**RV DV: RISC-V Arch Test**

**RISC-V Arch Test – Task: 7**

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

<https://github.com/daniyalahmed-10xe/RISC-V_Arch_Test-Task_7.git>

**Test Description:**

In this task ‘Machine Mode Locking’ was to be used. By setting the ‘MML’ bit of the ‘mseccfg’ field of the csr, the behaviour of ‘pmpcfgX’ can be changed while ‘MML’ is active. By setting the initial permissions in ‘pmpcfgX’ as read and execute, activating machine mode locking changed this to a locked shared region with execute and read permissions for machine-mode and access faults for supervisor and user modes. 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) the program succesfully executes and reads data from the region with machine mode lockdown. When the mode is changed to supervisor mode, the effect of machine mode lockdown kicks in and a instruction access fault can be seen since only machine mode has execute and read privilages in this region.

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

Setting up this region would give a load access exception since this is a shared region, this has to be solved by setting the ‘RLB’ bit that allows a locked region’s permissions to be modified without a hardware reset. This feature has to be used to dynamically modify the permissions of the region to prevent any unnecesarry exceptions.

**Following is the code written for this task:**

* **task7.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 t0, 0b10001101

csrw pmpcfg0, t0

li t0, 0x80001000

srli t0, t0, 2

csrw pmpaddr0, t0

li t0, 0b001

csrw mseccfg, t0

li t0, 0x80000000

lw t0, 0(t0)

li a0, 0

call switchMode

nop

li t0, 0x80000000

lw t0, 0(t0)

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

* **link.ld**

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:**



