Group E UVSim Project

Course: CS 2450 – Software Engineering Instructor:

Professor Kevin Burtt

Semester: Fall 2025

**UVSim – CS2450 Project**

This project was created for CS 2450 to simulate how a basic computer processes machine instruction.

UVSim is a small program that acts like a simple computer. It has 100 memory slots, an accumulator (used for math), and runs instructions written in a language called BasicML.

Programs are plain text files with signed 4-digit numbers. The first two digits are the command, and the last two digits are the memory location. The program starts running at memory address 00 and stops when it hits a HALT (43xx) instruction.

This version now includes a graphical user interface (GUI) built with Tkinter, so you can open and run BasicML programs without using the console.  
The GUI makes it easy to load a file, run the program, and see results right on the screen.

**What You Need**

* Python 3.10 or newer
* Tkinter (included with most Python installs)
* BasicML program text files (like Test1.txt or Test2.txt)

**How the GUI Works**  
The UVSim window is simple and easy to use.

|  |  |
| --- | --- |
| **Menu** | **Function** |
| ****File → Open**** | Opens a .txt BasicML file from any folder. The file’s contents appear in the large center text box (instruction editor). |
| ****File → Save / Save As**** | Saves your current program. “Save As” lets you choose a new name or location. |
| ****File → Close**** | Exits the program safely. |
| ****Settings → Change Color Scheme**** | Opens the color picker so you can choose a primary and off color for the app. Your colors are saved to config.json. |
| ****Settings → Reset to Default Colors**** | Resets the interface to UVU’s green (#4C721D) and white (#FFFFFF) color scheme. |
| ****Instruction Editor** (Center Box)** | The main text area where all BasicML instructions are shown. You can type new lines, edit commands, or delete lines here. |
| **Output Panel (Right Side)** | Displays messages, errors, and program output during and after execution. |
| **Input Box (Bottom)** | Used when a program asks for input (READ command). Type a number and press Enter or click Enter Button.. |
| **Run Button** | Runs the program loaded in the Instruction Editor. Shows results in the Output Panel. |
| **Reset Button** | Clears both the Instruction Editor and Output Panel so you can start fresh. |
| **Enter Button** | Sends your input value to the program. Works the same as pressing the Enter key. |

**Output** **Box:**

Displays printed results (from WRITE commands) and any errors during the program’s execution.

If your test files are in a different folder, you can move them next to main.py or select them manually through **Open File**.

**How to Run UVSim**

1. Open your terminal and run:

|  |
| --- |
| * python3 main.py |

1. The GUI window will open.
2. Click Open File and pick a BasicML text file (like Test1.txt).
3. Click Run Program to start running it.
4. When asked, enter a number in the pop-up box and click Enter.
5. You’ll see the output in the main text area.
6. Click Clear Output to reset or Exit to close the program.

**Error Handling**

UVSim includes error handling so it doesn’t crash when something goes wrong.

|  |  |
| --- | --- |
| Error Type | What Happens |
| Malformed File | If the file has bad or missing instructions, it shows an error message. |
| Division by Zero | Stops the program and shows a warning. |
| Invalid Input | Asks you to enter a number between −9999 and +9999. |
| Overflow | If a number is too big, it only keeps the last 4 digits (for example, 12345 → 2345). |

**BasicML Commands**

Each instruction takes one line in the text file and is read from top to bottom.

|  |  |  |
| --- | --- | --- |
| Code | Command | What it does |
| 10xx | READ | Ask the user for a number and save it in memory[xx] |
| 11xx | WRITE | Print value from memory[xx] |
| 20xx | LOAD | Load memory[xx] into the accumulator |
| 21xx | STORE | Save the accumulator into memory[xx] |
| 30xx | ADD | Add memory[xx] to the  accumulator (truncated if too big) |
| 31xx | SUBSTRACT | Subtract memory[xx] from the accumulator (truncated if too big) |
| 32xx | DIVIDE | Divide accumulator by  memory[xx] (error if divisor = 0) |
| 33xx | MULTIPLY | Multiply accumulator by  memory[xx] (truncated if too big) |
| 40xx | BRANCH | Jump to memory[xx] |
| 41xx | BRANCHNEG | Jump to memory[xx] if accumulator < 0 |
| 42xx | BRANCHZERO | Jump to memory[xx] if  accumulator == 0 |
| 43xx | HALT | Stop the program |

**Example Programs**

**Test1.txt**

A screenshot of a computer

AI-generated content may be incorrect.

This program asks for two numbers, adds them, and prints the result.

**Test2.txt**

A screenshot of a computer

AI-generated content may be incorrect.

This program asks for two numbers and prints the bigger one.

**InvalidFileTest.txt**

Used to test UVSim’s ability to handle broken or incomplete files.

**Color Customization**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A green and white sign

AI-generated content may be incorrect.**

## **Editing and Saving Files**

After loading a BasicML file, you can edit it directly in the GUI. You can add, modify, or delete lines using buttons in the interface. Use 'Save File' to save your changes under a new or existing name. Files can be saved anywhere on your computer through the file dialog.

## **Known Limitations**

* Currently supports one open file at a time.
* Cut, copy, and paste for multiple rows may be limited.
* Future updates will separate functions into smaller classes for easier maintenance.

**Wireframe 1**

A screenshot of a computer

AI-generated content may be incorrect.

**Wireframe 2**

A screenshot of a computer

AI-generated content may be incorrect.

These images show the GUI layout including buttons, menus, and text boxes for user interaction.

**Project Structure  
CS2450\_Project/**

* main.py
* gui.py
* uvsim.py
* tests/
* test\_arithmetic.py
* test\_control.py
* test\_io.py
* test\_load\_store.py
* Test1.txt
* Test2.txt
* InvalidFileTest.txt
* docs/
* README.pdf
* SRS.docx
* UVSim\_Wireframe.pdf
* UVSim Class Document.pdf
* meetings/
* Unit\_Tests.csv

**Future Improvements**

For the next milestone, we plan to:

* **Expand memory & format**
  + Increase memory to 250 lines (000–249).
  + Switch to 6-digit words (keep sign) and 3-digit addresses.
  + Prepend a zero to opcodes (e.g., 010 = READ).
* **Strict limits & validation**
* Block files with >250 lines.
* Reject any instruction that points outside 000–249.
* Keep overflow truncation for 6-digit math.
* **Old vs. new files**
* Support both formats and add one-way converter (4-digit → 6-digit).
* Conversion rules:
* If first two digits are a valid opcode → 0XX0AA (e.g., 1007 → 010007).
* Otherwise treat as number → 00NNNN (e.g., 5555 → 005555).
* **Multi-file editing**
* Open multiple files at once (tabs or sub-windows).
* Allow switch/edit between files; only one file runs at a time.
* Disable switching while a program is running.
* **GUI updates**
* Add tabs for files.
* Show format badge (4-digit / 6-digit) per tab.
* Add Convert to 6-digit action.
* Keep color theme support (UVU default, configurable).
* **Engine & design**
* Refactor to modules (e.g., FileLoader, Converter, Memory, CPU, Executor, TabManager).
* Centralize validation and limits.
* **Testing**
* New unit tests for:
* 6-digit math with overflow
* Address validation (in/out of range)
* File length limit (≤250)
* Conversion edge cases (opcode vs number)
* Multi-tab open/save/run flows
* **Docs to update**
* README (6-digit format, tabs, conversion).
* Use cases & SRS (new features).
* Class diagram (new classes).
* Wireframe (tabs + convert button).
* Meeting reports.