WAVM Custom opcodes not in WASM

In addition to the MVP WASM specification, WAVM implements the multi value and sign extension ops WASM proposals.

WAVM also implements the following unique opcodes, which are not part of WASM nor any WASM proposal.

Invariants

Many of these opcodes have implicit invariants about what's on the stack, e.g. "Pops an i32 from the stack" assumes that the top of the stack has an i32. If these conditions are not satisfied, execution is generally not possible. These invariants are maintained by WASM validation and Arbitrator codegen. (SeeOne Step Proof Assumptions .)

Codegen internal

These are generated when breaking down a WASM instruction that does many things into many WAVM instructions which each do one thing. For instance, a WASMlocal.tee is implemented in WAVM withdup and thenlocal.set , the former of which doesn't exist in WASM.

Other times, these opcodes help out an existing WASM opcode by splitting out functionality. For instance, the WAVMreturn opcode by itself does not clean up the stack, but its WASM->WAVM codegen includes a loop that utilizesIsStackBoundary to perform the stack cleanup specified for WASM'sreturn .

Opcode Name Description 0x8000 EndBlock Pops an item from the block stack. 0x8001 EndBlockIf Peeks the top value on the stack, assumed an i32. If non-zero, pops an item from the block stack. 0x8002 InitFrame Pops a caller module index i32, then a caller module internals offset i32, and finally a return InternalRef from the stack. Creates a stack frame with the popped info and the locals merkle root in proving argument data. 0x8003 ArbitraryJumpIf Pops an i32 from the stack. If non-zero, jumps to the program counter in the argument data. 0x8004 PushStackBoundary Pushes a stack boundary to the stack. 0x8005 MoveFromStackToInternal Pops an item from the stack and pushes it to the internal stack. 0x8006 MoveFromInternalToStack Pops an item from the internal stack and pushes it to the stack. 0x8007 IsStackBoundary Pops an item from the stack. If a stack boundary, pushes an i32 with value 1. Otherwise, pushes an i32 with value 0. 0x8008 Dup Peeks an item from the stack and pushes another copy of that item to the stack. The above opcodes eliminate the need for the following WASM opcodes (which are transpiled into other WAVM opcodes):

- loop
- if/else
- br_table
- local.tee

Linking

This is only generated to link modules together. Each import is replaced with a local function consisting primarily of this opcode, which handles the actual work needed to change modules.

Opcode Name Description 0x8009 CrossModuleCall Pushes the current program counter, module number, and module's internals offset to the stack. Then splits its argument data into the lower 32 bits being a function index, and the upper 32 bits being a module index, and jumps to the beginning of that function.

Host calls

These are only used in the implementation of "host calls". Each of these has an equivalent host call method, which can be invoked from libraries. The exception isCallerModuleInternalCall , which is used for the implementation of all of thewavm_caller_* host calls. Those calls are documented inwavm-modules.md .

For these instruction descriptions, all pointers and offsets are represented as WASM i32s.

Opcode Name Description 0x800A CallerModuleInternalCall Pushes the current program counter, module number, and module's internals offset (all i32s) to the stack. Then, it retrieves the caller module internals offset from the current stack frame. If 0, errors, otherwise, jumps to the caller module at function (internals offset + opcode argument data) and instruction 0. 0x8010 GetGlobalStateBytes32 Pops a pointer and then an index from the stack. If the index is greater than or equal to the number of global state bytes32s, errors. If the pointer mod 32 is not zero, errors. If the pointer + 32 is outside the programs memory, errors. Otherwise, writes the global state bytes32 value of the specified index to the specified pointer in memory. 0x8011 SetGlobalStateBytes32 Pops a pointer and then an index from the stack. If the index is greater than or equal to the number of global state bytes32s, errors. If the pointer mod 32 is not zero, errors. If the pointer + 32 is outside the programs memory, errors. Otherwise, reads a bytes32 from the specified pointer in memory and sets the global state bytes32 value of the specified index to it. 0x8012 GetGlobalStateU64 Pops a pointer and then an index from the stack. If the index is greater than or equal to the number of global state u64s, errors. If the pointer mod 32 is not zero, errors. If the

pointer + 8 is outside the programs memory, errors. Otherwise, writes the global state u32 value of the specified index to the specified pointer in memory. 0x8013 SetGlobalStateU64 Pops a pointer and then an index from the stack. If the index is greater than or equal to the number of global state u64s, errors. If the pointer mod 32 is not zero, errors. If the pointer + 8 is outside the programs memory, errors. Otherwise, reads a u64 from the specified pointer in memory and sets the global state u64 value of the specified index to it. 0x8020 ReadPreImage Pops an offset and then a pointer from the stack. If the pointer mod 32 is not zero, errors. If the pointer + 32 is outside the programs memory, errors. Reads a 32 byte Keccak-256 hash from the specified pointer in memory. Writes up to 32 bytes of the preimage to that hash, beginning with theoffset byte of the preimage. Ifoffset is greater than or equal to the number of bytes in the preimage, writes nothing. Pushes the number of bytes written to the stack as an i32. 0x8021 ReadInboxMessage Pops an offset, then a pointer, and then an i64 message number from the stack. If the pointer mod 32 is not zero, errors. If the pointer + 32 is outside the programs memory, errors. Attempts to read an inbox message from the inbox identifier contained in the argument data (0 for the sequencer inbox, 1 for the delayed inbox) at the specified message number. If this exceeds the machine's inbox limit, enters the "too far" state. Otherwise, writes up to 32 bytes of the specified inbox message, beginning with theoffset byte of the message. Ifoffset is greater than or equal to the number of bytes in the preimage, writes nothing. Pushes the number of bytes written to the stack as an i32. 0x8022 HaltAndSetFinished Sets the machine status to finished, halting execution and marking it as a success. Edit this page Last updatedonApr 29, 2024Previous WASM to WAVMNext WAVM Floating point implementation