CSE 127 Computer Security

Alex Gantman, Spring 2018, Lecture 18

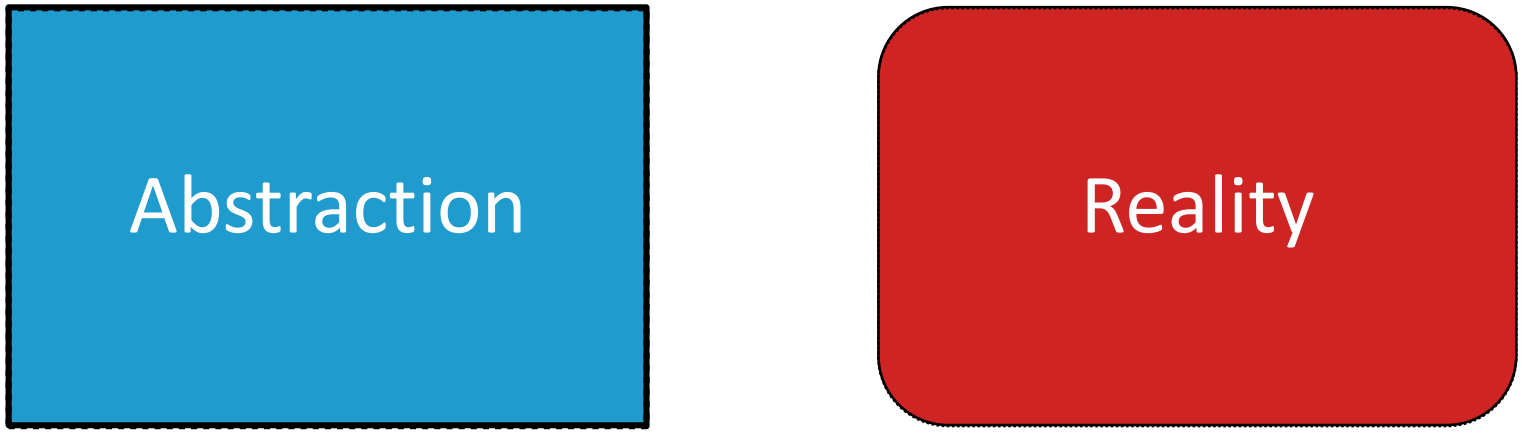
Hardware Security:
Meltdown, Spectre, Rowhammer

Vulnerabilities and Abstractions

"Layers of abstraction become boundaries of competence."

- Sergey Bratus

Vulnerabilities and Abstractions



Understanding Vulnerabilities

- How vulnerabilities are explained
  - Alice: “There’s a vulnerability in XY.”
  - Bob: “What is it?”
  - Alice: “Well, you know how X works?”
  - Bob: “Yeah.”
  - Alice: “And you know how Y works?”
  - Bob: “Yes.”
  - Alice: “…”
  - Bob: “Oh!”

Review: ISA and µArchitecture

Instruction Set Architecture (ISA)

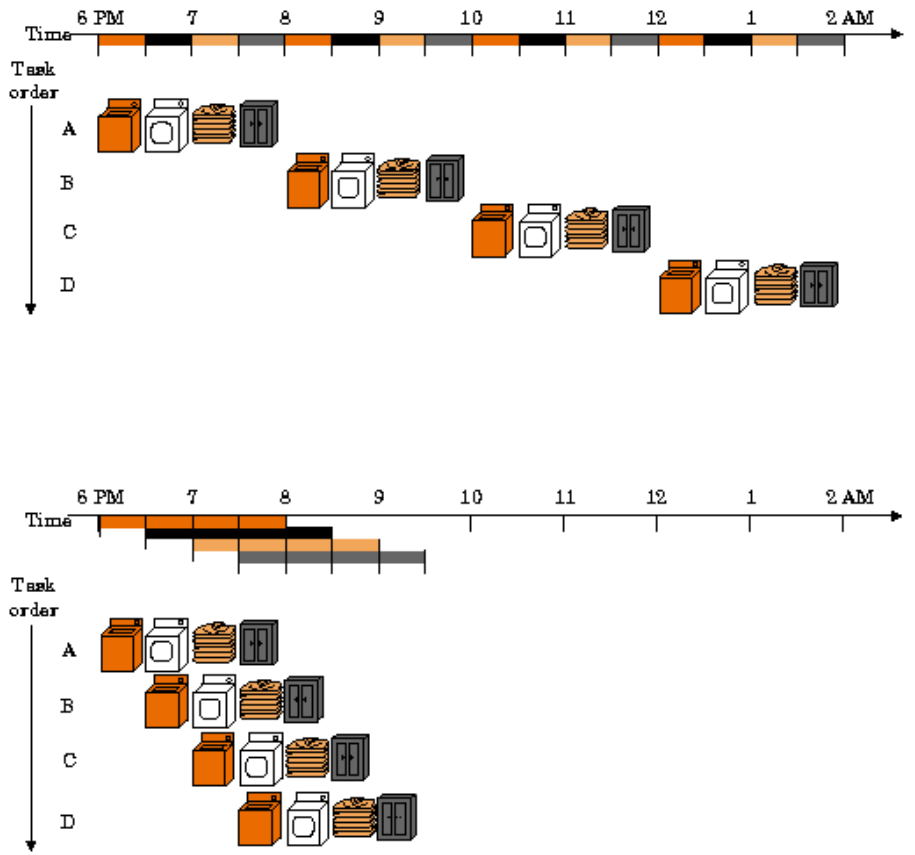
- Defined interface between hardware and software
  - Registers, instructions, etc.

µArchitecture

- Implementation of the ISA
- "Behind the curtain" details
  - E.g. cache specifics
- Spoiler: µArchitectural details may become architecturally visible

Review: Instruction Pipelining

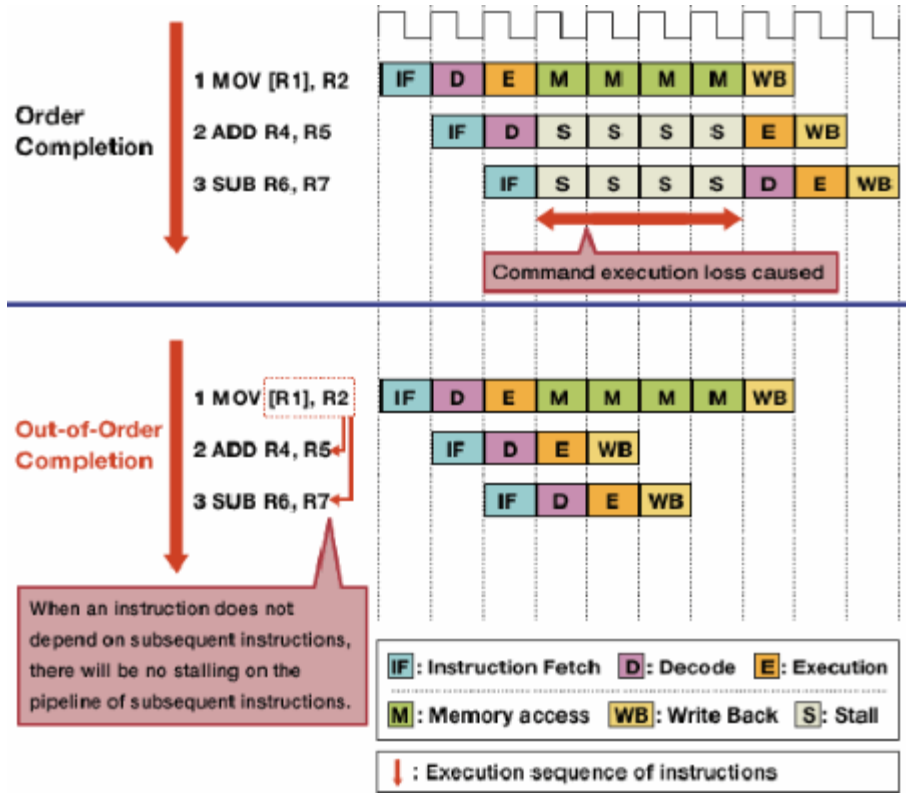
- Processors break up instructions into smaller parts so that these parts could be processed in parallel.
- µArchitectural optimization
  - Architecturally, instructions appear to be executed one at a time, in order
  - Dependencies are resolved behind the scenes



https://www.cs.fsu.edu/~hawkes/cda3101lects/chap6/index.html?$$$F6.1.html$$$

Review: Out-of-order Execution

- Some instructions can be safely executed in a different order than they appear.
- This may allow the processor to use available resources to "pre-compute" future instructions.
- µArchitectural optimization
  - Architecturally, it appears that instructions are executed in order.



https://renesasrulz.com/doctor_micro/rx_blog/b/weblog/posts/pipeline-and-out-of-order-instruction-execution-optimize-performance

Review: Speculative Execution

- Sometimes control flow depends on output of an earlier instruction.
  - E.g. conditional branch, function pointer
- Rather than wait to know for sure which way to go, the processor may "speculate" about the direction/target of a branch
  - Guess based on the past
  - If the guess is correct, performance is improved
  - If the guess is wrong, speculated computation is discarded and everything is re-computed using the correct value.
- µArchitectural optimization
  - At the architecture level, only correct, in-order execution is visible

Review: Virtual Memory

- Kernel virtual memory is mapped into every process
  - For efficiency
- Page table access control ensures that kernel pages are only accessible when the processor is in privileged mode

	Non-Secure				Secure	
EL0	App X	App Y	App X’	App Y’	App X”	App Y”
EL1	Guest OS A		Guest OS B		Secure OS	
EL2	Hypervisor					
EL3			Secure Monitor			

Review: Cache Side Channel

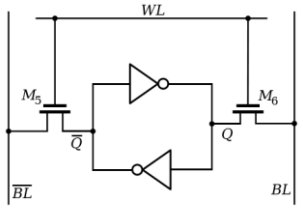
- Attacker process can infer which address the victim process accessed by monitoring cache contents
- Cache contents are monitored indirectly
  - Timing access to different memory locations
- Flush + Reload

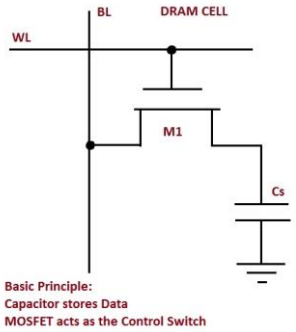
Meltdown and Spectre

Rowhammer

Review: RAM

- Volatile memory: data retained only as long as power is on
  - As opposed to persistent memory, which retains data even without power (e.g. flash, magnetic disks)
- Static RAM (SRAM) vs Dynamic RAM (DRAM)
  - SRAM: retains bit value as long as power is on, without any additional refresh
    - Faster
    - Lower density
    - Higher cost
  - DRAM: requires periodic refresh to maintain stored value
    - Refresh about every 64 ms
    - Higher density (higher capacity)
    - Lower cost

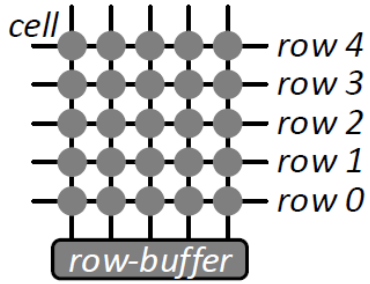




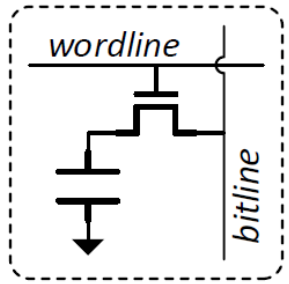
https://commons.wikimedia.org/wiki/File:SRAM_Cell_Inverter_Loop.png
https://allthingsvlsi.wordpress.com/tag/dram-cell/

Review: DRAM

- Cells are grouped into rows
  - ~1Kb per row
  - All cells in a row are refreshed together
- Refresh: read the row and write it back
- All access to individual cells happens via the "row buffer"



a. Rows of cells



b. A single cell

Rowhammer

- Repeatedly accessing a row, can cause bit flips in adjacent rows
  - ~ 1 in 1000 bits flip
  - Bit flips are repeatable
- Adjacent rows can belong to other process', or kernel's, memory space.

Rowhammer

- Attack scenario: attacker code is executing on the same machine as victim, but with less privileges
  - Example: userland attacking kernel, javascript attacking browser, etc.
- Exploit sketch
  - Characterize DDR flip locations
  - Identify protected target data to flip
    - Examples: page tables, su binaries, etc.
  - Maneuver protected data over flip location
    - Example: allocate all other memory
  - Hammer own memory locations to alter protected data

Rowhammer

	2015	2016	2017
Hammer technique	Single-sided (cycle)	Double-sided	One-location
Launch platform	Desktop Intel/AMD	Browser	SGX
		Mobile, ARM	
		Cloud server	
Protected data manipulation	Page-table spray (probabilistic)	Memory exhaustion (deterministic)	Memory monitoring (side-channel)
Protected data	Page table	Kernel object	Opcode

Additional Resources

- Google Project Zero
  - https://googleprojectzero.blogspot.com/
  - Rowhammer:
    - https://googleprojectzero.blogspot.com/2015/03/exploiting-dram-rowhammer-bug-to-gain.html
  - Spectre/Meltdown:
    - https://googleprojectzero.blogspot.com/2018/01/reading-privileged-memory-with-side.html
- Anders Fogh blogs
  - https://cyber.wtf/author/andersfogh1974/
  - https://dreamsofastone.blogspot.com/
- Paul Kocher on Spectre
  - https://www.youtube.com/watch?v=hqIavX_SCWc

Additional Resources

- TU Graz team
  - Moritz Lipp: https://mlq.me/
  - Daniel Gruss: https://gruss.cc/
  - Michael Shwarz: https://misc0110.net/web/
- VUSec
  - https://www.vusec.net/

Homework

- Final is on Saturday 6/9 3pm-6pm in TBD

Next Lecture…

Secure Development Lifecycle