15 Functional Requirements

1. Users can step through the program execution one instruction at a time for debugging purposes.
2. A command should be available to display the current memory state and register values at any time.
3. The program must support saving and loading memory states to and from a file.
4. Error messages should be descriptive and provide guidance when invalid opcodes or inputs are encountered.
5. Users can set breakpoints at specific instructions to pause execution automatically.
6. The program should allow modifying memory values manually before execution starts.
7. Input validation must ensure only valid integer values are accepted when reading user input.
8. Executed instructions and their effects should be logged for debugging purposes.
9. A help menu should explain supported opcodes and their corresponding functions.
10. The software should prevent infinite loops by enforcing a maximum instruction execution limit.
11. Users should have the option to execute programs in different modes, such as normal or debug mode.
12. A command must be available to reset the memory and accumulator without restarting the program.
13. The program should allow users to choose between keyboard input and file input for data entry.
14. Program will execute in one continuous flow.
15. Invalid memory accesses should trigger an error instead of allowing unintended behavior.

3 Non-Functional Requirements

1. The program can run on any machine that supports python program language.
2. The simulator must handle invalid input by preventing crashes and providing error messages.
3. The system should be designed in an intuitive interface for non-technical users.