**A Virtual Machine Simulating a 6502 Like Processor**

**Project Overview:**

The project aims to write and deploy a simple Virtual Machine (VM) which simulates a 6502-like processor. It simulates various hardware components, such as the CPU, memory, and I/O devices, allowing them to execute instructions, store and retrieve data, and interact with the external world, much like a real computer.

**Introduction:**

A VM is a program that acts like a computer. It simulates a CPU along with a few other hardware components, allowing it to perform arithmetic, read and write to memory, and interact with I/O devices, just like a physical computer.

**Why is VM used?**

1. VMs are useful for executing code in a secure or isolated way.
2. It can reproduce the behaviour of some specific computer which are not in production anymore. This is especially important for organizations that rely on older applications that may not be compatible with modern operating systems.
3. A VM could offer a standard platform which provides portability for all of them.

**Examples of some complex VM:** VMWare, VirtualBox, Qemu.

**NOTE**: These modern VM’s are complex in design and perform a very wide range of functionalities to simulate modern processors. Our project simulates a much simpler processor architecture (a 6502 like processor) with limited features.

**Project Scope:**

The scope of this project includes:

* Deciding the functionalities to be implemented in the VM.
* Writing the software to implement the functionalities.
* Testing the VMs to ensure they meet the project's objective.
* Running basic programs and checking the status of registers, etc.
* Documentation the entire process for reference.

**Design:**

### Processor

### Registers - A register is a slot for storing a single value on the CPU.

### General purpose register

### Flag register

### Program counter

### Stack pointer register

### ALU

### Stack Memory - The stack is a list of data words. It uses the Last In First Out (LIFO) access method which is the most popular access method in most of the CPU.

### Clock - The clock signal dictates the timing of instruction execution. Each instruction in a computer program takes a specific number of clock cycles to complete. The clock signal ensures that instructions are executed in the correct order and at the appropriate speed.

### Memory

### Mechanism of loading Instructions to the Memory

### 

**Functional Requirements –** Implementing various registers, main memory and the stack and, simulating the clock, Instruction Fetching and Execution mechanism.

**Non-Functional Requirements –** Tracker to keep track of the status of various registers, memory locations, etc when a program is executed.

**Tools:**

1. **Programming Language -** C, C++
2. **Version Control –** Git
3. **Utility –** Make, GDB Debugger