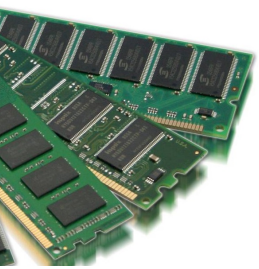
Abusing hardware for fun and profit

Agenda

- Cache-based Covert channels w/ demo
- Spectre and Meltdown from covert channels

Process isolation + OS (CS 423)



Threading, etc

… OS paging …

Virtual memory

0x00000000

Process Memory

0xffffffff

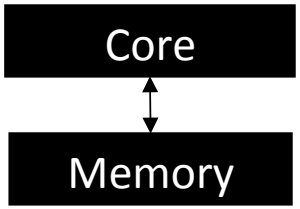
…OS services…

Communication to other processes v
e.g., #include <sockets.h>, send(), re

Programs run on processors

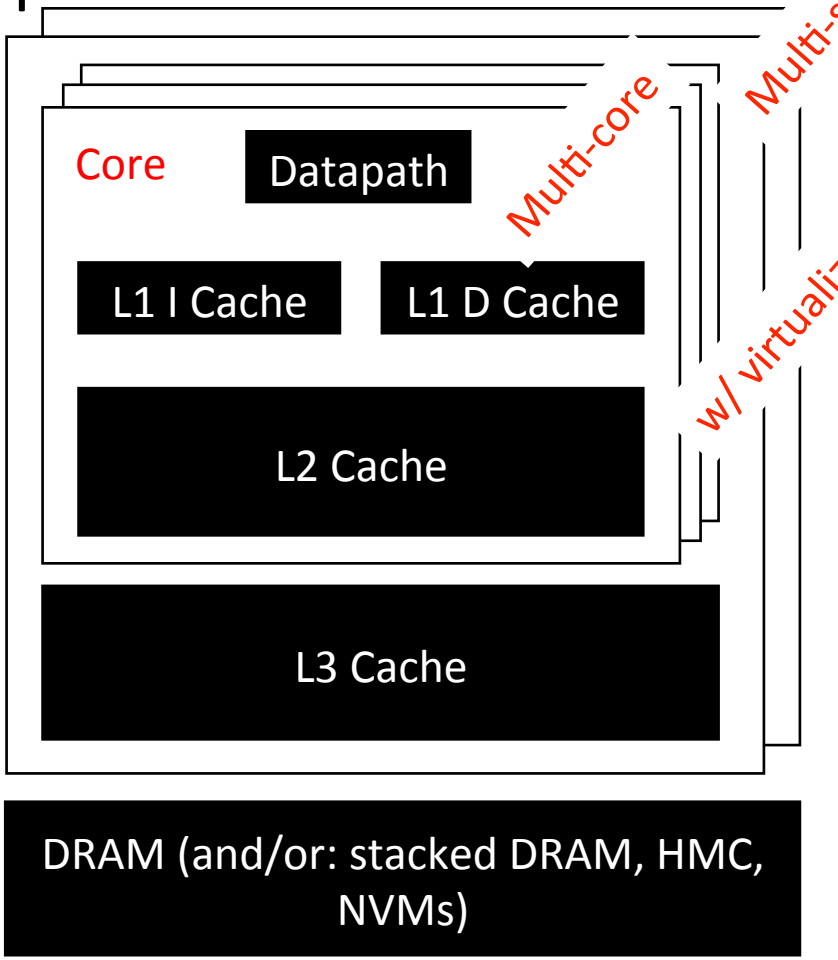
Cache = on-chip memory, faster to access than

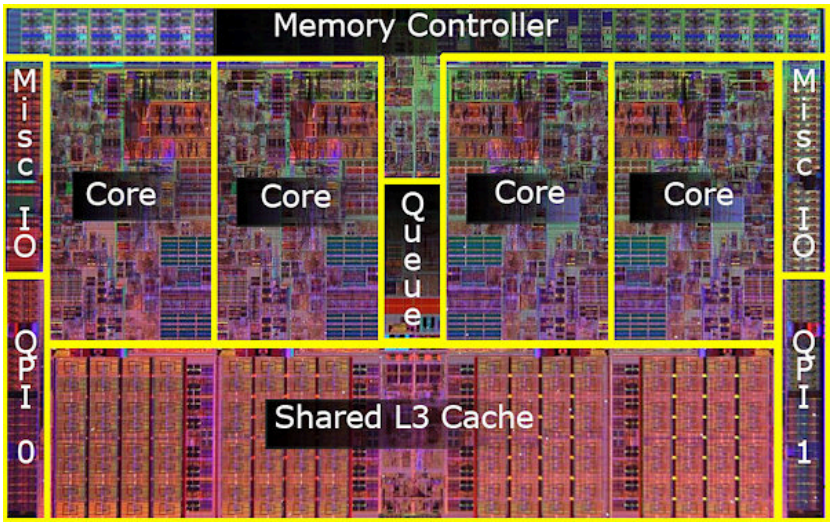
Processor that OS would have you see …



OS swaps work on/off

- Real processors





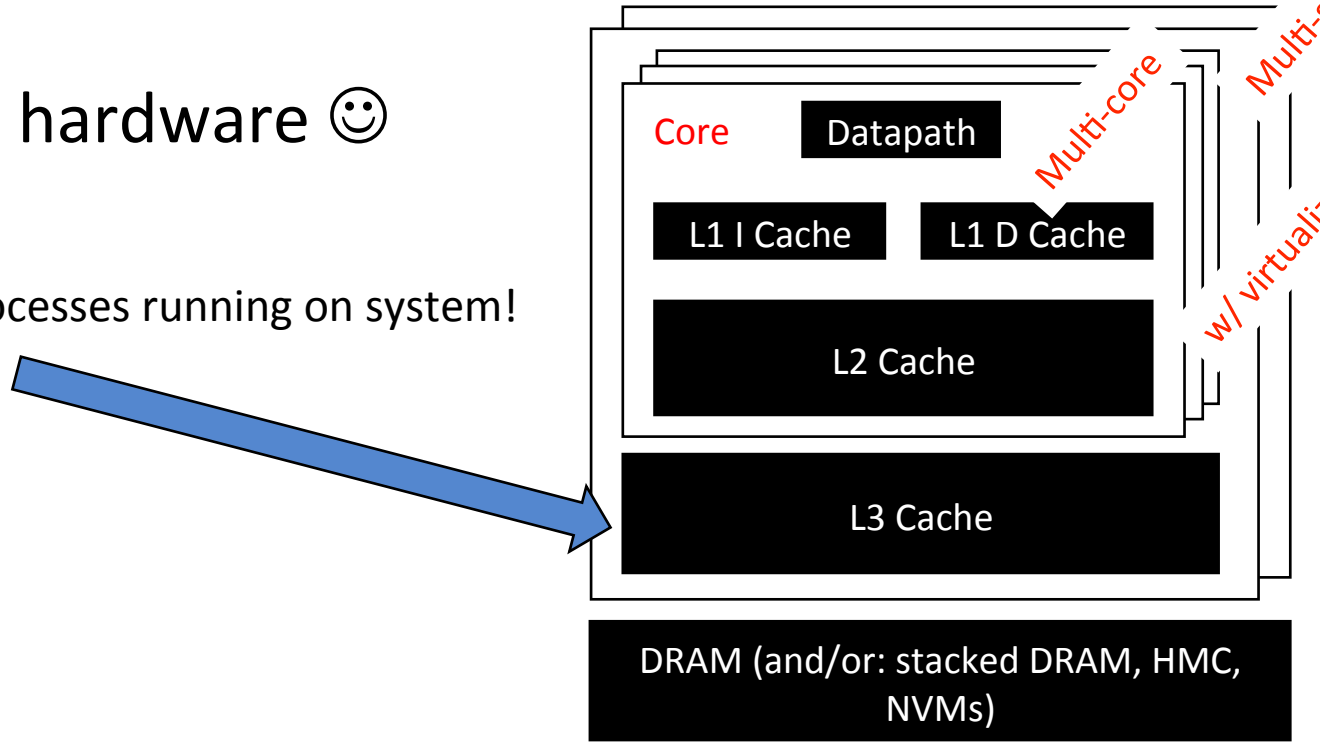
Hardware Covert Channels

Talk to your friends without the OS's help or knowledge

No header files → no socket/etc, no OS-sanctioned communication

Exploit properties of your hardware ☺

L3 cache shared by all processes running on system!



Processor caches

tivation

Programs have locality

Memory access cost ∝ memory size

ck placement/replacement policies

us where blocks can live and when

e-facing API: Read(addr)
Write(addr, word)

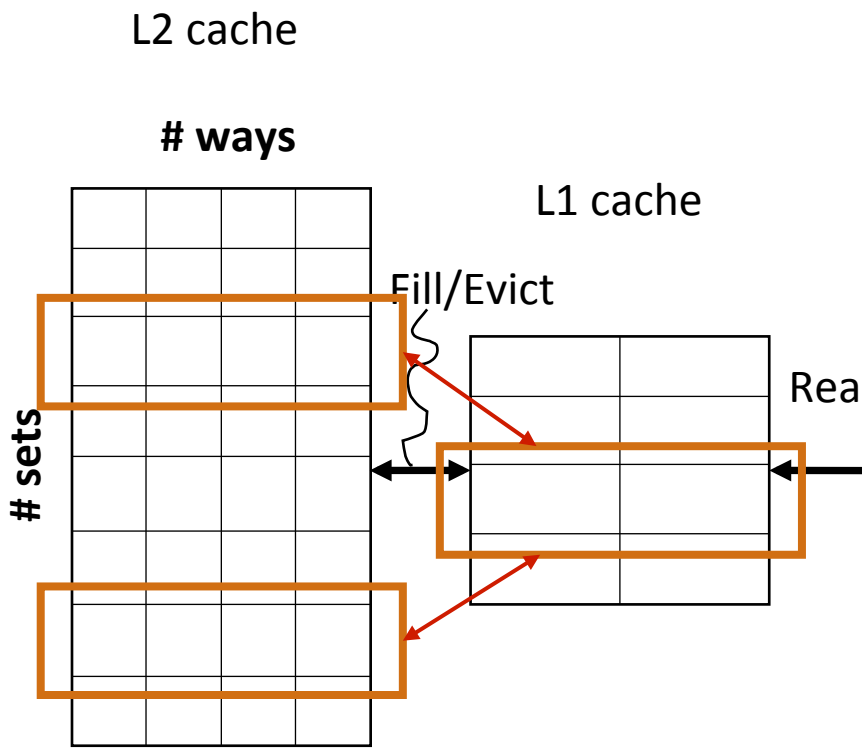
kend API: Evict(addr)
Fill(addr, line)

Which set?

2 or more L_{i+1} cache sets statically map to

Which way?

Determined by replacement policy.



Why is cache design relevant?

Two processes can agree on “dead drops” on the processor hardware, to pass information under the OS’s nose

We made a virtual “wire”, now what?

Remember TCP?

Virtual wire +
de-noising +
re-transmission +
wrapper API
=

	Example
User / Application	Firefox browser
Application layer	HTTP
Transport layer	TCP
Network layer	IP
Link layer	Ethernet driver
Hardware layer	~~Ethernet~~ Cache pressure!

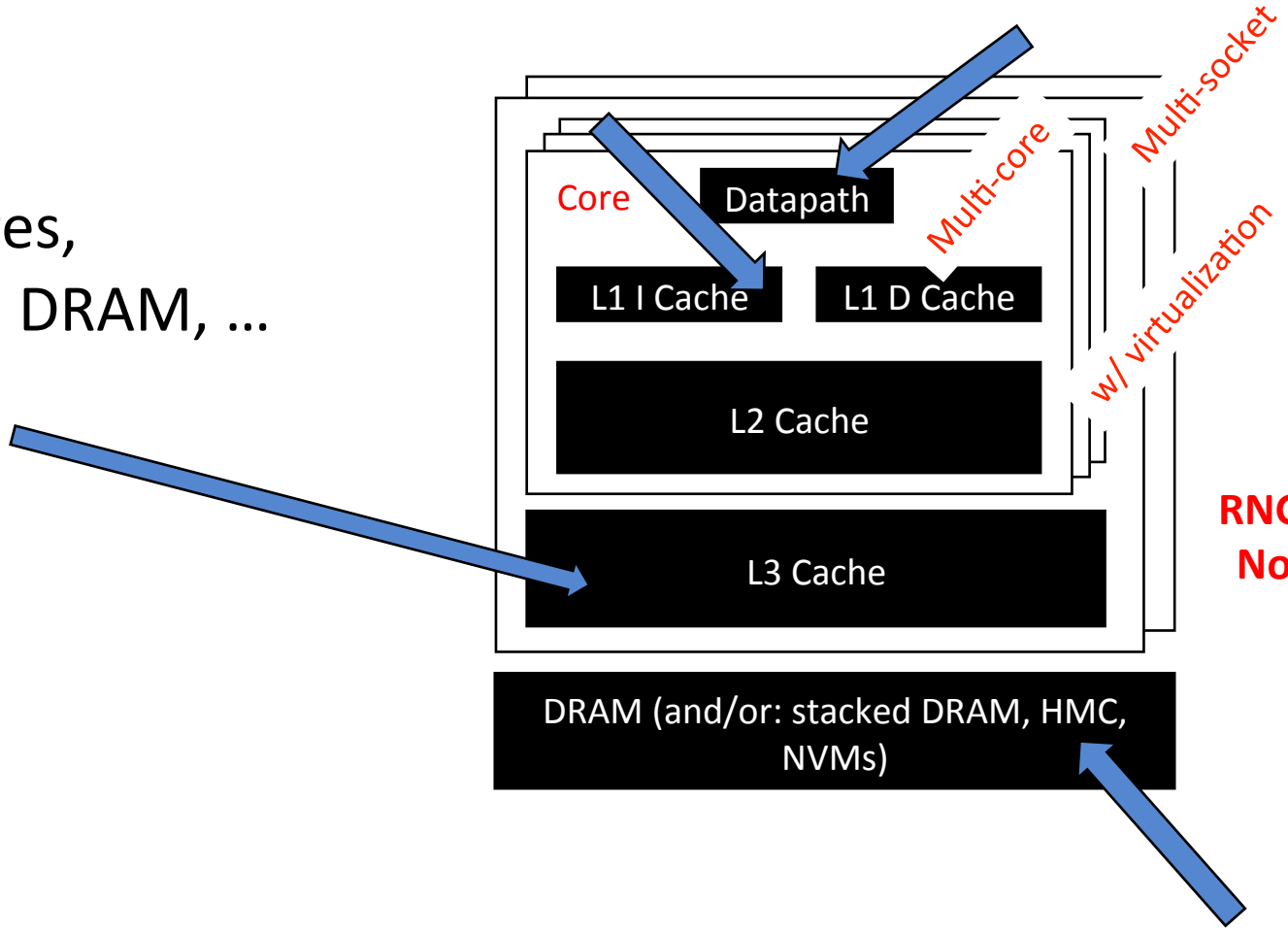
Demo

Fun! How else can I do this?

Processes share …

branch predictors, cores,

caches, RNG modules, DRAM, …



All of which can (and have) been turned into “virtual wires”

And they are pretty fast (~ 1 Mb/sec on the high end)

Practical uses

Talk to your friends for fun

Malware can inter-communicate w/o OS realizing it

Different VMs sharing the same box on (e.g.) Amazon AWS can talk

Side channel attacks

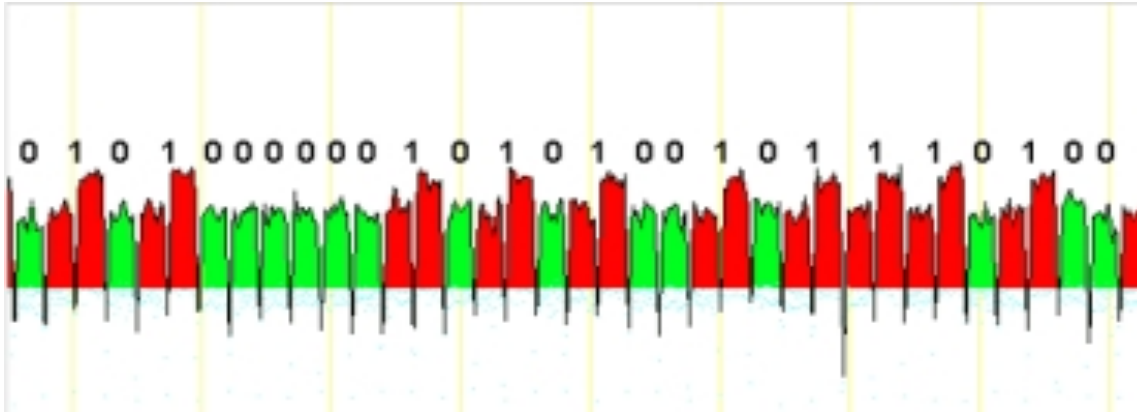
- Learn private information about co-resident processes

Side channel attacks

- Shared resource pressure can also lead to side channel attacks
- E.g., RSA encryption msg = $\text{Decrypt}_{key}(\text{Encrypt}_{key}(msg))$

```
uareMult(x, e, N):
et e_n, ..., e_1 be the bits of e
 ← 1
or i = n down to 1 {
  y ← Square(y)                (S)
  y ← ModReduce(y, N)          (R)
  if e_i = 1 then {
    y ← Mult(y, x)             (M)
    y ← ModReduce(y, N)        (R)
  }

eturn y
```



ngredients

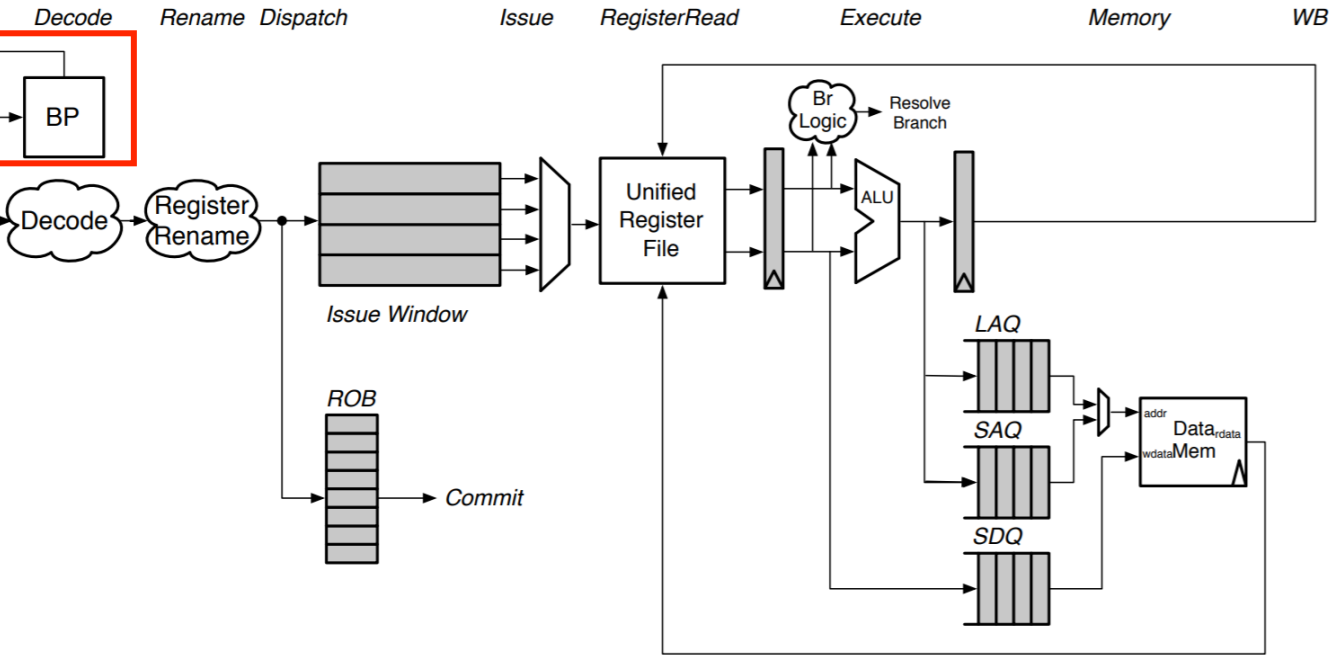
Cover channel

Speculation

OS mapped to process address space (for Meltdown)

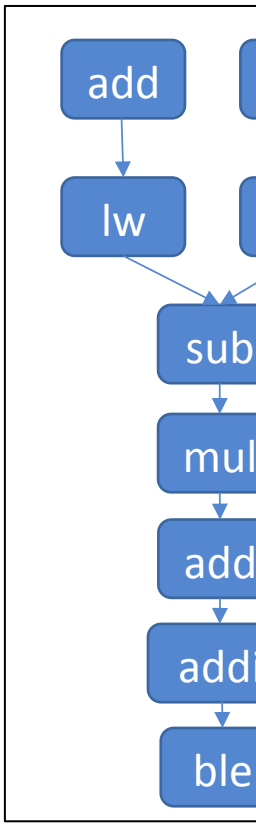
Branch prediction (for Spectre)

Out of order, speculative processor core




```
  xor sum, 0, 0
  xor d,   0, 0

loop:
  add $t0, d, &P1
  lw  P1d, 0($t0)
  add $t0, d, &P2
  lw  P2d, 0($t0)
  sub $t0, P1d, P2d
  mul $t0, $t0, $t0
  add sum, sum, $t0
  addi d, d, 1
  ble loop, d, LEN
post:
  blt end, best, sum
  add best, sum, 0
end:
```



OM spec; https://github.com/ccelio/riscv-boom-doc/raw/gh-pages/boom-spec.pdf