WAVM Modules

WASM natively has a notion of modules. Normally, in WASM, a module is the entire program. A.wasm file represents one module, and generally they aren't combined. An exception to this is C compiled via Clang, where wasm files are also used as object files, butits linking scheme is not supported in other languages.

In WAVM this is extended to make the executing program composed of multiple modules. These may call each other, and library modules may write to their caller's memory to return results.

The entrypoint module

The entrypoint module is where execution begins. It calls modules'start functions if specified, and then calls the main module's main function, which is language specific. For Go it sets argv to["js"] to match the JS environment, and callsrun . For Rust it callsmain with no arguments.

Library exports

Libraries may export functions with the name patternmodule__name , which future libraries or the main module can import as"module" "name" .

For instance, this is used for wasi-stub to provide functions rust imports according to the WebAssembly System Interface.

Floating point operations

To provide floating point operations for future libraries, the soft float library exports functions which perform floating point ops. These have the same name as the WASM instruction names, except. is replaced with_ . Their type signature is also the same, except allf32 s andf64 s are bitcasted toi32 s andi64 s.

Future modules can implicitly use these by using WASM floating point operations, which are replaced at the WASM->WAVM level with bitcasts and cross module calls to these functions.

WAVM guest calls

Libraries may call the main module's exports via"env" "wavm_guest_call__*" .

For instance, go-stub calls Go's resume function when queueing async events viawavm_guest_call_resume() , and then retrieves the new stack pointer withwavm_guest_call_getsp() .

Caller module internals call

Every stack frame retains its caller module and its caller module's "internals offset", which is the first internal function index. WAVM appends 4 "internal" functions to each module, which perform a memory load or store of 1 or 4 bytes.

Viawavm_caller_{load,store}{8,32} , a library may access its caller's memory, which is implemented by calling these internal functions of the caller's module. Only libraries can access their caller's memory; the main module cannot.

For instance, this is used to read arguments from and write return values to the Go stack, when Go calls into go-stubEdit this page Last updatedonMar 19, 2024 Previous WAVM Floating point implementation Next Arbitrum FAQ