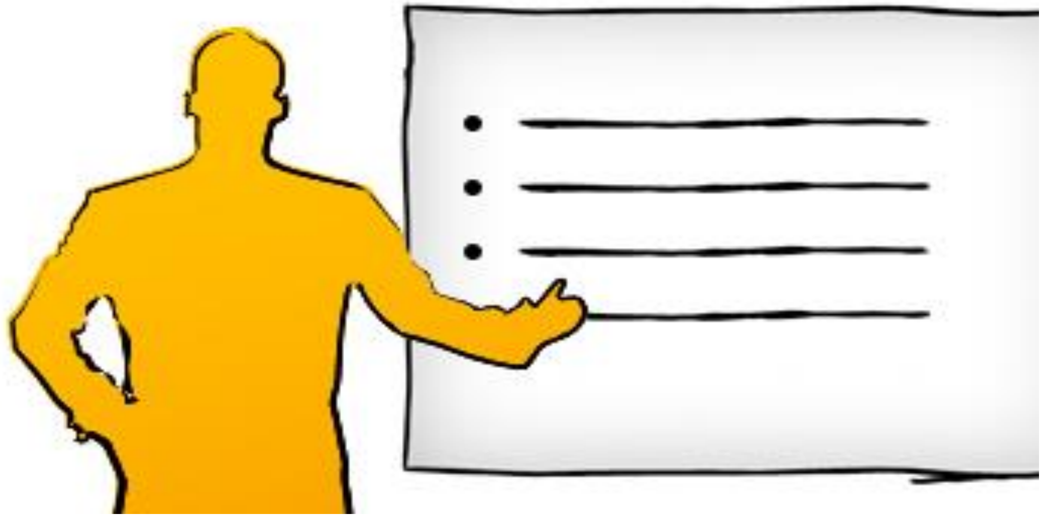
Foundational HPC SYSTEMS 2020 and Beyond

Steven Wallach (swallach@micron.com)

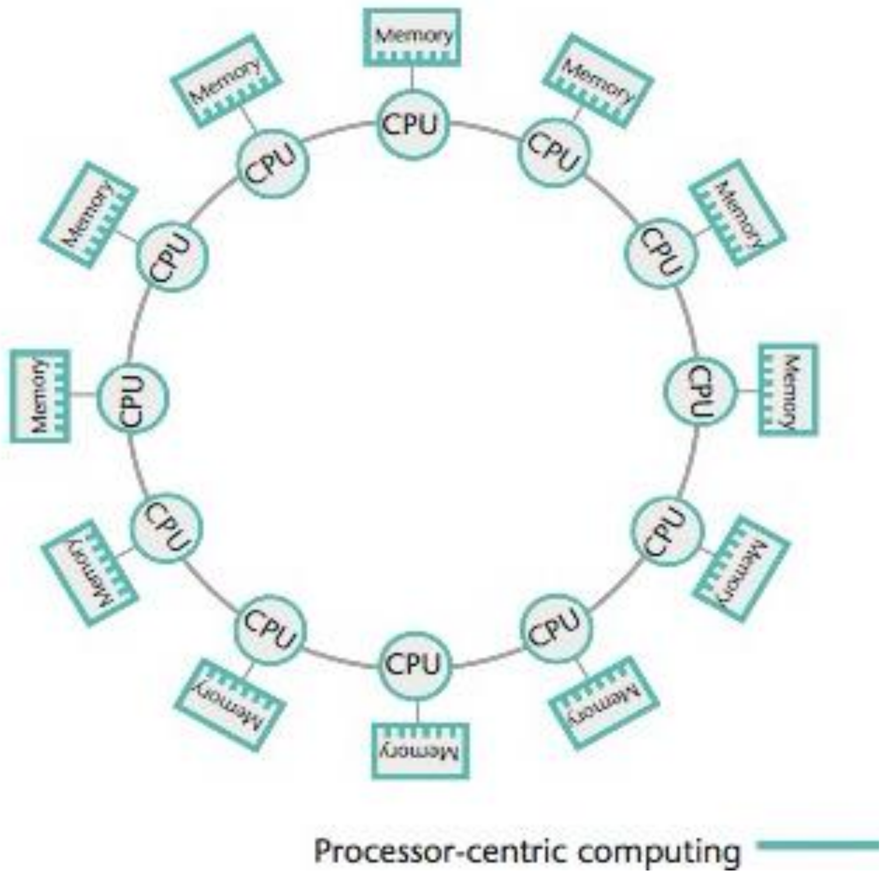
Presentation

- Architecture 1.0
- Architecture 2.0
  - Foundational
  - Upward compatibility with RV64
  - Accessing Encrypted Memory
  - Secure Micro-architecture
    - Spectre and Meltdown
    - Heap Fung Hui
- Architecture 3.0
- References



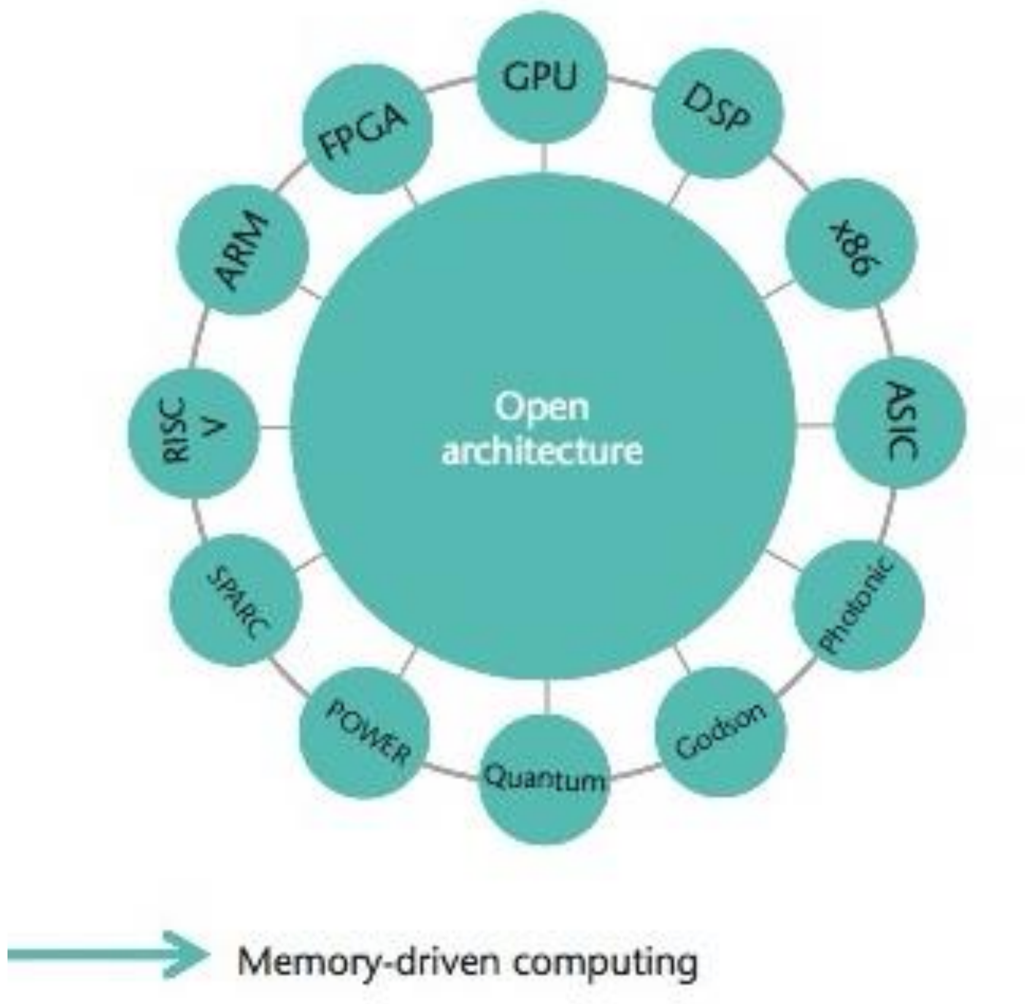
Architecture 1.0 (Mark Hill [1])

- The **definition** of Architecture 1.0 is inadequate to protect information
- Processor Centric
  - Speculation
  - Pipelining
  - Clock Cycle Flat
- Stan Williams [2] “The end of Moore’s law could be the best thing that has happened in computing since the beginning of Moore’s law. Confronting the end of an epoch should enable a new era of creativity by encouraging computer scientists to invent biologically inspired devices, circuits, and architectures implemented using recently emerging technologies. “

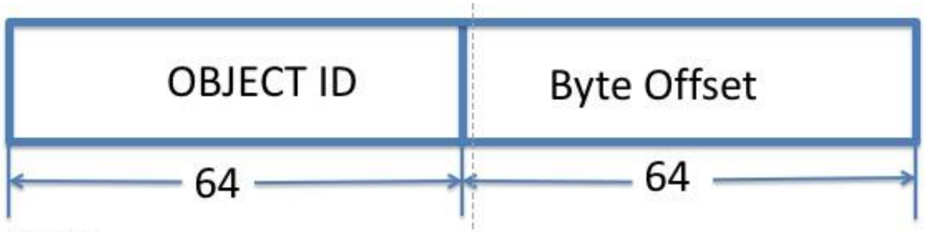


Architecture 2.0

- What this means
  - NOT just ISA
  - Security within micro-architecture
    - There are evil people
    - Covert Channels
- Memory centric computing [2]
  - Numerical Processing is trivial
  - Price/Performance/Watt
  - Achieving efficient memory access
    - Is VERY HARD
  - Memory references are homogeneous, processing is NOT
  - **Caches now need to be protected machine state**

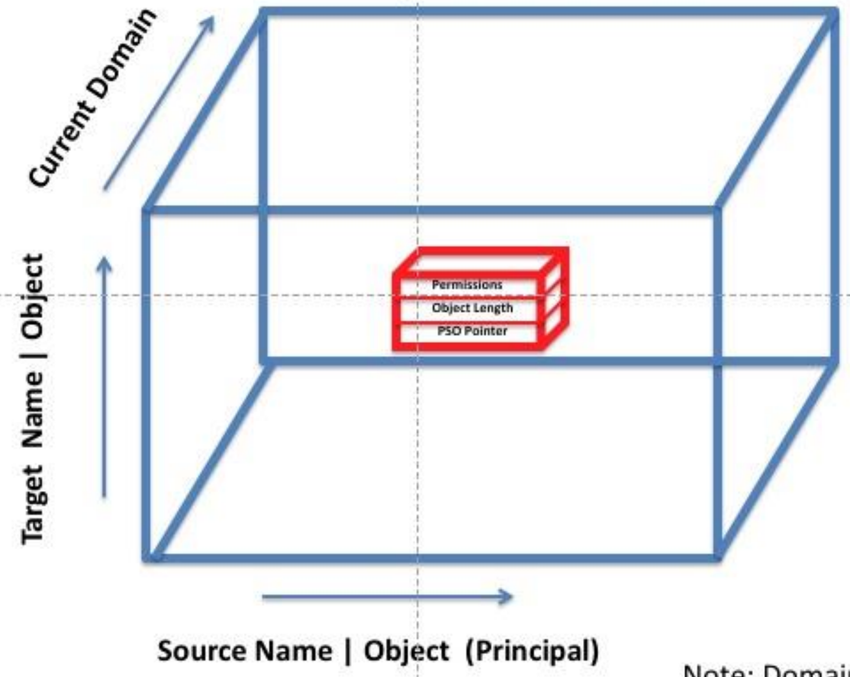


Summary of RV128 [5]



- 128 Bits
- Object ID – Unique Identifier
  - a software (or hardware) structure that is considered to be worthy of a distinct name.
- Indexing is 64 bits – A[i]
  - Program Counter
  - Stack Pointer (CI format - Loads and Stores)
- ISA independent
  - Like routing IP packets (Vendor Independent)
- Persistent across time and space
- Protection and memory management are independent

Protection - ACL (matrix) – uses OBJECT names -maintained by Name Server-



Note: Domain and Process
Are NOT unique

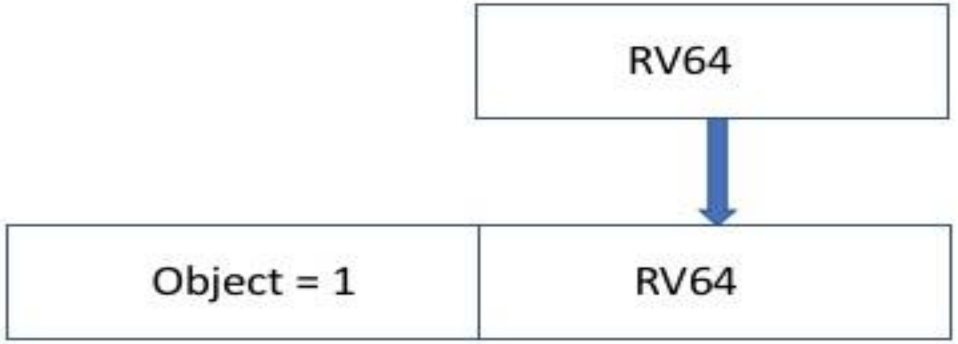
2016_NOV_RISCV_WORKSHOP 26

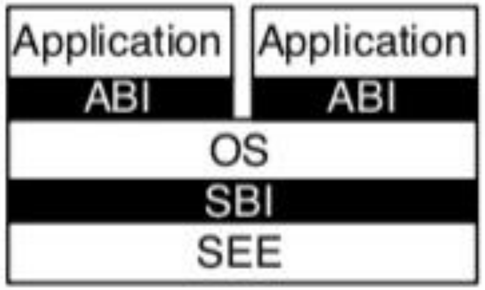
Compatibility with RV64

- Execute RV64 a.out. [8]
- Map 64 bit virtual into 128 virtual
- System calls to intermediate server
  - 64 bit pointers to 128 and then kernel call
- Use RV64 utilities (e.g. Editor) to create RV128 files

Mapping RV64 and System Calls

- RV64 mapped to RV128





Level	Encoding	Name	Abbreviation
0	00	User/Application	U
1	01	Supervisor	S
2	10	*Reserved*	
3	11	Machine	M

Table 1.1: RISC-V privilege levels.

Secure Micro- Architecture

- RV128 to include static object to denote kernel (system wide) and 32 and 64 bit address spaces
  - Object = 0 - Kernel Object (static across all nodes)
  - Object = 1 - RV64 (static across all nodes)
  - Object = 2 - Rv32 (static across all nodes)
  - When and where appropriate assign object id's to deal with semantics of high level operations as well as ACL entries
- Encrypted Memory
- Spectre
- Meltdown

Encrypted Memory

- Identified with Object ID
  - Much like with files, select which objects are encrypted
  - Keys kept in TRUSTED ZONE
- Two versions
  - One key – shared by all users [9]
  - Multiple Keys – Team in/out (attribute-based encryption) [3]
    - “each user’s key is associated with a tree-access structure where the leaves are associated with attributes.”
    - “A user is able to decrypt a ciphertext if the attributes associated with a ciphertext satisfy the key’s access structure.”

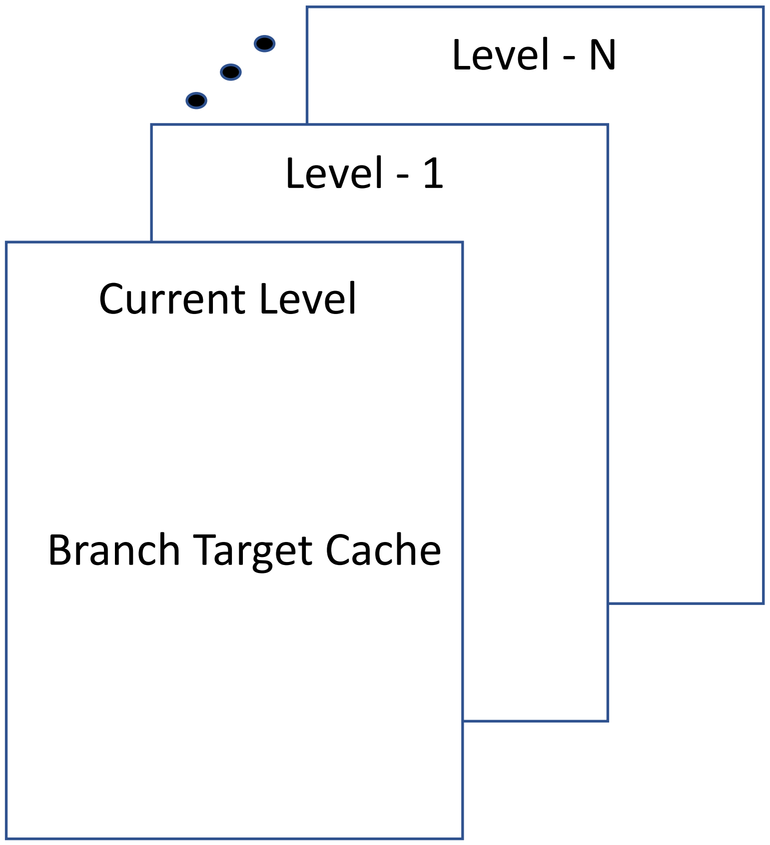
Spectre ([1])

```
branch (R1 >= bound) goto error ; Speculate branch not taken
load R2 ← memory[train+R1]      ; Speculate load & speculate cache hit
and R3 ← R2 && 0xffff           ; Speculate AND
load R4 ← memory[save+SIZE+R3]  ; Speculate load & speculate cache hit
```

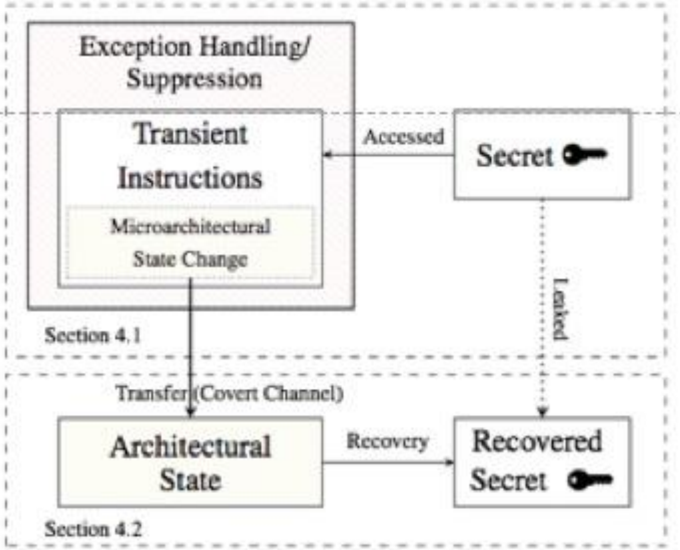
- Fix
  - If instructions after branch references OBJECT= 0, or not the current PC Object, don't speculate
  - All caches have a speculative state. Speculative loads only modify **THIS** state and **NOT** the operational state. This state is flushed or moved to operational state if branch taken

Spectre [7] - Variant 2: Branch Target Injection

- "So far, this has only been used to infer information about where code is located (in other words, to create interference from the victim to the attacker); however, the basic hypothesis of this attack variant is that it can also be used to redirect execution of code in the victim context (in other words, to create interference from the attacker to the victim; the other way around)".
- Unique BTB cache state for each subroutine level. (4 to 8)



Meltdown ([1] [7])




```
; rcx = kernel address
; rbx = probe array
retry:
mov al, byte [rcx]
shl rax, 0xc
jz retry
mov rbx, qword [rbx + rax]
```

TRAP!! (not branch) Under mis-speculation

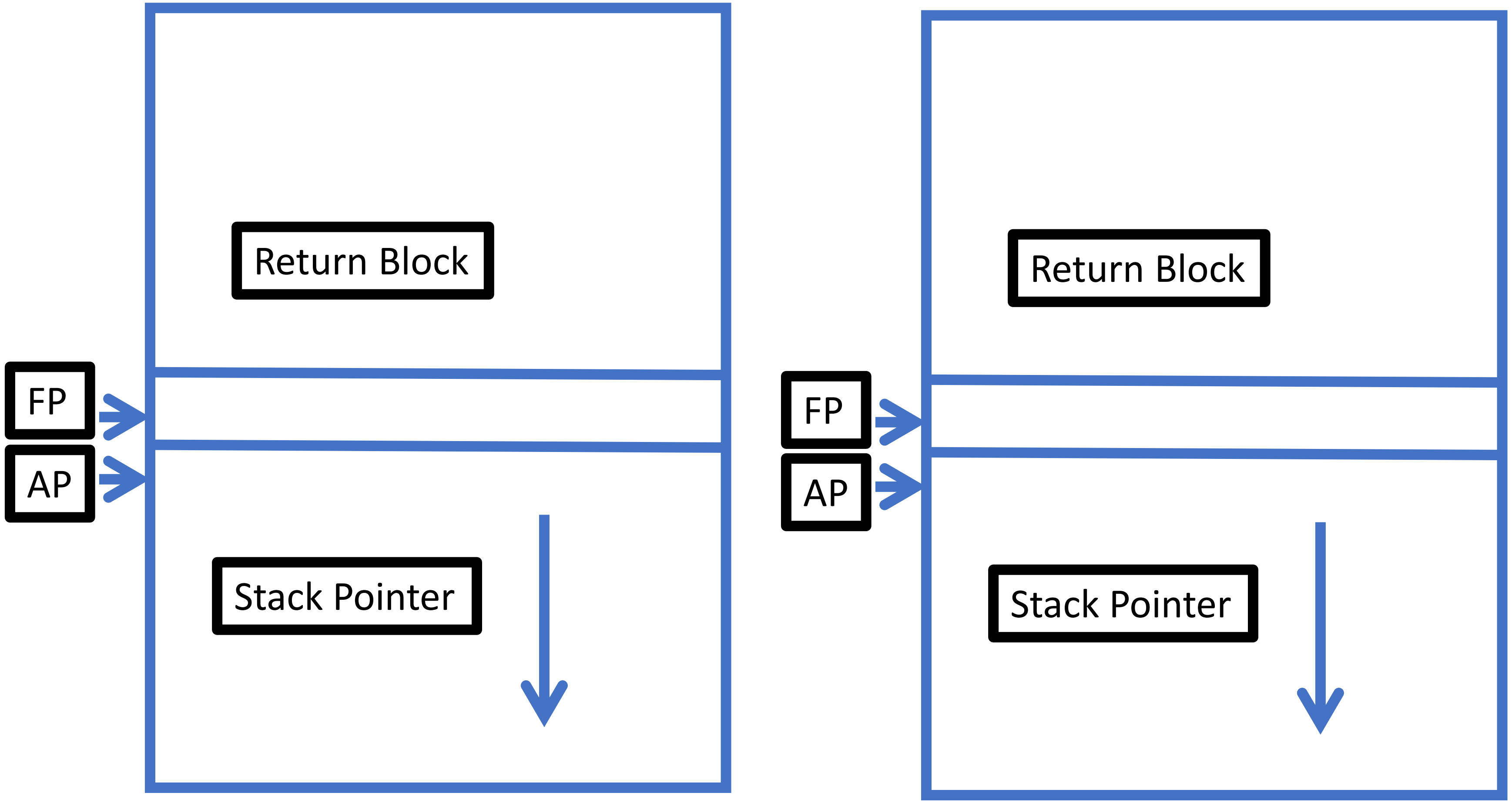
Listing 2: The core instruction sequence of Meltdown. An inaccessible kernel address is moved to a register, raising an exception. The subsequent instructions are already executed out of order before the exception is raised, leaking the content of the kernel address through the indirect memory access.

- "At first order, meltdown happens when a kernel address is manipulated in user space. In a flat address space, there are no protection mechanism that inhibit this. In a flat address architecture, one must access PTE (page table entries) to determine various protection authorizations." (variant 3 [7]).
- A Load referencing Object=0, traps if PC Object $\neq$ 0. No branches when speculating. (icache has branch or not determines performance)
- RV128 indexing is modulo 2^{64}
- Object number compares by OS system call processing, precludes passed pointer being kernel address. Unforgeable

Shadow Stack

- Independent, somewhat of address proposal
- Solves classical virus/malware attacks
  - Heap Overflow
    - Change return address
  - Writing over Stack
  - Heap Fung Hui Attacks [4]
- Hardware protected SP, AP, FP
  - Akin to domain crossing
  - Return via shadow stack

Shadow Stack - Architecture



Shadow Stack (Hardware Maintained)
Different Domain, Same virtual address
Return via THIS STACK

User Visible Stack
Can not write/read Shadow Stack

Summary

- WHAT IS ARCHITECTURE 3.0?
  - First we need to finish Architecture 2.0
    - **Language Based Security [10]**
  - We all need to read [6]
  - Then we can think about **<u>Technological Singularity & Security Supremacy</u>**

References

[1] Mark Hill, https://www.sigarch.org/a-primer-on-the-meltdown-spectre-hardware-security-design-flaws-and-their-important-implications/

[2] R. Stanley Williams, “The End of Moore’s Law”, Computing in Science & Engineering, IEEE CS and AIP, March/April 2017

[3] Vipul Goyal, Omkant Pandey , Amit Sahai, Brent Waters, “Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data”. https://eprint.iacr.org/2006/309.pdf

[4] https://en.wikipedia.org/wiki/Heap_feng_shui

[5] https://riscv.org/2016/12/5th-risc-v-workshop-proceedings/

[6] Schroder & Saltzer, “The protection of information in computer systems”, PROCEEDINGS OF THE IEEE, VOL. 63, NO. 9, SEPTEMBER 1975

[7] https://googleprojectzero.blogspot.com/2018/01/reading-privileged-memory-with-side.html

[8] https://en.wikipedia.org/wiki/Data_General_AOS

[9] https://www.micron.com/products/nonvolatile-memory-security

[10] https://en.wikipedia.org/wiki/Language-based_security