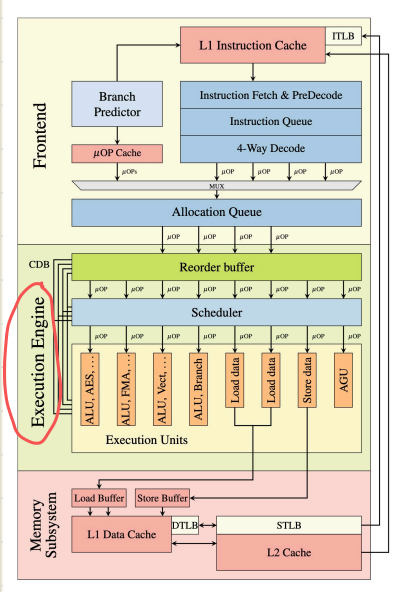
5/29/24 Meltdown

→ exploits side effects of microarchitecture optimization to allow user processes to read arbitrary kernel memory (including physical memory mapped into the kernel)

→ microarchitecture optimization

→ CPU pipeline: fetch, decode, execute, memory subsystem



→ every instr. goes through each stage

→ an instr. may stall in execute (e.g. mem access)

→ other execution units go idle :(

→ or we can schedule subsequent instr. execute

☆ out of order execution!

↳ results stored in temp buffer / register

↳ retired (visible) in order

☆ intel processors enforce perm. check at retire time, not execute time

→ **Cache attacks**

→ CPU caches: L1, L2, L3 — shared, physically indexed physically tagged.

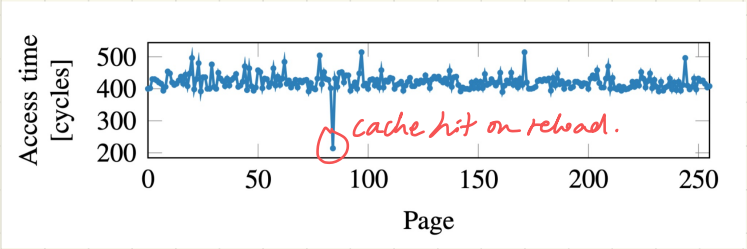
→ cache hit is noticeably faster

→ flush + reload

→ attacker/receiver flush every cache line, wait for a while

→ victim/sender access memory, causing certain cache lines to be filled

→ attacker/receiver reloads every cache line, measure the time for each access & learn the access pattern from victim



→ **kernel memory**: physical memory mapped into kernel memory,
kernel memory mapped into every process's VAS.

→ The attack:

(flush all cache lines)

① read 1 byte of kernel memory (should raise an exception)

② access (user_array[kernel byte × 4096])

↳ 256 pages large

※ if this is executed before the exception is raised, can use the temp result from ① & leave a foot print!

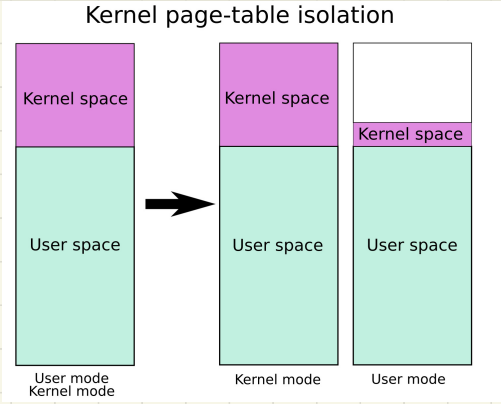
③ install a custom SIGSEGV handler, upon an exception, reload cache line to figure out the value of kernel byte

→ Mitigation:

Kernel page table isolation

→ only map a small part of kernel necessary for trap / interrupt entrance into every user VAS

→ switch to a full kernel page table once in kernel mode



Root Cause	Description	How it affects performance	Impact
Security Enhancements: max combined slowdown: 146% poll			
Kernel page-table isolation (§4.1.1)	Removes kernel memory mappings from the page table upon entering userspace to mitigate Meltdown.	A kernel entry/exit now swaps the page table pointer, which further leads to subsequent TLB misses.	`recv` 63%, `small-read` 60%