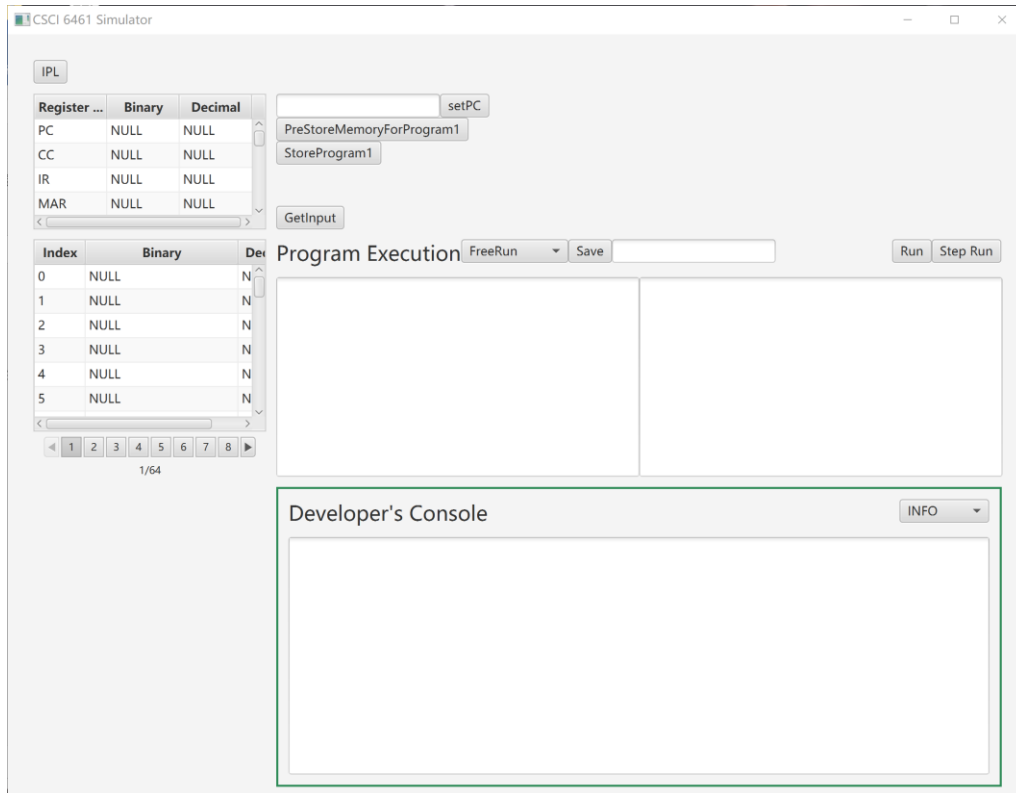
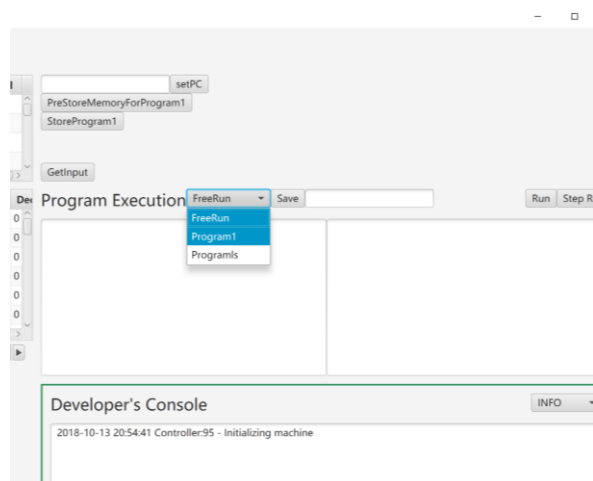


User GUIDE



- ① Open **simulator-1.0-SNAPSHOT.jar**
- ② Click Button **[IPL]**, the Simulator is Initialized.

How to Run Program1:



Program1 is pre-stored in a file, thus you can click button **[FreeRun]** and then chooses **[Program1]**.

To run it:

- ① Click Button **[PreStroeMemoryForProgram1]**

That will store some values into memory, which helps run the program1;

memory[8]	0000000001000000	64	for X1
memory[9]	0000000010101010	170	for X2
memory[85]	1111111111111111	65535	for compare
memory[86]	0000000001000000	64	
memory[87]	0000000001010100	84	

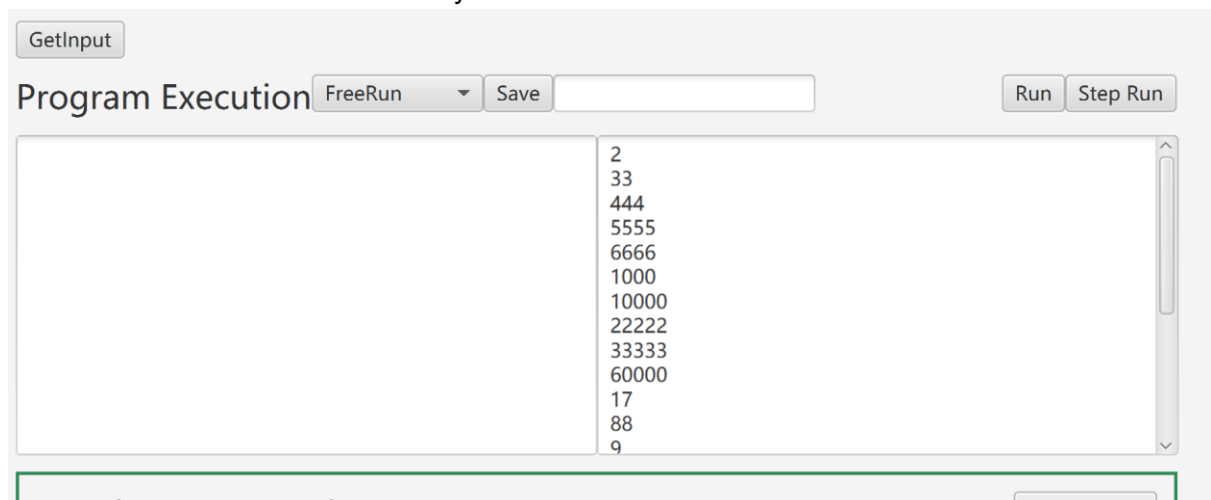
- ② Click Button **[StoreProgram1]**

That will store program1 form the memory[126] to memory[185]

And set the PC =126

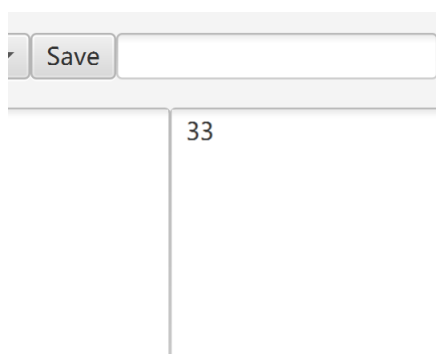
[For more detail of program1, please check program1.xlsx](#)

- ③ Enter 21 numbers in the box. The simulator will search the first 20 numbers for the number closest to the last number entered by the user.



Then Click Button **[GetInput]**

- ④ Then click **[Run]** or **[Step Run]** to run the program1. After each instruction, the Developer's Console will output information about what the simulator has done.
- ⑤ After calculation, the box outputs a number which is the closest one among 20 numbers.



Specifically, in our test, we input the following numbers

	A
1	2
2	33
3	444
4	5555
5	6666
6	1000
7	10000
8	22222
9	33333
10	60000
11	17
12	88
13	9
14	10
15	12
16	15
17	118
18	3298
19	5389
20	3213
21	32

It's evident that the correct answer is 33, which is consistent with the output from our simulator.

[Note: If you input a value that is not a number, it won't be IN to the simulator.](#)

How to Run Instructions

We have pre-stored a “Load and Store Test Program” for quick experiments.

- ① Choose **[programs]**, it will be loaded automatically.
- ② Input a value here, which is the start index for test program.

Program Execution

Programls

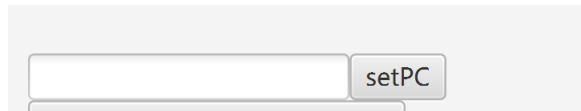
Save

100

0000110000011001
0000100000010100
0000110000000011
0000100000011110
1000010001011110
0000111100011101
0000101101010110
0000011000110100
0000101001010001
1000100001110100

- ③ Click Button **[Save]**, now the “**programs**” is stored to memory, start with 100, PC=100
- ④ Then click **[Run]** or **[Step Run]** to run it.

- ⑤ You can use **[SetPC]** to change PC, then run a instruction repeatedly.



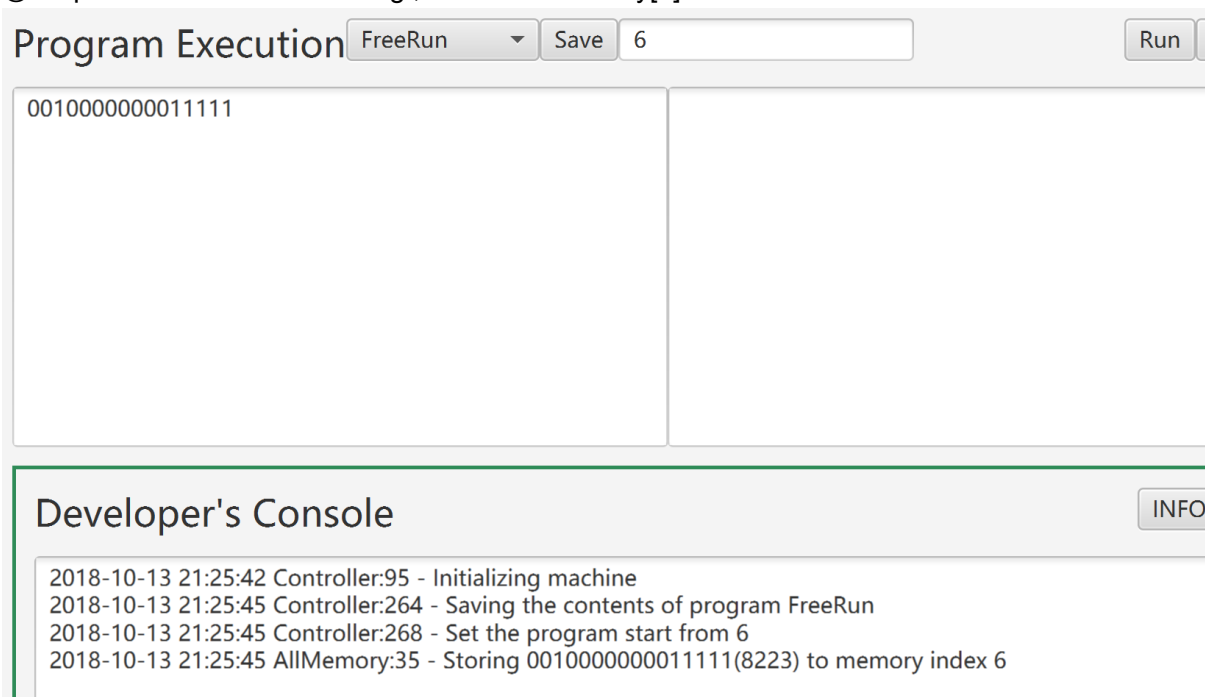
A screenshot of a web interface for setting the Program Counter (PC). It features a text input field and a button labeled "setPC".

Input a index value like "100" and click **[SetPC]**, PC = 100.

How to FreeRun

You can use instructions freely by choosing **[FreeRun]** and set your own instructions.

- ① Input an instruction in box.
Such as "0010000000011111" JZ R0,0,31
- ② Input an index and save. E.g., save it to memory[6]



A screenshot of a web application interface for program execution. The top section, titled "Program Execution", contains a dropdown menu set to "FreeRun", a "Save" button, an input field with the value "6", and a "Run" button. Below this is a large text area containing the instruction "0010000000011111". The bottom section, titled "Developer's Console", has an "INFO" button and displays a log of system events:

```
2018-10-13 21:25:42 Controller:95 - Initializing machine
2018-10-13 21:25:45 Controller:264 - Saving the contents of program FreeRun
2018-10-13 21:25:45 Controller:268 - Set the program start from 6
2018-10-13 21:25:45 AllMemory:35 - Storing 0010000000011111(8223) to memory index 6
```

- ③ Click **[Run]**
- ```
2018-10-13 21:26:18 AllMemory:73 - Fetching 0010000000011111(8223) from memory index 6
2018-10-13 21:26:18 AllMemory:76 - Hit cache
2018-10-13 21:26:18 Transfer:55 - JZ Jump to 31 when R0 =0
2018-10-13 21:26:18 AllMemory:73 - Fetching 0000000000000000(0) from memory index 31
2018-10-13 21:26:18 Miscellaneous:37 - HLT
```