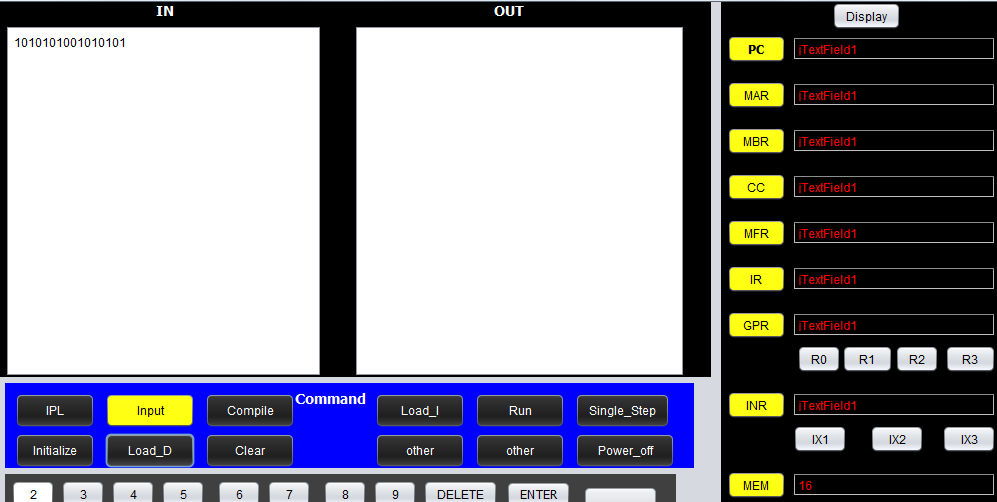
**Test Cases**

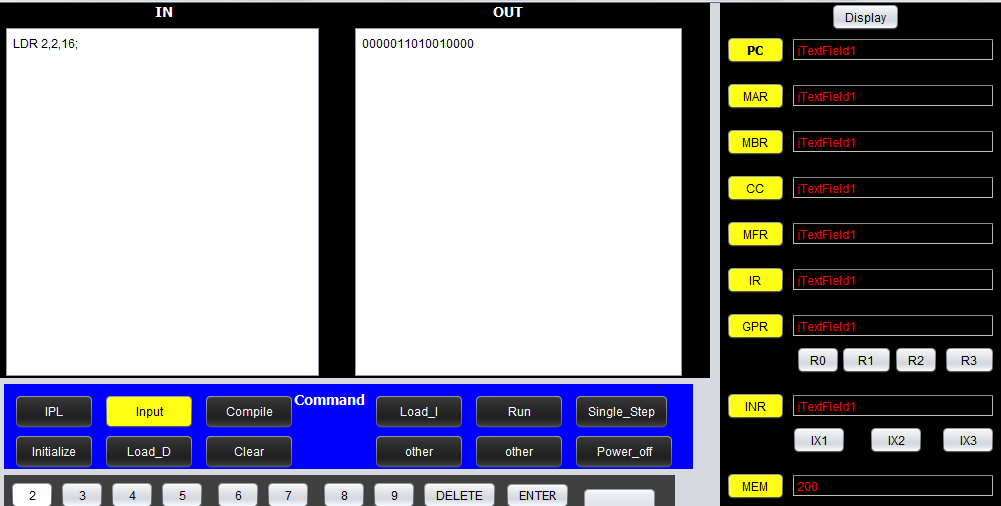
**Load/Store Instructions:**

1.LDR:

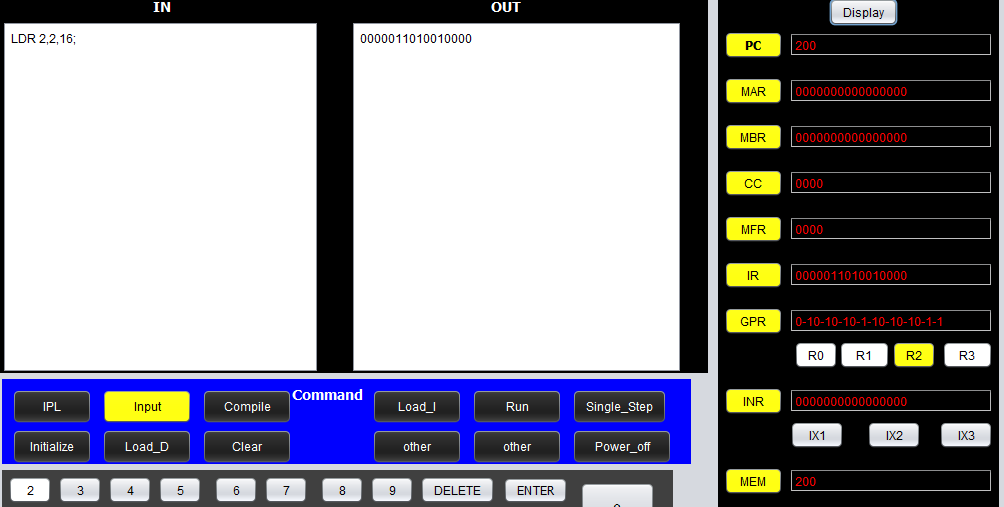
(1).Enter data into memory, click the Input button to make sure its color is green. Then use the keyboard to input data into the input text area. Click input button again to make sure its color turn into yellow. Enter address data into Mem text field(enter data when button is green, and turn it into yellow when finished). Finally, click the Load\_D button.



(2).Enter instruction into memory, make sure the input text field is empty, if it is not, click the Clear button. Click the Input button to make sure its color is green. Then use the keyboard to input instruction into the input text area. Click input button again to make sure its color turn into yellow. Then click Compile button, Enter address data into Mem text field(enter data when button is green, and turn it into yellow when finished). Finally, click the Load\_I button.

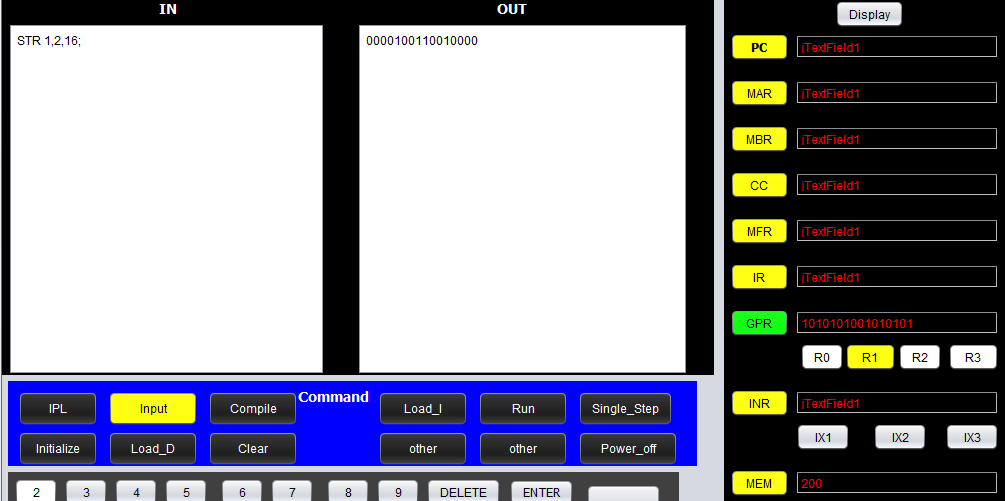


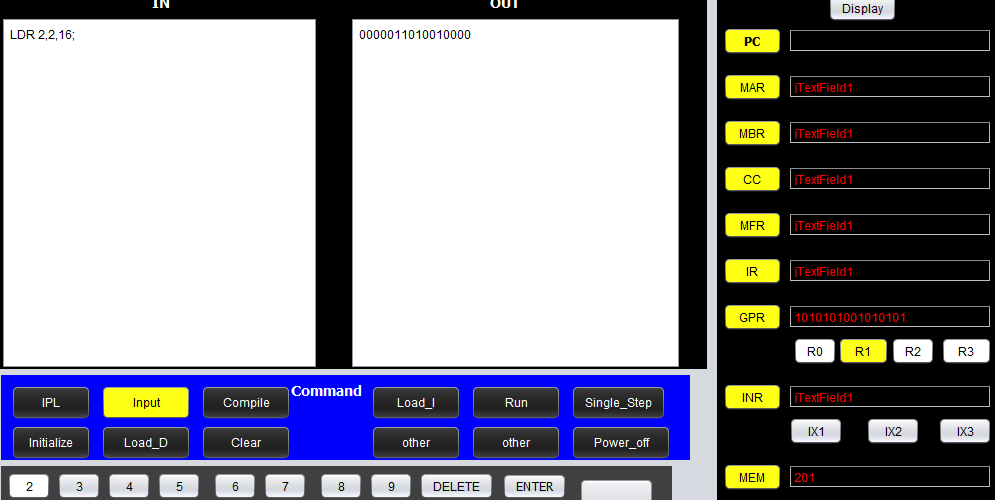
(3).Execute the program, first of all, set value of PC point to the first instruction of the program(input when green, yellow to finish). Then click the run or single\_step button. The program will be executed, we can click display button to check the result. The content of R2 shows that our instruction has been executed perfectly.



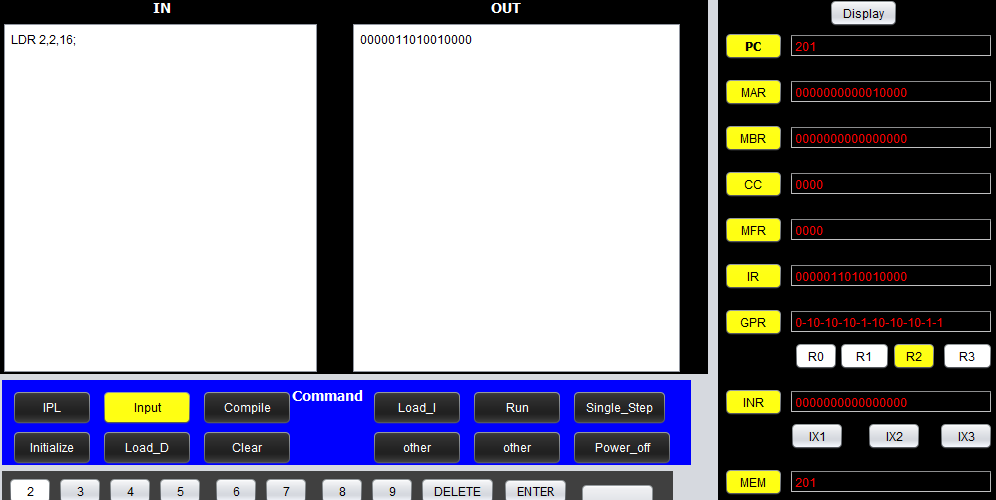
2.STR:

(1).Enter instruction into memory, click the Input button to make sure its color is green. Then use the keyboard to input instruction into the input text area. Click input button again to make sure its color turn into yellow. Then click Compile button, Enter address data into Mem text field(enter data when button is green, and turn it into yellow when finished). Finally, click the Load\_I button. To check the instruction is executed well, we need to read the data in the location of the memory where the STR instruction put data into. First of all, make sure the input text field is empty, if it is not, click the Clear button. Click the Input button to make sure its color is green. Then use the keyboard to input instruction into the input text area. Click input button again to make sure its color turn into yellow. Then click Compile button, Enter address data(the next address of the STR instruction) into Mem text field(enter data when button is green, and turn it into yellow when finished). Finally, click the Load\_I button.





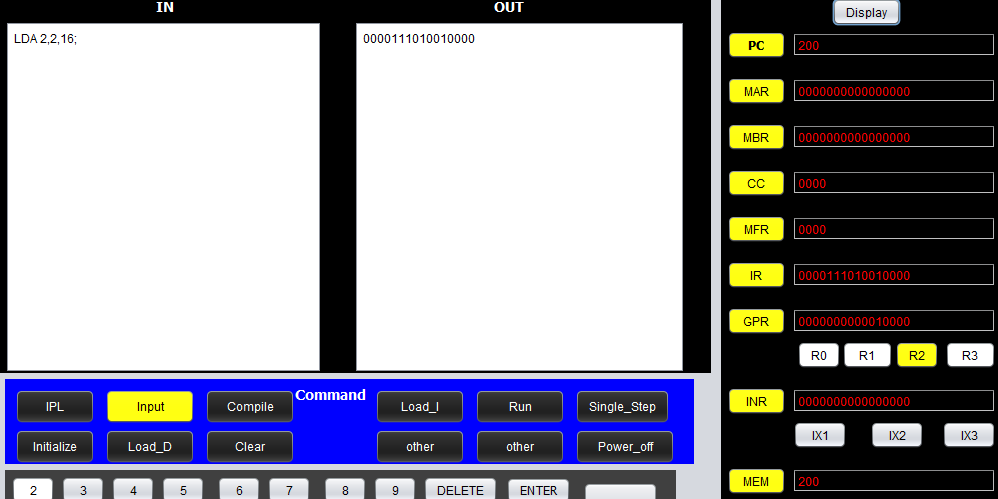
(2).Execute the program, first of all, set value of PC point to the first instruction of the program(input when green, yellow to finish). Then click the run or single\_step button. The program will be executed, we can click display button to check the result. The content of R1 shows that our instruction has been executed perfectly.



3.LDA:

(1).Enter instruction into memory, click the Input button to make sure its color is green. Then use the keyboard to input instruction into the input text area. Click input button again to make sure its color turn into yellow. Then click Compile button, Enter address data into Mem text field(enter data when button is green, and turn it into yellow when finished). Finally, click the Load\_I button.

(2).Execute the program, first of all, set value of PC point to the first instruction of the program(input when green, yellow to finish). Then click the run or single\_step button. The program will be executed, we can click display button to check the result. The content of R2 shows that our instruction has been executed perfectly.



4. LDX

(1) Enter data into memory:

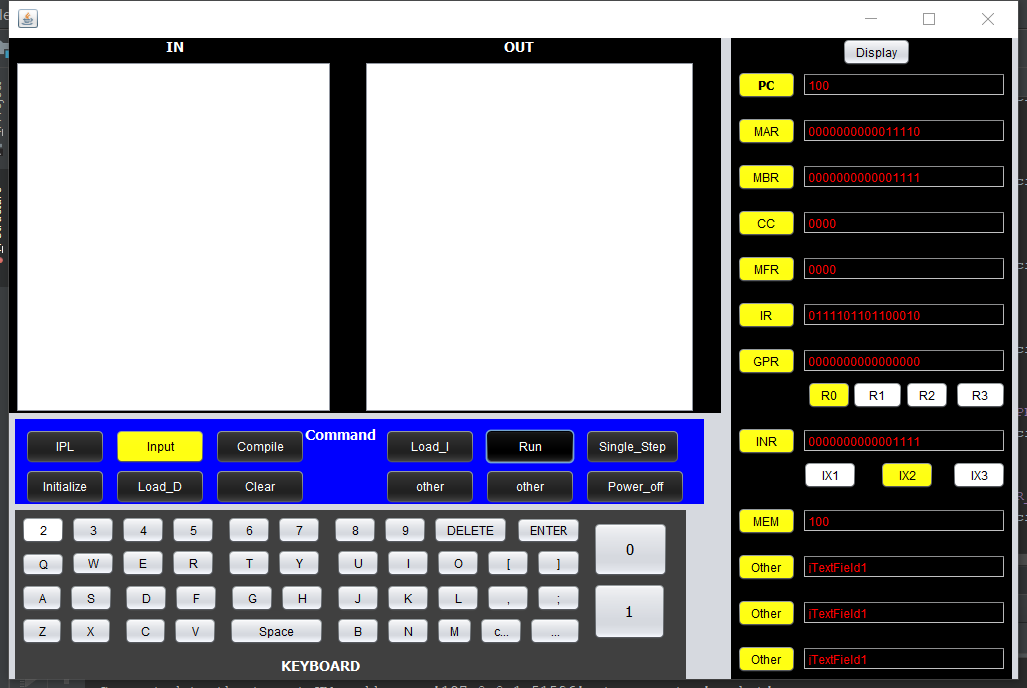
Click the button Initialize to initialize the memory;

Click the button Clear to clear the contents of Input and output text area;

Click the button Input to make sure the color of Input button is green, then input data “0000000000001111” into input text area. Click the button Input to make sure the color of input button is yellow.

Click the MeM button to make sure the color of MeM is green, input memory address 30 in the Mem text field, then click the Mem button.

Click Load\_D button.



（2）Enter instruction into memory

Click the button Clear to clear the contents of Inut and output text area;

Click the button Input to input instructions “LDX 2,30” in the text area, Click Input button again. Click the compile button to compile the instruction. Input memory address “100” in the MeM text field. Then Click Load\_I button.

(3) Click PC button to input memory address “100”, then Click PC button to make sure its color turn to yellow. Then click Single\_Step button. Then Click Display button to display the result.

We find that the result in IX2 is the same with the Input data.

5. STX

(1) enter “0000000000000001” into Index register1

Enter “0000000000000011” into index register2

(2) enter instructions “ STX 1,30” into memory location 100;

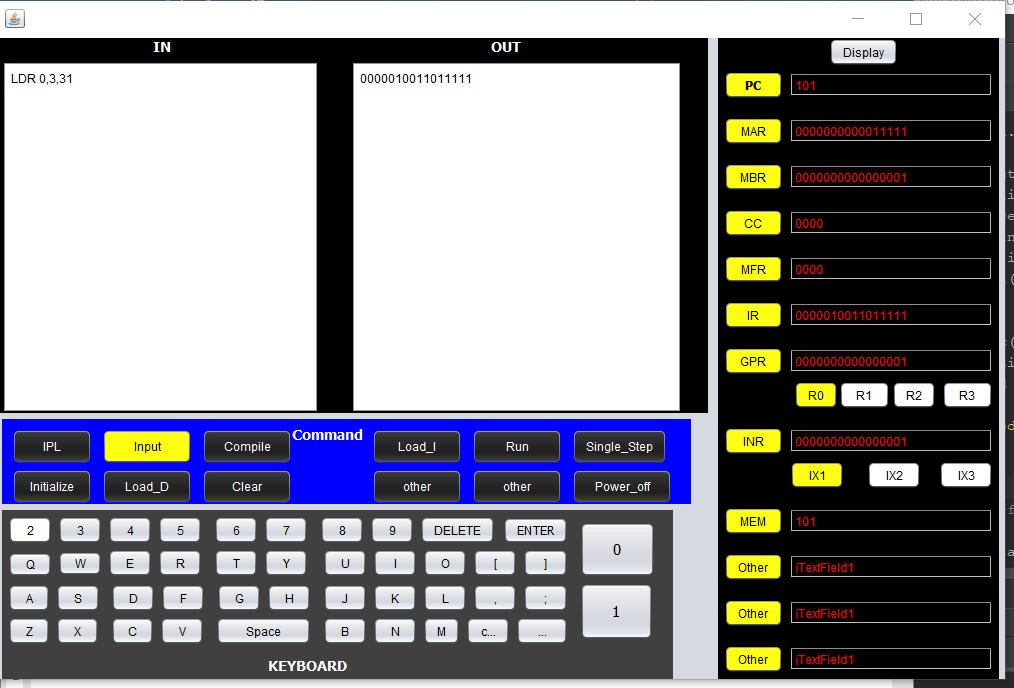
Enter instructions “ LDR 0,3,31” into memory location 101;

(3) set the PC as “100”, then Click the Single\_step button.

Set the PC as “101”, Click the single\_step button again.

Click the display button.

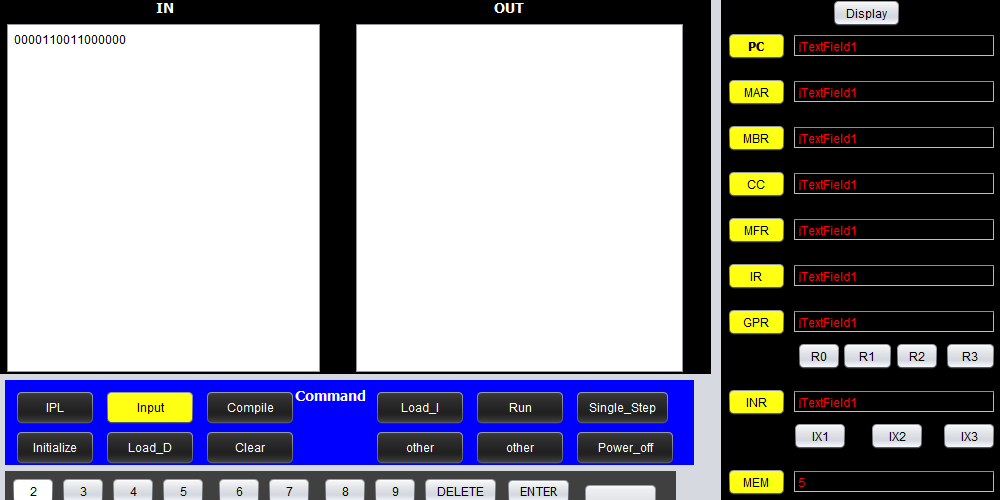
We can the find the content in index register IX1 is the same as that of GPR R0.

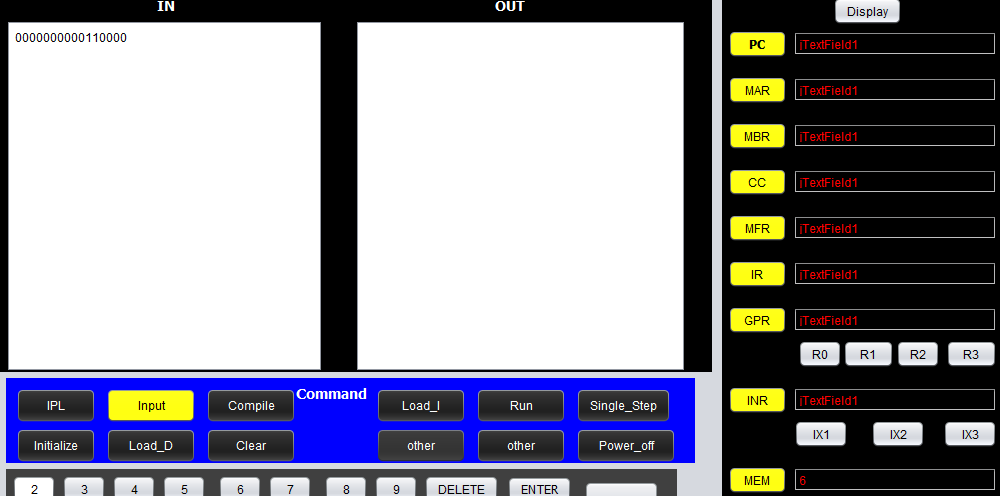


**Arithmetic and Logical Instructions**:

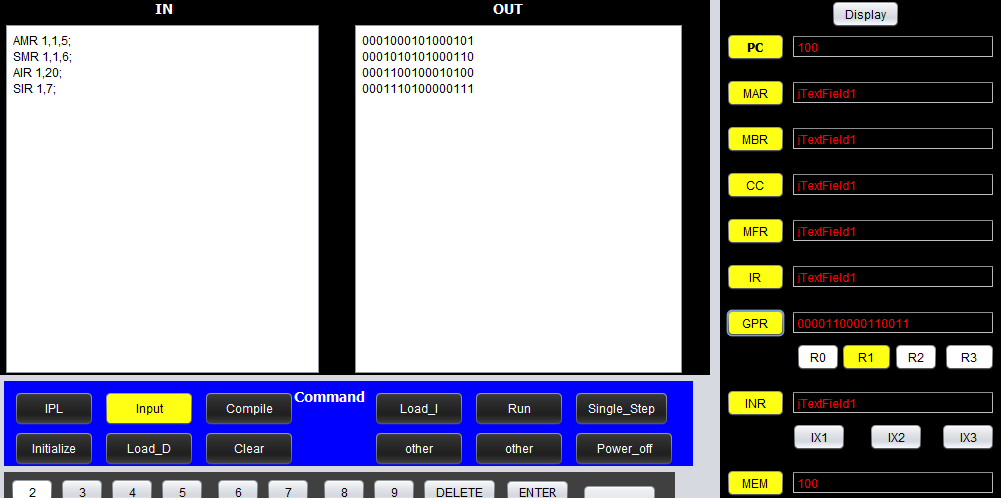
1.AMR+SMR+AIR+SIR:

1)Load data into memory: For example, we put 0000110011000000 into M[5]; and put 0000000000110000 into M[6]:

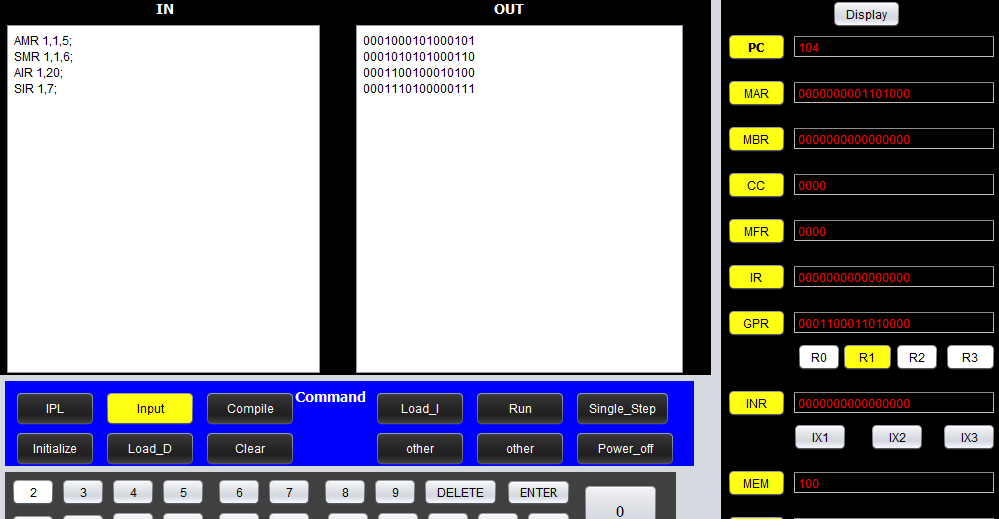




2) Load instruction into memory: For example, we put AMR 1,1,5; into M[100];put SMR 1,1,6; into M[101];put AIR 1,20; into M[102];put SIR 1,7; into M[103]. And set the value of R1 as 0000110000110011; set value of PC is 100. Then hit the run/single\_step button to operate the program.

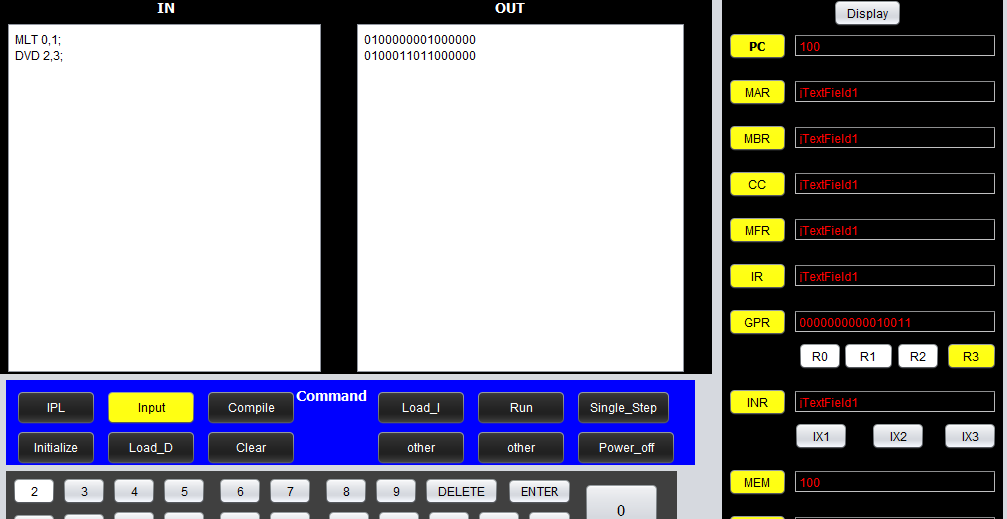


3)Then we can get the result:

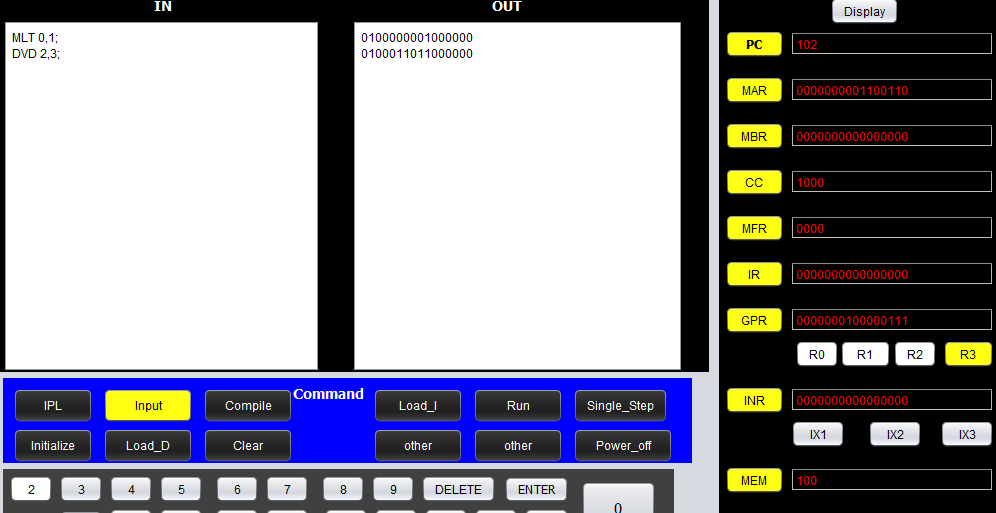


2.MLT+DVD:

1) Load instruction into memory: For example, we put MLT 0,1; into M[100];put DVD 2,3; into M[101]; and set the value of R0 as 0000001100110011; set the value of R1 as 0000000010110011; set the value of R2 as 0000110000110011; set the value of R3 as 0000000000010011;set value of PC as 100. Then hit the run/single\_step button to process the program.

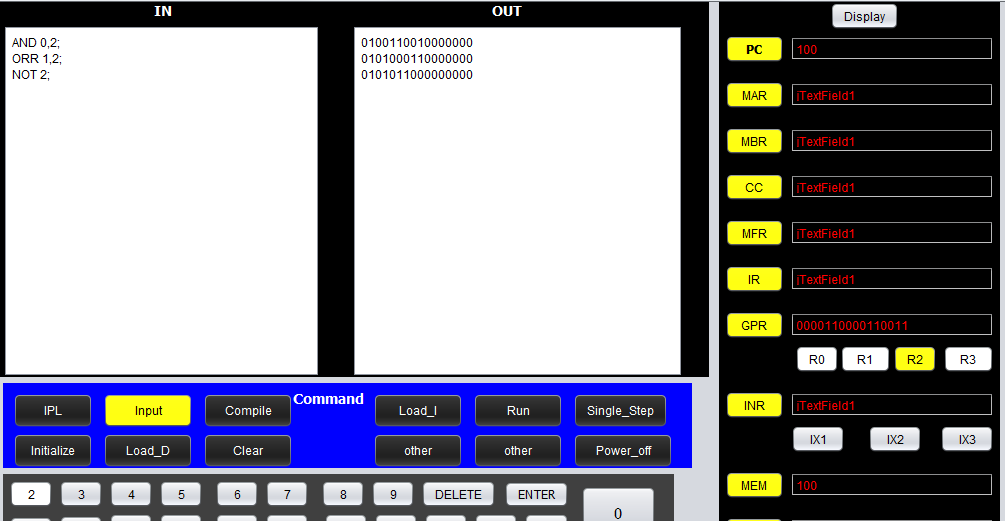


2)Then we can get the result:

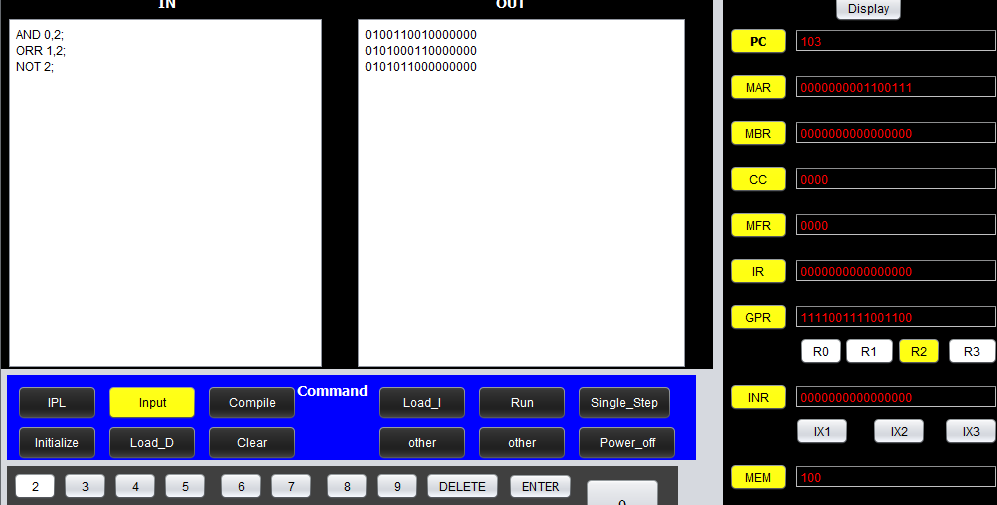


2.AND+ORR+NOT:

1) Load instruction into memory: For example, we put ADD 0,2; into M[100];put ORR 1,2; into M[101]; put NOT 2; into M[102]; and set the value of R0 as 0000001100110011; set the value of R1 as 0000000010110011; set the value of R2 as 0000110000110011; set value of PC as 100. Then hit the run/single\_step button to process the program.



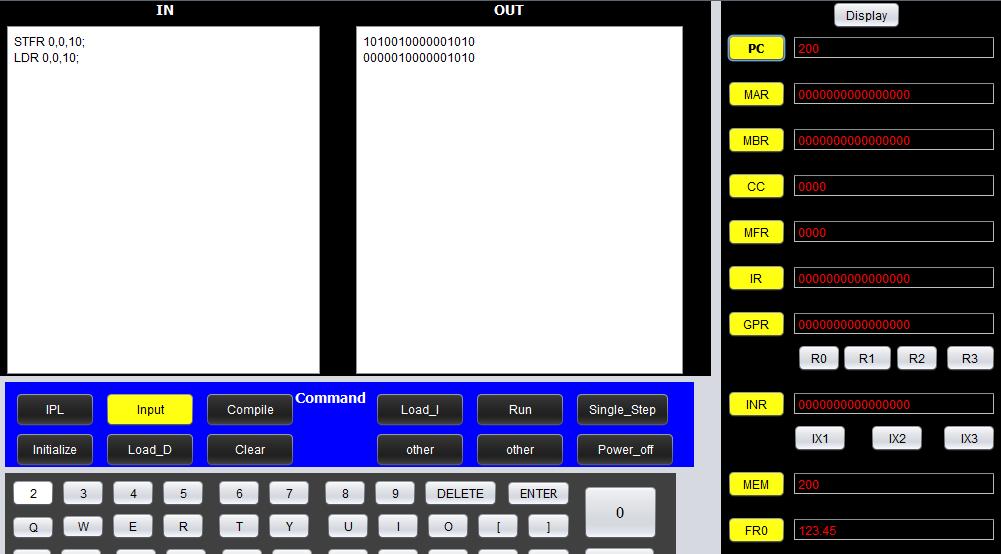
2)Then we can get the result:



**Test Floating and Vector Instructions**

**STFR:**

1)First of all, we test the STFR instruction, Load instruction into memory: For example, we put STFR 0,0,10; into M[200];put LDR 0,0,10; into M[201]. The first instruction store the value of fr0 into memory M[10], the second instruction load the data in the memory M[10] into register R0. So if we can get a new value of R0 after the program has been processed, it will prove that the STFR instruction working perfectly.

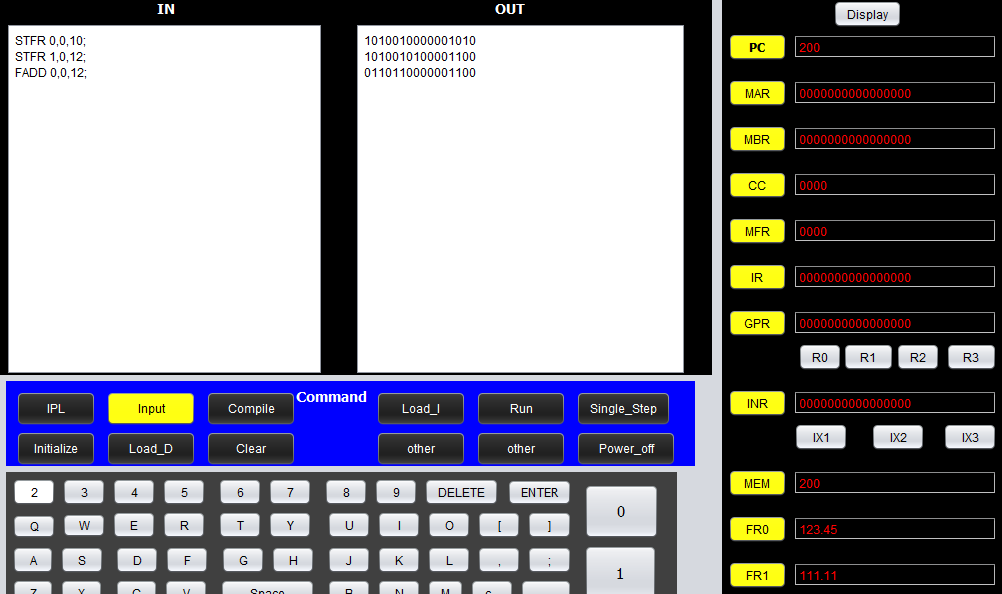


2)Then we can get the result, and the value of R0 has been updated, our STFR instruction is right:

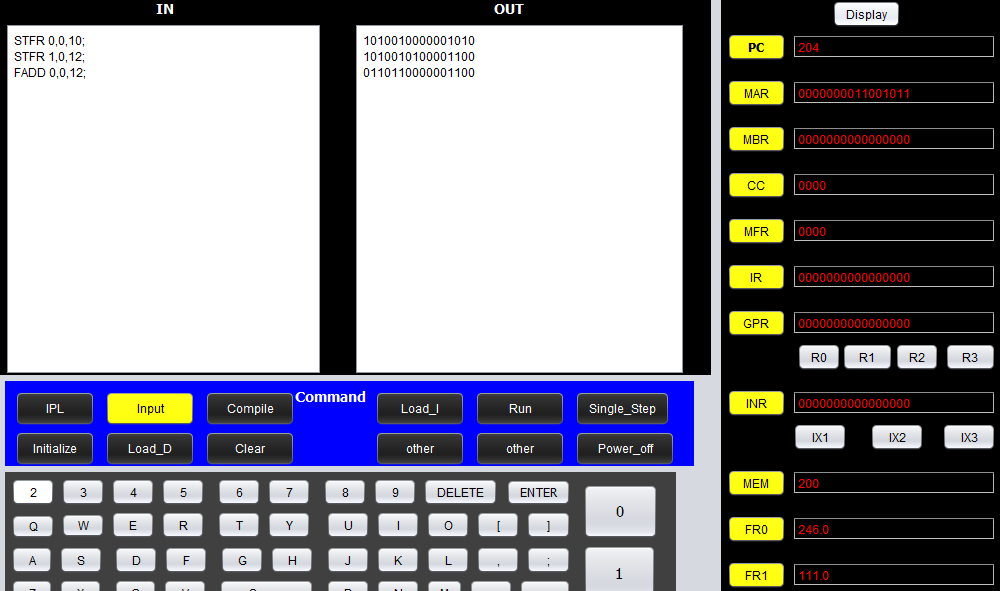


**FADD\FSUB:**

1)Then, we test the FADD\FSUB instruction, Load instruction into memory: For example, we put STFR 0,0,10; into M[200];put STFR 1,0,12; into M[201]; FADD 0,0,12; into M[202]. The first instruction store the value of fr0 into memory M[10], the second instruction store the value of fr1 into memory M[12], the third instruction add the value of fr0 with the value of M[12]. So if we can get a new value of fr0 after the program has been processed, it will prove that the FADD\FSUB instruction working perfectly.

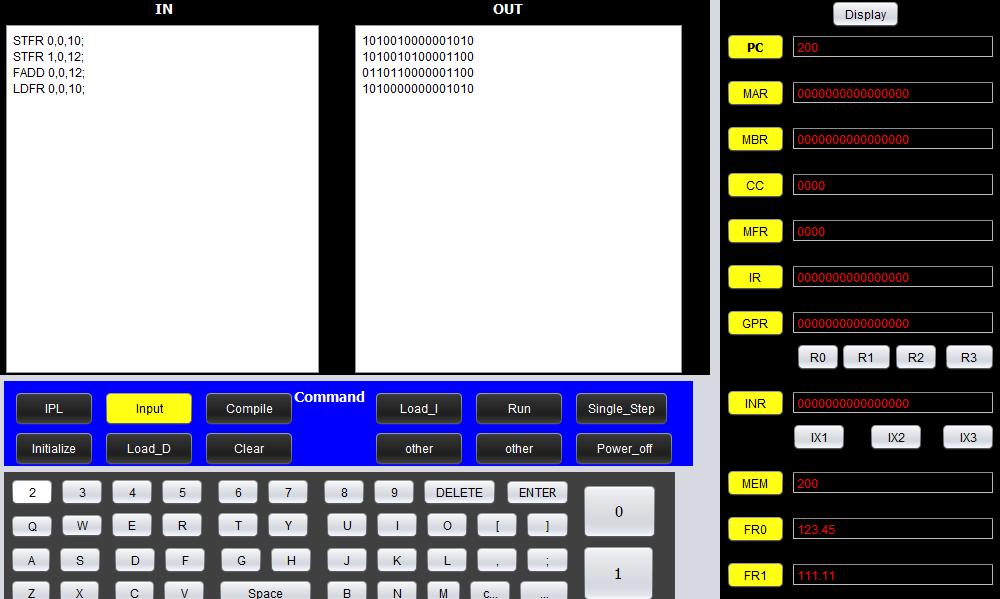


2)Then we can get the result, and the value of fr0 has been undated, our FADD\FSUB instruction is right:(due to the precision of this simulator, a little deviation exists)

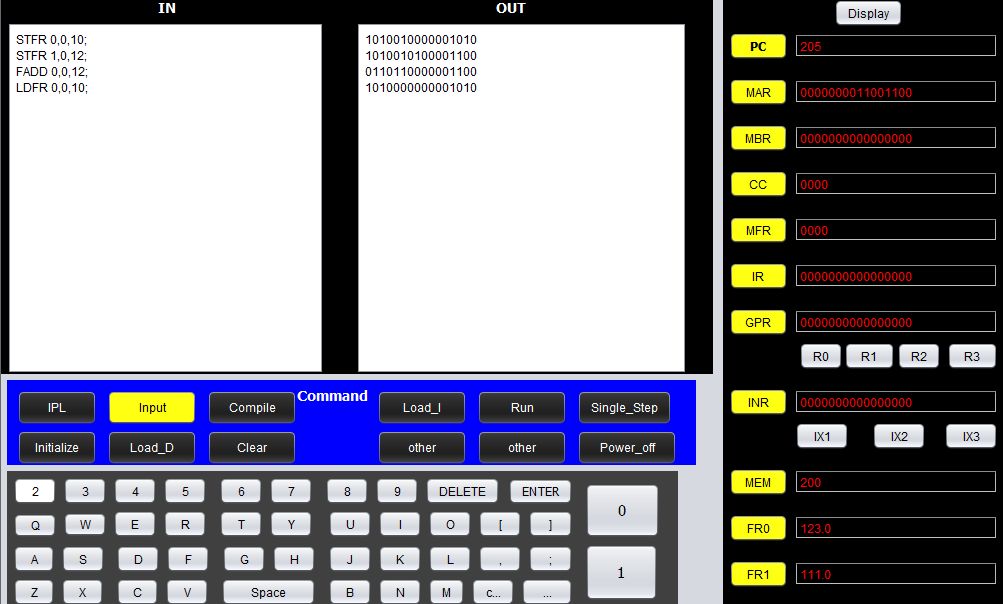


**LDFR:**

1)Then, we test the LDFR instruction, Load instruction into memory: For example, we put STFR 0,0,10; into M[200];put STFR 1,0,12; into M[201]; FADD 0,0,12; into M[202]; LDFR 0,0,10; into M[203]. After the third instruction has been processed, the value of fr0 will be changed, we have proved this point in the FADD test, the fourth instruction will change the value of fr0 with the value of M[10], which is the original value of fr0, . So if we can get the original value of fr0 after the program has been processed, it will prove that the LDFR instruction working perfectly.

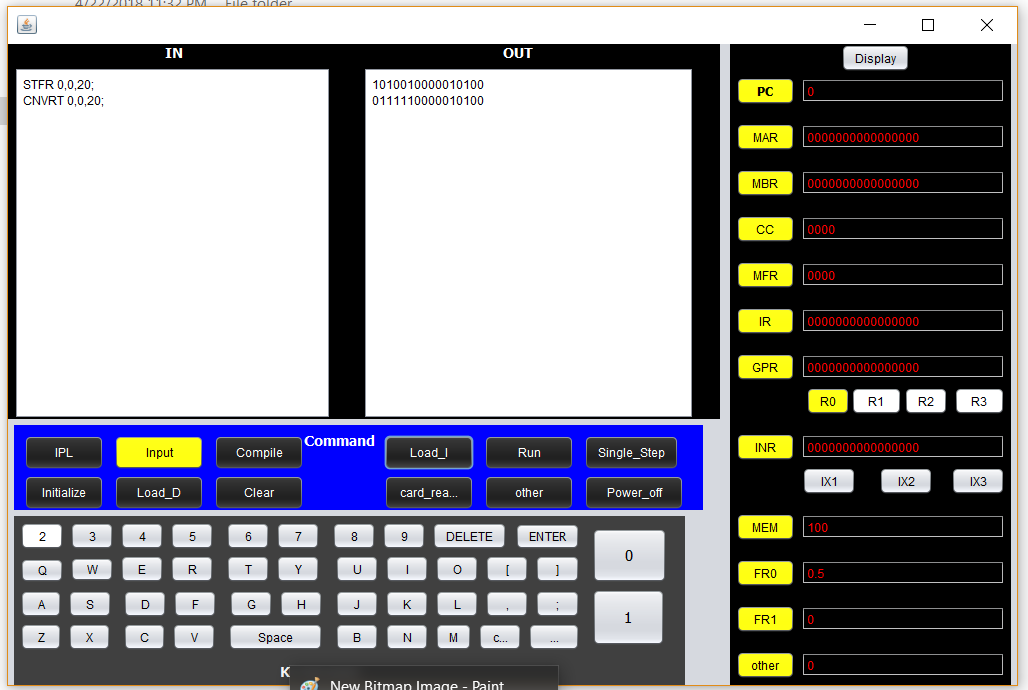


2)Then we can get the result, and the value of fr0 is same:

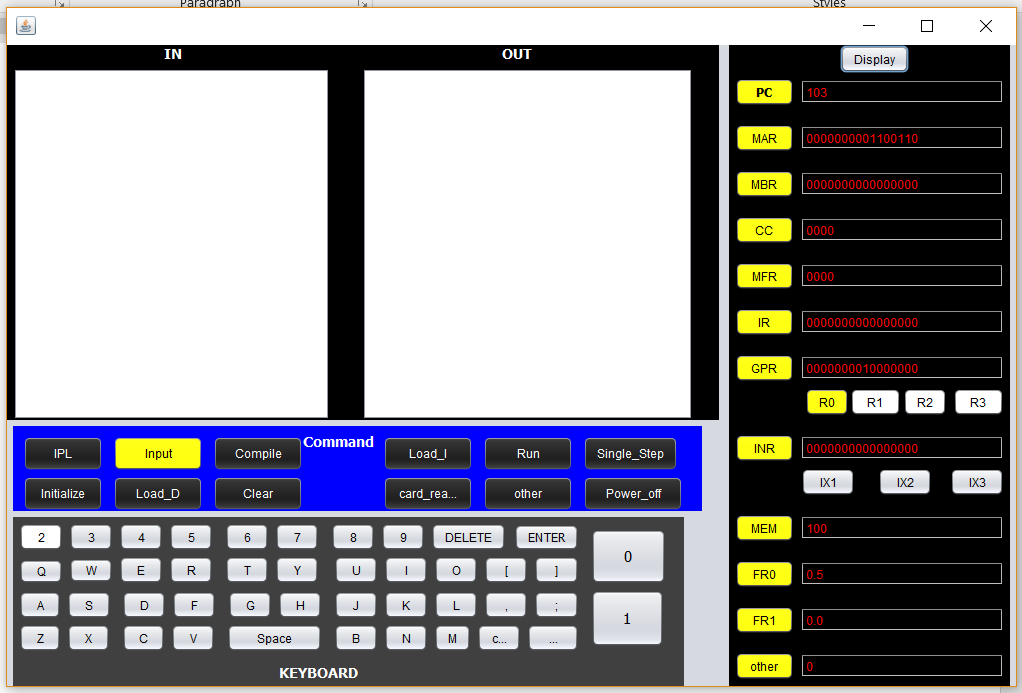


**CNVRT:**

Set FR0 = 0.5, GPR0 = 0, Execute the instruction “STFR 0,0,20; CNVRT 0,0,20;”



The correct result should be GPR0 = “000000001000000”, and the following diagram prove the result is correct.

****

**VADD\VSUB:**

set the GPR0 =1,GPR1=1,GPR2 =1, FR0 = 3.1

Program:

STR 0,0,13;

STR 1,0,14;

STR 2,0,15;

STR 0,0,10;

STR 1,0,11;

STR 2,0,12;

AIR 0,9;

AIR 1,12;

STR 0,0,20;

STR 1,0,21;

VADD 0,0,20;

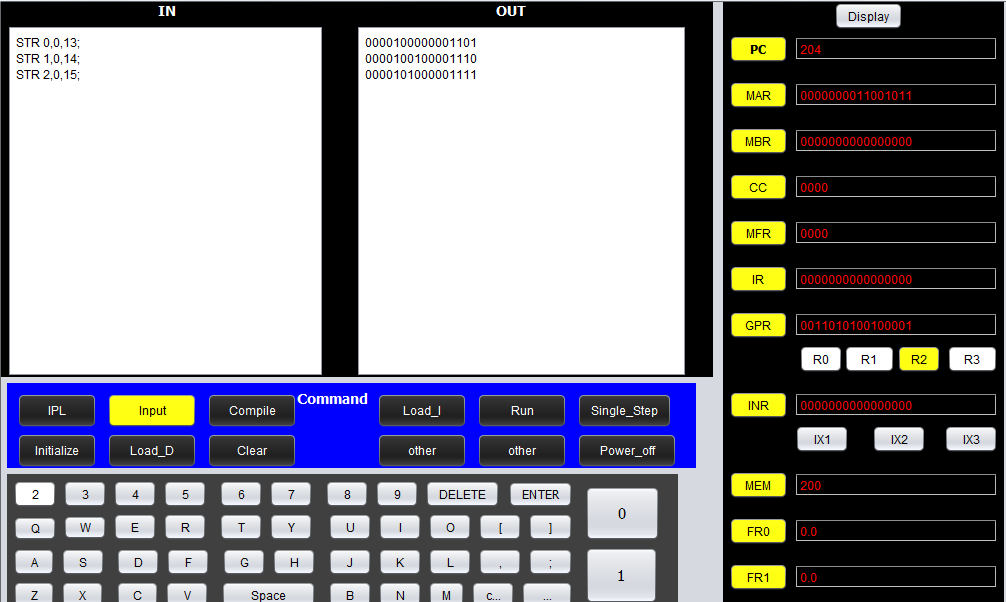
LDX 1,10;

LDX 2,11;

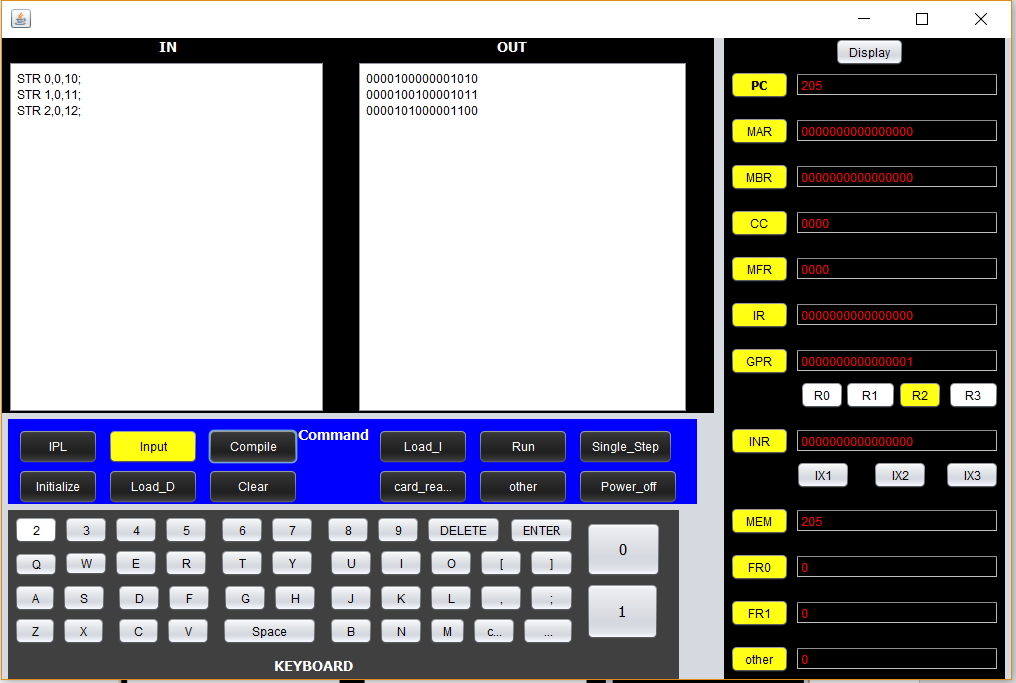
LDX 3,12;

The correct result should be : IX1 = 2, IX2 = 2, IX3 =2.

1)We test the VADD\VSUB instruction. First of all, we input three STR instruction into M[200] to store R0:1,R1:1,R2:1 into M[13]~M[15], set the PC as 200 and hit the run button to finish the storage of V2.

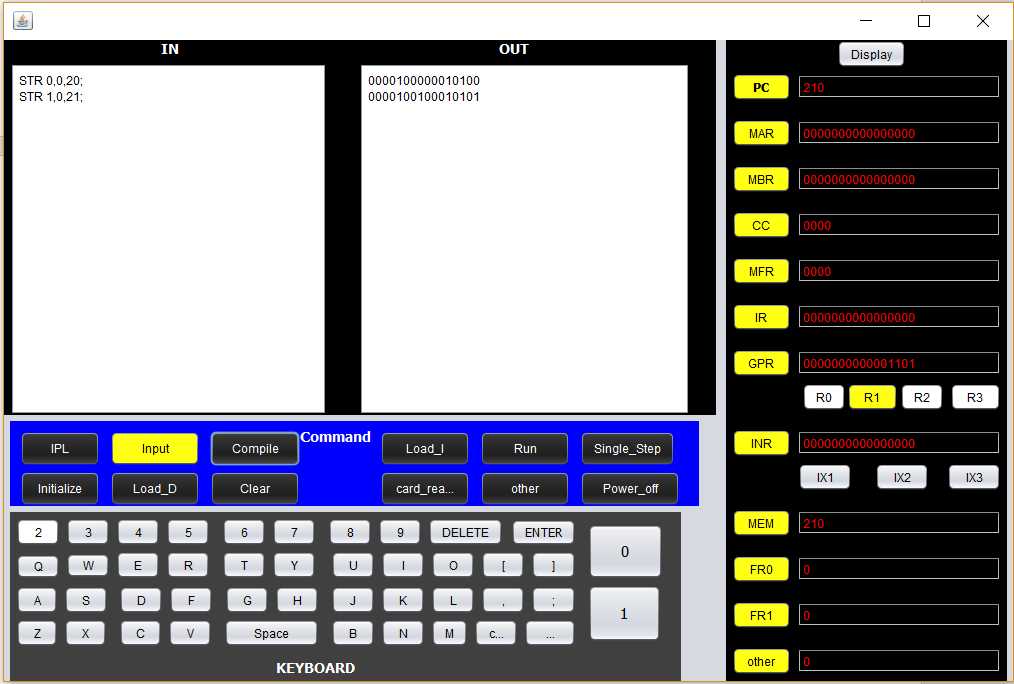


2)Then we input three STR instruction to M[205]. The aim of three STR instruction is to store three number R0:1,R1:1,R2:1 into M[10]~M[12]. Set the PC as 205 and hit run. Such that we store V1 in position 10,11,12 of memory.

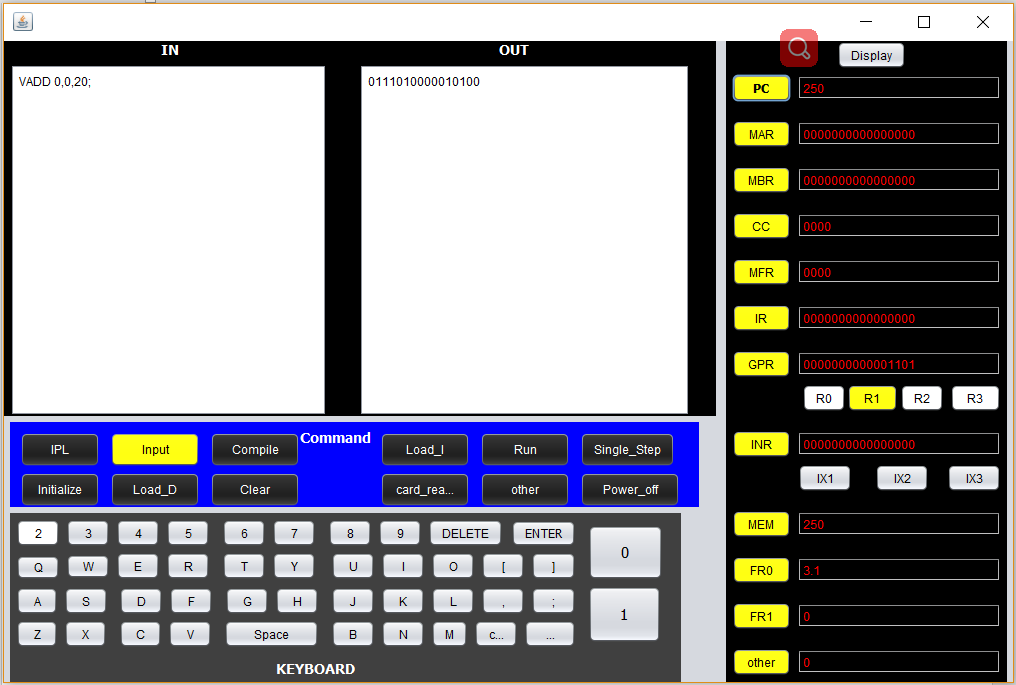


3)

Then we Store the memory address 10,13 in M[20],M[21]. 10 is the starting address of vetor1, 13 is the starting address of vector2.



4)



Use instruction “VADD 0,0,20;” to add vector1 to vector2. Set FR0 as 3.1.

5)



Load vector1 (M[10], M[11], M[12]) to index register, and check the result.