User Stories

User Story 1: As a Computer Science Student
Title: Executing Basic Machine Language Programs for Learning Purposes

Narrative:
As a computer science student,
I want to be able to load and execute BasicML programs using the UVSim simulator,
So that I can learn how machine language instructions are executed in a computer architecture.

Acceptance Criteria:

- The student can load a BasicML program from a text file into the UVSim simulator.
- The simulator executes the loaded program, correctly processing each BasicML instruction.
- The student can view the results of each instruction, including any changes to the memory and accumulator.
- The simulator provides clear error messages for invalid instructions or runtime errors.

User Story 2: As a Computer Science Instructor
Title: Providing an Interactive Learning Tool for Machine Language Concepts

Narrative:
As a computer science instructor,
I want to provide my students with an interactive tool like the UVSim simulator,
So that they can experiment with and understand the concepts of machine language and basic computer operations.

Acceptance Criteria:

- The instructor can use the simulator to demonstrate the execution of BasicML programs.
- The simulator includes features to step through each instruction, allowing for instructional pauses and explanations.
- The simulator accurately represents key concepts of machine language execution, such as memory management and arithmetic/logical operations.
- The tool is user-friendly and intuitive for students who are new to machine language concepts.

Use Cases

1. Use Case: Read Program from File

Actor: User (Student/Instructor)

Description:
The use case involves loading a BasicML program from a text file into UVSim's memory.

Main Steps:

1. The user starts the UVSim application.
2. The application prompts for the file path of the BasicML program.
3. The user inputs the file path.
4. The system validates the file path and reads the file.
5. The BasicML program is parsed and loaded into UVSim's memory.
6. The system confirms successful loading to the user, indicating the program is ready for execution.

Alternate Flows:

1. Invalid File Path: The system requests a new file path if the provided one is invalid.
2. Invalid File Content: If the file does not contain a valid BasicML program, the system notifies the user and halts the loading process.

2. Use Case: Execute READ Instruction

Actor: User (Student/Instructor)

Description:
Perform a READ operation to input a number from the keyboard into a specified memory location in UVSim.

Main Steps:

1. UVSim encounters a READ instruction.
2. The user is prompted to enter a number.
3. The entered number is validated and stored in the specified memory location.
4. The simulator moves to the next operation.

Alternate Flows:

1. Invalid Input: On non-numeric or invalid input, re-prompt for a valid number.
2. Memory Address Validation: Display an error and halt/skip if the memory address is invalid.

3. Use Case: Execute WRITE Instruction

Actor: User (Student/Instructor)

Description:
Execute the WRITE operation to display a word from a specific memory location onto the screen.

Main Steps:

1. The UVSim simulator encounters a WRITE instruction (BasicML code 11) during the execution of a program.
2. The simulator accesses the specified memory location as indicated by the operand of the WRITE instruction.
3. The value stored at this memory location is retrieved.
4. The simulator displays the retrieved value on the screen.
5. The simulator proceeds to the next instruction or operation.

Alternate Flows:

1. If the specified memory location is invalid or out of range, the simulator displays an error message.
2. Depending on the implementation, the simulator may either halt the execution or skip to the next instruction in case of an invalid memory location.

4. Use Case: Execute LOAD Instruction

Actor: User (Student/Instructor)

Description:
Perform a LOAD operation to transfer a word from a specific memory location to the accumulator in UVSim.

Main Steps:

1. UVSim encounters a LOAD instruction.
2. The word from the specified memory location is retrieved.
3. The retrieved word is loaded into the accumulator.
4. UVSim proceeds to the next instruction.

Alternate Flows:

1. If the specified memory location is invalid or out of range, the simulator displays an error message and may halt or move to the next instruction.

5. Use Case: Execute STORE Instruction

Actor: User (Student/Instructor)

Description:

Execute the STORE operation to transfer a word from the accumulator to a specific memory location in UVSim.

Main Steps:
1. UVSim encounters a STORE instruction.
2. The word in the accumulator is stored in the specified memory location.
3. UVSim continues to the next instruction.

Alternate Flows:
1. If the specified memory location is invalid or out of range, UVSim displays an error and may either halt or proceed to the next instruction.

6. Use Case: Execute ADD Instruction

Actor: User (Student/Instructor)

Description:
Perform an ADD operation to sum a word from a specific memory location with the word in the accumulator, leaving the result in the accumulator.

Main Steps:
1. UVSim encounters an ADD instruction.
2. The word from the specified memory location is added to the word in the accumulator.
3. The result is stored in the accumulator.
4. UVSim proceeds to the next instruction.

Alternate Flows:
1. If the specified memory location is invalid or out of range, UVSim displays an error and may halt or continue to the next instruction.

7. Use Case: Execute SUBTRACT Instruction

Actor: User (Student/Instructor)

Description:
Execute the SUBTRACT operation to deduct a word from a specific memory location from the word in the accumulator, leaving the result in the accumulator.

Main Steps:
1. UVSim encounters a SUBTRACT instruction.
2. The word in the accumulator is reduced by the word from the specified memory location.
3. The result is updated in the accumulator.

4. UVSim moves to the next instruction.

Alternate Flows:

1. If the specified memory location is invalid or out of range, UVSim displays an error and may either halt or proceed to the next instruction.

8. Use Case: Execute DIVIDE Instruction

Actor: User (Student/Instructor)

Description:
Perform a DIVIDE operation to divide the word in the accumulator by a word from a specific memory location, leaving the result in the accumulator.

Main Steps:

1. UVSim encounters a DIVIDE instruction.
2. The word in the accumulator is divided by the word from the specified memory location.
3. The result is updated in the accumulator.
4. UVSim proceeds to the next instruction.

Alternate Flows:

1. Divide by Zero: If the word at the specified memory location is zero, UVSim displays a divide by zero error and may halt or continue to the next instruction.
2. Invalid Memory Location: If the specified memory location is invalid or out of range, UVSim displays an error and may halt or proceed to the next instruction.

9. Use Case: Execute MULTIPLY Instruction

Actor: User (Student/Instructor)

Description:
Execute the MULTIPLY operation to multiply a word from a specific memory location with the word in the accumulator, leaving the result in the accumulator.

Main Steps:

1. UVSim encounters a MULTIPLY instruction.
2. The word from the specified memory location is multiplied by the word in the accumulator.
3. The product is stored in the accumulator.
4. UVSim continues to the next instruction.

Alternate Flows:

1. If the specified memory location is invalid or out of range, UVSim displays an error and may either halt or proceed to the next instruction.

10. Use Case: Execute BRANCH Instruction

Actor: User (Student/Instructor)

Description:
Execute the BRANCH operation to change the program execution flow to a specific memory location.

Main Steps:

1. UVSim encounters a BRANCH instruction.
2. The simulator changes its current execution point to the memory location specified by the operand of the BRANCH instruction.
3. Execution continues from the new memory location.

Alternate Flows:

1. If the specified memory location for branching is invalid or out of range, UVSim displays an error and may halt or skip the instruction.

11. Use Case: Execute BRANCHNEG Instruction

Actor: User (Student/Instructor)

Description:
Execute the BRANCHNEG operation to branch to a specific memory location if the accumulator's value is negative.

Main Steps:

1. UVSim encounters a BRANCHNEG instruction.
2. The simulator checks if the accumulator's value is negative.
3. If negative, UVSim changes the current execution point to the specified memory location.
4. If not negative, the simulator continues to the next instruction.

Alternate Flows:

1. If the specified memory location for branching is invalid or out of range, UVSim displays an error and may halt or skip the instruction.

12. Use Case: Execute BRANCHZERO Instruction

Actor: User (Student/Instructor)

Description:
Perform the BRANCHZERO operation to branch to a specific memory location if the accumulator's value is zero.

Main Steps:

1. UVSim encounters a BRANCHZERO instruction.
2. The simulator checks if the accumulator's value is zero.
3. If zero, UVSim changes the current execution point to the specified memory location.
4. If not zero, the simulator continues to the next instruction.

Alternate Flows:

1. If the specified memory location for branching is invalid or out of range, UVSim displays an error and may either halt or skip the instruction.

13. Use Case: Execute HALT Instruction

Actor: User (Student/Instructor)

Description:
Execute the HALT operation to pause or stop the execution of the program in the UVSim simulator.

Main Steps:

1. UVSim encounters a HALT instruction.
2. The simulator stops executing further instructions and pauses the program.
3. The current state of the program is maintained, allowing for potential resumption or review.

Alternate Flows:

1. There are no alternate flows for the HALT instruction as its sole purpose is to halt program execution.