Project Name: Build a Virtual CPU Emulator

1. Defining the Project Scope:

To start, we met as a team to determine the exact scope of our emulator. Our goal is to create a virtual CPU that can mimic the basic operations of a physical CPU. Here's what we defined:

- **Core Operations:** Our emulator will handle essential arithmetic and logic functions, such as ADD, SUBTRACT, and logical comparisons.
- **Memory Management:** We aim to include basic memory handling capabilities, like reading and writing data, to simulate a CPU's interaction with memory.
- **Instruction Set:** We'll build a simple instruction set architecture (ISA), which defines the list of operations (like LOAD, STORE, etc.) the emulator will support.
- **I/O Handling:** We also want our emulator to perform simple input and output operations to simulate basic I/O interactions.

By defining these features early on, we created a roadmap to guide us in later stages. This clarity helps us avoid scope creep and lets us stay focused on completing core components each week.

2. Gathering Resources:

We researched and gathered materials that explain the foundational concepts of CPU emulation, which includes topics like:

- **CPU Architecture Basics:** Resources on the functioning of a real CPU, covering components such as the ALU (Arithmetic Logic Unit), registers, and the fetch-decode-execute cycle.
- **Python Resources for Emulation:** We explored Python-specific libraries like *`struct`* (for handling binary data), which will be critical in simulating machine-level operations, and *`argparse`* for command-line functionality.
- **Documentation & Emulation Examples:** We reviewed documentation on creating virtual environments, sample emulators, and basic assembly language tutorials to understand how to handle low-level operations.

We also bookmarked online documentation, tutorials, and videos, ensuring all team members have a common knowledge base to refer to during the project.

3. Setting up the Development Environment:

After some discussion, we decided on **Python** as our primary programming language. Python is well-suited to this project because:

- **Readability:** Python's clear syntax allows all team members to read and contribute code more easily, making collaboration smoother.
- **Flexibility:** Python's extensive libraries provide a variety of tools for handling the complex aspects of CPU emulation, like binary operations and memory management.

- **Speed of Development:** Python enables rapid prototyping, which will help us test CPU components quickly as we develop them week by week.

With the language decided, we set up our development environment. Each team member installed Python and set up an IDE (we agreed on either PyCharm or VS Code). We established a project directory structure on our local machines that mirrors the GitHub repo, making it easy for everyone to follow a standardized workflow.

4. Setting up Version Control:

We created a **GitHub repository** for the project, where we initialized the codebase with a *`README.md`* file to document the project's goals and scope. Here's how we organized the version control process:

- **Branching Strategy:** Each team member works on individual branches based on specific tasks (like ISA design, ALU implementation, etc.), which we merge into the main branch after review.
- **Commit Protocol:** We agreed on a standard for commit messages to ensure everyone documents their changes clearly, making it easy to track who did what.
- **Pull Requests:** We'll use pull requests to review each other's code before merging, ensuring that each contribution is reviewed and aligned with project goals.

Setting up this workflow early means that all team members can contribute to the project simultaneously without conflicting code changes.

By the end of Week 1, we had a solid project foundation: a clear scope, resources, a functional development environment, and an organized version control system. This setup will help us stay organized and allow us to focus on coding the emulator's core functions in the coming weeks.