Design and implementation of Mystia, the new front-end programming language be compiled to WebAssembly

Researcher: Kajizuka Taichi
Date: March 9, 2025

Abstract: In this research, I suggest Mystia programming language and study about design and implementation of its compiler that outputs WAT (WebAssembly Text Format) as output target.
This project's goal is to be user-friendly choice more than C/C++, Rust, AssemblyScript and other exist rivals at web front-end development that requires high performance code for everyone especially beginner.

1. Language design

There's example code snippet of Mystia. I can say the syntax is inspired by Rust and OCaml. This program solves total value form 0 to specified value. Local variable declaring is done by the compiler with type inference automatically. However, type annotations in function are necessary for base of type inference.

```
fn total(number as int) as int = {
    let count = 1;
    let result = 0;

    while count < number loop {
        let result = result + count;
        let count = count + 1
    };
    result
};

total(10)
```

In language designing, I focused readability and explicitly for beginner. Take for example, `let` statement explicit shows purpose "assignment data in the variable".

2. Compiler implementation

The compiler of Mystia is written in Rust. If you compile earlier code snippet, the compiler generate WAT-style code like below.

```
(module
  (func $total (param $number i32) (result i32)
    (local $count i32) (local $result i32)
    (local.set $count (i32.const 1))
    (local.set $result (i32.const 0))

    (block $outer (loop $while_start
      (br_if $outer (i32.eqz (i32.lt_s
        (local.get $count) (local.get $number))))

      (local.set $result
        (i32.add (local.get $result) (local.get $count)))
      (local.set $count
        (i32.add (local.get $count) (i32.const 1)))

      (br $while_start)))
    (local.get $result))

  (func (export "_start") (result i32)
    (call $total (i32.const 10))))
```

The compiler's algorithm is very simple. There's AST (Abstract Syntax Tree) nodes such as expression operator and statement. Each node has methods `parse` and `compile`, to dispersing complex logics.

3. Overall

I experience compiler implementation, I realize WAT is very "easy to generate" format rather than LLVM IR and other assembly language. From now on, WASM is improving rapidly. So Mystia programming language is expectable as thing become one of choice at web development society too.