Task: Hypervisor Extension - Two-Stage Translation and Guest Memory Access

Learning Objectives:
1. Privilege hierarchy: M > HS > VS > VU
2. Two-stage address translation (vsatp + hgatp)
3. HLV/HSV instructions for guest memory access
4. Trap delegation (medeleg, hedeleg)
5. Key CSRs: vsstatus, hstatus, vsatp, hgatp

Task Description:

Implement a hypervisor managing a guest OS in VS-mode with VU-mode application.

Implementation Steps:

1. M-mode initialization:
   * Setup mtvec
   * Delegate exceptions via medeleg (bits 8, 10, 12, 13, 15)
   * Initialize hgatp for G-stage translation
   * Jump to HS-mode using mret

2. HS-mode hypervisor:
   * Setup stvec
   * Configure hstatus (enable virtualization)
   * Setup vsatp for guest first-stage translation
   * Jump to VS-mode (set hstatus.SPV=1, sstatus.SPP=1, use sret)

3. VS-mode guest OS:
   * Store test value 0xDEADBEEF at known address
   * Jump to VU-mode (set sstatus.SPP=0, use sret)

4. VU-mode application:
   * Pass VS memory address in a0
   * Execute ecall (traps to HS-mode)

5. HS-mode trap handler:
   * Check scause (should be 8 for VU-mode ecall)
   * Set hstatus.SPVP=0
   * Use HLV.WU to read from VS memory address in a0
   * Modify value (add 0x100)
   * Use HSV.W to write back
   * Set sepc to VS verification code
   * Set sstatus.SPP=1, keep hstatus.SPV=1
   * Execute sret to VS-mode

6. VS-mode verification:
   * Use LWU to load modified value (avoid sign extension)
   * Verify: 0xDEADBEEF + 0x100 = 0xDEADBFEF
   * Execute ecall to HS-mode

7. HS-mode exit:
   * Check scause (should be 10 for VS-mode ecall)
   * Execute ecall to M-mode

8. M-mode exit:
   * Check mcause (should be 9 for HS-mode ecall)
   * Skip ecall (mepc += 4)
   * Write success to tohost

Submit a PDF with Screenshots that shows:

- Confirm M→HS→VS→VU mode transitions worked
- Verify medeleg includes both bit 8 (VU ecall) and bit 10 (VS ecall)
- Confirm HS trap handler checks scause=10 for VS ecalls (NOT 9)
- Verify VU-mode uses command codes in a1 to distinguish different ecall purposes
- Verify VS-mode stored data correctly (0xDEADBEEF)
- Confirm HLV.WU (not HLV.W) read correct value from VS memory without sign extension
- Verify HSV.W wrote modified value (0xDEADBFEF) back to VS memory
- Confirm VS-mode uses LWU (not LW) to load and verify modified value
- Verify HLVX.WU successfully fetched instruction from VS code region
- Confirm two-stage translation occurred
- Verify proper mode transitions: VU→HS→VU→HS→VS→HS→M
- Confirm test exited successfully in M-mode with tohost=1

Also submit Zip file of your code, linker, makefile, spike logs.