**Class Definition Document**

**1. UVSim Class**

The **UVSim** class acts as the main processing unit of the simulator, handling program execution, memory management, and user input processing.

**Attributes**

• **ui**: A reference to the GUI instance.

• **log** (list): Stores execution logs.

• **memSpace** (int): Tracks the next available memory location.

• **counter** (int): Tracks the current instruction being executed.

• **memory** (dict): A dictionary storing 100 memory locations, initialized with "0000".

• **accum** (Accumulator): An instance of the Accumulator class.

• **record** (bool): A flag for logging execution.

**Methods**

**Program Loading**

**fileInputToMemory(inputFile)**

**Purpose**: Reads a program from a file and loads it into memory.

**Inputs**:

• inputFile (str): The path to the input file.

**Returns**:

• bool: True if the file is successfully loaded, False if an error occurs.

**Program Execution**

**wordProcess(step=False)**

**Purpose**: Fetches, decodes, and executes an instruction.

**Inputs**:

• step (bool, optional): If True, pauses execution after each instruction for debugging.

**Returns**:

• None

**process\_input(input\_word)**

**Purpose**: Handles user input for **READ** instructions and stores it in memory.

**Inputs**:

• input\_word (str): User-provided input.

**Returns**:

• None

**stepProgram()**

**Purpose**: Executes a single instruction and updates the execution log.

**Inputs**:

• None

**Returns**:

• None

**Branching Instructions**

**branch(value)**

**Purpose**: Unconditionally jumps to the specified memory location.

**Inputs**:

• value (int): The memory address to jump to.

**Returns**:

• None

**branchneg(value)**

**Purpose**: Jumps to a memory location if the accumulator value is negative.

**Inputs**:

• value (int): The memory address to jump to.

**Returns**:

• None

**branchzero(value)**

**Purpose**: Jumps to a memory location if the accumulator value is zero.

**Inputs**:

• value (int): The memory address to jump to.

**Returns**:

• None

**Memory & Execution Log**

**inspectCurrent()**

**Purpose**: Returns the current execution state of the program.

**Inputs**:

• None

**Returns**:

• dict: A dictionary containing execution details.

**inspectMemory()**

**Purpose**: Prints all memory contents for inspection.

**Inputs**:

• None

**Returns**:

• None

**logDisplay()**

**Purpose**: Displays the execution log.

**Inputs**:

• None

**Returns**:

• None

**Saving & Quitting**

**getAccumulator()**

**Purpose**: Retrieves the current accumulator value.

**Inputs**:

• None

**Returns**:

• int: The current value of the accumulator.

**saveMemory()**

**Purpose**: Saves the current memory state and accumulator value to a file.

**Inputs**:

• None

**Returns**:

• None

**update\_console(message)**

**Purpose**: Updates the GUI console with a given message.

**Inputs**:

• message (str): The message to be displayed.

**Returns**:

• None

**quit()**

**Purpose**: Stops execution and closes the application.

**Inputs**:

• None

**Returns**:

• None

**Converting words:**

**convert\_to\_six\_digits(word)**

**Purpose:** Converts a 4-digit word to the proper 6-digit format

**Inputs:**

• word(str): the word to be converted

**Returns**:

• word(str): converted word

**save\_converted\_program(file\_path)**

**Purpose:** Saves the converted 6-digit program to a file

**Inputs:**

• file\_path(str): where the file will be saved

**Returns**:

• 0 if saved correctly or -1 if there was an error

**detect\_format(line)**

**Purpose:** Detects how many digits the word is

**Inputs:**

• line(str): line to be counted

**Returns**:

• length of line

**2. Accumulator Class**

The **Accumulator** class manages arithmetic operations and interacts with memory.

**Methods**

**read(loc, input\_word)**

**Purpose**: Reads user input into memory at a specific location.

**Inputs**:

• loc (int): The memory address where input is stored.

• input\_word (str): The input value.

**Returns**:

• None

**write(loc, sign)**

**Purpose**: Outputs a value from memory to the console.

**Inputs**:

• loc (int): The memory address to read from.

• sign (int): Determines if the value should be displayed with a sign.

**Returns**:

• None

**load(loc, sign)**

**Purpose**: Loads a value from memory into the accumulator.

**Inputs**:

• loc (int): The memory address to load from.

• sign (int): Determines the sign of the loaded value.

**Returns**:

• None

**store(loc, sign)**

**Purpose**: Stores the accumulator’s value in memory.

**Inputs**:

• loc (int): The memory address where the value is stored.

• sign (int): Determines the sign of the stored value.

**Returns**:

• None

**add(loc, sign)**

**Purpose**: Adds a value from memory to the accumulator.

**Inputs**:

• loc (int): The memory address to add from.

• sign (int): Determines the sign of the addition.

**Returns**:

• None

**subtract(loc, sign)**

**Purpose**: Subtracts a memory value from the accumulator.

**Inputs**:

• loc (int): The memory address to subtract from.

• sign (int): Determines the sign of the subtraction.

**Returns**:

• None

**divide(loc, sign)**

**Purpose**: Divides the accumulator by a memory value.

**Inputs**:

• loc (int): The memory address of the divisor.

• sign (int): Determines the sign of the division.

**Returns**:

• None

**multiply(loc, sign)**

**Purpose**: Multiplies the accumulator by a memory value.

**Inputs**:

• loc (int): The memory address of the multiplier.

• sign (int): Determines the sign of the multiplication.

**Returns**:

• None

**3. UVSimUI Class**

The **UVSimUI** class is a Kivy `Screen` that provides a fully interactive graphical user interface for the **UVSim** virtual machine. It allows users to load, step through, and execute assembly-like programs visually while monitoring memory and system state.

**Components**

**File Input:** Text input and file chooser to load program files. Optionally reset the simulation after execution.

**Console Output:** Scrollable terminal display showing logs, messages, and prompts.

**Console Input:** Text input field for user-entered values during program execution.

**Memory Table:** Editable grid displaying memory addresses and values. Highlights the current instruction address during execution.

**Accumulator Display:** Shows the current accumulator value during execution.

**Program Counter Display:** Shows the address of the currently executing instruction.

**Control Buttons:** Execute, Step, Save, Quit, and Settings for color configuration.

**Screen Tabs:** Spinner (dropdown) and button to switch between or add new simulation screens (tabs).

**Methods**

**file\_handler(instance):**

**Purpose:** Handles file path input or selection and loads it into the simulator memory.

**file\_chooser\_handler(instance):**

**Purpose:** Processes file chooser selection and updates the interface.

**execute\_handler(instance):**

**Purpose:** Executes the loaded program until halt.

**step\_handler(instance):**

**Purpose:** Executes a single instruction step in the program.

**save\_file\_handler(instance):**

**Purpose:** Opens a file chooser to save the current memory state.

**quit\_handler(instance):**

**Purpose:** Quits the simulation by calling the simulator's quit logic.

**update\_accumulator(value):**

**Purpose:** Updates the accumulator text display.

**update\_program\_counter(value):**

**Purpose:** Updates the program counter field in the interface.

**console\_insert\_text(message):**

**Purpose:** Appends output to the console output box and scrolls to latest.

**focus\_console\_input():**

**Purpose:** Enables and focuses the console input field for user entry.

**refresh\_memory\_table(highlight\_index=None):**

**Purpose:** Rebuilds the memory grid view, highlighting the current instruction.

**validate\_and\_store(key, instance):**

**Purpose:** Validates and stores a word value edited by the user in memory.

**make\_reset\_button():**

**Purpose:** Converts the file selection button into a reset control post-execution.

**reset\_handler(instance):**

**Purpose:** Restores the GUI and simulator to its initial state for a new run.

**update\_screen\_spinner(screen\_names):**

**Purpose:** Updates the tab selector with the current screen names.

**add\_new\_screen(instance):**

**Purpose:** Adds a new UVSim screen (tab) to the application.

**on\_spinner\_select(spinner, text):**

**Purpose:** Changes the active screen based on tab selection.

**4. MyUVSimApp Class**

This class initializes the application and connects **UVSim** with **UVSimUI**.

**Methods**

**build()**

**Purpose**: Creates the UI and links it to the simulator.

**Returns**:

• None

**add\_new\_screen()**

**Purpose:** Create new screen and add it to the screen manager

**Inputs:**

• None

**Returns**:

• None

**update\_all\_spinners()**

**Purpose:** Updates the spinners with the proper screen names

**Inputs:**

• None

**Returns**:

• None