# Simulator User Manual for P4

CSCI 6461 Group 1 (2019 Fall)

1. **The User Interface:**

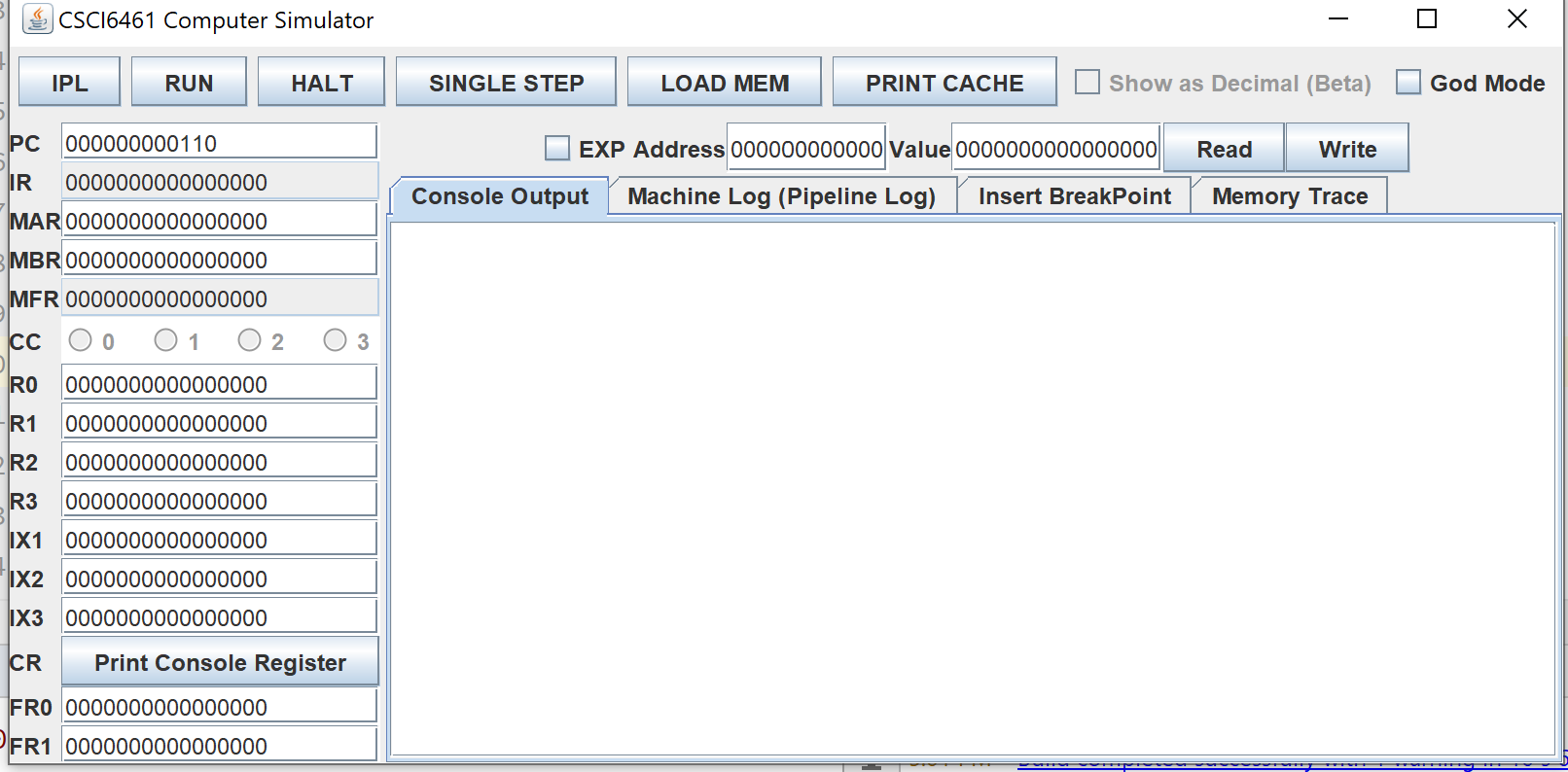


Figure The user interface of the simulator

Figure 1: The User Interface of the simulator

Figure 1 shows the simulator GUI interface.

**Main Control Area**

* **IPL Button** will reset the entire simulator. All data inside the memory and registers will be set to initial value.
* **Run Button** will continue to execute instruction driven by PC unless it meets a HALT instruction or manually click the HALT button on the GUI.
* **HALT Button** will send a HALT.
* **SINGLE STEP Button** will execute one instruction from the PC.
* **LOAD MEM Button** will read a CSV file and load the value to the memory.
* The CSV file should have 2 fields.
* <Memory Location(Binary String)>,<Data(Binary String)>.
* Here is an example:
  + 000000000001,0000011100011111
  + 000000000010,0000011100011111
  + 000000000011,0000011100011111
  + 000000000100,0000011100011111
* The example above load data "0000011100011111" to memory location 01,10,11,100.
* **PRINT CACHE Button** will print the entire cache data to the “Machine Log” Output Space.

**Register Field**

* Most of the register value is editable, while some of the registers like IR, MFR, CC is not changeable.
* To change the value of the register, type the value you want to set and press <Enter>.
* **Print Console Register Button** will print all the console register data to the “Machine Log” Output Space. (Console Register is used for IN/OUT Instruction, with DEVID 3-31).

**Memory Field**

* EXP Click-box is the Memory Expand button. When this box is checked, the max memory is set to 4096 words, while the default is 2048 words.
* To read the memory value, just type the address you want to read, click the Read button, and the value will appear on the value field.
* To write the memory, just type the address and the value and click Write button. If your input is legit, the value will be writing to the memory location.

**Display Area**

* **Console Output**

This is the place for the OUT Instruction to print something. This simulates the “Console Printer” for OUT Instruction.

* **Machine Log**

This is the place to display all the machine logs, which have a very specific detail to demonstrate what the simulator is doing.

You can check whether the Cache is HIT or MISS, the value of all dumped info, etc.

* **Insert BreakPoint (Extra Field Console)**

(Extra credit) This is the place to set a breakpoint. When clicking the RUN button, when PC meets any of the breakpoint values, the simulator will come to a stop so you can start single step for easy debugging.

* **Memory Trace (Extra Field Console, New in P4)**

(Extra credit) This task is will helps that user to dump memory that will print all the memory has been used with its value stored by click on the “Dump memory” button.

A screenshot of a social media post

Description automatically generated

Figure The interface for the memory trace loge

1. **Operate the simulator to Testing Program IVa**

Testing Program IVa is a simple program to test floating point add/subtract, vector add/subtract and floating point convention.

First the program try to **add** two floating numbers and save to memory location 28, and then **subtract** a floating point and save the result to memory location 26. Then it perform **converting** between floating point and integers, save it to memory location 24. Finally, it performs a **vector add/subtract** operation, which operate 2 vectors which length is 3, at the memory location 60 and 80.

To see the results, simply visiting those memory location use the Memory Field, or dump the entire memory using Memory Trace.

To see the result step by step, just simply click “SINGLE STEP”, this will give you detailed memory trace information for each step and clearly see the operations mentioned above is working.

**To run this testing program 1, follow the step below:**

* Open the simulator JAR file, and click the “IPL” Button.
* Click the “LOAD MEM” Button, and select the testing program CSV file.
* Click “RUN”
* It will require 20 numbers at first. When it pop-up a window to require a number, type the number directly. The range of the input is 0~65535.

*For Example: If you would like to input number “234” to be the first number, type “234”.*

* After the 20 numbers you input, it will require a new number. Just type the new number for the program to compare, and the simulator will display the closet number for the 20 numbers you input before and print it to the “Console Output”.

**To run this testing program 2, follow the step below:**

* Open the simulator JAR file, and click the “IPL” Button.
* Click the “LOAD MEM” Button, and select the testing program CSV file.
* Click “RUN”
* It will require load default paragraph here we have example txt file named TestProgram2-Paragraph.txt .
* Then it will ask for a user input word to search. If it find the word, it will print out sentences number and word number of sentence. Else, print nothing but the paragraph.

**To run this testing program IVa, follow the step below:**

* Open the simulator JAR file, and click the “IPL” Button.
* Click the “LOAD MEM” Button, and select the testing program CSV file.
* Click “**SINGLE STEP**” to see the changes in the memory for each instruction. The Assembly code for testing program IVa can be found in the source code folder.

1. **Design note for the simulation**

**A picture containing text

Description automatically generated**

Figure 1 Pipeline design

The figure 1 shows the basic structure of our pipeline design. Pipeline is implemented into 4 stages, each stages performs certrain operations like Fetch PC(A), Memory visiting(B)... . The machine logs demostrates each stage operation. Each instruction takes 4 cycles to finish, and when it comes to false fetch when perform branching, it will insert a bubble inside the pipeline (This can be identified when logging as BubbleInsert).

A screenshot of a social media post

Description automatically generated

Figure 2 Package CPU

A screenshot of a social media post

Description automatically generated

Figure 3 Package CPU\_complete

A screenshot of a cell phone

Description automatically generated

Figure 4 Package Computer

A close up of a piece of paper

Description automatically generated

Figure Package interface

A screenshot of a cell phone

Description automatically generated

Figure Package Memory