# User GUIDE

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

If you want to use the simulator, you must click IPL first.

How to Test Floating Point Instructions/Vector Operations

Here are some instructions:

* Deposit FR0 and memory[22] a floating point number first. (e.g. FR0 = 2.0, memory[22] = 3.25)
* FADD FR0,0,0,22 (FR0 = FR0 + memory[22])

0110110000010110

* FSUB FR0,0,0,22 (FR0 = FR0 – memory[22])

0111000000010110

* Deposit R0 0 or 1, if R0 = 0, deposit memory[23] a floating point number, if R0 = 1, deposit memory[23] a fixed point number.
* CNVRT R0,0,0,23 (if R0 = 0, R0 = (float->int) memory[23]; if R0 = 1, FR0 = (int->float) memory[23])

0111110000010111

* Deposit some values to memory and registers

|  |  |
| --- | --- |
| FR0 | 2.0 |
| Memory[20] | 100 |
| Memory[21] | 200 |
| Memory[100] | 1 |
| Memory[101] | 2 |
| Memory[200] | 3 |
| Memory[201] | 4 |

* VADD FR0,0,0,20

(Vector1 starts from 100(memory[20]) and Vector2 starts from 200(memory[21]))

1000110000010100

* VSUB FR0,0,0,20

1001000000010100

* memory[24] = 3;

memory[25] = 0.25(0010000000000000->Fixed-point decimal representation);

* LDFR FR0,0,0,24 (FR0 = memory[24]. memory[25])

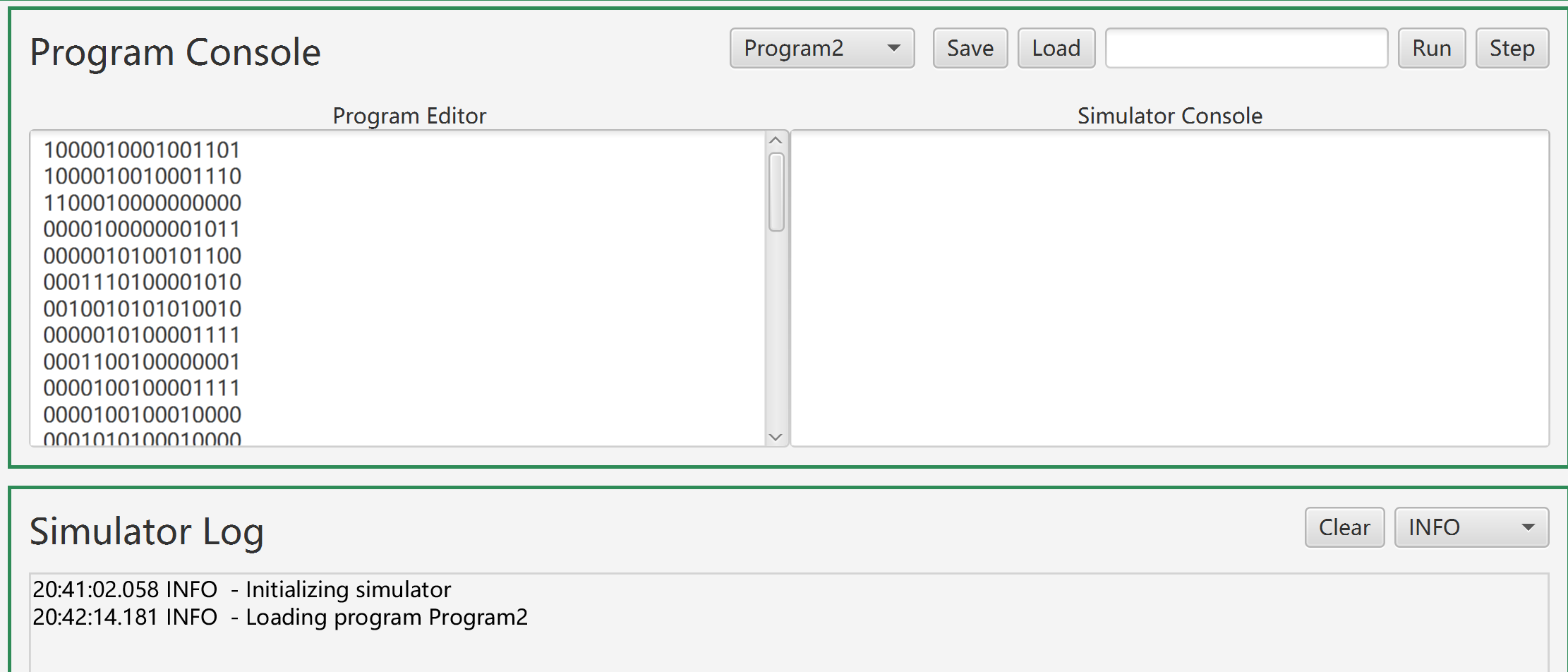
1010000000011000

* STFR FR0,0,0,24 (memory[24]. memory[25] = FR0)

1010010000011000

How to Run Program2 (included in Part3)

Program2 is pre-stored in a file, thus you can select Program2 in **ComboBox [programNameSelector]** and see pre-stored instructions for program 2 in **Program Editor**.



To test program 2:

1. Click Button **[PreStroeMemoryForProgram2]**

That will store some values into memory, which helps run the program2;

|  |  |  |  |
| --- | --- | --- | --- |
| memory[12] | 0000001000000000 | 512 | Word store start |
| memory[13] | 0000000001000000 | 64 | for X1 |
| memory[14] | 0000000001100000 | 96 | for X2 |
| memory[15] | 0000000000000001 | 1 | sentence number |
| memory[16] | 0000000000000001 | 1 | word number |

1. Click Button **[LoadProgram2]**

That will store instructions used program2 into memory [64-100] and set the PC =64.

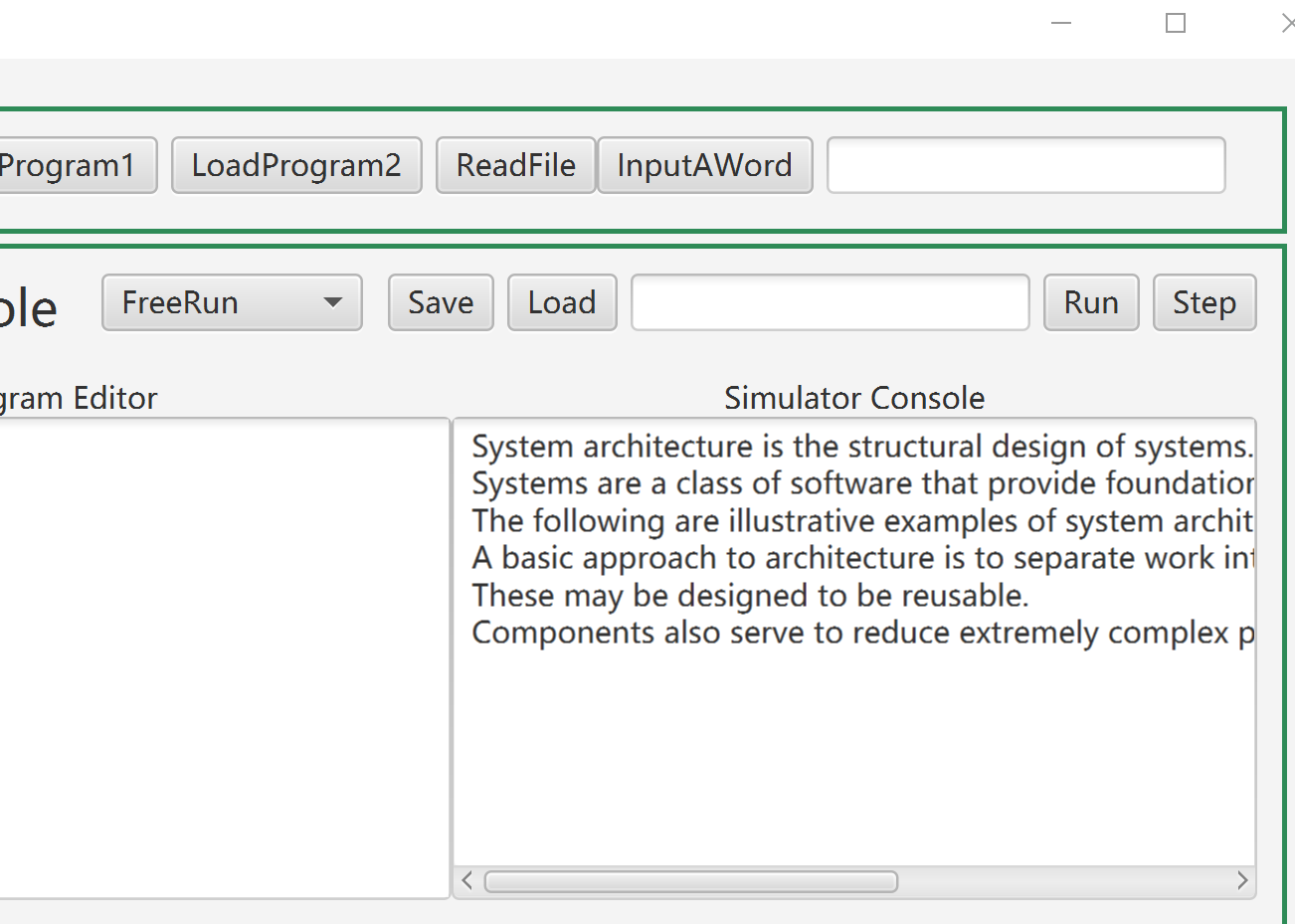
For more details of programs, please check **program2.xlsx**

If you want to load program2 manually, make sure program2 is stored from memory [64].

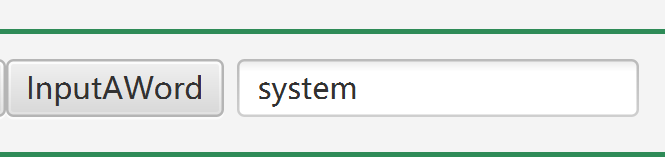
1. Click Button **[ReadFile]**

Our simulator will read “program2\_paragraph.txt” and store each word into memory automatically.

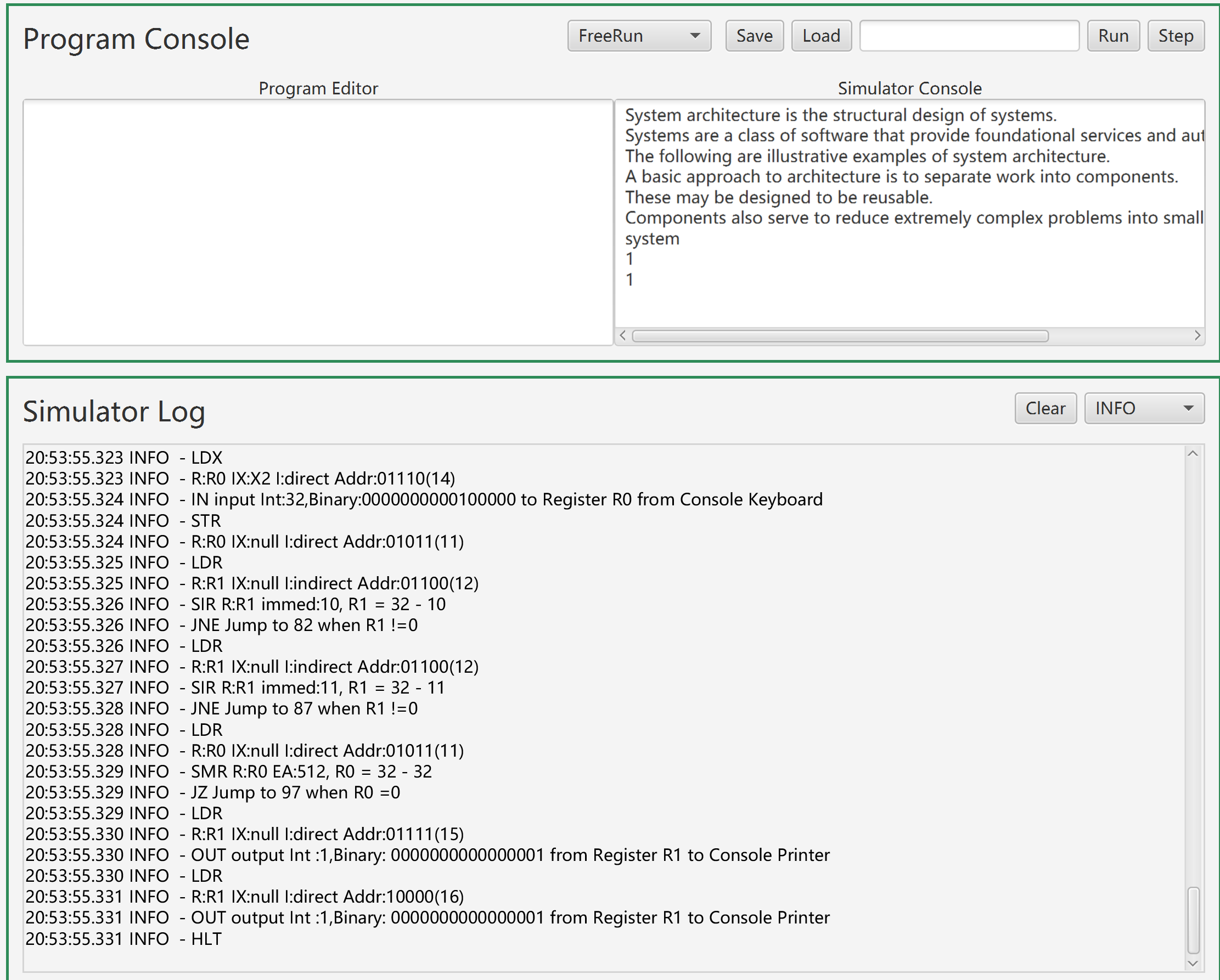
(“**simulator-1.0-SNAPSHOT.jar” and** “program2\_paragraph.txt” should *be* in the same folder.)



1. Input a word here, such as “system,” then click **[InputAWord]**.



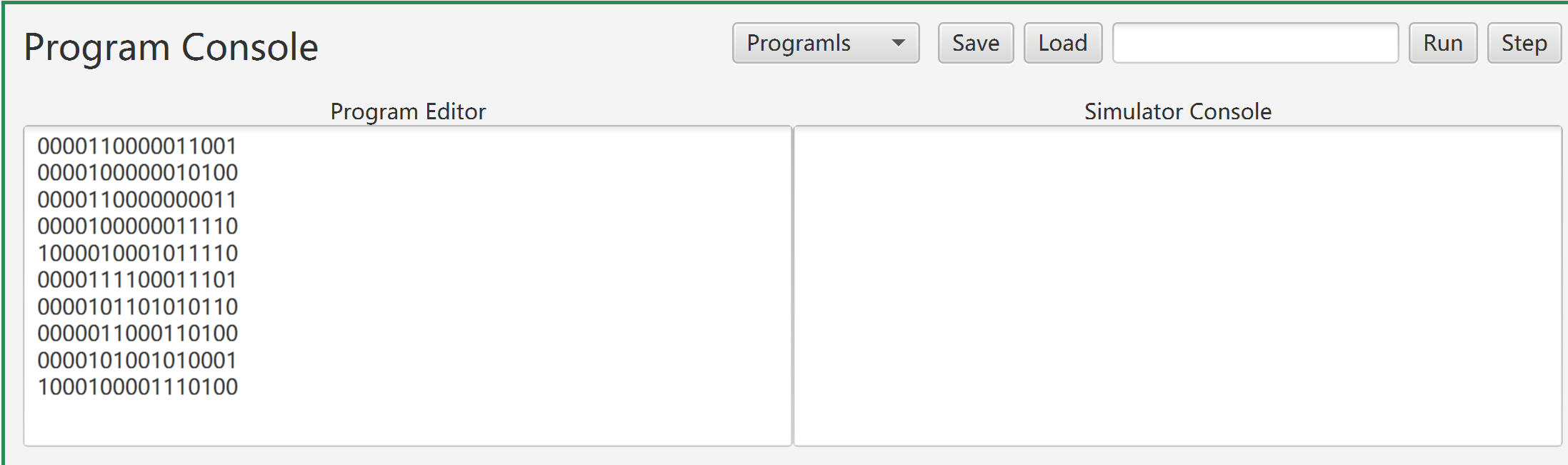
1. Then click **[Run]** or **[Step]** to run the program2. After executing instructions orderly, the Simulator Log will output information about what our simulator has done.
2. The word, sentence number and word number will output in Simulator Console.



A quick test on Instructions (included in Part2)

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

1. Select **[Programls],** it will be loaded automatically.



1. Input a number in the TextField if you need it.
2. Click Button **[Load],** and then the **“programls”** will be loaded to memory start from StartIndex(input in last step) or Default Location(no input), which is 32.
3. Then click **[Run]** or **[Step]** to run it.

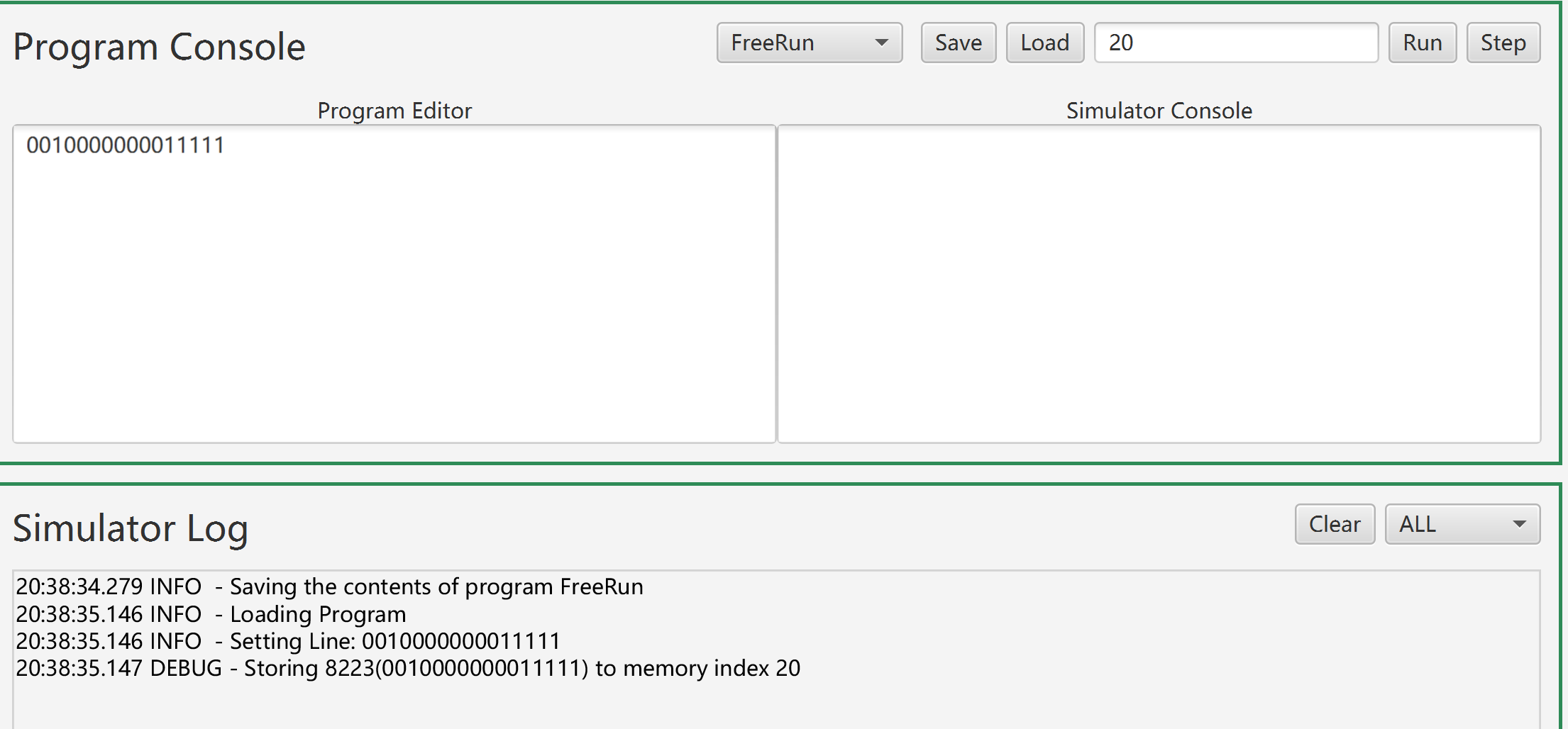
Test instructions freely (included in Part2)

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

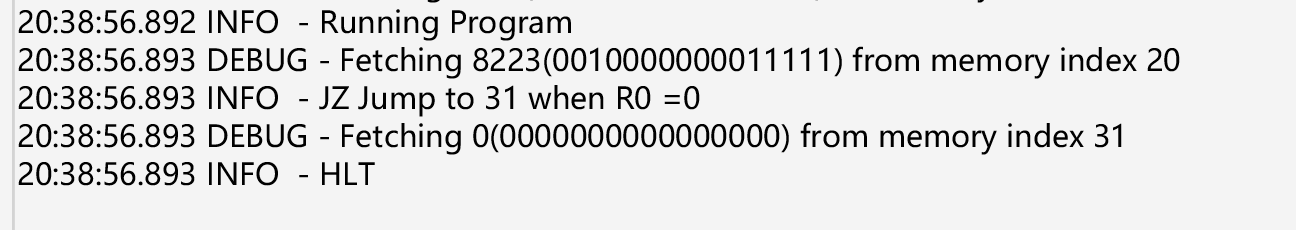
1. Input an instruction in Program Editor.

Such as “0010000000011111” (JZ R0,0,31)

1. Input in TextField an memory index where the instruction will be sored. E.g., we store it in memory[20]



1. Click **[Run]** or **[Step]**, this instruction is executed

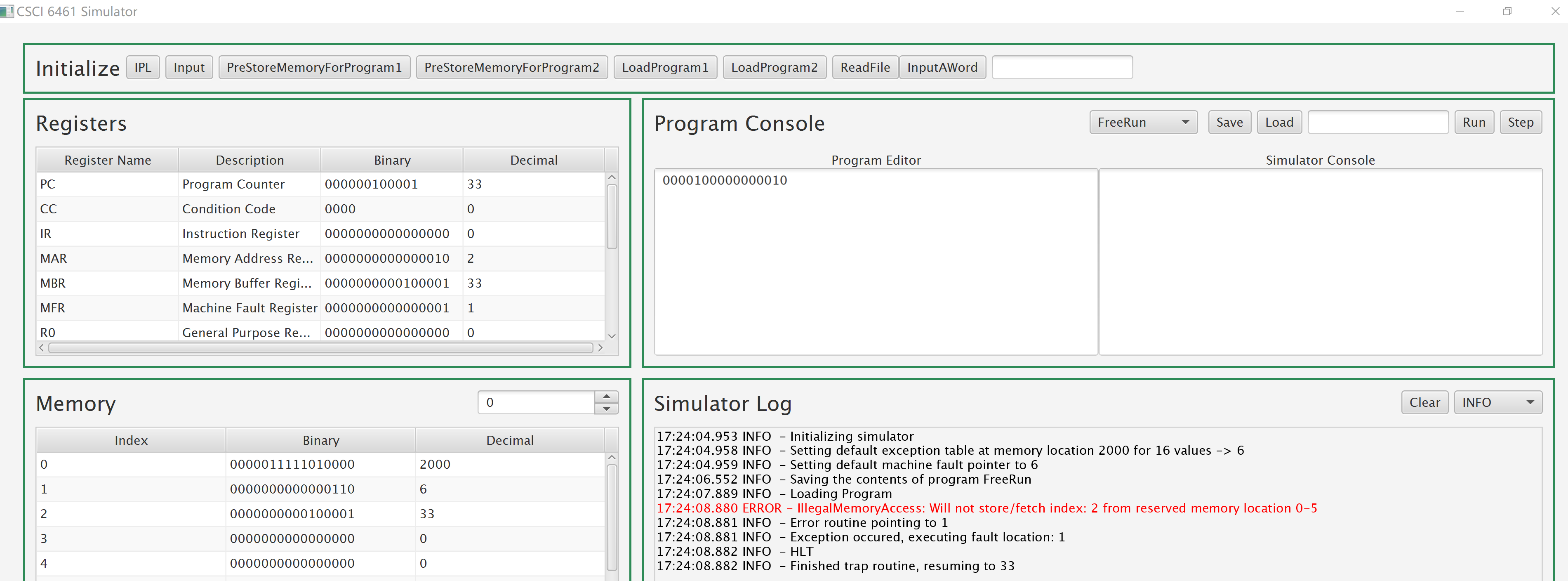


## Tests for Machine Fault and Trap (included in Part3)

1. Illegal Memory Address to Reserved Locations **MFR set to binary 0001**
2. Illegal TRAP code **MFR set to binary 0010**
3. Illegal Operation Code **MFR set to 0100**
4. Illegal Memory Address beyond 2048 (memory installed) **MFR set to binary 1000**
5. Illegal Memory Address to Reserved Locations

STR R0,0,2 0000100000000010

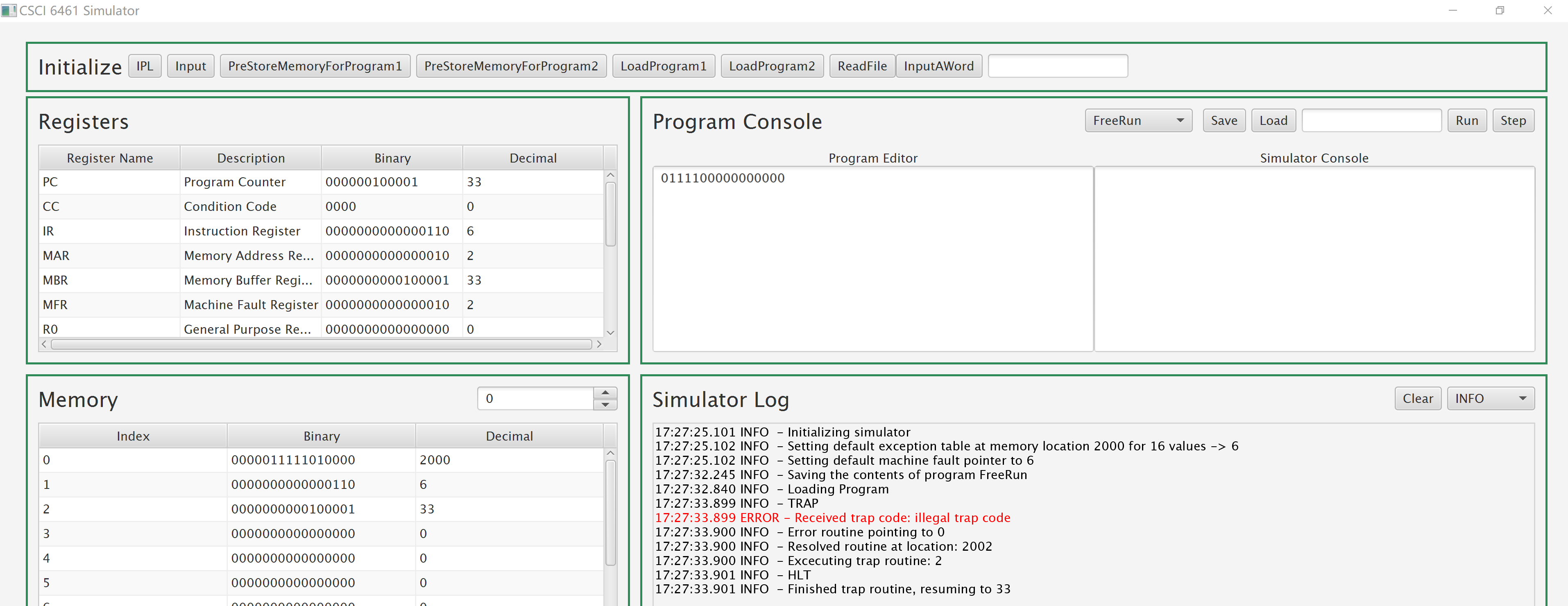
Memory[2] is a reserved location, so it will throw IllegalMemoryAccess Exception.



1. Illegal TRAP code

0111100000000000 TRAP 0

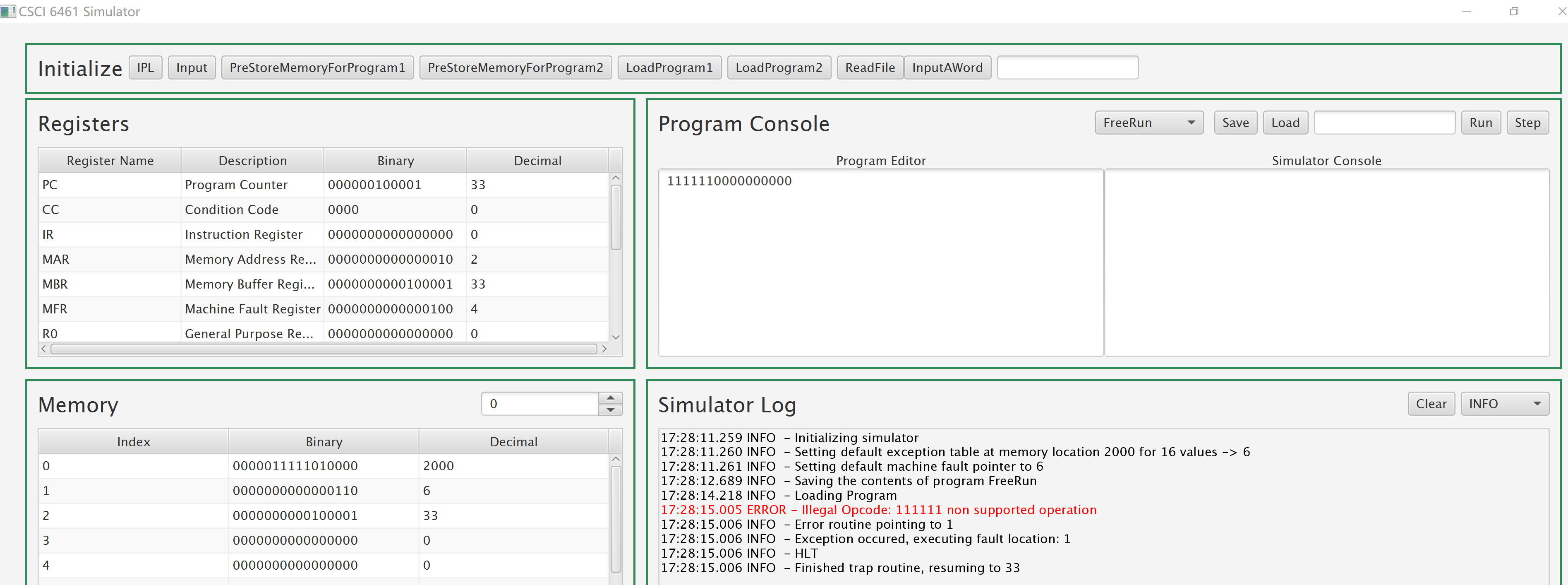
We don’t have routine 0, so Trap 0 will throw IllegalTrapCode Exception.



1. Illegal Operation Code

1111110000000000

“111111” is not an operation type, so it will throw IllegalOpcode Exception.

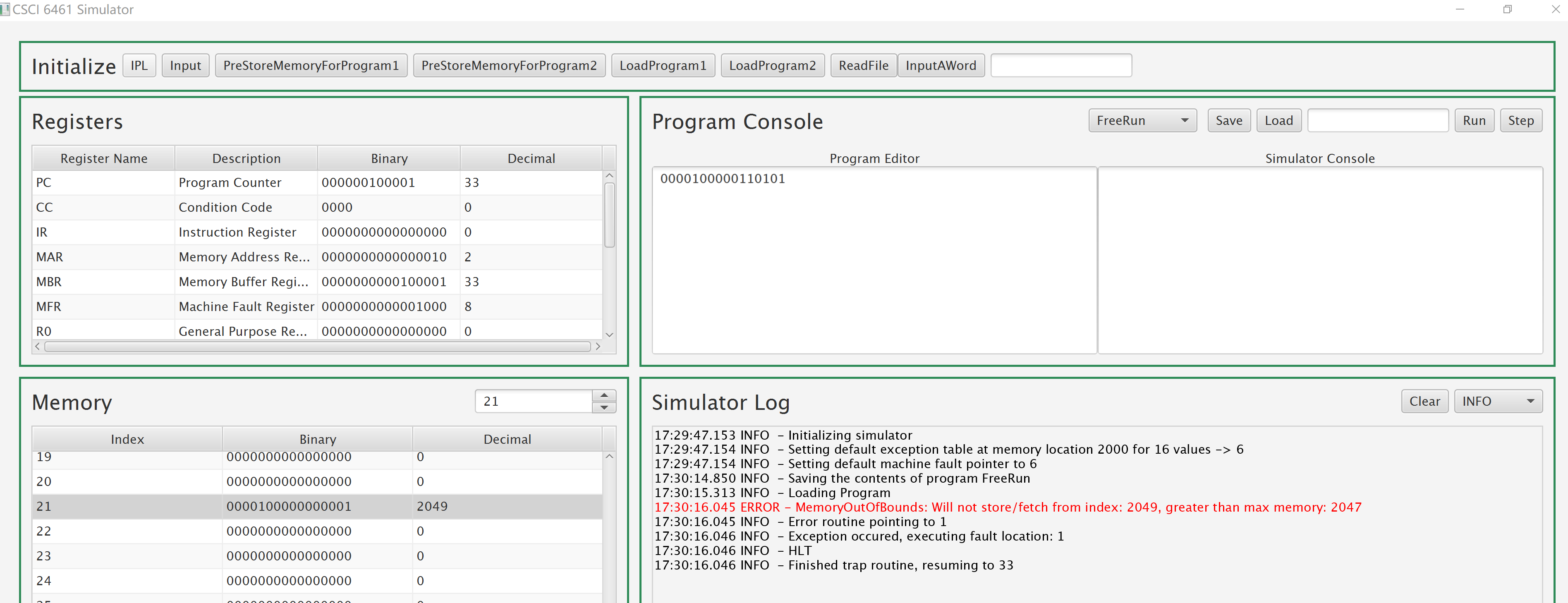


1. Illegal Memory Address beyond 2048

Memory [21] = 2049

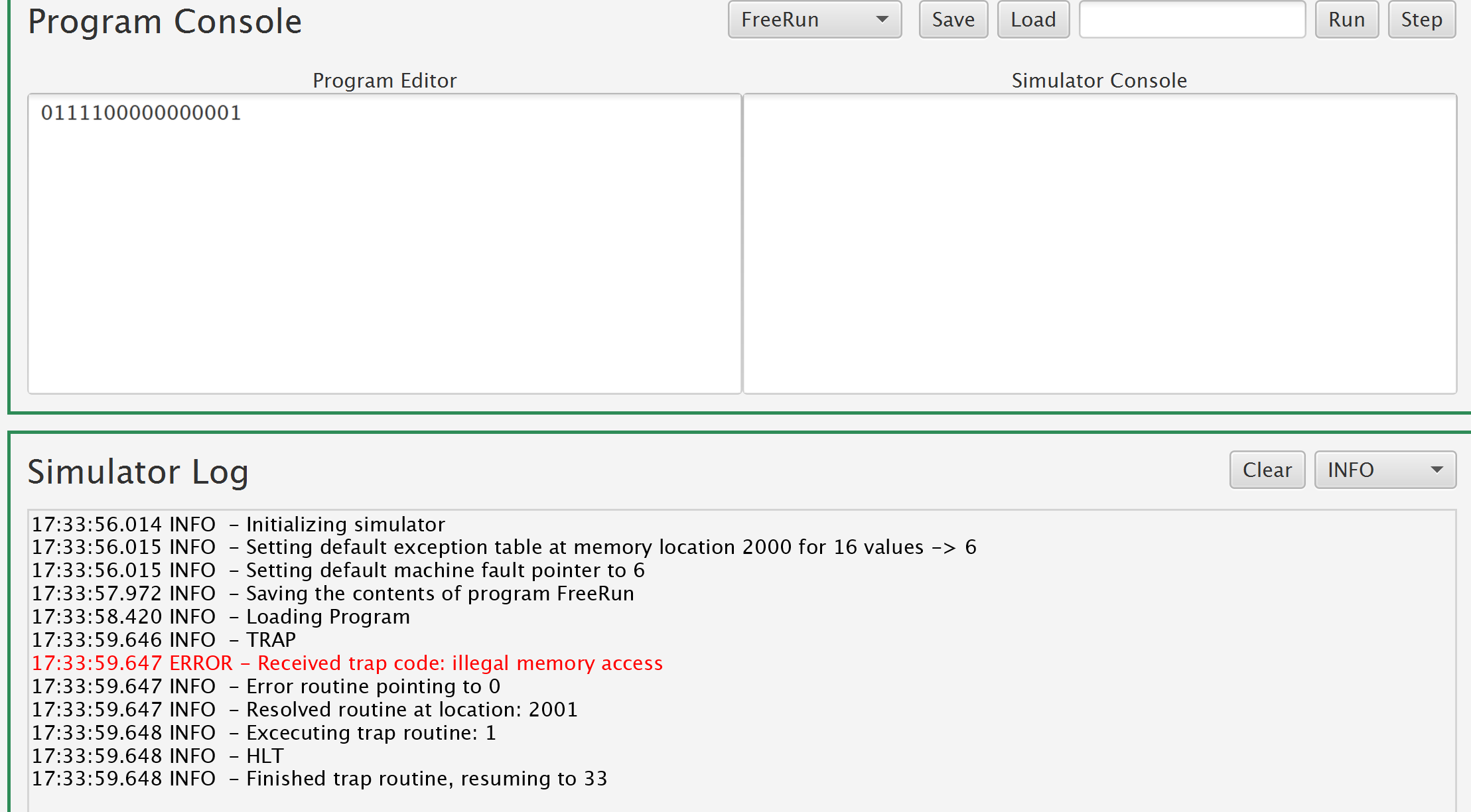
STR R0,1,21 0000100000110101

Memory[2049] is out of bounds, so it will throw MemoryOutOfBounds Exception.

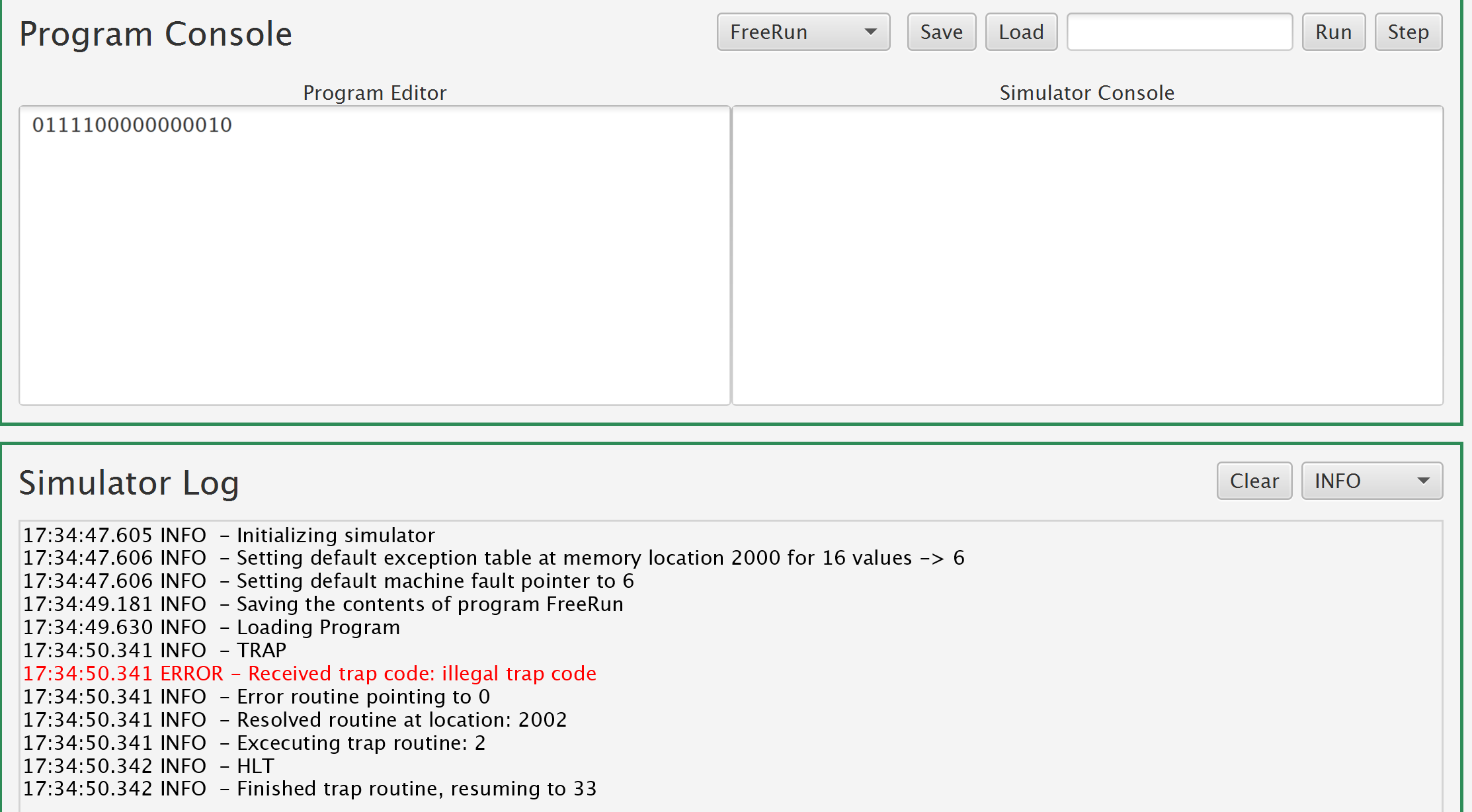


1. Trap

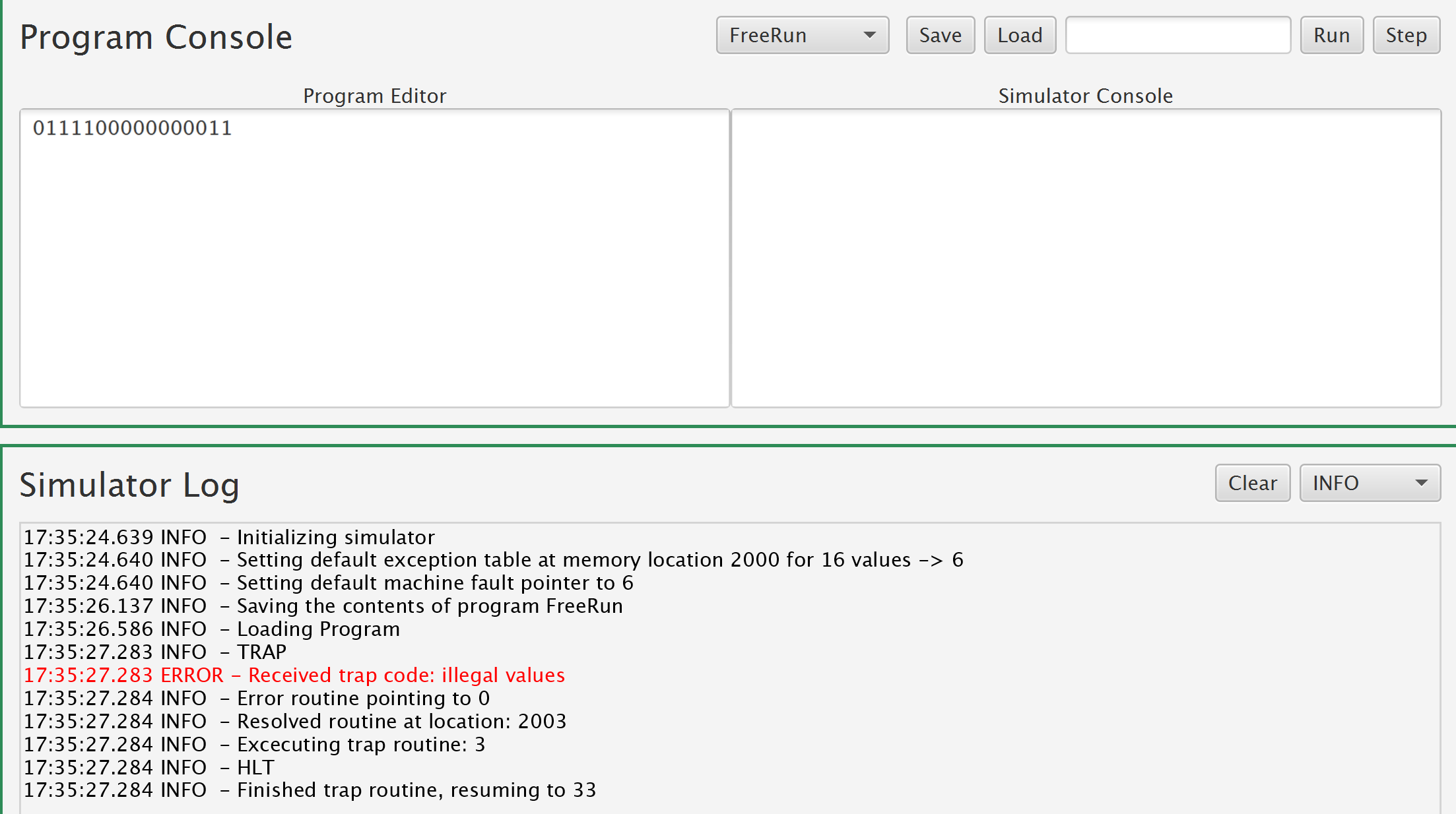
**0111100000000001 Trap 1: IllegalMemoryAccess**



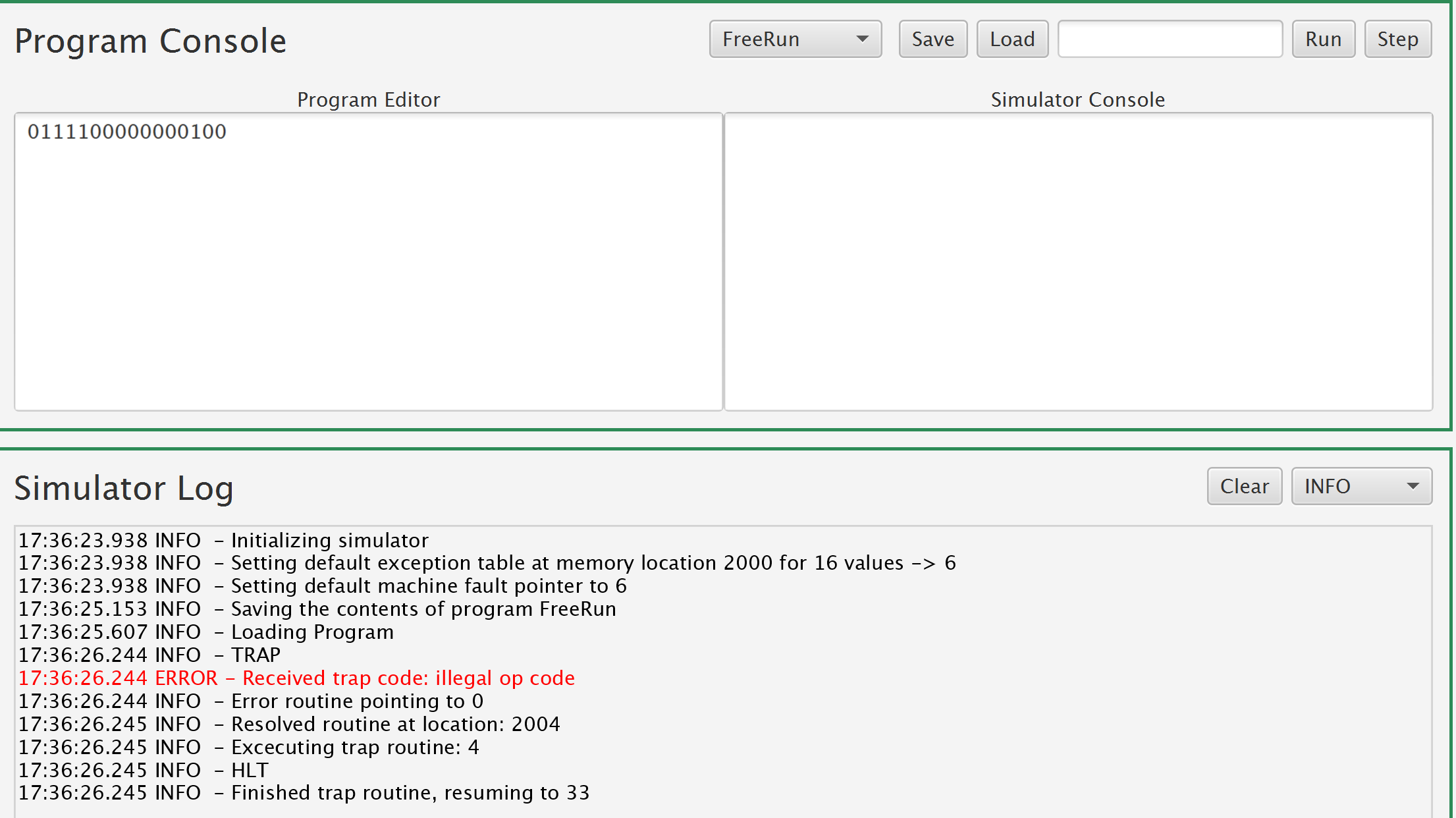
**0111100000000010 Trap 2: IllegalTrapCode**



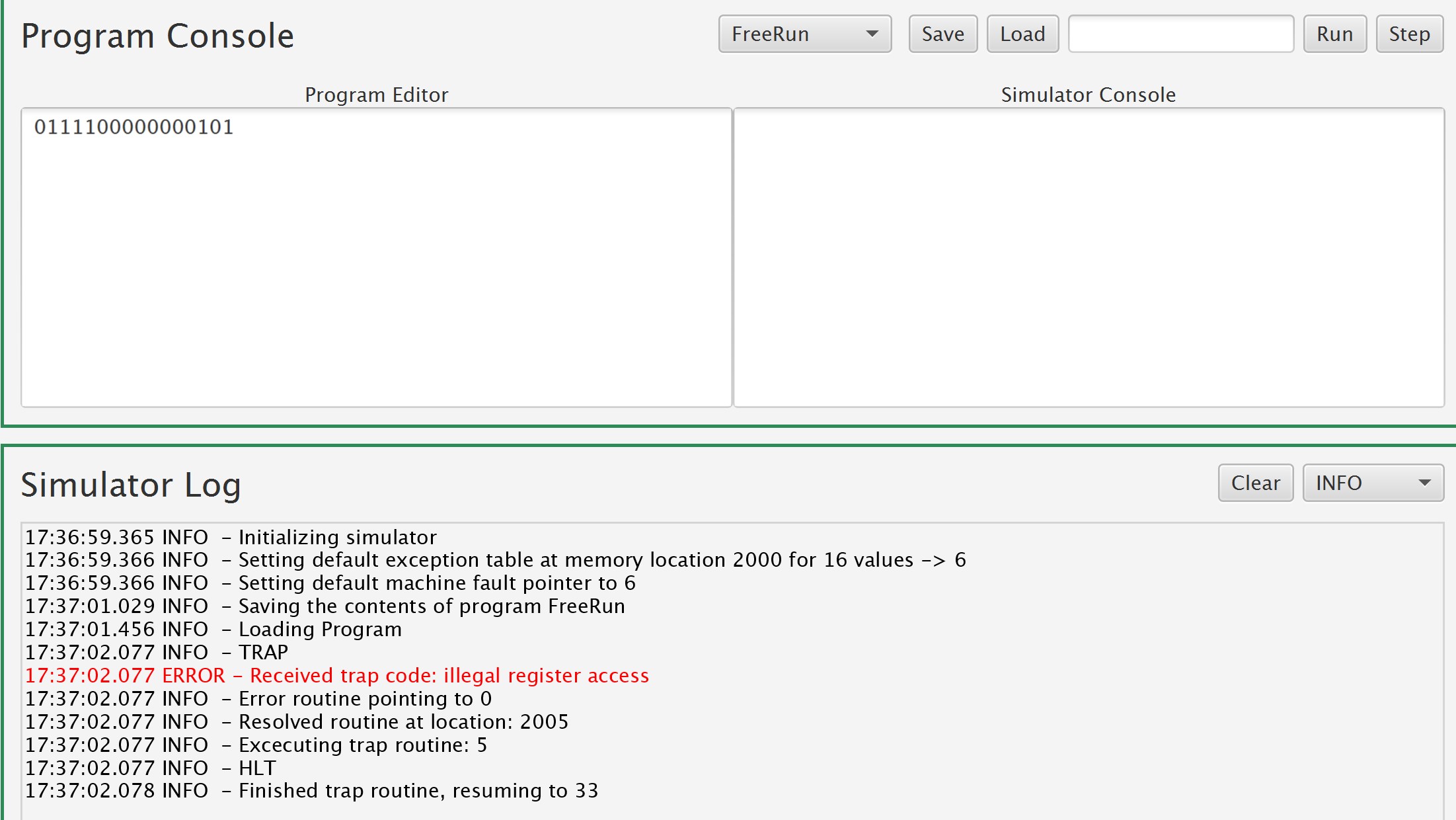
**0111100000000011 Trap 3: IllegalValue**



**0111100000000100 Trap 4: IllegalOpcode**



**0111100000000101Trap 5: IllegalRegisterAccess**



**0111100000001000 Trap 8: MemoryOutOfBounds**

