GIK INSITUTE OF ENGINEERING SCIENCES & TECHNOLOGY

COMPUTER ORGANIZATION & ASSEMBLY LANGUAGE (CE-222E)

Simulator Documentation

INSTRUCTOR NAME: PROF DR. GHULAM ABBAS

SUBMISSION DATE: 29 APRIL 2024

GROUP MEMBERS

- NAQI RAZA (2022574)
- SHAMEER AWAIS (2022428)
- ROOSHAN RIAZ (2022506)
- TAHA JUZAR (2022585)

Contents

Simulator Dependencies

The simulator is developed using python TKinter. As such, the simulator requires Tkinter library which can be downloaded and installed on visual studio code Using the link that is given below: https://tkinter.updatestar.com/en

Layout

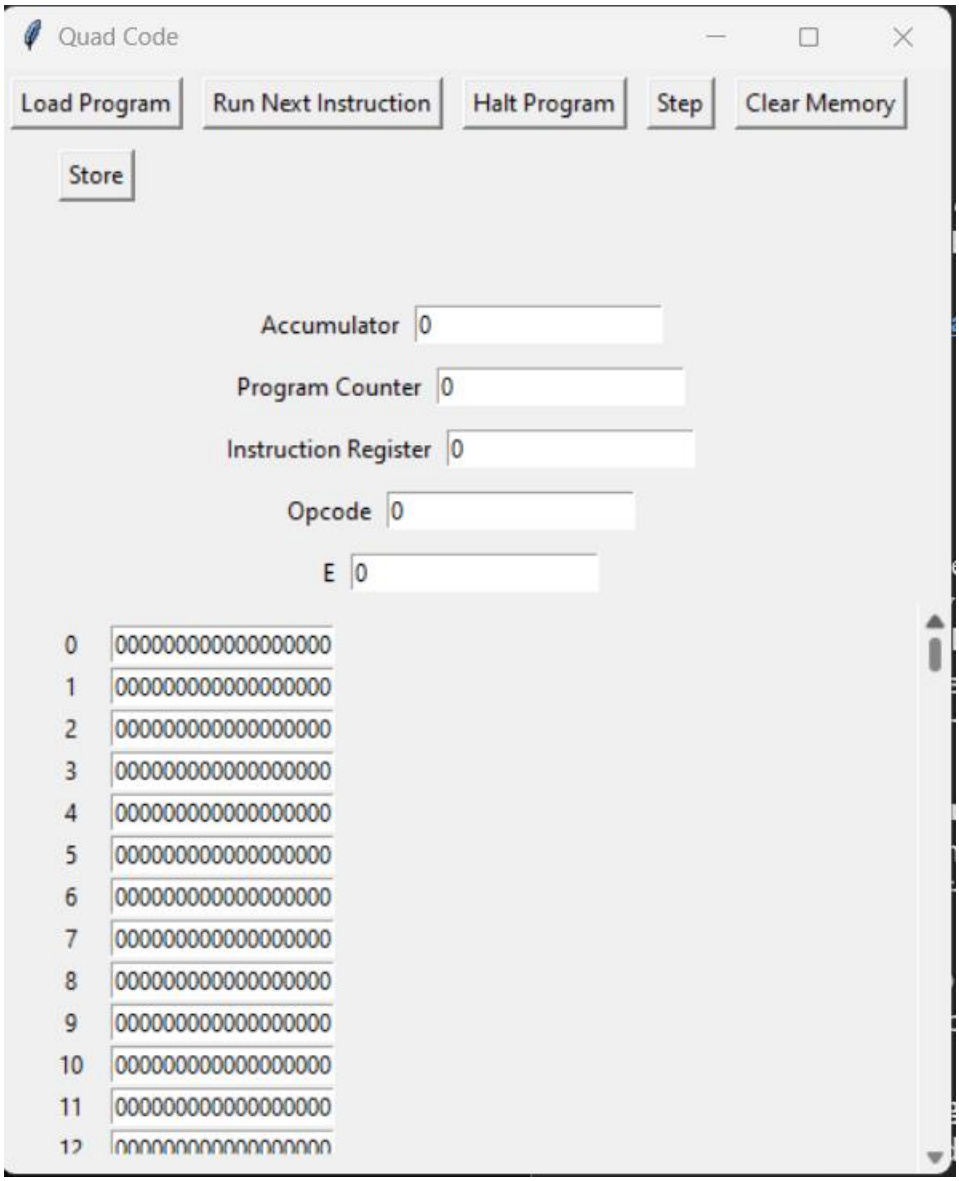


Figure 1: The layout of the simulator

Figure 1 shows the layout of the developed simulator.

Description of Components

A description of the components is given below.

The Main Memory:

A representation of the main memory of the computer. Updates whenever a program is loaded, and after each instruction's execution.

Control Buttons:

These buttons control the computer operations and consist of the following buttons:

- **Load Program From File to Memory:**
  Opens a dialog for the user to select and open a pre-written program from a text file. Automatically loads the instructions to the main memory.

- **Load Written Program to Memory:**
  Loads the program written in 'The Program Input' to the main memory.

- **Step:**
  Executes the instructions sequentially.

- **Run next Instruction:**
  Executes the next instruction.

- **Halt computer and reset:**
  Halts the execution of instructions and resets the computer to the default state.

- **Store the Instructions**:
  The stored instructions represent the program to be executed within the simulator. Storing them allows the simulator to retrieve and execute each instruction in sequence.

Registers

Registers are shown in the simulator such as Program Counter which points to the next instruction, Accumulator stores the value and Instruction Register holds the current instruction being executed.

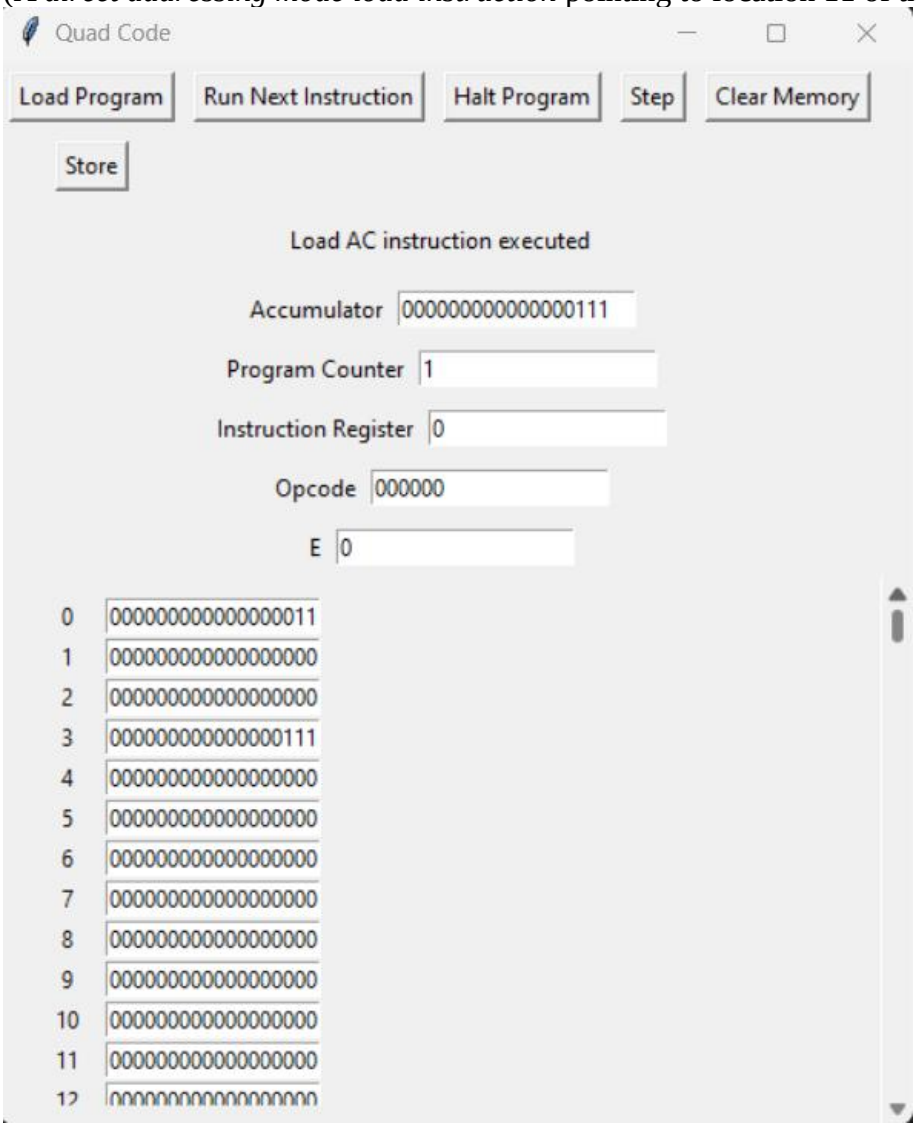
Opcode & E

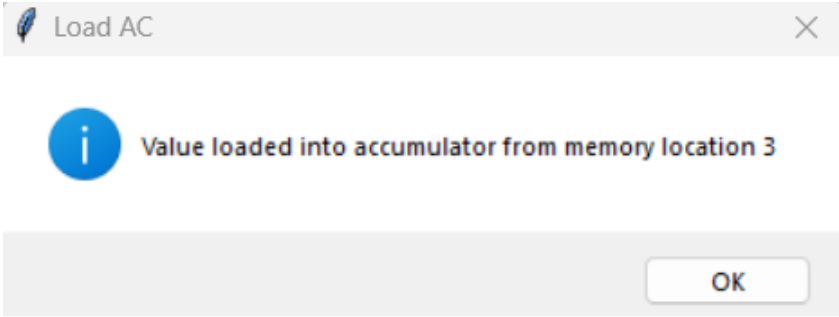
The opcode that is specified in the instruction is fetched and shown in the opcode box which makes it easier for the program to perform required operations in accordance with opcode.
E is for overflown bit. It serves its purpose in Arithmetic and Shift Operations.

Format for Writing Instructions

Memory Reference

Memory Reference Instructions are written such that 6 bits specify the opcode and addressing mode, while the remaining 12 bits are reserved for memory addressing.
An example of the
Load instruction is as follows:
00XXXX
(A *direct addressing mode load instruction* pointing to location 11 of the main memory)





Register-Operation Instructions

Register operation instructions are written in the form of HEX CODE:
Such as F800 for Clear AC

I/O Instructions

I/O instructions are also written in the form of HEX CODE:
Such as 1F080 for skipping the input flag.

Operands

The operands are specified in the memory location and also depend on which type of addressing mode is being used.

Loading a program to memory

The loading of a program can be accomplished from either a text file input. methods are described below.

Loading a program from a file

Clicking on the *'Load Program from File to Memory'* button opens a dialog (Figure 2) where the user can select a text file whose contents are loaded to both the textbox and the main memory. Figure 3 shows a loaded program.

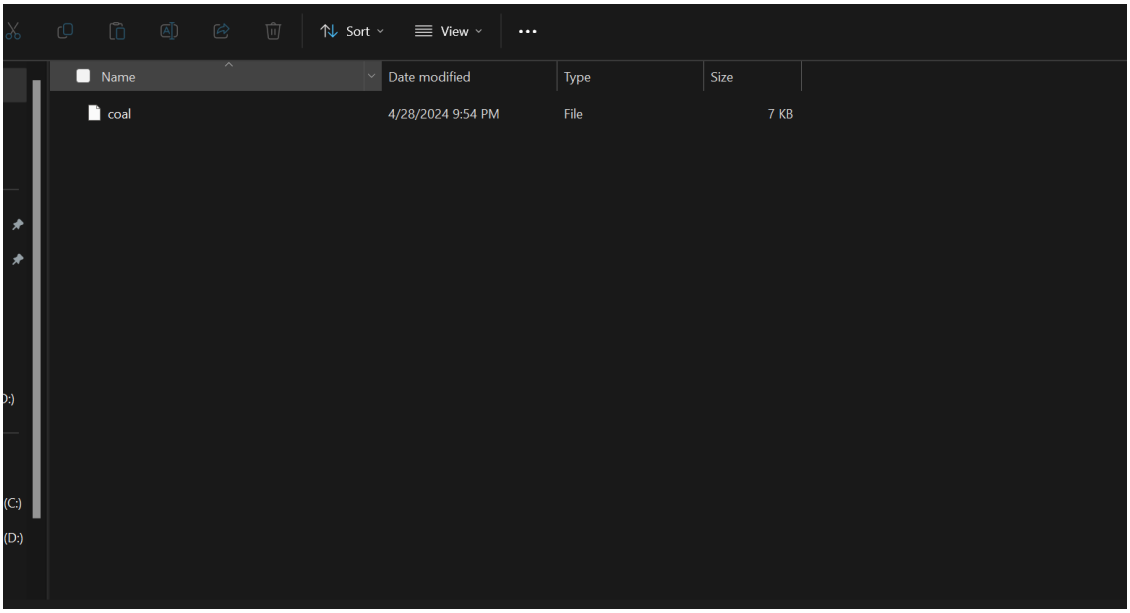


Figure 2: show a test file