Functional

1. The system shall allow users to load a BasicML program into memory, starting at location 00.

2. The system shall provide a 100-word memory space where each word is a signed four-digit decimal number.

3. The system shall interpret and execute BasicML instructions according to their respective operation codes.

4. The system shall include an accumulator register for performing arithmetic and logical operations.

5. The system shall prompt users for input when the READ (10) instruction is encountered.

6. The system shall display output in a dedicated area when the WRITE (11) instruction is executed.

7. ADD (30): Add a memory value to the accumulator.

8. SUBTRACT (31): Subtract a memory value from the accumulator.

9. MULTIPLY (33): Multiply the accumulator value with a memory value.

10. DIVIDE (32): Divide the accumulator value by a memory value (halt execution on division by zero).

11. LOAD (20): Load a memory value into the accumulator.

12. STORE (21): Store the accumulator value into a specific memory address.

13. BRANCH (40): Jump to a specific memory location.

14. BRANCHNEG (41): Jump if the accumulator is negative.

15. BRANCHZERO (42): Jump if the accumulator is zero.

16. HALT (43): Stop execution.

17. The system shall provide an error message and halt execution if an invalid instruction is encountered.

18. The system shall halt execution when encountering a termination instruction or an unrecoverable error.

19. The system shall provide logs of executed instructions and memory changes for debugging purposes.

Non-Functional

1. The system shall execute a full BasicML program within 0.5 seconds of clicking "Run" for typical workloads.

2. The GUI shall include properly labeled buttons, a clear layout, and tooltips explaining their functionality.

3. The project shall have documentation for all functions and classes and ensure that each module is loosely coupled.