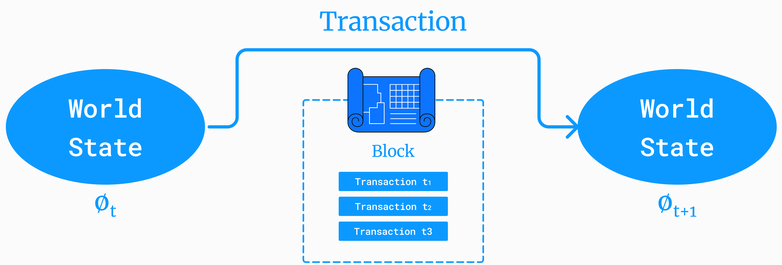
Section 1

# EVM

Inspired by Soliditylang Docs && Takenobu T. && @deliriusz

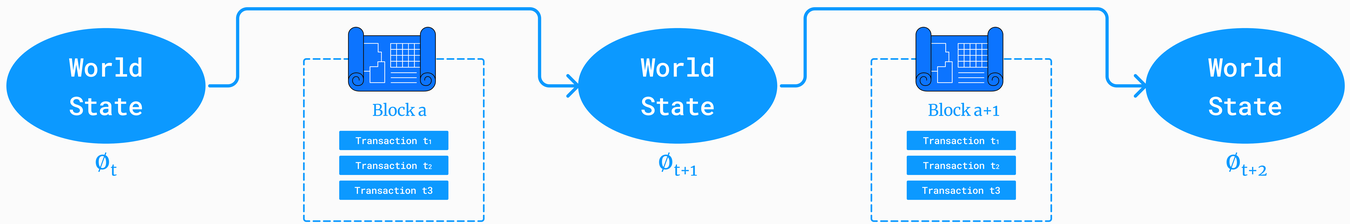
## A Transaction-Based State Machine




- Transactions are collacted into blocks.
- A block is a package of data.

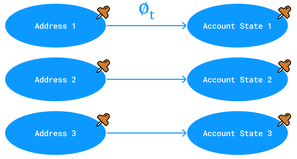
**Ethereum can be viewed as a transactionbased state machine**

## Chain of States



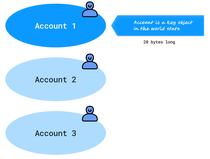

**From the viewpoint of the state, Ethereum can be seen as a state chain.**

## World State



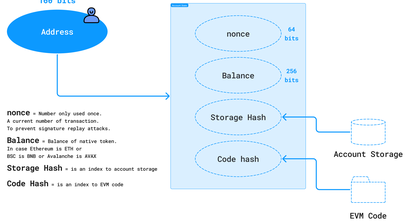

The world state is a mapping between address and account state.

## Account




An account is an object in the world state.

## Account State



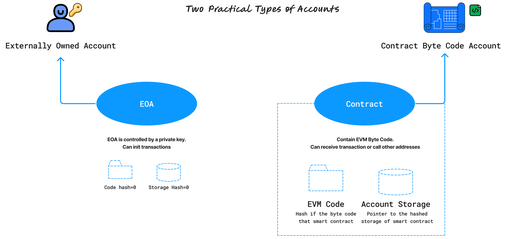

**nonce** = Number only used once.
A current number of transaction.
To prevent signature replay attacks.

**Balance** = Balance of native token.
In case Ethereum is ETH or
BSC is BNB or Avalanche is AVAX

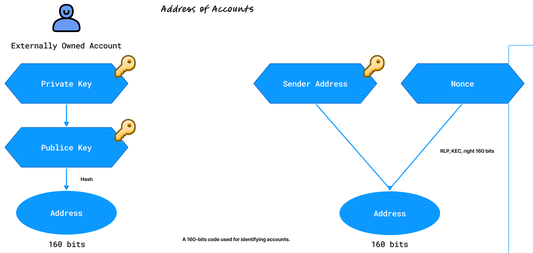
**Storage Hash** = is an index to account storage

**Code Hash** = is an index to EVM code

## Two Practical Types of Accounts




## Address of Accounts




A 160-bits code used for identifying accounts.

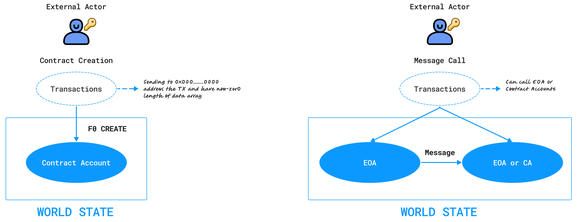
**F0 CREATE :** Create a new account with associated code
Creates a new contract using the initialisation code provided at the indicated offset in the memory. Enters a new sub-context of the calculated destination address and executes the initialisation code, then resumes the current context.
Should deployment succeed, the new account's code is set to the return data resulting from executing the initialisation code.
The destination address is calculated as the rightmost 20 bytes (160 bits) of the Keccak-256 hash of the rlp encoding of the sender address followed by its nonce. That is:

```
address = keccak256(rlp([sender_address,sender_nonce]))[12:]
```

**F5 CREATE2 :** Equivalent to CREATE, except the salt allows the new contract to be deployed at a consistent, deterministic address.
Should deployment succeed, the account's code is set to the return data resulting f rom executing the initialisation code.
The destination address is calculated as follows:

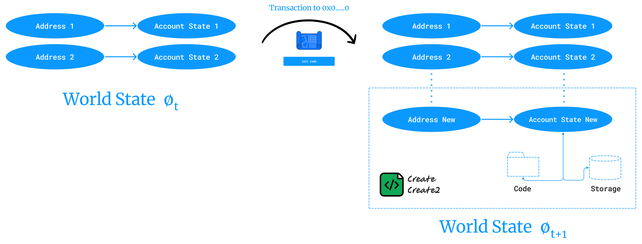
```
initialisation_code = memory[offset:offset+size]
address = keccak256(0xff + sender_address + salt + keccak256(initialisation_code))[12:]
```

## Two Practical Types of Transactions

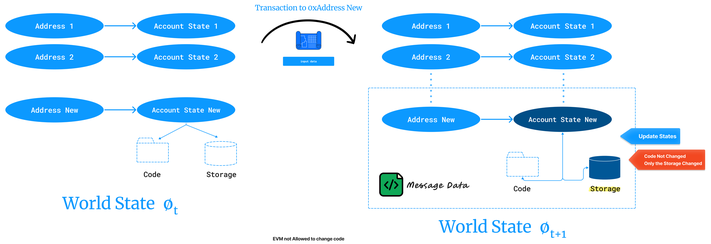



There are two practical types of transaction, contract creation and message call

## Contract Creation

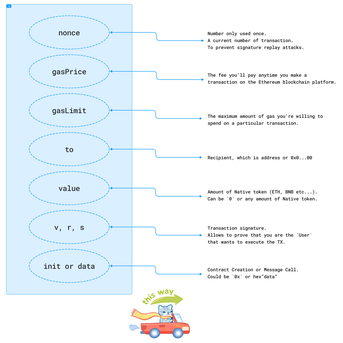



## Message Call




EVM not Allowed to change code

## Field of A Transaction



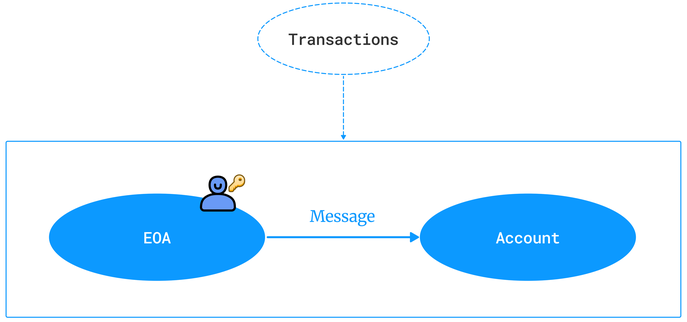
## Message

WORLD STATE

Message is passed between two Accounts.
Message is Data (as a set of bytes) and Value (Native Token as ETH)

## Message Trigger

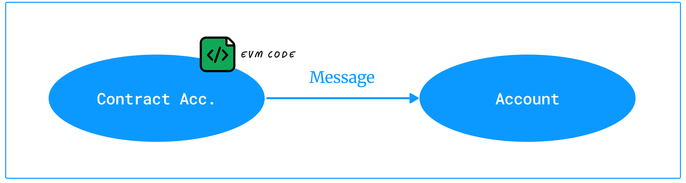
Triggered by TX




WORLD STATE

Transaction triggers an associated message.

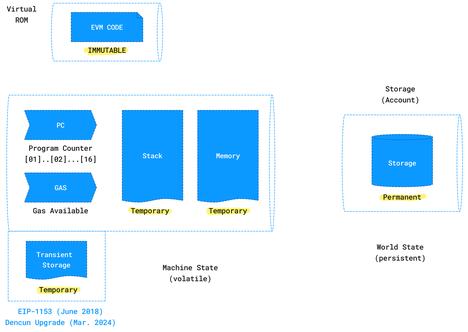
Triggered by EVM Byte Code




WORLD STATE

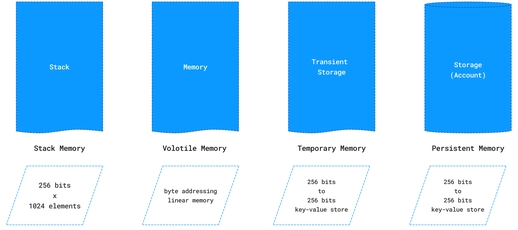
EVM can also send a message.

## EVM: Ethereum Virtual Machine



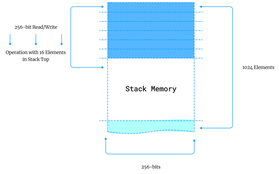

The EVM is a simple stack-based architecture.

## Machine Space of EVM



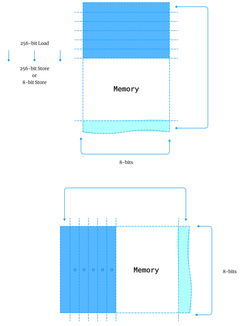

There are several resources as space.

## Stack



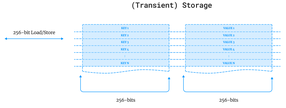

All operation are performed on the Stack.
Access with many instructions such as PUSH/POP/COPY/SWAP.....

## Memory




Memory is Linear and can be addressed at byte level.
Access with MSTORE/MSTORE8/MLOAD instructions.
All locations in Memory are well-defined initially as Zero.

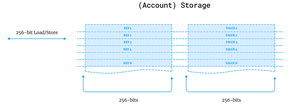
## Transient Storage




- solidity v0.8.24 OPCODES (TLOAD, TSTORE)
- solidity v0.8.28 Syntax (transient)
- cheaper than persistent storage
- exist only until the end of TX

Storage is a key-value store that maps 256-bit words to 256-bit words.
Access with SSTORE/SLOAD instructions.
All locations in storage are will-defined initially as zero.

## Account Storage




Storage is a key-value store that maps 256-bit words to 256-bit words.
Access with SSTORE/SLOAD instructions.
All locations in storage are will-defined initially as zero.