Stack Trace

Overview

TEAL (Transaction Execution Approval Language) is an assembly-like stack language used to write apps for the Algorand blockchain. *Stack Trace* is a simple tool to aid in debugging TEAL by parsing TEAL opcodes, visualizing the stack and storage after the execution of each opcode and logging the global/local states on completion of the TEAL program. The initial focus of *Stack Trace* is to support basic mathematical and storage functions, which can be particularly challenging to mentally track during the authoring of TEAL code.

Progress report:

Stack Trace currently supports the following opcodes:

- *int*
- *+*
- *-*
- ***
- */*
- *byte*
- *store*
- *load,*
- *app_local_get*
- *app_global_get*
- *app_local_put*
- *app_global_put*

Extending:

To add more opcodes, simply add the opcode as a new key to the *opCodes* object in opcodes.js. A complete list and description of TEAL opcodes is available at:

https://developer.algorand.org/docs/get-details/dapps/avm/teal/opcodes/

The format of a *Stack Trace* opcode value should look like:

```
app_global_put: {
        pushes: {
            number: 0,
            type: "any"
        },
        pops: {
            number: 2,
            type: "any"
        },
        op: function(stack,args,undefined,accounts,app_global){
            app_global[args[0]] = args[1]
        },
        inline: false
    }
```

Opcode object keys:

- ***pushes.number***: number of values (if any) that will be added to the stack
- ***pops.number:*** number of values (if any) that will be removed from the stack and used as arguments for the *op* function
- ***op***: function that will be executed by the opcode (all *op* functions must take in the args *(stack, args, storage, accounts, app_global)*
- ***inline****: boolean value that indicates whether the opcode takes arguments from the stack or inline*

Usage:

Stack Trace is very simple to use, as it currently has no dependencies. After cloning the repository, simply paste the TEAL code to evaluate into the *teal* var in main.js. To add dummy transactions and/or app global/local state values, simply modify the *txns*, *app_global*, and/or *accounts* variables in main.js. In the terminal, enter:

```
node main.js
```

The terminal will log the supported opcodes, then proceed to evaluate each opcode and log data in the following format:

```
OpCode:
+
Pops:
2
Type:
uint64
[ 125, 150 ]
Stack after opcode +:
[ 275 ]
```

Finally, on completion of the TEAL program, the tracer will log the current local/global states:

```
App Global State:
{ depositAmount: 200, staked: 90 }
App Local State:
{ amt: 150 }
```