Task 4

Write an assembly test to check mstatus.FS and mstatus.SD behavior

a. **Assembly test**
   - Write a test to check the mstatus.FS filed (OFF, CLEAN and DIRTY) (How can you check mstatus.SD field using mstatus.FS? Figure out yourself)

Github Link:

https://github.com/BilalAli10x/Riscv_Arch_Test/tree/main/BilalAli_Task4

➢ Test Description

This test is verifying correct behavior of:

- mstatus.FS field (Floating-Point Status)
- mstatus.SD field (State Dirty bit)

The test is:

- Forcing FS = OFF
- Forcing FS = CLEAN
- Forcing FS = DIRTY
- Checking SD behavior using FS transitions
- Performing self-checking using conditional branches
- Reporting PASS/FAIL through tohost

This test is validating correct architectural behavior of floating-point state tracking.

➢ Understanding the Fields

mstatus.FS (Bits [14:13])

Value	Meaning
00	OFF
01	CLEAN
10	Reserved
11	DIRTY

mstatus.SD (Bit [31])

SD = 1 If any of the following fields are DIRTY:

- FS or XS or VS

Otherwise: SD = 0

In this test, only FS is used.

So SD depends entirely on FS.

Implementation

1. Setting FS = OFF

Clearing bits [14:13]:

FS = 00

Verifying:

- Extracting FS
- Comparing with 0
- Extracting SD (bit 31)
- Expecting SD = 0

If not matching -> FAIL.

2. Setting FS = CLEAN

Writing:

FS = 01

Verifying:

- FS == 01
- SD == 0

CLEAN state does not set SD.

If mismatch -> FAIL.

3. Setting FS = DIRTY

Writing to floating-point register:

fmv.w.x ft0, t0

This is:

- Modifying FP state
- Hardware automatically updating FS → DIRTY (11)

Verifying:

- FS == 11
- SD == 1

Because:

If FS == DIRTY → SD must be 1

If not -> FAIL.

Self-Checking Mechanism

Using:

- bne -> for FS mismatch
- bnez -> for SD mismatch

On any failure:

gp = 2 -> FAIL

On success:

gp = 1 -> PASS

Writing to:

tohost

Expected Output (Spike)

Case 1: FS=OFF

- FS = 00
- SD = 0

Case 2: FS=CLEAN

- FS = 01
- SD = 0

Case 3: After FP write

- FS = 11
- SD = 1

Final result:

gp = 1

Spike Log Snapshot Example:

```
335   core   0: >>>>  trap_handler
336   core   0: 0x80000140 (0x342022f3) csrr    t0, mcause
337   core   0: 3 0x80000140 (0x342022f3) x5  0x00000001
338   core   0: 0x80000144 (0x00100313) li      t1, 1
339   core   0: 3 0x80000144 (0x00100313) x6  0x00000001
340   core   0: 0x80000148 (0x02628463) beq     t0, t1, pc + 40
341   core   0: 3 0x80000148 (0x02628463)
342   core   0: 0x80000170 (0x34109073) csrw    mepc, ra
343   core   0: 3 0x80000170 (0x34109073) c833_mepc 0x800000c0
344   core   0: 0x80000174 (0x30200073) mret
345   core   0: 3 0x80000174 (0x30200073) c768_mstatus 0x00000080 c784_mstatush 0x00000000
346   core   0: 0x800000c0 (0x1280006f) j       pc + 0x128
347   core   0: 3 0x800000c0 (0x1280006f)
348   core   0: 0x800001e8 (0x00100193) li      gp, 1
349   core   0: 3 0x800001e8 (0x00100193) x3  0x00000001
```

Reference to Specification

RISC-V Privileged Architecture Specification

- **Section 3.1.6.6** - mstatus, Extension Context Status
- **Section 3.1.6.6** - mstatus, SD **(Page =29)**