# Theory of Blockchain

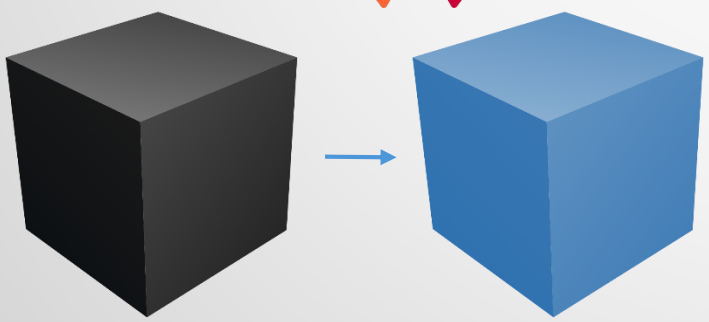
## Session 9:

## Ethereum – Part 1

Module 2 – Ethereum Virtual Machine & Gas

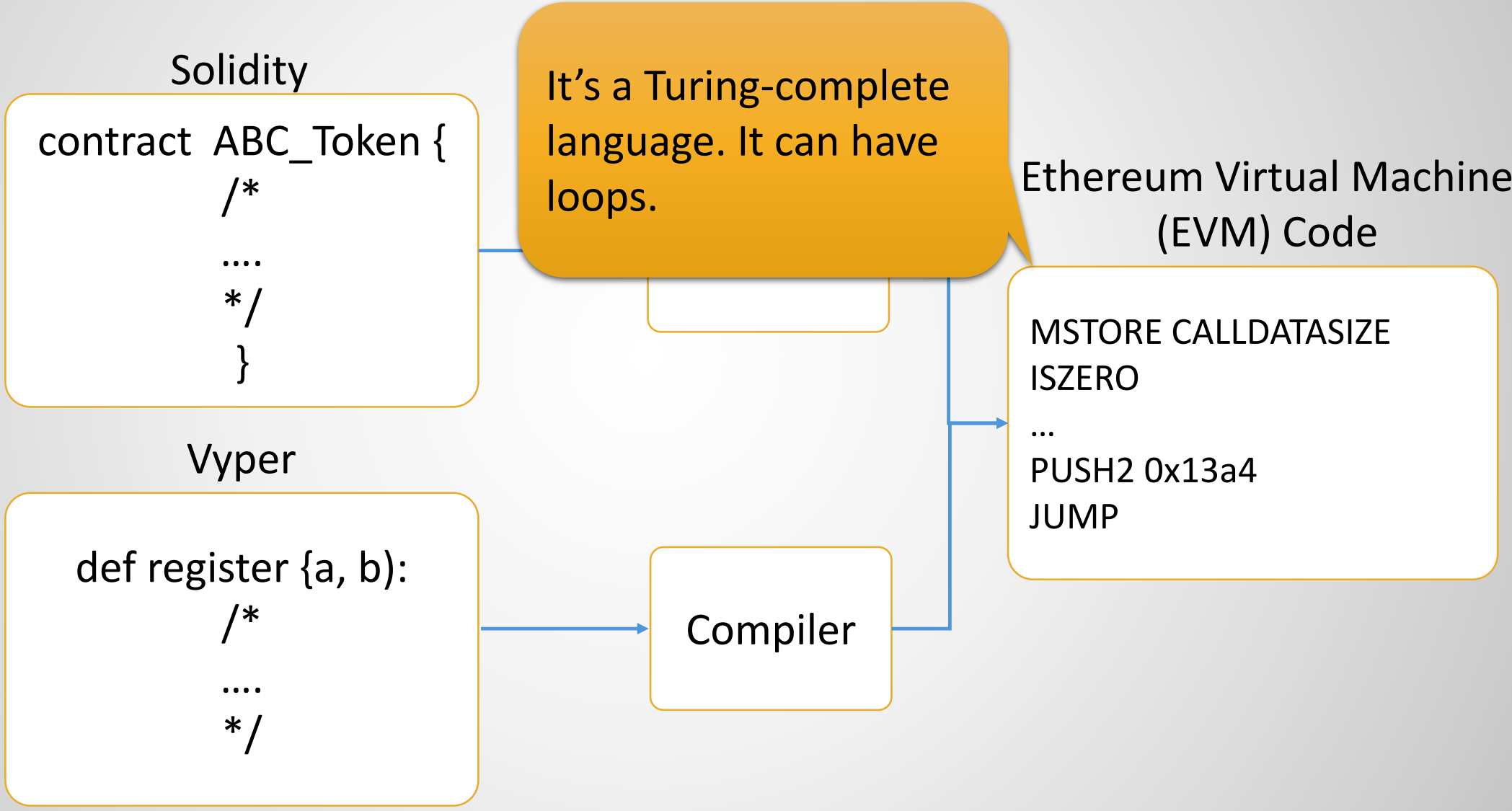
# Smart Contract (remember?)

Smart Contract is a piece of code or computer program, stored in blockchain.



```
#
# Example ETX subcurrency
#

data owner
data exchange
data market_id
data balances[2^160](balance)

extern exchange: [deposit:[int256,int256,int256]:int256]

def init():
    self.owner = msg.sender
    self.balances[msg.sender].balance = 1000000 * 10 ** 5

def transfer(recipient, amount):
    # Prevent negative send from stealing funds
    if recipient <= 0 or amount <= 0:
        return(0)

    # Get user balance
    balance = self.balances[msg.sender].balance
```

# Ethereum Virtual Machine (EVM)



# Ethereum Virtual Machine (EVM)

EVM code comes from a low-level stack-based bytecode language (do u see similarities with JVM?).

EVM codes are machine-readable codes that run in EVMs on network nodes.

Every node who wants to verify the blocks, can run the associated contracts and the transactions in its EVM.

# EVM – Necessity of Gas Definition

But if every node verifies the run of contracts and (unlike Bitcoin) we can have loops, then some codes might never end.

- Malicious entities can create DoS

It’s not always possible to judge if a code ultimately terminates before actually running it (halting problem).

```
pragma solidity ^0.4.10;
contract TestLoop {
    function test() returns (uint) {
        uint x = 0;
        for (uint i = 5; i > 1; i++) {
            x = x*i + x;
        }
        return x;
    }
}
```

That is why gas was created.

# EVM – Necessity of Gas Definition

- Gas fuels smart contract execution.
- Every operation in the EVM code requires certain amount of gas to be run.
  - So does a whole program/contract

- Each transactions has:
  1) **Startgas/Gaslimit** : the maximum amount (number) of gas it is willing to spend.
  2) **gasprice:** the fee (in ether or wei) that it is going to pay per unit of gas.

1 ETH =
10^18 Wei

# EVM – Necessity of Gas Definition

- Gas fuels smart contract execution.
- Every operation in the EVM code requires certain amount of gas to be run.
  - So does a whole program/contract

- Each transactions has:
  1) **Startgas/Gaslimit** : the maximum amount (number) of gas it is willing to spend.
  2) **gasprice:** the fee (in ether or wei) that it is going to pay per unit of gas.

At the beginning of a transaction, startgas*gasprice is deducted from sender's account (the account poking the contract). If the contract executes successfully, the remaining gas is returned to the sender. If gas runs out, execution is aborted and nothing is refunded.

# EVM – Necessity of Gas Definition

- More complicated transactions involving smart contracts require more computational work, so they require a higher gas limit than a simple payment.
  - A standard ETH transfer requires a (gas) limit of 21,000 units of gas.
- **Example:** If you put a gas limit of 50,000 for a simple ETH transfer, the EVM would consume 21,000, and you would get back the remaining 29,000. However, if you specify too little gas, for example, a gas limit of 20,000 for a simple ETH transfer, the EVM will consume your 20,000 gas units attempting to fulfill the transaction, but it will not complete. The EVM then reverts any changes, but since the miner/validator has already done 20k gas units worth of work, that gas is consumed.
  - Gas mechanism prevents Denial of Service (DoS) attack.

# Gas Calculation – Before London Upgrade

- The way transaction fees on the Ethereum network were calculated changed with the London Upgrade in August 2021.

**Before London Upgrade:**

- Let's say Alice had to pay Bob 1 ETH. In the transaction, the gas limit was 21,000 units, and the gas price was 200 gwei.

- Total fee would be: Gas units (limit) * Gas price per unit i.e 21,000 * 200 = 4,200,000 gwei or 0.0042 ETH.

# Gas Calculation – After London Upgrade

- Let's say Jordan has to pay Taylor 1 ETH. In the transaction, the gas limit is 21,000 units and the base fee is 10 gwei. Jordan includes a tip of 2 gwei.

1 ETH = 10^9 gwei

- The total fee would now be: units of gas used * (base fee + priority fee) where the base fee is a value set by the protocol and the priority fee is a value set by the user as a tip to the validator.

→ 21,000 * (10 + 2) = 252,000 gwei or 0.000252 ETH (0.00021 + 0.000042).

- When Jordan sends the money, 1.000252 ETH will be deducted from his account. Taylor will be credited 1.0000 ETH. Validator receives a tip of 0.000042 ETH. Base fee of 0.00021 ETH is burned.

- Additionally, Jordan can also set a max fee (maxFeePerGas) for the transaction. The difference between the max fee and the actual fee is refunded to Jordan, i.e. refund = max fee - (base fee + priority fee).

(ethereum.org)

# Gas Calculation – After London Upgrade

- Let's say Jordan has to pay Taylor 1 ETH. In the transaction, the gas limit is 21,000 units and the base fee is 10 gwei. Jordan includes a tip of 2 gwei.

1 ETH = 10^9 gwei

- The total fee would now be: units of gas used * (base fee + priority fee) where the base fee is a value set by the protocol and the priority fee is a value set by the user as a tip to the validator.

→ 21,000 * (10 + 2) = 252,000 gwei or 0.000252 ETH (0.00021 + 0.000042).

- When Jordan sends the money, 1.000252 ETH will be deducted from his account. Taylor will be credited 1.0000 ETH. Validator receives a tip of 0.000042 ETH. Base fee of 0.00021 ETH is burned.

- Additionally, Jordan can also set a max fee (maxFeePerGas) for the transaction. The difference between the max fee and the actual fee is refunded to Jordan, i.e. refund = max fee - (base fee + priority fee).

Jordan can set a maximum amount to pay for the transaction to execute and not worry about overpaying "beyond" the base fee when the transaction is executed.

# But Whom Should the Gas be Paid to?

- The gas (tip) goes to the validator/miner who includes the transactions in a block.

To be eligible for inclusion in a block the offered price per gas must at least equal the base fee. The base fee is calculated independently of the current block and is instead determined by the blocks before it - making transaction fees more predictable for users. When the block is mined this base fee is "burned", removing it from circulation.

The base fee is calculated by a formula that compares the size of the previous block (the amount of gas used for all the transactions) with the target size. The base fee will increase by a maximum of 12.5% per block if the target block size is exceeded. This exponential growth makes it economically non-viable for block size to remain high indefinitely.

# What Comes Next …

- We learned the concept of EVM and gas.
- We got familiar with the smaller currency units in the Ethereum network and how transaction fees are calculated.
- We next learn about the consensus mechanisms of Ethereum and their alternatives.

See you in the next module …