Garbage Collection with Two Types

WebAssembly CG
August 11, 2020

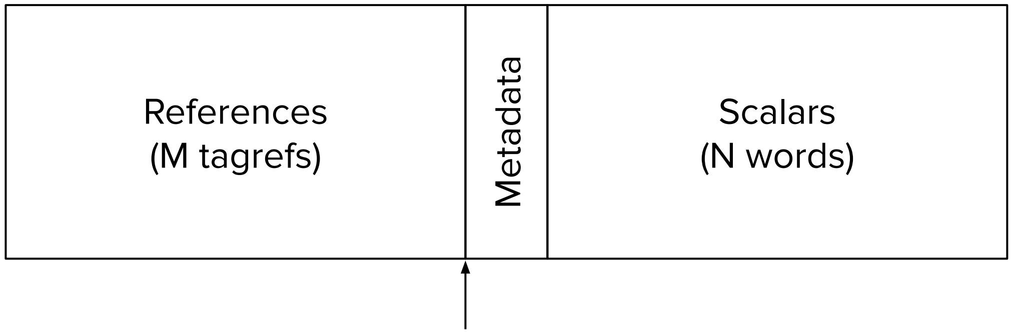
https://github.com/WebAssembly/gc/issues/109

Overview

- Simple intermediate; not intended to replace the existing GC proposals.
- Only two types, no subtyping or type parameters.
- Intended to be extensible to a future iteration with more fine-grained types.

structref

- Combination of a list of references with a chunk of linear memory (“scalars”).
- Easy to implement at a low level using “butterfly” structure.
- Simple, if inefficient, lowering for most types in common use.



structref operations

- structref.new
  - Allocate a new structref with the given number of reference and scalar fields.
- structref.get_bounds
- structref.assert_bounds/br_unless_bounds
  - Ensure that the structref has *at least* the number of reference and scalar fields. Important for removing bounds checks.
- structref.get/set_tagref
- structref.get/set_<numeric>
- structref.eq

tagref

- Provides a way to implement polymorphism by packing a type tag with a value.
- Some types have canonical tags, used to enable specialized representations.
    - At minimum, structref and funcref should have canonical tags.
- Custom tag definition TBD, but possibly reusing “events” from exception handling.
- Unlike structref, tagref is immutable and does not have equality operations, allowing optimized layout.

tagref operations

All operations have a $tag id attached.

- tagref.pack
- tagref.unpack
  - Traps on mismatched tag.
- tagref.test
  - Boolean tag check.
- tagref.br_unless_unpack
  - Branches on mismatched tag.

Limitations

- All references have to be unpacked to be used.
  - Opportunities for optimization (e.g. inferring a more fine-grained type) exist.
- Interior pointers are fat. Potentially very fat in the case of nested structures, requiring both reference and data offsets.

Summary

Potentially simple stepping stone to GC with fine-grained typing needs embedder / compiler support to help gather data.

Anyone interested?