 # Author : Muhammad Imran
 # Date: 18-02-2026
 # Module : RISCV Arch Test
 # Section: RISCV Arch Test
 # Task Name: Task 6

Github: Task 6

Test Description:

This test verifies the behavior of a locked PMP region and demonstrates how the Smepmp extension (mseccfg.RLB bit) allows modification of a locked PMP entry without reset.

Output Explanation:

The test starts in Machine Mode and sets up a trap vector to handle expected memory access faults (Load, Store, and Instruction Access Faults).

Then it sets the mseccfg.RLB bit so that the locked PMP entries can be re-configured later.

```
core   0: 3 0x8000000c (0x0340006f)
core   0: 0x80000040 (0x00400293) li      t0, 4
core   0: 3 0x80000040 (0x00400293) x5  0x00000004
core   0: 0x80000044 (0x74729073) csrw    mseccfg, t0
core   0: 3 0x80000044 (0x74729073) c1863_mseccfg 0x00000004
```

A NAPOT region is defined for the pmpnapot code section.

The entry is configured with Full Permissions (Read, Write, Execute) and the Lock bit is set.

```
core   0: 3 0x8000004c (0xfb828293) x5  0x80002000
core   0: 0x80000050 (0x0022d293) srli    t0, t0, 2
core   0: 3 0x80000050 (0x0022d293) x5  0x20000800
core   0: 0x80000054 (0x1ff2e293) ori     t0, t0, 511
core   0: 3 0x80000054 (0x1ff2e293) x5  0x200009ff
core   0: 0x80000058 (0x3b029073) csrw    pmpaddr0, t0
core   0: 3 0x80000058 (0x3b029073) c944_pmpaddr0 0x200009ff
core   0: 0x8000005c (0x09f00293) li      t0, 159
core   0: 3 0x8000005c (0x09f00293) x5  0x0000009f
core   0: 0x80000060 (0x3a029073) csrw    pmpcfg0, t0
core   0: 3 0x80000060 (0x3a029073) c928_pmpcfg0 0x0000009f
```

Validation: The test jumps to the region to confirm that code execution, reading, and writing all work within M-Mode under a locked entry.

```
core   0: 3 0x80000064 (0x79d010ef) x1  0x80000068
core   0: 0x80002000 (0x123452b7) lui     t0, 0x12345
core   0: 3 0x80002000 (0x123452b7) x5  0x12345000
core   0: 0x80002004 (0x67828293) addi    t0, t0, 1656
core   0: 3 0x80002004 (0x67828293) x5  0x12345678
core   0: 0x80002008 (0x55500313) li      t1, 1365
core   0: 3 0x80002008 (0x55500313) x6  0x00000555
core   0: 0x8000200c (0x00000397) auipc   t2, 0x0
core   0: 3 0x8000200c (0x00000397) x7  0x8000200c
core   0: 0x80002010 (0xff438393) addi    t2, t2, -12
core   0: 3 0x80002010 (0xff438393) x7  0x80002000
core   0: 0x80002014 (0x0063a023) sw      t1, 0(t2)
core   0: 3 0x80002014 (0x0063a023) mem 0x80002000 0x00000555
core   0: 0x80002018 (0x0003ae03) lw      t3, 0(t2)
core   0: 3 0x80002018 (0x0003ae03) x28 0x00000555 mem 0x80002000
core   0: 0x8000201c (0x00008067) ret
```

QUESTION 1: What two things happen when you configure a locked pmp region?

1. PMP rules are now strictly enforced even for Machine Mode (normally M-mode has full access to all memory).

2.The pmpcfg and associated pmpaddr registers become read-only. They cannot be modified until a system reset.

But with Smepmp extension:

If mseccfg.RLB = 1, locked entries can be modified without reset. Since it is already set at the start of the code we can modify the locked PMP configurations

Enable only read permissions for locked pmp entry

```
core   0: 0x80000068 (0x09900293) li      t0, 153
core   0: 3 0x80000068 (0x09900293) x5  0x00000099
core   0: 0x8000006c (0x3a029073) csrw    pmpcfg0, t0
core   0: 3 0x8000006c (0x3a029073) c928_pmpcfg0 0x00000099
```

Now according to the updated configuration M-mode will follow the same PMP rule and can not write and execute the locked PMP region .

Trying to fetch an instruction from the PMP region with no **X** permissions will give **inst_access_fault**

```
core   0: 3 0x80000070 (0x791010ef) x1  0x80000074
core   0: exception trap_instruction_access_fault, epc 0x80002000
core   0:           tval 0x80002000
core   0: >>>>  trap_vector
core   0: 0x80000010 (0x342022f3) csrr    t0, mcause
```

Trying to read from the PMP region with only **R** permissions will read the PMP region successfully.

```
core   0: 3 0x80000074 (0x00002297) x5  0x80002074
core   0: 0x80000078 (0xf8c28293) addi    t0, t0, -116
core   0: 3 0x80000078 (0xf8c28293) x5  0x80002000
core   0: 0x8000007c (0x0002a303) lw      t1, 0(t0)
core   0: 3 0x8000007c (0x0002a303) x6  0x00000555 mem 0x80002000
```

Trying to store will also give store_access_fault because it do not have **W** permissions

```
core   0: 3 0x80000084 (0xaaa38393) x7  0x000aaaaa
core   0: 0x80000088 (0x0072a023) sw      t2, 0(t0)
core   0: exception trap_store_access_fault, epc 0x80000088
core   0:           tval 0x80002000
core   0: >>>>  trap_vector
```

Finally the program exited in machine mode.