ETK: Ethereum Object Format (EOF) implementation

Gaston Ezequiel Zanitti
Ethereum Protocol Fellowship

ethereum foundation

fellowship program

Section 1

My journey through the EPF

My journey through the EPF

Mainly interested in compilers and virtual machines.

Contacted the **Ipsilon team** and started with some warm-up work:

- Implemented **MCOPY (EIP-5656)** in **Huff** (merged!)
- Implemented **MCOPY (EIP-5656)** in **ETK** (merged!)
- Small app to run **EOF** tests in **evmone** (merged!)

Research on possible features/improvements for an EVM-targeted PL:

- **Assets** and **Caller capabilities** from **Flint**
- **Typestate** from **Obsidian**

My journey through the EPF

Two possible scenarios for my time at the Ethereum Protocol Fellowship:

1) Further investigate some ideas and implement a **PoC of a PL targeting the EVM**
2) **Implement EOF** in some compiler in order to gain a deeper understanding of the EVM

I contacted Mario and we talked about it:

- A language, even a PoC, was going to be a very big task for one person and four months.
- Decided to work on a **Rust-based compiler**, to gain experience in this language as well.

Section 2

Project description

Project description

This project aims to implement the **Ethereum Object Format (EOF)** specification within the **ETK language.**

Why I think EOF is important?

- Introduces a **versioned container format for EVM bytecode** which offers a **mechanism for managing breaking changes**.
- A series of **improvements for language developers**
- Lately, becoming more **controversial**

Specification:

EIP-3540: EVM Object format v1

EIP-3670: Code validation

EIP-4200: Static Relative jumps

EIP-4750: Functions

EIP-5450: Stack validation

EIP-6206: JUMPF instruction

EIP-7480: Data section access instructions

EIP-663: Unlimited SWAP & DUP instructions

EIP-7069: Revamped CALL instruction

Project description

Obviously, I started by implementing some EIPs:

- **EIP-663: Unlimited SWAP and DUP instructions**
- **EIP-4200: Static relative jumps**
- **EIP-6206: JUMPF instruction**

But, when implementing **EIP-4750: Functions** there first main issue arise:

- ETK doesn’t have a way to deprecate/rename opcodes. Always the last hardfork was used.
- Conversations with Lightclient and Sam Wilson about this: **We should simplify the backend first**

```
% push(Label1)   <---  DYNAMIC PUSH: BYTECODE IN "PENDING STATE"
sload
swap1
add
%% user_macro ()   <---  NEW "PENDING STATE" IS NEEDED error

Label 1:   <---  FIRST "PENDING STATE" IS CLOSED
swap 2
% macro user_macro ()
    push1 0
% end
```

Backend simplification

- %push(...) decide size at compile time
- all compiled bytecode is temporarily stored in a separate array
- calling a macro needs the backend to have zero pending bytecodes.
- after a dynamic push, only happens when the label is defined.

Hardfork selection

- To choose a hardfork it was necessary to modify and recompile ETK.
- Changes to be able to choose hardfork by flag: ***eas --hardfork london***
- Introduce a new macro **%hardfork(…)** to ensure that the code is compiled in the correct hardfork range.

Essential modifications to implement EOF

Project description (status)

- **Backend simplification** is being reviewed by Sam
  - We found a small bug when combining dynamic push and expressions: **%push(label + 512)**
- **Hardfork selection** goes on top of the backend simplification. Still on hold.
- In the meantime:
  - Solving old issues: **#124**, **#108**, **#82**
  - Implementing some long-overdue features: Issue **#106**
- Proposal to create a **website with a small playground**. I will work on this when I finish with the EOF implementation.

Section 3

Future of the project

Future of the project

- More than 7 PR’s waiting to be merged. Backend simplification is the main obstacle.

Next week

Merge the code on hold & finish EOF implementation

Next month

Work on website, ETK binding to JS and playground

Future?

Would love to continue cooperating in the development of ETK.

Lot of work to do!

Section 4

Self-evaluation & Feedback

Self evaluation & Feedback

- Excellent experience.
- Improved my knowledge of Rust & the EVM.

The good thing:

- The possibility to choose a project and work independently.
- Incredibly knowledgeable people willing to help

The bad:

- Totally understandable, but the back and forth with mentors is sometimes a bit slow.

Thank you!

Gaston Ezequiel Zanitti
Ethereum Protocol Fellowship - 4th cohort
gzanitti@gmail.com

@gzanitti