ETHEREUM :

Ethereum is a platform powered by blockchain technology that is best known for its native cryptocurrency, called ether, or ETH, or simply ethereum. The distributed nature of blockchain technology is what makes the Ethereum platform secure, and that security enables ETH to accrue value.

Ethereum, just like any other blockchain, can be visualized as a transaction-based state machine. The idea is that a genesis state is transformed into a final state by executing transactions incrementally. The final transformation is then accepted as the absolute undisputed version of the state. In the following diagram, the Ethereum state transition function is shown, where a transaction execution has resulted in a state transition.

Elements of Ethereum Block chain:

In the following section, you will be introduced to various components of the Ethereum network and the blockchain. First, the basic concept of the EVM is givenin the next section.

Ethereum virtual machine(EVM)

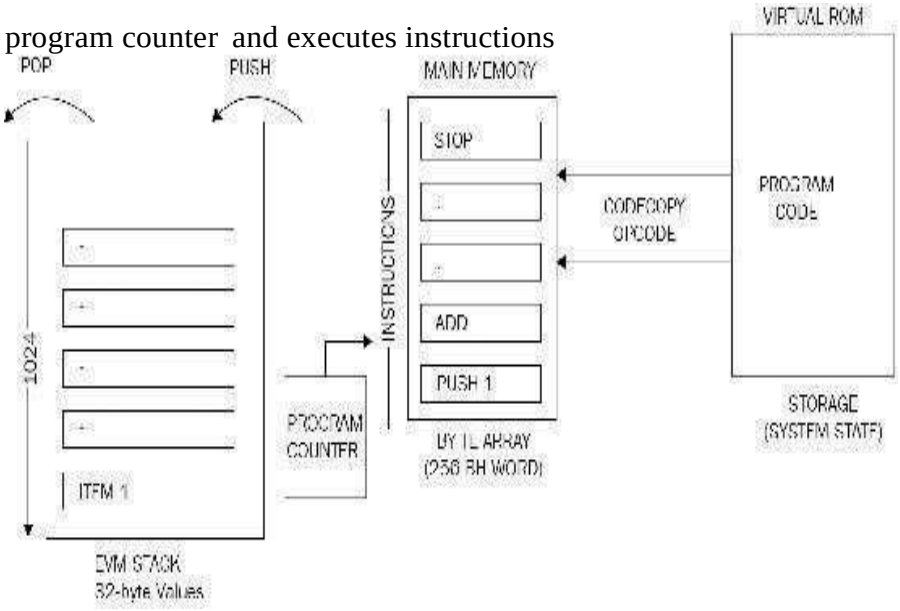
EVM is a simple stack-based execution machine that runs bytecode instructions in order to transform the system state from one state to another. The word size of the virtual machine is set to 256-bit. The stack size is limited to 1024 elements and is based on the **LIFO** (**Last in First Out**) queue. EVM is a Turing-complete machine but is limited by the amount of gas that is required to runany instruction. This means that infinite loops that can result in denial of service attacks are not possible due to gas requirements.

EVM also supports exception handling in case exceptions occur, such as not having enough gas or invalid instructions, in which case the machine would immediately halt and return the error to the executing agent.EVM is a fully isolated and sandboxed runtime environment. EVM is a stack-based architecture.EVM is big-endian by design and it uses 256-bit wide words. This word size allows for Keccak 256-bit hash and elliptic curve cryptography computations.

EVM also supports exception handling in case exceptions occur, such as not having enough gas or invalid instructions, in which case the machine would immediately halt and return the error to the executing agent.EVM is a fully isolated and sandboxed runtime environment.

As discussed earlier, EVM is a stack-based architecture. EVM is big-endian by design and it uses 256-bit wide words. This word size allows for Keccak 256-bit hash and elliptic curve cryptography computations.

The following diagram shows the design of the EVM where the virtual ROM stores the program code that is copied into main memory using CODECOPY. The main memory is then read by the EVM by referring to the program counter and executes instructions



EVM operation

EVM optimization is an active area of research and recent research has suggested that EVM can be optimized and tuned to a very fine degree in order toachieve high performance. Research into the possibility of using Web assembly (WASM) is underwayalready. WASM is developed by Google, Mozilla, and Microsoft and is now being designed as an open standard by the W3C community group. The aim of WASM is to be able to run machine code in the browser that will result in execution at native speed. Similarly, theaim of EVM 2.0 is to be able to run the EVM instructionset (Opcodes) natively in CPUs, thus making it faster and efficient.

PRE-COMPILED CONTRACTS:

There arefour precompiled contracts in Ethereum. Hereis the list of these contracts and details.
The elliptic curve public key recovery function

ECDSARECOVER (Elliptic curve DSA recover function) is available at address 1. It is denoted as ECREC and requires 3000 gas for execution. If the signature is invalid, then no output is returned by this function. Public key recovery is a standard mechanism by which the public key can be derived from the privatekey in elliptic curve cryptography.
The ECDSA recovery function is shown as follows:

ECDSARECOVER(H, V, R, S) = Public Key

It takes four inputs: H, which is a 32 byte hash of the message to be signed and *V*, *R*, and *S*, which represent the ECDSA signature with the recovery ID and produce a64 byte public key. *V*, *R*, and *S* have been discussed in detail previously in this chapter.

The SHA-256 bit hash function

The SHA-256 bit hash function isa precompiled
contract that is available at address 2 and produces a SHA256 hash of the input. It is almost like a pass-through function. Gas requirement for SHA-256 (SHA256) depends on the input data size. The output is a 32 bytevalue.

The RIPEMD-160 bit hashfunction

The RIPEMD-160 bit hash function is used to provide RIPEMD 160 -bit hash and is available at address 3. Theoutput of this function is a 20-byte value. Gas requirement, similar to SHA-256, is dependent on the amount ofinput data.

The identity function:

The identity function is available at address 4 and is denoted by the ID. It simply defines output as input; in

other words, whatever input is given to the ID function, it will output the same value. Gas requirement is calculated by a simple formula: $15 + 3 [I_d/32]$ where I_d is the input data. This means that at a high level, the gas requirement is dependent on the size of the input data albeit with some calculation performed, as shown in the preceding equation.All the previously mentioned precompiled contracts canbecome native extensions and can be included in the EVM opcodes in the future.

ACCOUNTS AND ITS TYPES :

Accounts are one of the main building blocks of the Ethereum blockchain. The state is created or updated asa result of the interaction between accounts. Operations performed between and on the accounts represent state transitions. State transition is achieved using what's called the Ethereum state transition function, which works as follows:

1. Confirm the transaction validity by checking the syntax, signaturevalidity, and nonce.
2. Transaction fee is calculated and the sending address is resolved using the signature. Furthermore, sender's account balance is checked and subtracted accordingly and nonce is incremented. Anerror is returned if the account balance is not enough.
3. Provide enough ether (gas price) to cover the cost of the transaction. This is charged per byte incrementally according to the size of the transaction.
4. In this step, the actual transfer of value occurs. The flow is from the sender's account to receiver's account. The account is created automatically if the destination account specified in the transaction does not exist yet. if the destination account is a contract, then the contract code is executed. If enough gas is available, then the contract code willbe executed fully; otherwise, it will run up to the point where it runs out ofgas.
5. In cases of transaction failure due to insufficient account balance orgas, all state changes are rolled back with the exception of fee payment, which is paid to the miners.
6. Finally, the remainder (if any) of the fee is sent back to the sender as change and fee is paid to the miners accordingly. At this point, thefunction returns the resulting state.

TYPES OF ACCOUNTS : There are two types of accounts in Ethereum:

- Externally owned contacts
- Contract accounts

The first is **externally owned accounts** (**EOAs**) and the other is contract accounts. EOAs are similar to accountsthat are controlled by a private key in bitcoin. Contract accounts are the accounts that have code associated with them along with the private key. An EOA has ether balance, is able to send transactions, and has no associated code, whereas a **Contract Account** (**CA**) has ether balance, associated code, and the ability to get triggered and execute code in response to a transaction or a message that due to the Turing - completeness property of the Ethereum blockchain, the code within contract accounts can be of any level of complexity. The code is executed by EVM by each mining node on the Ethereum network. In addition, contract accounts are able to maintain their own permanent state and can call oth er contracts. It is envisaged that in the serenity release, the distinction between externally owned accounts and contract accounts may be eliminated.

Blocks

As discussed earlier, blocks are the main building blocksof a blockchain. Ethereum blocks consist of various components, which are described as follows:

- The block header
- The transactions list
- The list of headers of ommers or uncles

The transaction list is simply a list of all transactions included in the block. In addition, the list of headers of Unclesisalso included inthe block. The most important and complex part is the block header.

BLOCK HEADER

Block headers are the most critical and detailed components of an Ethereum block. The header contains valuableinformation, which is described in detail here.

PARENT HASH

This is the Keccak 256-bit hash of the parent (previous)block's header.

OMMERS HASH

This is the Keccak 256-bit hash ofthe list of Ommers (Uncles) blocks included in the block.

BENEFICIARY

Beneficiary field contains the 160-bit address of the recipient that will receive the mining reward once theblock is successfully mined.

STATE ROOT

The state root field contains the Keccak 256-bit hash ofthe root node of the state trie. It is calculated after all transactions have been processed and finalized.

TRANSACTIONS ROOT

The transaction root is the Keccak 256 -bit hash of the root node of the transaction trie. Transaction trie representsthe list of transactions included in the block.

RECEIPTS ROOT

The receipts root is the keccak 256 bit hash of the root node of the transaction receipt trie. This trie is composedo f receipts of all transactions included in the block.

Transaction receipts are generated after each transaction is processed and contain u seful post- transaction information. More details on transactionreceipts.

LOGS BLOOM

The logs bloom is a bloom filter that is composed of the logger address and log topics from the log entry of eachtransaction receipt of the included transaction list in the block. Logging is explained in detail in the next section.

DIFFICULTY

The difficulty level of the current block.

NUMBER

Thetotal number ofall previous blocks; the genesisblock is block zero.

GAS LIMIT: The field contains the value that represents the limit setonthe gas consumption per block.

GAS USED: The field contains the total gas consumed by thetransactions included in the block.

TIMESTAMP : Timestamp is the epoch Unix time of the time of blockinitialization.

EXTRA DATA: Extra data field can be used to store arbitrary datarelated to the block.

MIXHASH:

Mixhash field contains a 256-bit hash that once combined with the nonce is used to prove that adequate computational effort has been spent in order to create this block.

NONCE

Nonce is a 64-bit hash (a number) that is used to prove, in combination with the mixhash field, that adequate computational effort has been spent in order to create this block.

The following figure shows the detailed structure oftheblock and block header: