Class Definition Document

Class Name: Memory

Purpose: Manages the memory array and related operations.

Attributes:

- **memory: List of integers representing the memory.**

Methods:

1. `__init__(self, size=100):` **Initializes the memory with a given size (default 100).**
    - **Preconditions: None**
    - **Postconditions: Initializes the memory with a size of 100 (or given size), filled with zeroes.**
2. `read(self, address):` **Reads the value from a specified memory address.**
    - **Preconditions:** `address` **is within the valid range of memory addresses.**
    - **Postconditions: Returns the value at the specified memory address.**
3. `write(self, address, value):` **Writes a value to a specified memory address.**
    - **Preconditions:** `address` **is within the valid range of memory addresses.** `value` **is an integer.**
    - **Postconditions: The value at the specified memory address is updated to** `value`**.**
4. `reset(self):` **Resets the memory to its initial state.**
    - **Preconditions: None**
    - **Postconditions: The memory is reset to its initial state, filled with zeroes.**
5. `__str__(self):` **Returns a string representation of the memory.**
    - **Preconditions: None**
    - **Postconditions: Returns a string representation of the memory.**

Class Name: CPU

Purpose: Manages the accumulator, instruction counter, and executes instructions.

Attributes:

- **accumulator: Integer representing the accumulator.**
- **instruction_counter: Integer representing the instruction counter.**
- **memory: Instance of the Memory class.**
- **outputs: List of strings to log the outputs.**
- **output_function: Function to handle output (used by GUI).**
- **WORD_SIZE: Integer representing the word size for overflow checks.**

Methods:

1. **`__init__(self, memory, output_function):` Initializes the CPU with memory and output function.**
    - **Preconditions: `memory` is an instance of the Memory class. `output_function` is a callable function.**
    - **Postconditions: Initializes the CPU with the given memory and output function.**
2. **`check_overflow(self, value):` Checks and handles overflow for arithmetic operations.**
    - **Preconditions: `value` is an integer.**
    - **Postconditions: Returns the value, adjusted for overflow if necessary.**
3. **`execute_instruction(self, instruction):` Executes a given instruction.**
    - **Preconditions: `instruction` is a valid instruction (integer within the expected opcode range).**
    - **Postconditions: Executes the given instruction and updates the CPU state accordingly.**
4. **Instruction-specific methods (e.g., `read`, `write`, `load`, `store`, `add`, `subtract`, `divide`, `multiply`, `branch`, `branchneg`, `branchzero`, `halt`):**
    - **Preconditions: `operand` is a valid operand (integer within the valid range for memory addresses or values).**
    - **Postconditions: The specific operation is performed, and the CPU state is updated accordingly.**
5. **`reset(self):` Resets the CPU to its initial state.**
    - **Preconditions: None**
    - **Postconditions: The CPU is reset to its initial state.**

Class Name: Simulator

Purpose: Coordinates the interaction between memory, CPU, and program loading.

Attributes:

- **memory: Instance of the Memory class.**
- **cpu: Instance of the CPU class.**
- **loader: Instance of the ProgramLoader class.**

Methods:

1. **`__init__(self, output_function):` Initializes the simulator with memory, CPU, and program loader.**
    - **Preconditions: `output_function` is a callable function.**
    - **Postconditions: Initializes the simulator with memory, CPU, and program loader instances. Sets up the output function for the CPU.**
2. **`load_program_from_file(self, file_path):` Loads a program from a specified file path and loads it into memory.**
    - **Preconditions: `file_path` is a valid path to a readable file containing the program.**
    - **Postconditions: Loads the program from the specified file and writes it into memory.**
3. **`load_program(self, program):` Loads a program (a list of instructions) into memory.**
    - **Preconditions: `program` is a list of integers representing instructions. The length of `program` does not exceed the memory size.**
    - **Postconditions: The program is loaded into memory, with each instruction written to consecutive memory addresses.**
4. **`run(self):` Executes the loaded program.**
    - **Preconditions: A valid program is loaded into memory.**
    - **Postconditions: Executes the loaded program until a halt condition is met or the end of memory is reached. Outputs the final accumulator value and memory state.**
5. **`reset(self):` Resets the simulator's memory and CPU to their initial states.**
    - **Preconditions: None**
    - **Postconditions: The simulator's memory and CPU are reset to their initial states.**

Class Name: SimulatorGUI

Purpose: Manages the graphical user interface for the simulator.

Attributes:

- **simulator: Instance of the Simulator class.**
- **root: Instance of the Tkinter root window.**
- **button_frame: Frame for holding buttons.**
- **load_button: Button for loading a program.**
- **run_button: Button for running the loaded program.**
- **quit_button: Button for quitting the application.**
- **output_text_label: Label for the output text area.**
- **output_text: Text widget for displaying output.**
- **scrollbar: Scrollbar for the output text widget.**

Methods:

1. **`__init__(self, simulator):` Initializes the GUI with the given simulator.**
    - **Preconditions: `simulator` is an instance of the Simulator class.**
    - **Postconditions: Initializes the GUI components and links them to the simulator.**
2. **`load_program(self):` Prompts the user to select a program file and loads it into the simulator.**
    - **Preconditions: A valid file selection dialog is shown to the user.**
    - **Postconditions: The selected program file is loaded into the simulator.**
3. **`run_program(self):` Runs the loaded program and displays the output in the text widget.**
    - **Preconditions: A valid program is loaded in the simulator.**
    - **Postconditions: The simulator runs the program, and the output is displayed in the text widget.**
4. **`output_function(self, message):` Handles output by displaying messages in the text widget and prompting the user for input if needed.**
    - **Preconditions: `message` is a string.**
    - **Postconditions: The message is displayed in the text widget. If user input is required, it is handled appropriately.**
5. **`start(self):` Starts the Tkinter main loop to run the GUI.**
    - **Preconditions: The GUI components are initialized.**
    - **Postconditions: The Tkinter main loop is started, and the GUI is running.**

Class Name: ProgramLoader

- **Purpose: Handles loading programs from files.**
- **Attributes: None**
- **Methods:**

- **`load_program_from_file(file_path)`: Static method to load a program from a file and return it as a list of integers.**