

# AetherRISC: System Architecture & Technology Stack

**Project:** AetherRISC

**Version:** 0.0.0 (Pre-Alpha)

**Date:** 2025-12-21

**Standard Compliance:** IEEE 1471 (Recommended Practice for Architectural Description of Software-Intensive Systems)

## 1. Architectural Pattern: "Interactive Hybrid"

The system implements a **Hybrid Client-Server Architecture** utilizing the **.NET Blazor "InteractiveAuto" Render Mode**. This architecture enables a distinct separation of concerns between the management plane (Server) and the execution plane (Client/Desktop).

### 1.1 Execution Contexts

1. **Management Plane (Server-Side Rendering / SSR):**
  - **Responsibility:** Authentication, Project Database Access, Docker Compilation Orchestration, Collaboration Hub (SignalR).
  - **Runtime:** ASP.NET Core on Linux/Windows Server.
  - **Benefit:** Instant First Contentful Paint (FCP) and secure access to backend infrastructure.
2. **Execution Plane (Client-Side WebAssembly / WASM):**
  - **Responsibility:** Cycle-Accurate Simulation, Real-time Visualization, Heuristic Analysis, Runtime Intervention.
  - **Runtime:** Mono Wasm Runtime (Browser) or .NET CLR (Desktop).
  - **Benefit:** Zero-latency execution loop (>100 KHz) running on the user's hardware, eliminating server compute costs for simulation.

## 2. Technology Stack Specification

### 2.1 Core Frameworks

- **Target Framework:** .NET 10 Preview (Language Version: C# 14).
- **Web Framework:** Blazor Web App (InteractiveAuto).
- **Desktop Framework:** .NET MAUI (Multi-platform App UI) wrapping Blazor via WebView2.

### 2.2 Critical Libraries

- **Simulation & Logic:**
  - System.Runtime.Intrinsics: utilized for SIMD acceleration of Vector (RV-V) operations.
  - ELFSharp: For parsing ELF binary containers and DWARF debug symbols.
- **Data & State:**
  - Google.Protobuf: High-performance binary serialization for state snapshots and client-server sync.
  - Microsoft.EntityFrameworkCore.Sqlite: Local database provider for the Server mode.
- **Presentation:**
  - Blazor.Extensions.Canvas: HTML5 Canvas Interop for high-frequency rendering (Pipeline/PCB).
  - Monaco Editor: The VS Code editor engine, wrapped for Blazor.

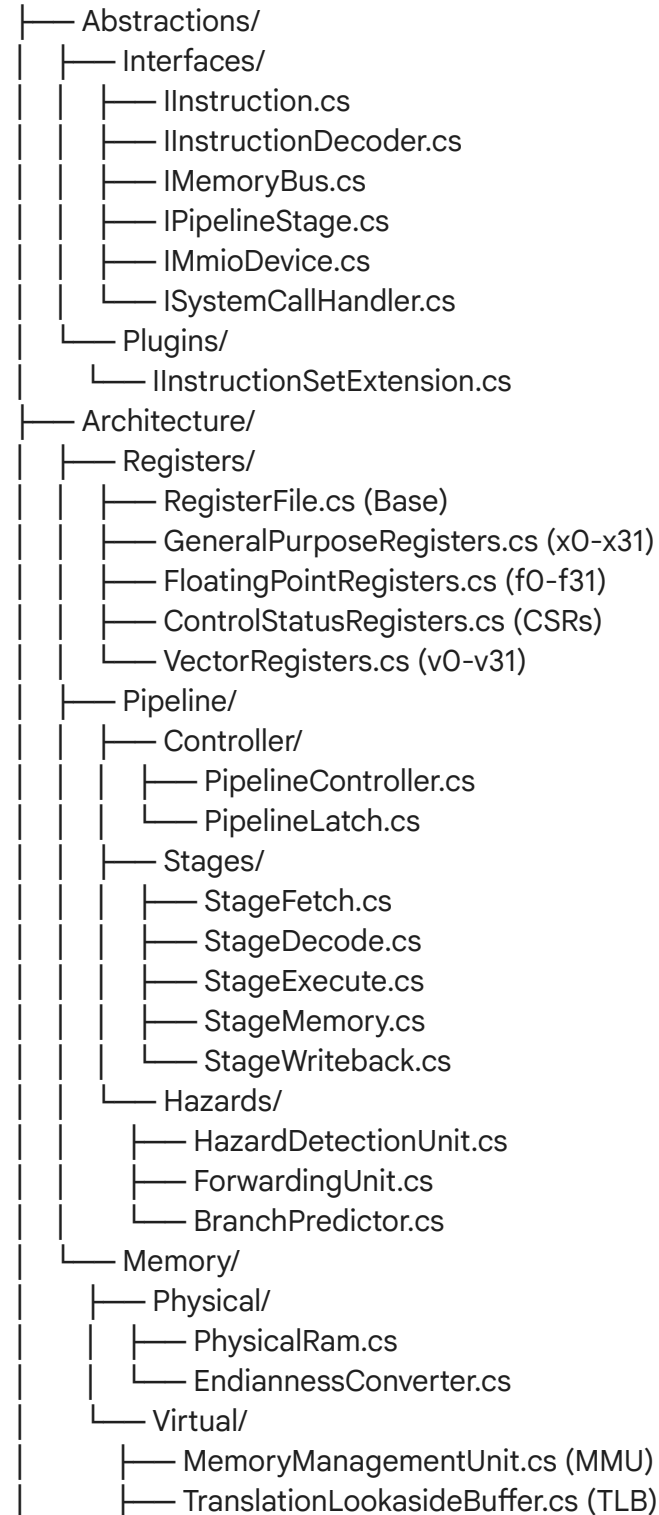
## 3. Solution Topology & File Structure

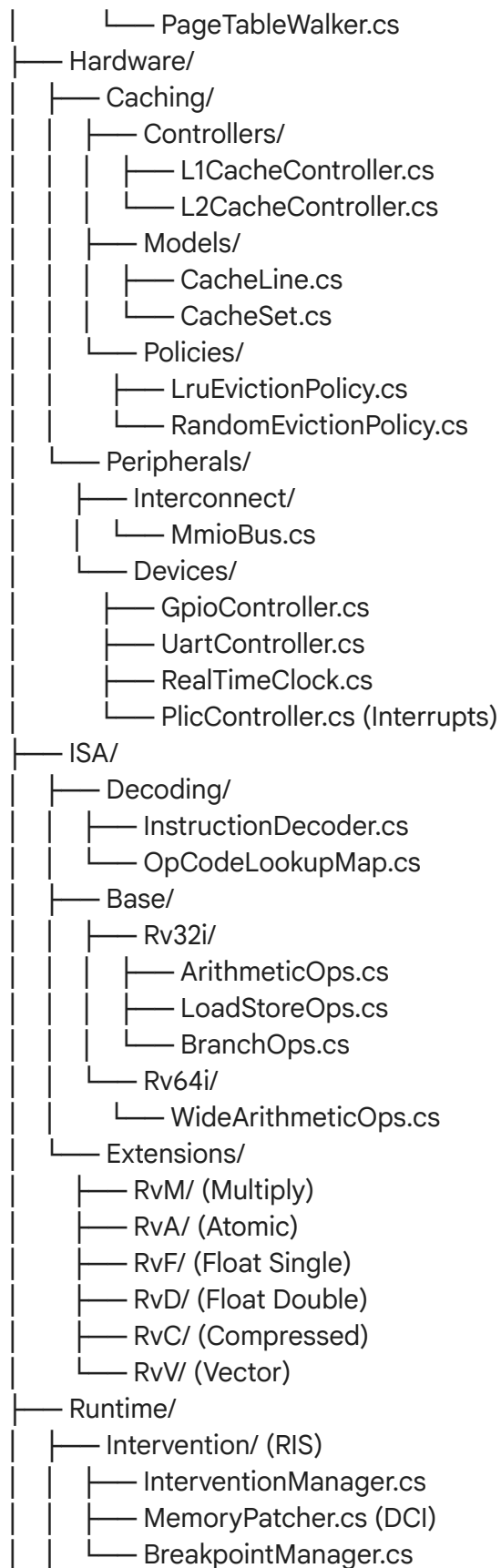
The codebase is strictly partitioned following the **Clean Architecture** (Onion Architecture) pattern. The Core logic has zero dependencies on UI or Infrastructure.

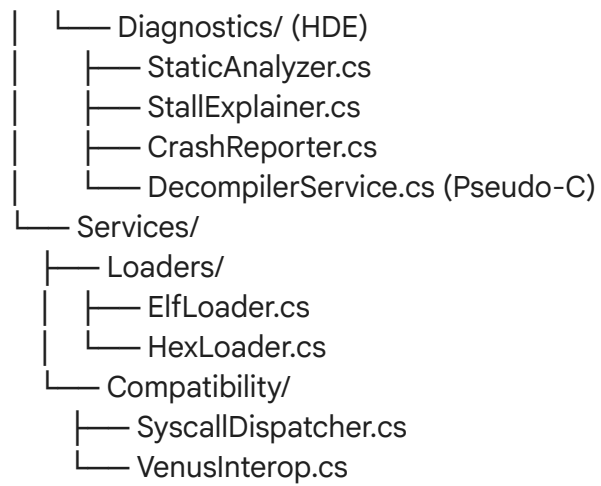
### 3.1 AetherRISC.Core (Domain Layer)

*The "Brain" - Pure .NET Standard Class Library.*

AetherRISC.Core/



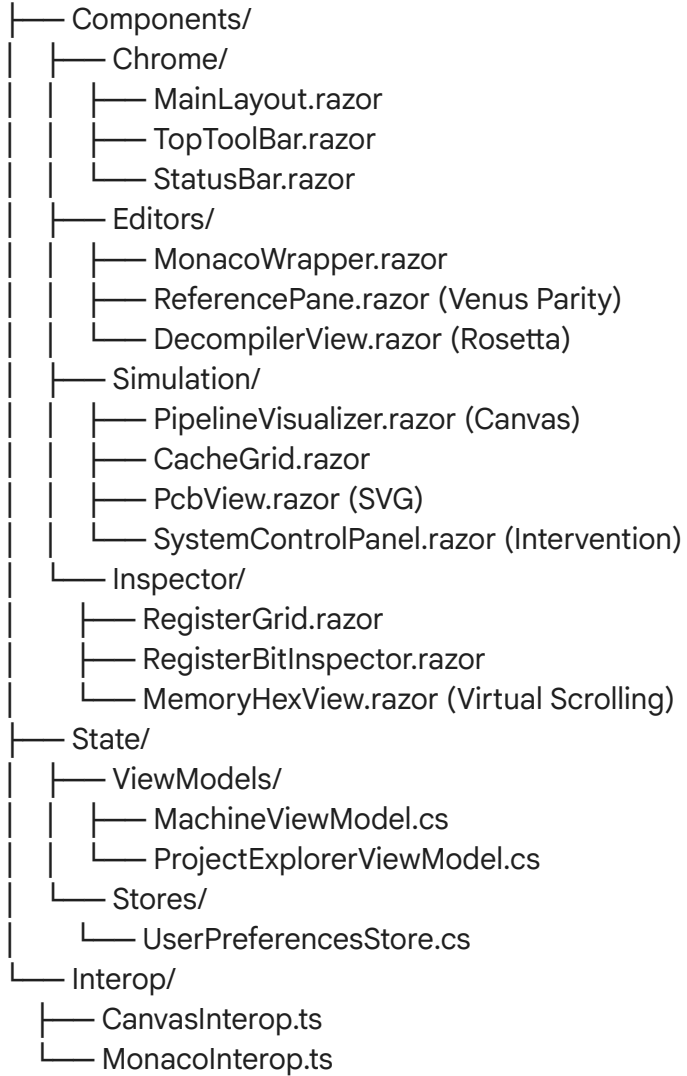




## 3.2 AetherRISC.UI (Presentation Layer)

*Razor Class Library (RCL) shared by Web and Desktop.*

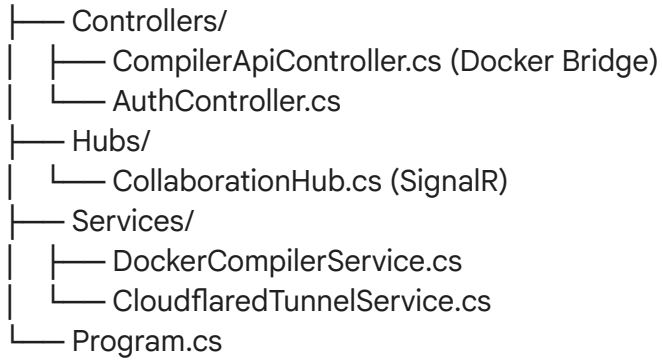
AetherRISC.UI/



### 3.3 AetherRISC.Web (Infrastructure Layer)

*ASP.NET Core Host.*

AetherRISC.Web/



### 3.4 AetherRISC.Desktop (Infrastructure Layer)

*.NET MAUI Host.*

AetherRISC.Desktop/



## 4. External Interfaces & Data Contracts

### 4.1 Client-Server Contract (Protobuf)

To ensure rapid state synchronization for the "Cloud Save" feature:

```
message MachineStateSnapshot {
  uint64 cycle_count = 1;
  uint32 pc = 2;
  repeated uint32 registers_gpr = 3 [packed=true];
  repeated uint64 registers_fpr = 4 [packed=true];
  // Delta compression: only send changed memory pages
  map<uint32, bytes> memory_pages = 5;
}
```

### 4.2 Extension Contract

To allow third-party instruction sets without recompiling the Core:

```
public interface IInstructionSetExtension
{
  string Name { get; }
  Version Version { get; }
  // Registers new opcodes into the decoder tree
  void RegisterInstructions(IInstructionRegistry registry);
  // Optional: Register custom CSRs
  void RegisterCsrs(ICsrRegistry registry);
}
```

## 5. Deployment & Containerization

### 5.1 Compiler Backend

The system utilizes a specialized Docker container for polyglot support.

- **Image:** aetherrisc/toolchain:latest
- **Contents:** riscv64-unknown-elf-gcc, rustc, zig.
- **Operation:** Ephemeral instantiation per compilation request with strict resource limits (CPU/RAM quotas) to prevent DoS.



## 5.2 Release Artifacts

- **AetherRISC.Desktop:** MSIX (Windows), Deb (Linux), App (macOS).
- **AetherRISC.Web:** Docker Image (aetherrisc/web-host).
- **AetherRISC.Core:** NuGet Package (for headless CI/CD usage).