

Branch Hinting

Vote to phase 4

Yuri Iozzelli

Proposal Summary

Improve the performance of machine code generated by Wasm engines, by hinting that a particular conditional branch destination is very likely/unlikely.

This allows the engine to make better decisions for code layout (improving instruction cache hits) and register allocation.

The hints are listed in the custom section `metadata.code.branch_hint` in binary format, and in the corresponding custom annotation in text format.

Live draft: <https://webassembly.github.io/branch-hinting/>






How we got here

- first proposed: Sept 2020
 - just add 2 instructions! `br_if_likely`, `br_if_unlikely`
- Maybe a custom section is more appropriate. But then:
 - need a custom section format to reference instructions
 - need a text version, to not lose the info when converting to text, and for easier manual annotation
 - make the format general so it can be used by future similar features
 - the reference interpreter now needs to handle custom sections!
 - where does the spec for this go?

Phase status

Currently phase 3.

Requirements for phase 4:

- Two or more Web VMs have implemented the feature and pass the test suite  (V8, JSC, SM).
- At least one toolchain has implemented the feature  (Cheerp, wasm-tools, wabt).
- 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  .

Vote for phase 4

Extra - V8 Implementation

br_on_null

```
void WasmGraphBuilder::BrOnNull(Node* ref_object, Node** null_node,
                                Node** non_null_node) {
    BranchExpectFalse(gasm_>WordEqual(ref_object, RefNull()), null_node,
                      non_null_node);
}
```

br_if

```
void BrIf(FullDecoder* decoder, const Value& cond, uint32_t depth) {
    SsaEnv* fenv = ssa_env_;
    SsaEnv* tenv = Split(decoder->zone(), fenv);
    fenv->SetNotMerged();
    WasmBranchHint hint = WasmBranchHint::kNoHint;
    if (branch_hints_) {
        hint = branch_hints_->GetHintFor(decoder->pc_relative_offset());
    }
    switch (hint) {
        case WasmBranchHint::kNoHint:
            builder_->BranchNoHint(cond.node, &tenv->control, &fenv->control);
            break;
        case WasmBranchHint::kUnlikely:
            builder_->BranchExpectFalse(cond.node, &tenv->control, &fenv->control);
            break;
        case WasmBranchHint::kLikely:
            builder_->BranchExpectTrue(cond.node, &tenv->control, &fenv->control);
            break;
    }
    builder_->SetControl(fenv->control);
    SetEnv(tenv);
    BrOrRet(decoder, depth, 1);
    SetEnv(fenv);
}
```

Extra - Benchmarks

Measured different workloads running in CheerpX (X86 VM and JIT compiler). CheerpX generate branch hints in JITted modules when checking for slow paths.

Average speedup: 7-10%.

The average C/C++ program will likely not benefit as much, but interpreters, runtimes, VMs, JITs, ... will.

Benchmark environment available here: <https://yuri91.github.io/webvm-benches>