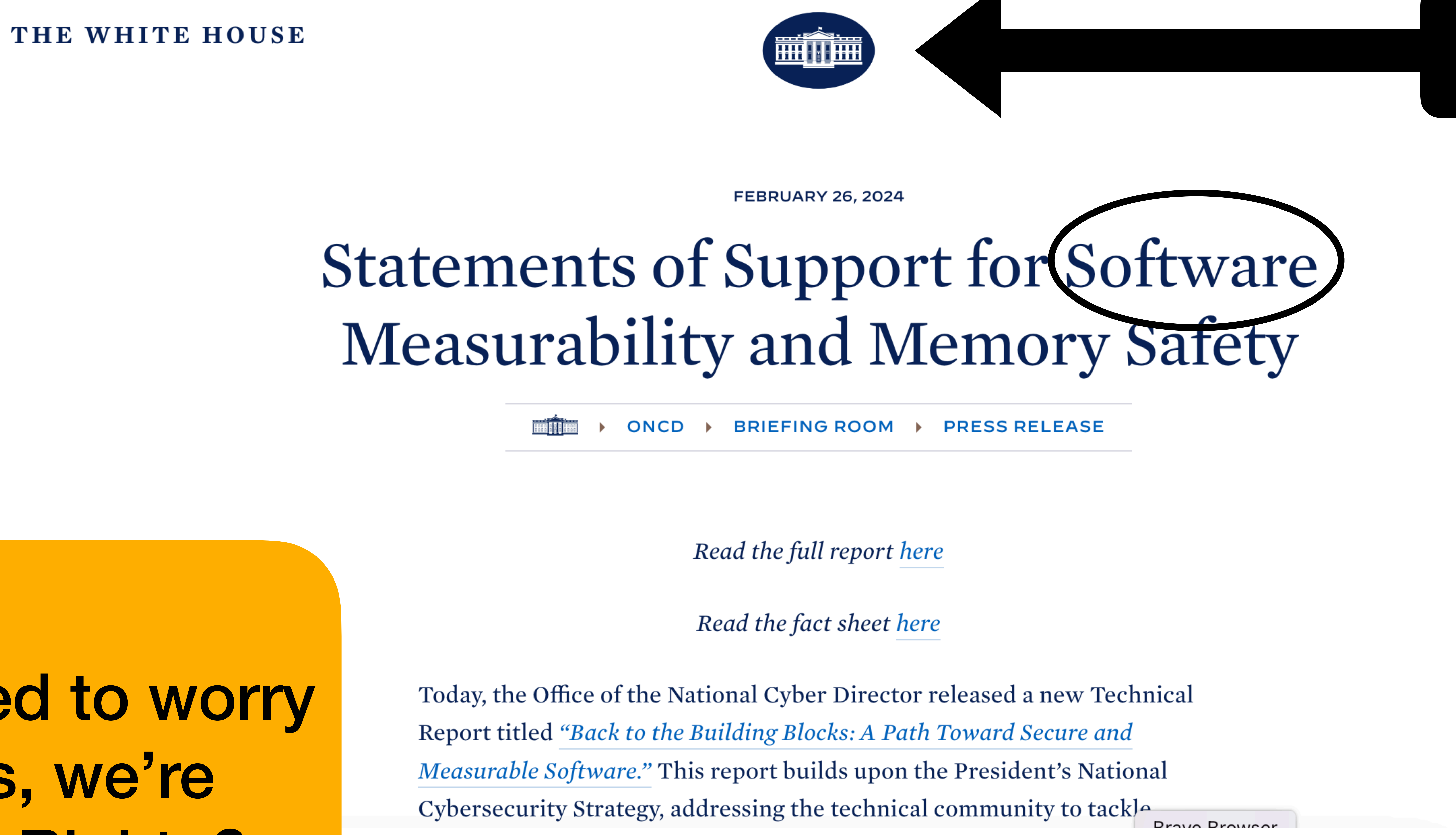
Hardware Capabilities

“Software uses hardware, so…”



THE WHITE HOUSE

FEBRUARY 26, 2024

Statements of Support for Software Measurability and Memory Safety

ONCD ▸ BRIEFING ROOM ▸ PRESS RELEASE

Read the full report here

Read the fact sheet here

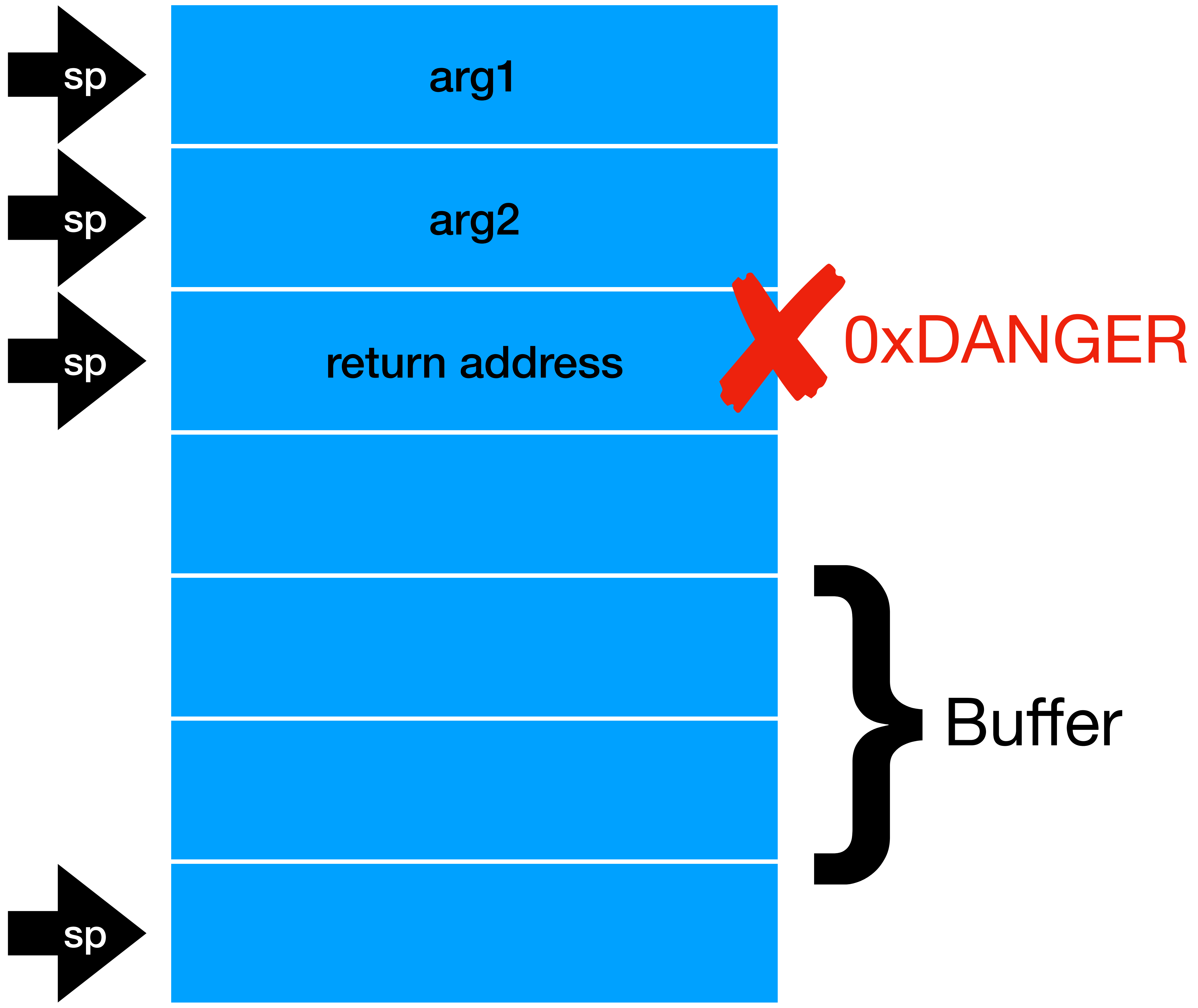
Today, the Office of the National Cyber Director released a new Technical Report titled *“Back to the Building Blocks: A Path Toward Secure and Measurable Software.”* This report builds upon the President’s National Cybersecurity Strategy, addressing the technical community to tackle

Joe Biden!

We don’t need to worry about this, we’re architects! Right..?

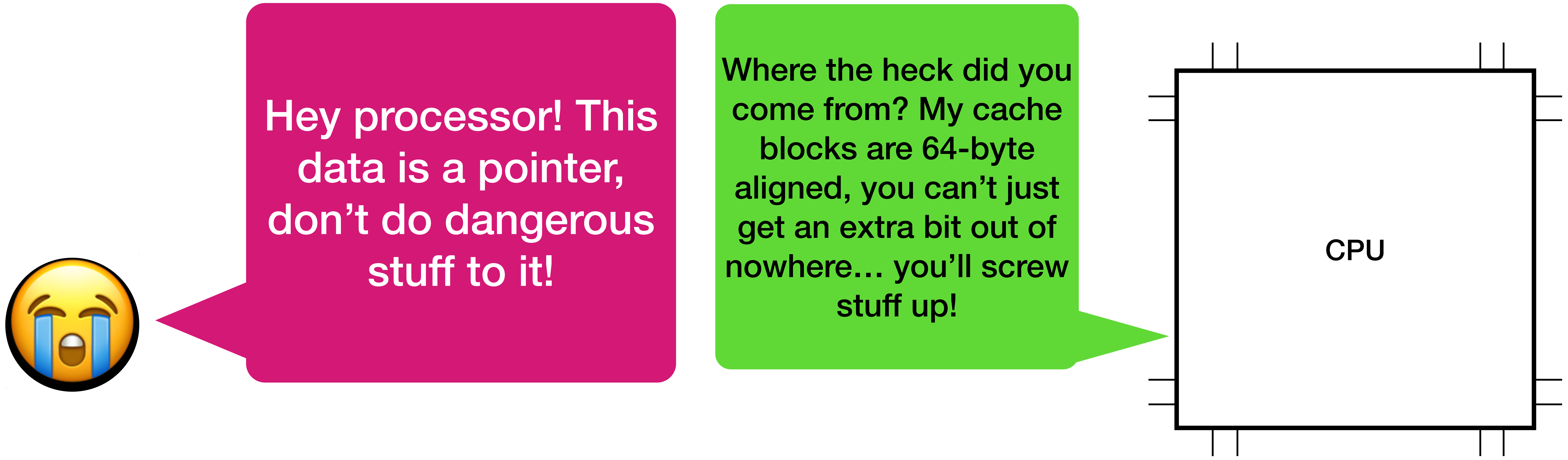
“Software uses hardware, so…”

```
int foo(int arg1, int arg2) {
  int buffer[arg1];
  for (int i = 0; i < arg2; i++) {
    buffer[i] = /* something */;
  }
  return 0;
}
```



“Software uses hardware, so…”

- Hardware doesn’t know about data types…
- What if we just had one extra bit per data word…


Hey processor! This data is a pointer, don’t do dangerous stuff to it!
Where the heck did you come from? My cache blocks are 64-byte aligned, you can’t just get an extra bit out of nowhere… you’ll screw stuff up!
CPU

“Software uses hardware, so…”

- We can arrange these extra bits in a *table*!
- Any time we access memory, we’ll also fetch these extra bits
- We can even cache them!

extra bit
extra bit
extra bit
extra bit
extra bit
I already have so many tables, and so many caches to protect these tables…
CPU

Remember page tables?

- Each process gets a table of virtual addresses that it can access
- Each entry maps to a physical address
- The TLB is used to cache page table entries

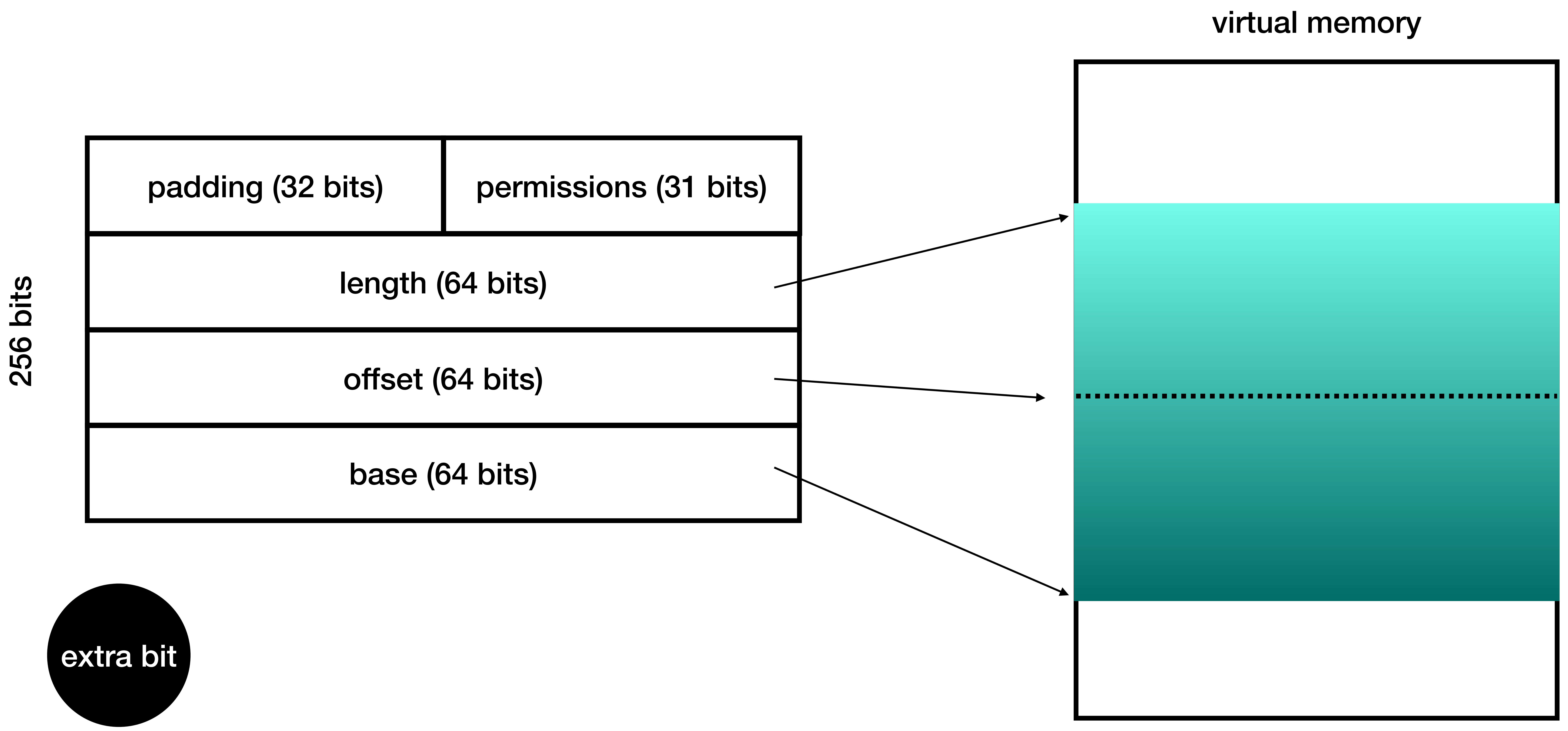
Hey processor! The "process" abstraction isn't all that great!

extra bit

“Software uses hardware, so…”

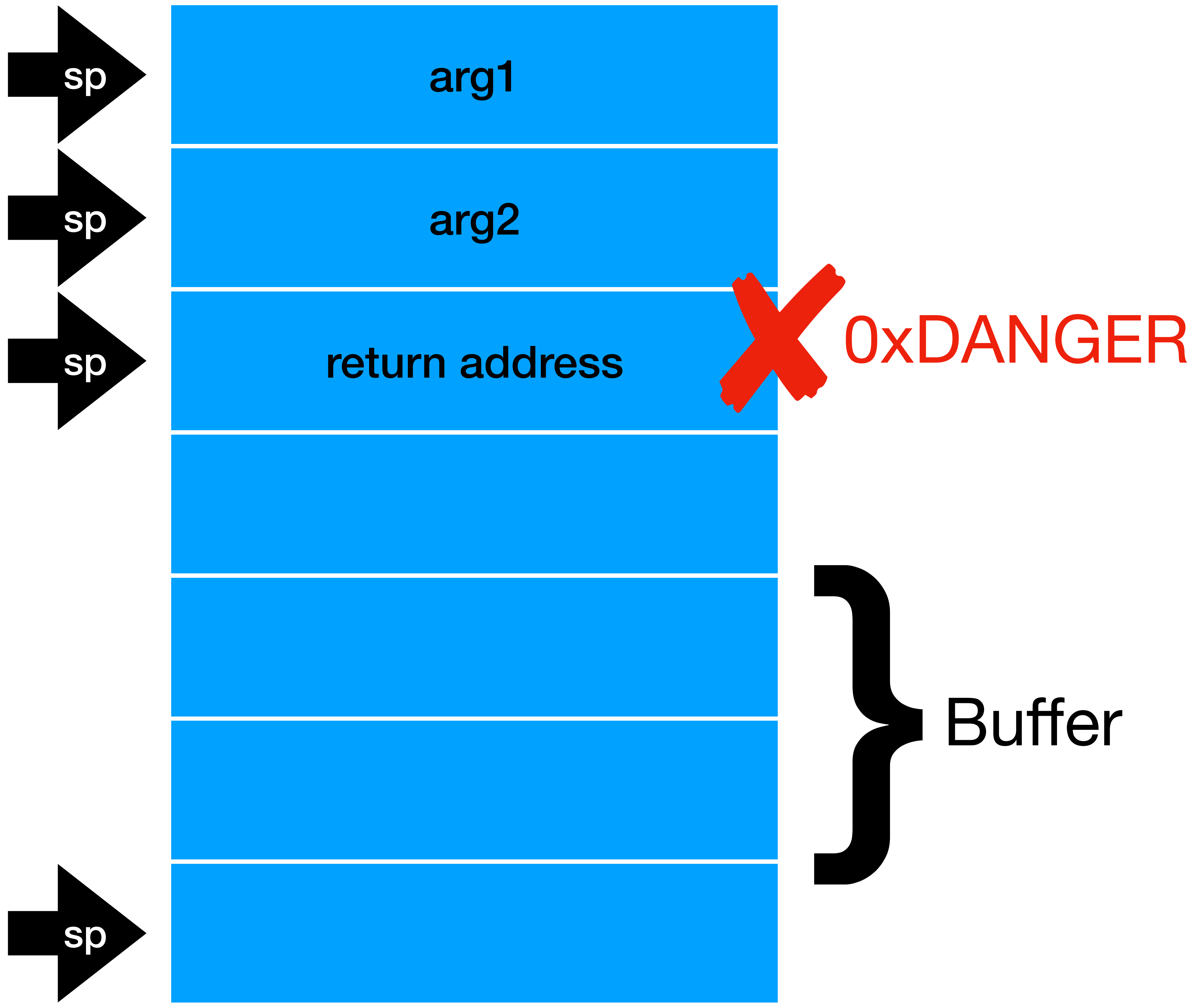
- We want better hardware abstractions to implement security primitives
    - Tag “data references” to say that they are a capability, now you have security!
    - Change the purpose/role of the MMU (memory management unit: TLB, etc.) to better suit control flow

Capabilities!

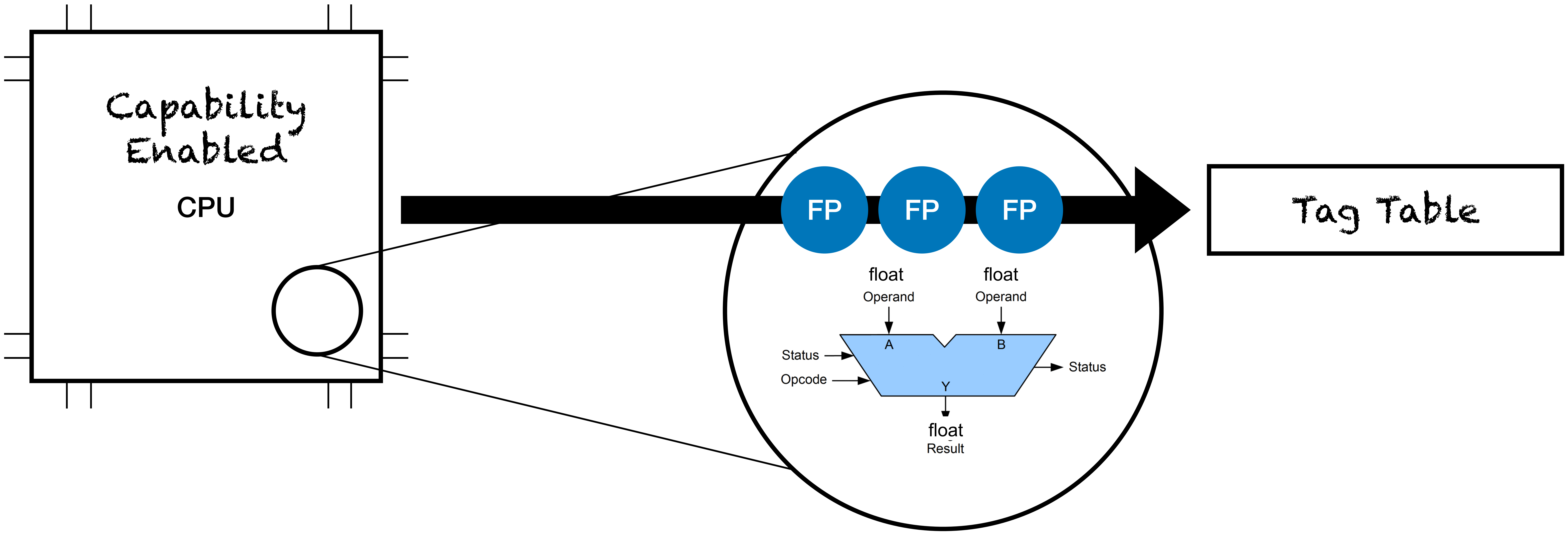


Chat with your neighbor!

```
int foo(int arg1, int arg2) {
  int buffer[arg1];
  for (int i = 0; i < arg2; i++) {
    buffer[i] = /* something */;
  }
  return 0;
}
```



Capability Hardware



Capability Processors

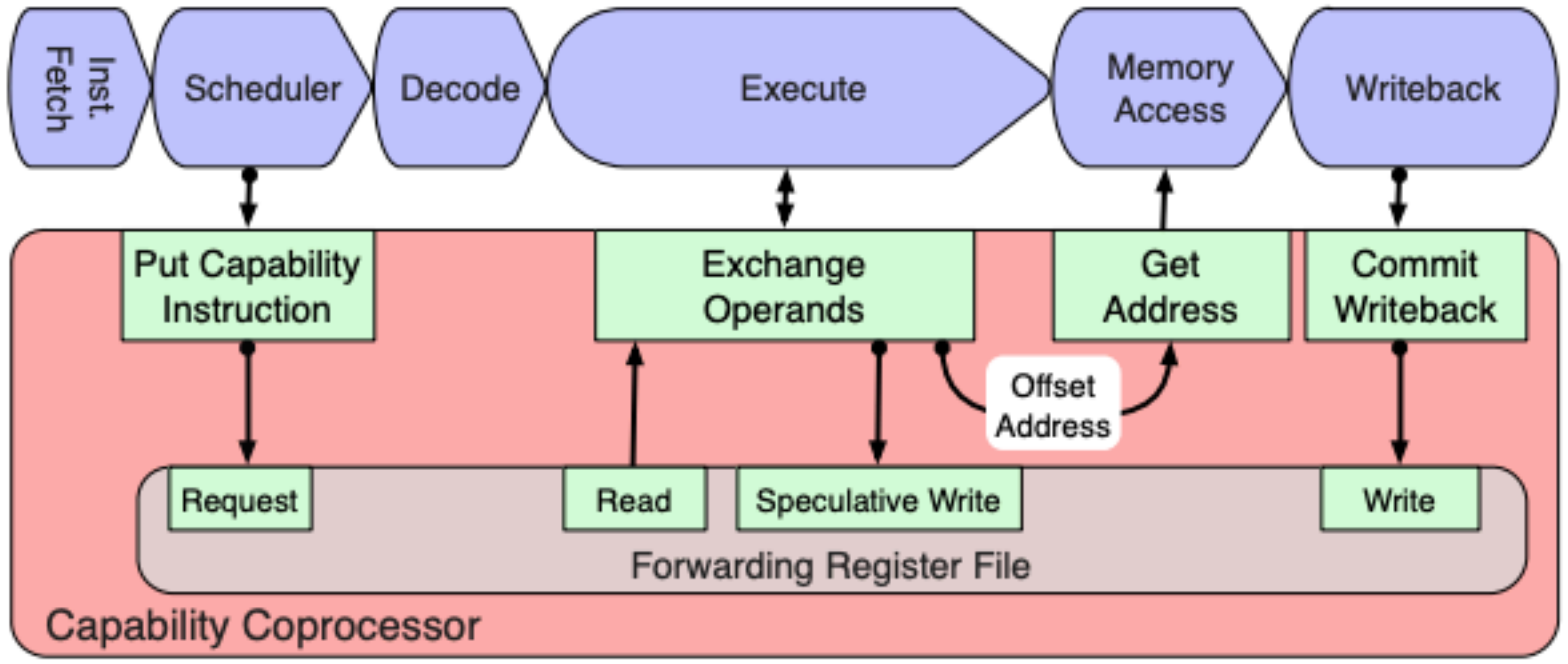
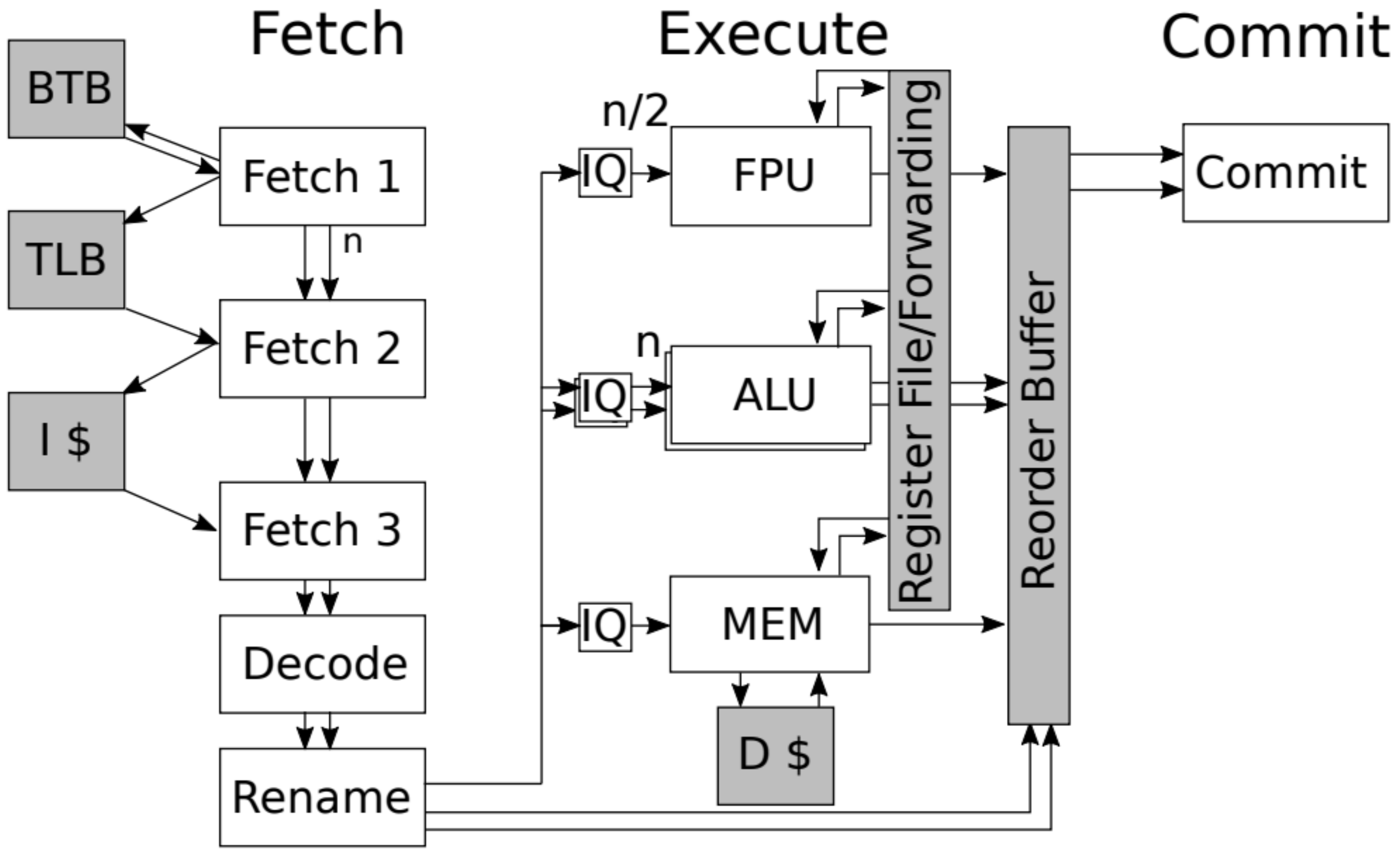


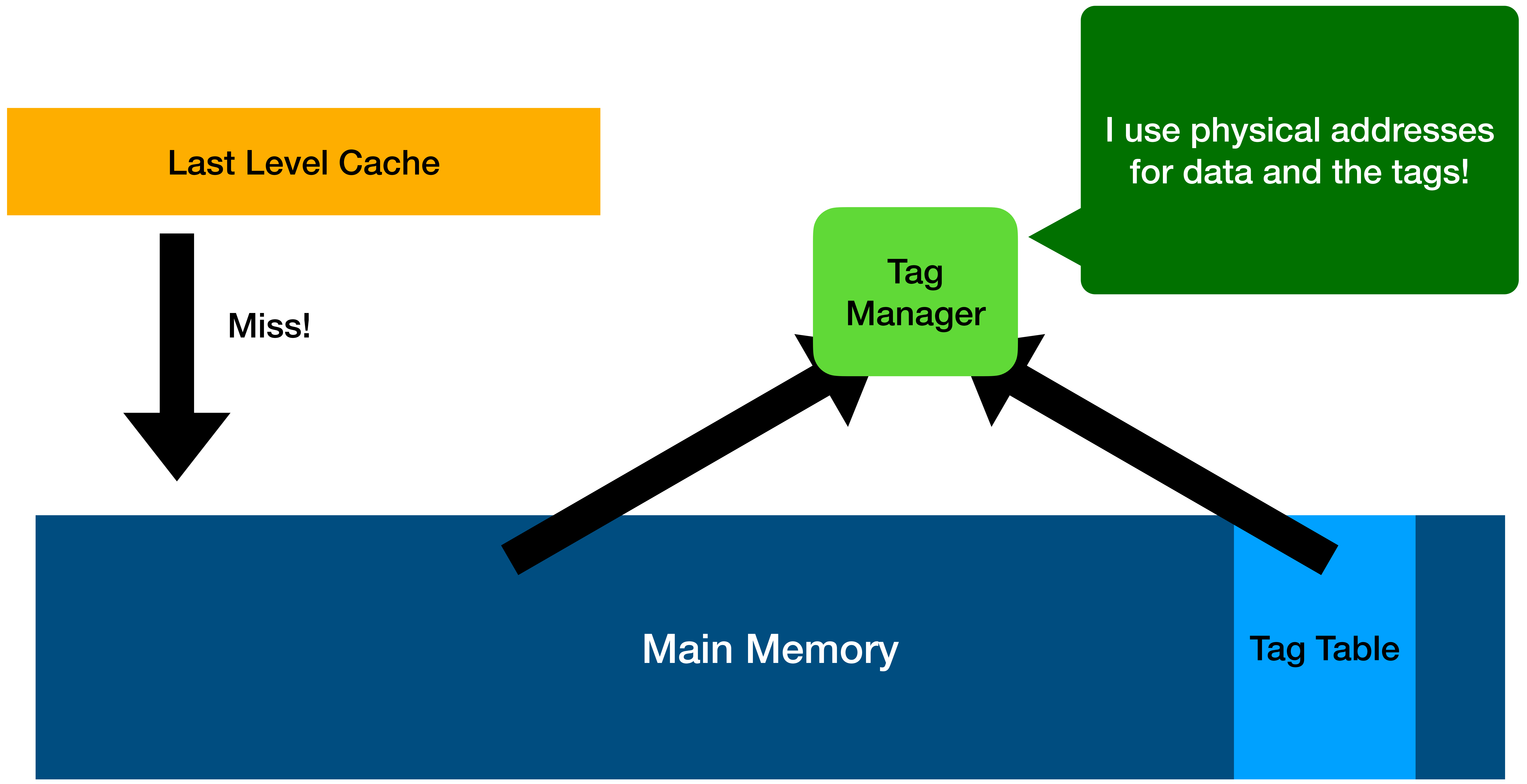
Figure 2: BERI pipeline with capability coprocessor

Capability Processors



TOOOBA pipeline, without capabilities

Capability Hardware



Chat with your neighbor!

- Why is it important to maintain a tag?
- Why is it difficult for a tag manager to verify a capability at a virtual address?
- What is difficult about using physical addresses in the tag manager?

The Capability Address Space

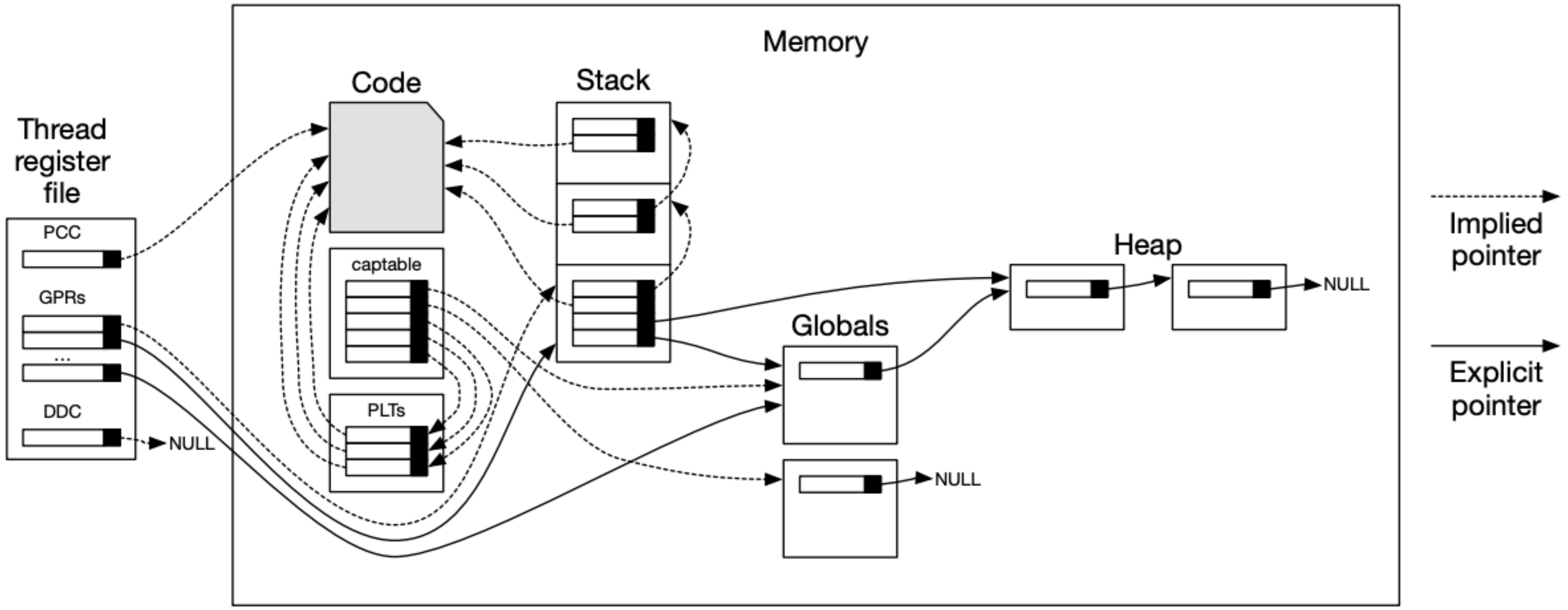
- No virtual addresses… how to do isolation?
  - Use capabilities!
- Virtual memory == process specific memory

Physical Memory
0x00
max

The Capability-Centric Process

- A process is just a set of objects that are strictly defined:
    1. Process registers
    2. Code to execute
    3. Global memory space
    4. Stack address space
    5. Heap address space
- Let’s define each of these as capabilities!

The Capability Centric Process



Chat with your neighbor!

- What are some pros and cons of enforcing isolation with virtual memory versus capabilities?

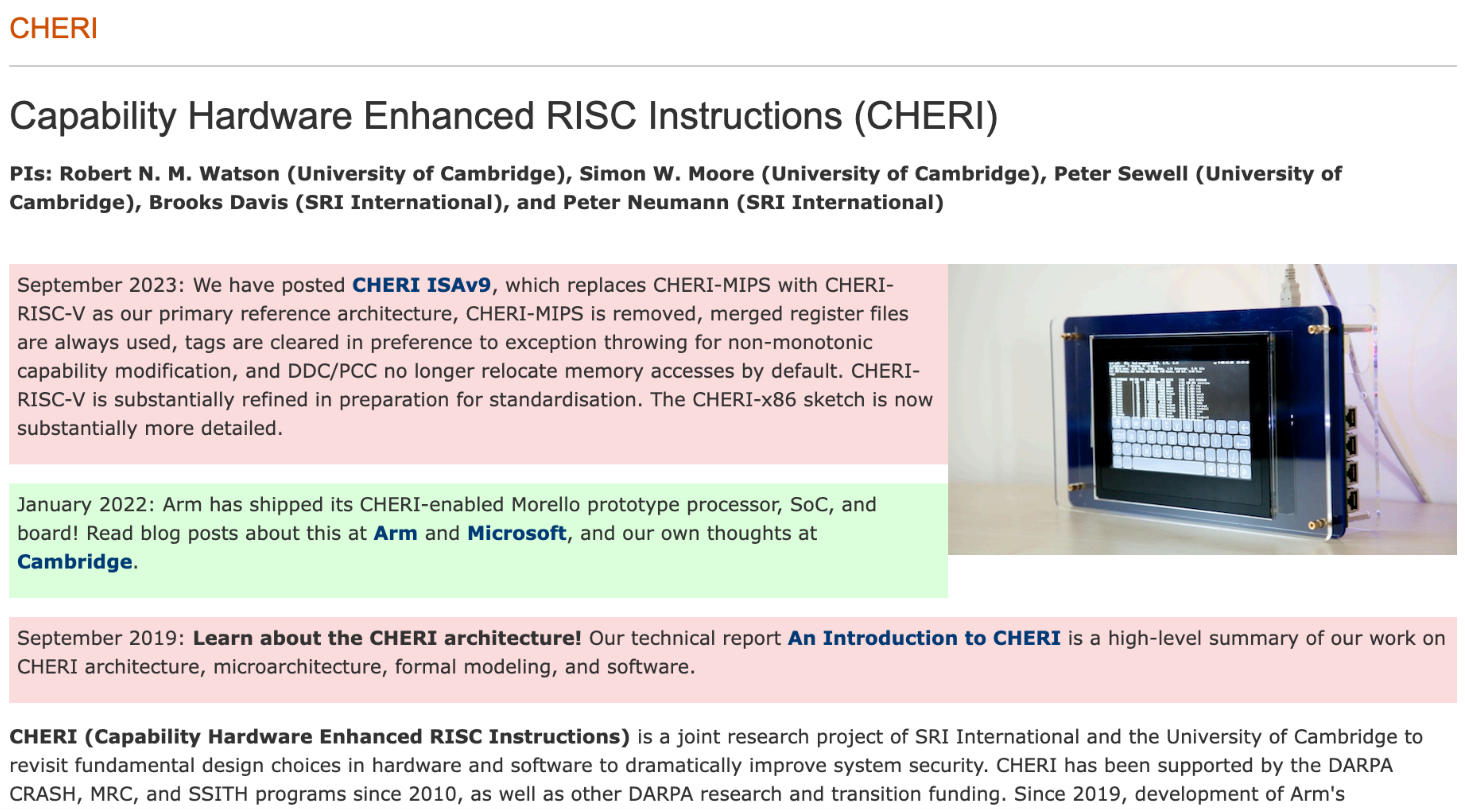
The Capability Nanokernel

- What is a nanokernel? A less functional microkernel (which is a less functional kernel)
    - The absolute bare minimum OS!
- The job of the nanokernel is to create a process
- How does it do this…?

Putting it all together…

- In order to have hardware enforced pointers, the authors of the capabilities framework suggest:
  - Hardware-awareness of data format
  - In-memory and cache-assisted tables to identify data formats
  - New OS semantics and paradigms for memory isolation and processes
  - Bug-free use of the ABI!

Capabilities Today

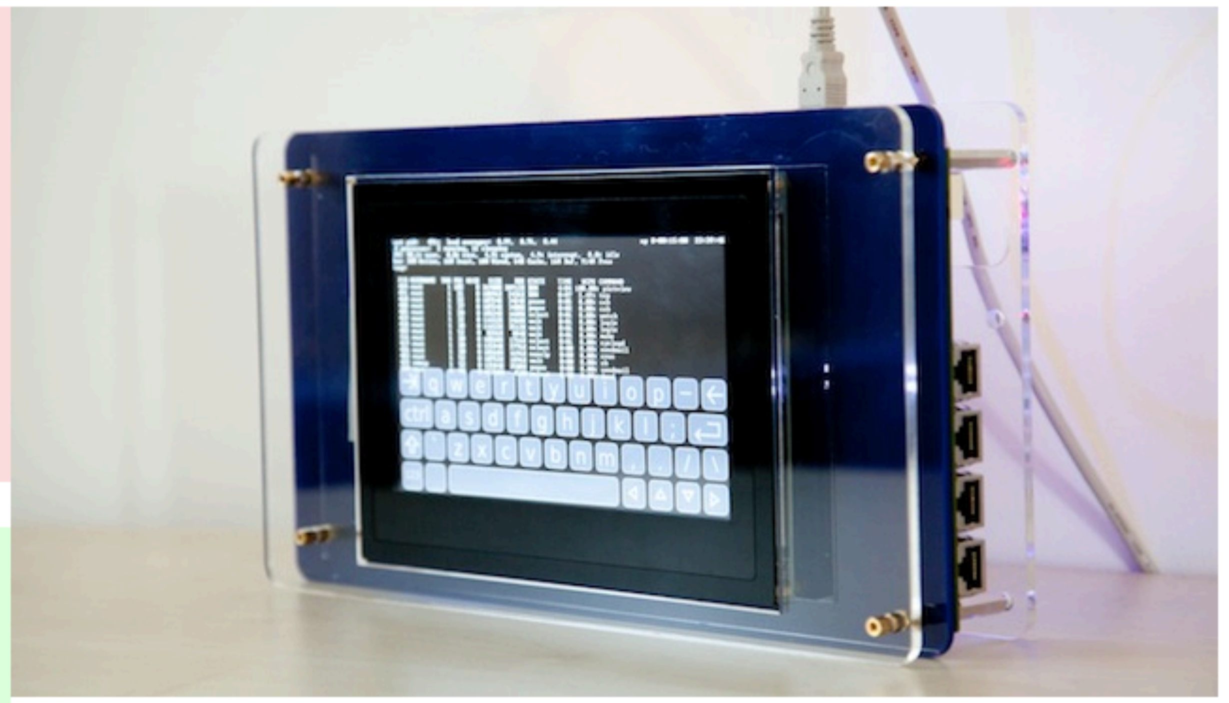


CHERI

Capability Hardware Enhanced RISC Instructions (CHERI)

PIs: Robert N. M. Watson (University of Cambridge), Simon W. Moore (University of Cambridge), Peter Sewell (University of Cambridge), Brooks Davis (SRI International), and Peter Neumann (SRI International)

September 2023: We have posted **CHERI ISAv9**, which replaces CHERI-MIPS with CHERI-RISC-V as our primary reference architecture, CHERI-MIPS is removed, merged register files are always used, tags are cleared in preference to exception throwing for non-monotonic capability modification, and DDC/PCC no longer relocate memory accesses by default. CHERI-RISC-V is substantially refined in preparation for standardisation. The CHERI-x86 sketch is now substantially more detailed.



January 2022: Arm has shipped its CHERI-enabled Morello prototype processor, SoC, and board! Read blog posts about this at **Arm** and **Microsoft**, and our own thoughts at **Cambridge**.

September 2019: **Learn about the CHERI architecture!** Our technical report **An Introduction to CHERI** is a high-level summary of our work on CHERI architecture, microarchitecture, formal modeling, and software.

CHERI (Capability Hardware Enhanced RISC Instructions) is a joint research project of SRI International and the University of Cambridge to revisit fundamental design choices in hardware and software to dramatically improve system security. CHERI has been supported by the DARPA CRASH, MRC, and SSITH programs since 2010, as well as other DARPA research and transition funding. Since 2019, development of Arm's