Appendix N: Understanding Gas

Return to Reference Architecture (RA) or Return to Appendices

Overview

Return to Top

On Ethereum, each low-level operation available in the Ethereum Virtual Machine (EVM) is called an Operational Code (OPCODE). There are OPCODES for the following categories[1)]:

- Stop and Arithmetic Operations
- Comparison and Bitwise Logic Operations
- Accessing Environmental Information,
- Accessing Block Information
- Accessing Stack, Memory, Storage, and Flow Operations
- Executing Push Operations
- Performing Duplication Operations
- Conducting Exchange Operations
- Performing Logging Operations
- Executing System Operations

A spreadsheet is available documenting the Ethereum OPCODDES and the cost of each OPCODE: See EVM-OPCODE-GAS-COSTS.

Each OPCODE (i.e, ADD) within each category has an associated cost that is expressed in terms of Gas units. Gas is an abstract number that represents the relative complexity of operations. For example, the integer ADD (i.e., addition) operation uses 3 gas units while the Integer MUL (i.e.m multiplication) uses 5 gas units. Therefore the MUL OPCODE is more complex than ADD OPCODE.

Gas Units

Return to Top

The Smart Contract written in Solidity is compiled into a stream of EVM Bytcodes. EVM Bytecode The Ethereum Yellow Paper Appendix G, provides a cost for processing each

Gas Units are the calculated cost of every operation the EVM can perform.

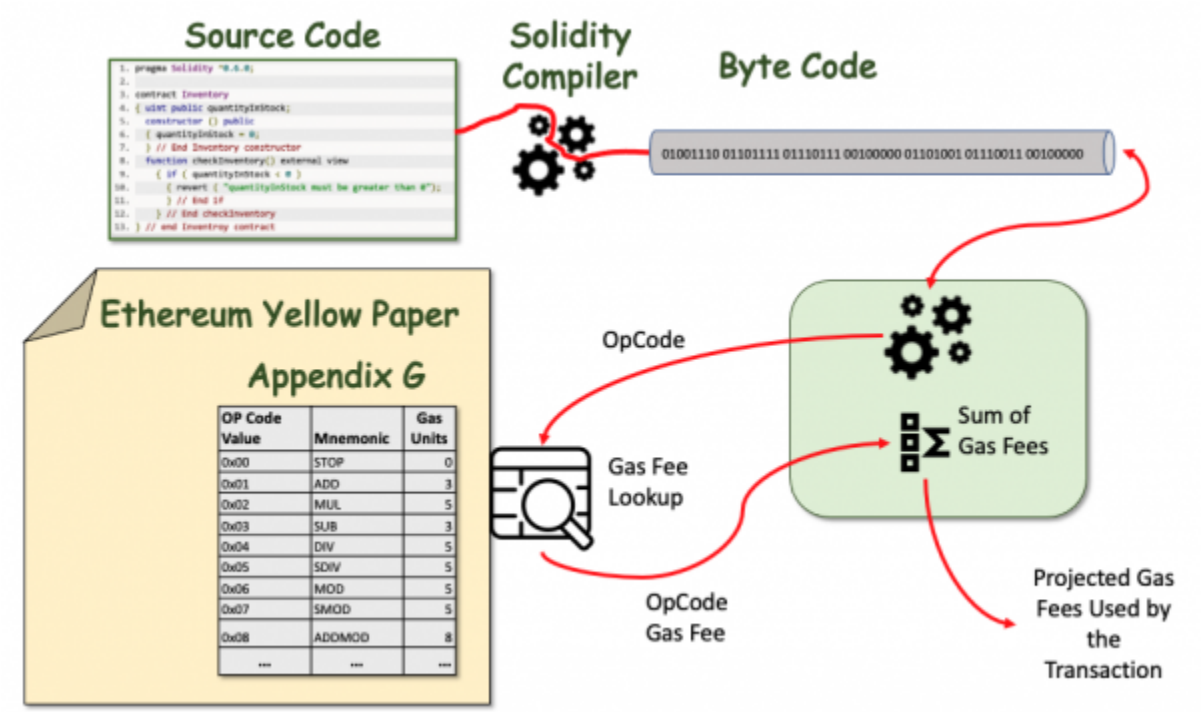


Figure 1: Schematic of how the Gas Fees for a Transaction are calculated.

Gas Price

Return to Top

The number of Gas units each operation consumes remains constant. However, the cost of the gas is dynamic and dictated by market conditions and reflects the amount each user is willing to pay to get their operations executed (i.e., the gas price expressed in Gwei which equals 0.000000001 ETH). The Gas value is attached to the user's transaction when the Transaction is submitted. Note, that as the price of an ETH varies, so will the gas price, but the amount of gas remains constant. In other words, the gas is associated with the relative cost of the operation and the gas price reflects how valuable the end-user thinks their operation is. The actual cost to the end-user is market-driven. The total fee the end-user pays is equal to **`gas_price * gas_used`**.

Miners are paid from the transaction fee and consequently, the miners to maximize profits prioritize transactions with a higher gas price. So, in essence, the higher the gas price you are willing to pay, the faster your transaction will be processed.

The following is a summary of an excellent article by Danny Ryan [2].

> *A miner executes the computation associated with each transaction being included in a block, resulting in an updated state. Upon successfully mining a block, a miner broadcasts the block to the network. Each of the other miners and non-mining nodes verifies the validity of the transactional computation and resulting state change before accepting the block as valid, incorporating the block into their copy of the blockchain, and moving on to the next block.*[3]

Processing Costs

Return to Top

The sample data shown in the first part of Table 1 rows 1-2, demonstrates the cost of Gas for performing simple **add** operations on the Ethereum blockchain. The first row in the table shows the cost of performing a single addition on the Ethereum Blockchain. Note: the same would occur for a single subtraction.

> **Line 1:** The cost to perform a simple addition once is about \$0.000002655 U.S. Dollars. This seems pretty inexpensive until the problem is expanded to do the same calculation 1 Million times.
> **Line 2:** The cost to perform a simple addition one million times is about \$26.550000000 U.S. Dollars. This becomes significant and should be of concern for most projects. As a comparison, using the Amazon Web Services (AWS) cloud and a Python program, adding the two numbers together one Million times takes about 0.04 seconds at a CPU cost of \$0.0059/hour making the calculations \$0.000001639/second or \$0.000000066 for the entire operation on AWS. The results are stunning: \$0.000000066 on AWS versus \$26.55 on Ethereum. The Ethereum calculation is about 400 Million times more expensive (or 40 Million if you are willing to pay a low gas price).

Data Storage

Return to Top

The sample data shown in the first part of Table 1 rows 3-5, demonstrates the cost of Gas for storing data on the blockchain. This could be a simple, single value such as the number of days until a Smart COntract expires (i.e., simple integer) or it could be something more ambitious such as a short story, a music file, or a video.

Storing data on the blockchain is extremely expensive. The reason for the cost is the storage of data on a blockchain is immutable and is replicated across tens of thousands of Ethereum Nodes in the Node Network. Therefore, uploading a short story, a music file, or a video onto the blockchain is cost-prohibitive:

> **Line 3:** The cost of storing 256-bit word is about \$0.171 U.S. Dollars. requires about 20,000 gas. This is about 6,000 times more expensive than adding two numbers together (see **Line 1**)
> **Line 4:** The cost of storing 1MB is about \$18.75
> **Line 5:** The cost of storing 1GB is about \$18750.00

Another bottleneck in storing large amounts of data is the current Block Gas Limit of approximately 4700000 gas/block. At this cap of gas per block, it would take over 132 blocks to write 1 MB of data to the blockchain, and that is assuming you can manage to hog all of the gas per block and that there are no other operations required!

Analysis of Cost Data

Return to Top

Table 1: The gas costs on Ethereum for calculation and storage on the blockchain.[4)]
Given: Median Gas Price on **23 August 2017**

- 1 Gwei equals 0.000000001 ETH)
- median gas price (28 Gwei)
- USD/ETH exchange rate ($295/ETH)

Processing								
Row Number * Task	**Gas required**	**Cost (ETH)**	**Cost (USD)**	**Ops per ETH**	**Ops per USD**	**Ops per Block**	**Blocks to complete OP**	
1	**Add or subtract two integers**	3	0.000000009	0.000002655	11111111.11	37664.78343	1566666.667	0.0000006382978230
2	**Add two Integers, 1 Million times**	3000000	0.09	26.550000000	11.1111111	0.037664783	1.566666667	0.638297872
Storage								
Task	**Gas required**	**Cost (ETH)**	**Cost (USD)**	**Ops per ETH**	**Ops per USD**	**Ops per Block**	**Blocks to complete OP**	
3	**Save a 256-bit word to Storage**	20000	0.0006	0.171	1666.666667	5.649711751	243	0.004255319
4	**Save 1MB to Storage (31250 256-bit words)**	625000000	18.75	5531.25	0.053333333	0.000180791	0.00752	132.9787234
5	**Save 1GB to Storage (31250 256-bit words)**	6250000000000	18750.00	5531250	0.00005333333	0.00000018079	0.00000752	132978.7234
Note: The data mentioned above are only true for a public Ethereum blockchain. The data could be totally different for private or permissioned blockchains.								

1)

Dr. Gavin Wood, ETHEREUM: A Secure Decentralized Gernerised Transaction Ledger, H.2 Instruction Set, BERLIN VERSION b2d0dbf Page: 30, 6 May 2022, Accessed: 11 May 2022, https://ethereum.github.io/yellowpaper/paper.pdf

2) , 3) , 4)

Danny Ryan, Hackernoon, , 29 May 2017, Accessed: 11 May 2022, https://hackernoon.com/ether-purchase-power-df40a38c5a2f

From:
https://omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:n_gas&rev=1654102346

Last update: **2022/06/01 12:52**