# Project Documentation: Designing Assembler of SimpleRISC Processor in C++ with GUI in Python

BY:  
K. Ajay (2302VL02) and Anurag Semwal (2302CM02)

## 1. Introduction

This project focuses on designing an assembler for the SimpleRISC processor. The assembler, implemented in C++, converts assembly code into machine code. Additionally, a graphical user interface (GUI) was developed in Python to provide a user-friendly interface for inputting assembly code and viewing the corresponding machine code output.

## 2. Objectives

- To develop an assembler that translates SimpleRISC assembly instructions into corresponding machine code.  
- To create a GUI that simplifies user interaction with the assembler.  
- To ensure efficient performance and accuracy in instruction translation.

## 3. Project Components

## 3.1 Assembler and Emulator (Implemented in C++)

- Reading and parsing SimpleRisc assembly code.  
- Converting mnemonics into binary machine code.  
- Handling labels and memory addresses.  
- Error detection and handling.

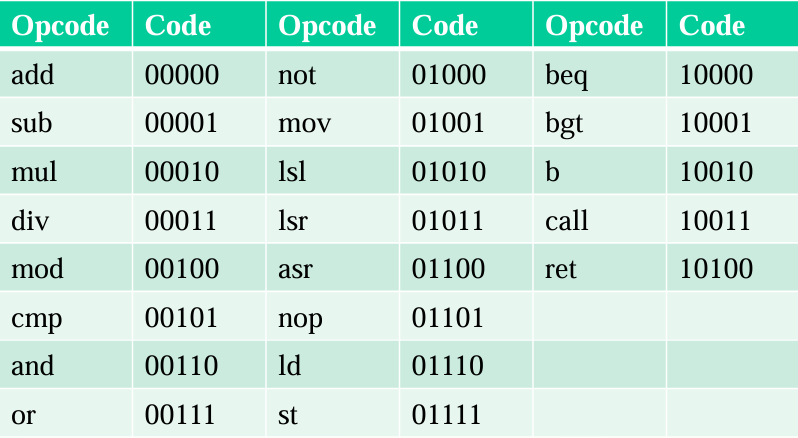
- Can handle multiple nested branch statements using stack memory,

## 3.2 Graphical User Interface (Implemented in Python)

- Inputting assembly code.  
- Displaying machine code output.  
- Highlighting syntax errors.  
- Allowing easy debugging and testing.

-Executes the program and provides user with information of updated memory, register, runtime errors

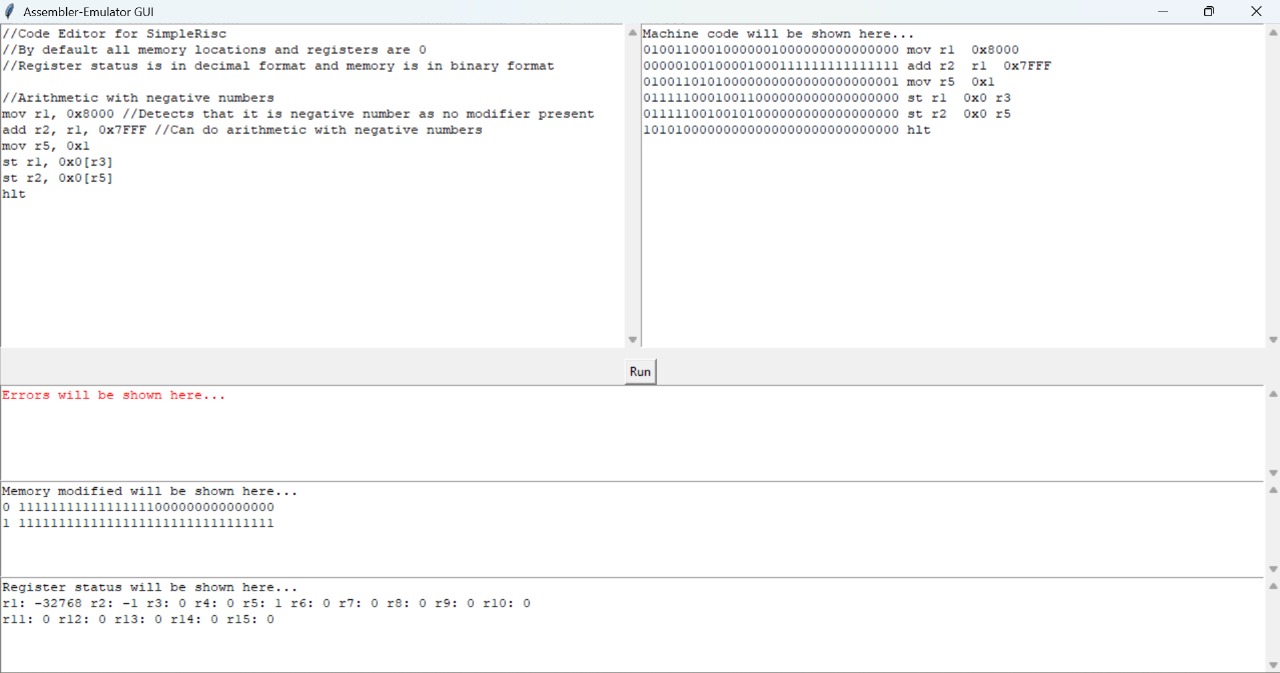
**Instruction Set:**



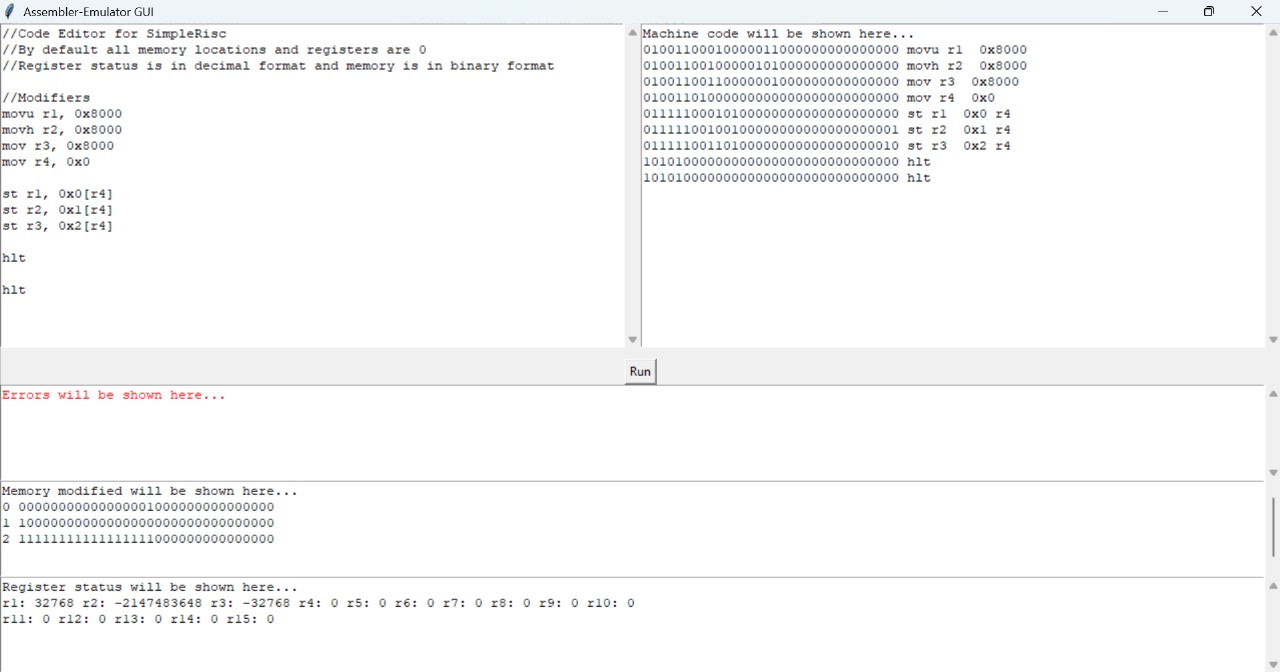
## Project Screenshots:

## 

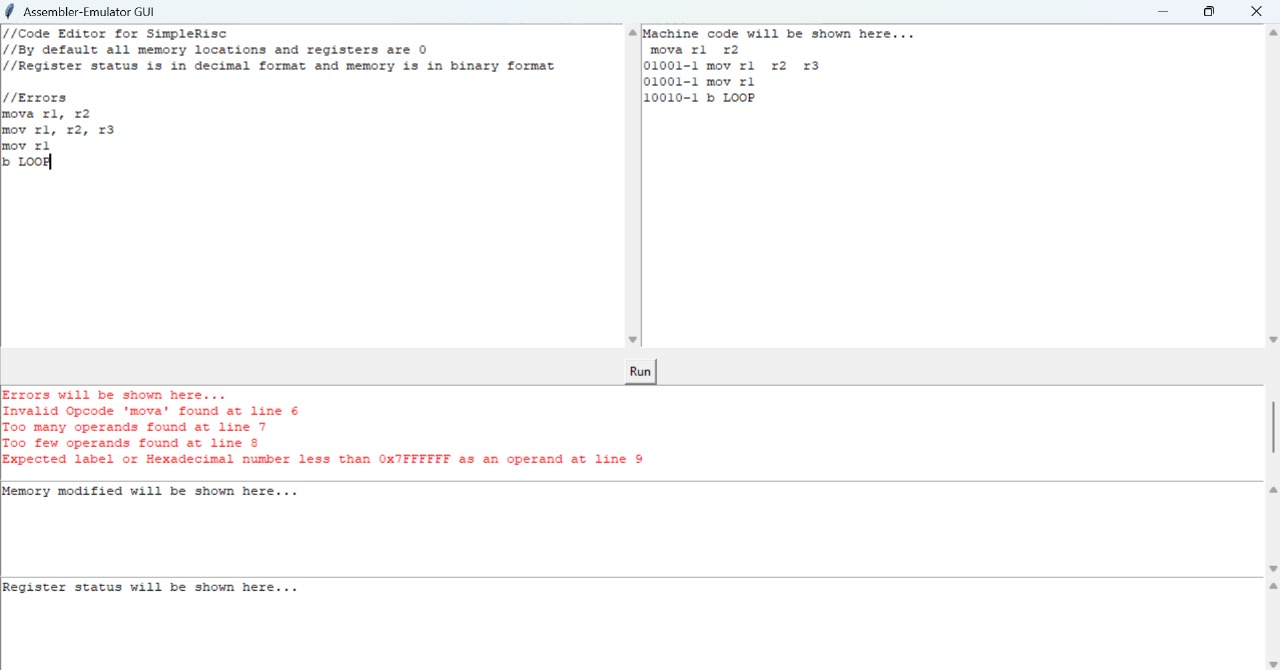
Recursion



Arithmetic with negative numbers



Modifiers



Errors

## 4. Work Distribution

## K. Ajay:

- Developed the assembler and emulator in C++.  
- Implemented instruction parsing and translation logic.  
- Managed label resolution and memory addressing.  
- Handled error detection and debugging.

## Anurag Semwal:

- Designed and developed the GUI in Python.  
- Integrated the assembler and emulator with the GUI.  
- Implemented features for input validation and real-time feedback.  
- Ensured smooth user interaction and debugging support.

## 5. Challenges Faced

## 5.1 Assembler and Emulator Challenges

- Handling label resolution efficiently.  
- Managing different instruction formats correctly.  
- Debugging errors in instruction parsing.

- Handling recursion, modifiers and ensuring valid data was entered

## 5.2 GUI Challenges

- Integrating C++ assembler with the Python GUI.  
- Ensuring real-time response for user inputs.  
- Providing clear error messages and feedback.  
- Handling cross-platform compatibility.

## 6. Limitations

Didn’t code emulator part of Arithmetic Shift Right Operation (ASR)

## 7. Conclusion

The project successfully implemented an assembler for SimpleRISC using C++, with a GUI built in Python for ease of use. Despite challenges in integration, debugging, and performance optimization, the final system efficiently converts assembly code into machine code and provides a user-friendly interface for interaction.

### Acknowledgment

We acknowledge the guidance and support provided by our professors and peers throughout the development of this project.