All Your Assembly Belongs to Rust

Automated Lifting for Uniform Testing and Verification

Charly Castes, Gurvan Debaussart & Thomas Bourgeat.

Problem

Verifying programs with inline assembly is **hard!**

- Assembly is particularly error prone, yet ubiquitous in low-level and performance-oriented software
- Standard tools do **not** work: no Asan, no Miri, and no Kani
- Cross-compilation and privileged instructions make testing harder

Solution

Lift assembly into equivalent high-level code!

- **Automatically** and **seamlessly** lift assembly into the host language, such as Rust
- Leverage formal ISA specification to **emulate** assembly behavior
- Compatible with **existing** testing and verification tools

Rust ISA model based on Sail



Supported assembly features

- Arithmetic operations
- Local control flow
- Loads & Stores
- Functions calls
- System instructions
- Traps

Test and Verify Low-Level Software

Miralis

*Flash*point

- Data Center Systems Laboratory
- Verification and Computer Architecture Laboratory

Automatic assembly lifting

```
1  let mut result: u64 = 0;
2  let mut sign: u64 = 0;
3  soft_asm!(
4    "srai {sign},   {x}, 63",
5    "xor  {result}, {x}, {sign}",
6    "sub  {result}, {result}, {sign}",
7    x      = in(reg)    x as u64,
8    sign   = inout(reg) sign,
9    result = inout(reg) result,
10   softcore(crate::with_core)
11 );
```

1. *Allocate registers*
2. *Insert `core` state*
3. *Transform instructions into calls on `core`*
4. *Handle traps*

```
1  let mut result: u64 = 0;
2  let mut sign: u64 = 0;
3  {
4    core.set(reg::X5, x as u64);
5    core.set(reg::X6, sign);
6    core.set(reg::X7, result);
7    match core.execute(
8  ast::SHIFTIOP(reg::X5,63,reg::X6,sop::SRAI)
9    ) { ... };
10   match core.execute(
11 ast::RTYPE(reg::X5,reg::X6,reg::X7,rop::XOR)
12   ) { ... };
13   match core.execute(
14 ast::RTYPE(reg::X6,reg::X7,reg::X7,rop::SUB)
15   ) { ... };
16   sign   = core.get(reg::X6);
17   result = core.get(reg::X7);
18 };
19 // We can test and verify the above code
20 // with any standard tools!
```

Check it out!

PLOS'26 Paper

Source Code