# **CISC SIMULATOR**

# **USER MANUAL DOCUMENT**

**INDEX OF CONTENTS**

Introduction...............................................................................................................................4

Debugging Panel.......................................................................................................................4

Registers and Indicators Region...............................................................................................5

Memory Interface.....................................................................................................................6

Controllers ...............................................................................................................................6

IPL…………………………………………………………………………………...……….7

Single Step…………………………………………………………………………….……...7

Instruction………………………………………………………………….…………………8

1. **INTRODUCTION**

This simulator is a simulation of a Complex Instruction Set Computer (CISC). To execute instructions and initialize the simulator, the user should run the run.jar file located in the. Root directory of the submitted project file. The run.jar file can be executed either by double clicking the file directly. An alternative way to run the file is to use the terminal. In this case, the cd command can be used to relocate to the project file and then the instruction java -jar run.jar can be executed.

The subsequent sections of this document detail the usage of the simulator.

1. **DEBUGGING PANEL**

Debugging panel shows all the data about the Registers, Indicators, and Memory in the PC and can be composed manually. This is illustrated in the figure 1 below.

A picture containing calendar

Description automatically generated

**Figure 1: 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.

* 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

Calendar

Description automatically generated

**Figure 2: Registers and indicators region of the CISC simulator**

* 1. **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.

Graphical user interface, application

Description automatically generated

**Figure 3: Memory Interface**

* 1. **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**

1. **IPL**

In our user interface, there is a button labeled IPL (initial program load). When the user presses execute, the machine is preloaded with a program that demonstrates how it operates and stops at the beginning. When you press the IPL button, our console will display some values for some registers as well as some messages. The messages you see in the terminal are the results of the instructions we ran to demonstrate the functionality of our machine, and the values you see for the various registers are the results of running the mentioned instructions.

1. **SINGLE STEP**

When a program is run one step at a time, that is known as a single step. Typically, to identify a bug or problem that is preventing the application from working correctly.

1. **INSTRUCTION**

Executing the command is the simulator's primary function. In front of each instruction there are 16 radio buttons that correspond to the 16 bits of the instruction.

A picture containing shape

Description automatically generated

**Figure 4: Instruction Interface**

You can enter an instruction in the same way that you can pick or deselect registers. After that, you can push “execute” to execute the instruction. When the execute button is pressed, all register values are updated appropriately.

For instance, after performing a load instruction that loads data into one of the index registers, the value associated with that index register will be shown on the user interface in front of that particular index register. Executing that instruction will also alter the value of any further registers, which will likewise be updated and shown.