Wanna introduce my recent work about a compiler for WebAssembly to the EVM bytecode

Really need opinions from the community since feedbacks are required to shape it

repo: GitHub - clearloop/zink: Rustic programming language that targets the Ethereum Virtual Machine

The Zink project mainly provides a single-pass compiler zinkc

which compiles WASM to EVM bytecode, the source code of your smart contract could be any language you like!

rust any-language -> WASM -> Zink Compiler -> EVM bytecode ...

For using rust

as the language of the smart contracts will unlock all of the following features:

- Safe

: rustc

is wathcing you! Furthermore, after compiling your rust code to WASM,

zinkc

will precompute all of the stack and memory usages in your contracts to ensure they are safe in EVM bytecode as well!

- High Performance

: The optimizations are provided by the three of rustc

, wasm-opt

and zinkc

, your contracts will have the smallest size with strong performance

in EVM bytecode at the end!

- Compatible

: All of the no_std

libraries in rust are your libraries, furthermore, you can use your solidity contracts as part of your zink contracts and your zink contracts as part of your solidity contracts

- Easy Debugging

: Developing your smart contracts with only one programming language! zink will provide everything you need for developing your contracts officially based on the stable projects in rust like the foundry

tools.

Fibonacci Example

fib(n)

Zink

Solidity@0.8.21

0

110

614

1

110

614

2

262

1322

3

414

2030

4

718

3446

5

1174

5570

/// Calculates the nth fibonacci number using recursion.

[no_mangle]

pub extern "C" fn recursion(n: usize) -> usize { if n < 2 { n } else { recursion(n - 1) + recursion(n - 2) } }

As an example for the benchmark, calculating fibonacci sequence with recursion, missed

vyper because it doesn’t support recursion…Zink is 4x faster on this for now, but it is mainly caused by the implementation is not completed yet (missing logic to adapt more situations)