# **CISC SIMULATOR**

# **USER MANUAL DOCUMENT**

Tejasvi Chigurupati

Vivek Kommareddy

Isha Paliwal

Yashwanth Raj Varadharajan

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **SECTION** | **CONTENT** | **PAGE NO** |
| 1 | Introduction | 3 |
| 2 | Running the Simulator | 3 |
| 3 | Debugging Panel | 7 |
| 3.1 | Registers and Indicators Region | 7 |
| 3.2 | Memory Interface | 8 |
| 3.3 | Controllers | 9 |
| 4 | IPL | 9 |
| 5 | Instruction | 10 |

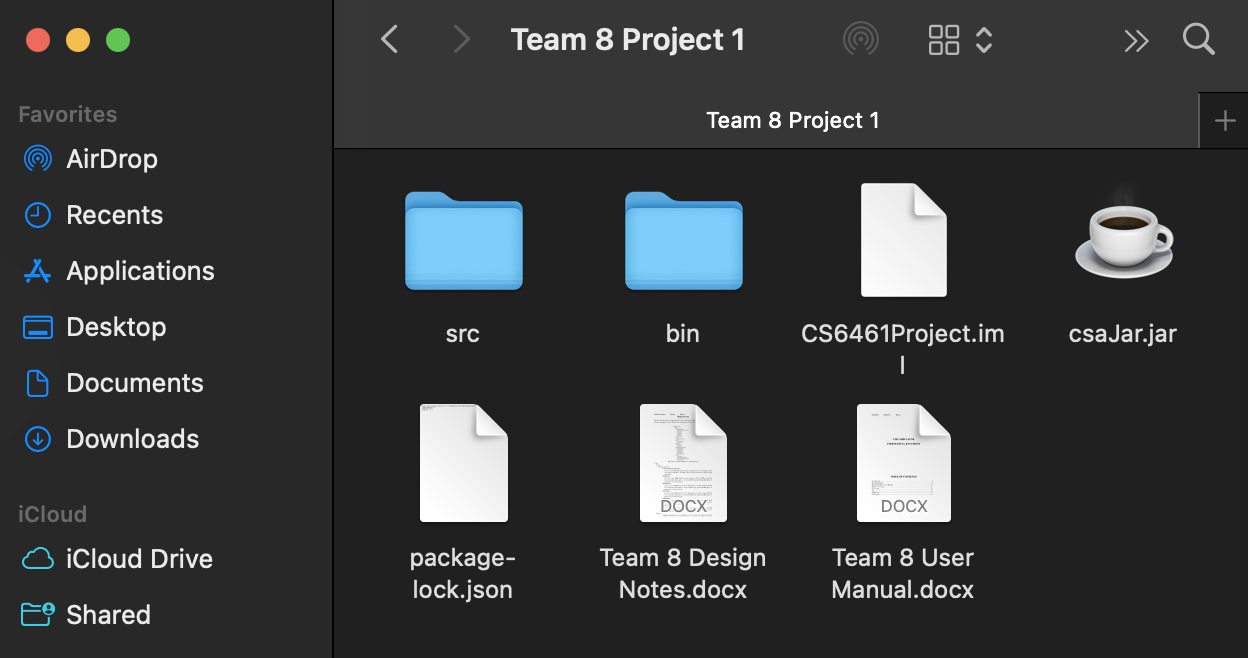
**1. INTRODUCTION**

This simulator emulates the operation of a Complex Instruction Set Computer (CISC). To start using the simulator and execute instructions, you can launch the **"csaJar.jar"** file found in the project's root directory. You have two options for running the file: you can either double-click it directly, or you can use the terminal. If you choose the terminal method, navigate to the project directory using the **"cd"** command, and then execute the command **"java -jar run.jar."**

The following sections of this document provide a comprehensive explanation of how to use the simulator.

**2. RUNNING THE SIMULATOR**

First, we need to download and extract the files. The contents of the file can be visualized as follows:

Figure 1: Contents of project folder

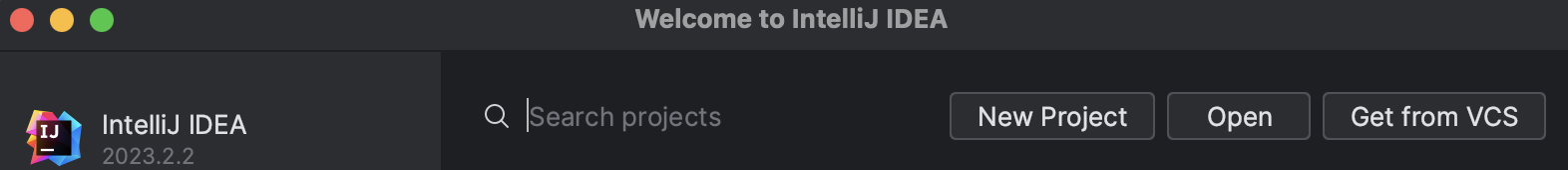
We can run the simulator directly by double clicking on the **csaJar.jar** file. Once done, the following interface will open

A screenshot of a computer

Description automatically generatedFigure 2: Simulator

Alternatively, you can open the file through Code editor. We have used IntelliJ IDE and JDK version 17 to develop the simulator.

Open the IDE and click on Open option.

Figure 3: IDE Start page

Now navigate to project location and open the file. Choose JDK 17 installed in your computer as compiler.

Once you have selected the project, the IDE will open the contents, that can be visualized as follows:

**A screenshot of a computer program

Description automatically generated**Figure 4: IntelliJ Interface

Now click on Run ‘csaJar.jar’ to run the simulator.

A screenshot of a computer

Description automatically generated

Figure 5: Run file from IDE

**3. DEBUGGING PANEL**

The debugging panel displays information regarding the Registers, Indicators, and Memory within the PC. Users also have the capability to manually configure and input data into this panel, as depicted in Figure 1 below.

A screenshot of a computer

Description automatically generatedFigure 6: GUI of the CISC Simulator

The panel is separated into three main sections – the registers and indicators region, the memory interface and the control buttons.

**3.1. Registers and Indicators Region**

**R0- -R3:** These are General-Purpose Registers and containing 16 bits

**X1- -X3:** These are the Index Registers and having 16 bits

**MAR:** Memory address register and the size of it is 16 bits

**MBR:** Memory Buffer Register and it is 16 bits in size

**MSR:** Model specific Register -16 bits in size

**PC:** Program Counter -12 bits in size

**IR:** Instruction Register and the size is 16 bits

**CC:** Condition code -4 bits in size

**MFR:** Machine Fault Register -4 bits in size

A screenshot of a computer

Description automatically generatedFigure 7: Registers and indicators region of the CISC simulator

**3.2. Memory Interface**

Utilizing this memory interface, you will be able to either store or on the other hand load a value to/from memory.

• **Store:** You should input a valid address (one between 0 and 2047) and a value (between 0 and 65535) in the address and value textboxes, respectively, in order to save a value in memory.

• **Load:** Simply enter a valid address in the address textbox and click the load button to display the content of that memory address in the value textbox if you want to load the contents of a specific address of memory.

A screenshot of a computer

Description automatically generated

Figure 8: Memory Interface

**3.3. Controllers**

The controller part of the simulator currently has two buttons. The functions of these buttons are highlighted below

|  |  |
| --- | --- |
| **Button** | **Function** |
| IPL | Pre-load a program from I/O |
| Single Step | One step will be executed at a time |

Table 1: Controller functionalities

**4. IPL**

Within our user interface, there exists a button denoted as "IPL" (Initial Program Load). When the user initiates this function by pressing "execute," the machine is preloaded with a demonstration program that showcases its functionality, and it halts at the program's inception point.

Upon activating the IPL button, the console will present specific register values along with informative messages. The messages displayed in the terminal represent the outcomes of executed instructions designed to illustrate the machine's capabilities. Simultaneously, the register values showcased correspond to the results derived from the execution of these mentioned instructions.

**5. INSTRUCTION**

The simulator's primary function is to execute commands. In front of each instruction, there are 16 radio buttons, each corresponding to one of the 16 bits of the instruction.



Figure 9: Instruction Interface

You have the option to input an instruction just as you would select or deselect registers. Once you've entered the instruction, you can press the "execute" button to carry out that instruction. When the "execute" button is pressed, all register values are updated accordingly.

For instance, if you perform a load instruction that loads data into one of the index registers, the value linked to that specific index register will be displayed on the user interface beside that particular register. Executing this instruction will also modify the values of any other registers involved, and these changes will also be reflected and displayed on the interface.