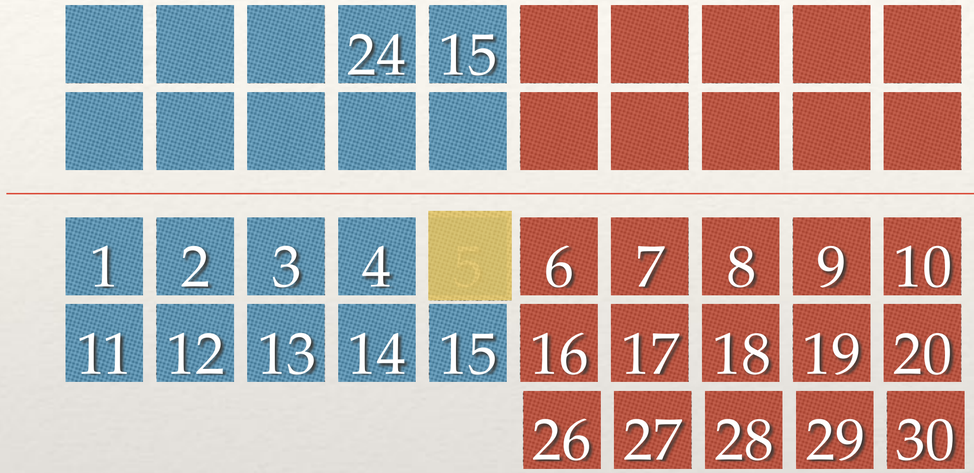
DRAM

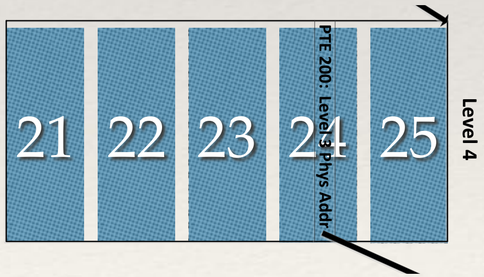


Pagetable in DRAM

EVICT+TIME in Cache

- Let's evict

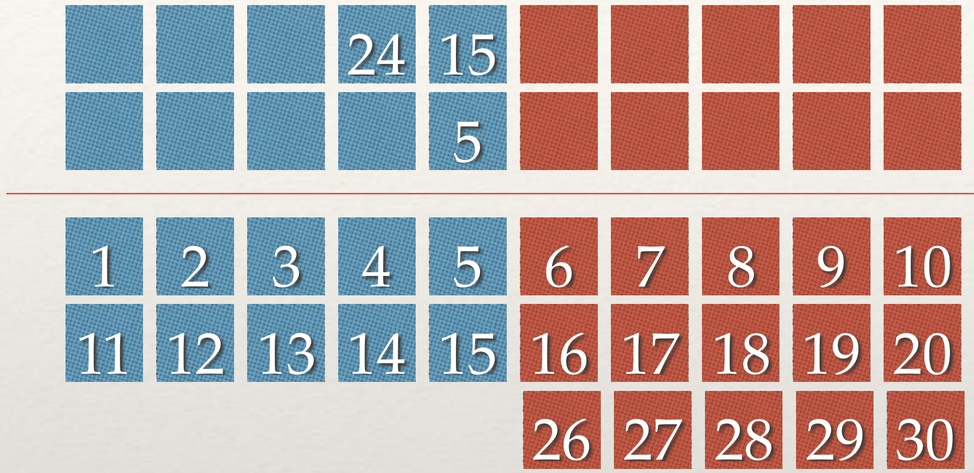


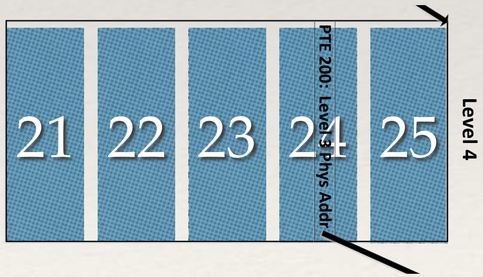


Pagetable in DRAM

EVICT+TIME in Cache

- Let's evict

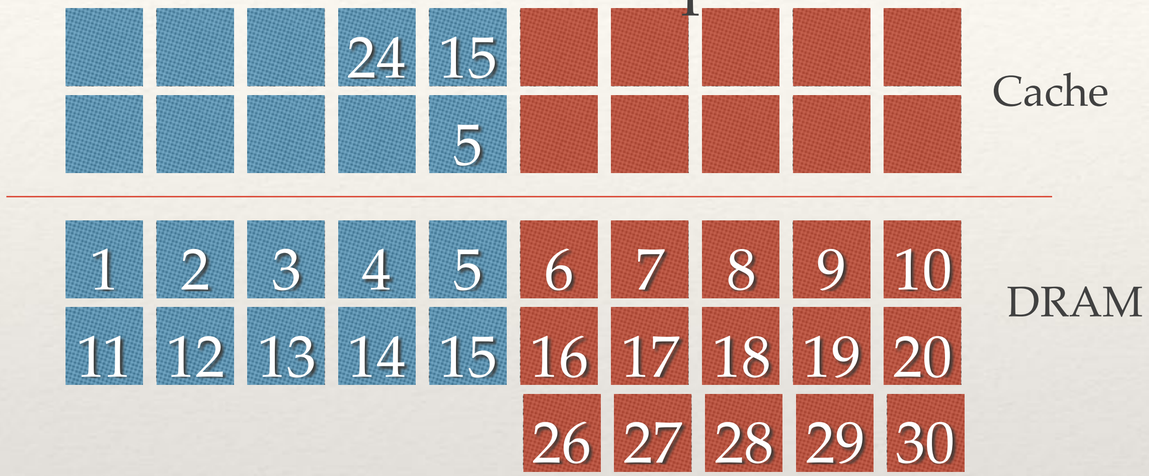


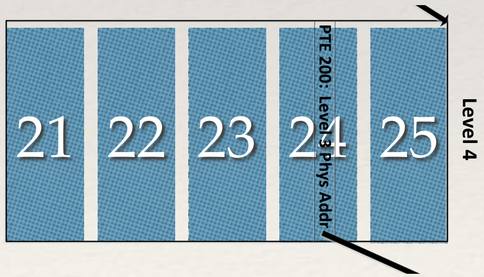


Pagetable in DRAM

EVICT+TIME in Cache

- Eviction done - let's do lookup

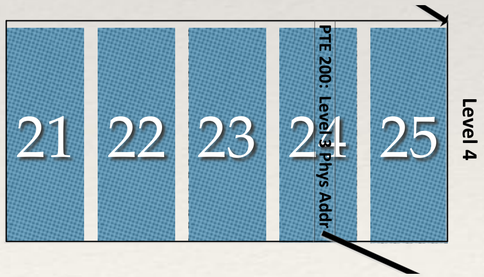




Pagetable in DRAM

EVICT+TIME in Cache

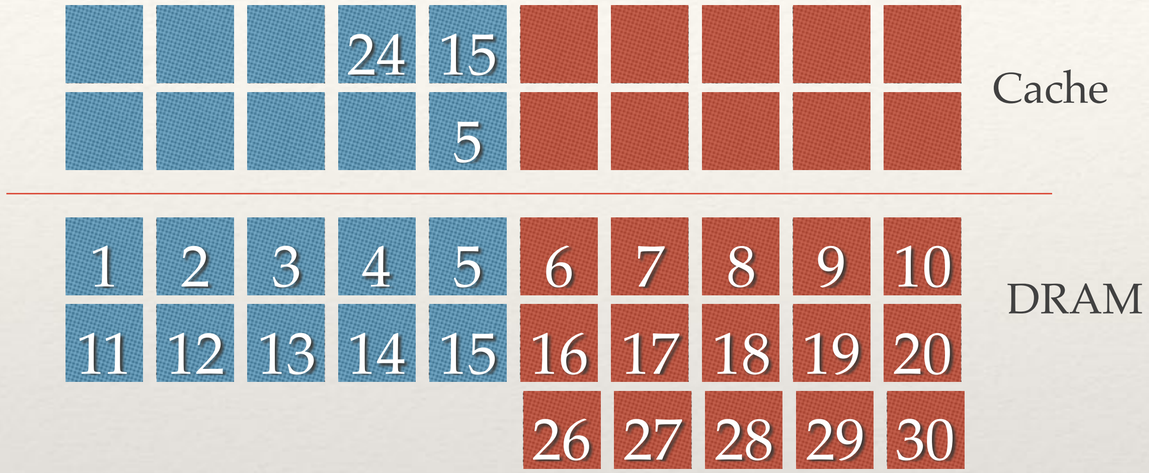
- Eviction done - let's do lookup

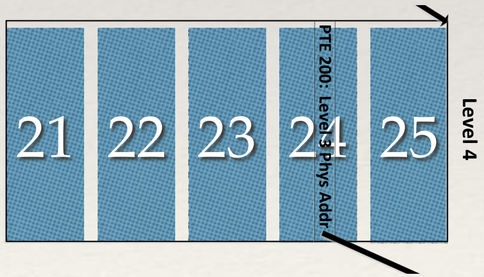


Pagetable in DRAM

EVICT+TIME in Cache

- Still cached

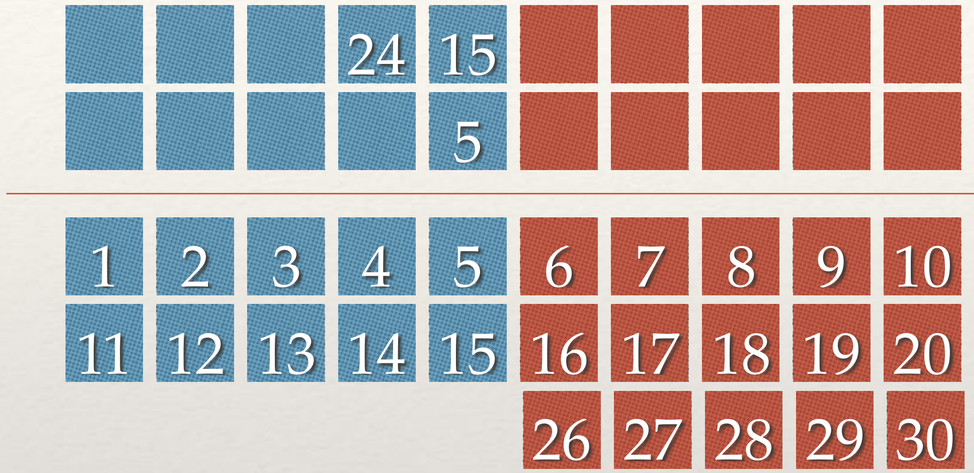


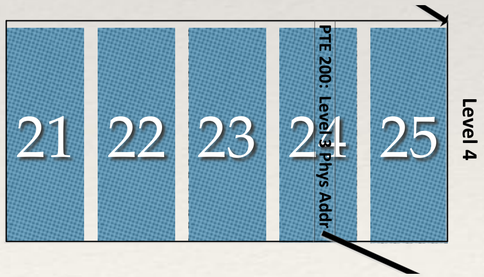


Pagetable in DRAM

EVICT+TIME in Cache

- Let's evict

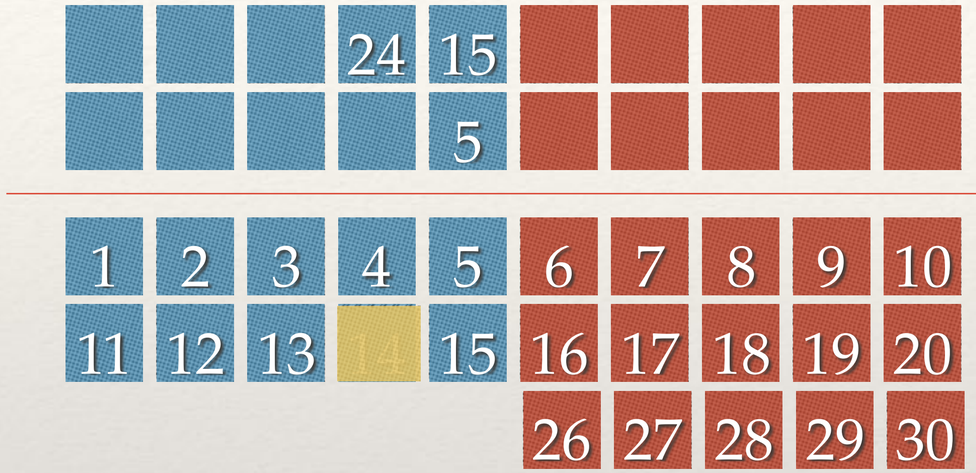


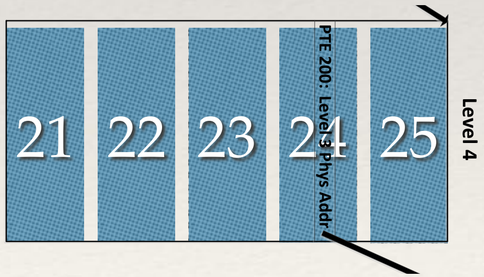


Pagetable in DRAM

EVICT+TIME in Cache

- Let's evict

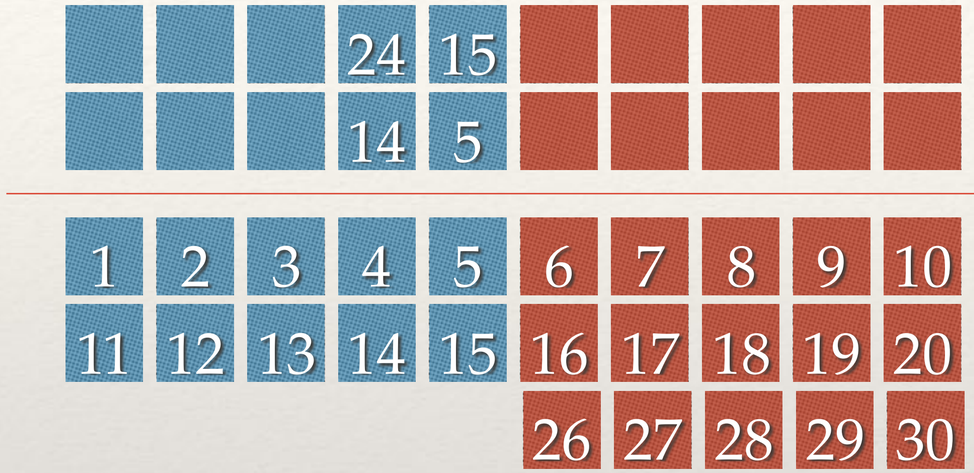


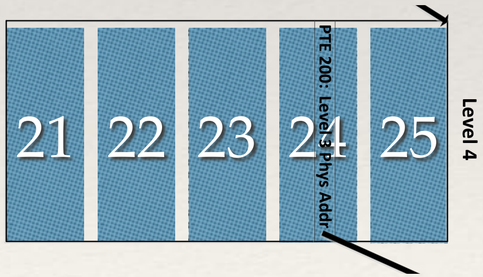


Pagetable in DRAM

EVICT+TIME in Cache

- Let's evict

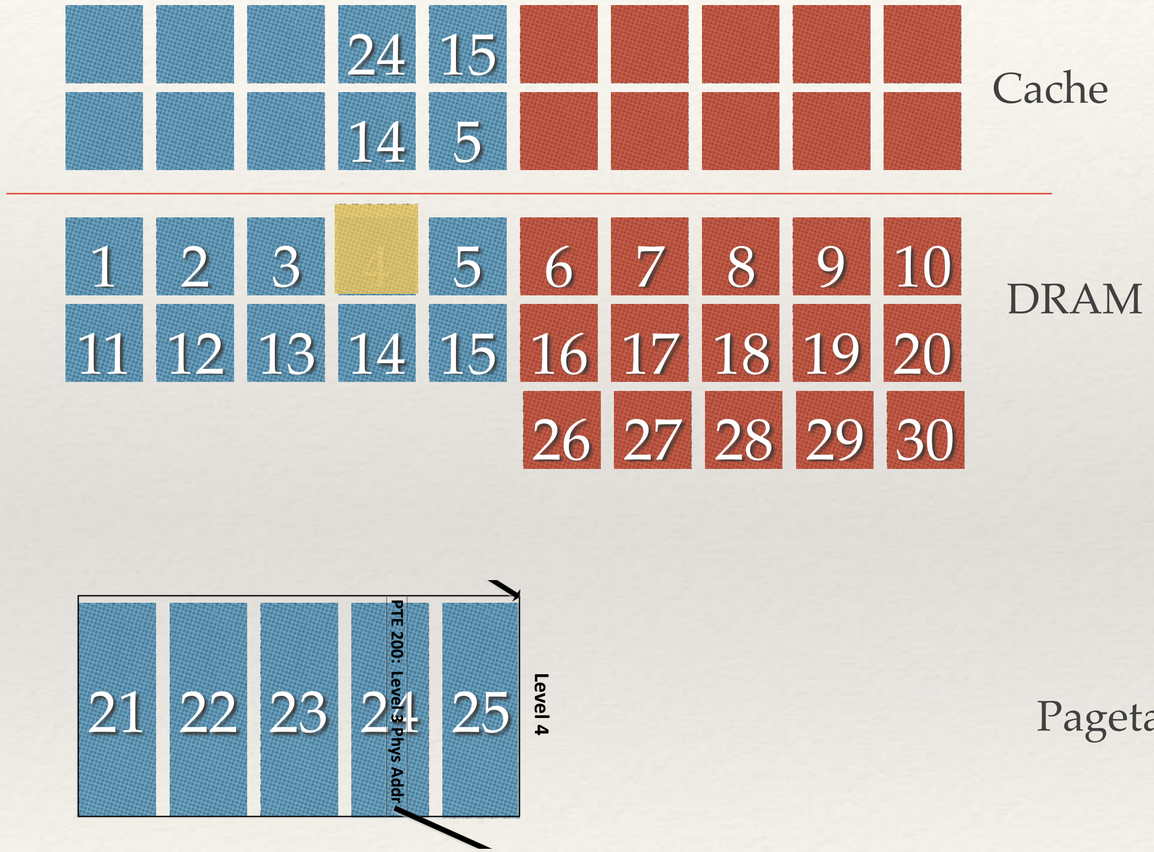




Pagetable in DRAM

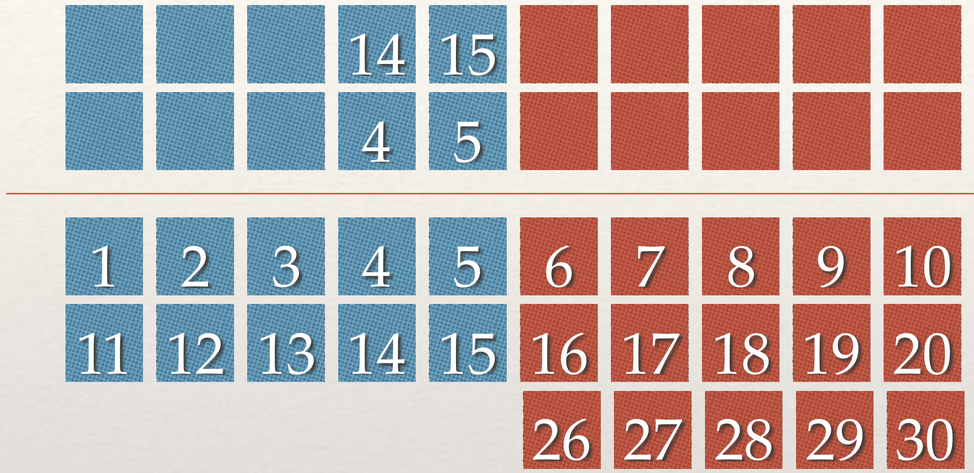
EVICT+TIME in Cache

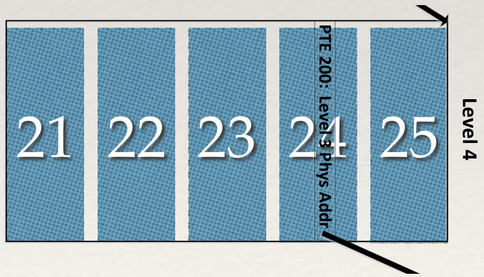
- Let's evict



EVICT+TIME in Cache

- Let's evict

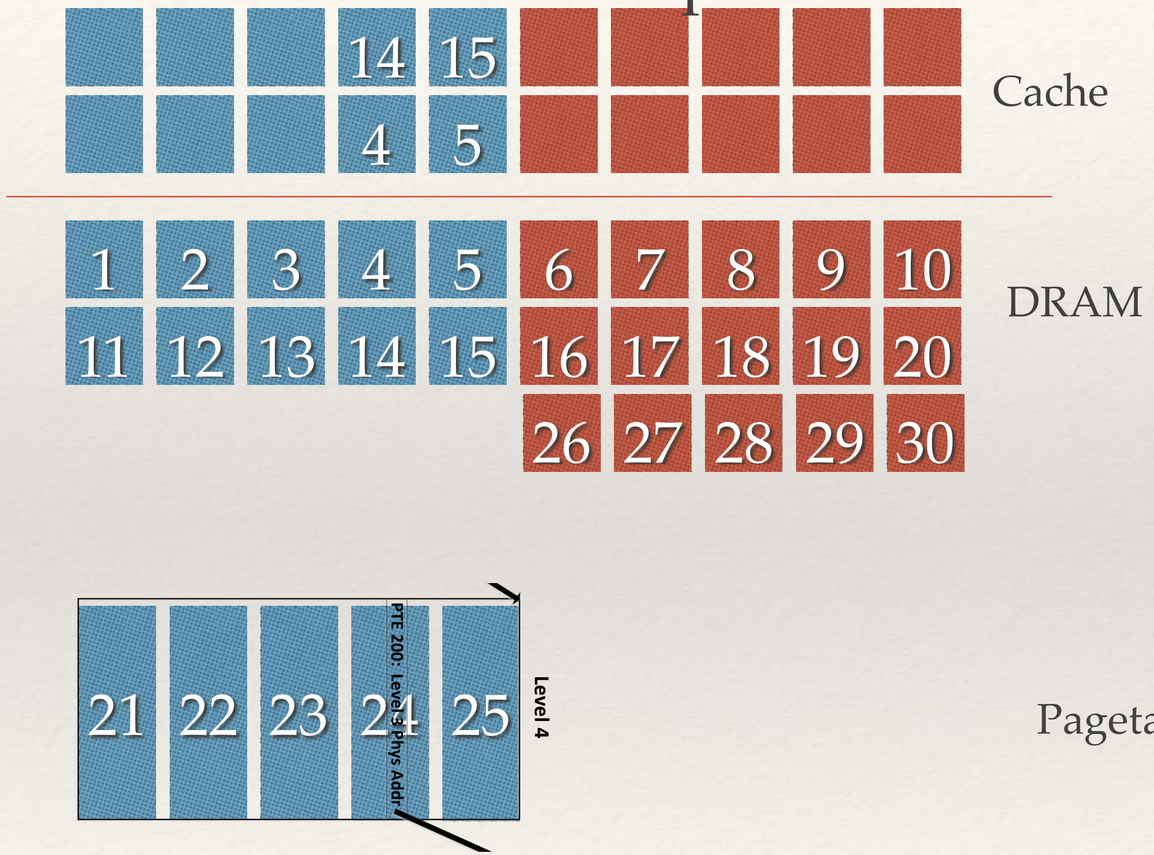




Pagetable in DRAM

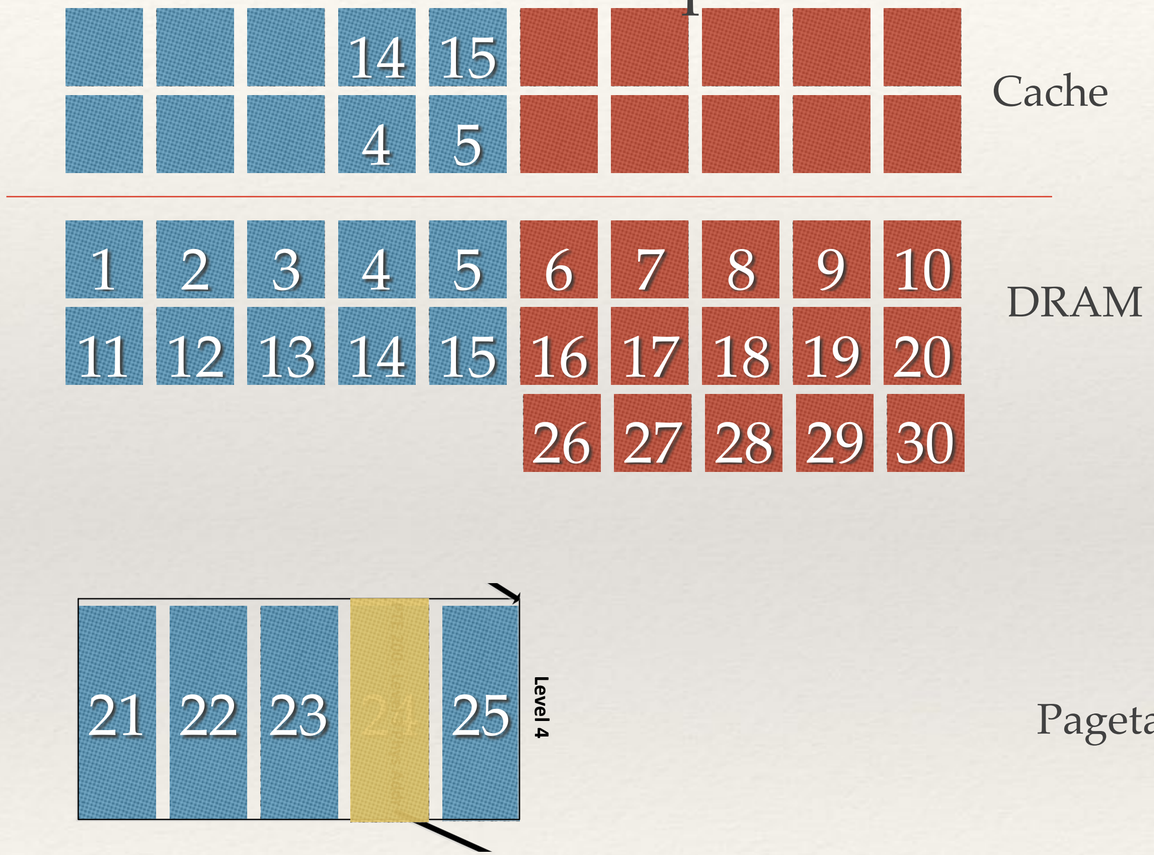
EVICT+TIME in Cache

- Eviction done - let's do lookup



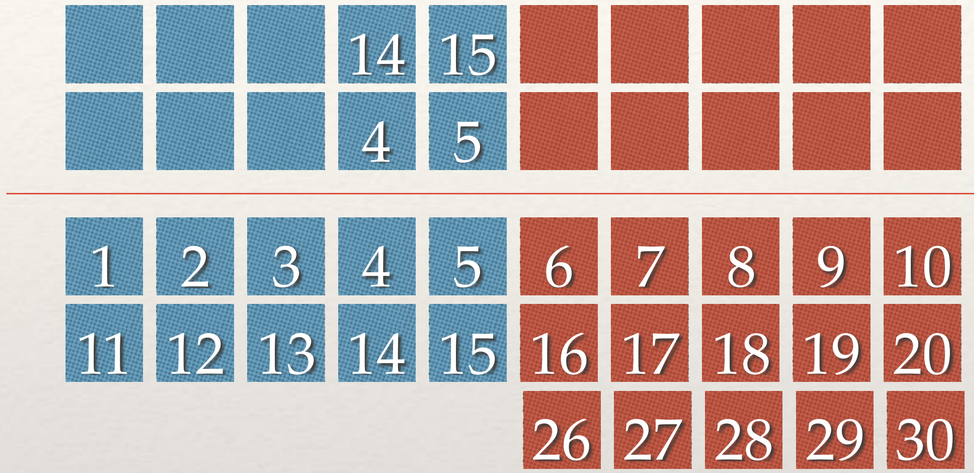
EVICT+TIME in Cache

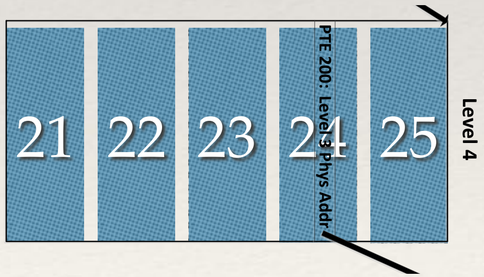
- Eviction done - let's do lookup



EVICT+TIME in Cache

- Uncached now

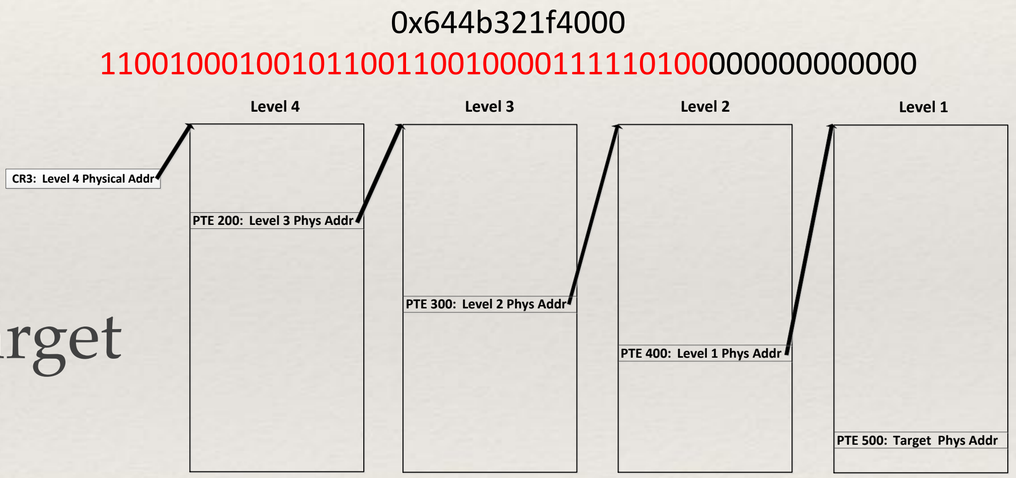




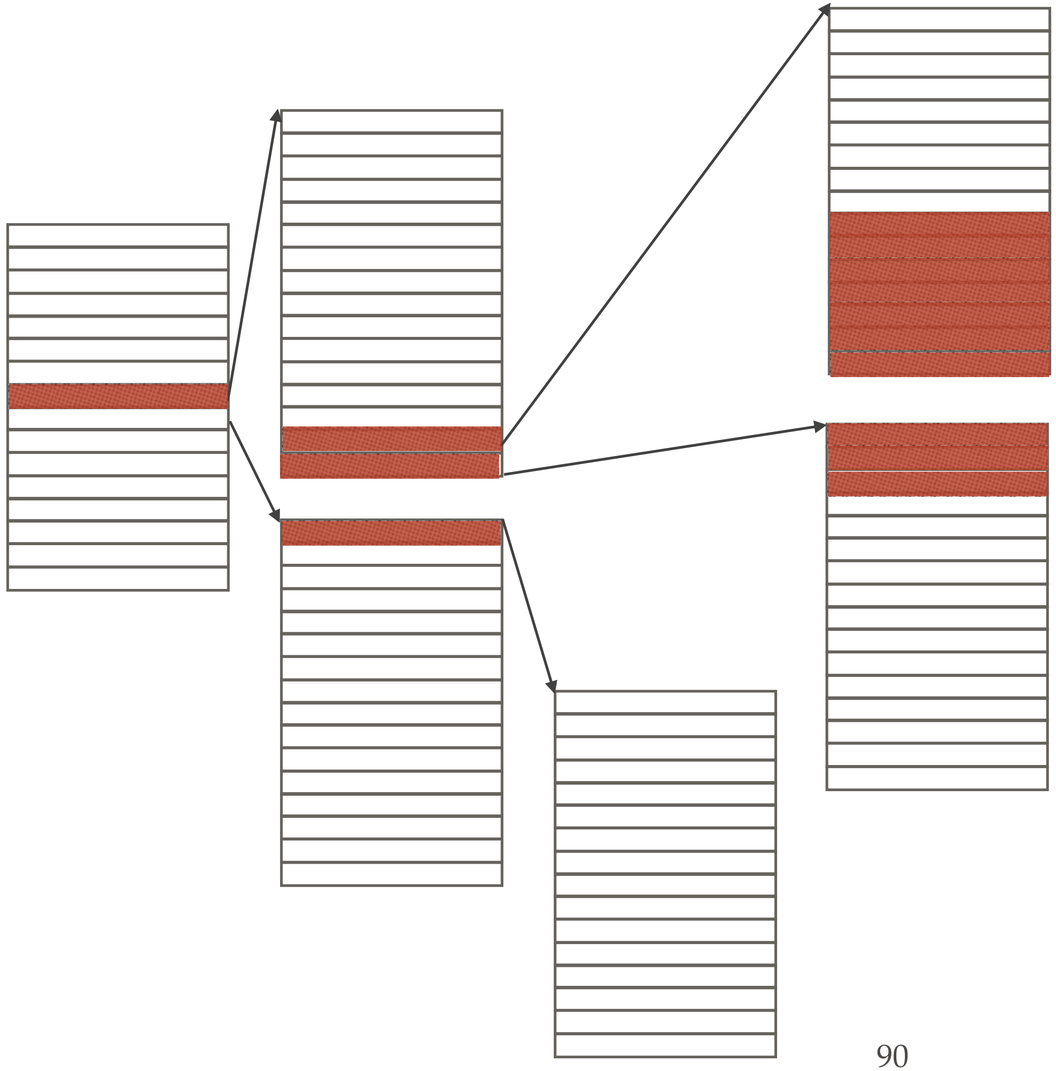
Pagetable in DRAM

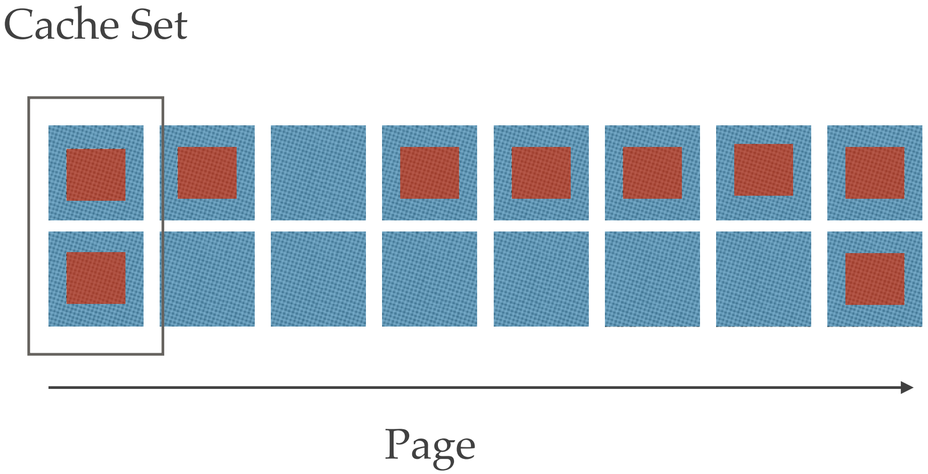
Ambiguity

- 2 sources of ambiguity remaining
- Which are the 4 levels
- 8 slots per cacheline
- We have to vary the target
- This varies the slots



Big picture: cached page tables



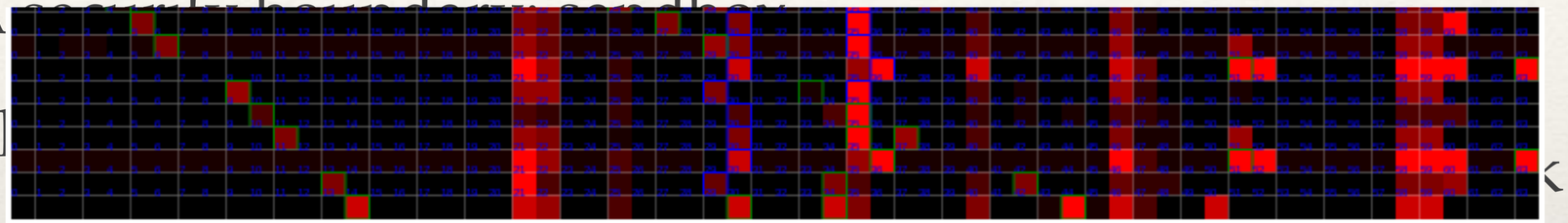


Outline

- Justification: ASLR
- Side Channels
- Pagetable walks
- CPU Caches
- EVICT+TIME
- **JavaScript**
- Results
- Demo

JavaScript Complications

- A
- Th
- Ir
- Lo
- Contiguous virtual address space



```
24.457 got level 4 - start slot 148, address 0x94000
24.993 got level 3 - start slot 295, address 0x24e94000
24.993 estimated remaining entropy 6 slot solutions: -1,-1,295,148
68.737 got level 4 - start slot 0, address 0x0
69.502 got level 3 - start slot 359, address 0x2ce00000
70.259 got level 2 - start slot 411, address 0x66ece00000
88.041 got level 1 - start slot 238, address 0x7766ece00000
88.041 estimated remaining entropy 0 slot solutions: 238,411,359,0
data: 0x7766ece00000, code slots: -1,-1,295,148, code: 0x7966e4e94000
```

k of JIT

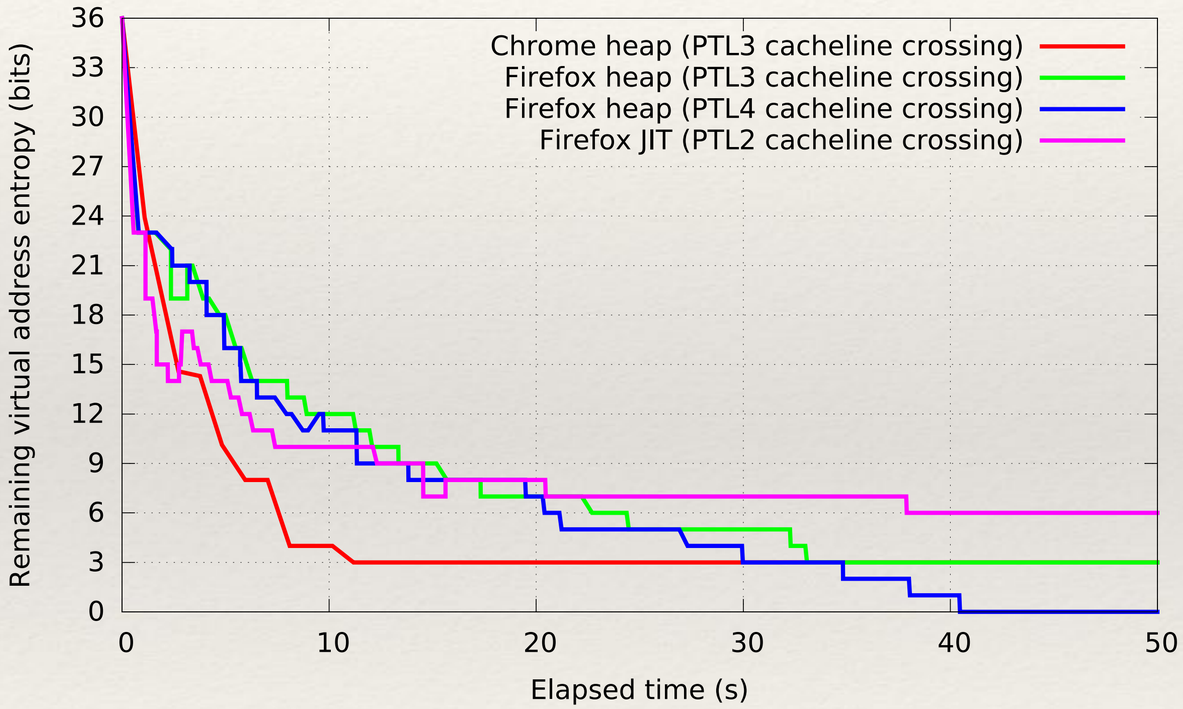
he iTLB

Prototypes in Firefox and Chrome

Outline

- Justification: ASLR
- Side Channels
- Pagetable walks
- CPU Caches
- EVICT+TIME
- JavaScript
- **Results**
- Demo

Results: speed



Results: tested microarchitectures

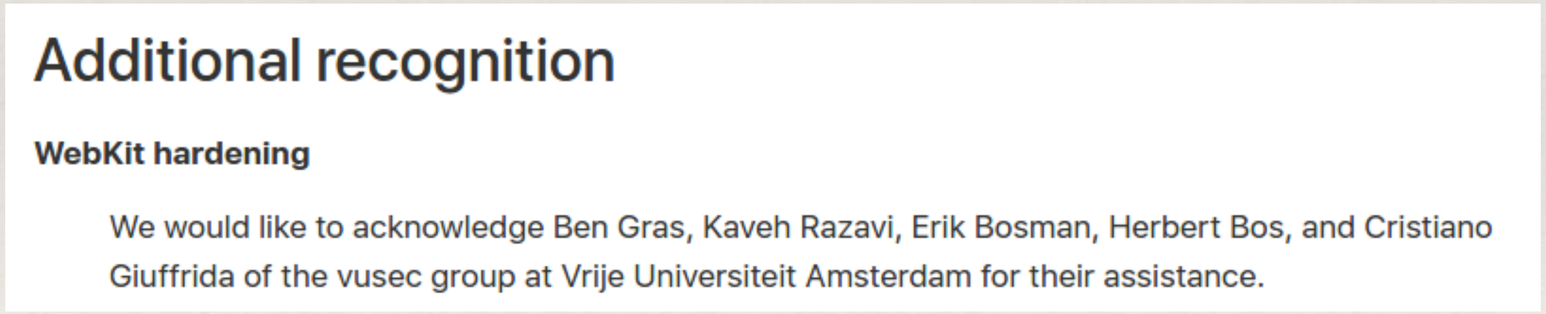
CPU Model	Microarchitecture	Year
Intel Xeon E3-1240 v5	Skylake	2015
Intel Core i7-6700K	Skylake	2015
Intel Celeron N2840	Silvermont	2014
Intel Xeon E5-2658 v2	Ivy Bridge EP	2013
Intel Atom C2750	Silvermont	2013
Intel Core i7-4500U	Haswell	2013
Intel Core i7-3632QM	Ivy Bridge	2012
Intel Core i7-2620QM	Sandy Bridge	2011
Intel Core i5 M480	Westmere	2010
Intel Core i7 920	Nehalem	2008
AMD FX-8350 8-Core	Piledriver	2012
AMD FX-8320 8-Core	Piledriver	2012
AMD FX-8120 8-Core	Bulldozer	2011
AMD Athlon II 640 X4	K10	2010
AMD E-350	Bobcat	2010
AMD Phenom 9550 4-Core	K10	2008
Allwinner A64	ARM Cortex A53	2016
Samsung Exynos 5800	ARM Cortex A15	2014
Samsung Exynos 5800	ARM Cortex A7	2014
Nvidia Tegra K1 CD580M-A1	ARM Cortex A15	2014
Nvidia Tegra K1 CD570M-A1	ARM Cortex A15; LPAE	2014

Outline

- Justification: ASLR
- Side Channels
- Pagetable walks
- CPU Caches
- EVICT+TIME
- JavaScript
- Results
- **Demo**

Reception

- Intel, AMD, ARM: CVE 2017-5925, 2017-5926, 2017-5927
- Chrome, Firefox, Safari, Edge: CVE-2017-5928
- Apple mitigation in iOS, Safari and tvOS updates



Additional recognition

WebKit hardening

We would like to acknowledge Ben Gras, Kaveh Razavi, Erik Bosman, Herbert Bos, and Cristiano Giuffrida of the vusec group at Vrije Universiteit Amsterdam for their assistance.

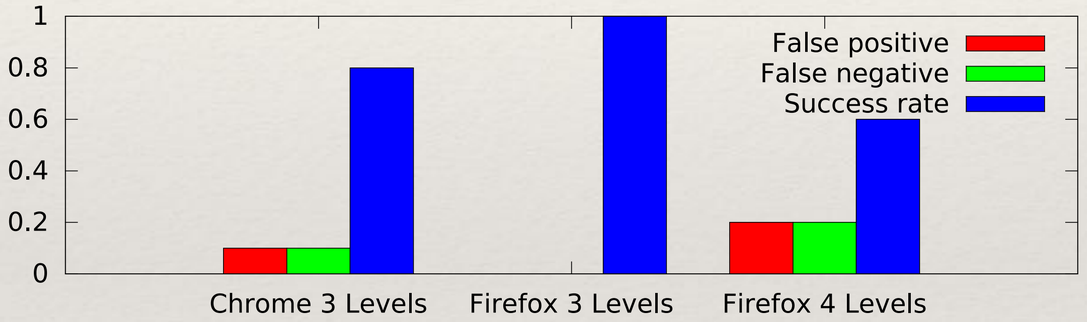
- Thank you NCSC-NL

Conclusion

- There is an micro-architectural MMU cache side channel
- Exploitable from JavaScript
- It breaks ASLR in sandboxed environments

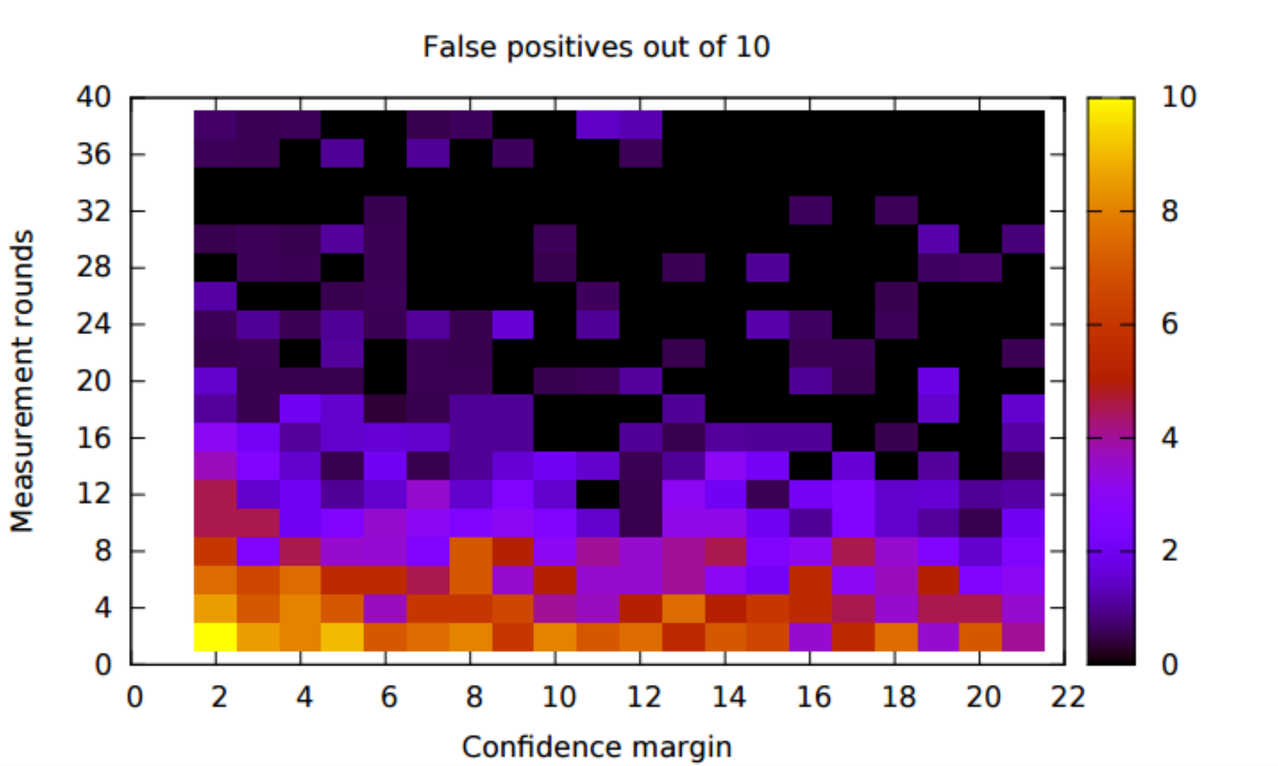
- Project page: https://www.vusec.net/projects/anc/
- Native code: https://github.com/vusec/revanc
- Twitter @vu5ec

Results: reliability



Results: noise

- Repeat measurements vs confidence margin



Who are we

- VUSec systems security academic research group
- Defensive & offensive security projects using systems techniques
- Aim to publish at S&P, CCS, NDSS, Usenix Security, SOSP, OSDI
- VU University in Amsterdam
- I am an intern with Cisco ASRG now