

--	--	--	--	--	--	--	--	--	--

Internal Assessment Test II – May 2025

Internal Assessment Test II - May 2025										
Sub:	Blockchain Technology					Sub Code:	BCS613A	Branch:	CSE	
Date:	.2025	Duration:	90 minutes	Max Marks:	50	Sem / Sec:	VI / A, B, C		OBE	
<u>Answer any FIVE FULL Questions</u>								MARKS	CO	RBT
1	Discuss the concept of Gas in Ethereum. Why is Gas important, and how does it impact transaction execution?							10	CO4	L2
2	What is a Genesis Block? Describe its significance in maintaining the security and continuity of the blockchain							10	CO3	L2
3	Briefly discuss about the operation of Ethereum Virtual Machine (EVM) with the help of a neat diagram.							10	CO4	L2
4	What are Ricardian Contracts? Compare and contrast Ricardian Contracts and Smart Contracts.							10	CO5	L2
5	Discuss real-world applications of Hyperledger Fabric and Corda in industries like finance, healthcare, and supply chain management.							10	CO5	L2
6	Give the different types of Transactions in Ethereum and also explain the fields included in these transactions.							10	CO4	L2

CI

CCI

HoD

USN

--	--	--	--	--	--	--	--	--	--

Internal Assessment Test II –May 2025

Internal Assessment Test II - May 2025										
Sub:	Blockchain Technology					Sub Code:	BCS613A	Branch:	CSE	
Date:	2025	Duration:	90 minutes	Max Marks:	50	Sem / Sec:	VI / A, B, C		OBE	
<u>Answer any FIVE Questions</u>								MARKS	CO	RBT
1	Discuss the concept of Gas in Ethereum. Why is Gas important, and how does it impact transaction execution?							10	CO3	L2
2	What is a Genesis Block? Describe its significance in maintaining the security and continuity of the blockchain							10	CO4	L2
3	Briefly discuss about the operation of Ethereum Virtual Machine (EVM) with the help of a neat diagram.							10	CO3	L2
4	What are Ricardian Contracts? Compare and contrast Ricardian Contracts and Smart Contracts.							10	CO4	L2
5	Discuss real-world applications of Hyperledger Fabric and Corda in industries like finance, healthcare, and supply chain management.							10	CO5	L2
6	Give the different types of Transactions in Ethereum and also explain the fields included in these transactions.							10	CO5	L2

CI

CCI

HoD

Q 1 : Discuss the concept of Gas in Ethereum. Why is Gas important, and how does it impact transaction execution? (10 Marks)

1. Introduction to Gas in Ethereum (2 Marks)

In the Ethereum blockchain, **Gas** is a fundamental unit that measures the computational effort required to perform transactions and execute smart contracts. It serves as a **fee mechanism** to allocate resources fairly and ensure that the Ethereum Virtual Machine (EVM) functions efficiently. Gas is paid in **Ether (ETH)**, the native cryptocurrency of Ethereum.

2. Importance of Gas in Ethereum (4 Marks)

Gas plays a crucial role in the Ethereum ecosystem for the following reasons:

1. Prevents Network Abuse

- Gas ensures users pay for computation, preventing spam and denial-of-service (DoS) attacks.

2. Fair Resource Allocation

- The Ethereum network has limited processing power. Gas fees help in distributing these resources fairly.

3. Transaction Prioritization

- Validators (previously miners) prefer transactions with **higher gas prices**, ensuring faster inclusion in the blockchain.

4. Incentive Mechanism

- Gas fees are given as rewards to validators, encouraging them to process transactions and maintain the network.
-

3. Gas and Transaction Execution (4 Marks)

Gas directly impacts how transactions are executed on the Ethereum network:

- **Gas Limit**
 - The maximum amount of gas a user is willing to consume for a transaction. If the execution exceeds the gas limit, the transaction **fails**, but the gas is **still consumed**.
- **Gas Price (in Gwei)**
 - It is the amount the user is willing to pay per unit of gas. It affects the **speed of transaction** processing—higher gas prices lead to quicker confirmation.
- **Transaction Fee Calculation**
$$\text{Transaction Fee} = \text{Gas Used} \times \text{Gas Price}$$
- **EIP-1559 Upgrade (London Hard Fork)**
 - Introduced a **base fee** (burned) and an optional **tip** to incentivize validators. This improved transaction fee predictability.

4. Conclusion (1 Mark)

Gas is a vital part of Ethereum's infrastructure. It not only facilitates transaction processing but also maintains the economic and operational stability of the network. By regulating computation and storage costs, gas ensures efficient and secure execution of operations within Ethereum.

Q:2 What is a Genesis Block? Describe its significance in maintaining the security and continuity of the blockchain. (10 Marks)

1. Definition of Genesis Block (2 Marks)

The **Genesis Block** is the **first block** in any blockchain. It is the **foundation** upon which the entire blockchain is built. In Ethereum and Bitcoin, this block is **hardcoded** into the software and has no previous block (i.e., its **parent hash is null or zero**).

- Also referred to as **Block 0** or **Block #0**.
 - It is created manually and distributed to all network nodes.
-

2. Characteristics of the Genesis Block (2 Marks)

- Has no predecessor (Parent Hash = 0).
 - Contains initial configuration parameters, like:
 - Network difficulty
 - Initial timestamp
 - Initial allocation of cryptocurrency (e.g., pre-mined coins or token distribution).
 - Cannot be changed or removed without changing the entire blockchain.
-

3. Significance in Blockchain Security (3 Marks)

1. Trust Anchor

- All subsequent blocks are validated based on the Genesis Block. It is the root of the blockchain's trust.

2. Tamper Resistance

- Since every block contains the hash of its previous block, altering the Genesis Block would break the entire chain, making tampering highly detectable.

3. Cryptographic Linking

- It initiates the cryptographic link between blocks, enabling immutability and consistency.
-

4. Significance in Continuity (2 Marks)

1. Ensures Blockchain Integrity

- Acts as a permanent starting point that nodes agree on, ensuring a consistent ledger across the decentralized network.

2. Synchronization Reference

- New nodes joining the network use the Genesis Block to start syncing with the rest of the blockchain.

5. Conclusion (1 Mark)

The **Genesis Block** plays a foundational role in blockchain technology. It ensures **security** through cryptographic linking and **continuity** by serving as the trusted starting point. Without it, the entire chain would lack structure and trustworthiness.

Q 3 Briefly discuss about the operation of Ethereum Virtual Machine (EVM) with the help of a neat diagram.

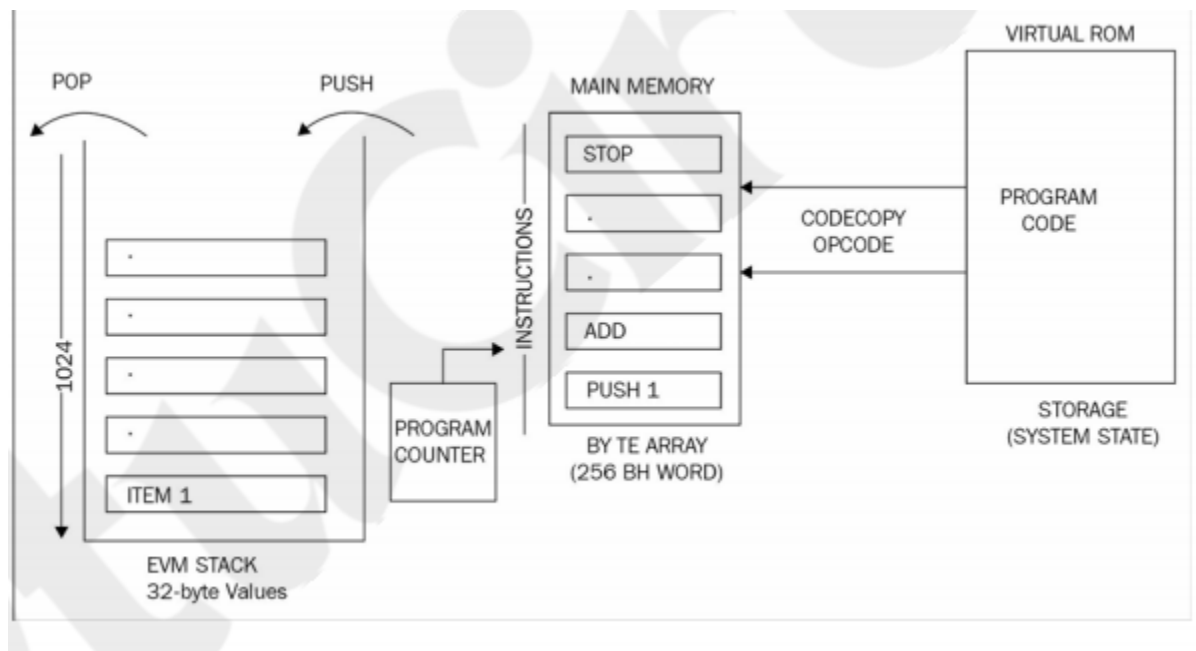
Elements of the Ethereum blockchain

- 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 run any 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. The code that runs on the EVM does not have access to any external resources, such as a network or filesystem.
- A 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.

There are two types of storage available to contracts and EVM. The first one is called memory, which is a **byte array**. When a contract finishes the code execution, the memory is cleared. It is akin to the concept of RAM.

- The other type, called **storage**, is permanently stored on the blockchain. It is a key value store. Memory is unlimited but constrained by gas fee requirements. The storage associated with the virtual machine is a word addressable word array that is non-volatile and is maintained as part of the system state.
- Keys and value are 32 bytes in size and storage. The program code is stored in a virtual read only memory (virtual ROM) that is accessible using the CODECOPY instruction. The CODECOPY instruction is used to copy the program code into the main memory. Initially, all storage and memory is set to zero in the EVM. 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 step by step. The program counter and EVM stack are updated accordingly with each instruction execution.



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 to achieve high performance. Research into the possibility of using Web assembly (WASM) is underway already. 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

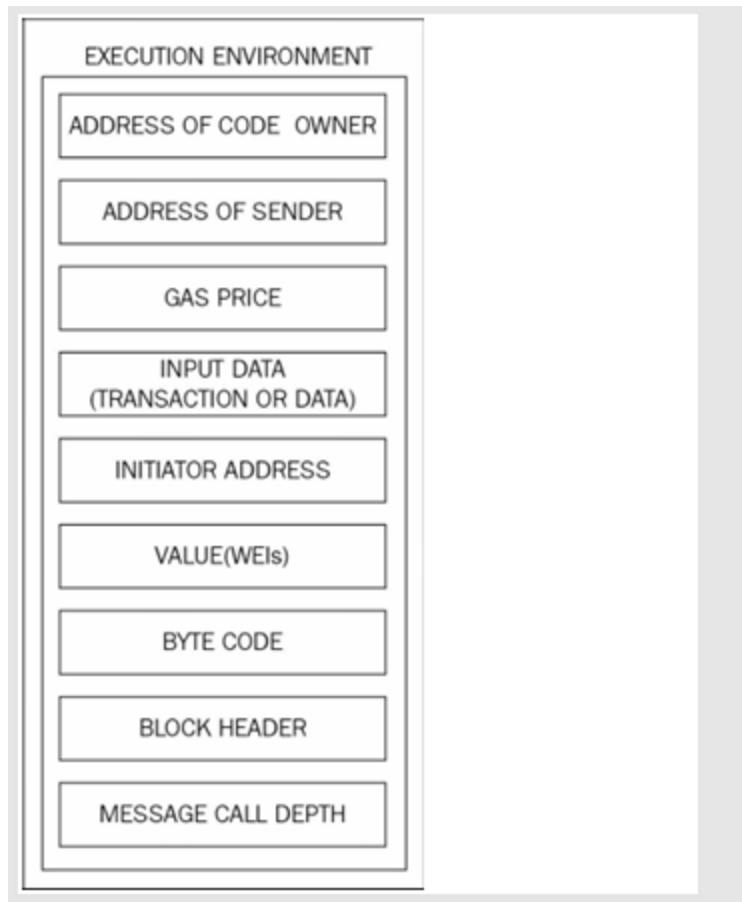
is to be able to run machine code in the browser that will result in execution at native speed. Similarly, the aim of EVM 2.0 is to be able to run the EVM instruction set (Opcodes) natively in CPUs, thus making it faster and efficient.

Execution environment

There are some key elements that are required by the execution environment in order to execute the code. The key parameters are provided by the execution agent, for example, a transaction. These are listed as follows:

1. The address of the account that owns the executing code.
2. The address of the sender of the transaction and the originating address of this execution.
3. The gas price in the transaction that initiated the execution.
4. Input data or transaction data depending on the type of executing agent. This is a byte array; in the case of a message call, if the execution agent is a transaction, then the transaction data is included as input data.
5. The address of the account that initiated the code execution or transaction sender. This is the address of the sender in case the code execution is initiated by a transaction; otherwise, it's the address of the account.
6. The value or transaction value. This is the amount in **Wei**. If the execution agent is a transaction, then it is the transaction value.
7. The code to be executed presented as a byte array that the iterator function picks up in each execution cycle.
8. The block header of the current block
9. The number of message calls or contract creation transactions currently in execution. In other words, this is the number of CALLs or CREATEs currently in execution.

The execution environment can be visualized as a tuple of nine elements, as follows:



Q 4 : What are Ricardian Contracts? Compare and contrast Ricardian Contracts and Smart Contracts.


(10 Marks)

1. Definition of Ricardian Contracts (3 Marks)

A **Ricardian Contract** is a digital contract format that is **both human-readable and machine-readable**. It acts as a **legal agreement** that can be parsed and interpreted by computer systems while remaining understandable by humans.

- Introduced by **Ian Grigg** in 1996.
- It represents a **bridge between legal prose and code execution**.


- It contains:
 - Legal text (terms and conditions)
 - Digital signature
 - Cryptographic hash (for identity and immutability)

 Example: A Ricardian contract can define the issuance of a token with legally binding terms and be digitally signed by both parties.

2. Definition of Smart Contracts (2 Marks)

A **Smart Contract** is a **self-executing code** deployed on a blockchain (like Ethereum) that automatically enforces rules and conditions written in code.

- Introduced by **Nick Szabo** in the 1990s.
- It is not a legal document but a program.
- Deployed on blockchain, it ensures **automation**, **trustlessness**, and **immutability** of execution.

 Example: A smart contract for a crowdfunding campaign automatically refunds users if the target amount is not reached.

3. Comparison Table (4 Marks)

Feature	Ricardian Contract	Smart Contract
Nature	Legal agreement with machine readability	Self-executing code
Readability	Human-readable + machine-readable	Mostly code-based; not directly human-readable

Execution	Not automatically executed	Automatically executed on blockchain
Binding	Can be legally binding	Not inherently legally binding
Immutability	Cryptographically hashed for integrity	Stored on blockchain, immutable
Use Case	Financial instruments, token issuance	dApps, escrow, voting, DeFi
Platform	Any digital system	Blockchain platforms like Ethereum

4. Conclusion (1 Mark)

Ricardian Contracts serve as a **hybrid between legal and digital systems**, ensuring legal enforceability along with machine processability. In contrast, **Smart Contracts** focus purely on **automated execution** within a blockchain environment. Both play complementary roles in the future of **digital agreements and decentralized applications**.

Q 5: Discuss real-world applications of Hyperledger Fabric and Corda in industries like finance, healthcare, and supply chain management.

(10 Marks)

1. Introduction (1 Mark)

Hyperledger Fabric and **Corda** are two popular **permissioned blockchain platforms** designed for enterprise use.

- **Hyperledger Fabric** is developed by **The Linux Foundation**.
- **Corda** is developed by **R3**, focused on financial institutions.

Both platforms are used across various industries to improve **transparency**, **security**, and **efficiency**.

2. Applications in Finance (3 Marks)

◆ Hyperledger Fabric

- **Trade Finance:** Used by **we.trade** (IBM and major banks) to manage cross-border trade with real-time tracking and smart contracts.
- **Asset Tokenization:** Enables digital representation of financial assets and secure peer-to-peer transfers.

◆ Corda

- **Interbank Settlements:** Used by banks like **HSBC** and **ING** for real-time gross settlement and reconciliation.
 - **Know Your Customer (KYC):** Enables secure sharing of verified identity information without duplication or data leakage.
-

3. Applications in Healthcare (2 Marks)

◆ Hyperledger Fabric

- **Medical Records Management:** Enables secure and interoperable sharing of **Electronic Health Records (EHR)** among hospitals and patients.
 - Example: **MediLedger Project** for pharmaceutical traceability.
- **Clinical Trials:** Ensures immutability and auditability of trial data.

◆ Corda

- **Insurance Claims:** Used in health insurance to automate and validate claims processing between hospitals and insurers with minimal fraud.

4. Applications in Supply Chain Management (3 Marks)

◆ Hyperledger Fabric

- **Food Traceability:** Used by **Walmart and IBM Food Trust** to track produce from farm to shelf, ensuring quality and safety.
- **Logistics:** Ensures transparency across multiple vendors, carriers, and warehouses.

◆ Corda

- **Document Exchange:** Secure sharing of logistics documents like Bills of Lading, purchase orders, and invoices in a tamper-proof way.
- **Supply Chain Finance:** Allows manufacturers and suppliers to access financing based on verified delivery milestones.

5. Conclusion (1 Mark)

Both **Hyperledger Fabric** and **Corda** provide robust blockchain solutions tailored for enterprise applications.

- **Hyperledger Fabric** offers **modular architecture** suitable for diverse industries.
- **Corda** excels in **financial-grade applications** with a strong focus on privacy and interoperability.

Their adoption in real-world use cases is transforming traditional workflows by enhancing **trust, automation, and efficiency**.

Q 6 : Give the different types of transactions in Ethereum and also explain the fields included in these transactions.

(10 Marks)

1. Introduction (1 Mark)

In Ethereum, a **transaction** is a cryptographically signed message sent from one account to another. Transactions are the **building blocks of operations** like transferring ETH, deploying smart contracts, and calling contract functions.

2. Types of Transactions in Ethereum (3 Marks)


Ethereum supports **three main types of transactions**:

- ◆ **1. Regular Transactions (Ether Transfer)**
 - Transfers ETH from one **Externally Owned Account (EOA)** to another EOA.
 - ◆ **2. Contract Deployment Transactions**
 - Sent from an EOA without a **to** address but with **compiled bytecode** in the data field.
 - Deploys a new **smart contract** on the blockchain.
 - ◆ **3. Contract Invocation Transactions**
 - Sent to an existing **smart contract address**, usually with **input data** (function signature + arguments).
 - Executes contract functions like token transfers or updates.
-

3. Fields in an Ethereum Transaction (5 Marks)

Each Ethereum transaction includes the following key fields:

Field	Description
Nonce	Number of transactions sent from the sender's address. Prevents double-spending and ensures correct order.
Gas Price	Amount of ETH the sender is willing to pay per unit of gas (in Gwei). Affects transaction speed.
Gas Limit	Maximum gas allowed for transaction execution. Prevents infinite loops in smart contracts.
To	Recipient address (can be an EOA or smart contract address). Empty for contract creation.
Value	Amount of ETH (in wei) to send. 0 if calling a contract function without transfer.
Data	Optional field for input data (e.g., function call in smart contract). Contains bytecode for contract creation.
v, r, s	Components of the digital signature , used to validate the sender's identity.
Chain ID	Identifies the network (e.g., Mainnet = 1), added to prevent replay attacks.

 **Note:** In Ethereum post-London hard fork (EIP-1559), transactions may also include:

- **Max Priority Fee per Gas** (Tip)
 - **Max Fee per Gas** (Total fee cap)
-

4. Conclusion (1 Mark)

Ethereum transactions enable various functionalities such as **value transfer**, **contract deployment**, and **smart contract interaction**. Each transaction includes essential fields that define its behavior, cost, and validation. Understanding these components is crucial for working with Ethereum effectively.