Run Wasm from Rust host

Roman Zaynetdinov

https://github.com/rust-finland/presentations

WebAssembly

WebAssembly (abbreviated Wasm) is a binary instruction format for a stack-based virtual machine. Wasm is designed as a portable target for compilation of high-level languages like C/C++/Rust, enabling deployment on the web for client and server applications.

- Efficient and fast
- Safe
- Open and debuggable
- Part of the open web platform

WebAssembly Applications

Inside the browser:

- Optimize existing applications
- Better support for languages that are cross compiled for the web

Outside the browser:

- Compute of untrusted code
- Embedding of applications

Application Sandboxing

- Each WebAssembly module executes in sandbox environment
- All external functions must be explicitly provided by host
- Designed for Web with non-Web scenarios in mind
- No JavaScript required

Memory

A linear memory is a contiguous, byte-addressable range of memory spanning from offset 0 and extending up to a varying memory size.

Linear memories (default or otherwise) can either be imported or defined inside the module.

Wasm Interpreter

wasmi from Parity

Alternatives:

- wasmer
- wasm-time from Cranelift

Host Functions

```
extern "C" {
      // Host provides this function
      fn multiply(a: i32, b: i32) -> i32;
}

// Usage
unsafe { multiply(a, b) }
```

Supported types: `i32`, `i64`, `f32`, `f64`

Export a Function

```
#[no_mangle]
pub extern "C" fn add(a: i32, b: i32) -> i32 {
      a + b
}
```

```
// Call from host
let res = instance.invoke_export("add", &[3.into(), 6.into()], &mut runtime)?;
assert_eq!(res, Some(9.into()));
```

Demo

- Call wasm function
- Call host function from wasm
- Pass a string to wasm
- Return a string from wasm
- Call an object’s method from wasm

Downsides

- A lot of tooling for Web use cases
- Provide API for everything
- Useful proposals are still coming
- Recent technology

Upsides

- A lot of interest in WebAssembly
- New projects frequently appearing

Questions?

Wasm Code Size

- Use wee_alloc allocator instead of default one
- Replace uncalled functions with `unreachable` (wasm-snip)
- Remove unneeded exports, imports, functions, etc (wasm-gc)
- Compiling with Link Time Optimizations (LTO)
- Optimize wasm binary with wasm-opt (Reference)
- Use twiggy to analyze generated wasm

Pass a String to and from Wasm (part 1)

```
function passStringToWasm(arg) {
  const buf = new TextEncoder('utf-8').encode(arg);
  const len = buf.length;
  const ptr = wasm.__wbindgen_malloc(len);
  let array = new Uint8Array(wasm.memory.buffer);
  array.set(buf, ptr);
  return [ptr, len];
}
```

Pass a String to and from Wasm (part 2)

```
function getStringFromWasm(ptr, len) {
  const mem = new Uint8Array(wasm.memory.buffer);
  const slice = mem.slice(ptr, ptr + len);
  const ret = new TextDecoder('utf-8').decode(slice);
  return ret;
}
```

Pass a String to and from Wasm (part 3)

```
export function greet(arg0) {
  const [ptr0, len0] = passStringToWasm(arg0);
  try {
    const ret = wasm.greet(ptr0, len0);
    const ptr = wasm.__wbindgen_boxed_str_ptr(ret);
    const len = wasm.__wbindgen_boxed_str_len(ret);
    const realRet = getStringFromWasm(ptr, len);
    wasm.__wbindgen_boxed_str_free(ret);
    return realRet;
  } finally {
    wasm.__wbindgen_free(ptr0, len0);
  }
}
```

Pass a String to and from Wasm (part 4)

```
pub extern "C" fn greet(a: &str) -> String {
    format!("Hello, {}!", a)
}

#[export_name = "greet"]
pub extern "C" fn __wasm_bindgen_generated_greet(
    arg0_ptr: *const u8,
    arg0_len: usize,
) -> *mut String {
    let arg0 = unsafe {
        let slice = ::std::slice::from_raw_parts(arg0_ptr, arg0_len);
        ::std::str::from_utf8_unchecked(slice)
    };
    let _ret = greet(arg0);
    Box::into_raw(Box::new(_ret))
}
```

References

- https://webassembly.org/
- https://github.com/paritytech/wasmi
- https://webassembly.github.io/wabt/demo/wat2wasm/
- https://webassembly.github.io/wabt/demo/wasm2wat/
- https://rustwasm.github.io/wasm-bindgen/contributing/design/index.html
- https://github.com/rustwasm/wee_alloc
- https://rustwasm.github.io/wasm-pack/
- https://rustwasm.github.io/book/game-of-life/code-size.html
- https://rustwasm.github.io/
- https://github.com/alexcrichton/wasm-gc
- https://github.com/rustwasm/twiggy
- https://github.com/WebAssembly/wabt