

CSE 127 Computer Security

Stefan Savage, Spring 2018, Lecture 19

Hardware Security:
Meltdown, Spectre, Rowhammer

Vulnerabilities and Abstractions



Abstraction



Reality

Review: ISA and μ Architecture

Instruction Set Architecture (ISA)

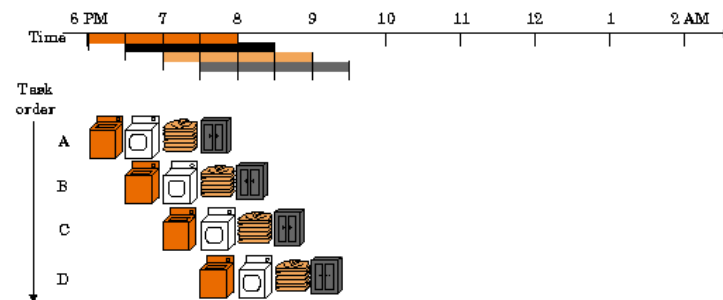
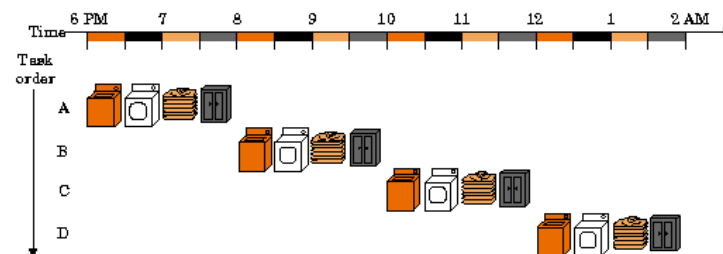
- Defined interface between hardware and software
 - Registers, instructions, etc.

μ Architecture

- Implementation of the ISA
- “Behind the curtain” details
 - E.g. cache specifics
- Spoiler: μ Architectural details can sometimes become “architecturally visible”

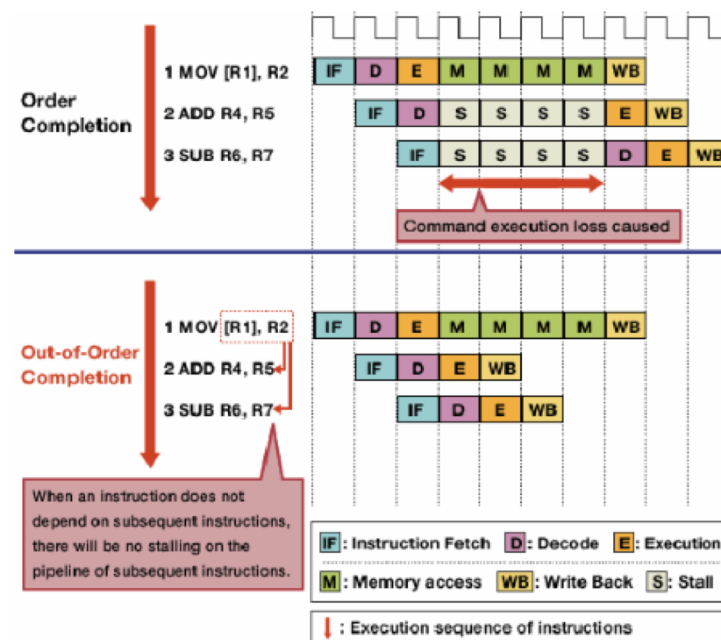
Review: Instruction Pipelining

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



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.
- μ Architectural optimization
 - Architecturally, it appears that instructions are executed in order.



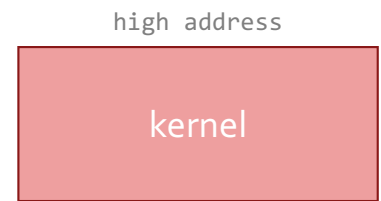
Review: Speculative Execution

- 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.
- μ Architectural 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

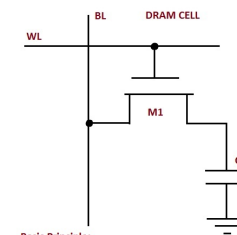
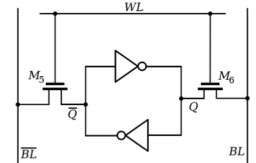


Meltdown and Spectre

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

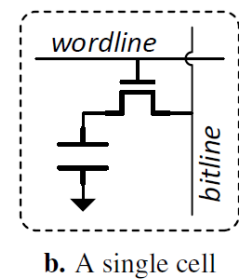
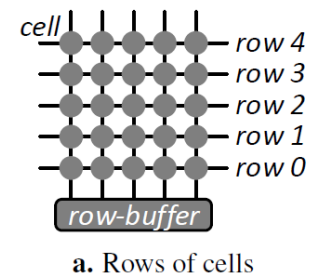


Basic Principle:
Capacitor stores Data
MOSFET acts as the Control Switch

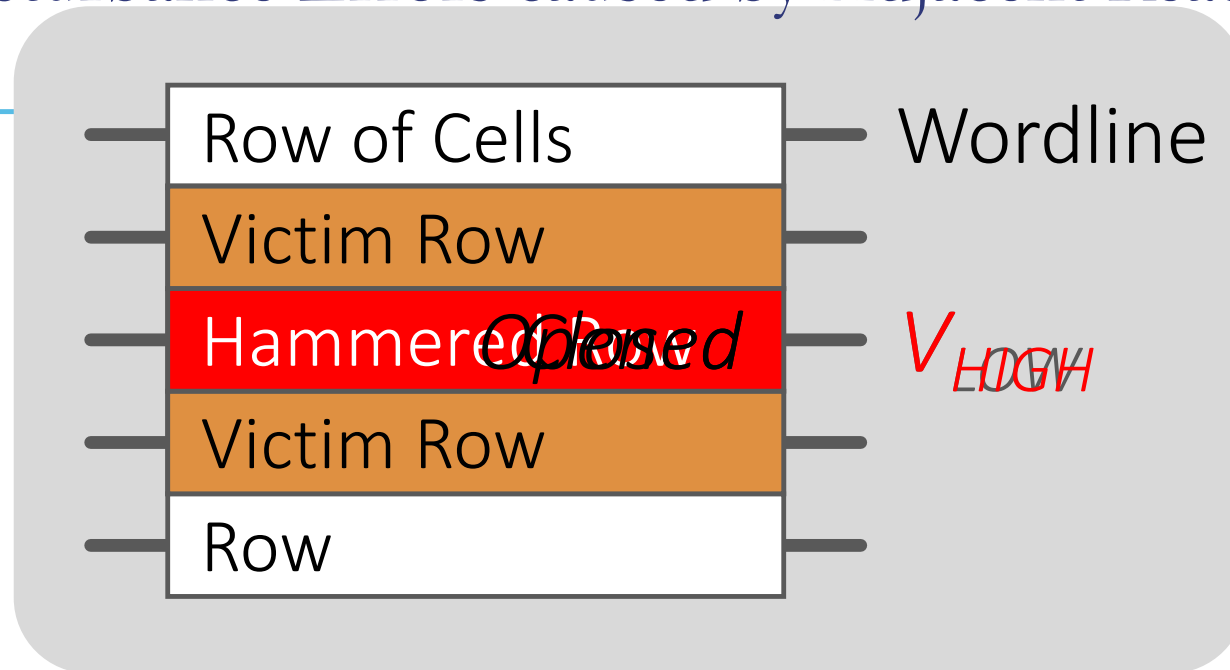
https://commons.wikimedia.org/wiki/File:SRAM_Cell_Inverter_Loop.png
<https://allthingsvlsi.wordpress.com/tag/dram-cell/>

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”
- As DRAM gets smaller (<35nm) more issues with *reliability* (retention, writes, and... reads)



Disturbance Errors caused by Adjacent Reads



Repeatedly opening and closing a row enough times within a refresh interval induces **disturbance errors** in adjacent rows in **most real DRAM chips you can buy today**

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.
 - Rowhammer attack lets attacker modify memory they can't access
- 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
- Amazing fact: **random** bit flips can be used to semi-deterministically take over computer

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=hqlavX_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/>

Much thanks!

- You've been a great class
- Try to make the world a bit less broken
- Good luck!

Next up

- Final: Saturday 3-6pm