

# WAVM Floating point implementation

Implementing correct, consistent, and deterministic floating point operations directly in WAVM (meaning both a Rust Arbitrator implementation and Solidity OSP implementation) would be an extremely tricky endeavor. WASM specifies floating point operations as being compliant to IEEE 754-2019, which is not deterministic, and full of edge cases.

Instead, floating point operations (apart from trivial bit-casts like `i32 <-> f32`) are implemented using the C Berkeley SoftFloat-3e library running inside WAVM. Arbitrator links other WAVM guests against this, by replacing float point operations with cross module calls to the library.

Berkeley SoftFloat does not implement all necessary floating point operations, however. Most importantly, it does not provide a `min` function, despite IEEE 754-2019 specifying one. The implementation of these operations, along with the export of convenient APIs for WASM opcode implementations, are contained in `bindings32.c` for 32 bit integers and `bindings64.c` for 64 bit integers.

This ensures that floating point operations are deterministic and consistent between Arbitrator and the OSP, as they are implemented exclusively using operations already known to be deterministic and consistent. However, it does not ensure that the floating point operations are perfectly compliant to the WASM specification. Go uses floating points in its JS<->Go WASM interface, and floating points may be used outside core state transition code for imprecise computations, but the former is well exercised as used in Nitro, and the latter generally doesn't rely on details like the minimum of NaN and infinity.

## Known divergences from the WASM specification

Floating point to integer truncation will saturate on overflow, instead of erroring. This is generally safer, because on x86, overflowing simply produces an undefined result. A WASM proposal exists to add new opcodes which are defined to saturate, but it's not widely adopted. [Edit this page](#) Last updated on Apr 29, 2024 [Previous WAVM Custom opcodes not in WASM](#) [Next WAVM Modules](#)