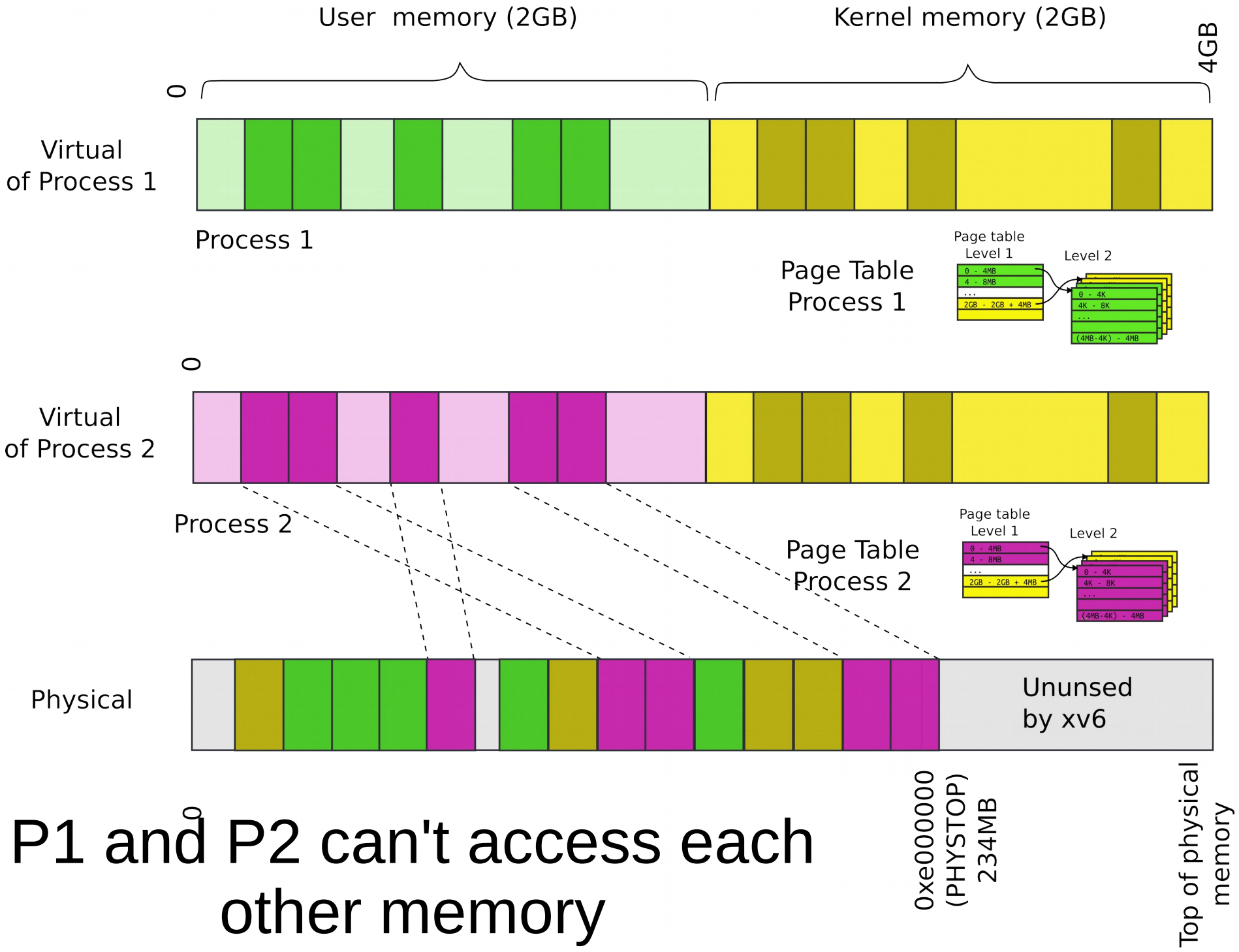
CS/EE3810: Computer Organization

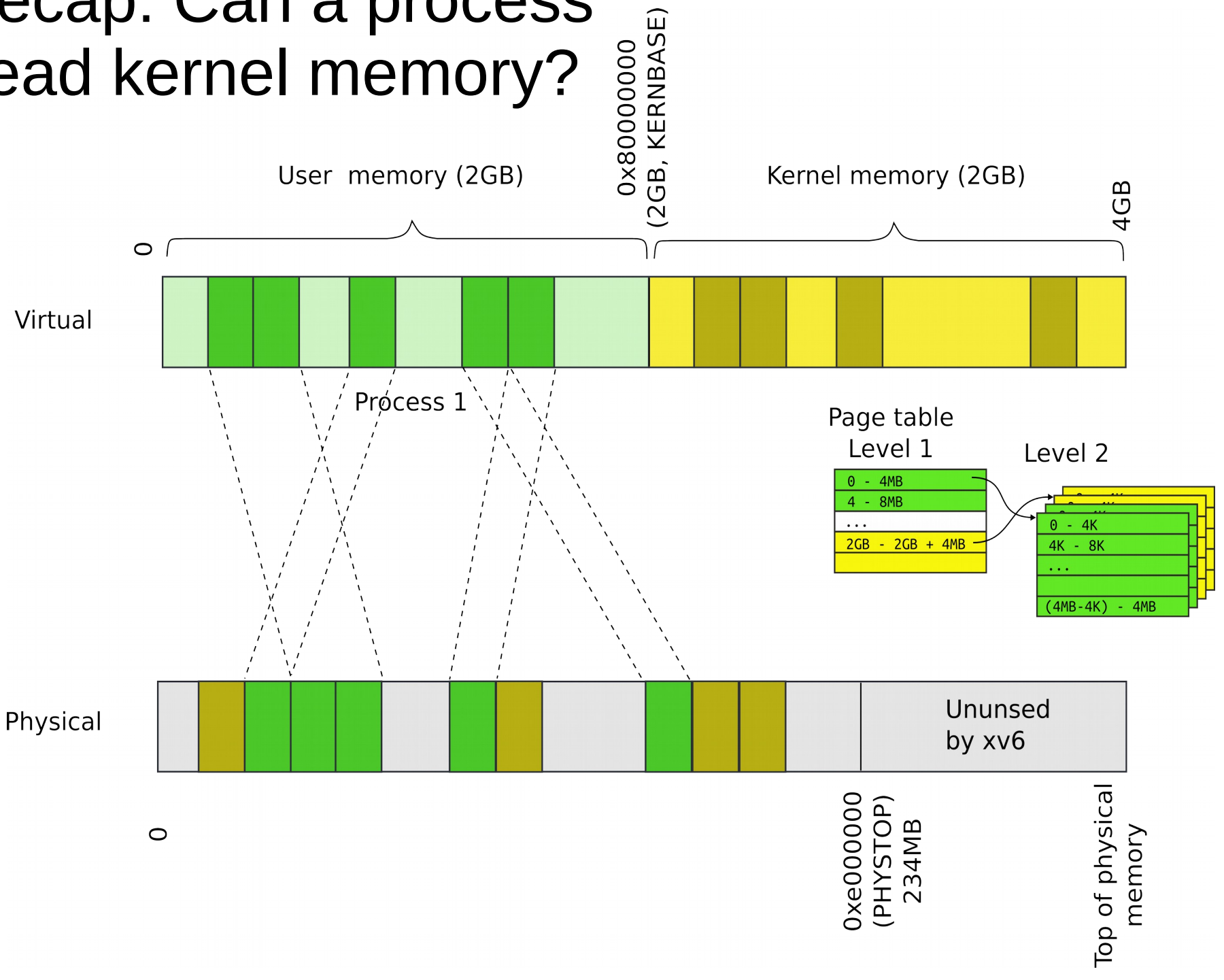
Lecture 18: Side channel attacks (Meltdown)

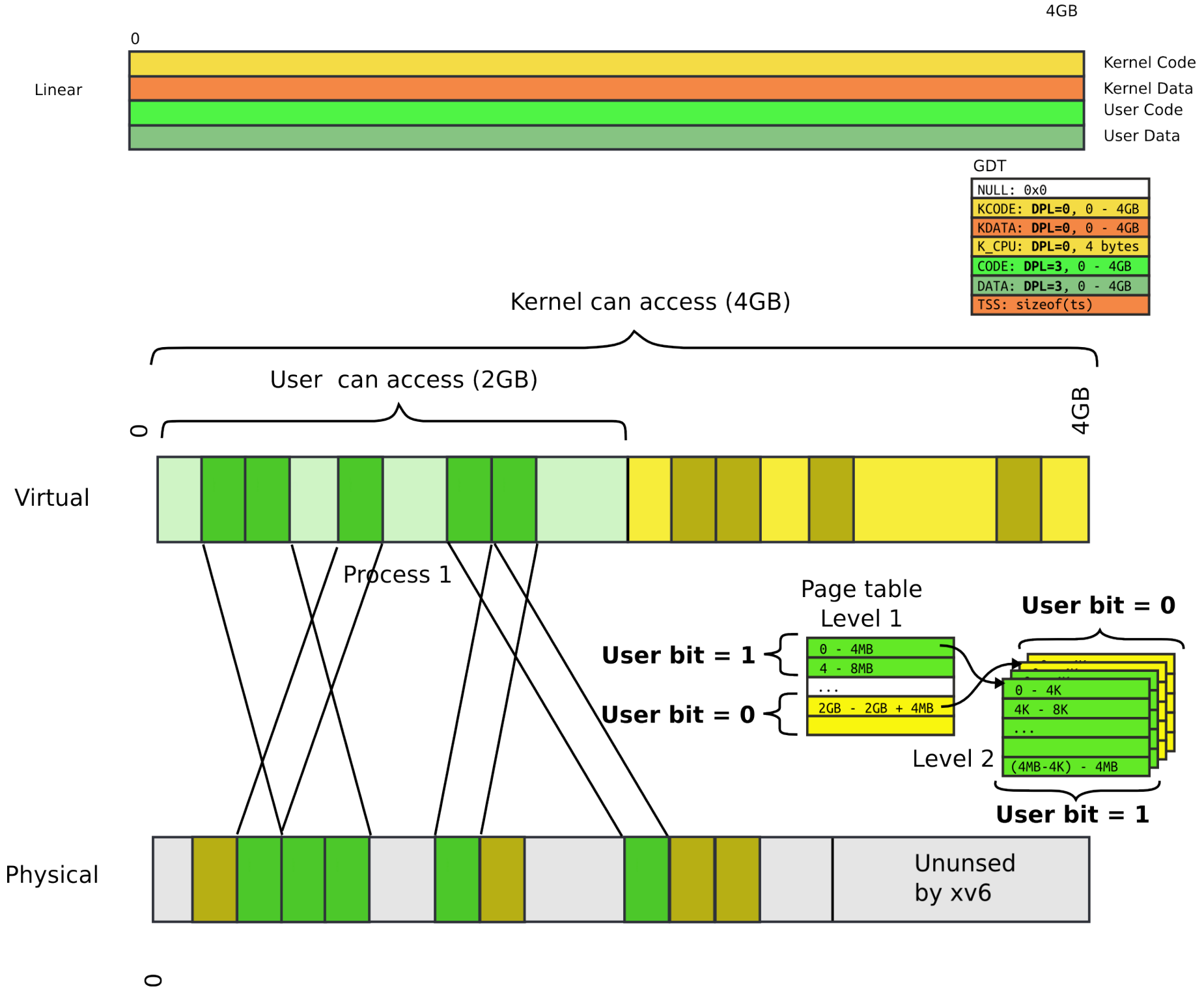
Anton Burtsev
December, 2022



P1 and P2 can't access each other memory

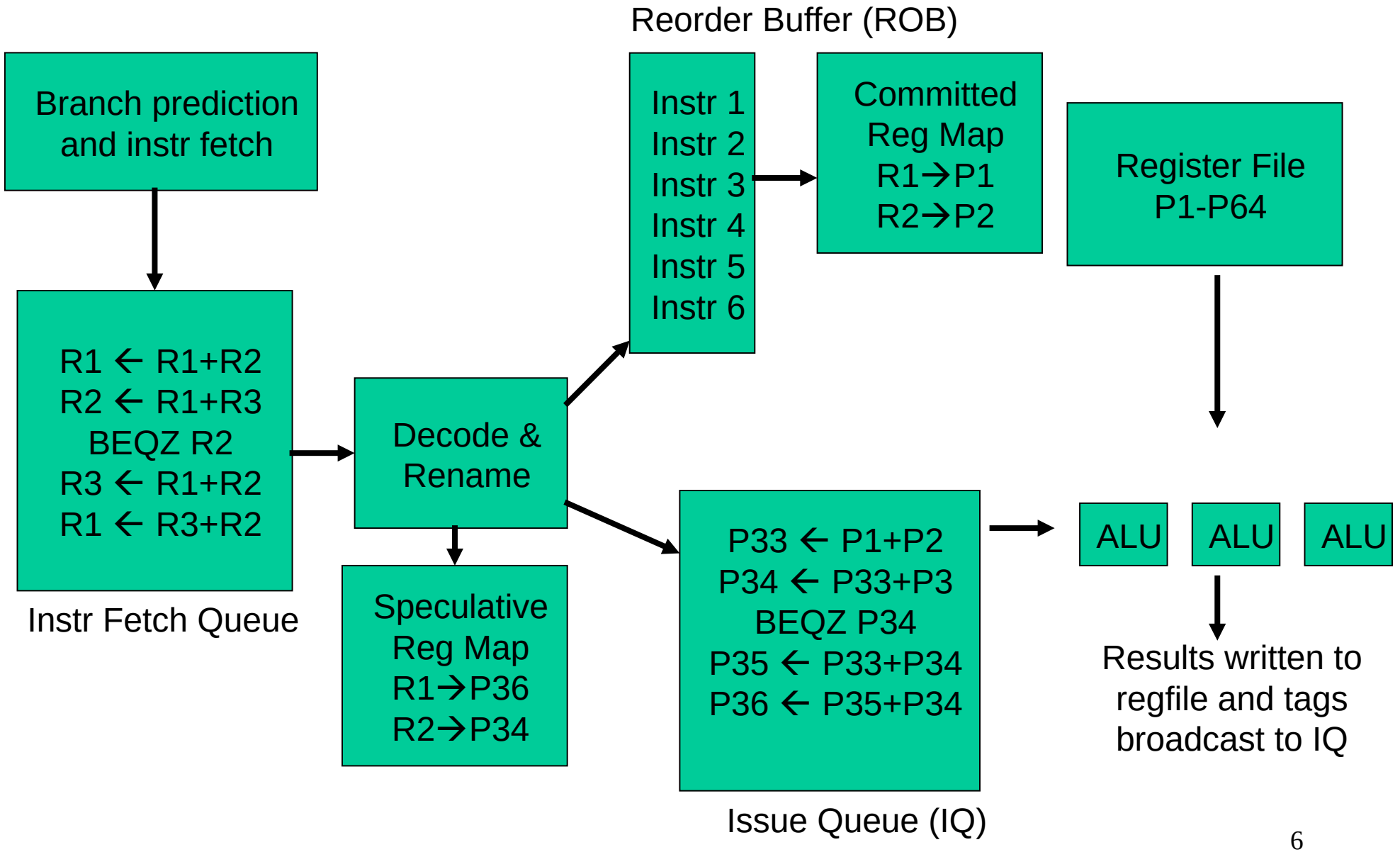
Recap: Can a process read kernel memory?

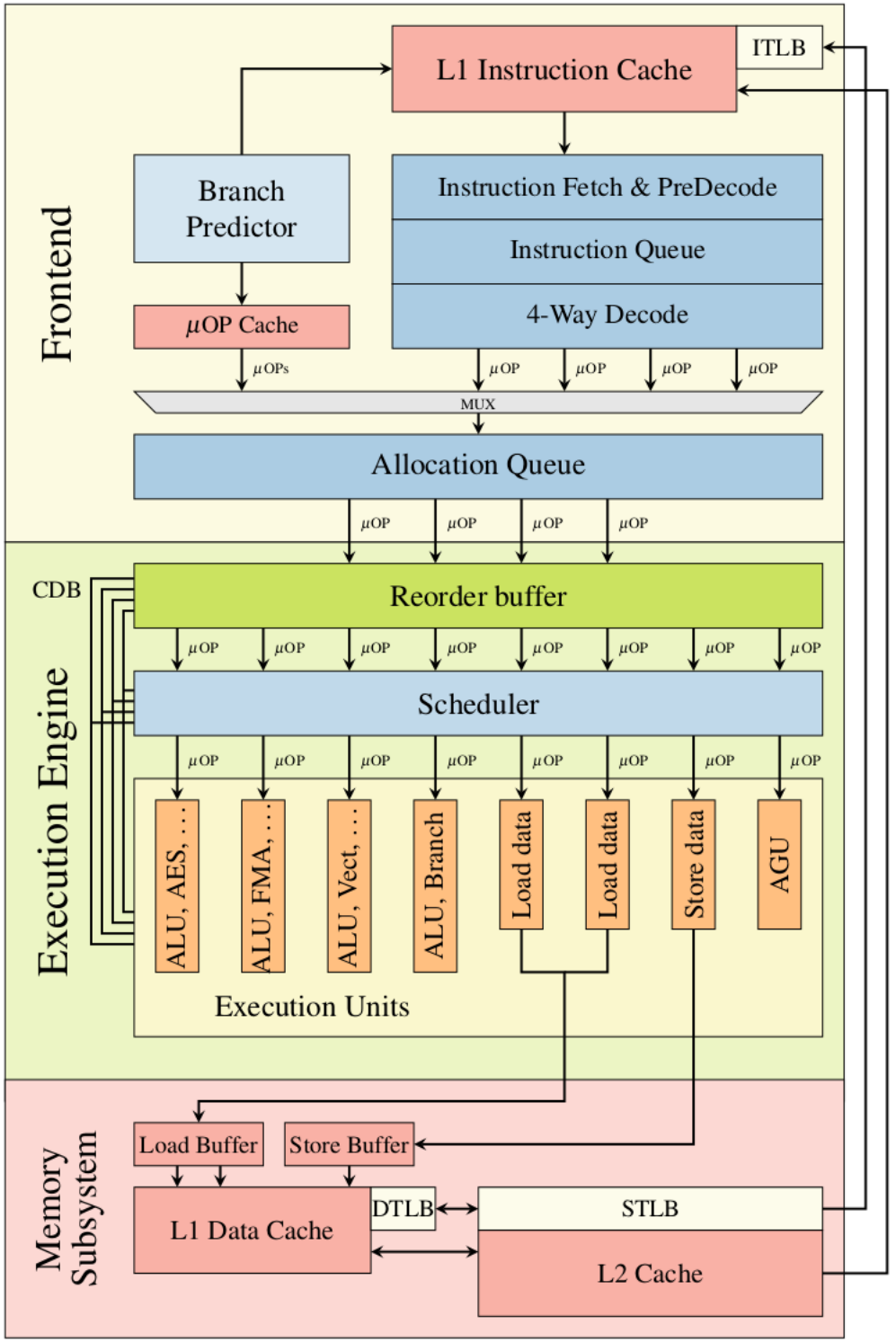



4GB
0
Linear
Kernel Code
Kernel Data
User Code
User Data
GDT
NULL: 0x0
KCODE: DPL=0, 0 - 4GB
KDATA: DPL=0, 0 - 4GB
K_CPU: DPL=0, 4 bytes
CODE: DPL=3, 0 - 4GB
DATA: DPL=3, 0 - 4GB
TSS: sizeof(ts)
Kernel can access (4GB)
User can access (2GB)
0
4GB
Virtual
Process 1
Page table
Level 1
User bit = 1
User bit = 0
0 - 4MB
4 - 8MB
...
2GB - 2GB + 4MB
User bit = 0
0 - 4K
4K - 8K
...
(4MB-4K) - 4MB
Level 2
User bit = 1
Physical
Ununsed
by xv6
0

Page tables and protection

The Alpha 21264 Out-of-Order Implementation

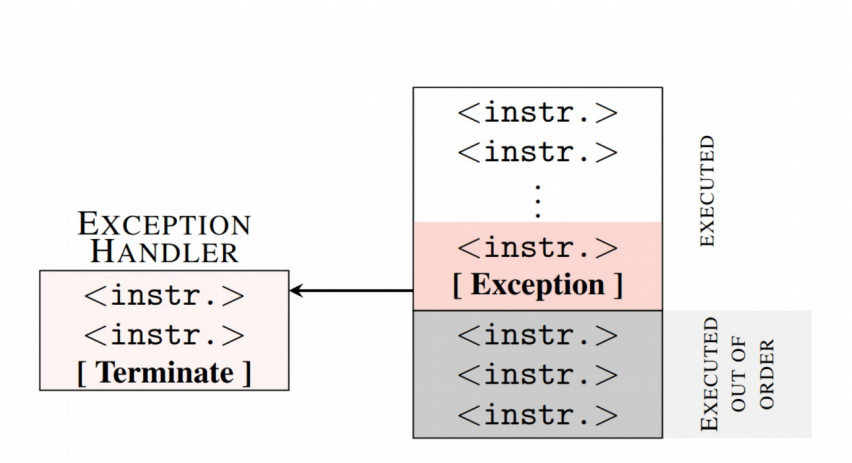




Skylake (simplified)

Exceptions and speculation

```
1 raise_exception();
2 // the line below is never reached
3 access(probe_array[data * 4096]);
```



Core of the attack

```
1 ; rcx = kernel address, rbx = probe array
2 xor rax, rax
3 retry:
4 mov al, byte [rcx]
5 shl rax, 0xc
6 jz retry
7 mov rbx, qword [rbx + rax]
```

Cache access time

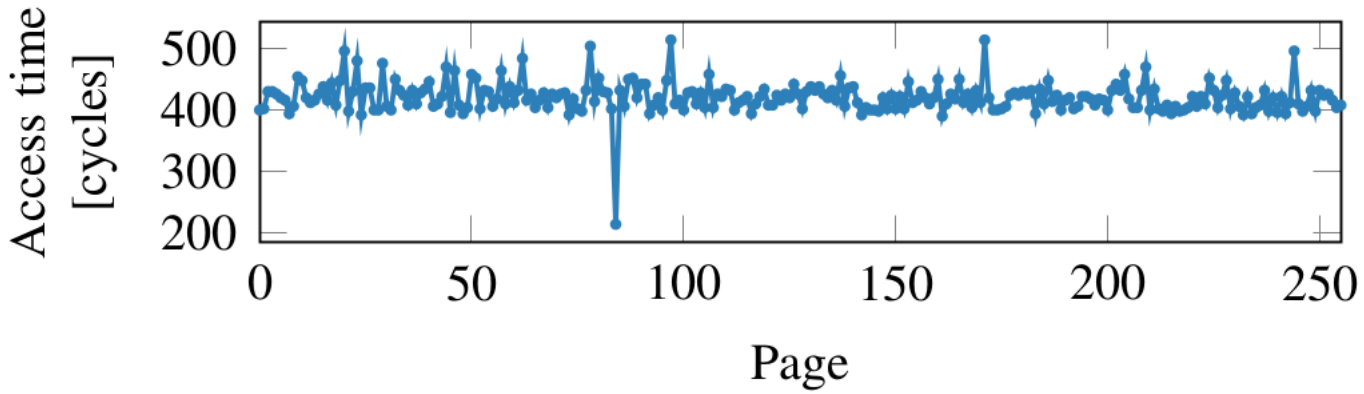


Figure 4: Even if a memory location is only accessed during out-of-order execution, it remains cached. Iterating over the 256 pages of `probe_array` shows one cache hit, exactly on the page that was accessed during the out-of-order execution.

- `data` = 84

Core of the attack

```
1 ; rcx = kernel address, rbx = probe array
2 xor rax, rax
3 retry:
4 mov al, byte [rcx]
5 shl rax, 0xc
6 jz retry
7 mov rbx, qword [rbx + rax]
```

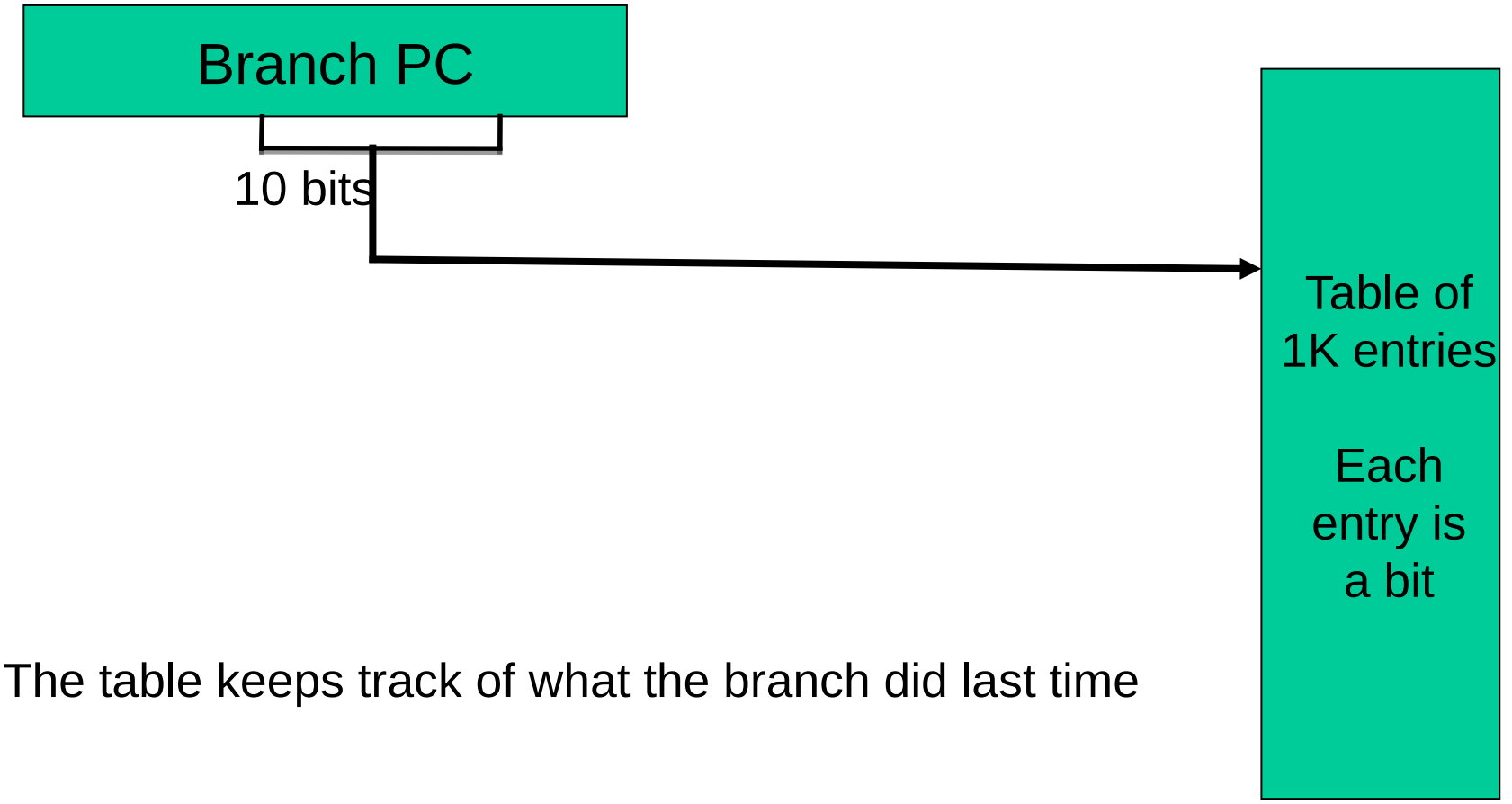
Exception suppression

2-Bit Bimodal Prediction

- For each branch, keep track of what happened last time and use that outcome as the prediction
- What are prediction accuracies for branches 1 and 2 below:

```
while (1) {
        for (i=0;i<10;i++) {                    branch-1
            …
        }
        for (j=0;j<20;j++) {                    branch-2
           …
        }
}
```

Bimodal 2-Bit Predictor



The table keeps track of what the branch did last time

Thank you!

Gadget

```
if (x < array1_size)
    y = array2[array1[x] * 4096];
```

Exceptions and speculation