Ethereum: Ethereum Virtual Machine (EVM)

return to the Ethereum Standards

Table 1: Data sheet for Ethereum Virtual Machine (EVM)

Title	Ethereum Virtual Machine
Acronym	EVM
Version	0.6.8
Document Number	Solidity Smart Contracts: EVM
Release Date	17 December 2019
About Specification	https://solidity.readthedocs.io/en/v0.6.8/introduction-to-smart-contracts.html#index-6
Download	https://github.com/ethereum/solidity/releases

Note: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Overview

The Ethereum Virtual Machine or EVM is the runtime environment for smart contracts in Ethereum. It is not only sandboxed but actually completely isolated, which means that code running inside the EVM has no access to network, filesystem or other processes. Smart contracts even have limited access to other smart contracts.

Accounts

There are two kinds of accounts in Ethereum which share the same address space: External accounts that are controlled by public-private key pairs (i.e. humans) and contract accounts which are controlled by the code stored together with the account.

Transactions

A transaction is a message that is sent from one account to another account (which might be the same or empty, see below). It can include binary data (which is called "payload") and Ether.

Gas

Upon creation, each transaction is charged with a certain amount of gas, whose purpose is to

limit the amount of work that is needed to execute the transaction and to pay for this execution at the same time. While the EVM executes the transaction, the gas is gradually depleted according to specific rules.

Storage, Memory and the Stack

The Ethereum Virtual Machine has three areas where it can store data- storage, memory and the stack,

Instruction Set

The instruction set of the EVM is kept minimal in order to avoid incorrect or inconsistent implementations which could cause consensus problems. All instructions operate on the basic data type, 256-bit words or on slices of memory (or other byte arrays). For a complete list, please see the list of opcodes as part of the inline assembly documentation. ***REVIEW EVM OPCODES***
Note: This link is outdated. Look in https://docs.soliditylang.org/en/v0.8.13/assembly.html] Solidity 0.8.13 and Yul (previously also called JULIA or IULIA)

Message Calls

Contracts can call other contracts or send Ether to non-contract accounts by the means of message calls. Message calls are similar to transactions, in that they have a source, a target, data payload, Ether, gas and return data. In fact, every transaction consists of a top-level message call which in turn can create further message calls.

Delegatecall / Callcode and Libraries

There exists a special variant of a message call, named delegatecall which is identical to a message call apart from the fact that the code at the target address is executed in the context of the calling contract and msg.sender and msg.value do not change their values.

Logs

It is possible to store data in a specially indexed data structure that maps all the way up to the block level. This feature called logs is used by Solidity in order to implement events.

Deactivate and Self-destruct

The only way to remove code from the blockchain is when a contract at that address performs the selfdestruct operation.