Instruction	Opcode	Function	Addressing Mode	Lanuch condition
JMP	1 1 0 0 0 0 0 0	Jump to certain address	/	
JC	1 1 1 0 0 1 0 0	Conditional jump	/	Carry: 1
JNC	1 1 1 0 0 0 0 0	Conditional jump	/	Carry: 0
JNN	1 1 0 1 0 0 0 0	Conditional jump	/	Negative:0
JZ	1 1 0 0 1 0 0 1	Conditional jump	/	Zero:1
JNZ	1 1 0 0 1 0 0 0	Conditional jump	/	Zero:0
JCN	1 1 1 1 0 1 1 0	Conditional jump	/	Carry:1 Negative:1
JNCN	1 1 1 1 0 0 1 0	Conditional jump	/	Carry:0 Negative:1
JCNN	1 1 1 1 0 1 0 0	Conditional jump	/	Carry 1 Negative:0
JNCNN	1 1 1 1 0 0 0 0	Conditional jump	/	Carry:0 Negative:0
JCZ	1 1 1 0 1 1 0 1	Conditional jump	/	Carry:1 Zero:1
JNCZ	1 1 1 0 1 0 0 1	Conditional jump	/	Carry:0 Zero:1
JCNZ	1 1 1 0 1 1 0 0	Conditional jump	/	Carry:1 Zero:0
JNCNZ	1 1 1 0 1 0 0 0	Conditional jump	/	Carry:0 Zero:0
JZN	1 1 0 1 1 0 1 1	Conditional jump	/	Zero:1 Negative:1
JNZN	1 1 0 1 1 0 0 1	Conditional jump	/	Zero:0 Negative:1
JNZNN	1 1 0 1 1 0 0 0	Conditional jump	/	Zero:0 Negative:0
JZNN	1 1 0 1 1 0 1 0	Conditional jump	/	Zero:1 Negative:0
Not Used	0 0 X X X X X X	Void opcode	/	/
	1 0 X X X X X X	Void opcode	/	/
LDA	1 0 0 0 0 0 1 0	Load Accumulator	Direct	/
STA	1 0 1 0 0 0 1 0	Save Accumulator	Direct	/
ADD	0 1 0 0 0 0 1 0	Add value to accumulator	Direct	/
ADD(immediate)	0 1 0 0 0 0 0 1	Add value to accumulator	Immediate	/

ADDC	0 1 0 0 1 0 1 0	Add value and carry to accumulator	Direct	/
ADDC(immediate)	0 1 0 0 1 0 0 1	Add value and carry to accumulator	Immediate	/
SUB	0 1 0 1 0 0 1 0	Subtract value from accumulator	Direct	/
SUB(Immediate)	0 1 0 1 0 0 0 1	Subtract value from accumulator	Immediate	/
SUBC	0 1 0 1 1 0 1 0	Subtract value and borrow from accumulator	Direct	/
SUBC(immediate)	0 1 0 1 1 0 0 1	Subtract value and borrow from accumulator	Immediate	/
INC	0 1 0 0 1 1 0 0	Increment accumulator	Inherent	/
DEC	0 1 0 0 0 1 0 0	Decrement accumulator	Inherent	/
AND	0 1 1 0 0 0 1 0	Logically AND the accumulator and value	Direct	/
AND(immediate)	0 1 1 0 0 0 0 1	Logically AND the accumulator and value	Immediate	/
OR	0 1 1 1 0 0 1 0	Logically OR the accumulator and value	Direct	/
OR(immediate)	0 1 1 1 0 0 0 1	Logically OR the accumulator and value	Immediate	/
INV	0 1 1 0 0 1 0 0	Logically invert the accumulator	Inherent	/
XOR	0 1 1 0 1 1 1 0	Logically XOR the accumulator and value	Direct	/
XOR(immediate)	0 1 1 0 1 1 0 1	Logically XOR the accumulator and value	Immediate	/
CLRA	0 1 1 0 0 1 0 0	Clear Accumulator	Inherent	/
CMP	0 1 1 1 1 1 1 0	Compare (executes Accumulator – argument) does not modify accumulator	Direct	/
CMP(immediate)	0 1 1 1 1 1 0 1	Compare (executes Accumulator – argument) does not modify accumulator	Inherent	/

Explain of button function:

Simulation:

Run: Single click this button to run your simulation based on loaded program.

Step: Single click this button to run your simulation step by step.

Reset: Single click this button to reset values in memory and all registers.

PC: Shows the value of program counter.

IR: Shows the value of instruction register.

AC: Shows the value of accumulator.

Date bus: shows the value on the data bus.

Condition signals:

The box is checked if there is carry/negative/zero happens during the process of simulation.

Memory Display: left block shows the instructions utilized in your code and their corresponding opcode. The right block contains the memory map and the values stored in specific memory address.

Program editor:

Open: Open a .txt file which contains your assembly code.

Save: Save your assembly code to a .txt format file

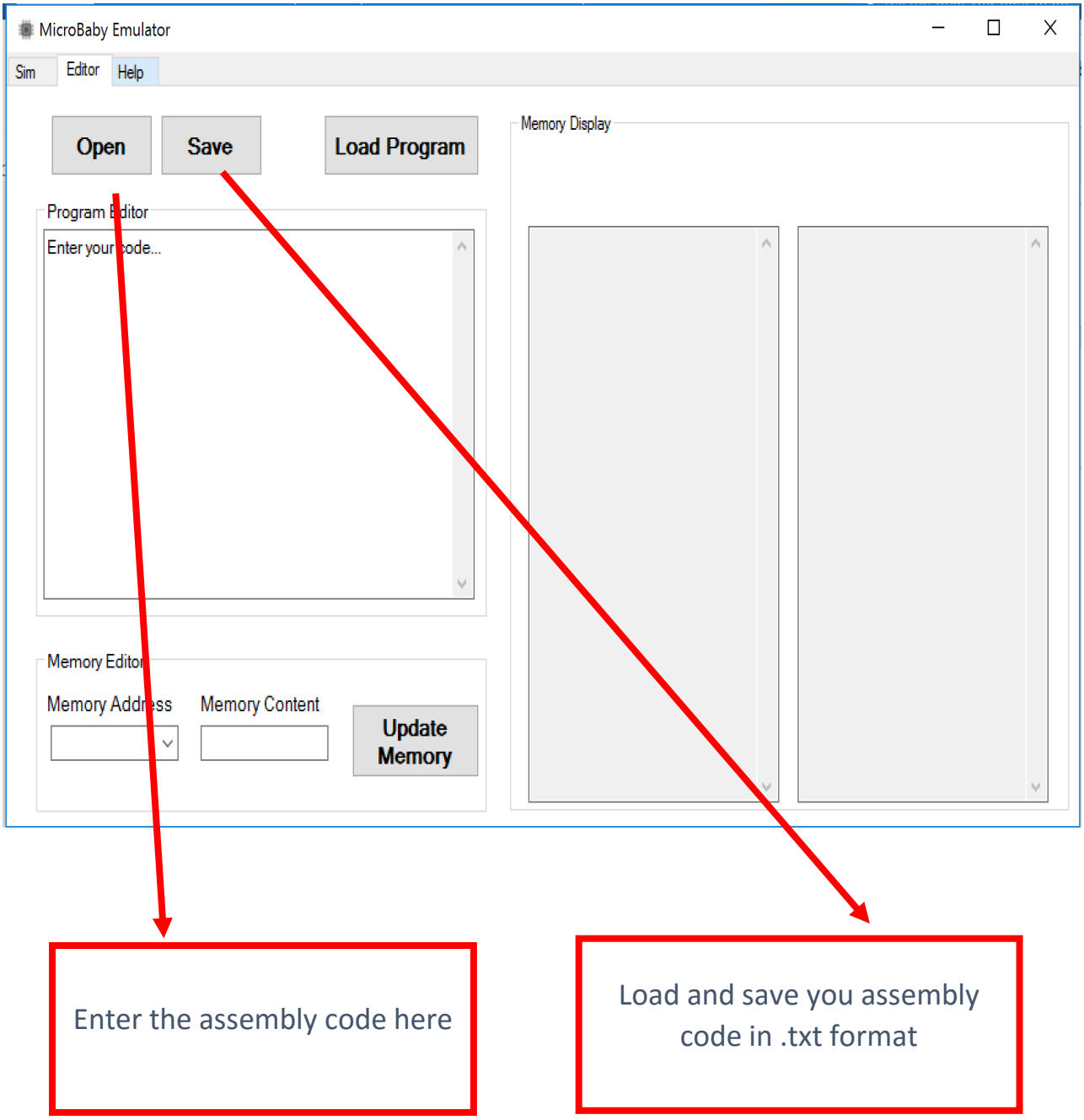
Load Program: Running the program based on the assembly code you have entered in the program editor.

Memory Display: left block shows the instructions utilized in your code and their corresponding opcode. The right block contains the memory map and the values stored in specific memory address.

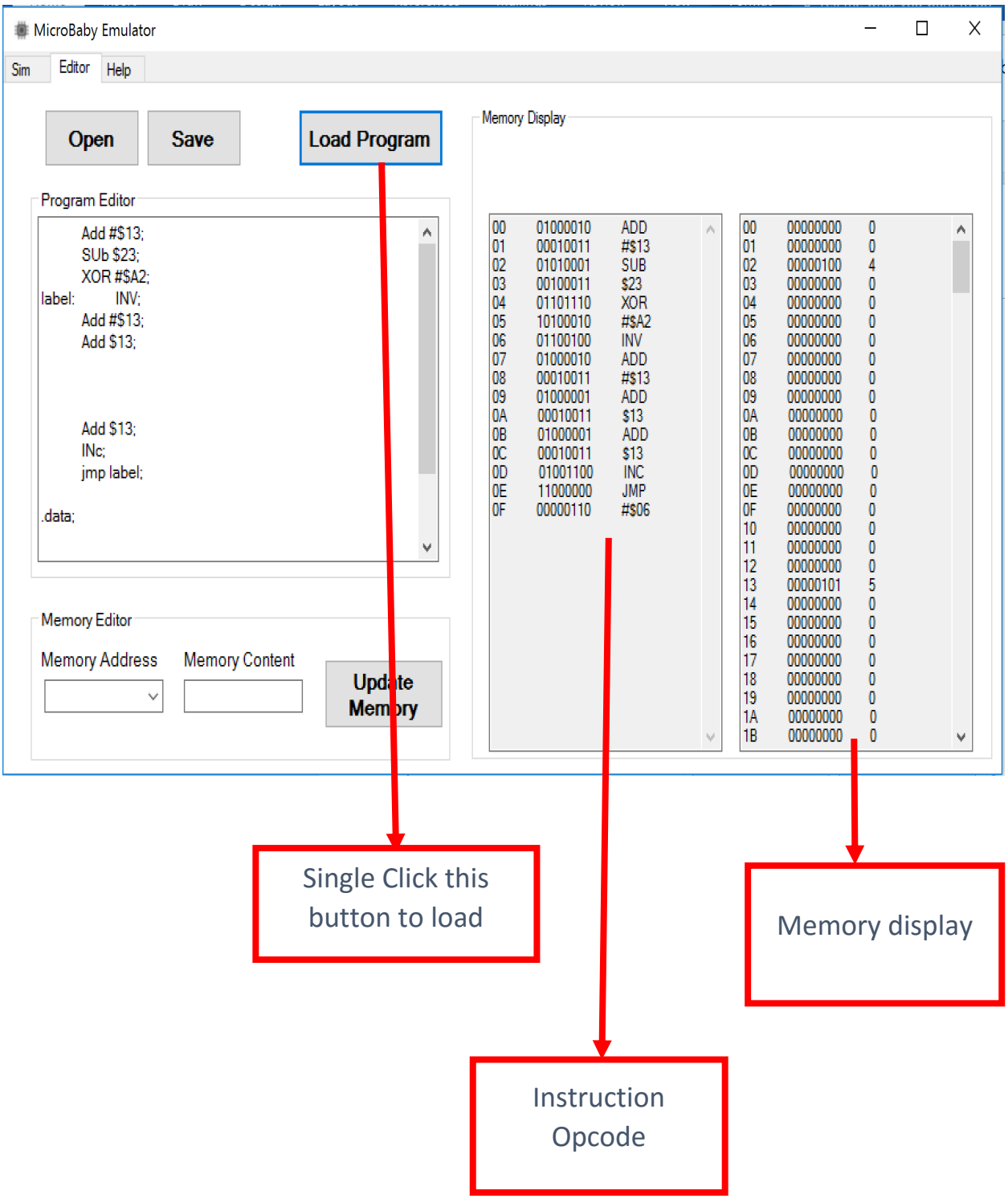
Memory Editor: the memory editor can assign values to specific memory address.

Program running instruction:

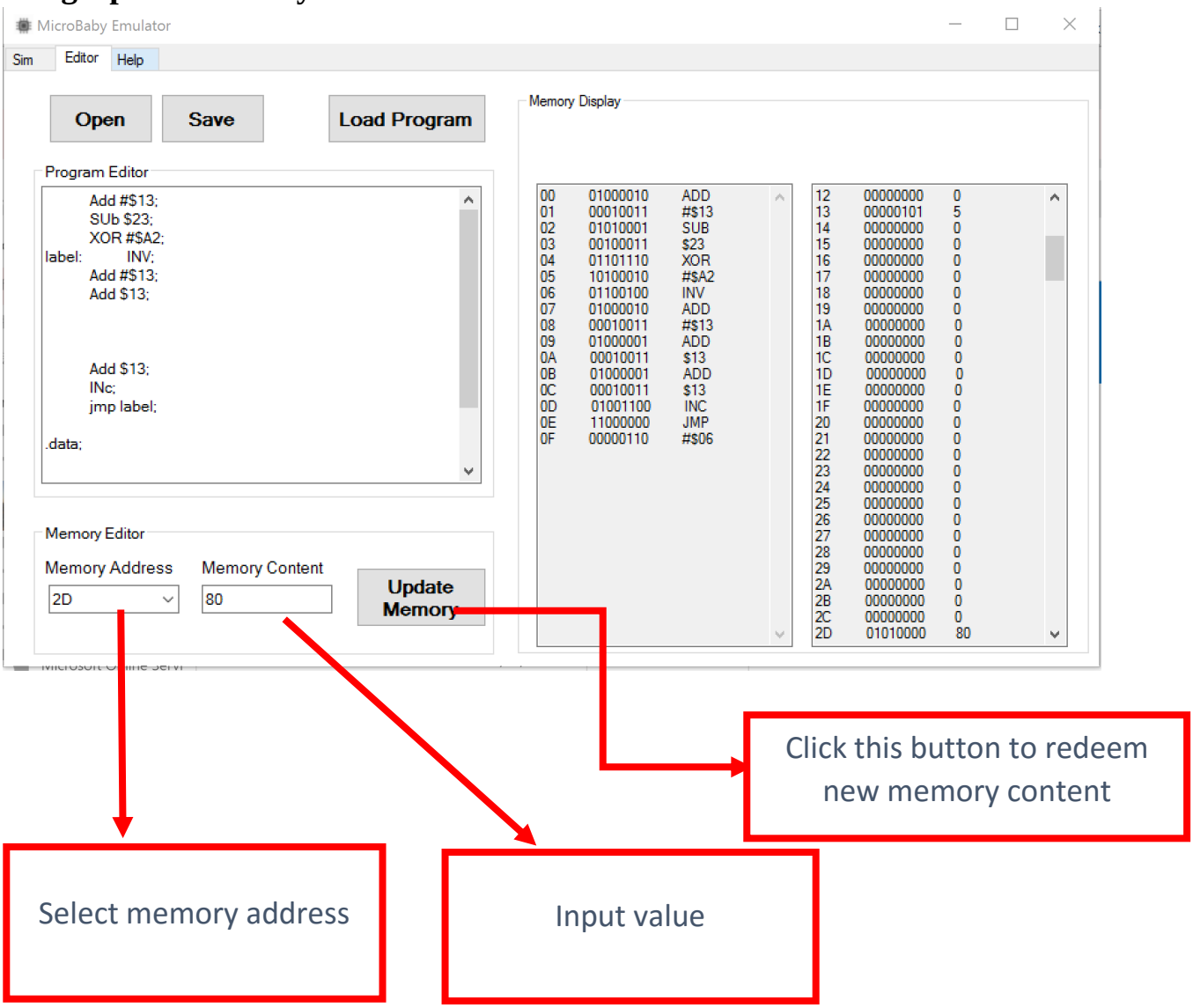
Step 1: Write or open your assembly code



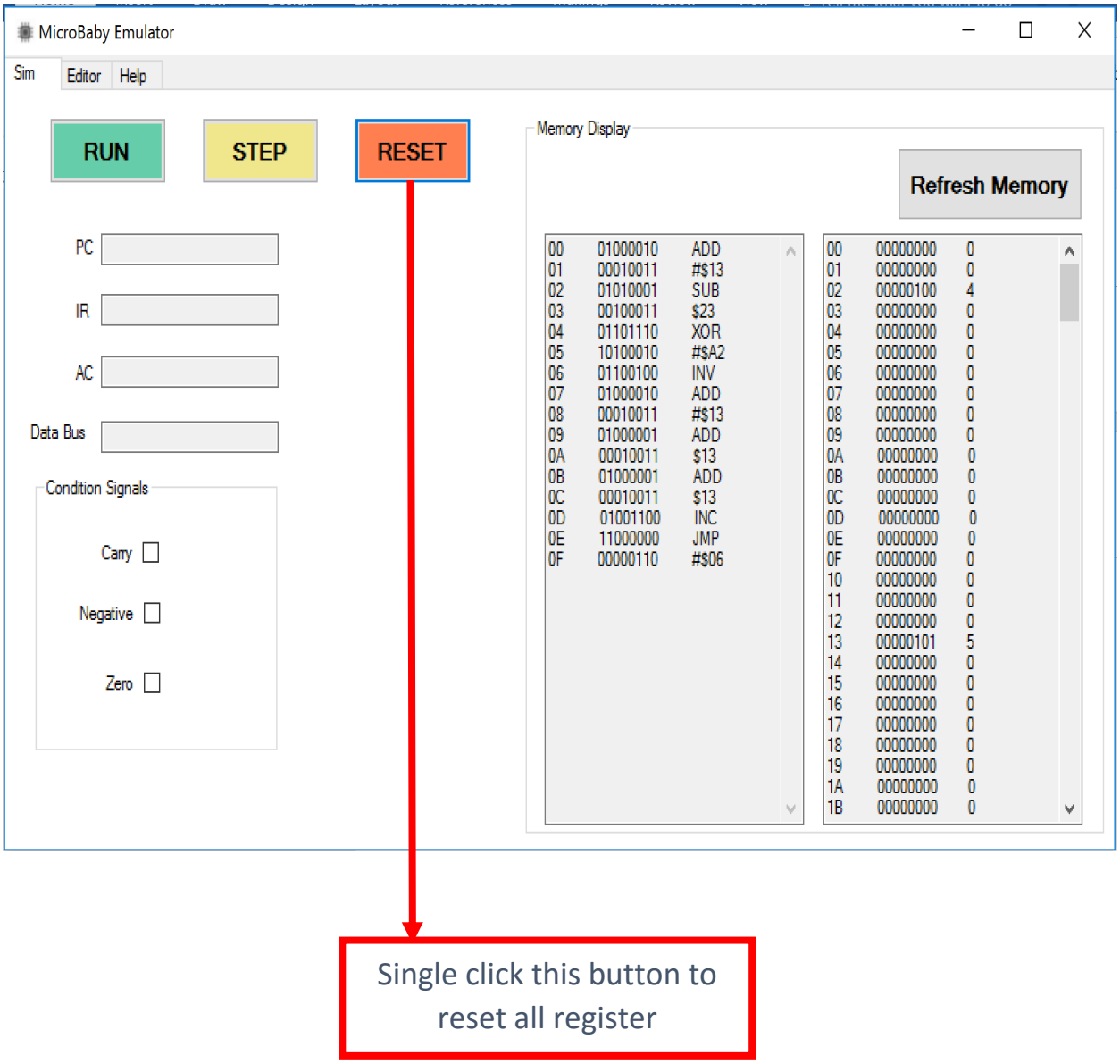
Step 2: Load assembly code to the converter



Additional tips: You can easily adjust the memory content in any address using Update memory button.

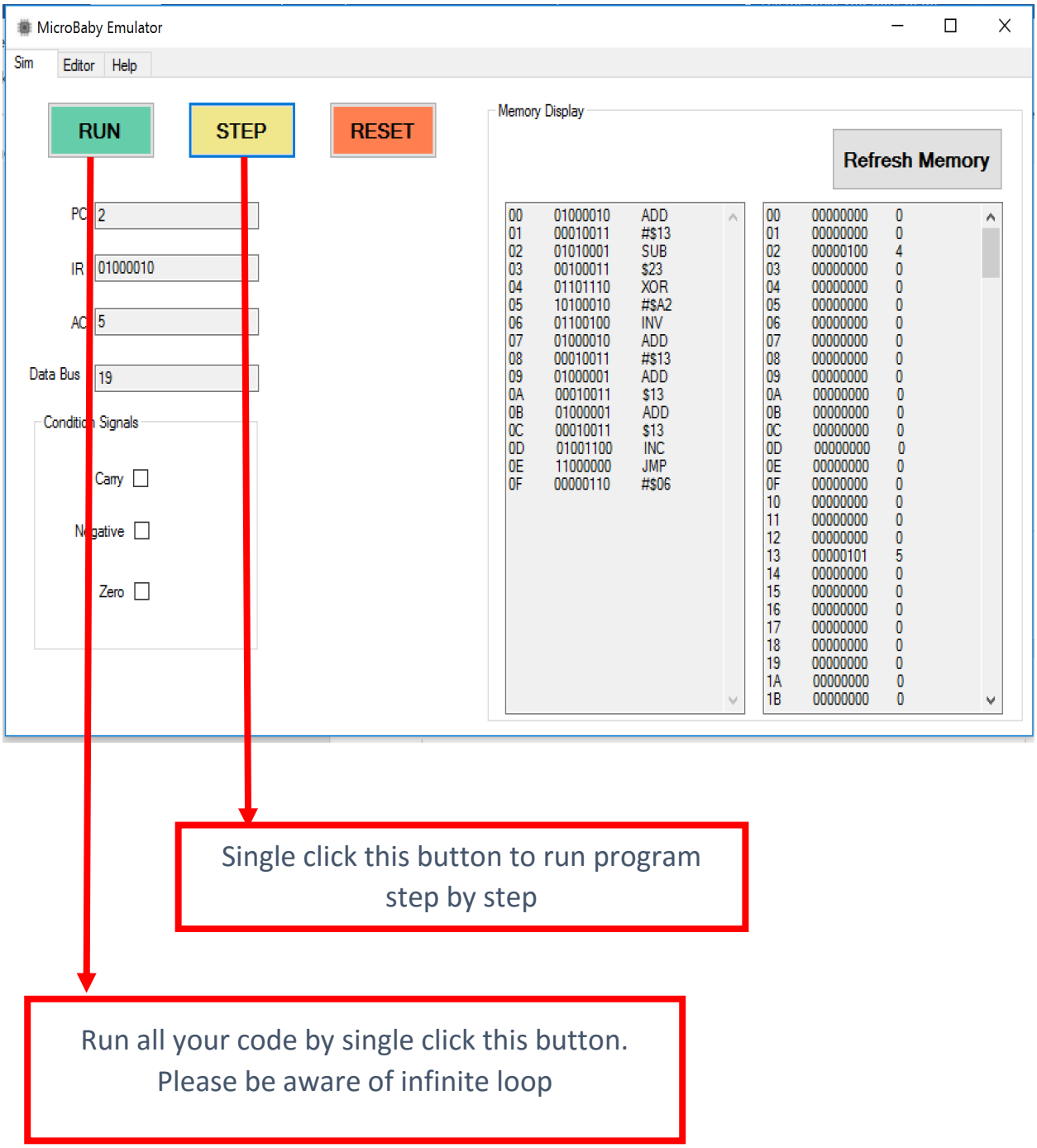


Step 3: Reset the all memory and register value before running the program



Single click this button to reset all register

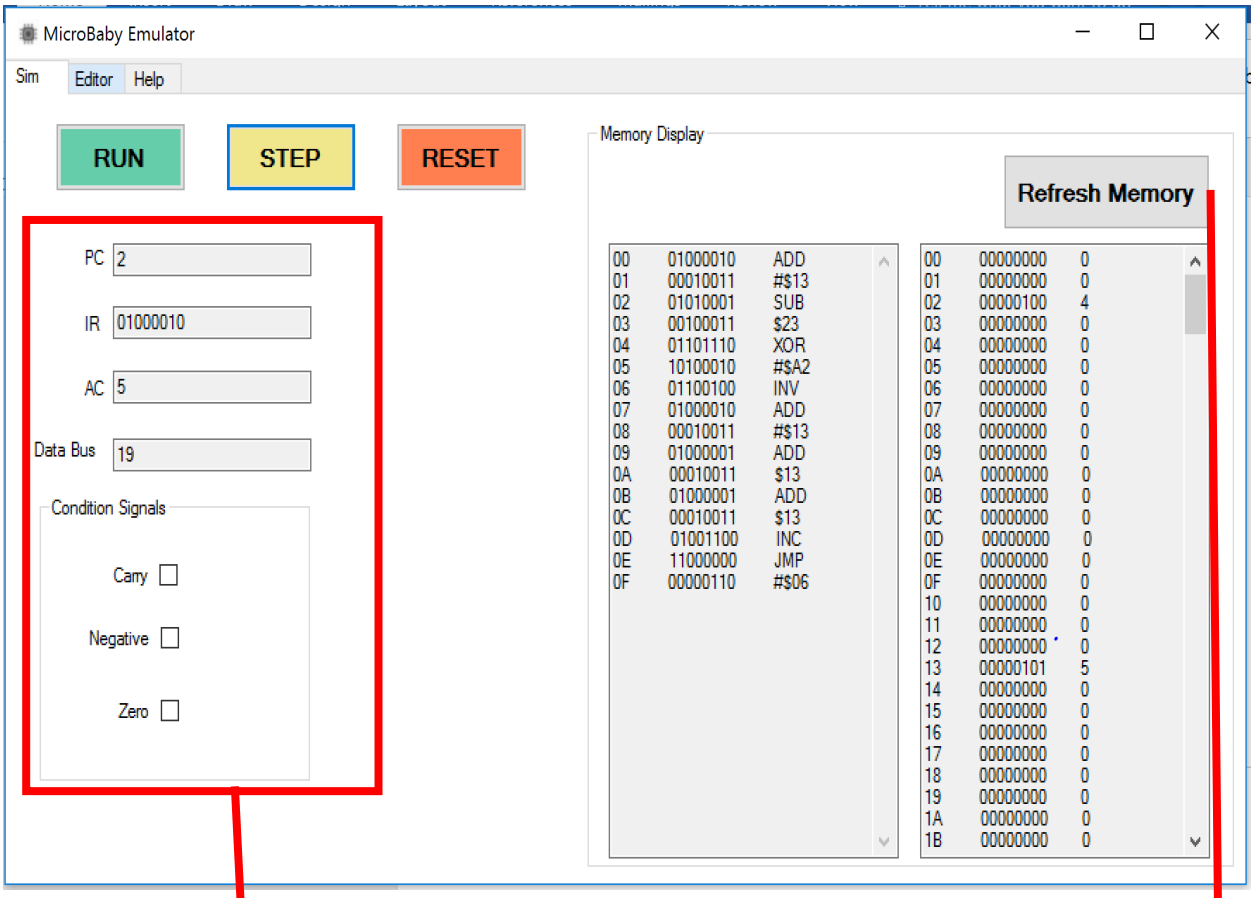
Step 4: Two ways to run simulation and fetch the results



Single click this button to run program step by step

Run all your code by single click this button. Please be aware of infinite loop

Step 5: Examine the results



The result of register and condition signals can be examined here

Refresh to get newest content of memory