Compiling to WebAssembly

Seminar project by Vincent Hofer

Assisted by Manuel Leuenberger & Oli Flückiger

Project goals

- Explore parser generators and WebAssembly specification
- Build a compiler pipeline targeting WebAssembly
- Develop a new simple language without prior compiling knowledge

What is WebAssembly?

“WebAssembly is a new type of code that can be run in modern web browsers. It is a binary instruction format for a stack-based virtual machine”

- Near-native performance
- Supported by all main browser manufacturers
- Has a textual representation, besides binary format
- Intended as compilation target for languages like C/C++

WebAssembly Text Format

```
def main {
    var x;
    x = 5;
    call foo(x);
};

def foo(num) {
    num*2;
};
```

```
(module
  (func $main (result f64)
    (local $x f64)
    (set_local $x (f64.const 5))
    (call $foo (get_local $x))
  )

  (func $foo (param $num f64) (result f64)
    (f64.mul (get_local $num)(f64.const 2))
  )
  (export "main" (func $main))
)
```

Development of my language and compiler

Iteration 1

```
2*3+(1+2)/3
```

Development of my language and compiler

Iteration 2

```
var x;
x = 3;
x + 2;
```

Development of my language and compiler

Iteration 3

```
def main{
    var a;
    a = 3;
    call add(a)(2);
};

def add(x)(y){
    x+y;
};
```

Development of my language and compiler

Iteration 4

```
def fib(x) {
    if (x<=2) {
        x=1;
    } else {
        x = call fib(x-2) + call fib(x-1);
    };
    x;
};
```

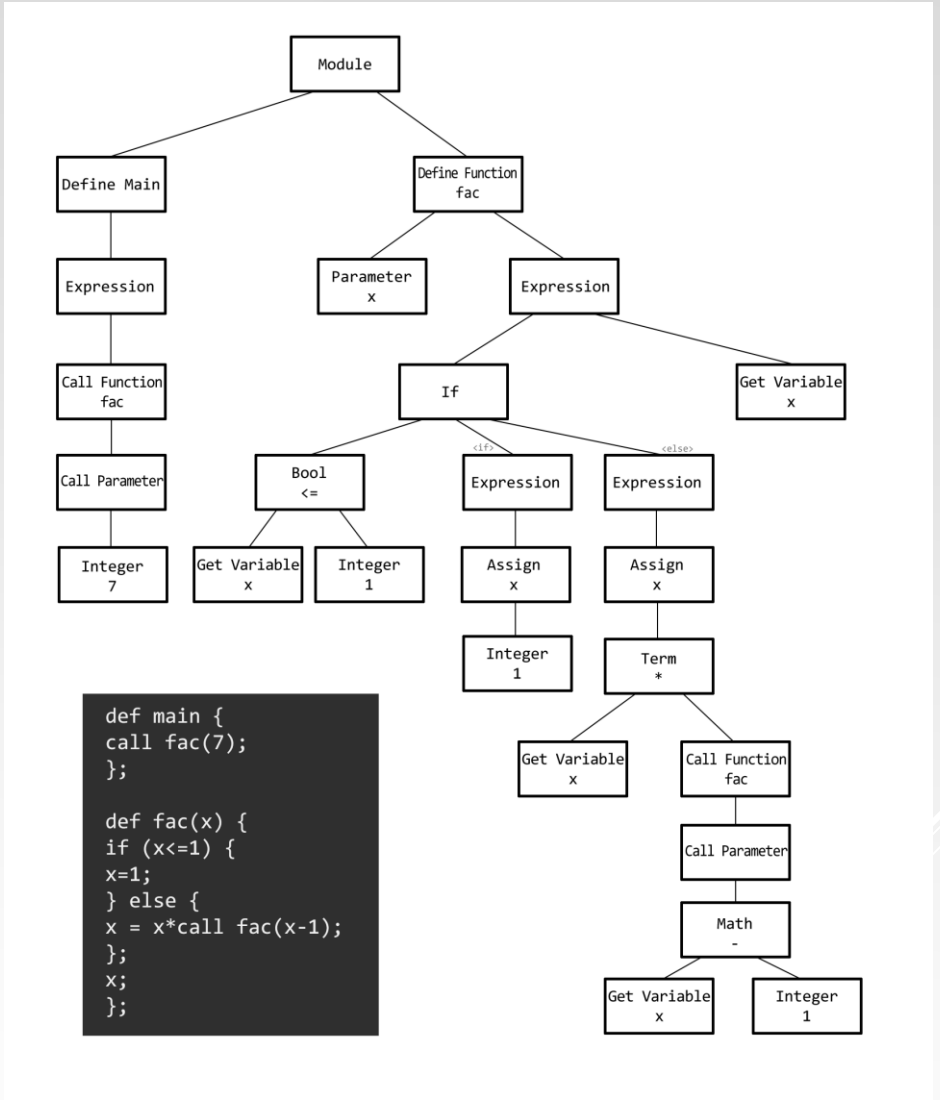
Development of my language and compiler

Iteration 5

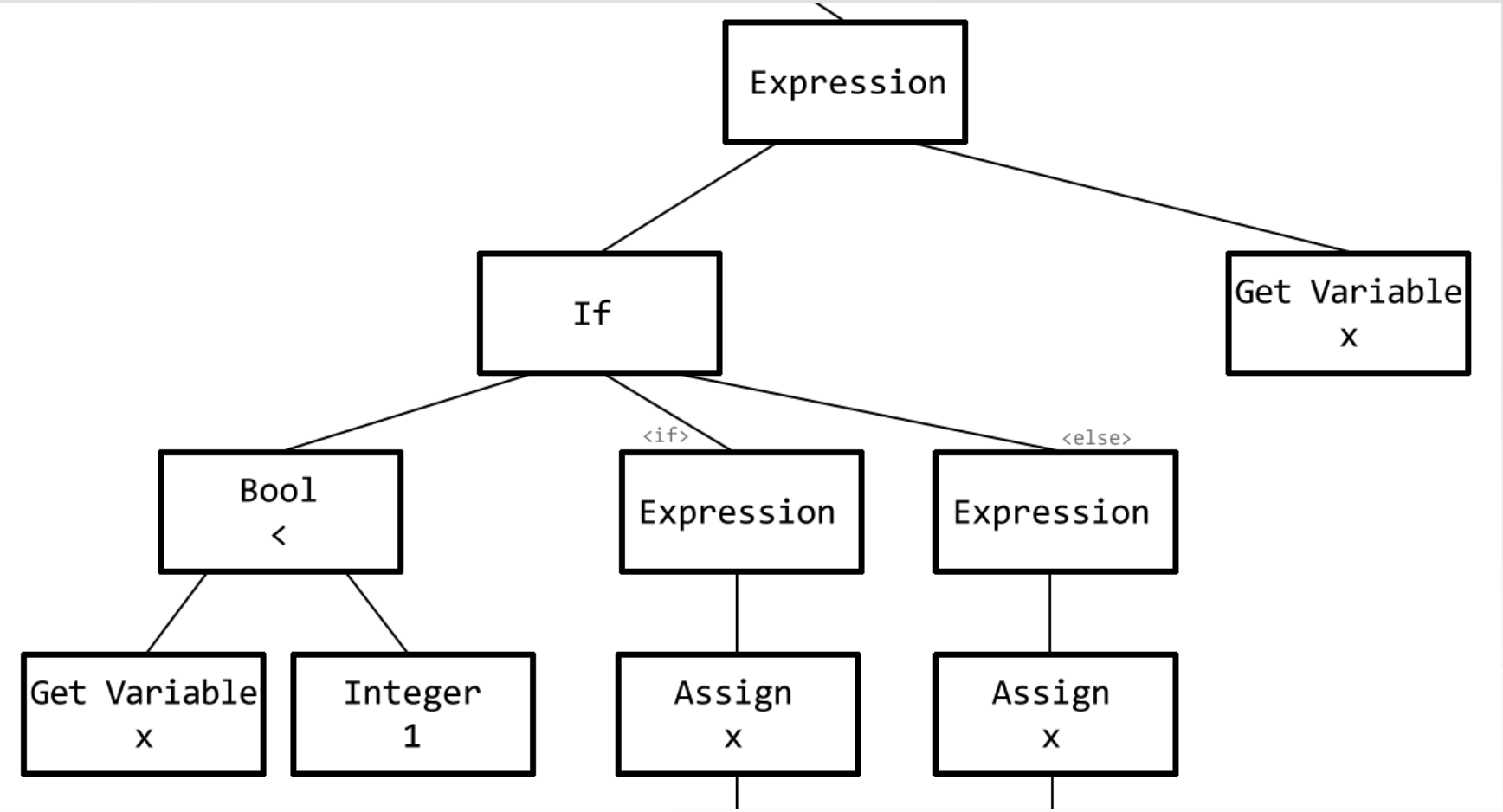
```
def main {
    var x;
    x = 3;
    array a [1 2 x 4 5];
    len a + get a [2];
};
```

Parser and AST

- Parser generator: PEG.js
- In the beginning: Direct output of text format
- Later: Secondary step through AST



Parser and AST



Squareroot implementation

WebAssembly Compiler Suite by Manuel Leuenberger

Source load save

```
def main {
    sqrt(2);
};
```

Grammar load save

```
          accept() {
        return visit.visitTerm(this);
    }
    }

    class Parameter extends Node {
          constructor(char) {
              super();
                  this.name = "Parameter";
                  this.data = [char];
          }
        accept() {
        return visit.visitParameter(this);
    }
    }

    class Squareroot extends Node {
          constructor(math) {
              super();
              this.name = "Squareroot";
              this.children = [math];
          }
          accept() {
              return visit.visitSquareroot(this);
          }
    }

    class Factor extends Node {
          constructor(exp) {
              super();
                  this.name = "Factor";
                  this.children = [exp];
          }
        accept() {
        return visit.visitFactor(this);
    }
    }

    class CreateArray extends Node {
        constructor(char,elements) {
            super();
            this.name = "Array";
            this.data = [char];
```

A C W

Output load save

```
1.4142135623730951
```

Source load save

```
def main {
    sqrt(2);
};
```

G

A

Compiler load save

```
            return "(f64.load offset=".concat(off,"(i32.const 0))");
        }
    } else { console.error("Error: Array",node.data[0],"not yet created"); return "";}
}

visitSetArrayElement(node) {
    if (memarr.includes(node.data[0])) {
        var i = memarr.indexOf(node.data[0]);
        var off;
        if (node.children[0].data[0] >= memarr[i+2]) {console.error("Error: Array index out of bound");}
        else {
            off = (Number(memarr[i+1]) + Number(node.children[0].data[0]))*8;
            return "(f64.store offset=".concat(off,"(i32.const 0)",node.children[1].accept(),")");
        }
    } else {console.error("Error: Array",node.data[0],"not yet created"); return "";}
}

visitArrayLength(node) {
    if (memarr.includes(node.data[0])) {
      var i = Number(memarr.indexOf(node.data[0]))
      return "(f64.const ".concat(memarr[i+2],")");
    } else {console.error("Error: Array",node.data[0],"not yet created"); return "";}
}

visitSquareroot(node) {
    return "(f64.sqrt ".concat(node.children[0].accept(),")");
}

visitFactor(node) {
    return node.children[0].accept();
}

visitInteger(node) {
    return "(f64.const ".concat(node.data[0],")");
}

visitGetVariable(node) {
    return "(get_local $".concat(node.data[0],")");
}
}

window.visit = new Visitor();
```

W

Output load save

```
1.4142135623730951
```

S G AST load save C WAT load save O

```
{
  "name": "Module",
  "data": [],
  "children": [
    {
      "name": "Define Main",
      "data": [],
      "children": [
        {
          "name": "Expression",
          "data": [],
          "children": [
            [
              {
                "name": "Squareroot",
                "data": [],
                "children": [
                  {
                    "name": "Integer",
                    "data": [
                      "2"
                    ],
                    "children": []
                  }
                ]
              },
              null
            ]
          ]
        }
      ]
    },
    []
  ]
}
```

```
(module
(memory $0 1)
(func $main (result f64)
(f64.sqrt (f64.const 2))
)
(export "main" (func $main))
)
```

WebAssembly vs. JavaScript

```
def fib(x) {
if (x<=2) {
x=1;
} else {
x = call fib(x-2) +
call fib(x-1);
};
x;
};
```

```
function fib(x) {
if (x <= 2) return 1;
return fib(x-2) +
fib(x-1);
}
```

Summary

- WebAssembly can be used alongside JavaScript
- Learned about compilers, grammars and ASTs
- Wrote a language targeting WebAssembly
- Compiler can be expanded in the future, e.g.
  - Data types
  - Strings
  - Objects

Source of image in slide 3: https://en.wikipedia.org/wiki/WebAssembly