RV DV: RISC-V Arch Test

RISC-V Arch Test – Task: 4

Following is the link to the github repository for this task:

https://github.com/daniyalahmed-10xe/RISC-V_Arch_Test-Task_4.git

Test Description:

In this task the ‘FS’ bits of the ‘mstatus’ csr had to be modified to observer the value of ‘SD’ bit of ‘mstatus’ csr. This was done by using a ‘for-loop-like’ structure and assigning every possible combination to ‘FS’ and then reading the ‘SD’ bit. This bit was then masked onto an output and bit shifted left so each bit that is set represents a dirty floating point. Note that since the provided linker file does not allocate any space for the stack, the ‘PROLOGUE’ and ‘EPILOGUE’ can not be used here and are commented out.

Whats’s the Actual Output? (Screenshots Attatched at the End of File):

According to the logfile (screenshot provided at the end) we get an output of ‘9’ or ‘1001’ which means that for ‘FS’ with the values ‘00’ and ‘11’, the bit ‘SD’ was set and for ‘FS’ with values ‘01’ and ‘10’, gthe bit ‘SD’ was unset.

Following are the answers to the questions asked in this task:

‘SD’ bit becomes set if any of the ‘FD’ or ‘XD’ bit fields have a value that represents a dirty state. So by checking ‘FD’ we can predict the value of ‘SD’; If ‘FD’ is in a dirty state then ‘SD’ would be set otherwise unset.

Following is the code written for this task:

- **<u>task4.S</u>**

```
#define RVTEST_DATA_BEGIN                                          \
        .pushsection .tohost,"aw",@progbits;                       \
        .align 6; .global tohost; tohost: .dword 0; .size tohost, 8;         \
        .align 6; .global fromhost; fromhost: .dword 0; .size fromhost, 8;   \
        .popsection;                                               \
        .align 4; .global begin_signature; begin_signature:
#define RVTEST_CODE_BEGIN                                          \
        .section .text.init;                                       \
        .align  6;                                                 \
        .global _start;                                            \
_start:                                                            \
    j main;                                                        \
            j writeToHost
RVTEST_CODE_BEGIN

main:
        # PROLOGUE
                # addi sp, sp, -16
                # sw ra, 0(sp)
                # sw gp, 4(sp)
                # sw tp, 8(sp)
                # sw fp, 12(sp)
        # END PROLOGUE

        # CODE

                la t0, trapVector
                csrw mtvec, t0

                la t0, trapVectorSupervisor
                csrw stvec, t0

                # li t0, 0x4
                # csrw medeleg, t0

                call testFloating
                mv t0, a0

                ecall
                ecall

        # END CODE

        # EPILOGUE
                # lw fp, 12(sp)
                # lw tp, 8(sp)
                # lw gp, 4(sp)
                # lw ra, 0(sp)
                # addi sp, sp, 16
        # END EPILOGUE

end_main: call writeToHost

writeToHost:

        # PROLOGUE
                # addi sp, sp, -16
                # sw ra, 0(sp)
                # sw gp, 4(sp)
                # sw tp, 8(sp)
                # sw fp, 12(sp)
        # END PROLOGUE

        # CODE

                li gp, 1
                sw gp, tohost, t5
        # END CODE

        # EPILOGUE
                # lw fp, 12(sp)
                # lw tp, 8(sp)
                # lw gp, 4(sp)
                # lw ra, 0(sp)
                # addi sp, sp, 16
        # END EPILOGUE

end_writeToHost: call writeToHost

testFloating:

        # PROLOGUE
                # addi sp, sp, -16
                # sw ra, 0(sp)
                # sw gp, 4(sp)
                # sw tp, 8(sp)
                # sw fp, 12(sp)
        # END PROLOGUE

        # CODE

                li t0, 0
                li t1, 4
                li a0, 0

                for1: bge t0, t1, end_for1
                        sll a0, a0, t0
                        csrr t2, mstatus
                        slli t3, t0, 13
                        or t2, t2, t3
                        csrw mstatus, t2
                        csrr t2, mstatus
                        li t3, 0x80000000
                        and t3, t2, t3
                        srli t3, t3, 31
                        or a0, a0, t3

                        addi t0, t0, 1
                        j for1
                end_for1:

        # END CODE

        # EPILOGUE
                # lw fp, 12(sp)
                # lw tp, 8(sp)
                # lw gp, 4(sp)
                # lw ra, 0(sp)
                # addi sp, sp, 16
        # END EPILOGUE

end_testFloating: ret

switchMode:

        # PROLOGUE
                # addi sp, sp, -16
                # sw ra, 0(sp)
                # sw gp, 4(sp)
                # sw tp, 8(sp)
                # sw fp, 12(sp)
        # END PROLOGUE

        # CODE

                mv t0, a0
                csrr t1, mstatus
                li t6, 0x1800
                not t6, t6
                and t1, t1, t6

                if1: bnez t0, else1
                        li t6, 0x0800
                        j end_if1
                else1:
                        li t6, 0x0000
                        j end_if1
                end_if1:

                or t1, t1, t6
                csrw mstatus, t1

        # END CODE

        # EPILOGUE
                # lw fp, 12(sp)
                # lw tp, 8(sp)
                # lw gp, 4(sp)
                # lw ra, 0(sp)
                # addi sp, sp, 16
                csrw mepc, ra
        # END EPILOGUE

end_switchMode: mret

trapVector:

        # PROLOGUE
                # addi sp, sp, -64
                # sw gp, 4(sp)
                # sw tp, 8(sp)
                # sw fp, 12(sp)
                # sw s0, 16(sp)
                # sw s1, 20(sp)
                # sw s2, 24(sp)
                # sw s3, 28(sp)
                # sw s4, 32(sp)
                # sw s5, 36(sp)
        # END PROLOGUE

        # CODE

                csrr s0, mcause
                li s1, 9
                li s2, 8
                li s3, 5
                li s4, 7
                li s5, 1

                if2: bne s0, s1, else2if3
                        li s1, 0x1800
                        j end_if23456
                else2if3: bne s0, s2, else3if4
                        li s1, 0x0800
                        j end_if23456
                else3if4: bne s0, s3, else4if5
                        li s1, 0
                        j end_if23456
                else4if5: bne s0, s4, else5if6
                        li s1, 0
                        j end_if23456
                else5if6: bne s0, s5, end_if23456
                        li s1, 0
                        j end_if23456
                end_if23456:

                csrr s0, mstatus
                or s0, s0, s1
                csrw mstatus, s0

        # END CODE

        # EPILOGUE
                # lw s5, 36(sp)
                # lw s4, 32(sp)
                # lw s3, 28(sp)
                # lw s2, 24(sp)
                # lw s1, 20(sp)
                # lw s0, 16(sp)
                # lw fp, 12(sp)
                # lw tp, 8(sp)
                # lw gp, 4(sp)
                # lw ra, 0(sp)
```

```
                # addi sp, sp, 64
                addi ra, ra, 4
                csrw mepc, ra
        # END EPILOGUE

end_trapVector: mret

trapVectorSupervisor:

        # PROLOGUE
                # addi sp, sp, -32
                # sw gp, 4(sp)
                # sw tp, 8(sp)
                # sw fp, 12(sp)
                # sw s0, 16(sp)
                # sw s1, 20(sp)
                # sw s2, 24(sp)
        # END PROLOGUE

        # CODE

                csrr s0, scause
                li s1, 1
                li s2, 2

                if7: bne s0, s1, else7if8
                        li s1, 0
                        j end_if78
                else7if8: bne s0, s2, end_if78
                        li s1, 0
                        j end_if78
                end_if78:

                csrr s0, sstatus
                or s0, s0, s1
                csrw sstatus, s0

        # END CODE

        # EPILOGUE
                # lw s2, 24(sp)
                # lw s1, 20(sp)
                # lw s0, 16(sp)
                # lw fp, 12(sp)
                # lw tp, 8(sp)
                # lw gp, 4(sp)
                # lw ra, 0(sp)
                # addi sp, sp, 32
                addi ra, ra, 4
                csrw sepc, ra
        # END EPILOGUE
```

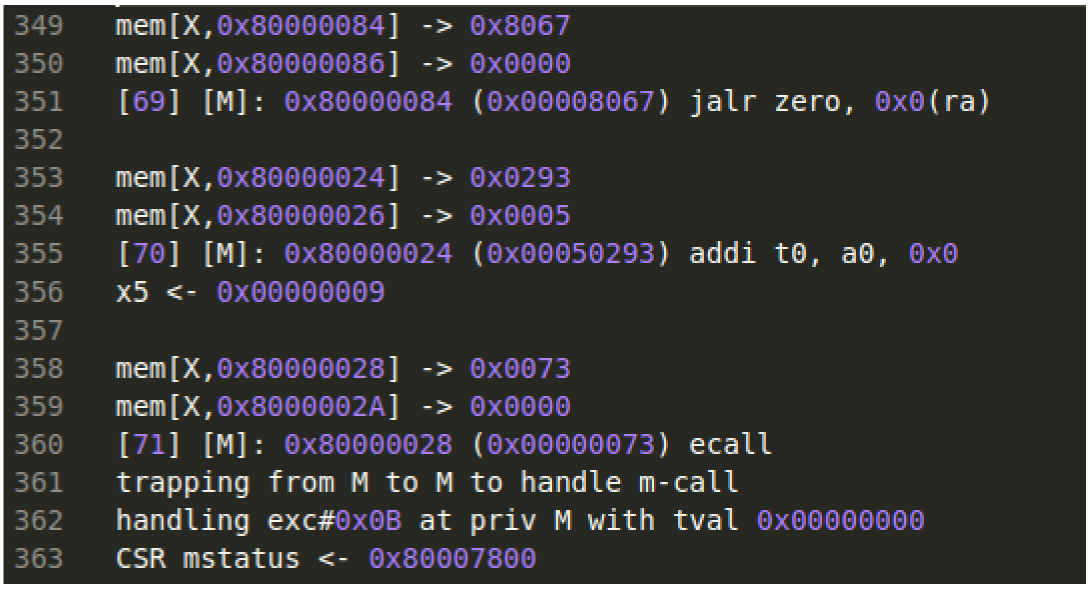
```
end_trapVectorSupervisor: sret

.data
base:
.word 0xcafebeef
RVTEST_DATA_BEGIN
```

- **<u>link.ld</u>**

```
OUTPUT_ARCH( "riscv" )
ENTRY(_start)
SECTIONS
{
  . = 0x80000000;
  .text.init : { *(.text.init) }
  . = ALIGN(0x1000);
  .tohost : { *(.tohost) }
  . = ALIGN(0x1000);
  .text : { *(.text) }
  . = ALIGN(0x1000);
  .data : { *(.data) }
  .bss : { *(.bss) }
  _end = .;
}
```

Following are the screenshots of the output for this task:



```
mem[X,0x80000084] -> 0x8067
mem[X,0x80000086] -> 0x0000
[69] [M]: 0x80000084 (0x00008067) jalr zero, 0x0(ra)

mem[X,0x80000024] -> 0x0293
mem[X,0x80000026] -> 0x0005
[70] [M]: 0x80000024 (0x00050293) addi t0, a0, 0x0
x5 <- 0x00000009

mem[X,0x80000028] -> 0x0073
mem[X,0x8000002A] -> 0x0000
[71] [M]: 0x80000028 (0x00000073) ecall
trapping from M to M to handle m-call
handling exc#0x0B at priv M with tval 0x00000000
CSR mstatus <- 0x80007800
```