Wasm Exception-Handling Update

Heejin and Ben

2024-06-05

Recap - Phase 3 and exnref

- In October, Wasm CG voted to adopt exnref into EH proposal
  - An alternative to the Phase 3 “lexical rethrow” semantics
  - Reasons: more compiler functionality, easier engine implementation, simpler formal spec, asyncify transformations, JS interop

Recap - New instructions

1. `try_table (catch|catch_ref (<tag> <label>))* | (catch_all|catch_all_ref <label>)* instrs* end`
   Introduces a new block scope where each `catch*` branches to the corresponding label with either or both of (extracted values, `exnref`) if an exception is caught
2. `throw_ref` takes an explicit `exnref` on the operand stack, not an immediate

```
block $label0 (result …, exnref)
  block $label1 (result …)
    block $label2 (result exnref)
      try_table (catch_ref ($tag0, $label0), catch ($tag1, $label1), …,
                 catch_all_ref $label2)
        …
      end
    end
  end
end
```

Recap - Binary format (reference)

- `try_table` instruction = `0x1f`
  - Catches encoded as a vector of immediates
  - Four possible catch types:
    - `catch <tag> $label` = `0x00 <tag> <label>`
    - `catch_ref <tag> $label` `0x01 <tag> <label>`
    - `catch_all $label` = `0x02 <label>`
    - `catch_all_ref $label` = `0x03 <label>`
- `throw_ref` instruction = `0x0a`
- `exnref` type = `-0x17`

Updates

- Implementation in the reference interpreter (Andreas)
- Formal specification finished (Andreas)
- Binaryen translator, LLVM support (Heejin)
- V8 implementation (Thibaud)
- Firefox implementation (Ryan)
- Kotlin implementation (Zalim)
- Wizard implementation (Ben)
- JS API (Derek)
- WAMR Phase 3, exnref implementation (Siemens, Amazon)

Deprecation Plan

- Deprecation (removal) of old instructions will only happen the usage is low enough
- New engines will only have to support `exnref`, `try_table`, and `throw_ref`
- Phase 3 vestiges will be removed
  - `try-catch-catch_all`
  - `rethrow`
  - `Try-delegate`
- Chrome is tracking usage counters

Exnref status summary

- Formal spec: ✅
- Reference Interpreter: ✅
- Binaryen translator: ✅
- JS API spec: mostly ✅
- LLVM backend: 💚 in progress

- Language support
  - Kotlin - ✅ under a flag
  - Dart - 💙 planned
  - Ocaml - pinged

- Engine support
  - V8: ✅
  - Firefox: ✅
  - JSC: 💙 planned
  - WAMR: 💚 in progress

Phase 4 Discussion

- We are close!

Entry requirements:

- Two or more Web VMs have implemented the feature and pass the test suite (where applicable). ✅
- At least one toolchain has implemented the feature (where applicable). ⭐
- The spec document has been fully updated in the forked repo. ✅
- The reference interpreter has been fully updated in the forked repo and passes the test suite. ✅
- The Community Group has reached consensus in support of the feature and consensus that its specification is complete. ⭐

- Discussion questions
  - Co-chairs consider more complete LLVM support preferable before advancement
  - What set does “at least one toolchain” refer to? What languages and/or compilers?