# EulerLisp VM

Leon Rische

*[2019-08-18 Sun 19:40]*

## Contents



## 1 Arithmetic Primitives

Arithmetic primitives work by assigning `val` to the result of combining `val` and `arg1` using the given primitive.

`(+ a b)` compiles to

```
<Code for b>
PUSH_VAULE
<Code for a>
POP_ARG1
ADD
```

Arguments are evaluated from right to left and pushed to the stack, then popped to `arg1` one by one. This is important for primitives that are not commutative, e.g. `/`.

This easily extends to an arbitrary number of arguments.

Consider `(+ a b c d)` which compiles to

```
<Code for d>
PUSH_VAULE
<Code for c>
PUSH_VAULE
<Code for b>
PUSH_VAULE
<Code for a>
```

```
POP_ARG1
ADD
POP_ARG1
ADD
POP_ARG1
ADD
```

Previously, primitives with more than two arguments were implemented using builtin functions. The changes in performance are negligible since arithmetic primitives with more than two arguments are rarely used.

Running Project Euler Problems 1 to 50: Before: 29.446s / 30.340s / 29.978s After: 28.455s / 28.356s / 29.118s