Execution Comparison Results: C vs Zig Emulators

Date: 2026-01-12 20:00 **Status**: In Progress **Purpose**: Document execution comparison results between C and Zig emulators

Overview

After fixing the C emulator PC advancement bug and implementing GVAR BIGATOMS mode in Zig, both emulators now execute identically for the first 5 instructions. This document tracks the comparison results and identifies remaining issues.

Comparison Methodology

Log Generation

Both emulators generate execution logs with unified format:

- C: `c_emulator_execution_log.txt` (1000 instructions)
- Zig: `zig_emulator_execution_log.txt` (stops on crash/error)

Comparison Process

1. Generate logs from both emulators
2. Compare line by line for PC, instruction, stack, and frame values
3. Identify first divergence point
4. Analyze bit-shifted values to find root cause

Results Summary

First 5 Instructions: ✅ Perfect Match (All Traced and Verified)

Line	C PC	C Instruction	Zig PC	Zig Instruction	Status
1	0x60f130	POP	0x60f130	POP	✅ Traced
2	0x60f131	GVAR	0x60f131	GVAR	✅ Traced, Fixed (BIGATOMS)
3	0x60f136	UNBIND	0x60f136	UNBIND	✅ Traced, Fixed (offset)
4	0x60f137	GETBASEPTR_N	0x60f137	GETBASEPTR_N	✅ Traced, Fixed (byte order)
5	0x60f139	COPY	0x60f139	COPY	✅ Traced, Verified
6	0x60f13a	TJUMP1	0x60f13a	TJUMP1	✅ Traced, Fixed (offset 3)

Tracing Documents:

- `c-emulator-address-xor-tracing.typ` - GVAR XOR addressing
- `c-emulator-unbind-tracing.typ` - UNBIND stack unwinding
- `c-emulator-getbaseptr-tracing.typ` - GETBASEPTR_N memory access
- `c-emulator-copy-tracing.typ` - COPY stack duplication

GVAR PC Advancement: ✅ Fixed

- C emulator: Advances PC by 5 bytes (`0x60f131 → 0x60f136`)
- Zig emulator: Now advances PC by 5 bytes (matches C)
- Root cause: Zig was using 2-byte atom numbers instead of 4-byte pointers
- Fix: Updated to BIGATOMS+BIGVM mode (5-byte instruction length)

Remaining Issues

Zig Emulator Crash After 48 Instructions

Status: ⚠️ Under Investigation

The Zig emulator crashes after approximately 48 instructions with `error.InvalidAddress`. This occurs after successful execution of the first 5 instructions.

Possible Causes:
1. Invalid address calculation in atom lookup
2. Memory access beyond virtual memory bounds
3. Incorrect address translation for atom pointers
4. Stack corruption causing invalid pointer dereference

Next Steps:
1. Add detailed tracing around instruction 48
2. Compare stack and frame state at crash point
3. Verify atom pointer values and address translation
4. Check for memory bounds violations

Key Findings

C Emulator PC Advancement Bug

Issue: PC was not being updated from `pccache` after opcode execution **Fix**: Added `PC = PCMAC` update before resetting `pccache` **Impact**: Enabled proper execution and comparison

GVAR BIGATOMS Mode

Issue: Zig assumed 2-byte atom numbers, C uses 4-byte pointers **Fix**: Updated instruction length to 5 bytes and added `getPointerOperand()` **Impact**: First 5 instructions now match exactly

Unified Logging Format

Implementation: Both emulators use identical log format with hex+octal+bit-shifts **Benefit**: Enables precise comparison and off-by-one-bit error detection

Verification Status

✅ C emulator executes correctly (source of truth) ✅ First 5 instructions match between C and Zig ✅ GVAR opcode implementation matches C behavior ⚠️ Zig crashes after 48 instructions (investigation needed)

Next Steps

1. Investigate Zig emulator crash at instruction 48
2. Add enhanced tracing around crash point
3. Compare stack/frame state at divergence
4. Fix address calculation or memory access issue
5. Verify execution continues to 1000 instructions

Related Documentation

- C Emulator PC Advancement Fix - Bug fix details
- Zig GVAR BIGATOMS Implementation - Opcode implementation
- Unified Logging Format - Log format specification