UVSim Design Document

# **UVSim**

UVSim is a simulator used by computer science students to learn Basic Machine Language. It provides a virtual environment to learn about memory, registers, and machine-level operations. It uses four digit numbers to interpret and execute these operations.

# **User Stories**

* **Student:** As a student, I want to load and execute a BasicML program in the simulator so I can learn and understand how machine language instructions work.
* **Instructor:** As an instructor, I want to track errors and executions so I can explain the behavior of a program and debug for my students.

# **Use Cases**

* **Read word into memory**
  + **Description:** Reads a signed four digit number, stores it in specified memory location
  + **Operation:** 10
  + **Steps:**
    1. Validates number
    2. Checks memory location, if in range 0-99 and free
    3. Stores at location
* **Write word to screen**
  + **Description:** Retrieves word from specified memory location and displays
  + **Operation:** 11
  + **Steps:**
    1. Validates memory location: 00-99
    2. Retrieves data
    3. Outputs
* **Arithmetic operation**
  + **Description:** Uses accumulator class to perform arithmetic (ADD, SUBTRACT, MULTIPLY, DIVIDE)
  + **Operation:** 30 - ADD, 31 - SUBTRACT, 32 - DIVIDE, 33 - MULTIPLY
  + **Steps:**
    1. Fetched operation code: first two digits
    2. Retrieves operand from memory address: last two digits
    3. Validates memory within range and occupied: 00-99
    4. Word at address retrieved
    5. Operation performed using accumulator and retrieved word
    6. Accumulator updated with result
* **Branch**
  + **Description:** The program starts counter at specified location
  + **Operation:** 40 - BRANCH, 41 - BRANCHNEG, 42 - BRANCHZERO, 43 - HALT
  + **Steps:**
    1. Fetched branch instruction and operand (memory address)
    2. Validates memory address: 00-99
    3. Counter updated based on instruction on address
* **Error handling**
  + **Description:** Simulator detects errors and outputs message, decides if program can continue
  + **Steps:**
    1. Detects error during execution
    2. Displays error message
    3. Halt program or prompt user to enter correct input
* **Memory dumping**
  + **Description:** Simulator displays all contents in memory for inspection
  + **Steps:**
    1. User issues a command to dump memory
    2. System displays memory values in rows with their memory address (e.g. 00: +1234)
* **Instruction inspection**
  + **Description:** Simulator displays the current instruction, state of accumulator, and program counter
  + **Steps:**
    1. User issues command to inspect current state
    2. System retrieves and displays:
       - Current instruction (decoded operation and operand)
       - Current value of accumulator
       - Current program counter
* **Load program**
  + **Description:** User loads a program from text file into memory
  + **Steps:**
    1. User specifies the input file containing the BasicML program
    2. System validates the file format
    3. System loads the instructions into memory starting at address 00
* **Execute program**
  + **Description:** Simulator executes the BasicML program sequentially
  + **Steps:**
    1. Counter starts at 00
    2. Fetches instruction at current program counter
    3. Decoded instruction and executes it
    4. Counter is updated
    5. Steps repeat until HALT instruction is reached or error thrown
* **Reset program**
  + **Description:** Simulator resets its state, clears memory, clears accumulator and clears program counter
  + **Steps:**
    1. User issues reset command
    2. System clears all memory contents
    3. Accumulator reset to zero
    4. Counter reset to 00
* **Save/load program**
  + **Description:** User can save current state and load up an older saved state
  + **Steps:**
    1. Save:
       - Save command issued by user
       - System writes memory, accumulator, and counter values to file
    2. Load:
       - Issues load command and specifies saved file
       - Reads file and restores memory, accumulator and program counter
* **Log execution**
  + **Description:** Simulator logs all executed instructions for debugging and analysis
  + **Steps:**
    1. System maintains log during program execution
    2. Each instruction, system records:
       - Instruction and address
       - Updated state of accumulator and memory
    3. Log displayed or saved upon program termination