

Advanced Topics in Computer Architecture

Secure Processors 2: Speculative Execution Attacks

Prof. Simon W. Moore



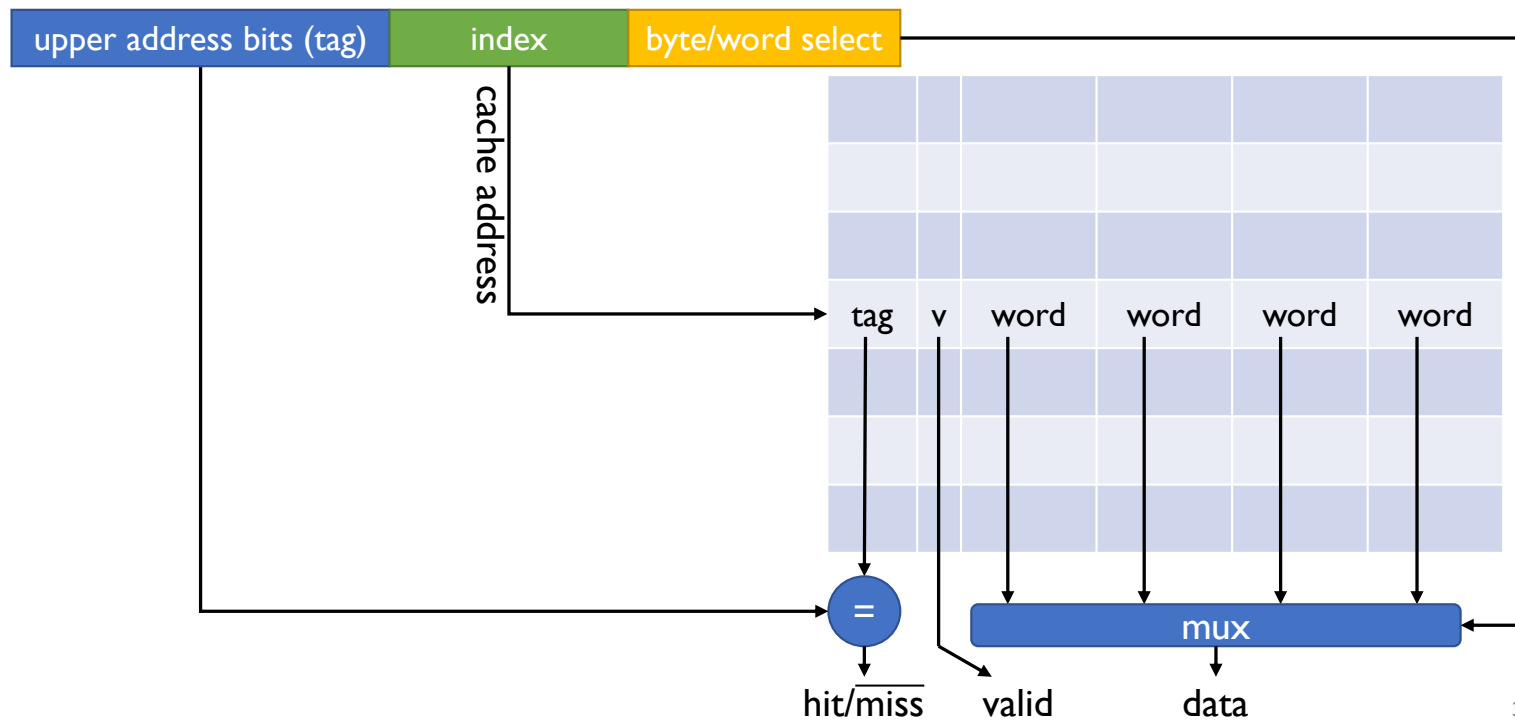
Computer Science & Technology

Copyright © Simon W. Moore, 2020

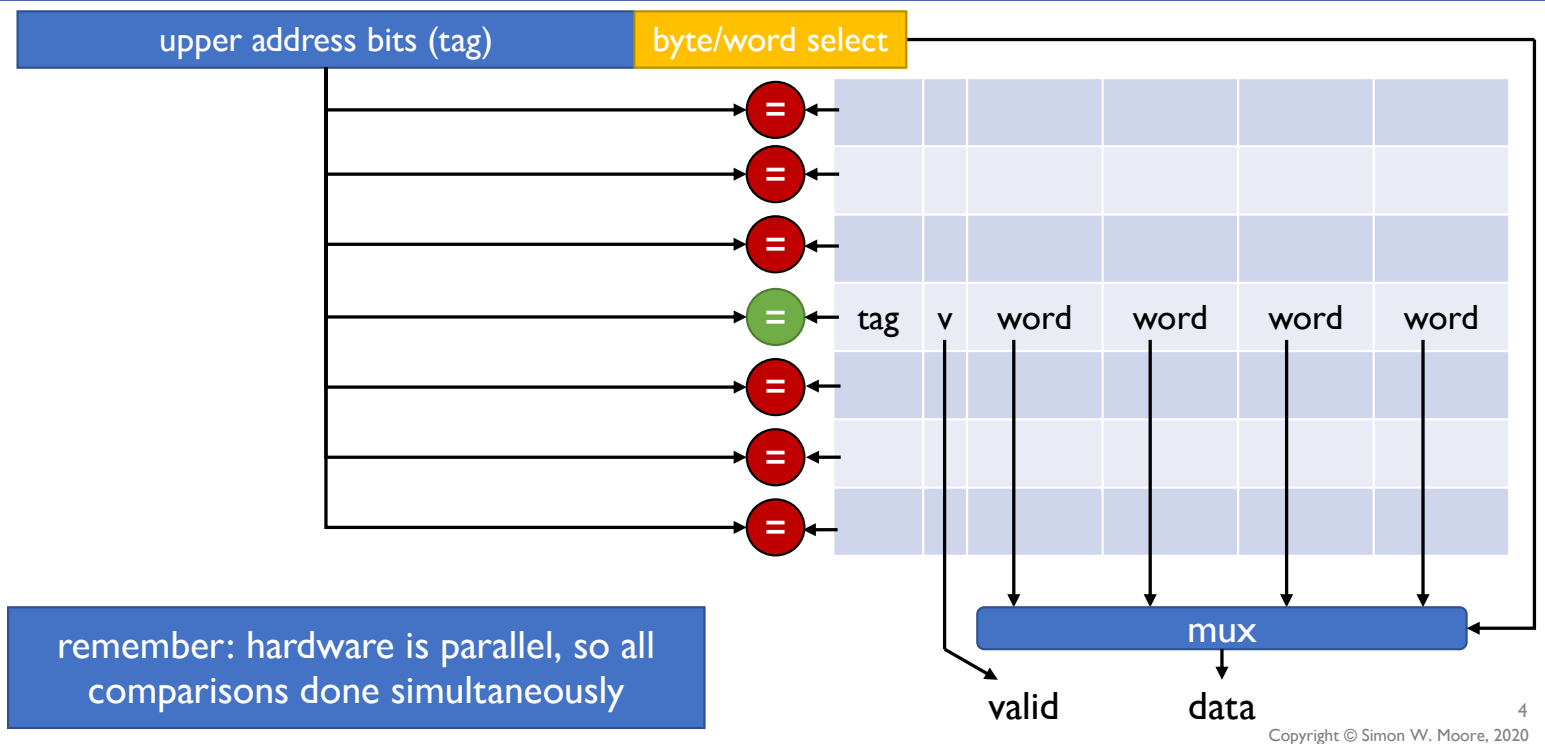
Introduction

- Quick introduction to caches and cache side channels
 - Typically used as a communication mechanism to speculative execution attacks
- Very brief introduction to branch prediction and speculation
- Very brief introduction to speculative execution attacks

Background: Simple direct-mapped cache



Background: Fully associative cache



Background: Set associative caches

- Set associative cache idea:
 - have N direct-mapped caches
 - reads look in all N caches for data
 - thus the cache has N-way associativity
- In use:
 - Set-associative caches are widely used
 - E.g., Intel Core i9-9990XE – 14 cores
 - 14 x L1 instruction and data caches are each: 32KiB 8-way set associative
 - 14 x L2 unified cache (instructions and data): 1MiB 16-way set associative
 - 1 x L3 last-level cache: 19.25MiB 11-way set associative
 - E.g., ARM A72 used in the Raspberry Pi 4 – 4 cores
 - 4 x L1 instruction cache 48KiB 3-way associative & L1 data cache 32KiB 2-way associative
 - 1 x L2 unified cache: 1MiB 16-way set associative

Background: Replacement Policy

- Direct mapped: no choice
- Set associative
 - Prefer non-valid entry, if there is one
 - Otherwise, choose among entries in the set
 - **Least-recently used (LRU)**
 - Choose the one unused for the longest time
 - Simple for 2-way, manageable for 4-way, too hard beyond that
 - **Not last used**
 - Approximates LRU and is simpler to implement for 8+ ways
 - **Random**
 - Gives approximately the same performance as LRU for high associativity
 - Simple to implement and avoids pathological misses

Cache timing side channels

- Synchronous prime and probe attack
 1. Prime: flush the cache (or fill it with data from addresses that will not be used next)
 2. Call code that you want to snoop on
 3. Probe: for each cache-line, time how long it takes to access the line using a fine-grained timer
 4. Repeat and signal average to remove any noise
- Asynchronous prime and probe attack
 - As above but attacker is in one process and trying to observe another process
 - More tricky to get the timing right, so often more repetitions and signal processing required
- Possible attack vector
 - Could allow JavaScript code inside a process/sandbox to observe the main application

Branch prediction and speculation

- Branch prediction is widely used
 - Avoids many pipeline stalls/refills
- Typical mechanism involves recording a history of:
 - where branches instructions are stored in memory (don't wait to fetch the instruction)
 - where the branch target was last time
 - statistical data on how likely the branch will be taken

Speculative Execution Attacks

- Some of the attacks named in the press:
 - Spectre and Meltdown: <https://meltdownattack.com/>
 - Foreshadow: <https://foreshadowattack.eu/>
- Core ideas:
 - Speculatively execute some code or read some data that the application is otherwise not allowed to access
 - Ensure that the speculative execution does some data-dependent memory accesses
 - Use cache side-channel analysis to determine the data
- So basically a combination of:
 - Efficient synchronous prime-and-probe cache attack
 - + Speculatively read data or execute code where you don't have the permissions

Further reading

- **The attack:** *Spectre Attacks: Exploiting Speculative Execution*
<https://spectreattack.com/spectre.pdf>
- **Example industry response:** ARM white paper: *Cache Speculation Side-channels* <https://developer.arm.com/support/arm-security-updates/speculative-processor-vulnerability>
- **Research into hardware mitigations:** *M16: Secure Enclaves in a Speculative Out-of-Order Processor*
<https://arxiv.org/abs/1812.09822>
- Further pointers:
 - <https://spectreattack.com/>
 - <https://meltdownattack.com/>
 - <https://foreshadowattack.eu/>