Task 7

Write a test to verify smepmp extension behaviour (Smepmp task)

a. Pre-requisites

- Read RISCV privileged architecture spec for PMP
- Read about Smepmp extension

b. Write an assembly test.

- Start your test in M mode.
- Set mseccfg.MML
- Add a rule with executable privileges that either is M-mode-only or a locked Shared-Region?? what happens and why
- How can you solve the above problem

Github Link:

https://github.com/BilalAli10x/Riscv_Arch_Test/tree/main/BilalAli_Task7

Test Description

This test verifies the behavior of the Smepmp (Supervisor Mode Execution Prevention PMP) extension, specifically:

1. How enabling mseccfg.MML before configuring PMP affects execution.
2. How configuring PMP before enabling MML allows proper execution.

The goal is to demonstrate the order-dependency: MML changes how PMP regions interpret privileges, even in M-mode.

Q): Add a rule with executable privileges that either is M-mode-only or a locked Shared-Region? what happens and why?

Observed Behavior:

- Setting MML before configuring the PMP region triggers an instruction access fault:

```
core   0: 0x8000000c (0x00100293) li      t0, 1
core   0: 3 0x8000000c (0x00100293) x5  0x00000001
(spike)
core   0: 0x80000010 (0x74729073) csrw    mseccfg, t0
core   0: 3 0x80000010 (0x74729073) c1863_mseccfg 0x00000001
(spike)
core   0: exception trap_instruction_access_fault, epc 0x80000014
core   0:           tval 0x80000014
(spike)
```

Reason:

When mseccfg.MML (Machine Mode Lockdown) is set to 1, the rules for memory access change instantly:

- Setting MML automatically enables the behavior of MMWP (Machine Mode Whitelist Policy).
- MML sets Machine Mode Lockdown, making PMP rules strictly enforceable.

- If PMP region is configured after MML is set, M mode loses its default “allow-all” permission. It can now only execute or access memory regions that are explicitly defined in PMP rule.
- At address **0x80000010,** enabled MML. And at address **0x80000014**, the processor tries to fetch the next instruction **(la t0, _data_end)**. Since nott configured **pmpcfg0** yet, there is no "whitelist" entry allowing execution at 0x80000014. The hardware sees a fetch to an unprotected region and triggers the **trap_instruction_access_fault**.

Q):How can you solve the above problem

➢ Solution:

- **Step 1: Configure PMP regions first**
  ```
  la t0, _data_end
  srli t0, t0, 2
  csrw pmpaddr0, t0
  li t4, 0x8C     # TOR + Execute-only
  csrw pmpcfg0, t4
  ```
- **Step 2: Enable Machine Mode Lockdown (MML) afterwards**
  ```
  li t0, 1
  csrw mseccfg, t0
  ```
- **Effect:**
  - Execution in M-mode works correctly.
  - No instruction access fault occurs.
  - PMP rules are already in place before enabling MML, ensuring proper enforcement.

Results:

- MML changes how PMP rules are enforced by making them strictly applied, even in M-mode.
- If MML is set before configuring PMP, any newly added PMP region may immediately restrict execution access.
- This leads to an instruction access fault because the processor checks the PMP configuration under lockdown rules.
- By configuring the PMP region before enabling MML, the intended permissions are properly established.
- When MML is later enabled, it applies on the already-correct configuration rather than applying restrictions to an incomplete setup.

- This confirms that SMEPMP behavior is order-dependent, and correct initialization sequence is essential to avoid execution faults.

Reference to Specification

RISC-V Privileged Architecture Specification

- **Section 3.7:PMP Section** - Physical Memory Protection CSRs
- **Section 3.7.1:PMP Section** - Address matching and L-bit behavior

SMEPMP Specification

- **Chapter 2:** Proposal behavior of bit 1