ethdebug/format
bringing modern debugging to ethereum

g. nicholas d’andrea // @gnidan
argot collective

debugging is hard because compilers must be smart

Especially on smart contract platforms, where bad performance costs money.

Compilers must have good optimizers, but optimizers obfuscate intent.

Debugging optimized code requires guessing from a world of multiple possibilities.

ethdebug makes good debugging less expensive (also possible :)

The format presents EVM code from the lens of ambiguous machine instructions.

Compilers output *accurate* contextual information about each instruction with various levels of precision (based on what optimization permits)

Debuggers evaluate execution and disambiguate based on observation of runtime.

fledgling ecosystem support

implementation progress

Solidity implementation is in progress, minimal experimental output since 0.8.29 (via feature flag).

Things are looking quite positive on the debugger side: Tenderly, Simbolik, and Walnut are all working on support.

Walnut is also building frameworks for evaluating compilers' ethdebug-compliance and looking at backwards compatibility solutions for older solc.

design approach

ethdebug/format is reference-implementation forward

Debugging is hard. Debugging is still hard even with good debugging data and compiler support.

To help make implementation easier, we are implementing this format while designing it.

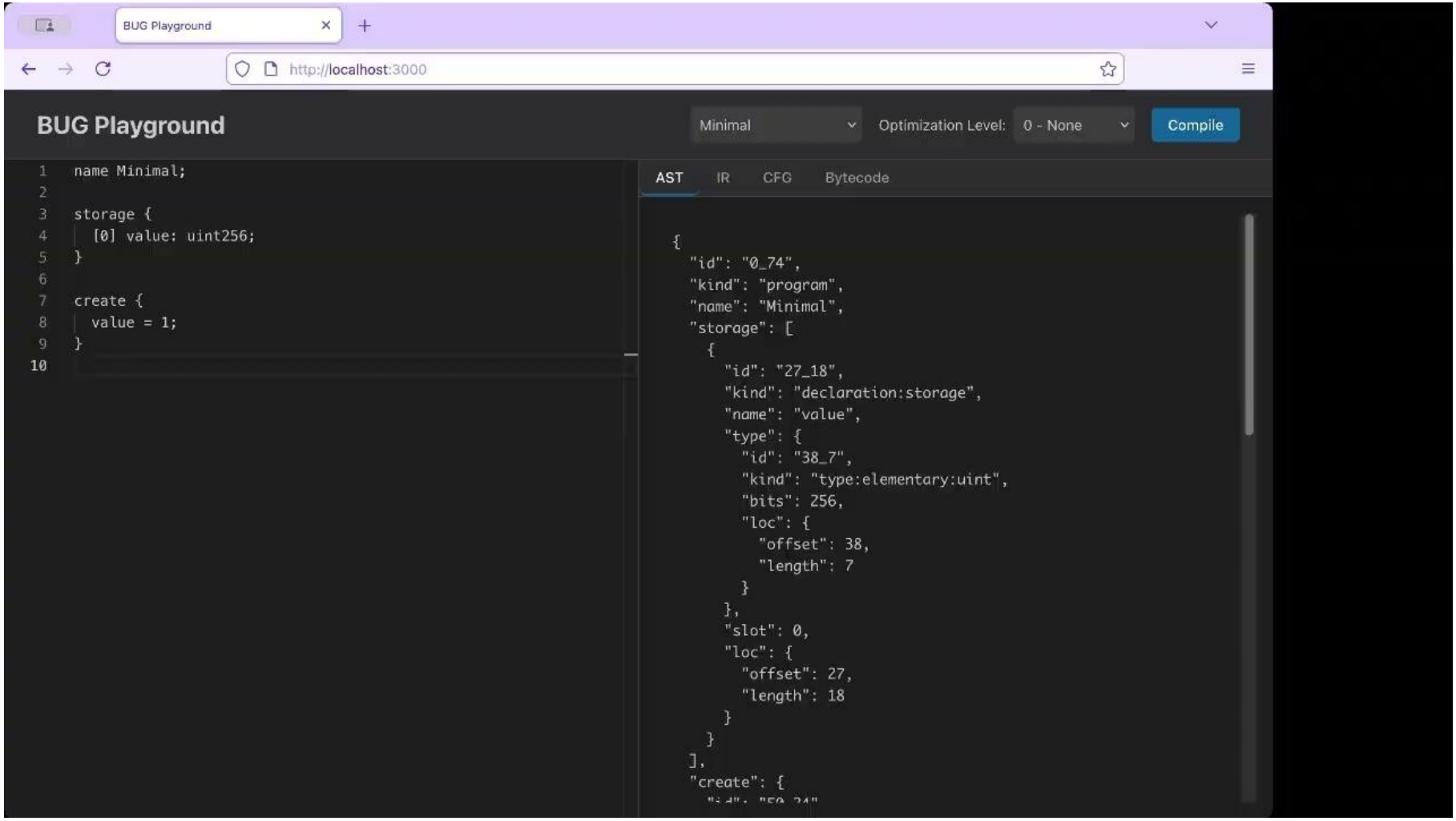
We've made a couple libraries to show debugger-side support for ethdebug, and have built a "toy compiler" for a fictional language to produce ethdebug data end-to-end.

Of course, reference implementations also help validate the format itself

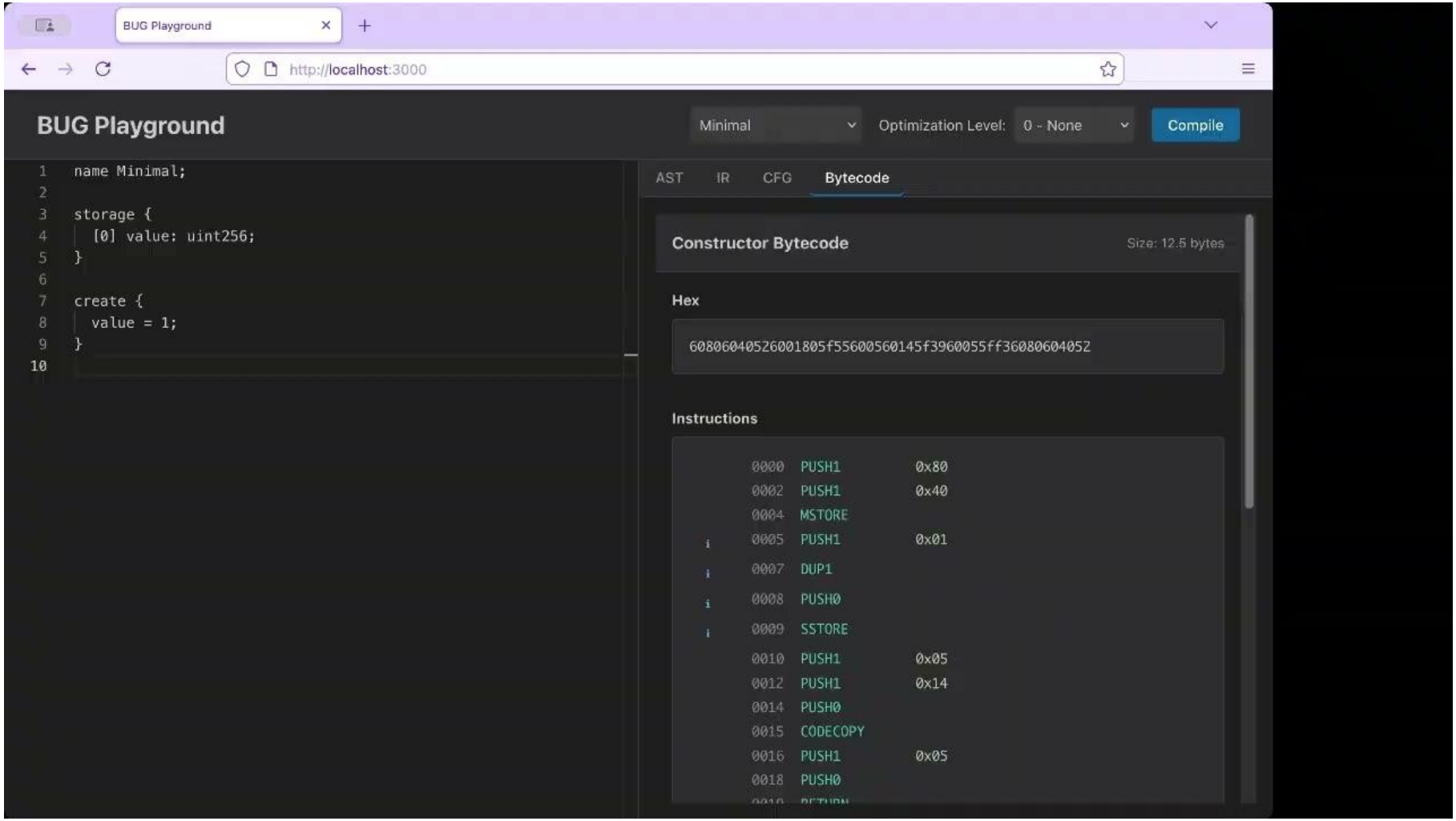
demos

Example ethdebug program (static)

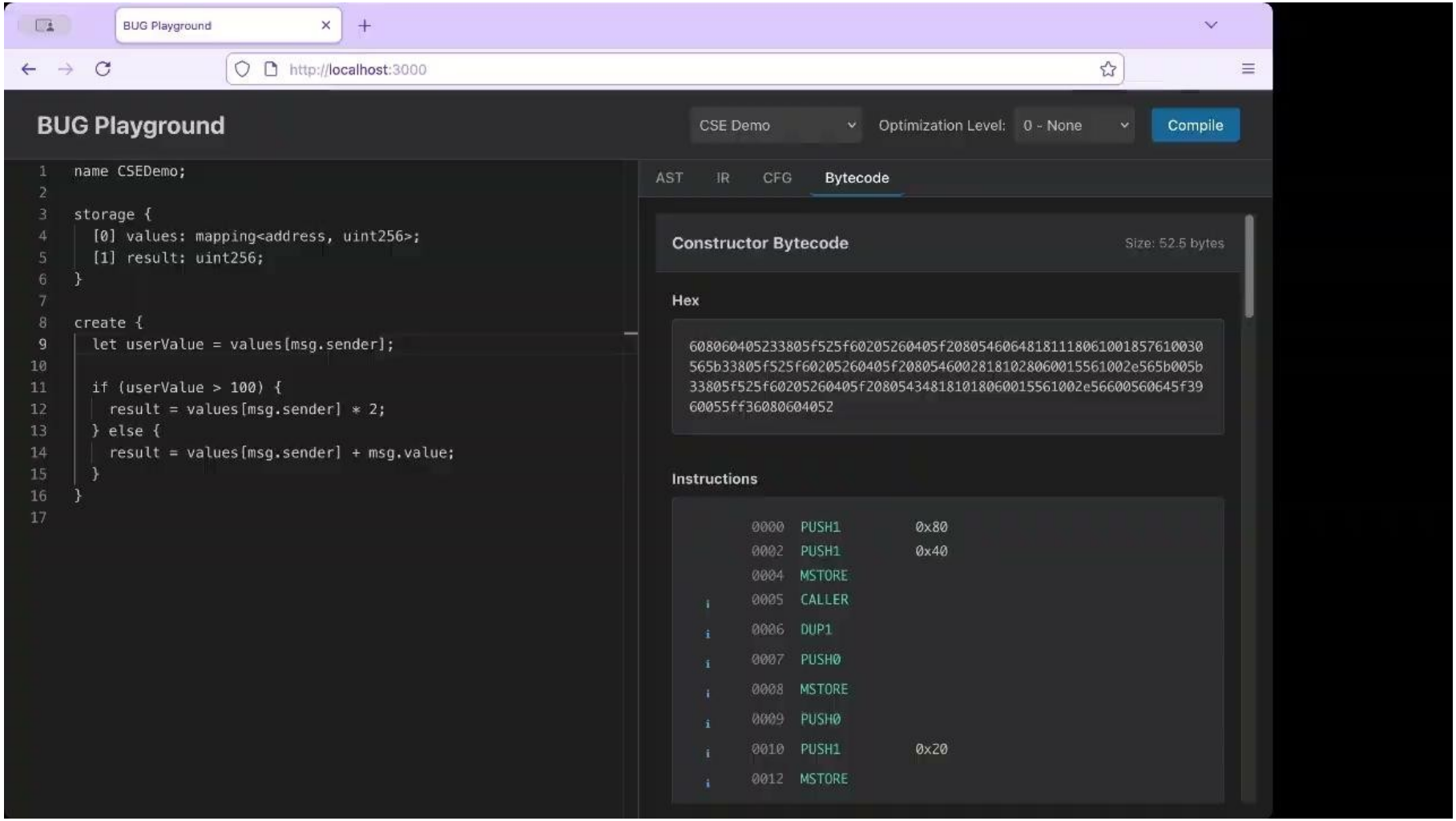
BUG demo (Minimal.bug)


BUG Playground
http://localhost:3000
BUG Playground
Minimal
Optimization Level:
0 - None
Compile
name Minimal;
storage {
[0] value: uint256;
}
create {
value = 1;
}
AST
IR
CFG
Bytecode
{
"id": "0_74",
"kind": "program",
"name": "Minimal",
"storage": [
{
"id": "27_18",
"kind": "declaration:storage",
"name": "value",
"type": {
"id": "38_7",
"kind": "type:elementary:uint",
"bits": 256,
"loc": {
"offset": 38,
"length": 7
}
},
"slot": 0,
"loc": {
"offset": 27,
"length": 18
}
}
],
"create": {

Common subexpression elimination (Optimizer off demo)


BUG Playground
http://localhost:3000
BUG Playground
Minimal
Optimization Level:
0 - None
Compile
1 name Minimal;
2
3 storage {
4 [0] value: uint256;
5 }
6
7 create {
8 value = 1;
9 }
10
AST
IR
CFG
Bytecode
Constructor Bytecode
Size: 12.5 bytes
Hex
60806040526001805f55600560145f3960055ff36080604052
Instructions
0000 PUSH1 0x80
0002 PUSH1 0x40
0004 MSTORE
0005 PUSH1 0x01
0007 DUP1
0008 PUSH0
0009 SSTORE
0010 PUSH1 0x05
0012 PUSH1 0x14
0014 PUSH0
0015 CODECOPY
0016 PUSH1 0x05
0018 PUSH0

Common subexpression elimination (Optimizer enabled)


BUG Playground
http://localhost:3000
BUG Playground
CSE Demo
Optimization Level:
0 - None
Compile
1 name CSEDemo;
2
3 storage {
4 [0] values: mapping<address, uint256>;
5 [1] result: uint256;
6 }
7
8 create {
9 let userValue = values[msg.sender];
10
11 if (userValue > 100) {
12 result = values[msg.sender] * 2;
13 } else {
14 result = values[msg.sender] + msg.value;
15 }
16 }
17
AST
IR
CFG
Bytecode
Constructor Bytecode
Size: 52.5 bytes
Hex
6080604052233805f525f60205260405f20805460648181118061001857610030
565b33805f525f60205260405f20805460028181028060015561002e565b005b
33805f525f60205260405f20805434818101806001556100e56600560645f39
60055ff36080604052
Instructions
0000 PUSH1 0x80
0002 PUSH1 0x40
0004 MSTORE
0005 CALLER
0006 DUP1
0007 PUSH0
0008 MSTORE
0009 PUSH0
0010 PUSH1 0x20
0012 MSTORE

closing remarks

ethdebug/format is finally not just theoretical

Every year, I present on a “hypothetical” ethdebug format.

This year, thanks to the community, things are becoming much more concrete.

Please stay tuned for much more soon!

thank you!

more information:

https://ethdebug.github.io/format

https://github.com/ethdebug/format

https://github.com/ethdebug/bugc

(shoutout to https://github.com/walnuthq for github.com/walnuthq/ethdebug-stats and more)