Project Name: Build a Virtual CPU Emulator

Objective:

Create a foundational Instruction Set Architecture (ISA) for a virtual CPU for critical data processing, memory access, and control flow operations. This ISA will comprise a rudimentary assembler to convert human-readable assembly code into executable machine code, define the fundamental instructions, and create a standardized instruction format. The objective is to develop a basic yet useful framework for CPU operation that will serve as a foundation for further development and improvement.

1. Define Basic Instructions (ADD, SUB, LOAD, STORE, etc.).

We'll create a basic instruction set to perform arithmetic, logical, memory, and control operations.

Instruction Set

Mnemonic	**Operation**	**Description**
ADD	Add	Add contents of two registers or a register and an immediate value.
SUB	Subtract	Subtract contents of two registers or a register and an immediate value.
LOAD	Load	Load data from memory into a register.
STORE	Store	Store data from a register in memory.
MOV	Move	Move an immediate value or data between registers.
AND	Logical AND	Perform a bitwise AND operation.
OR	Logical OR	Perform a bitwise OR operation.
JMP	Jump	Unconditionally jump to a specific address.
BEQ	Branch if Equal	Branch to an address if two values are equal.
NOP	No Operation	Do nothing for one clock cycle.

2. Instruction Formats

1. **R-Type (Register to Register Operations)**
    - Instructions: ADD, SUB, AND, OR

- Format: [Opcode] [R1] [R2] [Unused (4 bits)]
    - **Opcode**: Binary code representing the operation.
    - **R1**: Binary representation of the first register (4 bits).
    - **R2**: Binary representation of the second register (4 bits).
    - **Unused**: Always set to 0000 (4 bits).

Example: ADD R1 R2

Machine Code: 0001 0001 0010

2. **I-Type (Immediate Operations)**
    - Instructions: LOAD, STORE
    - Format: [Opcode] [R1] [Immediate (8 bits)]
        - **Opcode**: Binary code representing the operation.
        - **R1**: Binary representation of the register (4 bits).
        - **Immediate**: 8-bit binary representation of the immediate value.

Example: LOAD R1 15

Machine Code: 0011 0001 00001111

3. **J-Type (Jump Operations)**
    - Instructions: JMP, BEQ
    - Format: [Opcode] [Address (8 bits)]
        - **Opcode**: Binary code representing the operation.
        - **Address**: 8-bit binary representation of the target address.

Example: JMP 10

Machine Code: 1000 00001010

4. **NOP (No Operation)**
    - Instruction: NOP
    - Format: [Opcode]

    - **Opcode**: Binary code representing the operation.
    - Contains no operands or additional bits.

Example: NOP

Machine Code: 1010