Compiling to WebAssembly

FOSDEM 2021
Andy Wingo | wingo@igalia.com
wingolog.org | @andywingo
agenda

A hands-on intro
A tiny Scheme compiler
Missing pieces
Pandora’s box of unanswerables
supporting materials:
https://github.com/wingo/compiling-to-webassembly
Sample program: recursive fac

(define (fac n)
  (if (zero? n)
      1
      (* n (fac (- n 1))))
)

diving  right  in
https://webassembly.github.io/spec/core
type, func, table, memory, global, import, export, start, elem, data
(module
  (type ...) ...
  (func ...) ...
  (export ...) ...)

Assemble using `wat2wasm`: https://github.com/WebAssembly/wabt
Test in browser
simple
compiler
filling in the gaps

Closures: https://github.com/WebAssembly/function-references; or closure conversion

Tail calls: https://github.com/WebAssembly/tail-calls, but lagging implementations; sadness

Varargs: Shadow arg stack??

Threads: https://github.com/WebAssembly/threads; web workers

Dynamic linking: Sadness
filling in the gaps

Exceptions / non-local control flow: https://github.com/WebAssembly/exception-handling

Coroutines: Possible extension of exception-handling; or CPS conversion

GC: It’s complicated
One approach: i32 as value type, tagging, linear memory GC

- Bad js interop
- No access to stack roots
- Web browsers already have excellent GC
- Uncollectable cycles with JS
gc, take two

Reference types: everything is an externref

- Have to call run-time (JS) to allocate, access data
- Not so bad???

Take three: GC-managed records and arrays in WebAssembly? https://github.com/WebAssembly/gc

Related to decision on closures (typed function references)
Call JS; be called by JS; how do data types cross the boundaries?
What do you do for strings? Linear memory, or JS strings?
DOM access – expose WebGL to your language?
further questions

Compile your source programs, or ship an interpreter?

How much JS to use? (Should I just target JS?)

Use LLVM? Binaryen? Emscripten?

Fork or extend host language?
questions

https://wingolog.org/

@andywingo

wingo@igalia.com

Happy hacking!