

Code Execution

The code in Ethereum contracts is written in a low-level, stack-based bytecode language, referred to as "Ethereum virtual machine code" or "EVM code". The code consists of a series of bytes, where each byte represents an operation. In general, code execution is an infinite loop that consists of repeatedly carrying out the operation at the current program counter (which begins at zero) and then incrementing the program counter by one, until the end of the code is reached or an error or STOP or RETURN instruction is detected. The operations have access to three types of space in which to store data:

- The **stack**, a last-in-first-out container to which 32-byte values can be pushed and popped
- **Memory**, an infinitely expandable byte array
- The contract's long-term **storage**, a key/value store where keys and values are both 32 bytes. Unlike stack and memory, which reset after computation ends, storage persists for the long term.

The code can also access the value, sender and data of the incoming message, as well as block header data, and the code can also return a byte array of data as an output.

The formal execution model of EVM code is surprisingly simple. While the Ethereum virtual machine is running, its full computational state can be defined by the tuple (block_state, transaction, message, code, memory, stack, pc, gas), where block_state is the global state containing all accounts and includes balances and storage. Every round of execution, the current instruction is found by taking the pc-th byte of code, and each instruction has its own definition in terms of how it affects the tuple. For example, ADD pops two items off the stack and pushes their sum, reduces gas by 1 and increments pc by 1, and SSTORE pushes the top two items off the stack and inserts the second item into the contract's storage at the index specified by the first item, as well as reducing gas by up to 200 and incrementing pc by 1. Although there are many ways to optimize Ethereum via just-in-time compilation, a basic implementation of Ethereum can be done in a few hundred lines of code.

