

```
# Author : Muhammad Imran
# Date: 05-02-2026
# Module : RISCV Arch Test
# Section: RISCV Arch Test
# Task Name: Task 1
```

### Test Description:

This test implements a function which switches the privilege mode based on the input argument. If the argument is '0' it switches the privilege mode to supervisor mode modifying the **mstatus**, **mepc**, and **mret** CSR registers and if the argument is '1' it switches the privilege mode to User mode. It also implements the trap vector and trap handler function which jumps to the appropriate trap handler based on the exception raised. If there is User **ECALL** exception it jumps to the **u\_trap\_handler** which stores the context using **REG\_STORE**, handles the exception restore the context using **REG\_RESTORE** and return to 1 higher privilege mode, the trap handler itself runs in the machine mode, the trap delegation is not used. If there is Supervisor **ECALL** exception it jumps to the **s\_trap\_handler** which handles the exception and return to 1 higher privilege mode.

### Actual Output:

**\_start** is the entry point of the program. Which initializes the stack and setups the **mtvec** for trap handling. Any trap occurs during the execution of the program the program will jump to the address store in **mtvec**. Writing to **mtvec** verifies that the program starts in **M-mode[1]** because mtvec is machine mode csr and it can only be accessed in the machine mode. After that it jumps to the **main** from where the actual program execution starts.

```
1 core 0: 3 0x00001000 (0x00000237) x5 0x00001000
2 core 0: 0x000001004 (0x02028593) addi    a1, t0, 32
3 core 0: 3 0x000001004 (0x02028593) x11 0x00001020
4 core 0: 0x000001008 (0xf1402573) csrr    a0, mhartid
5 core 0: 3 0x000001008 (0xf1402573) x10 0x00000000
6 core 0: 0x00000100c (0x0182a283) lw      t0, 24(t0)
7 core 0: 3 0x00000100c (0x0182a283) x5 0x80001000 mem 0x000001018
8 core 0: 0x000001010 (0x00028067) jr      t0
9 core 0: 3 0x000001010 (0x00028067)
10 core 0: 0x80001000 (0x00006117) auipc   sp, 0x6
11 core 0: 3 0x80001000 (0x00006117) x2 0x80007000
12 core 0: 0x80001004 (0x00010113) mv      sp, sp
13 core 0: 3 0x80001004 (0x00010113) x2 0x80007000
14 core 0: 0x80001008 (0x00000317) auipc   t1, 0x0
15 core 0: 3 0x80001008 (0x00000317) x6 0x80001008
16 core 0: 0x8000100c (0x01030313) addi    t1, t1, 16
17 core 0: 3 0x8000100c (0x01030313) x6 0x80001018
18 core 0: 0x80001010 (0x30531073) csrw    mtvec, t1
19 core 0: 3 0x80001010 (0x30531073) c773_mtvec 0x80001018
20 core 0: 0x80001014 (0x0a40006f) j       pc + 0xa4
21 core 0: 0x80001014 (0x0a40006f)
```

**main** sets up the argument to call the **switch\_mode** function. Initially the argument is set to '0'. Finally it calls the **switch\_mode** function.

```
26 core 0: 3 0x800010bc (0x00000013)
27 core 0: 0x800010c0 (0x00000513) li    a0, 0
28 core 0: 3 0x800010c0 (0x00000513) x10  0x00000000
29 core 0: 0x800010c4 (0x008000ef) jal   pc + 0x8
30 core 0: 3 0x800010c4 (0x008000ef) x1   0x800010c8
31 core 0: 0x800010cc (0x300022f3) csrr  t0, mstatus
```

**switch\_mode** The **switch\_mode** function is called only in Machine mode. It updates the **mstatus MPP** field to select the target privilege mode and writes the destination address into **mepc**. Executing **mret** completes the privilege transition. When **a0=0**, execution switches to Supervisor mode, and when **a0=1**, execution switches to User mode. **MPP[2]**

```
47 core 0: 0x8000110c (0x00008f93) mv    t6, ra
48 core 0: 3 0x8000110c (0x00008f93) x31 0x800010e8
49 core 0: 0x80001110 (0x300022f3) csrr  t0, mstatus
50 core 0: 3 0x80001110 (0x300022f3) x5   0x00000000
51 core 0: 0x80001114 (0x00300313) li    t1, 3
52 core 0: 3 0x80001114 (0x00300313) x6   0x00000003
53 core 0: 0x80001118 (0x00b31313) slli  t1, t1, 11
54 core 0: 3 0x80001118 (0x00b31313) x6   0x00001800
55 core 0: 0x8000111c (0xffff34313) not   t1, t1
56 core 0: 3 0x8000111c (0xffff34313) x6   0xffffe7ff
57 core 0: 0x80001120 (0x0062f2b3) and   t0, t0, t1
58 core 0: 3 0x80001120 (0x0062f2b3) x5   0x00000000
59 core 0: 0x80001124 (0x00100f13) li    t5, 1
60 core 0: 3 0x80001124 (0x00100f13) x30  0x00000001
61 core 0: 0x80001128 (0x00050663) beqz a0, pc + 12
62 core 0: 3 0x80001128 (0x00050663)
63 core 0: 0x8000112c (0x03e50063) beq   a0, t5, pc + 32
64 core 0: 3 0x8000112c (0x03e50063)
65 core 0: 0x8000114c (0x341f9073) csrw  mepc, t6
66 core 0: 3 0x8000114c (0x341f9073) c833_mepc 0x800010e8
67 core 0: 0x80001150 (0x30029073) csrw  mstatus, t0
68 core 0: 3 0x80001150 (0x30029073) c768_mstatus 0x00000000
69 core 0: 0x80001154 (0x30200073) mret
70 core 0: 3 0x80001154 (0x30200073) c768_mstatus 0x00000080 c784_mstatush 0x00000000
```

**mepc** stores the next instruction to execute after **mret**. **mstatus** updates the privilege bits to allow execution in **S-mode**. **mret** switches to **S-mode** at the address stored in **mepc**.

After successfully switching to the **S\_mode** an **ECALL[3]** triggers the trap and program jumps to the **trap\_vector** using the address stored in the **mtvec** which at first stores the context using **SAVE\_REG** macro on the stack

```

77  core  0: 0x800010f4 (0x00000073) ecall
78  core  0: exception trap_user_ecall, epc 0x800010f4
79  core  0: >>> trap_vector
80  core  0: 0x80001018 (0xf8010113) addi    sp, sp, -128
81  core  0: 3 0x80001018 (0xf8010113) x2  0x80006f80
82  core  0: 0x8000101c (0x00112223) sw     ra, 4(sp)
83  core  0: 3 0x8000101c (0x00112223) mem   0x80006f84 0x800010e8
84  core  0: 0x80001020 (0x00212423) sw     sp, 8(sp)
85

```

Then the trap vector reads the **mcause** csr and discards the MSB bit because of exception. And jumps to the **s\_trap\_handler**.

```

144 core  0: 0x80001098 (0x342022f3) csrr   t0, mcause
145 core  0: 3 0x80001098 (0x342022f3) x5  0x00000008
146 core  0: 0x8000109c (0x00129293) slli   t0, t0, 1
147 core  0: 3 0x8000109c (0x00129293) x5  0x00000010
148 core  0: 0x800010a0 (0x0012d293) srlt   t0, t0, 1
149 core  0: 3 0x800010a0 (0x0012d293) x5  0x00000008
150 core  0: 0x800010a4 (0x00800313) li    t1, 8
151 core  0: 3 0x800010a4 (0x00800313) x6  0x00000008
152 core  0: 0x800010a8 (0x00900393) li    t2, 9
153 core  0: 3 0x800010a8 (0x00900393) x7  0x00000009
154 core  0: 0x800010ac (0x0a628663) beq   t0, t1, pc + 172

```

[2] **s\_trap\_handler** stores the address of the next instruction in the **mepc** that will be executed in machine mode after mret. Modify the **MPP** bits of the **mstatus** register to change the privilege mode to the machine mode. It can be verified by the below logs

```

156 core  0: 0x80001158 (0x341023f3) csrr   t2, mepc
157 core  0: 3 0x80001158 (0x341023f3) x7  0x800010f4
158 core  0: 0x8000115c (0x00438393) addi   t2, t2, 4
159 core  0: 3 0x8000115c (0x00438393) x7  0x800010f8
160 core  0: 0x80001160 (0x34139073) csrw   mepc, t2
161 core  0: 3 0x80001160 (0x34139073) c833_mepc 0x800010f8
162 core  0: 0x80001164 (0x300022f3) csrr   t0, mstatus
163 core  0: 3 0x80001164 (0x300022f3) x5  0x00000000
164 core  0: 0x80001168 (0x00300313) li    t1, 3
165 core  0: 3 0x80001168 (0x00300313) x6  0x00000003
166 core  0: 0x8000116c (0x00b31313) slli   t1, t1, 11

```

finally it restore the context and return to the address stored in the **mepc** using **mret** which successfully switches the mode to **M-mode** as it can be verified by the below logs

```

248 core 0: 0x800012cc (0x08010113) addi    sp, sp, 128
249 core 0: 3 0x800012cc (0x08010113) x2  0x80007000
250 core 0: 0x800012d0 (0x30200073) mret
251 core 0: 3 0x800012d0 (0x30200073) c768_mstatus 0x00000080 c784_mstatush 0x00000000
252 core 0: 0x800012f0 (0x00100000) la      t1, 1(12)

```

Now the program is again in M-mode, so we can call the **switch\_mode** function and change the privilege mode but this time it is called with argument **a0 = 1** which switches the mode to user

```

257 core 0: 3 0x80001100 (0x01de2223) mem 0x80001304 0x000000002
258 core 0: 0x80001104 (0x00100513) li     a0, 1
259 core 0: 3 0x80001104 (0x00100513) x10  0x00000001
260 core 0: 0x80001108 (0x018000ef) jal    pc + 0x18
261 core 0: 3 0x80001108 (0x018000ef) x1  0x8000110c

```

**switch\_mode** stores the address of the next instruction in **mepc** and changes the **MPP** filed of **mstatus** csr to '00' and return using **mret**. This time it will return in **U-mode**

```

279 core 0: 3 0x80001140 (0x03e50063)
280 core 0: 0x80001160 (0x341f9073) csrw   mepc, t6
281 core 0: 3 0x80001160 (0x341f9073) c833_mepc 0x8000110c
282 core 0: 0x80001164 (0x30029073) csrw   mstatus, t0
283 core 0: 3 0x80001164 (0x30029073) c768_mstatus 0x00000080
284 core 0: 0x80001168 (0x30200073) mret
285 core 0: 3 0x80001168 (0x30200073) c768_mstatus 0x00000088 c784_mstatush 0x00000000

```

After returning in **U-mode** and **ECALL** in user mode will trigger the trap and execution jumps to the **trap\_vect** using the address stored in **mtvec**

```

290 core 0: 0x80001117 (0x01de2223) sw      t1, 7(12)
291 core 0: 0 0x80001114 (0x01de2223) mem 0x80001304 0x00000003
292 core 0: 0x80001118 (0x00000073) ecall
293 core 0: exception trap_user_ecall, epc 0x80001118
294 core 0: >>> trap_vector
295 core 0: 0x80001018 (0xf8010113) addi    sp, sp, -128
296 core 0: 3 0x80001018 (0xf8010113) x2  0x80006f80
297 core 0: 0x80001018 (0xf8010113) la      t1, 1(12)

```

**Trap\_vector** stores the context using **SAVE\_REG** macro and read the **mcause**. It reads **macuse = 8** and jumps to the **u\_trap\_handler**

```

358 core 0: 3 0x80001094 (0x07f12e23) mem 0x80006ff0 0x8000110c
359 core 0: 0x80001098 (0x342022f3) csrr   t0, mcause
360 core 0: 3 0x80001098 (0x342022f3) x5  0x00000008
361 core 0: 0x8000109c (0x00129293) slli   t0, t0, 1
362 core 0: 3 0x8000109c (0x00129293) x5  0x00000010
363 core 0: 0x800010a0 (0x0012d293) srli   t0, t0, 1
364 core 0: 3 0x800010a0 (0x0012d293) x5  0x00000008
365 core 0: 0x800010a4 (0x00800313) li     t1, 8
366 core 0: 3 0x800010a4 (0x00800313) x6  0x00000008
367 core 0: 0x800010a8 (0x00900393) li     t2, 9
368 core 0: 3 0x800010a8 (0x00900393) x7  0x00000009
369 core 0: 0x800010ac (0x0c628063) beq   t0, t1, pc + 192

```

**u\_trap\_handler** stores the address of the next instruction in the **mepc** that will be executed in machine mode after mret. Modify the **MPP** bits of the **mstatus** register to change the privilege mode to the **s\_mode**. Finally it restores the context and return in s-mode. It can be verified by the below logs

```

371 core 0: 0x8000116c (0x341023f3) csrr t2, mepc
372 core 0: 3 0x8000116c (0x341023f3) x7 0x80001118
373 core 0: 0x80001170 (0x00438393) addi t2, t2, 4
374 core 0: 3 0x80001170 (0x00438393) x7 0x8000111c
375 core 0: 0x80001174 (0x34139073) csrw mepc, t2
376 core 0: 3 0x80001174 (0x34139073) c833_mepc 0x8000111c
377 core 0: 0x80001178 (0x300022f3) csrr t0, mstatus
378 core 0: 3 0x80001178 (0x300022f3) x5 0x00000080
379 core 0: 0x8000117c (0x00300313) li t1, 3
380 core 0: 3 0x8000117c (0x00300313) x6 0x00000003
381 core 0: 0x80001180 (0x00b31313) slli t1, t1, 11
382 core 0: 3 0x80001180 (0x00b31313) x6 0x00001800
383 core 0: 0x80001184 (0xffff34313) not t1, t1
384 core 0: 3 0x80001184 (0xffff34313) x6 0xffffe7ff
385 core 0: 0x80001188 (0x0062f2b3) and t0, t0, t1
386 core 0: 3 0x80001188 (0x0062f2b3) x5 0x00000080
387 core 0: 0x8000118c (0x00100313) li t1, 1
388 core 0: 3 0x8000118c (0x00100313) x6 0x00000001
389 core 0: 0x80001190 (0x00b31313) slli t1, t1, 11
390 core 0: 3 0x80001190 (0x00b31313) x6 0x00000800
391 core 0: 0x80001194 (0x0062e2b3) or t0, t0, t1
392 core 0: 3 0x80001194 (0x0062e2b3) x5 0x00000880
393 core 0: 0x80001198 (0x30029073) csrw mstatus, t0
394 core 0: 3 0x80001198 (0x30029073) c768_mstatus 0x00000880
395 core 0: 0x8000119c (0x00412083) lw ra, 4(sp)
396 core 0: 3 0x8000119c (0x00412083) j 0x8000119c 0x00000001

```

Now the program is in **S\_mode** another ECALL will be trapped and returned in **M\_mode**, finally the program can exit in **M-mode**.

Finally program exits by writing to the host

```

231 core 0: >>> M_MODE
232 core 0: 0x80001128 (0x1700006f) j pc + 0x170
233 core 0: 3 0x80001128 (0x1700006f)
234 core 0: 0x80001298 (0x00100193) li gp, 1
235 core 0: 3 0x80001298 (0x00100193) x3 0x00000001
236 core 0: 0x8000129c (0xffffffff17) auipc t5, 0xffff
237 core 0: 3 0x8000129c (0xffffffff17) x30 0x8000029c
238 core 0: 0x800012a0 (0xd63f2223) sw gp, -668(t5)
239 core 0: 3 0x800012a0 (0xd63f2223) mem 0x80000000 0x00000001
240 core 0: 0x800012a4 (0xff5ff06f) j pc - 0xc
241 core 0: 3 0x800012a4 (0xff5ff06f)

```

### **How does the test start in M-mode?**

The hart starts in Machine mode after reset, and the first executed instruction at `_start` runs in M-mode.

### **How is mode switching verified?**

By observing ECALL causes in the Machine-mode trap handler and confirming correct control flow in Spike logs.

### **How does the test exit?**

The test always returns to Machine mode and writes to the `tohost` register, which terminates Spike execution.

## **Reference to RISC-V Spec**

[1] Privilege Modes: Privileged Architecture v1.12, Sec 3.1

[2] Trap Handling: Sec 3.2-3.3 (mcause, mepc, mtvec, mret)

[3]ECALL Causes: Sec 4.3.1 (8 = U-mode, 9 = S-mode)