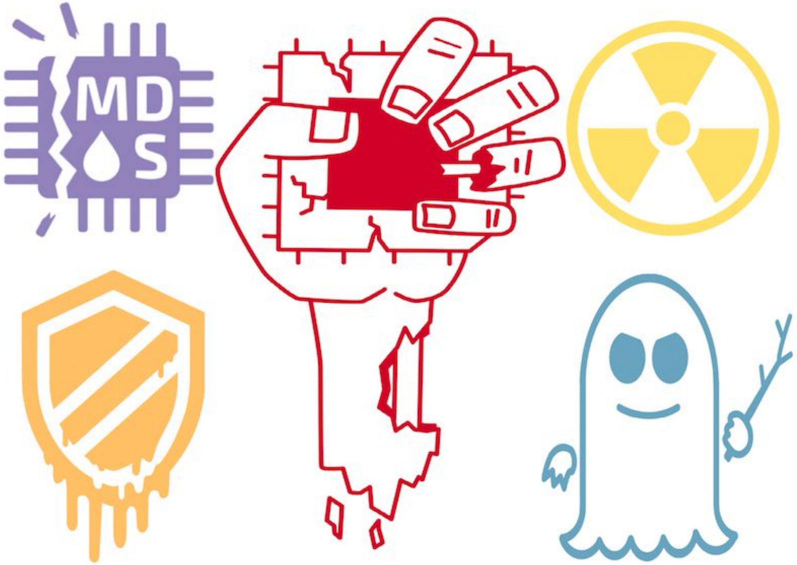
Why Audit Your CPU?

Searching for Undocumented CPU Behavior

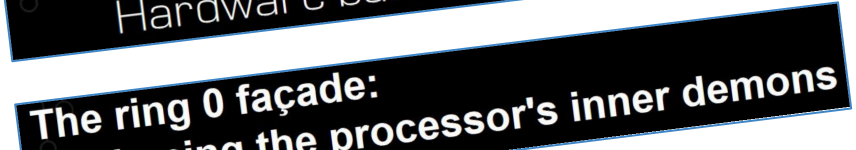
Catherine Easdon

Undocumented Behavior



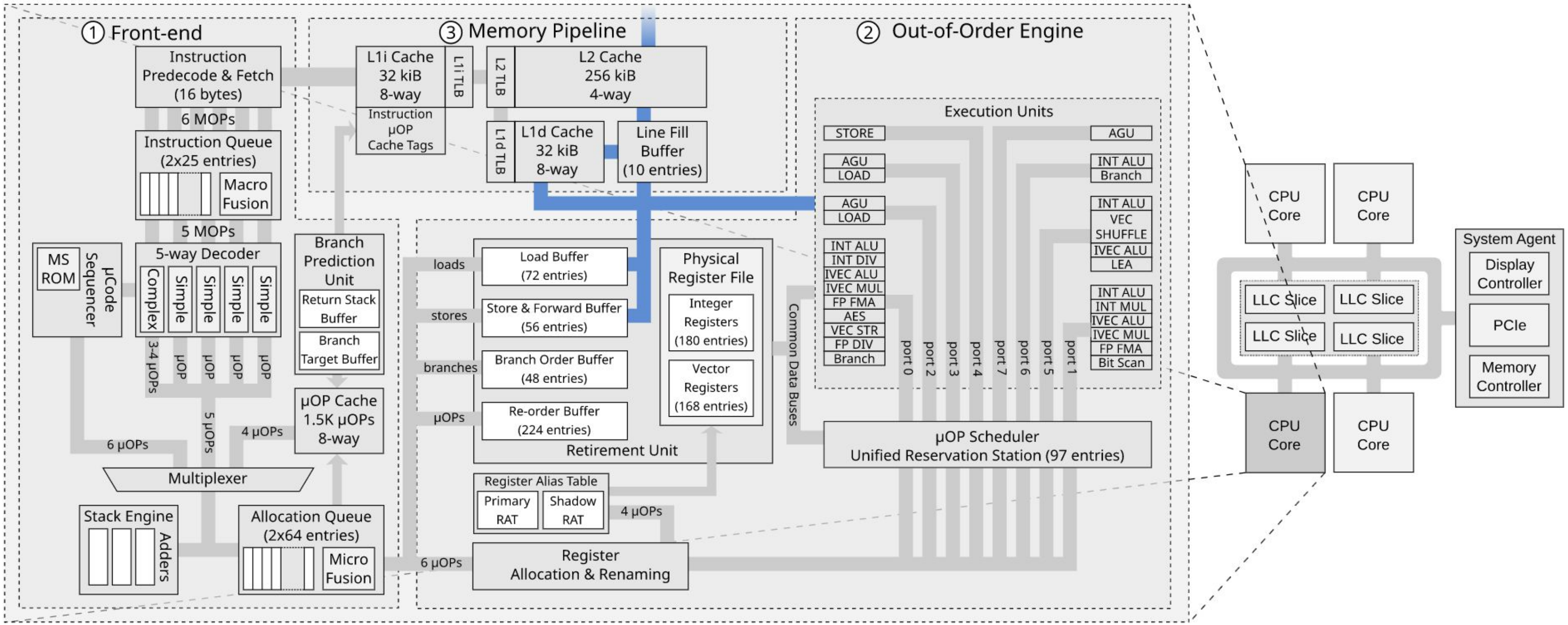
'Super-secret' debugger discovered in AMD CPUs

GOD MODE unlocked:
Hardware backdoors in x86 CPUs


The ring 0 façade:
awakening the processor's inner demons

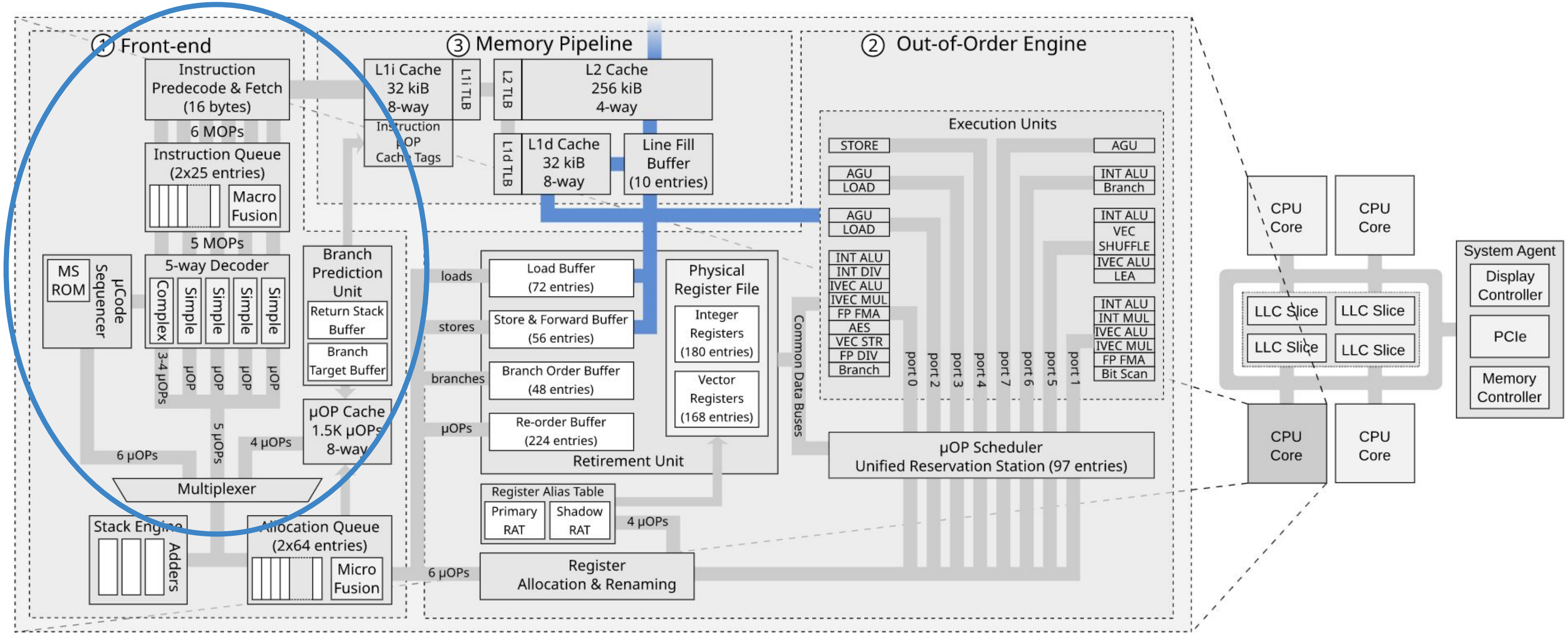
How to Hack a Turned-Off Computer, or Running Unsigned Code in Intel Management Engine

Where Should We Look First?



Where Should We Look First?

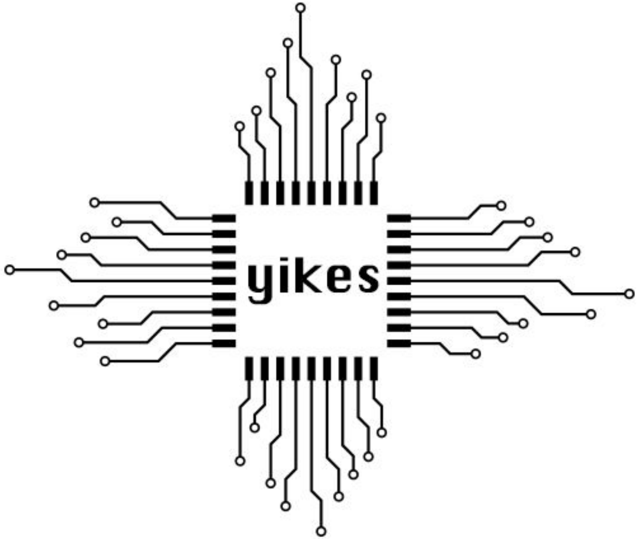
Instructions!



Undocumented Instruction Behavior

What are we looking for?

- **“Halt-and-catch-fire” instructions**
- **Privilege escalation vulnerabilities**
- **Debug instructions + backdoors**
- **Malicious microcode, SMM, ME/PSP**
- **Logic or manufacturing bugs**
- **Side channels or exploitable transient effects**

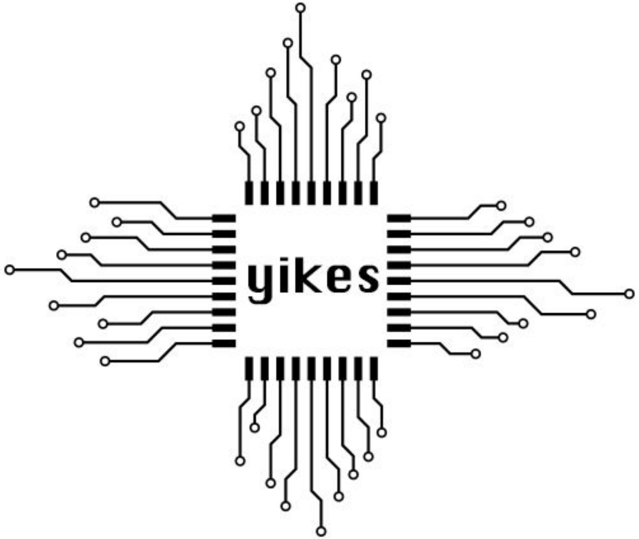


There are precedents for almost all of these!

Undocumented Instruction Behavior

Example: 2048 undocumented instructions found on one microcontroller

- 2 undocumented user-mode encodings *requiring dedicated logic in the decoder*
- Modify register state, read main memory … exploitability uncertain
- No response from manufacturer (>4mo)



OpcodeTester

Aims:

- Automate testing and analysis of undocumented instruction behavior (building on Sandsifter’s concept)
- Make CPU auditing as normal + straightforward as a virus scan

Currently supported: Intel x86 32/64-bit, RISC-V 32/64-bit; Linux user-mode, kernel driver, RISC-V machine mode

Future? *Improve analysis*, AMD x86, ARM, SMM, ME/PSP, specialized processors; bootable, Windows, mobile

Thank you for listening!

https://github.com/cattius/opcodetester/

Image Attribution

Slide 1: https://www.flickr.com/photos/130561288@N04/39793547952/

Slide 2: https://threatpost.com/behind-the-naming-of-zombieload-and-other-intel-spectre-like-flaws/144875/

Slides 3-4: https://mdsattacks.com/

Slides 5-6: https://blog.skyboxsecurity.com/intel-cpu-vulnerabilities-could-be-used-in-mds-attacks/