A WebAssembly Primer

Alexandre St-Louis Fortier
Stefan Knudsen
Cheuk Chuen Siow

COMP 520 - Group 14

April 22, 2016

WebAssembly is a low-level programming language designed to be efficient and faster than JavaScript.

WebAssembly has a binary format, `.wasm`, and an abstract tree format, `.wast`

We used the abstract tree format, which has a syntax based on S-expressions

Within a function, all local variables must be declared at the beginning

The following is the syntax that we used

Syntax

```
var: <int> | $<name>
name: (<letter> | <digit> | _ | . | + | - | *
| / | \ | ^ | ~ | = | < | > | ! | ? | @ | #
| $ | % | & | | | : | ' | `)+
string: ``(<char> | \n | \t | \\
| \' | \" | \<hex><hex>)*''
```

Types

```
value: <int> | <float>
type: i32 | i64 | f32 | f64
```

Operators

```
binop: eq | ne | lt_s | le_s | gt_s | ge_s | add
| sub | or | and | xor | mul | div_s | rem_s | shl_s
| shr_s | and | or | shr_s
unop:
```

We ended up defining our GoLite unary primitives in terms of the binary operations

Expressions

```
expr: ( block <name>? <expr>* )
| ( loop <name1>? <name2>? <expr>* )
;; = (block <name1>? (loop <name2>? (block <expr>*)))
| ( if <expr> ( then <name>? <expr>* )
( else <name>? <expr>* )? )
| ( if <expr1> <expr2> <expr3>? )
;; = (if <expr1> (then <expr2>) (else <expr3>?))
| ( br <var> <expr>? )
| ( return <expr>? )
;; = (br <current_depth> <expr>?)
```

More expressions

```
expr: ( call <var> <expr>* )
| ( get_local <var> )
| ( set_local <var> <expr> )
| ( <type>.load((8|16|32)_<sign>)?
<offset>? <align>? <expr> )
| ( <type>.store(8|16|32)? <offset>?
<align>? <expr> <expr> )
| ( <type>.const <value> )
| ( <type>.<binop> <expr> <expr> )
```

```
func:   ( func <name>? <type>? <param>*
<result>? <local>* <expr>* )
param:  ( param <type>* ) | ( param <name> <type> )
result: ( result <type> )
local:  ( local <type>* ) | ( local <name> <type> )
```

Header

We used the following functions for the `gen.ml` header:

```
expr:
( br_if <var> <expr>? <expr> )
( call_import <var> <expr>* )
module:  ( module <type>* <func>* <import>* <export>*
<table>* <memory>? <start>? )
import:  ( import <name>? <string> <string>
(param <type>* ) (result <type>)* )
start:   ( start <var> )
memory:  ( memory <int> <int>? <segment>* )
```

References

https://github.com/WebAssembly/spec/blob/master/ml-proto/README.md