**Software Requirement Specification**

**Functional Requirements**

1.1 Loading and Running Programs

1. The system should allow users to enter a BasicML program.
2. The program should be loaded into memory, using no more than 100 memory slots.
3. If there are errors in the instructions, the system should show an error message.

1.2 Executing Instructions

1. The system should run instructions one by one.
2. It should fetch, understand, and carry out each instruction.
3. The system should support math, moving data, controlling the flow of the program, and input/output operations.
4. After each instruction, the system should update the program counter to keep track of what to do next.

1.3 Math and Data Handling

1. The program must add the value from a specified memory location to the accumulator.
2. The program must subtract the value from a specified memory location from the accumulator.
3. The program must multiply the accumulator by the value from a specified memory location.
4. The program must divide the accumulator by the value from a specified memory location, ensuring division by zero is prevented.
5. It should support loading values from memory into the accumulator and storing them back into memory.

1.4 Input and Output

1. The system should allow the user to enter numbers when the READ instruction is used.
2. It should display numbers on the screen when the WRITE instruction is used.

1.5 Controlling the Program Flow

1. The system should allow jumping to different parts of the program using BRANCH instructions.
2. It should stop running when the HALT instruction is used.

**Non-Functional Requirements**

2.1 Speed and Performance

1. The system should process each instruction quickly (within a second).
2. Moving data and running commands should not slow down the system.

2.2 Reliability and Error Handling

1. The system should prevent crashes by checking for mistakes before running the program.
2. If an error happens, the system should give a clear message instead of stopping unexpectedly.

2.3 Ease of Use

1. The system should have a simple command-line interface that is easy to use.
2. Messages and feedback should be clear and helpful for the user.

2.4 Compatibility

1. The system should be written in Python and work on any computer that can run Python.

2.5 Future Maintenance and Improvements

1. The code should be organized in a way that makes it easy to add new features.
2. Important parts of the system should have comments explaining how they work.