Off-White Paper

An Anchor Specification for Ethereum

Micah Dameron

November 17, 2017

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.

The Zen of Python

Abstract

The goal of this paper[a] is threefold. First, we aim to create and expand concepts from the Yellowpaper that the Yellowpaper omits or fails to isolate from the reader in sufficient detail. Second, we propose a formally correct pseudocode to supplement the Yellowpaper's formal math. Finally, we give an open-source specification for Ethereum that is approachable, direct, and no-nonsense in style.

[a]Formally, *Blanched-Almond Paper*

Acknowledgements

Thank you to the Ethereum founders for creating a product worth writing about in minute detail. Thanks to the Ethereum Foundation for maintaining this product in its basic integrity. Thanks to the ConsenSys Mesh for supporting my work on this project, and for contributing your vast knowledge and expertise toward a good finished project. Finally, thank you to Dr. Gavin Wood for your technical astuteness and creative genius; your Yellowpaper has given Ethereum a soul worth decoding.

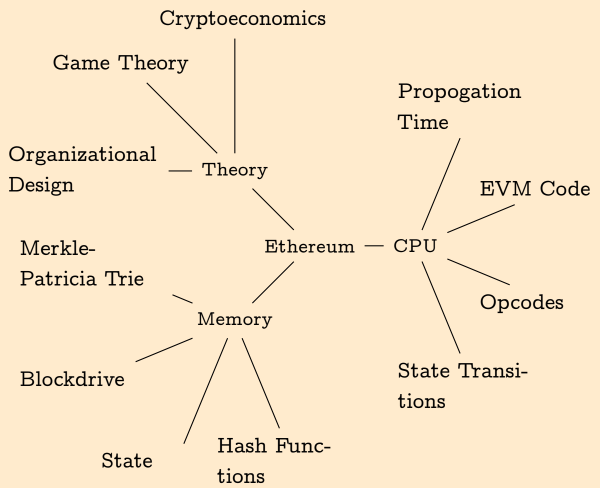
Contents

Part I. Theory and Practice

1 The Evolution of a Protocol

Over the past decade, blockchain technology has proven its longevity and veracity through a number of systems, most notably through Bitcoin, the first electronic currency of its kind to succeed. Bitcoin was successful in its mission to create currency based on a decentralized peer-to-peer Blockchain protocol, and Ethereum takes that concept a step further by creating a *globally-distributed virtual machine*that can ad-hoc run such currency applications, along with any other conceivable applications or programs. Using well-established concepts from the relevant areas of computer science, like MESSAGE-PASSING, TRANSACTION PROCESSING, and SHARED-STATE CONCURRENCE, the *Ethereum Protocol* creates an environment for developers to execute machine instructions with the same level of veracity and certainty as monetary transactions have on more standard Blockchains.

A Conceptual Map of the Ethereum Protocol



2 Data Structures

3 Serialization

4 Hacker Ethic

5 Message-Passing Interface

6 Singleton Pattern

7 State Machines

8 Processor Technology

9 Turing Machines

9.1 Turing-Completeness

10 Intrinsic Currency

The smallest unit of currency in Ethereum is the Wei, which is $1 * 10^(18)$ Ether. All units at the machine level are counted in Wei.

11 Computation Flow

12 ECDSA

12.1 Public Key Cryptography

Diffie Helman hypothesis–stated in their paper (citation to it)

Public Keys

Private Keys

13 Programming Languages

13.1 Lower-Level Lisp

and this is what turns your LLL code into EVM code and how it's executed in the evm

[a] A database backend is accessed by users indirectly through an external application, most likely an Ethereum client; see also: state database

[b] A bytearray is specific set of bytes [data] that can be loaded into memory. It is a structure for storing binary data, e.g. the contents of a file.

[c] This permanent data structure makes it possible to easily recall any previous state with its root hash keeping the resources off-chain and minimizing on-chain storage needs.

[d] Formally **world state** $= \sigma$.

[a] σ is the world state at a certain given time, and n is the number of transactions or contract creations by that account.

[b] A particular path from root to leaf in the **state database** that encodes the STORAGE CONTENTS of the account.

[c] A message call is any interaction with the account on-chain.

[d] Formal notation is $\sigma[a]_c$

[e] More formal notation is $\texttt{KEC}(\mathbf{b}) = \boldsymbol{\sigma}[a]_c$

[f] This formal notation is $\sigma[a]_b$

13.2 Solidity

and this is what turns your solidity code into EVM code and how it's executed in the EVM

Part II. Memory and Storage

14 World State

Also known simply as "state", this is a MAPPING of **addresses** and **account states** (RLP data structures), this is also known as *state*, or σ. This mapping is not stored on the blockchain, rather it is stored as a Merkle-Patricia trie in a DATABASE BACKEND[a] that maintains a mapping of bytearrays to bytearrays.[b] The cryptographic internal data going back to the root node represents the *State* of the Blockchain at any given root, i.e. at any given *time*.[c]As a whole, the state is the sum total of database relationships in the **state database**. The state is an inert position on the chain, a position between prior state and post state; a block's frame of reference, and a defined set of relationships to that frame of reference.

15 Data Structures

15.1 Byte Arrays

15.2 Bit Sequences

Message Calls are either bit sequences or byte arrays.

15.3 The Block

15.4 State Database

15.5 Merkle-Patricia Trees

RLP

Well-Formed RLP

Account State

The account state is the state of any particular account during some specified world state σ.[d]

Nonce The **nonce** aspect of an ACCOUNT'S STATE is the number of transactions sent from, or the number of contract-creations by, the address of that account.[a]

Storage Root The **storage root** aspect of an ACCOUNT'S STATE is the hash of the trie[b]

Code Hash The **code hash** aspect of an ACCOUNT'S STATE is the HASH OF THE EVM CODE of this account. Code hashes are STORED in the **state database**. Code hashes are permanent and they are executed when the address belonging to that account RECEIVES a message call.[cde]

Balance The amount of **Wei** OWNED by this account. [f]

15.6 Transaction Receipts

Bloom Filter

Part III. Processing and Computation

16 State Transition Function

State Transitions come about through a what is known as the State Transition Function; this is an abstraction of several operations in Ethereum which comprise the overall act of computing changes to the *machine state* prior to adding them to the *world state*, that is, through them being finalized and rewards applied to a given miner. `apply_rewards` and `block_beneficiary` are here.

16.1 Mining

Ethash

GHOST Protocol

16.2 Verification

Verifies Ommer headers

Ommers

Ommershash

Is-Sibling Property

16.3 Transactions

Transactions are the bread and butter of state transitions, that is of block additions, which are all the computation performed in one block. Each transaction applies the execution changes to the *machine state*, a temporary state which consists of all the temporary changes in computation that must be made before a block is finalized and added to the world state.

16.4 Execution

Execution Model

Message Calls

Contract Creation

Account Creation

16.5 Halting

Execution Environment

16.6 Gas

Miner Choice Miners choose which gas prices they want to accept.

Gasprice

Gaslimit Any unused gas is refunded to the user.

Gasused

Machine State

Substate The `substate` is an emergent, ever-changing ball of computational energy that is about to be applied to the main state. It is the *meta state* by which transactions are decided valid and to be added to the blockchain.

EVM Code

Opcodes

But what exactly are these computer instructions that can be executed with the same level of veracity and certainty as Bitcoin transactions? How do they come about, what makes them up, how are they kept in order, and what makes them execute? The first part of answering this question is understanding opcodes. In traditional machine architectures, you may not be introduced to working with processor-level assembly instructions for some time. In Ethereum however, they are essential to understanding the protocol because they are the most minute and subtle (yet HUGELY important) things going on in the Ethereum Blockchain at any moment, and they are the real "currency," that Ethereum trades in. I'll explain what I mean by that in a minute. First, let's

go over a few Opcodes:[a]

Data	Opcode	Gas	In	Out
0x00	STOP	0	2	1
0x01	ADD	3	2	1
0x02	MUL	5	2	1
0x03	SUB	3	2	1
0x04	DIV	5	2	1

The STOP Opcode is used in order to stop a computation once it has completed, or to halt a computation if it has run out of gas. The ADD, MUL, SUB, and DIV operations are addition, multiplication, subtraction and division operations. The In/Out columns refer to inputs (to `machine_state`), the state which decides every new `world_state`.

Part IV.
Appendix

A Opcodes

Data	Opcode	Gas	Input	Output
0x00	STOP	0	0	0
0x01	ADD	3	2	1
0x02	MUL	5	2	1
0x03	SUB	3	2	1
0x04	DIV	5	2	1
0x05	SDIV	5	2	1
0x06	MOD	5	2	1
0x07	SMOD	5	2	1
0x08	ADDMOD	8	3	1
0x09	MULMOD	8	3	1
0x0a	EXP	10	2	1
0x0b	SIGNEXTEND	5	2	1
0x10	LT	3	2	1
0x11	GT	3	2	1
0x12	SLT	3	2	1
0x13	SGT	3	2	1
0x14	EQ	3	2	1
0x15	ISZERO	3	1	1
0x16	AND	3	2	1
0x17	OR	3	2	1
0x18	XOR	3	2	1
0x19	NOT	3	1	1
0x1a	BYTE	3	2	1
0x20	SHA3	30	2	1
0x30	ADDRESS	2	0	1
0x31	BALANCE	400	1	1
0x32	ORIGIN	2	0	1
0x33	CALLER	2	0	1
0x34	CALLVALUE	2	0	1
0x35	CALLDATALOAD	3	1	1
0x36	CALLDATASIZE	2	0	1
0x37	CALLDATACOPY	3	3	0
0x38	CODESIZE	2	0	1
0x39	CODECOPY	3	3	0
0x3a	GASPRICE	2	0	1
0x3b	EXTCODESIZE	700	1	1
0x3c	EXTCODECOPY	700	4	0

0x3d	RETURNDATASIZE	2	0	1
0x3e	RETURNDATACOPY	3	3	0
0x40	BLOCKHASH	20	1	1
0x41	COINBASE	2	0	1
0x42	TIMESTAMP	2	0	1
0x43	NUMBER	2	0	1
0x44	DIFFICULTY	2	0	1
0x45	GASLIMIT	2	0	1
0x50	POP	2	1	0
0x51	MLOAD	3	1	1
0x52	MSTORE	3	2	0
0x53	MSTORE8	3	2	0
0x54	SLOAD	200	1	1
0x55	SSTORE	0	2	0
0x56	JUMP	8	1	0
0x57	JUMPI	10	2	0
0x58	PC	2	0	1
0x59	MSIZE	2	0	1
0x5a	GAS	2	0	1
0x5b	JUMPDEST	1	0	0
0xa0	LOG0	375	2	0
0xa1	LOG1	750	3	0
0xa2	LOG2	1125	4	0
0xa3	LOG3	1500	5	0
0xa4	LOG4	1875	6	0
0xf0	CREATE	32000	3	1
0xf1	CALL	700	7	1
0xf2	CALLCODE	700	7	1
0xf3	RETURN	0	2	0
0xf4	DELEGATECALL	700	6	1
0xf5	CALLBLACKBOX	40	7	1
0xfa	STATICCALL	40	6	1
0xfd	REVERT	0	2	0
0xff	SUICIDE	5000	1	1

References

[7] D. G. Wood, *Ethereum: A secure decentralised generalised transaction ledger*, https://github.com/ethereum/yellowpaper, 2017 (cit. on p. 17).

[8] *I am a developer and i don't understand the yellow paper. • r/ethereum*. [Online]. Available: https://www.reddit.com/r/ethereum/comments/4bbnp4/i_am_a_developer_and_i_dont_understand_the_yellow/ (cit. on p. 17).

Glossary

abstract machine An abstract machine is a conceptual model of a computer that describes its own operations with perfect accuracy. Since abstract machines are theoretical, all possible outputs can be determined beforehand. 15

Address A 160-bit (20-byte) code used for identifying Accounts. 15

addresses 160-bit (20-byte) identifiers–the last 20 characters of an Ethereum address. 15

Autonomous Object A notional object existent only within the hypothetical state of Ethereum. Has an intrinsic address and thus an associated account; the account will have non-empty associated EVM Code. Incorporated only as the Storage State of that account. 15

Balance A value which is intrinsic to accounts; the quantity of Wei in the account. All EVM operations are associated with changes in account balance. 15

beneficiary The 160-bit address to which all fees collected from the successful mining of this block be transferred. 15

Bit The smallest unit of electronic data storage: there are eight bits in one byte. The Yellowpaper gives certain values in bits (e.g. 160 bits instead of 20 bytes). 15

block header The information in a block besides transaction information. It consists of a dozen parts: (lists the 12 parts). 15

blockchain A consensus-based record of agreement where chunks of data[a] (called blocks) are stored with cryptographic hashes linking each Block to the next, ensuring veracity. 15

Contract A piece of EVM Code that may be associated with an Account or an Autonomous Object. 15

Dapp An end-user-visible application hosted in an Ethereum browser.. 15

design pattern a pattern of design in OOP. 15

Ethereum Runtime Environment The environment which is provided to an Autonomous Object executing in the EVM. Includes the EVM but also the structure of the world state on which the relies for certain I/O instructions including CALL & CREATE. 15

Ethereum Browser [b] A cross-platform GUI of an interface similar to a simplified browser (a la Chrome) that is able to host applications, the backend of which is purely on the Ethereum protocol.. 15

Ethereum Foundation The non-profit organization in charge of executing the development processes of Ethereum in line with the Whitepaper. 15

Ethereum Virtual Machine A secure, consensus-based global computer, and the subject of this paper. 15

EVM Assembly The human readable version of EVM code. 15

EVM Code The bytecode that the EVM can natively execute. Used to formally specify the meaning and ramifications of a message to an Account. 15

External Actor A person or other entity able to interface to an Ethereum node, but external to the world of Ethereum. It can interact with Ethereum through depositing signed Transactions and inspecting the blockchain and associated state. Has one (or more) intrinsic Accounts. 15

Gas The fundamental network cost unit. Paid for exclusively by Ether (as of PoC-4), which is converted freely to and from Gas as required. Gas does not exist outside of the internal Ethereum computation engine; its price is set by the Transaction and miners are free to ignore Transactions whose Gas price is too low. 15

[a] As of November 17, 2017, roughly between 1 and 20 kilobytes in size[1]

[b] a.k.a. Ethereum Reference Client

hacker ethic A maxim purporting that knowledge about systems should be free and unhindered[2]. 15

leaf node the bottom-most node in a particular tree, of blocks, one half of the "key" the other half being the root node, which creates the path between. 15

Lower-Level Lisp The Lisp-like Low-level Language, a human-writable language used for authoring simple contracts and general low-level language toolkit for trans-compiling to. 15

Message Data (as a set of bytes) and Value (specified in Wei) that is passed between two Accounts, either through the deterministic operation of an Autonomous Object or the cryptographically secure signature of the Transaction. 15

Message The act of passing a message from one Account to another. If the destination account is associated with non-empty EVM Code, then the VM will be started with the state of said Object and the Message acted upon. If the message sender is an Autonomous Object, then the Call passes any data returned from the VM operation. 15

Object Synonym for Autonomous Object. 15

public key A term originating from *cryptography* and corresponding to **private key**, this is the 42-byte (i.e. 42-character) string of ASCII digits which transacts on the public network. 15

root node the uppermost node in a particular tree, of blocks, representing a single world state$^{\sigma}$ at a particular time. 5, 15

serialization Serialization is the process of converting an object into a stream of bytes in order to store the object or transmit it to memory, a database, or a file. Its main purpose is to save the state of an object in order to be able to recreate it when needed. The reverse process is called deserialization.[3]. 15

singleton A design pattern in Object-Oriented Programming which specifies a class with one instance but with a global point of access to it[4]. 15

specification Technical descriptions, instructions, and definitions from which other people can create working prototypes. 15

state A permanent, static, state `state`. 15

state-transition . 15

State Database A database backend that maintains a mapping of bytearrays to bytearrays. 15

state machine A state machine rests in a universal, stable, singular condition, called a state. State machines transition to new states given certain compatible inputs.. 15

state database A database stored off-chain, [i.e. on the computer of some user running an Ethereum client] which contains a trie structure mapping bytearrays [i.e. organized chunks of binary data] to other bytearrays [other organized chunks of binary data]. The RELATIONSHIPS between each node on this trie constitute a MAP, a.k.a. a MAPPING of all PREVIOUS STATES on the **EVM** which a client might need to reference. 5, 6, 5, 6, 15

Storage State The information particular to a given Account that is maintained between the times that the Account's associated EVM Code runs. 15

transaction An input message to a system that, because of the nature of the real-world event or activity it reflects, is required to be regarded as a single unit of work guaranteeing to either be processed completely or not at all.[5]. 15

Transaction A piece of data, signed by an External Actor. It represents either a Message or a new Autonomous Object. Transactions are recorded into each block of the blockchain. 15

trie A tree-structure for organizing data, the position of data in the tree contains the particular path from root to leaf node that represents the key (the path from root to leaf is "one" key) you are searching the trie structure for. The data of the key is contained in the trie relationships that emerge from related nodes in the trie structure. 5, 15

Whitepaper A conceptual map, distinct from the Yellowpaper, which highlights the development goals for Ethereum as a whole[6]. 15

Yellowpaper Ethereum's first formal specification,[7] written by Dr. Gavin Wood, one of the founders of Ethereum. Notorious for its difficulty to read,[8] it provides both a boon and an obstacle to potential adopters of Ethereum. 15

Acronyms

ERE Ethereum Runtime Environment. 15

EVM Ethereum Virtual Machine. 15

LLL Lower Level Lisp. 15

OOP Object-Oriented Programming. 15

YP Yellowpaper. 15

Index