UVSim Design Document

Overview

UVSim is a software simulator designed to help computer science students learn machine language and computer architecture. It allows users to write, load, and execute programs in BasicML, a simple machine language. UVSim emulates a basic virtual machine with a CPU, an accumulator, a memory unit, and a set of predefined instructions.

System Components

1. **Memory (UVSimMemory)**: Stores program instructions and data.
2. **Operations (UVSimOperations)**: Implements the execution of BasicML instructions.
3. **Virtual Machine (UVSim)**: Controls program execution, manages the instruction cycle, and interacts with memory and operations.

User Stories

Story 1: Running a BasicML Program

As a computer science student, **I want to** enter a BasicML program into UVSim, **so that** I can execute it and observe its behavior.

Story 2: Debugging a Program

As a student, **I want to** view the memory contents at any time, **so that** I can debug my BasicML program effectively.

Use Cases

1. Load a BasicML Program

Actor: User
Description: The user inputs a list of BasicML instructions, which are loaded into memory.
Precondition: The program consists of valid BasicML instructions.
Postcondition: The program is stored in UVSim memory starting at location 00.

2. Execute a BasicML Program

Actor: User
Description: The user runs the loaded program. UVSim executes instructions sequentially.
Precondition: A valid program is loaded in memory.
Postcondition: The program executes until a HALT instruction is encountered.

3. Read Input

Actor: User
Description: The user provides input when prompted by a READ instruction.
Precondition: The program has a READ instruction.
Postcondition: The input is stored in memory at the specified location.

4. Write Output

Actor: User
Description: The UVSim prints a value stored in memory to the screen.
Precondition: The program has a WRITE instruction.
Postcondition: The value at the specified memory location is displayed.

5. Load a Value into the Accumulator

Actor: System
Description: A LOAD instruction loads a memory value into the accumulator.
Precondition: The specified memory location contains a valid value.
Postcondition: The accumulator holds the loaded value.

6. Store a Value in Memory

Actor: System
Description: A STORE instruction saves the accumulator value into memory.
Precondition: The accumulator holds a valid value.
Postcondition: The value is stored in the specified memory location.

7. Perform Arithmetic Operations

Actor: System
Description: ADD, SUBTRACT, MULTIPLY, or DIVIDE instructions modify the accumulator.
Precondition: The specified memory location contains a valid number.
Postcondition: The accumulator reflects the new computed value.

8. Branch to Another Instruction

Actor: System
Description: A BRANCH instruction changes program execution to another location.

Precondition: The specified memory location is a valid instruction.
Postcondition: The program counter is updated.

9. Conditional Branching

Actor: System
Description: BRANCHNEG or BRANCHZERO checks accumulator conditions before jumping.
Precondition: The accumulator holds a value.
Postcondition: The program counter updates only if conditions are met.

10. Halt Execution

Actor: System
Description: A HALT instruction stops execution.
Precondition: The program is running.
Postcondition: Execution ceases, and control returns to the user.

11. Display Memory Contents

Actor: User
Description: The user requests to view memory contents.
Precondition: Memory contains values.
Postcondition: Memory contents are displayed.

12. Handle Invalid Instructions

Actor: System
Description: UVSim detects an invalid instruction and raises an error.
Precondition: The program contains an invalid opcode.
Postcondition: The system displays an error message and halts execution.

13. Handle Invalid Memory Access

Actor: System
Description: UVSim prevents access to invalid memory locations.
Precondition: A memory address is out of range.
Postcondition: An error message is displayed, and execution stops.

14. Prevent Division by Zero

Actor: System
Description: The simulator checks for division by zero.
Precondition: A DIVIDE instruction is executed.
Postcondition: If division by zero is attempted, an error is raised.

15. Exit the Simulator

Actor: User
Description: The user terminates the UVSim program.
Precondition: UVSim is running.
Postcondition: The program exits, releasing system resources.

Conclusion

UVSim provides an interactive way for students to understand machine language and CPU operations. It allows users to write, load, execute, and debug BasicML programs with a simple and intuitive interface.