Adamnite Virtual Machine and PL research
February-May
Original Contributors: Archie Chaudhury, Ayo Adeskan, Aaron, Bibek Paul

The goal for this project is to research best practices for implementing a Virtual Machine (VM) and Smart Contracts Platform. While not meant to be final, this project is supposed to be a high-level implementation and set a foundation for the mainnet launch. The project will be split into 3 sub-milestones: VM research, VM high-level implementation, and Smart Contract Validation.

This document is meant to serve as a place to write down notes, talk about various implementations, suggest potentially good resources, and add diagrams/drawings. By the end, this document will hopefully serve as a base and resource for other contributors as the protocol is implemented. Each process, defined as two sub-milestones, is listed below; the beginning will typically contain helpful papers, books, or code repositories that were useful. The middle will usually contain some high-level implementations and provide a description of some of the unique innovations that separate Adamnite from previous smart contract based platforms. The conclusion will include a description of the final implementation, along with a link to any code repositories or demos.

As a final note, do note that this is mostly an exploration designed to take a deeper look at some of the protocols underlying the project, and serve as a base for subsequent research/development to be conducted on. It is by no means a final product, nor is it meant to finalize any of the core protocols surrounding the Adamnite protocol. This document will likely need to be adjusted over time as more contributors join and as the original contributors continue learning more. Just like biological animals, software is never finalized: it only reaches certain stages in its evolution.

VM Research and Implementation

The Adamnite VM, like VMs on other blockchain smart contract based platforms, is actually a distributed VM. A distributed VM is simply a VM that does not reside in one physical machine; rather, it is distributed across all the nodes in the network. Each node continuously reads the blockchain, and updates its state based on operation-calls made to the VM. Each node is also directly processing the VM to validate the operations and transitions to the state that are made by the smart contracts being processed by the network.
For Adamnite, the VM itself will be quite similar to other distributed VMs, with differences perhaps pointing to additional security measures and increased efficiency. On a high level, the Adamnite VM will need to define opcodes and memory, be able to process contracts compiled to

assembly, and be able to automatically execute this data directly on the blockchain. Below, some good beginning resources on VMs are outlined, along with links.

Create your own VM

Paper on VMs

Video on Cardano's VM Implementation

GoETH VM Implementation

Article on problems with Ethereum VM

Go VM Example

Java VM Example (Actually want to be closer to this, but with blockchain operations)

IOST VM Example

Feel free to add any additional resources that you might find helpful. The goal for the research period is to develop a solid understanding of Virtual Machines and gain enough knowledge to start developing a basic implementation for the Adamnite VM. Feel free to jot down any ideas, draft notes, or even start developing early on the Adamnite Github. The Notes Section below is for any notes or diagrams that might help guide what a potential implementation could look like.

Day 1 (Friday) will mostly focus on note taking and learning about virtual machine implementation. It will focus on developing some basic smart contract parsing techniques.

Day 2 (Saturday) will focus on developing the actual VM along with parsing for basic smart contracts. This will include basic operations, the processing of these operations, fee specification, and developing a basic connection to the distributed ledger.

Day 3 (Sunday) will focus on optimization and documenting protocol ideas to make the VM more efficient/effective.

The notes below are from a previous build session. Feel free to reference these along with the technical paper to help guide the development of the virtual machine.

Notes (organized by date)

2/16/22: The most important aspects of a virtual machine is the overall structure and the bytecodes assigned to it. The Adamnite VM will be stack-based; a majority of its bytecodes will be operations that correspond to adding or removing items from stack of items, along with calls to external environmental information. In that sense, the Adamnite VM should be very similar to VMs built on other blockchain based platforms. However, there will be some key differences. One innovation that we could actively look toward is implementing broader library support and API calls. That should be a primary goal, or achievement, of the build session. A high-level implementation that could separate Adamnite from other blockchain-based platforms is the inclusion of libraries and API calls that will make it a lot more efficient for developers.

2/17/22: The Adamnite Virtual Machine will essentially be updating the overall state of the blockchain: the various opcodes will essentially be defining various additions, retractions, and changes to the state of the network. The state can be thought of as a cryptographic snapshot of all the account's current balances and data at any given time. Changes to these accounts as dictated through on-chain transactions or smart contract application calls, are processed through blocks, which when committed executes a state transition function that transitions the blockchain from one state to another.

A set of proposed operations for the virtual machine to follow:

- Basic Arithmetic Operations (ADD, SUB, etc)
- Stack Operations (POP,PUSH,etc)
- Execution Operations (PAUSE,JUMP, etc)
- Account Operations (CREATE, INTERACT, etc)
- Block Operations (HASH, TIMESTAMP, VALIDATOR ADDRESS, etc)

Furthermore, the Adamntie Virtual Machine should allow for contracts to be modular; sub-routines and programs should be split into modules and be enabled to be downloaded effectively.

For pull-requests, please put your code in a new branch. This branch could be for an individual or be dedicated to multiple people working together.