Learn Huff

And become an EVM chad

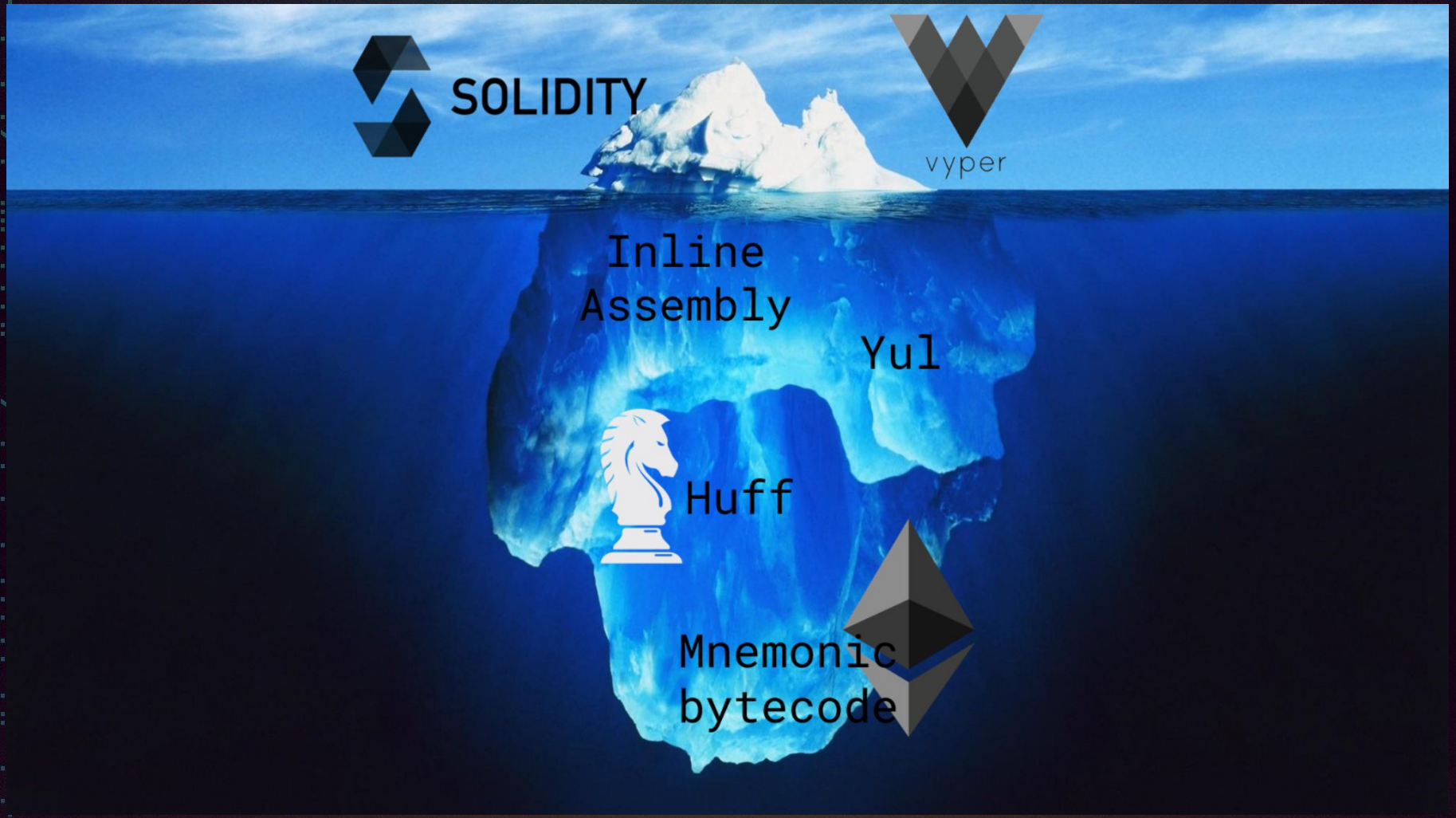
Clément Lakhal

Huff enjoyer

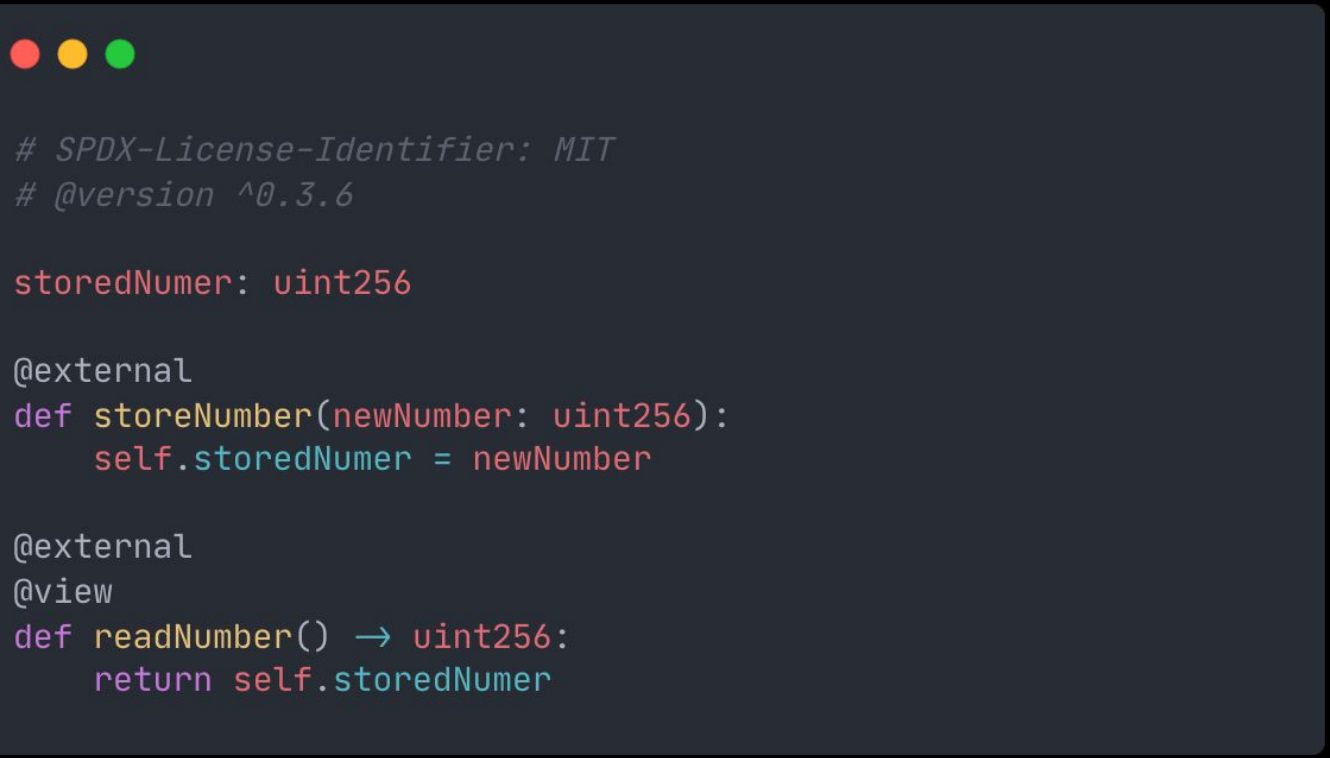
What the hell is Huff?

What is Huff?

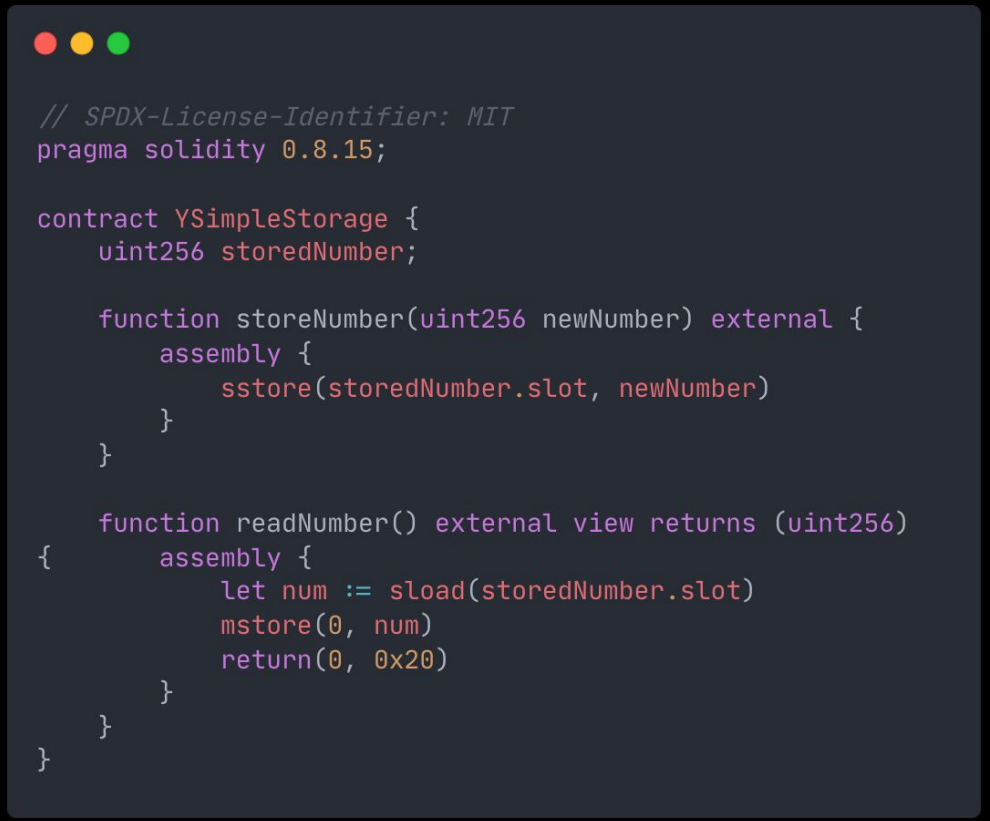
Huff is a low-level programming language designed for developing highly optimized smart contracts that run on the Ethereum Virtual Machine (EVM).


SOLIDITY
vyper
Inline
Assembly
Yul
Huff
Mnemonic
bytecode

```solidity
// SPDX-License-Identifier: MIT
pragma solidity 0.8.15;

contract SSimpleStorage {
    uint256 storedNumber;

    function storeNumber(uint256 newNumber) external {
        storedNumber = newNumber;
    }

    function readNumber() external view returns (uint256) {
        return storedNumber;
    }
}
```



```
# SPDX-License-Identifier: MIT
# @version ^0.3.6

storedNumer: uint256

@external
def storeNumber(newNumber: uint256):
    self.storedNumer = newNumber

@external
@view
def readNumber() -> uint256:
    return self.storedNumer
```



```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.15;

contract YSimpleStorage {
    uint256 storedNumber;

    function storeNumber(uint256 newNumber) external {
        assembly {
            sstore(storedNumber.slot, newNumber)
        }
    }

    function readNumber() external view returns (uint256)
{       assembly {
            let num := sload(storedNumber.slot)
            mstore(0, num)
            return(0, 0x20)
        }
    }
}
```

```
object "YYSimpleStorage" {
     code {
        // Contract deployment
        datacopy(0, dataoffset("runtime"), datasize("runtime"))
        return(0, datasize("runtime"))
     }
     object "runtime" {
        code {
            // The MAIN function of the contract, function dispatcher
            switch selector()
            // cast sig "storeNumber(uint256)"
            case 0xb6339418 {
                storeNumber(decodeAsUint(0))
            }
            // cast sig "readNumber()"
            case 0xb63d343f{
                returnUint(readNumber())
            }
            default {
                revert(0, 0)
            }

            function storeNumber(newNumber) {
                sstore(0, newNumber)
            }

            function readNumber() -> storedNumber{
                storedNumber := sload(0)
            }

            /* ---------- calldata decoding functions ----------- */
            function selector() -> s {
                s := div(calldataload(0),
0x100000000000000000000000000000000000000000000000000000000)
            }

            function decodeAsUint(offset) -> v {
                let pos := add(4, mul(offset, 0x20))
                if lt(calldatasize(), add(pos, 0x20)) {
                    revert(0, 0)
                }
                v := calldataload(pos)
            }

            function returnUint(v) {
                mstore(0, v)
                return(0, 0x20)
            }
        }
     }
}
```

```
PUSH1 0x80 PUSH1 0x40 MSTORE CALLVALUE DUP1 ISZERO PUSH2 0x10 JUMPI PUSH1 0x0 DUP1 REVERT JUMPDEST POP
PUSH2 0x150 DUP1 PUSH2 0x20 PUSH1 0x0 CODECOPY PUSH1 0x0 RETURN INVALID PUSH1 0x80 PUSH1 0x40 MSTORE
CALLVALUE DUP1 ISZERO PUSH2 0x10 JUMPI PUSH1 0x0 DUP1 REVERT JUMPDEST POP PUSH1 0x4 CALLDATASIZE LT
PUSH2 0x36 JUMPI PUSH1 0x0 CALLDATALOAD PUSH1 0xE0 SHR DUP1 PUSH4 0xB6339418 EQ PUSH2 0x3B JUMPI DUP1
PUSH4 0xB63D343F EQ PUSH2 0x57 JUMPI JUMPDEST PUSH1 0x0 DUP1 REVERT JUMPDEST PUSH2 0x55 PUSH1 0x4 DUP1
CALLDATASIZE SUB DUP2 ADD SWAP1 PUSH2 0x50 SWAP2 SWAP1 PUSH2 0xC3 JUMP JUMPDEST PUSH2 0x75 JUMP JUMPDEST
STOP JUMPDEST PUSH2 0x5F PUSH2 0x7F JUMP JUMPDEST PUSH1 0x40 MLOAD PUSH2 0x6C SWAP2 SWAP1 PUSH2 0xFF
JUMP JUMPDEST PUSH1 0x40 MLOAD DUP1 SWAP2 SUB SWAP1 RETURN JUMPDEST DUP1 PUSH1 0x0 DUP2 SWAP1 SSTORE POP
POP JUMP JUMPDEST PUSH1 0x0 DUP1 SLOAD SWAP1 POP SWAP1 JUMP JUMPDEST PUSH1 0x0 DUP1 REVERT JUMPDEST
PUSH1 0x0 DUP2 SWAP1 POP SWAP2 SWAP1 POP JUMP JUMPDEST PUSH2 0xA0 DUP2 PUSH2 0x8D JUMP JUMPDEST DUP2 EQ
PUSH2 0xAB JUMPI PUSH1 0x0 DUP1 REVERT JUMPDEST POP JUMP JUMPDEST PUSH1 0x0 DUP2 CALLDATALOAD SWAP1 POP
PUSH2 0xBD DUP2 PUSH2 0x97 JUMP JUMPDEST SWAP3 SWAP2 POP POP JUMP JUMPDEST PUSH1 0x0 PUSH1 0x20 DUP3
DUP5 SUB SLT ISZERO PUSH2 0xD9 JUMPI PUSH2 0xD8 PUSH2 0x88 JUMP JUMPDEST JUMPDEST PUSH1 0x0 PUSH2 0xE7
DUP5 DUP3 DUP6 ADD PUSH2 0xAE JUMP JUMPDEST SWAP2 POP POP SWAP3 SWAP2 POP POP JUMP JUMPDEST PUSH2 0xF9
DUP2 PUSH2 0x8D JUMP JUMPDEST DUP3 MSTORE POP POP JUMP JUMPDEST PUSH1 0x0 PUSH1 0x20 DUP3 ADD SWAP1 POP
PUSH2 0x114 PUSH1 0x0 DUP4 ADD DUP5 PUSH2 0xF0 JUMP JUMPDEST SWAP3 SWAP2 POP POP JUMP INVALID LOG2 PUSH5
0x6970667358 0x22 SLT KECCAK256 LOG4 CODECOPY 0xBE 0xE PUSH14 0xB7CE87077085C83AC565C40B20EC EQ 0xDE SAR
SWAP7 REVERT DIFFICULTY ORIGIN AND MSTORE 0x23 0xF 0xDF PUSH12 0x64736F6C634300080F003300
```

```
6080604052348015610010576000 80fd5b506101508061002060003960 00f3fe608060405234801561
001057600080fd5b506004361061003657600035 60e01c8063b63394181461003b578063b63d343f14
610057575b600080fd5b61005560048036038101906100509190 6100c3565b610075565b005b61005f
61007f565b60405161006c91906100ff565b604051809103 90f35b8060008190555050565b60008054
905090565b600080fd5b60008190509190 50565b6100a08161008d565b81146100ab57600080fd5b50
565b60008135905061 00bd81610097565b92915050565b60006020828403121561 00d9576100d86100
88565b5b60006100e784828501 6100ae565b91505092915050565b6100f98161 008d565b82525050 56
5b60006020820190506101146000830184 6100f0565b9291505056fea2646970667358221220a439be
0e6db7ce87077085c83ac565c40b20ec14de1d96fd44321652230fdf6b64736f6c634300080f0033
```

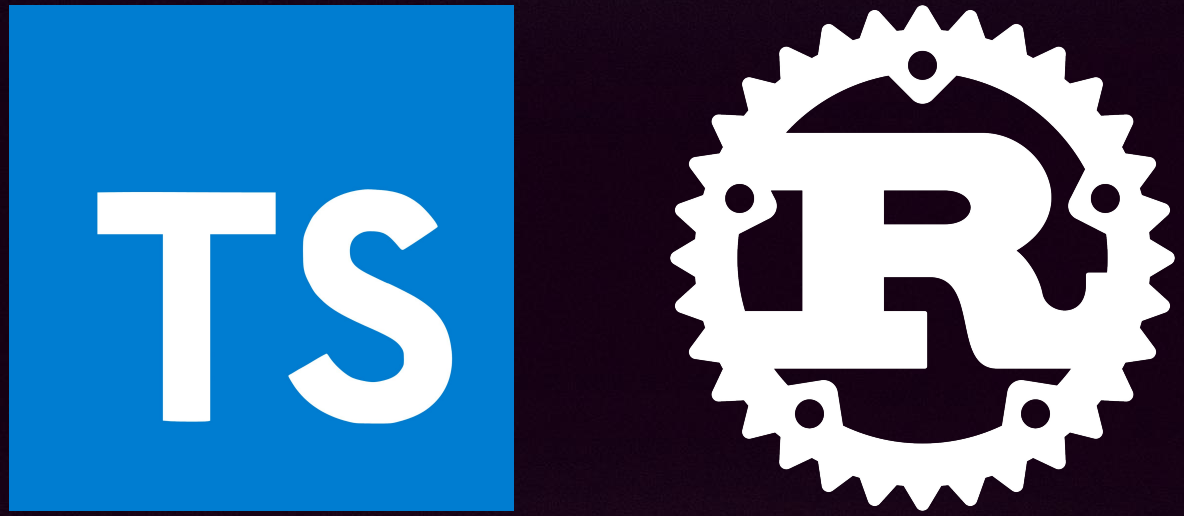
Why Huff?

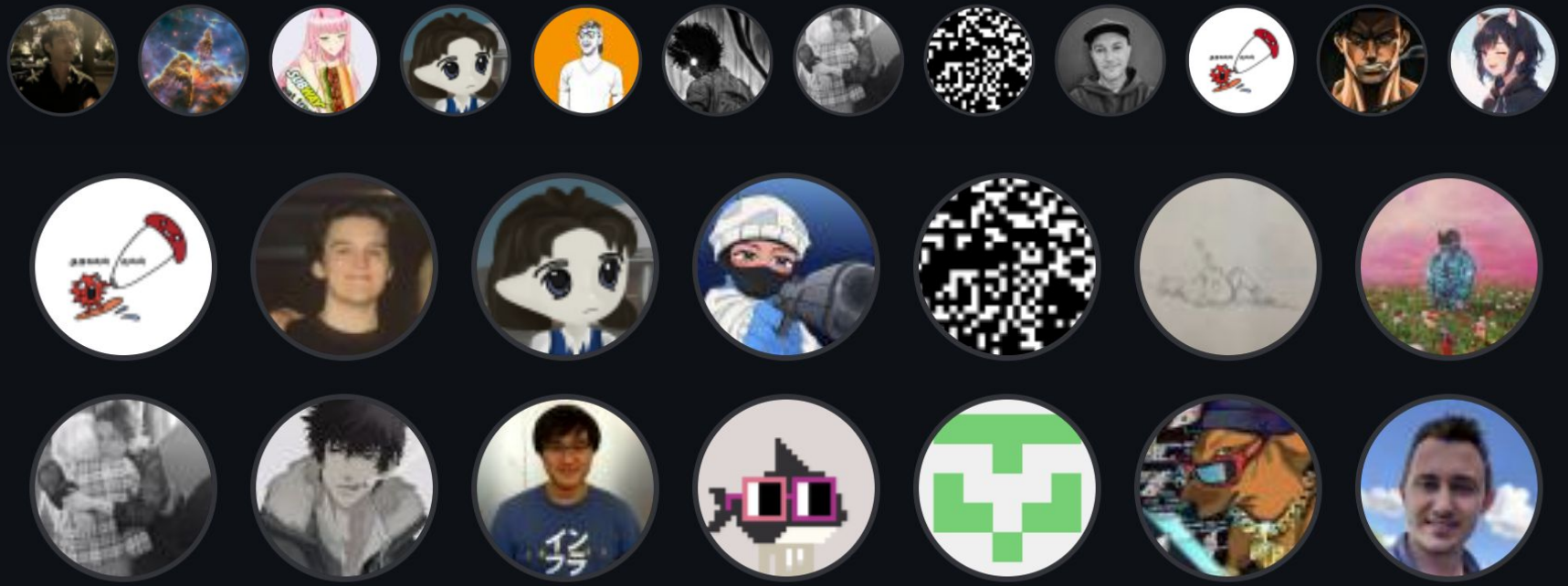

Aztec

weierstrudel is a highly optimized smart contract that performs elliptic curve scalar multiplication on the short Weierstrass 254-bit Barreto-Naehrig curve, formerly used by ZCash and currently available as a precompile smart-contract in the Ethereum protocol.

```
/**
 * @title It's a default greedy wnaf slice! For documentation see WNAF_GREEDY_SLICE
 * @dev if the scalar `w`'s least-significant set bit is in position 1, then we don't need to perform an
initial bit shift.
 *          this macro is the same as WNAF_GREEDY__SLICE, except we don't have `<num_shifted_bits_2>
shr`.
 *          So really it's just a more greedy version of the above, as we do less work for the same
effect.
 */
template<num_shifted_bits, wnaf_pointer_total_increase, wnaf_pointer_base_increase>
#define macro WNAF_GREEDY__SLICE_DEFAULT = takes(0) returns(0) {
    // w w o k v s m
    dup1 0x3e0 and dup7 add                         // w' w o
    dup3 <wnaf_pointer_total_increase> add swap3    // o w' w o'
    dup4 mload
    0x02 add                                // n o w' w o'
    dup1 dup6 mstore
    <wnaf_pointer_base_increase> sub add

    mstore                                          // w o'
    <num_shifted_bits> shr 0x10 add                 // w' o'
    dup1 WNAF_TABLE_MASK() and mload jump
}

/**
 * @title This is our jump table. We store this directly inside the contract bytecode to save on gas
 *          (we can use `codecopy` to initialize our table instead of a bunch of `mstore8` opcodes)
 * @dev We don't use a packed jump table, because that would require us to apply a mask after loading
our jump label from memory
 **/
#define jumptable WNAF_GREEDY__JUMP_TABLE {
    lsb_0 // 0 00000
    lsb_1 // 1 00001
    lsb_2 // 2 00010
    lsb_1 // 3 00011
    lsb_3 // 4 00100
    lsb_1 // 5 00101
    lsb_2 // 6 00110
    lsb_1 // 7 00111
    lsb_4  // 8 01000
    lsb_1  // 9 01001
    lsb_2  // 10 01010
    lsb_1  // 11 01011
    lsb_3  // 12 01100
    lsb_1  // 13 01101
    lsb_2  // 14 01110
    lsb_1  // 15 01111
}

#define macro WNAF_GREEDY__INIT_JUMP_TABLE = takes(0) returns(1) {
    __tablesize(WNAF_GREEDY__JUMP_TABLE) __tablestart(WNAF_GREEDY__JUMP_TABLE) 0x00 codecopy
}
```

"Huff was designed with a complete disregard for semantics, safety or basic consistency. Any similarities to production code are purely incidental and Huff should on no account be used by anybody."


TS

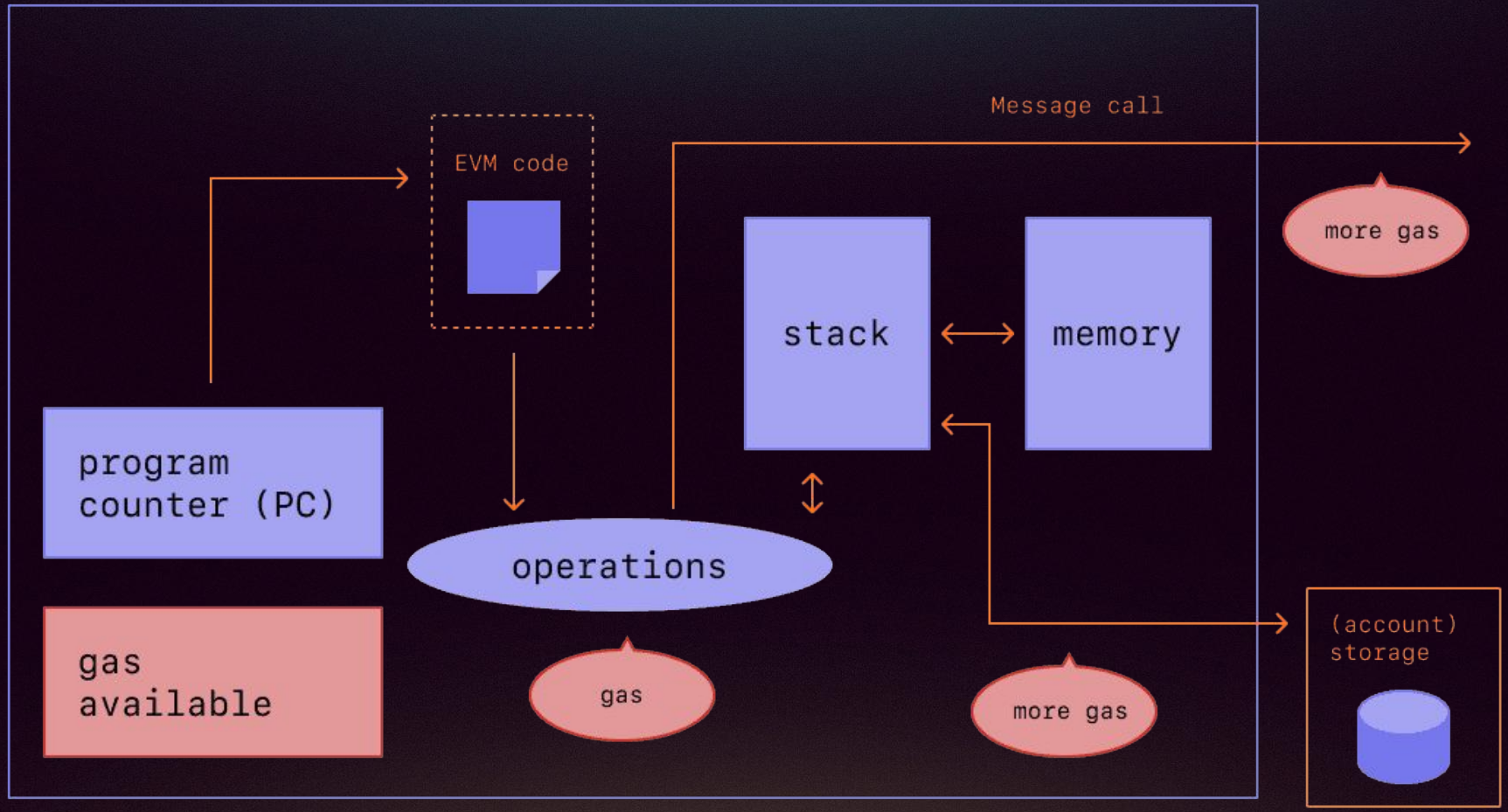


+ 15 contributors

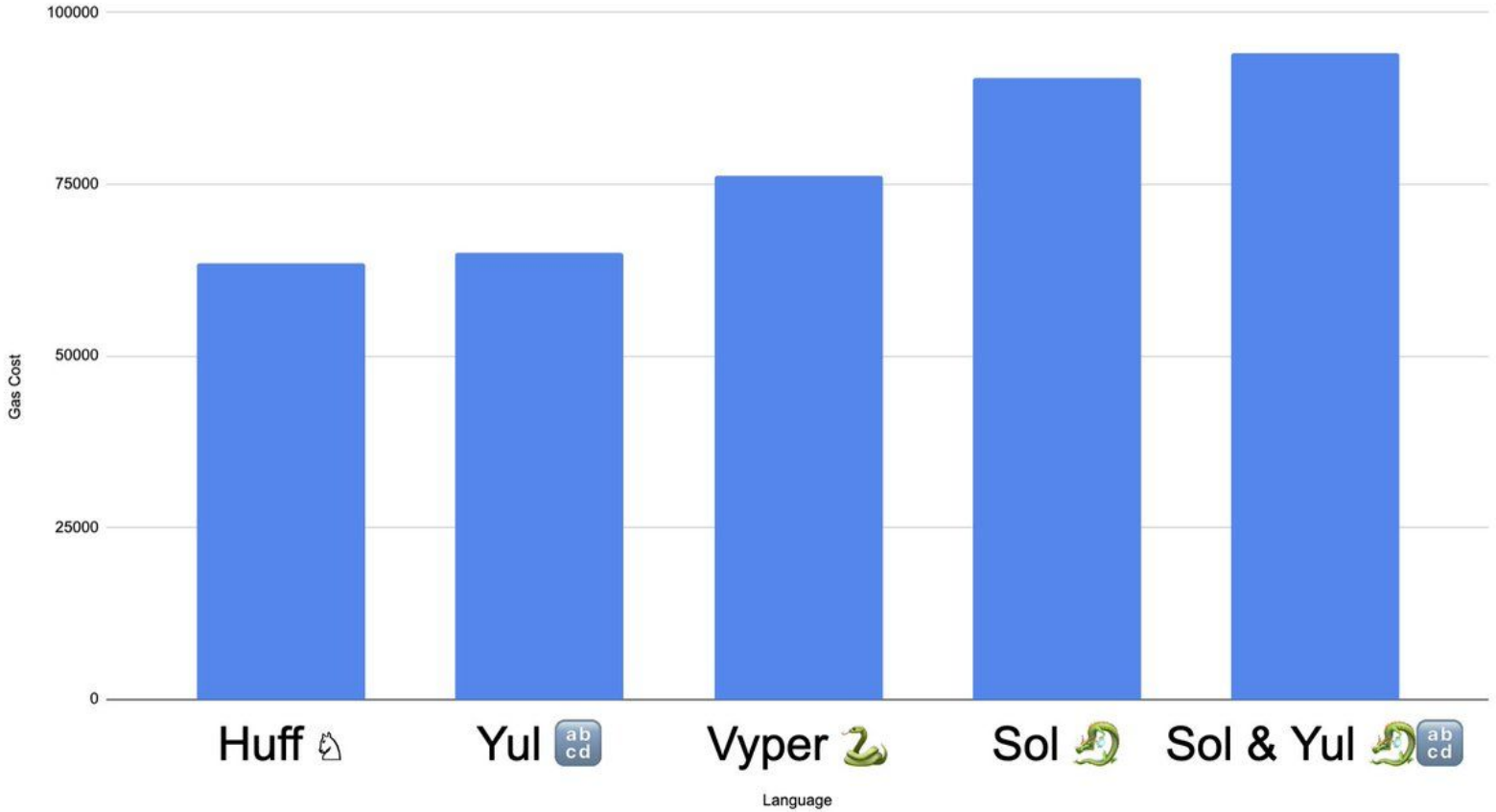
```
/* Interface */
#define function storeNumber(uint256) nonpayable returns ()
#define function readNumber() view returns (uint256)

/* Storage Slots */
#define constant VALUE_LOCATION = FREE_STORAGE_POINTER()

/* Methods */
#define macro SET_VALUE() = takes (0) returns (0) {
    0x04 calldataload   // [value]
    [VALUE_LOCATION]    // [ptr, value]
    sstore              // []
    stop
}

#define macro GET_VALUE() = takes (0) returns (0) {
    // Load value from storage.
    [VALUE_LOCATION]   // [ptr]
    sload               // [value]

    // Store value in memory.
    0x00 mstore

    // Return value
    0x20 0x00 return
}

#define macro MAIN() = takes (0) returns (0) {
    // Identify which function is being called.
    0x00 calldataload 0xE0 shr
    dup1 0xb6339418 eq set jumpi
    dup1 0xb63d343f eq get jumpi

    set:
        SET_VALUE()
    get:
        GET_VALUE()
}
```

Really, why the hell would
I ever want to use Huff?

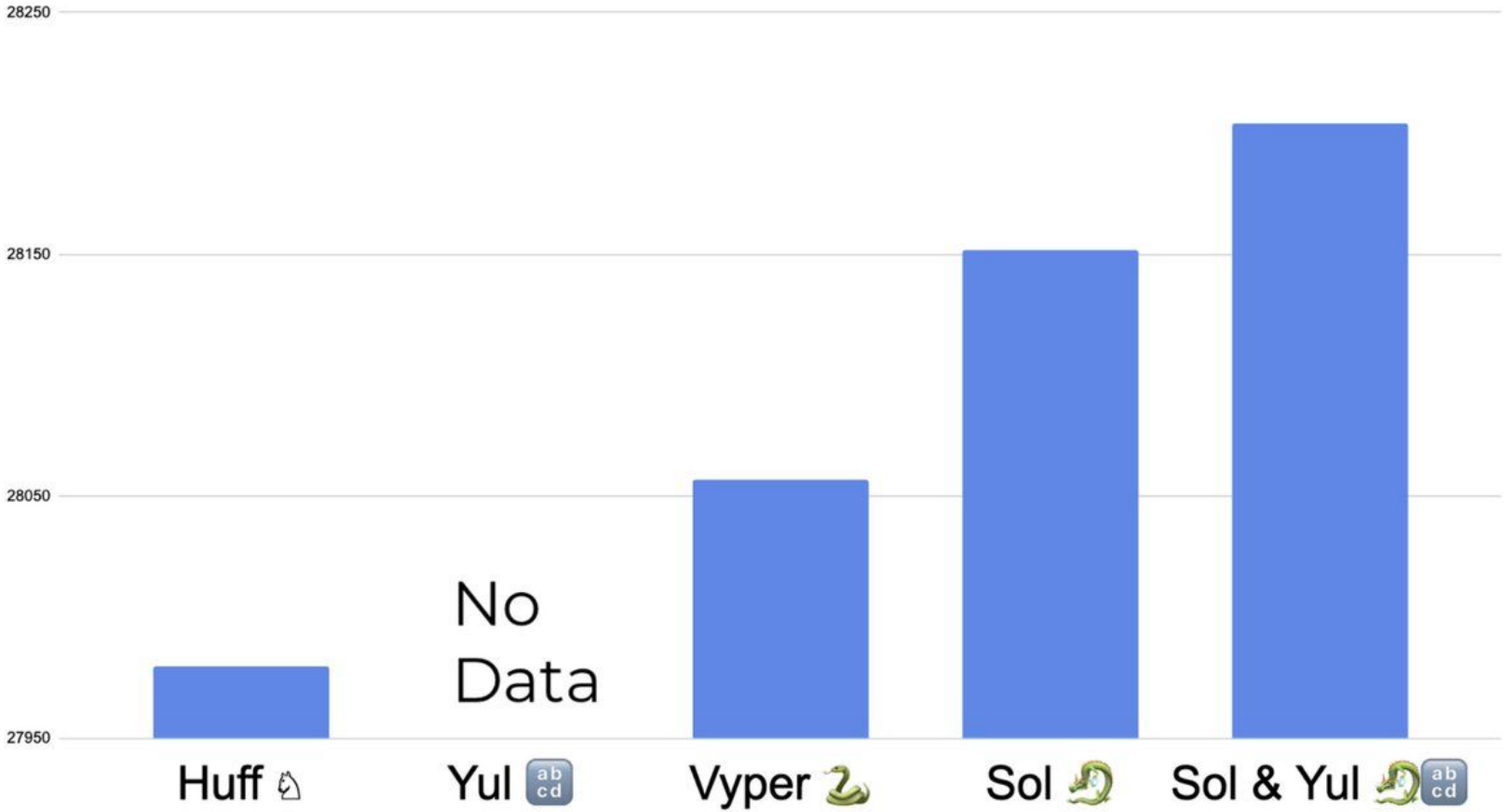
OPCODE	NAME	MINIMUM GAS	STACK INPUT	STACK OUPUT	DESCRIPTION	Expand
00	STOP	0			Halts execution	
01	ADD	3	a b	a + b	Addition operation	
02	MUL	5	a b	a * b	Multiplication operation	
03	SUB	3	a b	a - b	Subtraction operation	
04	DIV	5	a b	a // b	Integer division operation	
05	SDIV	5	a b	a // b	Signed integer division operation (truncated)	
06	MOD	5	a b	a % b	Modulo remainder operation	
07	SMOD	5	a b	a % b	Signed modulo remainder operation	
08	ADDMOD	8	a b N	(a + b) % N	Modulo addition operation	
09	MULMOD	8	a b N	(a * b) % N	Modulo multiplication operation	
0A	EXP	10	a exponent	a ** exponent	Exponential operation	
0B	SIGNEXTEND	5	b x	y	Extend length of two's complement signed integer	
10	LT	3	a b	a < b	Less-than comparison	
11	GT	3	a b	a > b	Greater-than comparison	
12	SLT	3	a b	a < b	Signed less-than comparison	


Message call
EVM code
more gas
stack
memory
program
counter (PC)
operations
gas
available
gas
more gas
(account)
storage

Gas Cost vs. Language - SimpleStorage Contract Creation



Gas Cost vs Language - SimpleStorage Read & Write



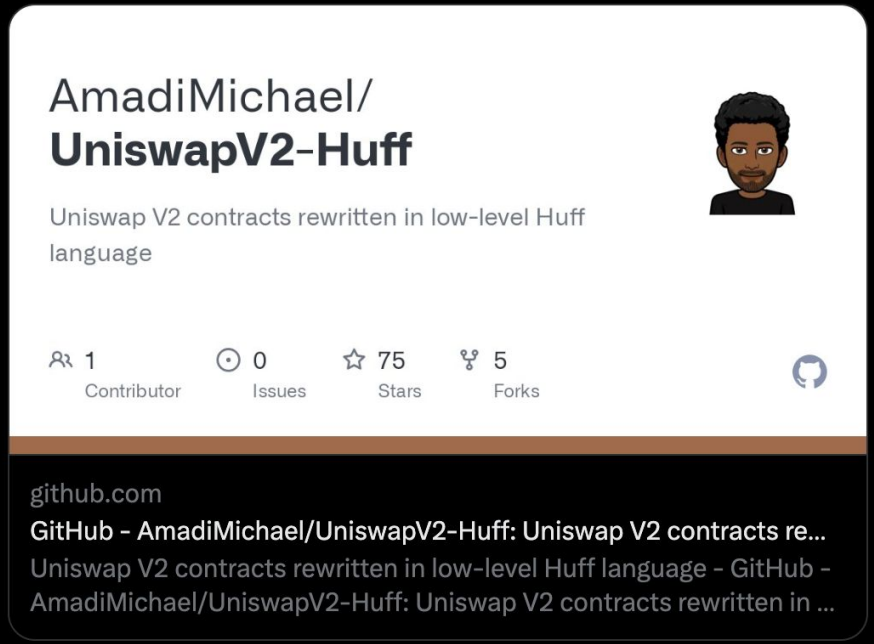
Pinned Tweet

michaelamadi.eth @AmadiMichaels · Feb 28

Took some time and a lot of debugging but it's completed.

Uniswap V2 written in @huff_language

- UniswapV2Pair
- UQ112x112



31 49 319 40.1K

moodle zoup
@moodlezoup

Replying to @jtriley_eth and @Zac_Aztec

it works:


FOUNDRY

huffmate • ci passing version v1.1 license MIT chat 167 online

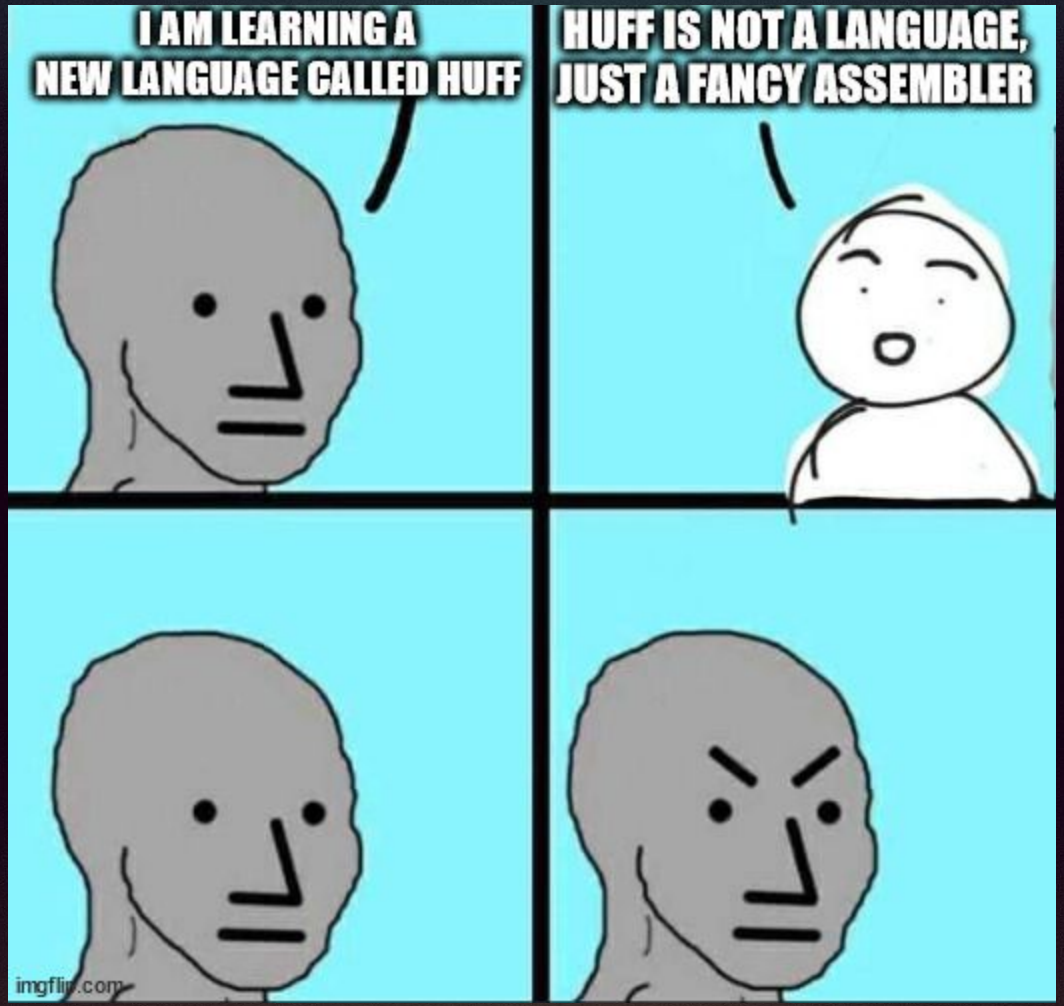
A set of **modern**, **opinionated**, and **secure** Huff contracts.

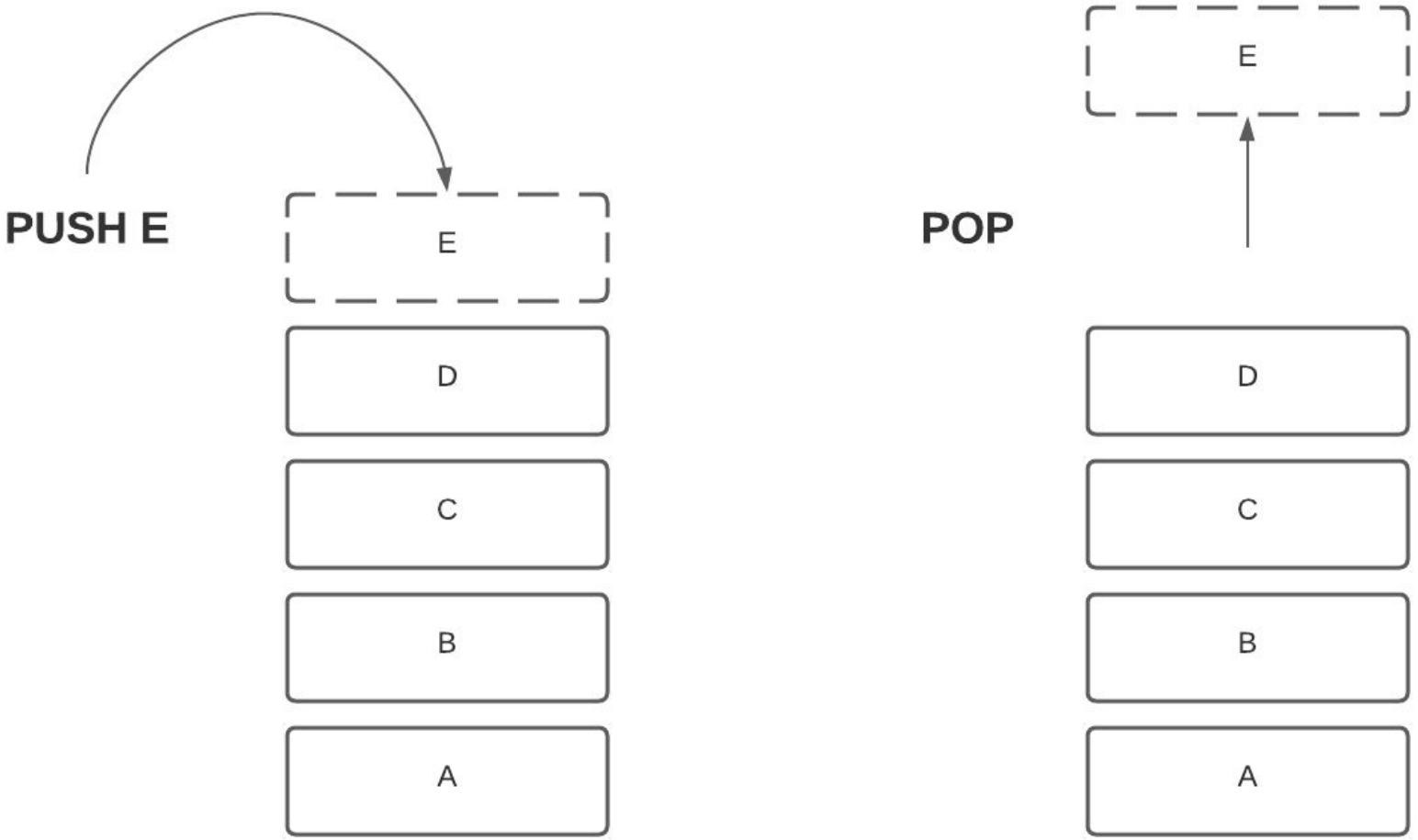
> ⚠ Warning
>
> These contracts are **unaudited** and are not recommended for use in production.
>
> Although contracts have been rigorously reviewed, this is **experimental software** and is provided on an "as is" and "as available" basis. We **do not give any warranties** and **will not be liable for any loss** incurred through any use of this codebase.

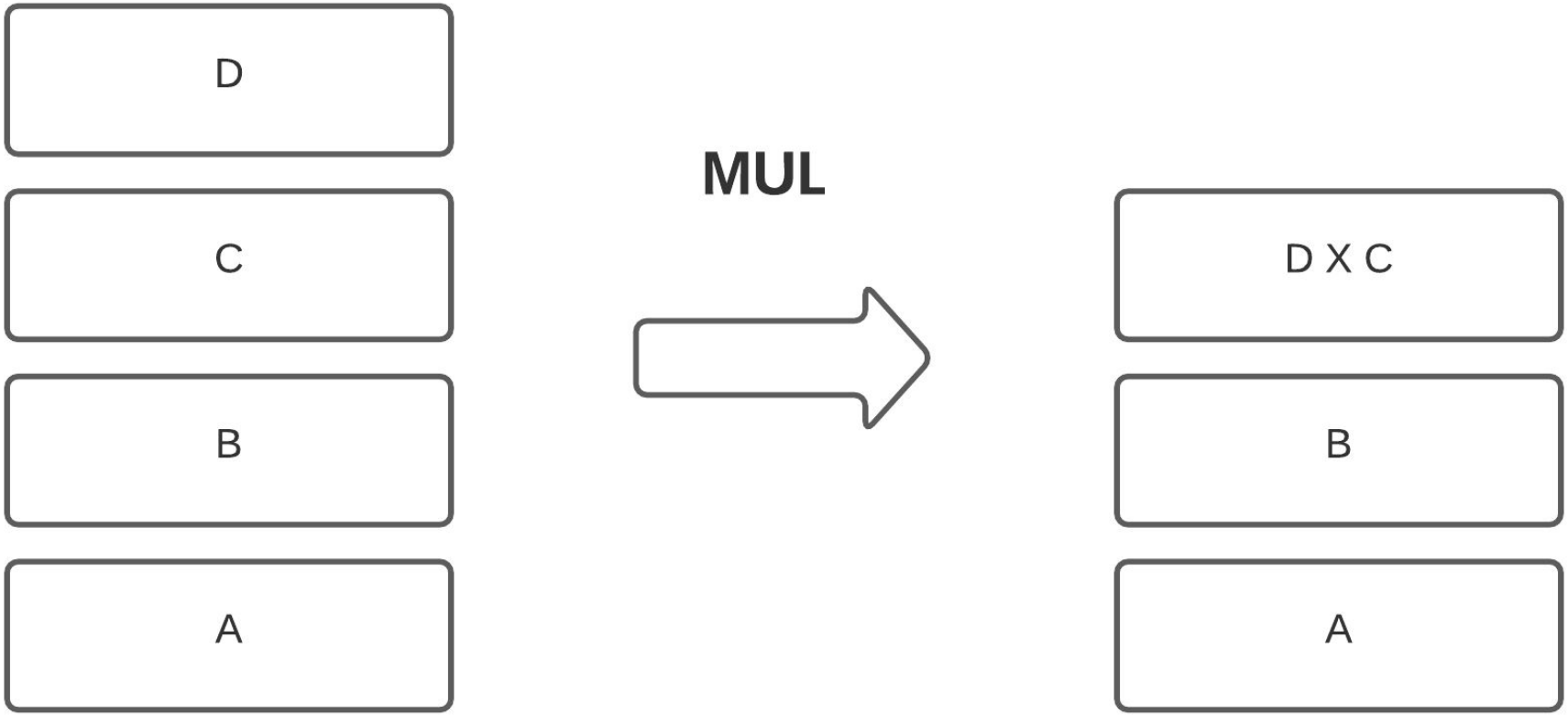
Usage

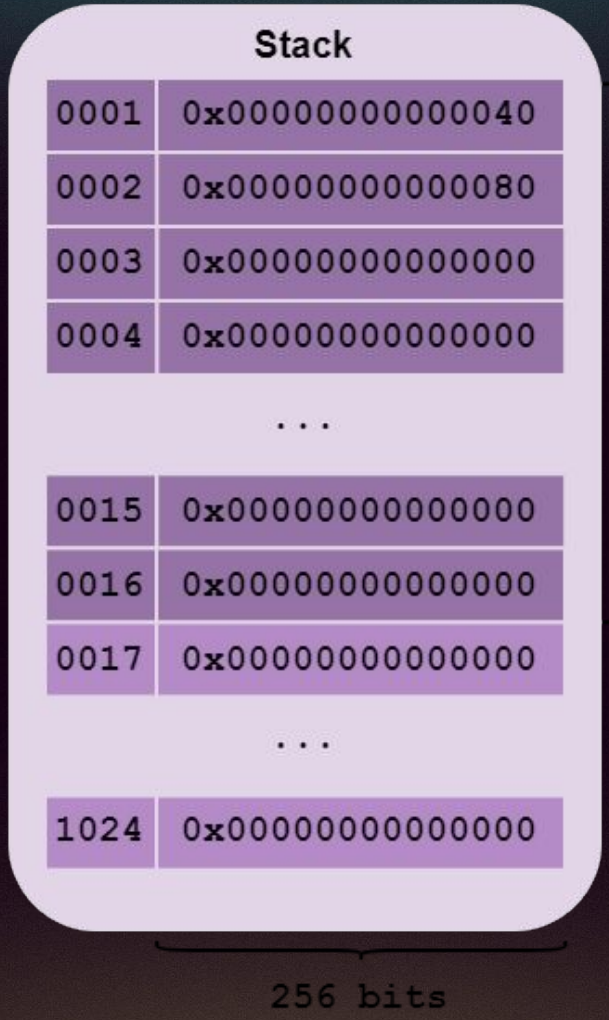
Recommended To install with **Foundry**:

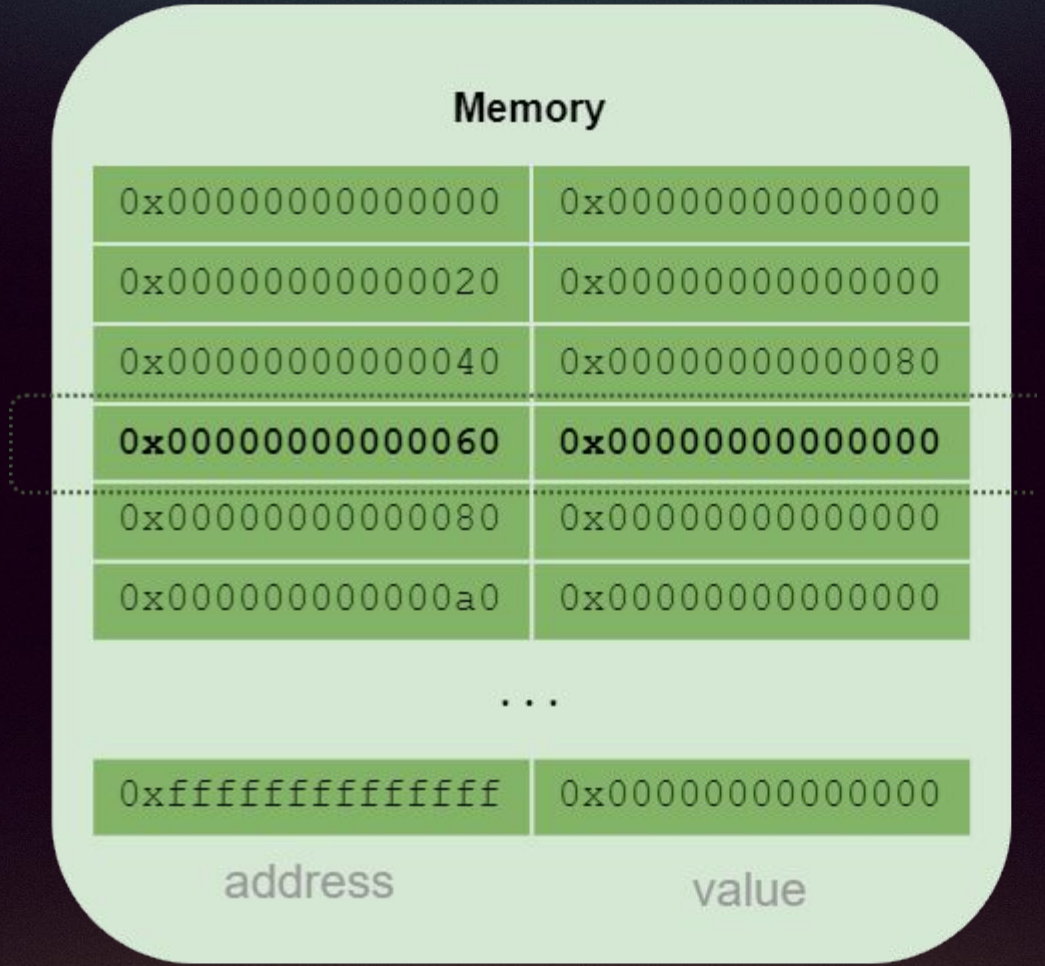
```
forge install pentagonxyz/huffmate
```

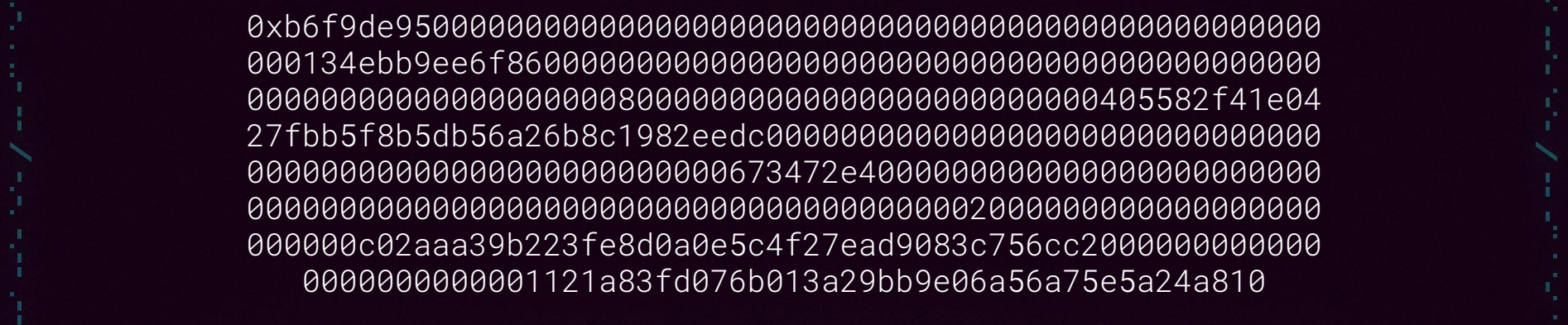

I AM LEARNING A NEW LANGUAGE CALLED HUFF
HUFF IS NOT A LANGUAGE, JUST A FANCY ASSEMBLER
imgflip.com


PUSH E
E
D
C
B
A
POP
E
D
C
B
A


D
C
B
A
MUL
D X C
B
A


Stack
0001 0x00000000000040
0002 0x00000000000080
0003 0x00000000000000
0004 0x00000000000000
...
0015 0x00000000000000
0016 0x00000000000000
0017 0x00000000000000
...
1024 0x00000000000000
256 bits


Memory
0x00000000000000
0x00000000000000
0x00000000000020
0x00000000000000
0x00000000000040
0x00000000000080
0x00000000000060
0x00000000000000
0x00000000000080
0x00000000000000
0x000000000000a0
0x00000000000000
...
0xffffffffffffff
0x00000000000000
address
value


0xb6f9de9500000000000000000000000000000000000000000000000000
000134ebb9ee6f8600000000000000000000000000000000000000000000
0000000000000000000800000000000000000000405582f41e04
27fbb5f8b5db56a26b8c1982eedc0000000000000000000000000000
0000000000000000000000673472e4000000000000000000000000
0000000000000000000000000000000000000200000000000000000
000000c02aaa39b223fe8d0a0e5c4f27ead9083c756cc2000000000000
00000000000001121a83fd076b013a29bb9e06a56a75e5a24a810

EVM Playground CANCUN

Yul

Current: 0 Total: 0

```
object "Contract" {
    // This is the constructor code of the contract.
    code {
        // Deploy the contract
        datacopy(0, dataoffset("runtime"), datasize("runtime"))
        return(0, datasize("runtime"))
    }

    object "runtime" {
        code {
            // Protection against sending Ether
            if gt(callvalue(), 0) {
                revert(0, 0)
            }

            // Dispatcher
            switch selector()
            case 0x6d4ce63c {
                returnUint(get())
            }
            case 0x371303c0 {
                inc()
```

Value to send Wei Run

Loading Solidity compiler soljson-v0.8.27+commit.40a35a09...
Solidity compiler loaded

MEMORY

STACK

STORAGE

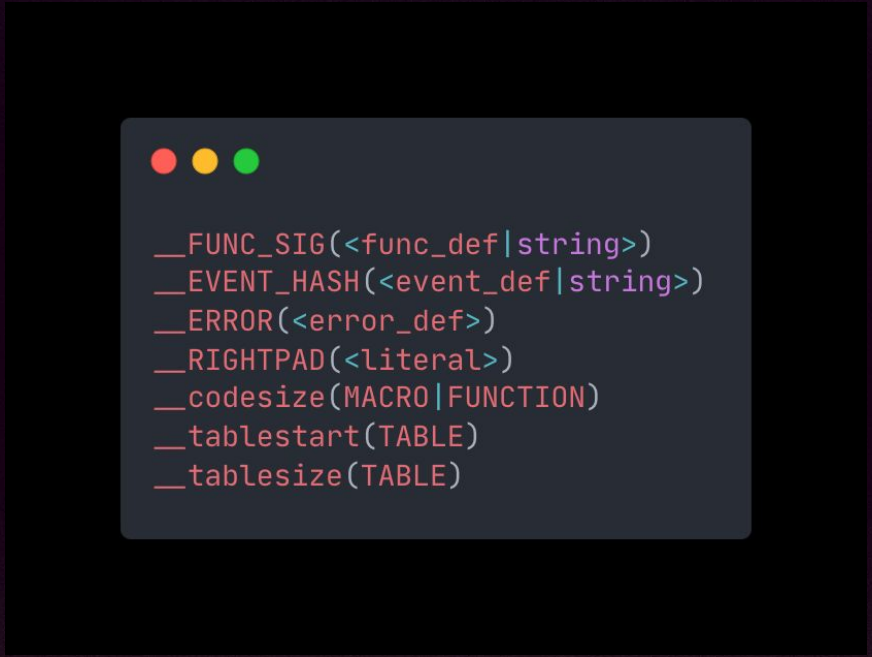
TRANSIENT STORAGE

RETURN VALUE

Interface Definition

```
// Interface
#define function allowance(address,address) view returns (uint256)
#define function approve(address,uint256) nonpayable returns ()
#define function balanceOf(address) view returns (uint256)
#define function DOMAIN_SEPARATOR() view returns (bytes32)
#define function nonces(address) view returns (uint256)
#define function permit(address,address,uint256,uint256,uint8,bytes32,bytes32) nonpayable returns ()
#define function totalSupply() view returns (uint256)
#define function transfer(address,uint256) nonpayable returns ()
#define function transferFrom(address,address,uint256) nonpayable returns ()

// Events
#define event Approval(address indexed, address indexed, uint256)
#define event Transfer(address, address, uint256)

// Metadata
#define function decimals() nonpayable returns (uint8)
#define function name() nonpayable returns (string)
#define function symbol() nonpayable returns (string)
```

Built-in Functions



Custom Errors

```
// Define our custom error
#define error PanicError(uint256)
#define error Error(string)

#define macro PANIC() = takes (1) returns (0) {
    // Input stack:          [panic_code]
    __ERROR(PanicError)    // [panic_error_selector, panic_code]
    0x00 mstore            // [panic_code]
    0x04 mstore            // []
    0x24 0x00 revert
}

#define macro REQUIRE() = takes (3) returns (0) {
    // Input stack:          [condition, message_length, message]
    continue jumpi         // [message_length, message]

    __ERROR(Error)         // [error_selector, message_length,
message_length
    0x00 mstore            // [message_length, message]
    0x20 0x04 mstore       // [message_length, message]
    0x24 mstore            // [message]
    0x44 mstore            // []

    0x64 0x00 revert

    continue:
        pop                // []
}
```

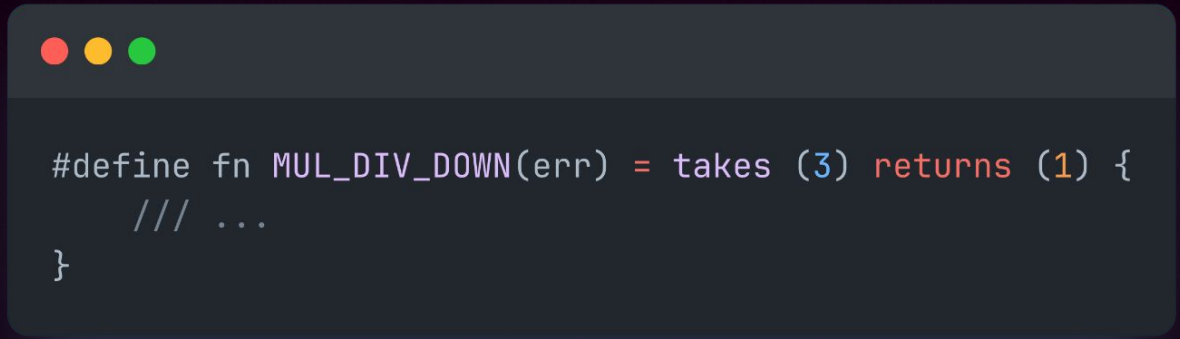
Constants

```
#define constant NUM = 0x420
#define constant HELLO_WORLD = 0x48656c6c6f2c20576f726c6421
#define constant FREE_STORAGE = FREE_STORAGE_POINTER()

...

[NUM] // [0x420] - the constant's value is pushed to the stack
```

Jump labels

```
#define macro MAIN() = takes (0) returns (0) {
    // Store "Hello, World!" in memory
    0x48656c6c6f2c20576f726c6421
    0x00 mstore // ["Hello, World!"]

    // Jump to success label, skipping the revert statement
    success     // [success_label_pc, "Hello, World!"]
    jump        // ["Hello, World!"]

    // Revert if this point is reached
    0x00 0x00 revert

    // Labels are defined within macros or functions, and are designated
    // by a word followed by a colon. Note that while it may appear as if
    // labels are scoped code blocks due to the indentation, they are simply
    // destinations to jump to in the bytecode. If operations exist below a label,
    // they will be executed unless the program counter is altered or execution is
    // halted by a `revert`, `return`, `stop`, or `selfdestruct` opcode.
    success:
        0x00 mstore
        0x20 0x00 return
}
```

Functions

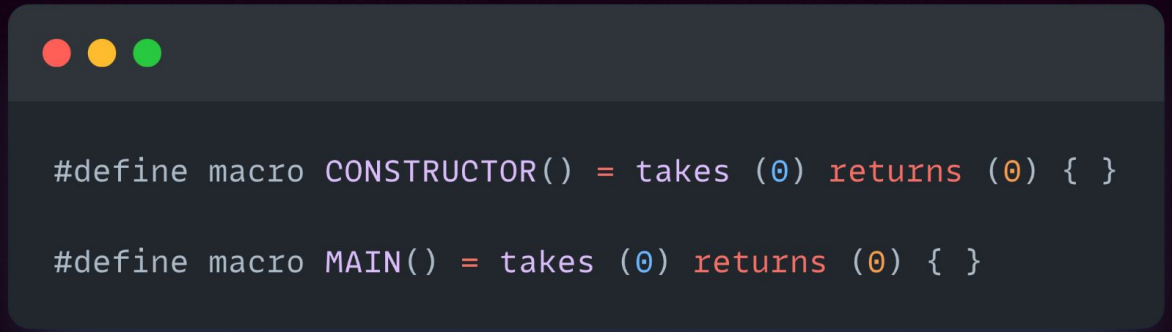


```
#define fn MUL_DIV_DOWN(err) = takes (3) returns (1) {
    /// ...
}
```

Macros

```
#define macro OWNER() = takes (0) returns (0) {
  [OWNER] sload                    // [owner]
  0x00 mstore                      // []
  0x20 0x00 return
}
```

Macros

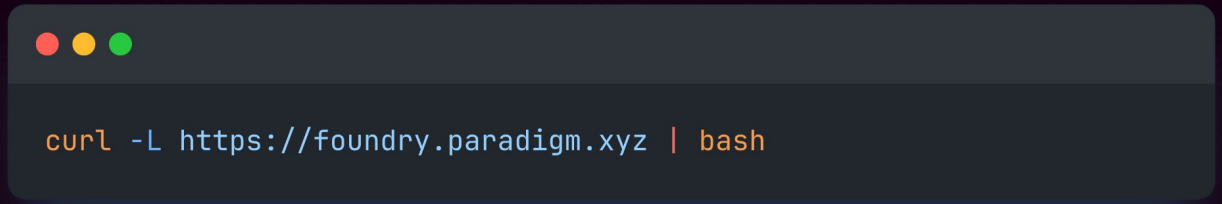


```
#define macro CONSTRUCTOR() = takes (0) returns (0) { }

#define macro MAIN() = takes (0) returns (0) { }
```

Jump Tables

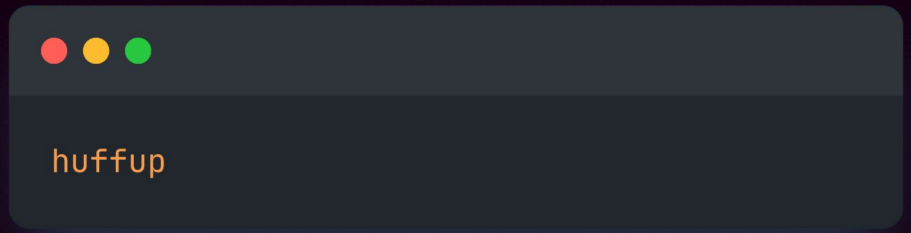
```
// Define a function
#define function switchTest(uint256) pure returns (uint256)

// Define a jump table containing 4 pcs
#define jumptable SWITCH_TABLE {
    jump_one jump_two jump_three jump_four
}

#define macro SWITCH_TEST() = takes (0) returns (0) {
    // Codecopy jump table into memory @ 0x00
    __tablesize(SWITCH_TABLE)   // [table_size]
    __tablestart(SWITCH_TABLE)  // [table_start, table_size]
    0x00
    codecopy

    0x04 calldataload           // [input_num]

    // Revert if input_num is not in the bounds of [0, 3]
    dup1                        // [input_num, input_num]
    0x03 lt                     // [3 < input_num, input_num]
    err jumpi

    // Regular jumptables store the jumpdest PCs as full words,
    // so we simply multiply the input number by 32 to determine
    // which label to jump to.
    0x20 mul                    // [0x20 * input_num]
    mload                       // [pc]
    jump                        // []

    jump_one:
        0x100 0x00 mstore
        0x20 0x00 return
    jump_two:
        0x200 0x00 mstore
        0x20 0x00 return
    jump_three:
        0x300 0x00 mstore
        0x20 0x00 return
    jump_four:
        0x400 0x00 mstore
        0x20 0x00 return
    err:
        0x00 0x00 revert
}

#define macro MAIN() = takes (0) returns (0) {
    // Identify which function is being called.
    0x00 calldataload 0xE0 shr
    dup1 __FUNC_SIG(switchTest) eq switch_test jumpi

    // Revert if no function matches
    0x00 0x00 revert

    switch_test:
        SWITCH_TEST()
}
```

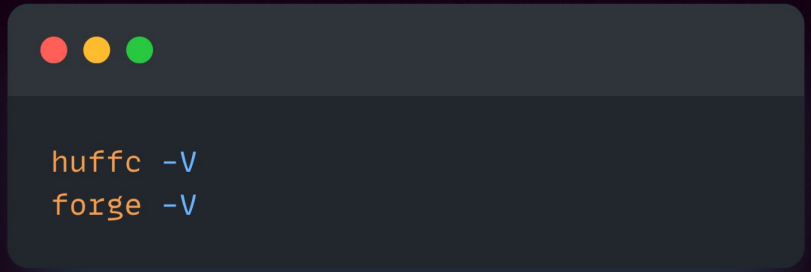
Let's code!

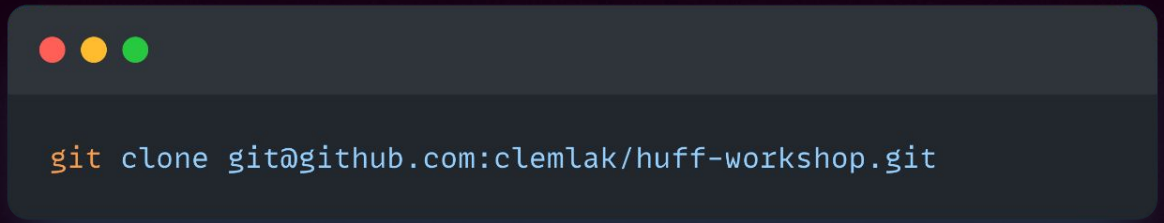

curl -L https://foundry.paradigm.xyz | bash

```
curl -L https://foundry.paradigm.xyz | bash
```

```
curl -L get.huff.sh | bash
```


huffup

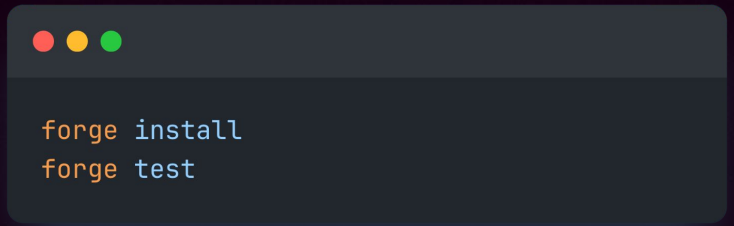

huffc -V
forge -V


git clone git@github.com:clemlak/huff-workshop.git

Time to
cook!

```
#define function compute(uint256,uint256) view returns (uint256)

#define macro COMPUTE() = takes (0) returns (0) {
    0x04 calldataload
    0x24 calldataload
    add
    0x00 mstore
    0x20 0x00 return
}

#define macro MAIN() = takes (0) returns (0) {
    0x00 calldataload 0xE0 shr
    dup1 __FUNC_SIG(compute) eq compute jumpi

    0x00 0x00 revert

    compute:
        COMPUTE()
}
```

```
// SPDX-License-Identifier: Unlicense
pragma solidity ^0.8.15;

import {HuffDeployer} from "foundry-huff/HuffDeployer.sol";
import {Test} from "forge-std/Test.sol";

contract ComputeTest is Test {
    Compute public compute;

    function setUp() public {
        compute = Compute(HuffDeployer.deploy("Compute"));
    }

    function test_compute(uint256 a, uint256 b) public {
        uint256 c = compute.compute(a, b);
        unchecked {
            assertEq(c, a + b);
        }
    }
}

interface Compute {
    function compute(uint256, uint256) external view returns (uint256);
}
```


forge install
forge test

```
/* Interface */
#define function setValue(uint256) nonpayable returns ()
#define function getValue() view returns (uint256)

/* Storage Slots */
#define constant VALUE_LOCATION = FREE_STORAGE_POINTER()

/* Methods */
#define macro SET_VALUE() = takes (0) returns (0) {
    0x04 calldataload   // [value]
    [VALUE_LOCATION]    // [ptr, value]
    sstore              // []
}

#define macro GET_VALUE() = takes (0) returns (0) {
    // Load value from storage.
    [VALUE_LOCATION]   // [ptr]
    sload               // [value]

    // Store value in memory.
    0x00 mstore

    // Return value
    0x20 0x00 return
}

#define macro MAIN() = takes (0) returns (0) {
    // Identify which function is being called.
    0x00 calldataload 0xE0 shr
    dup1 __FUNC_SIG(setValue) eq set jumpi
    dup1 __FUNC_SIG(getValue) eq get jumpi

    0x00 0x00 revert

    set:
        SET_VALUE()
    get:
        GET_VALUE()

}
```

```
// SPDX-License-Identifier: Unlicense
pragma solidity ^0.8.15;

import "foundry-huff/HuffDeployer.sol";
import "forge-std/Test.sol";
import "forge-std/console.sol";

contract SimpleStoreTest is Test {
    /// @dev Address of the SimpleStore contract.
    SimpleStore public simpleStore;

    /// @dev Setup the testing environment.
    function setUp() public {
        simpleStore = SimpleStore(HuffDeployer.deploy("SimpleStore"));
    }

    /// @dev Ensure that you can set and get the value.
    function testSetAndGetValue(uint256 value) public {
        simpleStore.setValue(value);
        console.log(value);
        console.log(simpleStore.getValue());
        assertEq(value, simpleStore.getValue());
    }
}

interface SimpleStore {
    function setValue(uint256) external;
    function getValue() external returns (uint256);
}
```

Thank you!

Clément Lakhal

@huff_language

huff.sh