Project design plan

Enhancing Cryptographic Application Performance on RISC-V Processors with B and K Extensions.

Student name: Harsh Mayank Dabhi

Student ID : A00010544

Supervisor: Dr Xiaojun Wang

Course: MEng in Electronic and Computer Engineering

Project Design Plan: Performance Evaluation of RISC-V B and K Extensions for AES and SHA Acceleration (Software-Based Simulation)

1. Research Question

Enhancing Cryptographic Application Performance on RISC-V Processors with B and K Extensions.

2. Project Scope

This project will include the following:

- **Technologies:**
  - RISC-V ISA Github repo
  - RISC-V Bit-Manipulation ('B') extensions
  - RISC-V Cryptography ('K') extensions
  - QEMU simulator for RISC-V
  - RISC-V GCC compiler toolchain
  - AES (Advanced Encryption Standard) algorithm (e.g., AES-128, AES-256)
  - SHA (Secure Hash Algorithm) algorithms (e.g., SHA-256, SHA-512)
  - Linux environment (either native installation or via Docker for environment setup)
  - GitHub for code versioning and maintenance.
  - Data analysis and visualization tools (e.g., Python with Pandas/Matplotlib)
- **Research Areas:**
  - Performance analysis of cryptographic algorithms on RISC-V through software simulation.
  - Impact of ISA extensions on execution time and throughput, as simulated for software targeting these extensions.
  - Simulating workload methodologies for performance evaluation.
- **The project will aim to evaluate (through simulation):**
  - Execution time/cycle counts for AES and SHA software operations compiled with and without flags/intrinsics for B/K extensions.
  - Throughput of AES and SHA software operations.
  - Code size and potentially stack usage implications of different compilation targets.
- **The following areas/technologies/considerations will likely NOT be addressed in depth:**
  - Hardware synthesis or hardware implementation, physical area cost (beyond referencing literature findings e.g.,), or physical power consumption analysis. All evaluations are software-based simulations.

- In-depth analysis of every subset of B and K extensions; focus will be on those most relevant to AES/SHA and supported by the chosen compiler and simulation environment for software implementation.
- Extensive exploration of different cryptographic modes of operation beyond common examples.
- Formal verification of the cryptographic implementations.

3. Design Approach

The project will address the research question through the following design approach, emphasizing software solutions and simulation:

1. **Environment Setup:**
    - Revisit and enrich the existing literature survey.
    - Set up the development environment:
        - Install a Linux distribution or configure a Docker container.
        - Install the RISC-V GCC toolchain from official sources.
        - Install QEMU with RISC-V support.
        - Initialize a GitHub repository for version control**.**
        - Install necessary dependencies for both the compiler and QEMU, including libraries for AES/SHA if using standard implementations as a baseline.
2. **Tool Familiarization & Baseline Establishment:**
    - Thoroughly understand the documentation for Docker , QEMU simulator, RISC-V GCC compiler (focusing on flags related to ISA extensions like B and K), and GitHub.
    - Compile and simulate simple test cases for RISC-V on QEMU to ensure the toolchain and simulator are working correctly. Commit initial setup to GitHub.
3. **Implementation & Simulation (AES & SHA - Software Focus):**
    - **Software-only Baseline:**
        - Obtain or develop standard C/C++ implementations of AES and SHA algorithms.
        - Compile these implementations using the RISC-V GCC toolchain without any specific B or K extension flags (targeting a baseline RISC-V core like RV32/RV64).

- **Software Implementation Targeting B and K Extensions:**
    - Understand how to instruct the RISC-V compiler to generate code that would take advantage of B and K extension instructions if run on compatible hardware.
    - Modify or develop versions of the AES and SHA software implementations to be compiled in a way that targets the B and/or K extensions. The literature survey indicates that K extensions provide instructions for core AES/SHA operations.
    - Compile these "extension-aware" software versions.
    - Simulate their execution on QEMU, configured to model a RISC-V core that supports the relevant B and K extensions (ensuring QEMU can correctly interpret and model the performance of code compiled for these extensions).

4. **Testing & Validation:**
    - Develop small test cases for AES encryption/decryption and SHA hashing with known input/output vectors to verify the functional correctness of all software implementations (baseline and those compiled targeting extensions).
5. **Performance Measurement & Data Collection:**
    - For each software implementation (baseline, compiled for K-extensions, and compiled for B extensions):
        - Run simulations with varying workloads.
        - Capture key performance parameters: execution time (or cycle counts), throughput.
        - Capture code size of the compiled binaries and observe memory footprint during simulation.
    - Organize collected data into CSV/Other files. Commit data and analysis scripts to GitHub.
6. **Analysis of Results:**
    - Compare the simulated performance metrics of the software compiled to target extensions against the baseline software-only compilation. The literature notes significant performance gains with K extensions.
    - Analyze the simulated impact of B extensions and K extensions, and their combined effect, on the software's performance.
    - Discuss trade-offs based on simulation results.
7. **Data Visualization & Thesis Generation:**
    - Utilize data visualization tools to create plots illustrating performance comparisons.
    - Draft the thesis/report, incorporating the literature survey, methodology, results from simulation, analysis, and conclusions.

4. Detailed Timeline

Phase	Weeks	Activity	Tasks	Milestone
Preliminary phase	Week 1-3 (March 3 - 23)	Understanding goal and context	Defining goals and constraints	
	Week 3-4 (March 24-30)	Selecting /Installing relevant IDE for implementation and OS (linux/ubuntu)	Set up the development environment	Successfully setup IDE and OS
Phase 1: Foundations	Week 5-8 (March 31 -April 27)	Tool Familiarization & Environment Setup	- Research & decide on Linux/Docker setup. Install. - Install RISC-V GCC toolchain & QEMU. - Set up a GitHub repository. - Deep dive into Docker, QEMU, RISC-V Compiler documentation. - Understand GitHub for version control and data maintenance.	Working RISC-V dev & sim environment. Proficient with Docker, QEMU, RISC-V Compiler basics, and GitHub.

			- Initial "hello world" test for toolchain & QEMU, commit to GitHub.	
Phase 2: Baseline	Week 8-11 (April 21 to May 18)	Baseline Software Implementation & Testing	- Obtain/Implement baseline C versions of AES & SHA. - Develop initial test cases. - Compile & run baseline AES/SHA on QEMU. - Establish initial performance measurement techniques. Commit to GitHub.	Baseline AES & SHA software working in QEMU with basic performance data.
Phase 3: AES Extensions	Week 11 - 14 (May 19 to June 8)	AES Software Implementation Targeting Extensions	- Research compiler flags/intrinsics for B extensions for software. - Modify/Develop AES software to be compiled targeting B-extensions. - Compile & test. Debug. Commit.	Functionally correct AES software compiled for extensions.

Phase 4: SHA Extensions	Week 14 - 17 (June 9 to June 29)	SHA Software Implementation Targeting Extensions	- Modify/Develop SHA software to be compiled targeting K-extensions - If feasible, a version targeting K-extensions. - Compile & test. Debug. Commit.	Functionally correct SHA software compiled for extensions.
Phase 5: Data Capturing	Week 17 - 19 (June 30 - July 13)	Comprehensive Performance Simulation & Data Gathering	- Design workload scenarios. - Automate simulation runs for all software versions. - Systematically collect performance data (execution time/cycles, throughput), code sizes. - Organize data into CSVs. Commit.	Comprehensive raw performance data from simulations collected and organized.
Phase 6: Analysis	Week 19 - 20 (July 14 - July 20)	Data Analysis & Visualization	- Use Pandas/Matplotlib to process CSV data. - Generate plots. - Perform analysis & interpret results.	Performance analysis complete, key visualizations created.

			Commit scripts & plots.	
Phase 7: Reporting	Week 21 - 23 (July 21 to Aug 10)	Thesis/Report Writing	- Structure & write report sections (Intro, Lit Review, Methodology, Results, Analysis, Conclusion). - Incorporate data, visualizations. - Refine.	First full draft of the project report.
Phase 8: Finalize	Week 23 -24 (Aug 11 to Aug 18)	Review, Refinement, Contingency & Final Submission	- Review draft. Seek feedback. - Revisions. - Final proofreading & formatting. - Buffer for unexpected delays.	Final project report/portfolio submission ready.

5. Success Criteria

This project will be considered successful if the following criteria are met:

- A functional RISC-V development and simulation environment using Docker/Linux, QEMU, a RISC-V compiler, and GitHub is successfully established and utilized.

- Baseline (software-only) and "extension-aware" (software compiled to target K-extensions, and B-extensions) versions of AES and SHA algorithms are implemented and run correctly in the QEMU simulator.
- Performance metrics, primarily simulated execution time/cycle counts and throughput, are successfully collected for all implemented software versions across a defined set of workloads.
- The collected data is systematically analyzed, and clear comparisons are made to quantify the simulated performance impact of compiling software to target the B and K extensions.
- The results are presented clearly, using appropriate visualizations, in a final project report.
- The project is completed and documented according to the defined timeline, demonstrating the planned software-based design approach.