USER MANUAL

M.Abdullah – 2022323

Ahmed Naeem – 2022069

Arsalan Khan – 2022115

M.Saadullah - 2022420

4 BIT OP-CODE SIMULATOR

**Table of Contents**

1. [Introduction](#Introduction)
2. [Getting Started](#gettingstarted)
   1. [Where to get the Simulator](#wheretoget)
   2. [Launching the Simulator](#launchingsimulator)
   3. [Instruction Set](#instructionset)
3. [Simulator GUI Layout](#GUIlayout)
4. [Using the Simulator](#usingsimulator)
5. [Conclusion](#conclusion)

**Introduction**

This manual provides instructions on how to use the Simulator. The Simulator is a graphical user interface (GUI) that allows you to load, run, and step through code. The Interpreter is a backend component that executes the code and updates the GUI.

**Getting Started**

**Where to get the Simulator:**

The code and sample programs for the simulator can be found at <https://github.com/abdullahimtiazyousafzai/Assembly-Language-Simulator>

(The simulator is not final and may be updated, it works fine for now)

**Launching the Simulator**

To launch the Simulator, run the **simulator.py** script. This will open the Simulator GUI.

**Instruction Set**

* **LOM (0000)**

The Load from Memory LOM instruction is used to load contents of memory into the Accumulator register.

**Format: LOM <**memory-address**>**

* **STO (0001)**

The Store into memory STO instruction is used to store contents of the Accumulator Register into the Memory.

**Format: STO <**memory-address**>**

* **ADD (0010)**

The ADD, as the name showcases, is used to add contents of the Data Register DR with ACC accumulator. The data is loaded from memory onto the DR before adding it with ACC.

**Format: ADD <**memory-address**>**

* **SUB (0011)**

SUB instruction is used to subtract the contents of DR from ACC, firstly the contents of memory are loaded to DR before subtraction.

**Format: SUB** <memory-address>

* **MUL (0100)**

The MUL instruction is used to multiply the contents of the DR with ACC repeatedly until the Counter CTR register is not 0, Firstly the multiplicand is loaded onto the CTR, is negated and incremented because only the INC instruction is available so the CTR can only be incremented. Both the DR and ACC hold the same value and are added repeatedly.

**Format: MUL** <memory-address>

* **AND (0101)**

The AND instruction is used for logical AND between ACC and DR and stored in ACC, the data is loaded onto the DR before AND function.

**Format: AND** <memory-address>

* **OR (0110)**

The OR instruction is used for a logical OR between ACC and DR, data is loaded to the DR register before OR operation.

**Format: OR** <memory-address>

* **XOR (0111)**

As the above logical operations, the XOR instruction also performs logical XOR between the DR and ACC and store in ACC.

**Format: XOR** <memory-address>

* **NOT (1000)**

This instruction is used to complement the ACC register.

**Format: NOT**

* **INR (1001)**

This instruction is used to increment the ACC register.

**Format: INR**

* **DEC (1010)**

It is used to decrement the ACC register.

**Format: DEC**

* **JUM (1011)**

The JUM instruction is used to jump the PC to a specific memory location in the RAM. This can be used for loops.

**Format: JUM** <memory-address>

* **JUZ (1100)**

This is a conditional jump which only jumps the PC to a specific location when the ACC is 0. For example, this can be used to jump to the end of the program

**Format: JUZ** <memory-address>

* **INP (1101)**

This is used to move the data from the input register INPR to ACC.

**Format: INP**

* **OUT (1110)**

The OUT instruction is used to send the data of the ACC register to OUT register which will be displayed on the screen or printer.

**Format: OUT**

* **FIN (1111)**

This instruction is used to mark the end of a program, the PC does not move further than this instruction.

**Format: FIN**

**Simulator GUI Layout**

**The Simulator GUI is divided into several sections:**

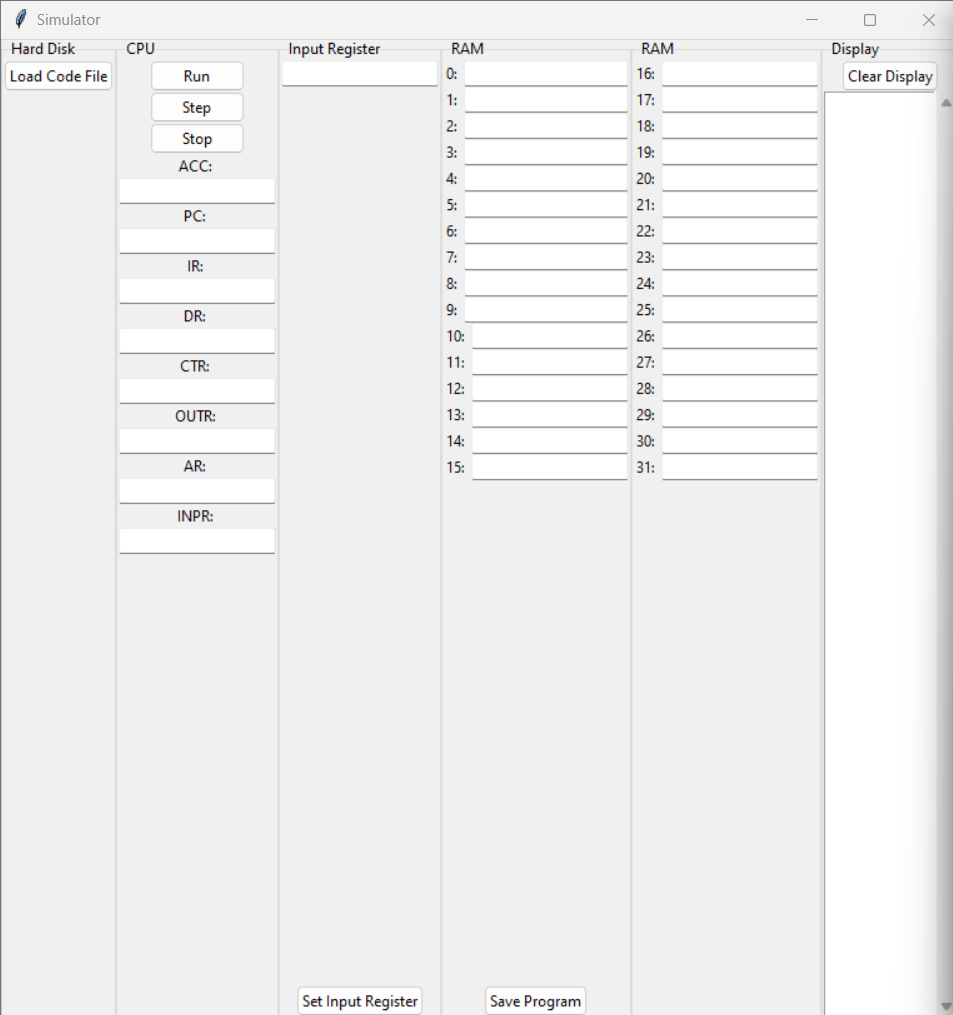
**Hard Disk**: This is where you can load your code file.

**CPU**: This section displays the current state of the CPU registers and provides controls to run, step through, or stop the execution of the code.

**RAM**: This section displays the current state of the RAM.

**Display**: This section displays the output of the code.

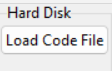
**Input Register**: This section is where you can input data using the input register column in the simulator

****

**Using the Simulator**

**Loading a Code File:**

To load a code file into the Simulator, click on the "Load Code File" button in the Hard Disk section. This will open a file dialog where you can select your code file. The code will be loaded into the RAM cells and displayed in the Display section.



**Running the Code:**

To run the code, click on the "Run" button in the CPU section. The Interpreter will execute the code and update the state of the CPU registers and RAM cells. The output of the code will be displayed in the Display section.



**Stepping Through the Code:**

To step through the code one instruction at a time, click on the "Step" button in the CPU section. The Interpreter will execute the current instruction and update the state of the CPU registers and RAM cells. The output of the current instruction will be displayed in the Display section.



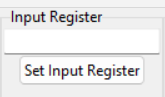
**Stopping the Code Execution:**

The "Stop" button in the CPU section used to set the finished flag true which stops the execution of the program.



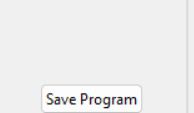
**Setting the input register:**

The input register can be set by the set input register button located in the Input Register column.



**Saving the Program:**

To save the current state of the RAM cells as a program, click on the "Save Program" button in the RAM section. This will open a file dialog where you can specify the location and name of the program file.



Note: The code saved from RAM will not run until loaded again, the load code window will appear automatically

**Clearing the Display:**

To clear the Display section, click on the "Clear Display" button in the Display section.

****

**Conclusion**

This concludes the user manual for the Simulator. If you have any further questions, please refer to the source code or contact the developer.