

RISC-V Arch Test

Task 1

Test Description:

This test verifies correct privilege mode switching and trap handling in a RISC-V system using Machine, Supervisor, and User modes.

The test starts execution in Machine mode, installs a machine-mode trap handler, and defines a function `switch_mode()` that transitions execution to Supervisor mode or User mode based on an input argument.

The function `switch_mode(a0)` is implemented to abstract privilege switching.

- If $a0 = 0$, execution switches from Machine mode to Supervisor mode.
- If $a0 = 1$, execution switches from Machine mode to User mode.

This function is only invoked in Machine mode and internally programs `mstatus.MPP`, `mepc`, and executes `mret` to perform the transition.

Privilege transitions are performed using standard RISC-V mechanisms:

- `mstatus.MPP` to specify the next privilege level
- `mepc` to specify the return address
- `mret` to perform the privilege switch

Transition from Supervisor mode to User mode is performed using standard S-mode mechanisms:

- `sstatus.SPP` to specify the next privilege level
- `sepc` to specify the return address
- `sret` to perform the privilege switch

The test further verifies correct execution mode by issuing ecall instructions from Supervisor and User modes. These ECALLs generate traps into Machine mode, where the trap handler examines the mcause register and returns execution to the next higher privilege mode:

- ECALL from User mode returns to Supervisor mode
- ECALL from Supervisor mode returns to Machine mode

At the end, the test ensures that execution always exits in Machine mode, as required by RISC-V compliance tests.

Program Flow:

- Program starts in Machine mode after reset (`_start -> main`).
- Machine-mode trap handler address is written to mtvec.
- `switch_mode(0)` is called to switch from Machine -> Supervisor.
- From Supervisor mode, the program switches to User mode using `sret`.
- An ecall is executed in User mode to verify the current privilege level.

0 0	8 9	Environment call from U-mode Environment call from S-mode
--------	--------	--

- Trap handler detects `mcause = 8` (User ECALL) and returns execution to Supervisor mode.
- Another ecall is executed in User mode.
- Trap handler detects `mcause = 9` (Supervisor ECALL) and returns execution to Machine mode.
- Test exits successfully from Machine mode by writing PASS signature to `tohost`.

Trap handler:

```
trap_vector:  
  
    csrr t0, mcause           /* Read trap cause */  
    csrr t1, mepc             /* Read exception PC */  
  
    /* Check USER ECALL (mcause = 8) */  
    li t2, 8  
    beq t0, t2, handle_user_ecall  
  
    /* Check for SUPERVISOR ECALL (mcause = 9) */  
    li t2, 9  
    beq t0, t2, handle_supervisor_ecall  
  
    /* Unexpected trap failure */  
    j test_fail
```

Inside the trap handler, the mcause register is read to determine the source of the exception.

When mcause = 8, the trap handler identifies the exception as a User ECALL.

```
80002008: 00800393          addi  t2,zero,8  
8000200c: 00728863          beq   t0,t2,8000201c <handle_user_ecall>  
  
core 0: 0x80002008 (0x00800393) li      t2, 8  
core 0: 3 0x80002008 (0x00800393) x7 0x00000008
```

When mcause = 9, the trap handler identifies the exception as a Supervisor ECALL.

```
80002010: 00900393          addi  t2,zero,9  
80002014: 02728a63          beq   t0,t2,80002048 <handle_supervisor_ecall>  
  
core 0: 0x80002010 (0x00900393) li      t2, 9  
core 0: 3 0x80002010 (0x00900393) x7 0x00000009
```

handle_user_ecall:

When user_ecall is detected in trap handler. This handler will first clear the MPP bits and then set the MPP bits to supervisor mode as given in the implementation requirement that the handler will jump to 1 higher mode.

```
handle_user_ecall:  
    /* Return to Supervisor Mode */  
    csrr t3, mstatus  
    li  t4, ~(3 << 11)  
    and t3, t3, t4                      /* Clear MPP */  
    li  t4, (1 << 11)                   /* MPP = S(01) */  
    or   t3, t3, t4  
    csrw mstatus, t3  
  
    addi t1, t1, 4                      /* Skip ecall */  
    csrw mepc, t1  
    mret
```

```
8000202c: 00001eb7          lui    t4,0x1  
80002030: 800e8e93          addi   t4,t4,-2048 # 800 <_start-0x7fffff800>
```

```
core 0: 0x80002030 (0x800e8e93) addi t4, t4, -2048  
core 0: 3 0x80002030 (0x800e8e93) x29 0x000000800
```

```
core 0: >>> user_entry  
core 0: 0x8000001c (0x00000073) ecall  
core 0: exception trap_user_ecall, epc 0x8000001c
```

handle_supervisor_ecall:

When supervisor_ecall is detected in trap handler. This handler will first clear the MPP bits and then set the MPP bits to machine mode as given in the implementation requirement that the handler will jump to 1 higher mode.

```
handle_supervisor_ecall:  
    /* Return to Machine mode */  
    csrr t3, mstatus  
    li  t4, ~(3 << 11)  
    and t3, t3, t4                      /* Clear MPP */  
    li  t4, (3 << 11)                   /* MPP = M(11) */  
    or   t3, t3, t4  
    csrw mstatus, t3  
  
    addi t1, t1, 4  
    csrw mepc, t1  
    mret
```

```

80002058: 00002eb7          lui    t4,0x2
8000205c: 800e8e93          addi   t4,t4,-2048 # 1800 <_start-0x7ffffe800>

core 0: 0x8000205c (0x800e8e93) addi t4, t4, -2048
core 0: 3 0x8000205c (0x800e8e93) x29 0x00001800

core 0: >>> s_after_user
core 0: 0x80000020 (0x00000073) ecall
core 0: exception trap_supervisor_ecall, epc 0x80000020

```

switch_mode() function:

This function switch_mode(a0) is implemented to abstract privilege switching.

- If a0 = 0, execution switches from Machine mode to Supervisor mode.

```

set_supervisor:
    li    t1, (1 << 11)           /* MPP = Supervisor (01) */
    or    t0, t0, t1
    csrw mstatus, t0
    la    t2, supervisor_entry
    csrw mepc, t2
    mret

```

set_supervisor set the MPP bit to Supervisor mode (01)

```

80000058 <set_supervisor>:
80000058: 00001337          lui    t1,0x1
8000005c: 80030313          addi   t1,t1,-2048 # 800 <_start-0x7fffff800>

core 0: 0x8000005c (0x80030313) addi t1, t1, -2048
core 0: 3 0x8000005c (0x80030313) x6 0x00000800

```

It then jumps to supervisor_entry label which will switch the mode from supervisor to user using sstatus and sret.

```

80000088: 00000397          auipc t2,0x0
8000008c: f9438393          addi   t2,t2,-108 # 8000001c <user_entry>

```

```
core 0: 0x8000008c (0xf9438393) addi    t2, t2, -108
core 0: 1 0x8000008c (0xf9438393) x7  0x8000001c
```

In user mode, when there comes an ecall which will eventually jumps back to supervisor mode and another ecall will make the privilege level goes back to Machine mode after that it will jump to test_pass and execution will stop.

Test_pass:

```
test_pass:
    li gp, 0x55555555      /* signature for test pass */
    sw gp, tohost, t0
    j write_tohost
```

On successful program execution, it will jump to test_pass label which is loading a value 0x55555555 as a pass signature to the register gp. This value is only a marker which symbolizes that the test is passed.

```
80000098 <test_pass>:
80000098:  555551b7          lui    gp,0x55555
8000009c:  55518193          addi   gp,gp,1365 # 55555555 <_start-0x2aaaaaab>
```

```
core 0: 0x8000009c (0x55518193) addi    gp, gp, 1365
core 0: 3 0x8000009c (0x55518193) x3  0x55555555
```