Definitions & Acronyms

- UVSim: A virtual machine that simulates the execution of BasicML programs.
- BasicML: A machine language executed by UVSim.
- Accumulator: A register storing intermediate results of computations.
- Opcode: The first two digits of a BasicML instruction that specify the operation.
- Operand: The last two digits of a BasicML instruction specifying a memory address.
- PC (Program Counter): Tracks the execution location of the current instruction.

Functional Requirements

- **User Interface**
    - The system shall provide a text input area for users to enter BasicML programs.
    - The system shall provide a Load button to store the entered program into memory.
    - The system shall display 100 memory locations in a table with each row showing:
        - Address (00-99)
        - Stored value (signed 4-digit number)
    - Users shall be able to manually edit memory values before execution.
    - The system shall have buttons to Run, Step, Halt, and Reset the program.
- Execution and Debugging
    - The system shall execute instructions sequentially from memory starting at location 00.
    - The system shall provide a Step button to execute one instruction at a time.
    - The system shall display the Accumulator value in real-time.
    - The system shall display the Program Counter (PC) as it updates during execution.
    - The system shall provide an error message and halt execution if an invalid instruction is encountered.
- Input & Output Operations
    - The system shall prompt users for input when the READ (10) instruction is encountered.
    - The system shall display output in a dedicated area when the WRITE (11) instruction is executed.
- Arithmetic and Memory Operations
    - The system shall support arithmetic operations on the accumulator:
    - ADD (30): Add a memory value to the accumulator.
    - SUBTRACT (31): Subtract a memory value from the accumulator.
    - MULTIPLY (33): Multiply the accumulator value with a memory value.
    - DIVIDE (32): Divide the accumulator value by a memory value (halt execution on division by zero).

    - The system shall support memory load/store operations:
    - LOAD (20): Load a memory value into the accumulator.
    - STORE (21): Store the accumulator value into a specific memory address.
- Control Flow Operations
    - The system shall allow branching using:
    - BRANCH (40): Jump to a specific memory location.
    - BRANCHNEG (41): Jump if the accumulator is negative.
    - BRANCHZERO (42): Jump if the accumulator is zero.
    - HALT (43): Stop execution.

Non-Functional Requirements

- Performance
    - The system shall execute a full BasicML program within 0.5 seconds of clicking "Run" for typical workloads.
- Usability
    - The GUI shall include properly labeled buttons, a clear layout, and tooltips explaining their functionality.
- Portability
    - The software shall be compatible with Windows, macOS, and Linux without modification