Task 3

Write an assembly test to handle exceptions in S mode

a. Assembly test

- Start your test in M mode and then switch to U mode
- Implement a trap handler for S mode to handle exceptions
- Generate an illegal instruction exception and handle this exception in S mode (all the exceptions are by default handled in M mode)

➢ Solution:

Github Link:

https://github.com/BilalAli10x/Riscv_Arch_Test/tree/main/BilalAli_Task3

Test Description

The objective of this task is to test **exception handling in S-mode** while leveraging the **mode-switch function implemented in Task 1.**

Key Steps:

1. Start execution in Machine mode **(M-mode)**.
2. Set up Machine trap vector **(mtvec)** to point to a Machine-mode trap handler.
3. Set up Supervisor trap vector **(stvec)** to point to a Supervisor-mode trap handler.
4. Delegate illegal instruction exceptions to **S-mode (medeleg)**.
5. Use Task 1 function **switch_mode** with argument a0 = 1 to switch **M -> U** mode.
6. Generate an **illegal instruction** in U-mode to trigger a Supervisor-mode trap.
7. Verify **trap_handler_s** correctly handles the illegal instruction and resumes execution in U-mode.
8. Trigger ECALL from U-mode -> handled in Machine mode -> returns to Supervisor mode.
9. Trigger ECALL from S-mode -> handled in Machine mode -> returns to Machine mode.
10. Test exits in Machine mode, signaling success via tohost.

Implementation:

- **Mode Switching (Task 1 function):**

```
 li a0, 1        # 1 = USER
 call switch_mode # Task 1 function: only callable from
M-mode
```

    - Argument 1 ensures **M -> U** transition.

    - **switch_mode** handles **MPP** bits and uses **mret** to switch privilege.
- **Supervisor Trap Handling:**
    - **Illegal instruction** in U-mode triggers S-mode trap **via medeleg**.
    - **trap_handler_s** checks **scause = 2** (Illegal Instruction) and skips instruction using **sepc**.
    - Returns to U-mode via **sret.**
- **Machine Trap Handling:**
    - Handles ECALLs:
        - mcause = 8 -> USER_ECALL -> return to S-mode
        - mcause = 9 -> SUPERVISOR_ECALL -> return to M-mode
    - Advances **mepc** and sets **mstatus.MPP** accordingly.

Test Flow

1. M-mode: Start of execution. Set up trap vectors and delegation.
2. M → U: Call Task 1 function switch_mode with a0 = 1.
3. U-mode: Execute illegal instruction → traps to S-mode.
4. S-mode: Handles illegal instruction → resumes U-mode.
5. U-mode: ECALL → handled by M-mode → returns to S-mode.
6. S-mode: ECALL → handled by M-mode → returns to M-mode.
7. Exit and write 1 to tohost.

Expected Output

- Illegal instruction successfully handled in S-mode.
- ECALL transitions:
    - U -> M -> S
    - S -> M
- Test exits in Machine mode, signaling pass **via tohost.**

Spike Log Snapshot Example:

During S-mode trap handling:

```
core   0: >>>>  trap_handler_s
core   0: 0x800000c0 (0x142022f3) csrr    t0, scause
core   0: 1 0x800000c0 (0x142022f3) x5  0x00000002
```

Final Pass Output:

```
core   0: 0x80000174 (0x30200073) mret
core   0: 3 0x80000174 (0x30200073) c768_mstatus 0x000000a0 c784_mstatush
0x00000000
```

```
core   0: 0x80000040 (0x1380006f) j       pc + 0x138
core   0: 3 0x80000040 (0x1380006f)
core   0: 0x80000178 (0x00100193) li      gp, 1
core   0: 3 0x80000178 (0x00100193) x3  0x00000001
core   0: 0x8000017c (0x00001297) auipc   t0, 0x1
core   0: 3 0x8000017c (0x00001297) x5  0x8000117c
core   0: 0x80000180 (0xe832a223) sw      gp, -380(t0)
core   0: 3 0x80000180 (0xe832a223) mem 0x80001000 0x00000001
```

Reference to Specification

- RISC-V Privileged Architecture Specification
  - **Section 3.1.6:** mstatus, MPP
  - **Section 4.1.1:** sstatus, SPP
  - **Section 3.1.15:** mcause (ECALL 8 & 9)
- Trap return instructions: **mret, sret**