RV DV: RISC-V Arch Test

RISC-V Arch Test – Task: 3

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

https://github.com/daniyalahmed-10xe/RISC-V_Arch_Test-Task_3.git

Test Description:

This task required us to switch to user mode from machine mode and trap an illegal instruction exception to s-mode from u-mode. This was done by creating a trap handler ‘trapVectorSupervisor’ for ‘stvec’ and setting the ‘medeleg’ bit 2 in the csr to let the m-mode know that illegal instruction interrupts (bit 2 of ‘medeleg’) should be handled by s-mode instead of m-mode. 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) when the illegal instruction exception in called the program traps to the trap handler pointed to by ‘stvec’ and succesfully traps to s-mode from u-mode by calling the ‘trapVectorSupervisor’ trap handler. This then just moves to the next instruction.

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

N/A

Following is the code written for this task:

- **task3.S**

```
#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

		li a0, 1
		call switchMode

		.word 0x00000000

		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

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
```

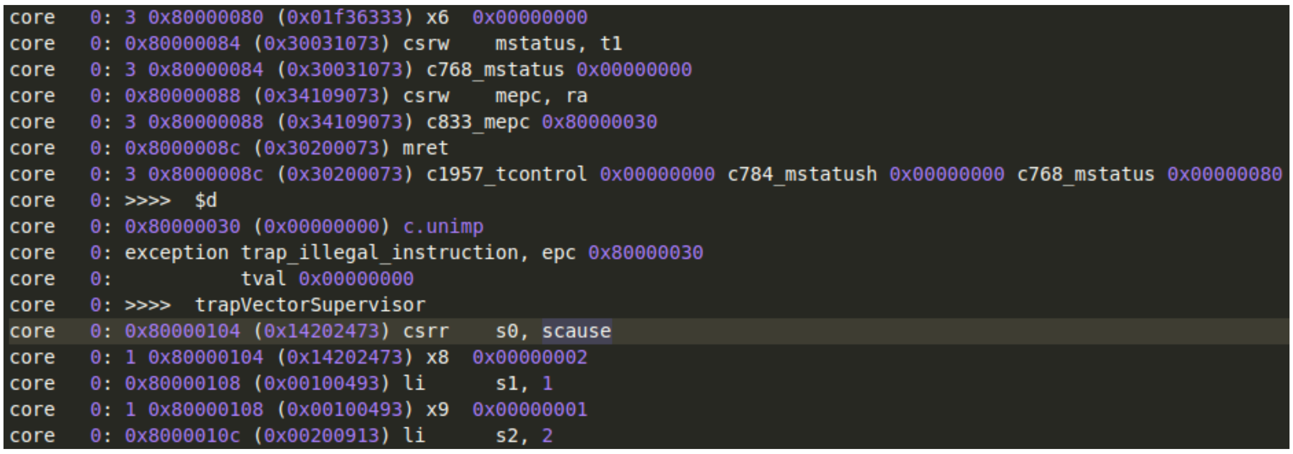
```
end_switchMode: mret

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:



```
core   0: 3 0x80000080 (0x01f36333) x6  0x00000000
core   0: 0x80000084 (0x30031073) csrw    mstatus, t1
core   0: 3 0x80000084 (0x30031073) c768_mstatus 0x00000000
core   0: 0x80000088 (0x34109073) csrw    mepc, ra
core   0: 3 0x80000088 (0x34109073) c833_mepc 0x80000030
core   0: 0x8000008c (0x30200073) mret
core   0: 3 0x8000008c (0x30200073) c1957_tcontrol 0x00000000 c784_mstatush 0x00000000 c768_mstatus 0x00000080
core   0: >>>>  $d
core   0: 0x80000030 (0x00000000) c.unimp
core   0: exception trap_illegal_instruction, epc 0x80000030
core   0:           tval 0x00000000
core   0: >>>>  trapVectorSupervisor
core   0: 0x80000104 (0x14202473) csrr    s0, scause
core   0: 1 0x80000104 (0x14202473) x8  0x00000002
core   0: 0x80000108 (0x00100493) li      s1, 1
core   0: 1 0x80000108 (0x00100493) x9  0x00000001
core   0: 0x8000010c (0x00200913) li      s2, 2
```