AetherRISC: General Specification & System Requirement Specification (SRS)

Project: AetherRISC

Version: 0.0.0 (Pre-Alpha)

Date: 2025-12-21

Document Status: Draft

Standard Compliance: IEEE 830-1998 (SRS), IEEE 1016-2009 (SDD)

1. Introduction

1.1 Purpose

The purpose of this document is to define the comprehensive functional and non-functional requirements for **AetherRISC**, a high-fidelity, deterministic RISC-V runtime environment and simulation platform. This specification serves as the primary reference for system architects, developers, and verification engineers.

AetherRISC is engineered to supersede legacy educational simulators—specifically Venus, RARS, MARS, and RARS Nova—by providing a professional-grade toolchain that bridges the gap between pedagogical visualization and industrial systems analysis.

1.2 Scope

AetherRISC functions as a **Hybrid (InteractiveAuto)** application, operating fundamentally as a client-side execution engine to ensure zero-latency simulation, while leveraging server-side capabilities for persistence and compilation services.

The system encompasses:

- **Core Simulation:** A cycle-accurate implementation of the RISC-V Unprivileged ISA (RV32I/RV64I) with support for standard extensions (M, A, F, D, C, V).
- **Embedded Systems Emulation:** A virtualized hardware layer supporting Memory-Mapped I/O (MMIO), GPIO, and protocol-level simulation (UART, I2C).
- **Runtime Diagnostics:** Deterministic state analysis tools for hazard detection, pipeline visualization, and fault root-cause analysis.
- **Polyglot Development:** Integrated support for high-level language compilation (C, Rust, Zig) to ELF binaries alongside traditional Assembly development.

1.3 Definitions, Acronyms, and Abbreviations

- **DCI (Dynamic Code Injection):** The capability to insert machine code into a running process via memory trampolining.
- **DRA (Direct Register Access):** The mechanism permitting out-of-band modification of register states.
- **HDE (Heuristic Diagnostic Engine):** A deterministic subsystem for analyzing processor state and explaining stalls or exceptions.
- **ISA (Instruction Set Architecture):** The abstract model of the computer.
- **MMIO (Memory-Mapped I/O):** A method of performing I/O between the CPU and peripheral devices in which they share the same address space.
- **RIS (Runtime Intervention Subsystem):** The supervisory module enabling DCI, DRA, and state checkpointing.
- **WASM (WebAssembly):** A binary instruction format for a stack-based virtual machine, allowing client-side execution in web browsers.

2. Legacy Supersedence Strategy

To ensure a frictionless migration path for academic institutions and individual users, AetherRISC guarantees **100% Functional Parity** with the following legacy tools:

2.1 Venus Parity

- **Syscall Compatibility:** Full support for the Venus ecall convention (IDs 1 through 57), including file system operations (open, read, write, close), heap management (sbrk), and process termination.
- **File Format Support:** Native ingestion of Venus project files and assembly scripts.
- **Instruction Reference:** Implementation of an integrated, context-sensitive instruction reference pane identical in content to the Venus help tab.
- **Web-Based Workflow:** Replication of the client-side, local-execution model used by Venus web.

2.2 RARS / RARS Nova Parity

- **Unified Syscall Table:** Support for the superset of RARS system calls.
- **MMIO Device Interfaces:** Re-implementation of standard RARS peripherals including the Bitmap Display, Digital Lab Sim (7-Segment displays), and Keyboard/TTY interfaces.
- **Asset Management:** Capability to import and utilize .jar archive assets where applicable for legacy project compatibility.

2.3 MARS Compatibility

- **Transitional Support:** Backward compatibility layer for MIPS-to-RISC-V transitional curricula, offering syntax translation or emulation where semantically feasible.

3. System Features & Functional Requirements

3.1 Simulation Core

- **ISA Compliance:** The kernel shall implement the RISC-V User-Level ISA Specification version 2.2 or later.
- **Pipeline Simulation:**
  - The system shall simulate a configurable 5-stage pipeline (Fetch, Decode, Execute, Memory, Writeback).
  - The system shall visualize inter-stage latches and control signals.
- **Memory Model:**
  - The system shall support a configurable Flat Memory Model (1KB to 4GB).
  - The system shall support Big Endian and Little Endian byte ordering, configurable at runtime.

3.2 Runtime Intervention Subsystem (RIS)

The RIS provides supervisory controls enabling the operator to manipulate the machine state during execution for debugging, fault injection, and "what-if" analysis.

- **Direct Register Access (DRA):**
  - The system shall provide a UI mechanism to read and modify any register (GPR, FPR, CSR) at any point in the clock cycle.
  - Modifications via DRA shall trigger immediate re-evaluation of dependent pipeline stages.
- **Dynamic Code Injection (DCI):**
  - The system shall allow the insertion of arbitrary opcodes into the instruction stream at runtime.
  - **Mechanism:** The system shall utilize a memory trampolining technique, relocating the injection target to a high-memory patch zone, executing the injected code, executing the displaced instruction, and branching back to the next sequential address.
- **State Checkpointing (Time Travel):**
  - The system shall implement a Memento pattern to serialize the full Machine State (PC, Registers, Memory Delta) to a ring buffer.
  - The system shall allow "Rewind" operations to restore previous states deterministically.

3.3 Heuristic Diagnostic Engine (HDE)

The HDE replaces "AI" or probabilistic models with deterministic, rule-based static analysis to provide verifiable explanations for processor behavior.

- **Hazard Analysis:**
  - The engine shall algorithmically identify Data Hazards (RAW, WAR, WAW) by comparing source and destination registers across pipeline stages.
  - The engine shall identify Control Hazards resulting from branch instructions.
  - **Output:** The system shall display deterministic explanations (e.g., "Stall injected:

Instruction at ID depends on result of Instruction at EX (x5)").

- **Exception Root Cause Analysis (RCA):**
  - Upon exception (trap), the engine shall scan the execution history buffer.
  - The engine shall trace back to the instruction causing the invalid state transition (e.g., analyzing mcause and mtval to explain a Load Access Fault).

3.4 Embedded Systems Simulation

- **MMIO Controller:**
  - The system shall implement a dedicated bus logic for routing load/store operations to mapped peripherals based on a configurable mmio_map.json.
- **Virtual PCB Interface:**
  - The system shall render an interactive Printed Circuit Board using Scalable Vector Graphics (SVG).
  - **Binding:** The system shall bind specific memory addresses to visual attributes of the SVG (e.g., 0xFFFF0000 bit 0 controls the fill color of an LED element).
  - **Interaction:** User events on the PCB (clicks, toggles) shall write values to mapped input addresses.

3.5 Polyglot Toolchain & Decompilation

- **Compilation Service:**
  - The system shall provide a containerized service (Docker) to compile high-level languages (C, Rust, Zig) into RISC-V ELF binaries.
  - The system shall parse DWARF debug information to map source lines to assembly instructions.
- **Deterministic Decompiler (The "Rosetta" View):**
  - The system shall project raw machine code into "Pseudo-C" syntax in real-time.
  - **Method:** The decompiler shall utilize Basic Block analysis and pattern matching (e.g., addi xA, xA, 1 $\rightarrow$ xA++).
  - **Symbol Resolution:** The decompiler shall utilize the Symbol Table to replace raw register indices with user-defined aliases (e.g., Score++ instead of x10++).

3.6 Visualization Subsystems

- **Cache Visualization:**
  - The system shall visualize the state of L1 and L2 caches, including Set/Way organization, Tag storage, and Valid/Dirty bits.
  - The system shall animate cache hits, misses, and eviction policies (LRU/Random).
- **Branch Prediction Visualization:**
  - The system shall visualize the Branch History Table (BHT) and Branch Target Buffer (BTB).
  - The system shall indicate prediction states (Strongly Taken, Weakly Taken, etc.) and visualize misprediction penalties.

4. Specific Requirements - Interfaces

4.1 User Interfaces

- **General Layout:** The application shall adopt a standard IDE layout (Explorer Left, Editor Center, Diagnostics Right/Bottom).
- **Theming:** The UI shall support High Contrast, Dark, and Light themes compliant with WCAG 2.1 accessibility standards.
- **Responsiveness:** The layout shall adapt to varying viewport sizes, maintaining usability on tablet and desktop form factors.

4.2 Hardware Interfaces (Simulated)

- **GPIO:** General Purpose Input/Output controller with configurable pin count.
- **UART:** Universal Asynchronous Receiver-Transmitter for serial console emulation.
- **I2C/SPI:** Serial bus protocols for sensor emulation.

4.3 Software Interfaces

- **Plugin API:** The system shall expose an IIsaExtension interface allowing dynamic loading of .NET assemblies to define custom opcodes.
- **GDB Stub:** The system shall implement the GDB Remote Serial Protocol (RSP) to allow external debuggers (e.g., VS Code, GDB CLI) to attach to the simulation session via a TCP socket or WebSocket bridge.

5. Non-Functional Requirements

5.1 Performance

- **Clock Speed:** The client-side WebAssembly engine shall sustain a minimum effective clock speed of 100 KHz on standard reference hardware (equivalent to Intel Core i5-8250U or better).
- **Latency:** UI updates for register changes shall occur within 16ms (60 FPS) of the state change event.

5.2 Reliability & Determinism

- **Reproducibility:** Given identical initial state and input vector, the simulation shall produce bit-identical results across all supported platforms (Windows, Linux, macOS, Web).
- **Fault Isolation:** User code crashes (Exceptions) must be trapped by the simulator and must not crash the host application.

5.3 Extensibility

- **Modular Architecture:** The system design shall adhere to the Single Responsibility Principle. Core simulation logic must be decoupled from UI rendering logic.

- **Configuration:** System parameters (RAM size, Cache topology, Extension enablement) must be configurable via JSON files without recompilation.

6. Verification

- **Architectural Compliance:** The system shall pass 100% of the official RISC-V Architectural Test Suite (RISCOF) for the implemented extensions.
- **Legacy Validation:** The system shall successfully execute the standard "Golden" test suites provided by the Venus and RARS projects.