**UVSim Software Requirements Specification**

Functional Requirements

1. Load Program from File
   * The system shall allow users to load a program from a text file containing valid UVSim assembly instructions.
   * The system shall support various file encodings, including UTF-8 with or without BOM.
2. Save Program to File
   * The system shall allow users to save their written program as a text file.
   * The system shall provide options for the file format and encoding.
3. Edit Instructions
   * The system shall provide an interface for users to add, modify, and delete instructions in the program.
   * The system shall validate input to ensure it follows the UVSim instruction format (+/-NNNN).
4. Execute Program Step-by-Step
   * The system shall allow users to execute the program one instruction at a time for debugging purposes.
   * The system shall highlight the currently executing instruction in the memory display.
5. Execute Program Continuously
   * The system shall execute all instructions in the program sequentially until completion or an error occurs.
   * The system shall provide a way to pause continuous execution.
6. Display Memory State
   * The system shall provide a way for users to view the memory content before, during, and after execution.
   * The system shall update the memory display in real-time during program execution.
7. Modify Memory Contents
   * The system shall allow users to manually edit memory values directly in the memory display.
   * The system shall validate all manual inputs to ensure they follow the UVSim memory format.
8. Error Handling and Debugging Information
   * The system shall detect and report errors such as invalid opcodes, memory access violations, and divide-by-zero errors.
   * The system shall display error messages in a dedicated console area of the GUI.
9. GUI Controls for Program Management
   * The system shall provide buttons and menu options for loading, saving, editing, and executing programs.
   * The system shall include a reset button to return the simulator to its initial state.
10. Instruction Set Support
    * The system shall support the full UVSim instruction set, including:
      + Input/Output operations (READ, WRITE)
      + Load/Store operations (LOAD, STORE)
      + Arithmetic operations (ADD, SUBTRACT, DIVIDE, MULTIPLY)
      + Control operations (BRANCH, BRANCHNEG, BRANCHZERO, HALT)
11. User Input Interface
    * The system shall provide a modal dialog for user input when requested by a READ instruction.
    * The system shall validate input to ensure it meets the requirements of the UVSim architecture.
12. Execution Console
    * The system shall include a console display that shows program output and execution status messages.
    * The console shall maintain a history of program output and error messages for the current session.
13. Status Display
    * The system shall always display the current values of the accumulator and instruction pointer.
    * The system shall highlight changes to these values to make them more visible during execution.
14. Program Reset Functionality
    * The system shall provide a mechanism to reset the program execution to its initial state.
    * The reset function should clear the accumulator, reset the instruction pointer, and optionally clear memory.
15. Responsive Design
    * The system should have a responsive interface that adapts to different screen sizes and resolutions.
    * The memory display shall include scrolling capabilities to handle the entire memory space.

Non-Functional Requirements

1. Performance Requirements
   * The system shall process instructions with a delay of no more than 50ms per instruction, ensuring efficient operation even when utilizing 100% of memory capacity.
   * The GUI shall remain responsive during program execution, with updates occurring within 100ms.
2. Usability Requirements
   * The system should have a clear and intuitive GUI with labeled buttons, appropriate spacing, and visual feedback.
   * The system should provide visual cues to help users understand the state of program execution, including highlighting the current instruction.
   * The system shall be usable by individuals with basic computer skills without requiring specialized knowledge of compilers or virtual machines.
3. Reliability Requirements
   * The system should reject invalid user inputs such as:
     1. Entering non-numeric values where numeric input is expected.
     2. Referencing memory locations outside the valid range (0–99).
     3. Leaving required fields blank. These validations will be implemented using real-time checks and visual feedback (e.g., red border with tooltip explanation).
   * The system shall prevent submission of invalid data by disabling the "Run" or "Load" button until all inputs are valid, confirmed through test cases.
   * The system shall autosave the current program and memory state to a temporary file every 60 seconds and reload the last autosaved state upon restart after a crash, confirmed by simulating application failure.