WASM to WAVM

Not all WASM instructions are 1:1 with WAVM opcodes. This document lists those which are not, and explains how they're expressed in WAVM. Many of the WAVM representations use opcodes not in WASM, which are documented inwavm-custom-opcodes.md .

block

andloop

In WASM, a block contains instructions. Branch instructions exit a fixed number of blocks, jumping to their destination. A normalblock 's destination is the end of the block, whereas aloop 's destination is the start of the loop.

In WAVM, instructions are flat. At transpilation time, any branch instructions are replaced with jumps to the corresponding block's destination. This means that WAVM interpreters don't need to track blocks, and thus block instructions are unnecessary.

if

andelse

These are translated to a block with anArbitraryJumpIf as follows:

begin block with endpoint end conditional jump to else [instructions inside if statement] branch else: [instructions inside else statement] end

br

andbr_if

br andbr_if are translated intoArbitraryJump andArbitraryJumpIf respectively. The jump locations can be known at transpilation time, making blocks obsolete.

br_table

br_table is translated to a check for each possible branch in the table, and then if none of the checks hit, a branch of the default level.

Each of the non-default branches has a conditional jump to a section afterwards, containing adrop for the selector, and then a jump to the target branch.

local.tee

local.tee is translated to a WAVMDup and then aLocalSet .

return

To translate a return, the number of return values must be known from the function signature. A WAVMMoveFromStackToInternal is added for each return value. Then, a loop checksIsStackBoundary (which implicitly pops a value) until it's true and the stack boundary has been popped. Next, aMoveFromInternalToStack is added for each return value to put the return values back on the stack. Finally, a WAVMReturn is added, returning control flow to the caller.

Floating point instructions

A floating point library module must be present to translate floating point instructions. They are translated by bitcastingf32 andf64 arguments toi32 s andi64 s, then a cross module call to the floating point library, and finally bitcasts of any return values fromi32 s andi64 s tof32 s andf64 s. Edit this page Last updatedonApr 29, 2024 Previous One Step Proof Assumptions Next WAVM Custom opcodes not in WASM