RV DV: RISC-V Arch Test

RISC-V Arch Test – Task: 5

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

https://github.com/daniyalahmed-10xe/RISC-V_Arch_Test-Task_5.git

Test Description:

This task sets up a two tiered page table and creates page table enteries for the given physical and virtual addresses for the given permissions. 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) a page fault can be observed when attempting to load data from a location without read permissions.

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

N/A

Following is the code written for this task:

- **task5.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, -32
                    # sw ra, 0(sp)
                    # sw gp, 4(sp)
                    # sw tp, 8(sp)
```

```
        # sw fp, 12(sp)
        # sw s1, 16(sp)
# END PROLOGUE

# CODE

        la t0, trapVector
        csrw mtvec, t0

        la t0, trapVectorSupervisor
        csrw stvec, t0

        # li t0, 0x4
        # csrw medeleg, t0

        call setupPageTable

        li a0, 0x00401000           # Virtual Address
        la a1, PhysicalMemory       # Physical Address
        li a2, 0b111                # Permissions (XWR)
        li a3, 0b1                      # Level
        call constructPTE

        mv t0, a0

        li a0, 0x00802000           # Virtual Address
        la a1, PhysicalMemory
        addi a1, a1, 64             # Physical Address
        li a2, 0b101                # Permissions (X-R)
        li a3, 0b1                      # Level
        call constructPTE

        mv t1, a0

        li a0, 0x02003000           # Virtual Address
        la a1, PhysicalMemory
        addi a1, a1, 512            # Physical Address
        li a2, 0b110                # Permissions (XW-)
        li a3, 0b1                      # Level
        call constructPTE

        mv t2, a0

        li a0, 0
        call switchMode

        lw t3, 0(t1)
        sw t3, 0(t2)
        sw t3, 0(t0)

        ecall
        ecall
```

```
        # END CODE

        # EPILOGUE
                # lw s1, 16(sp)
                # lw fp, 12(sp)
                # lw tp, 8(sp)
                # lw gp, 4(sp)
                # lw ra, 0(sp)
                # addi sp, sp, 32
        # 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

setupPageTable:

        # 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, L1PageTable
                la t1, L2PageTables
                li t2, 128
                li t3, 0

                for1: bge t3, t2, end_for1
                        sw t1, 0(t0)
                        addi t0, t0, 4
                        addi t1, t1, 512

                        addi t3, t3, 1
                        j for1
                end_for1:

                li t0, 1
                slli t0, t0, 31
                la t1, L1PageTable
                srli t1, t1, 12
                or t0, t0, t1
                csrw satp, t0

        # 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_setupPageTable: ret

constructPTE:

        # 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)
                # sw s3, 28(sp)
        # END PROLOGUE

        # CODE

                mv s0, a0
                mv s1, a1
                mv s2, a2
```

```
                mv s3, a3

                la t0, L1PageTable          # Root Address of L1 Page Table

                srli t1, s0, 12
                andi t1, t1, 0x3FF          # Get Address of L1 Page Table from VA
                srli t2, s0, 22             # Get Address of L2 Page Table from VA

                li t6, 0xFFF
                and t3, s1, t6              # Get Offset from Physical Address
                or s0, s0, t3               # VA has same Offset as PA
                slli s2, s2, 1
                or s0, s0, s2               # Append Permissions to VA

                li t4, 4
                mul t1, t1, t4
                mul t2, t2, t4
                add t0, t0, t1              # Get Address of L1 PTE
                lw t0, 0(t0)                # Get Address of L2 Page Table from L1 Page
Table
                add t0, t0, t2              # Get Address of L2 PTE
                sw s1, 0(t0)                # Store Physical Address in L2 Page Table

                mv a0, s0                       # Return VA

        # END CODE

        # EPILOGUE
                # 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, 32
        # END EPILOGUE

end_constructPTE: 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
.bss
    PhysicalMemory: .space 4096
    L1PageTable: .space 128
    L2PageTables: .space 16384
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: