RISC-V 64I-Zba Processor Coding Challenge

Objective

Design and implement a RISC-V 64-bit processor core that implements the **RV64I** base integer instruction set with the **Zba** (address generation) extension.

Core Requirements

1. **Microarchitecture Requirements:**
    - 3 or 5 stage pipelined implementation (your choice)
    - Separate instruction and data memories
2. **Interface Requirements:**
    - Clock and reset signals
    - Instruction memory interface (read-only)
    - Data memory interface (read/write)
    - No need for peripheral interfaces or interrupts

Deliverables (Submit as a ZIP file):

1. **Design Documentation**
    - **Design Diagram** in *pdf* format using drawIO **OR** clear, legible hand-drawn diagram (scanned PDF)
    - Should show:
        - Datapath components (ALU, registers, PC, etc.)
        - Control signals and their sources
        - Instruction flow through the pipeline
        - Memory interfaces
        - Zba-specific logic implementation
2. **RTL Implementation**
    - SystemVerilog files with clear module hierarchy
    - Code must be synthesizable and well-commented

3. **Test Program**
    - A C program (test.c) that demonstrates your processor's capabilities
    - Must include:
        - Basic arithmetic operations
        - Memory access (load/store)
        - Branching logic
        - **At least 3 different Zba instruction usages** with comments explaining their purpose
4. **Toolchain Commands**
    - A text file (build_commands.txt) showing:
        - Commands to compile C code to RISC-V assembly
        - Commands to assemble to object file
        - Commands to link and create ELF
        - Commands to extract machine code for simulation
5. **Testbench**
    - A comprehensive SystemVerilog testbench (tb_processor.sv)
    - Must include:
        - Clock and reset generation
        - Instruction memory initialization with your test program
        - Data memory monitoring
        - Register value checking after execution
        - Self-checking assertions for key instructions
        - Simulation termination condition

Note: This challenge tests your understanding of RISC-V architecture, digital design principles, and hardware/software co-verification. We're looking for clean, working implementations rather than overly optimized designs.