SEC RISC-V Workshop 2020

# Real-time Thread Isolation and Trusted Execution on Embedded RISC-V

**RISE Research Institutes of Sweden**

***Cybersecurity Lab***

Samuel Lindemer, Gustav Midéus, Shahid Raza

First International Workshop on Secure RISC-V Architecture Design Exploration (SECRISC-V'20)

# Overview

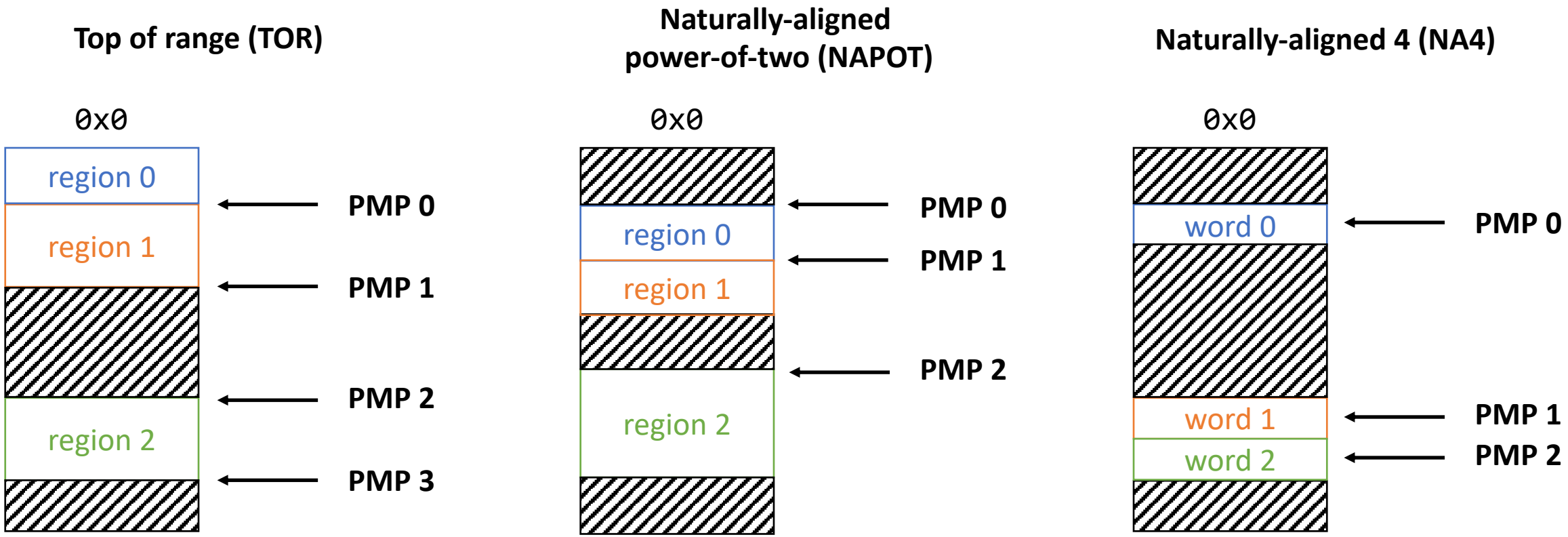
- RISC-V Physical Memory Protection (PMP) is not yet supported by many embedded operating systems.
- We implemented support for *userspace* on Zephyr RTOS using PMP.
  - Threads are sandboxed to prevent a malicious compromise.
  - Programming errors cause thread termination rather than silent kernel corruption.
- We propose an extension to the RISC-V ISA.
  - Standard RISC-V cannot support userspace and *secure enclaves* simultaneously.
  - This could be done with a two-tiered PMP design.

SEC RISC-V Workshop 2020

# Implementing Userspace on Zephyr RTOS

- Motivation for this work
- Performance
- Quirks of the RISC-V ISA

# PMP Addressing Modes




RISC-V supports up to 16 PMP registers, though some implementations have 8.

# Current RTOS Support for RISC-V PMP

- Most open source RTOSes support RISC-V architectures, but
  - all code (kernel and threads) run in M-mode
  - PMP hardware is never used
- Hardware-backed memory protection requires support from the build system.
  - TOR addressing is the easiest to implement.
  - NAPOT addressing requires perfect alignment of thread stacks.
- Zephyr RTOS has a robust build system for memory alignment.

| | 32-bit | 64-bit | PMP |
|---|---|---|---|
| **FreeRTOS** | ✓ | X | ✓ |
| **Mynewt** | ✓ | X | X |
| **NuttX** | ✓ | X | X |
| **Pharos** | X | ✓ | X |
| **RIOT** | ✓ | X | X |
| **Tock** | ✓ | X | TOR only |
| **Zephyr RTOS** | ✓ | ✓ | ✓* **(this work)** |

* pull request under review

# Zephyr RTOS System Call Performance

- Our userspace implementation supports both TOR and NAPOT.
  - Other architectures (ARC, Arm) only support NAPOT regions.
- System call performance
  - With TOR enabled, instructions do not execute in a predictable number of cycles.
  - NAPOT is significantly faster to compute in hardware.

| Mode | PMP | Instructions | CPU Cycles |
|---|---|---|---|
| M | N/A | 4 | 7 |
| U | TOR | 401 | 10000 to 20000 |
| U | NAPOT | 401 | 5276 |

*Zephyr RTOS system call latency on the HiFive1 Rev B. These results may not reflect the final upstream code.*

# Problem: Which state am I in?

- Zephyr kernel API calls begin by checking the current privilege level.
  - This facilitates a single API for both privileged and unprivileged code.
  - Calls from unprivileged code trap into the system call handler.
- **There is no way to check the current privilege level in RISC-V.**
- Other architectures (ARC, Arm and x86) can do this in one instruction.
- This results in a significant performance overhead and hacky software workarounds.

# Workaround 1 of 3: Implement a dedicated system call.

```
csrr t0, mcause
li t1, RISCV_USER_ECALL
li t2, RISCV_MACHINE_ECALL
li t3, __SYSCALL_IS_USER_CONTEXT
beq t0, t1, handle_user_syscall
beq t0, t2, handle_machine_syscall

handle_user_syscall:
        beq a7, t3, is_user_context
        ...
handle_machine_syscall:
        beq a7, t3, is_machine_context
        ...
```

*The **mcause** register indicates the reason the CPU has trapped into the ISR.*

*The **a7** register contains the system call ID.*

# Workaround 2 of 3: Attempt a restricted operation.

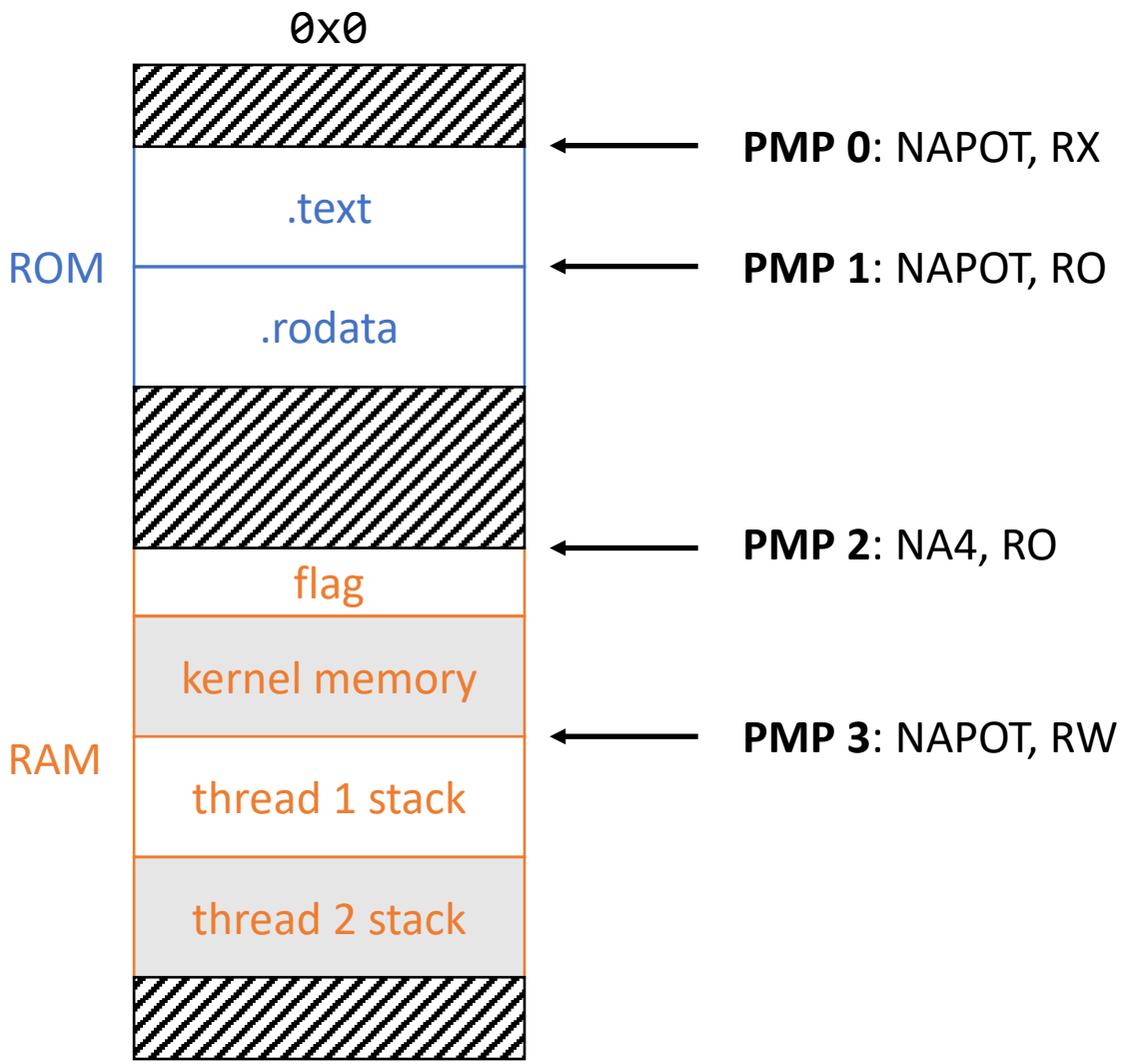
```
SECTION_FUNC(TEXT, arch_is_user_context)

arch_is_user_context_fault_start:
      csrr t0, mstatus
arch_is_user_context_fault_end:
      li a0, 0
      ret
arch_is_user_context_fixup:
      li a0, 1
      ret
```

*This operation might fault.*

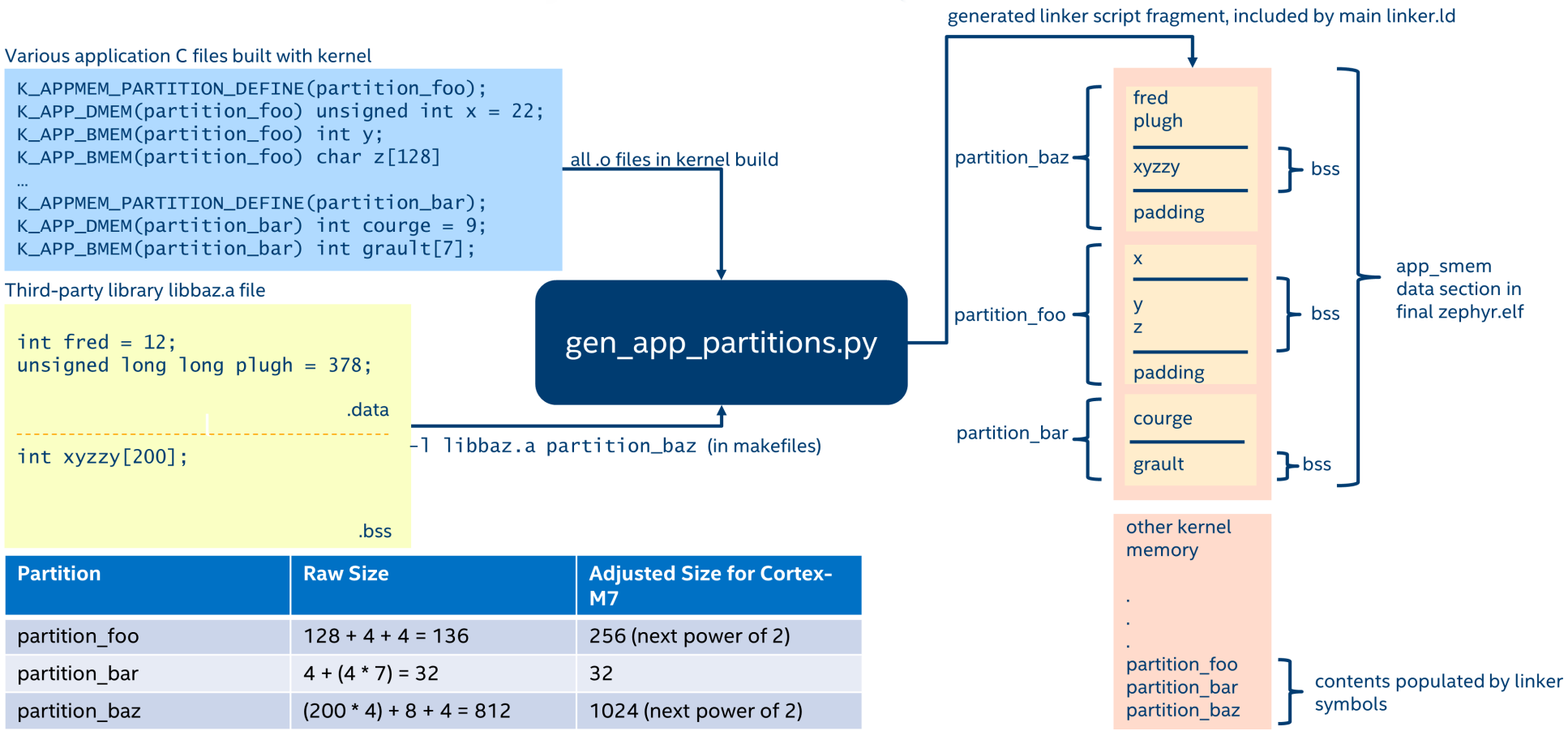
*Jump here from ISR if a fault occurred.*

# Workaround 3 of 3: Use a global flag.

- The CPU state is stored as a global flag in RAM.
  - Requires one PMP NA4 register.
  - The flag is toggled on each context switch.
- Better performance than the alternatives.



# Zephyr RTOS Memory Domains




| Partition | Raw Size | Adjusted Size for Cortex-M7 |
|---|---|---|
| partition_foo | 128 + 4 + 4 = 136 | 256 (next power of 2) |
| partition_bar | 4 + (4 * 7) = 32 | 32 |
| partition_baz | (200 * 4) + 8 + 4 = 812 | 1024 (next power of 2) |

https://docs.zephyrproject.org/latest/reference/usermode/memory_domain.html

SEC RISC-V Workshop 2020

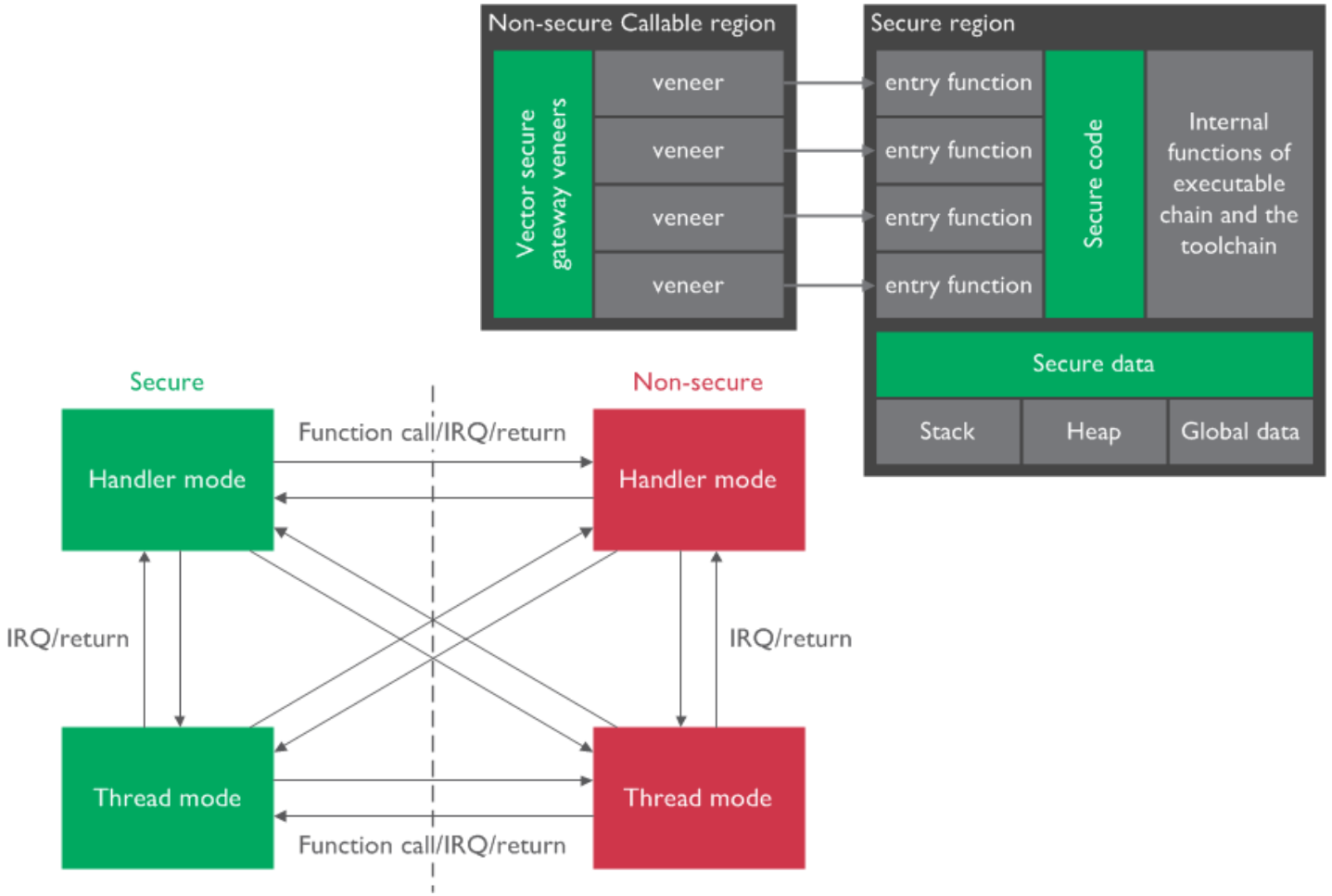
# Enclave Support in Embedded RISC-V Architectures

- Overview of related architectures
- Proposal for a new RISC-V based enclave architecture
- Implementation plan

# Related: Arm TrustZone-M™

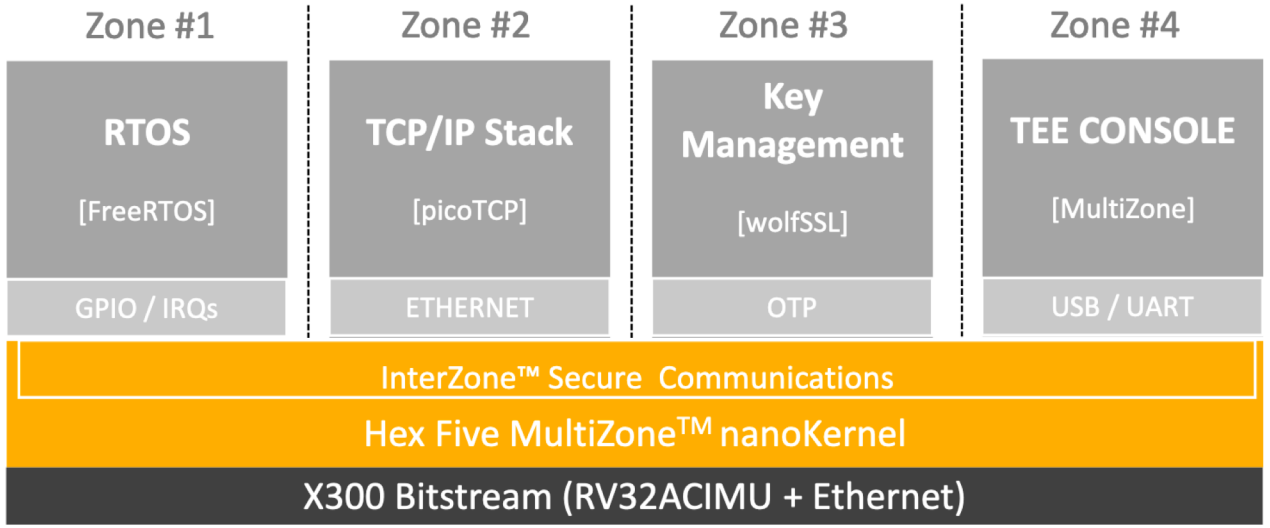
https://developer.arm.com/documentation/100690/0200/Switching-between-Secure-and-Non-secure-states?lang=en

- Each *world* has a dedicated MPU.
  - The *non-secure* world supports an RTOS (e.g., Zephyr).
  - The *secure* world runs critical applications (see Trusted Firmware-M).
- Untrusted code can make secure API calls via a *secure gateway*.
- This architecture supports <u>both</u> userspace memory protection and secure enclaves.
- *Can this be done on RISC-V?*
  - Not without extensions.



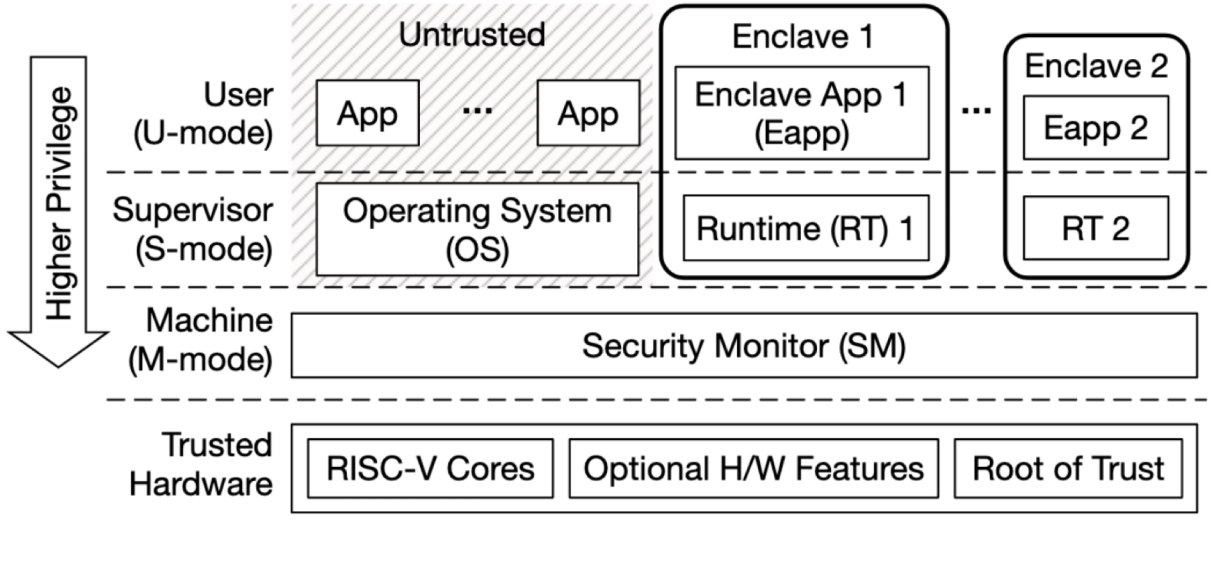
# Related: HexFive MultiZone™

- Application code runs in U-mode *zones* enforced by PMP.
  - Context switching is handled by a *security monitor* in M-mode.
  - The applications (e.g., an RTOS) cannot reconfigure PMP.
- This approach is <u>incompatible</u> with userspace threads.
- *Is this a reasonable tradeoff?*
  - We would argue no.




https://content.riscv.org/wp-content/uploads/2019/03/15.05-RISC-V-Security-Multizone-v-TrustZone-3-12-19.pdf

# Related: Keystone Enclave

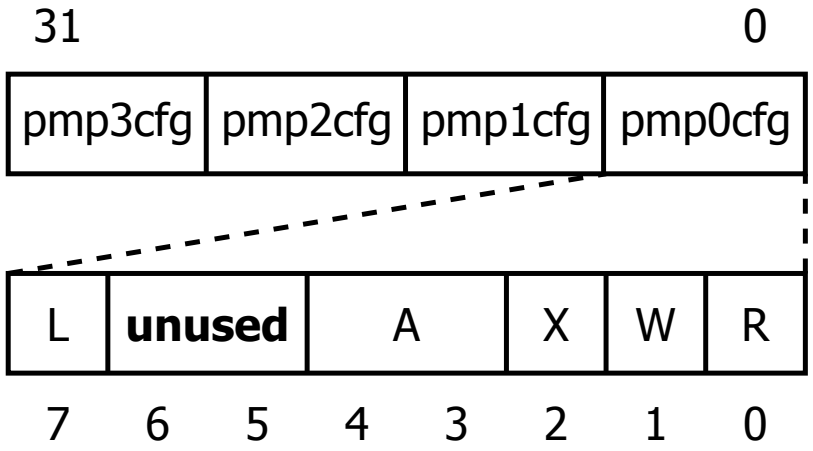


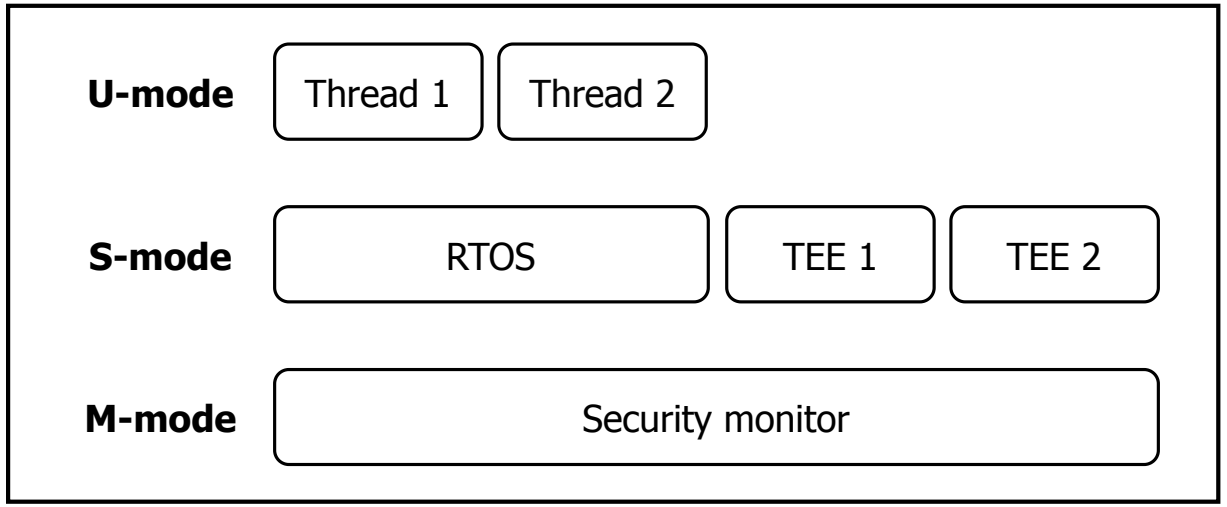

Dayeol Lee, David Kohlbrenner, Shweta Shinde, Krste Asanović, and Dawn Song. 2020. Keystone: an open framework for architecting trusted execution environments. In *Proceedings of the Fifteenth European Conference on Computer Systems* (*EuroSys '20*). Association for Computing Machinery, New York, NY, USA, Article 38, 1–16. DOI:https://doi.org/10.1145/3342195.3387532

- The OS and secure enclaves run in S-mode, separated by PMP.
- U-mode processes are protected by the MMU.
- Constrained embedded systems do not have an MMU.

# Proposed ISA Enhancement

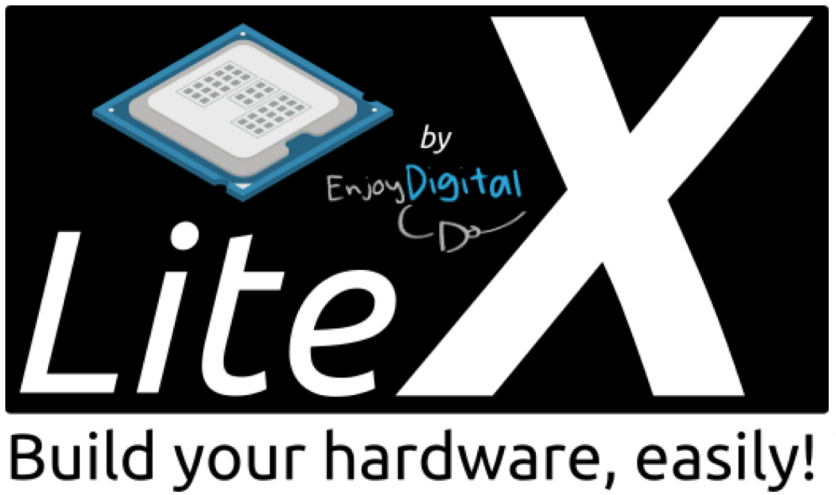
- We propose a modification to the ***pmpcfgX*** registers.
  - Bits 5 and 6 of each region field are currently unused.
  - One of these can be used as an S-mode *enable* flag.
  - S-mode would subdivide its memory space to restrict U-mode.
- This would facilitate embedded RISC-V implementations with three privilege levels.

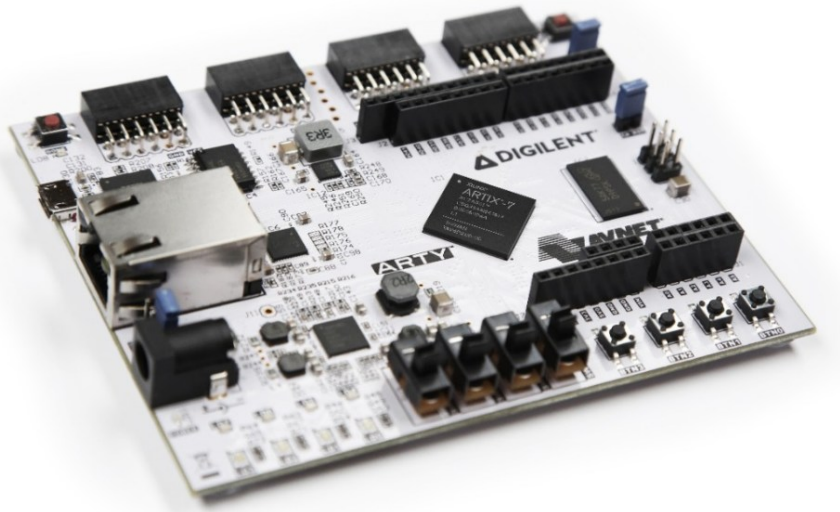


# Ongoing Work

- Userspace port to LiteX VexRiscv
  - VexRiscv is an open source embedded RISC-V implementation written in SpinalHDL.
  - Zephyr supports this platform but userspace has only been tested on the HiFive1 Rev B.
- S-mode PMP prototype on VexRiscv
- Software support for the new architecture
  - Viable security monitor and enclaves.
  - Additional Zephyr modifications.



SEC RISC-V Workshop 2020

# Thank you!

Samuel Lindemer

samuel.Lindemer@ri.se