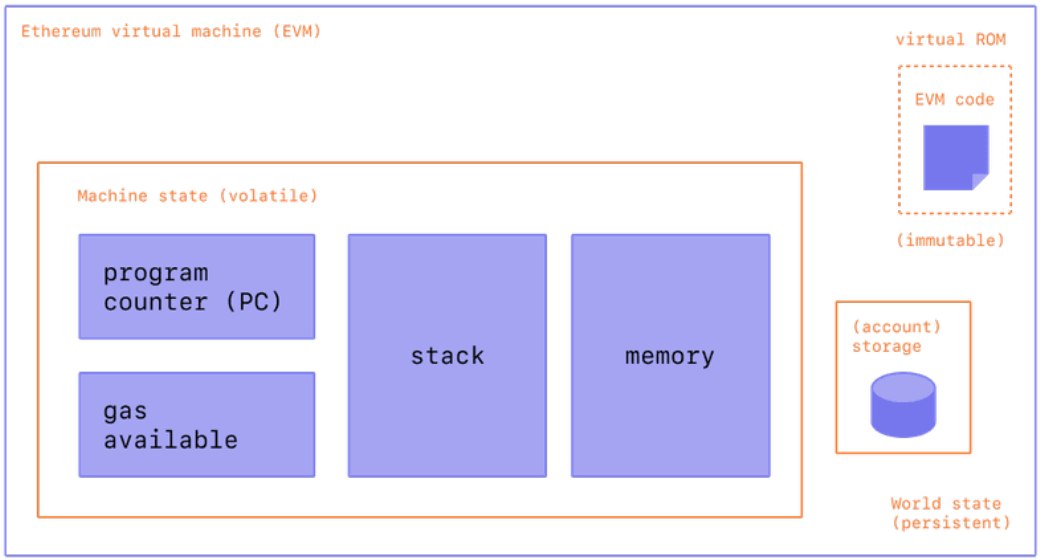
Ethereum Virtual Machine

Saravanan Vijayakumaran

Department of Electrical Engineering
Indian Institute of Technology Bombay

March 4, 2026

The Ethereum Virtual Machine



Source: `https://ethereum.org/developers/docs/evm`

- Stack-based computer that executes smart contract instructions
- Smart contracts are compiled down to EVM opcodes
- Stack-based architecture with 32-byte words
- Code is stored in a virtual ROM
- Stack and memory are volatile (do not persist across transactions)
- Storage is the persistent memory of smart contracts

EVM Resources

- The stack can hold a maximum of 1024 words
- Memory is a word-addressed byte array
  - While the maximum memory is theoretically 2^{256} bytes, the gas cost increases quadratically with memory size
- Account storage is a key-value map from words to words that persists across transactions
  - The storage of the current contract can be read and modified using `SLOAD` and `SSTORE` instructions
  - Storage of other contracts cannot be read
- The code of another contract can be read using the `EXTCODECOPY` instruction
- The data sent to a transaction can be read using `CALLDATALOAD` and `CALLDATACOPY`

EVM Opcodes

- The EVM has 1-byte opcodes
  - See full list at `https://www.evm.codes/`
- Each opcode has a gas cost associated with it
  - `ADD, SUB` have a gas cost of 3
  - `MUL, DIV` have a gas cost of 5
  - `MLOAD, MSTORE` have a gas cost of 3 + memory expansion cost
- Opcodes that access/modify storage or world state have a higher gas cost
  - `SLOAD` loads a word from storage
    - 2100 gas for cold keys
    - 100 gas for warm keys
  - `CALL` calls another contracts
    - 2600 gas for cold addresses
    - 100 gas for warm addresses
  - `BALANCE` gets the Ether balance of an address
    - 2600 gas for cold addresses
    - 100 gas for warm addresses
  - `CREATE` creates a new contract and has a minimum gas cost of 32000

Memory Expansion Gas Costs

- Some opcodes access memory locations by specifying an offset
- Example: `MSTORE`
  - Consumes the top two stack words as inputs
  - Top stack word is interpreted as offset in the memory to write
  - The next stack word is written to memory at offset location
- If the offset is larger than already accessed offset, it may trigger a memory expansion cost
- Memory is expanded in 32-byte increments
- Memory expansion cost formula

$$\text{memory\_size\_word} = \frac{\text{memory\_byte\_size} + 31}{32}$$

$$\text{memory\_cost} = \frac{(\text{memory\_size\_word})^2}{512} + 3 \times \text{memory\_size\_word}$$

Notable Gas Costs

- Every transaction has an initial cost of 21000 gas
  - Can cost more if it involves more than a simple transfer of Ether
- Calldata costs 4 gas per zero byte and 16 gas per non-zero byte
  - Calldata is available only during the execution of the current transaction
  - Storage is available across transactions and consequently costs more
  - The lower cost of calldata is crucial to the design of Layer 2 blockchains based on Ethereum
- `KECCAK256` costs gas equal to 30 + 6 per hashed word

Precompiles

- The EVM also offers a set of advanced operations using precompiled contracts
- These are contracts accessible through fixed addresses
  - See full list at `https://www.evm.codes/precompiled`
- Example: `ecPairing`
  - Located at address `0x08`
  - Computes an elliptic curve pairing
  - Used to verify Groth16-based SNARK proofs
  - Costs 45000 gas

References

- Yellow paper `https://ethereum.github.io/yellowpaper/paper.pdf`
- EVM Opcodes `https://www.evm.codes/`
- EVM Precompiles `https://www.evm.codes/precompiled`