Through this project, I gained a deeper understanding of RISC-V architecture and simulator design. Here are my key takeaways:

RISC-V ISA Differences: I learned how 32-bit (RV32I) and 64-bit (RV64I) architectures differ in handling registers, memory addresses, and instructions. For example, adapting sign-extension logic for immediates and ensuring 32-bit address masking were critical to avoid overflow errors.
Pipeline Mechanics: By modifying the simulator's pipeline stages (fetch, decode, execute, etc.), I saw firsthand how data hazards and control hazards impact performance. Forwarding data between stages became clearer through debugging stalls and dependencies.
Toolchain Integration:I practiced compiling RISC-V programs using riscv32-unknown-elf-gcc and testing them on the modified simulator. This highlighted the importance of aligning tool chain flags (e.g., -march=rv32i) with the target architecture.
Cross-Platform Debugging: Encountering issues like CRLF line endings in scripts taught me to use tools like dos2unix and always validate file formats when working across OS environments.
Code Maintainability: Balancing minimal code changes with correctness was challenging. For instance, preserving the LRU replacement logic in the cache while adapting addresses to 32-bit showed how small design choices affect scalability.

This project reinforced my low-level programming skills and emphasized the value of systematic testing—each small error in register handling or memory alignment had cascading effects, requiring careful isolation and validation.