

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

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

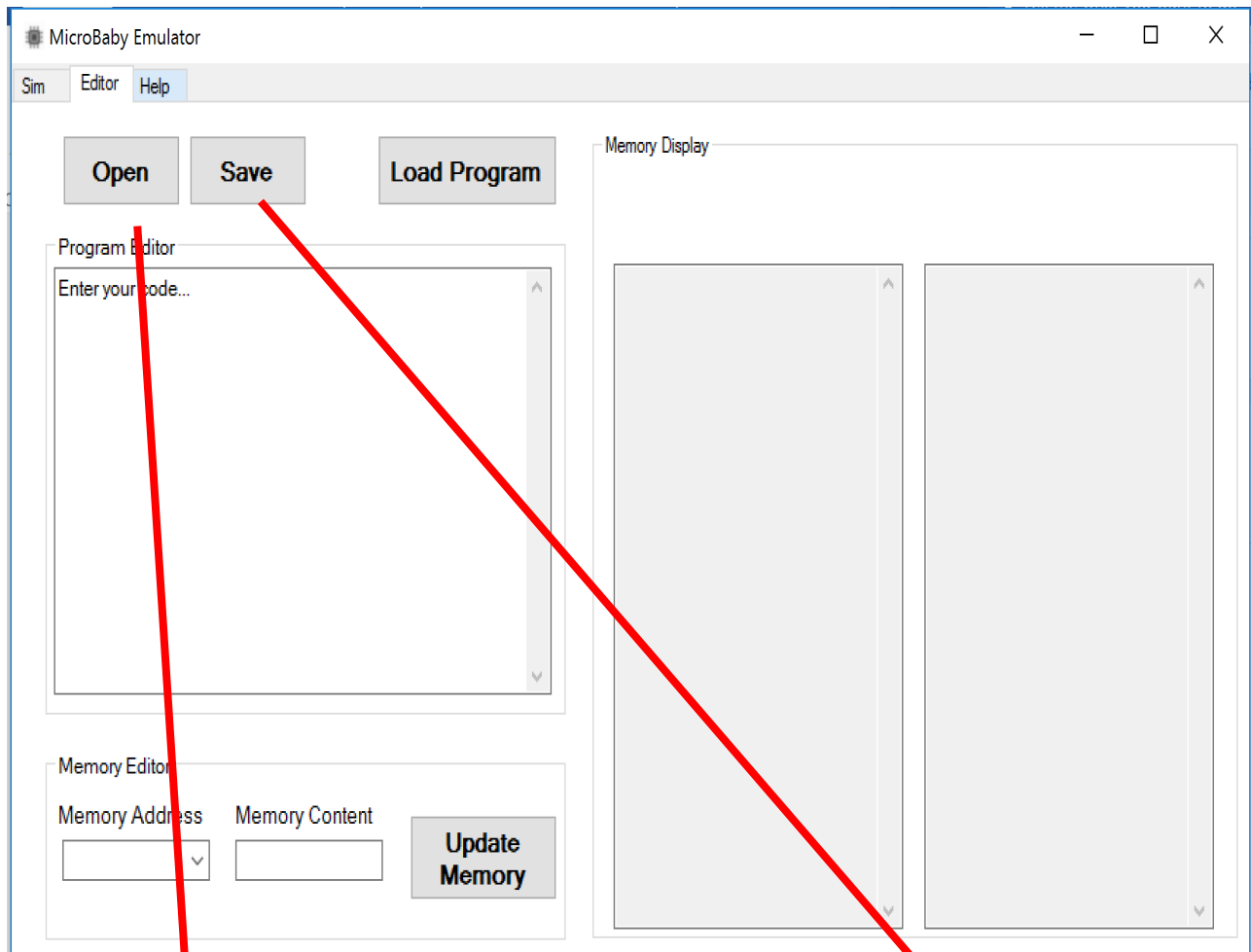
Load Program: Running the program based on the assembly code you have entered in the program editor.

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:

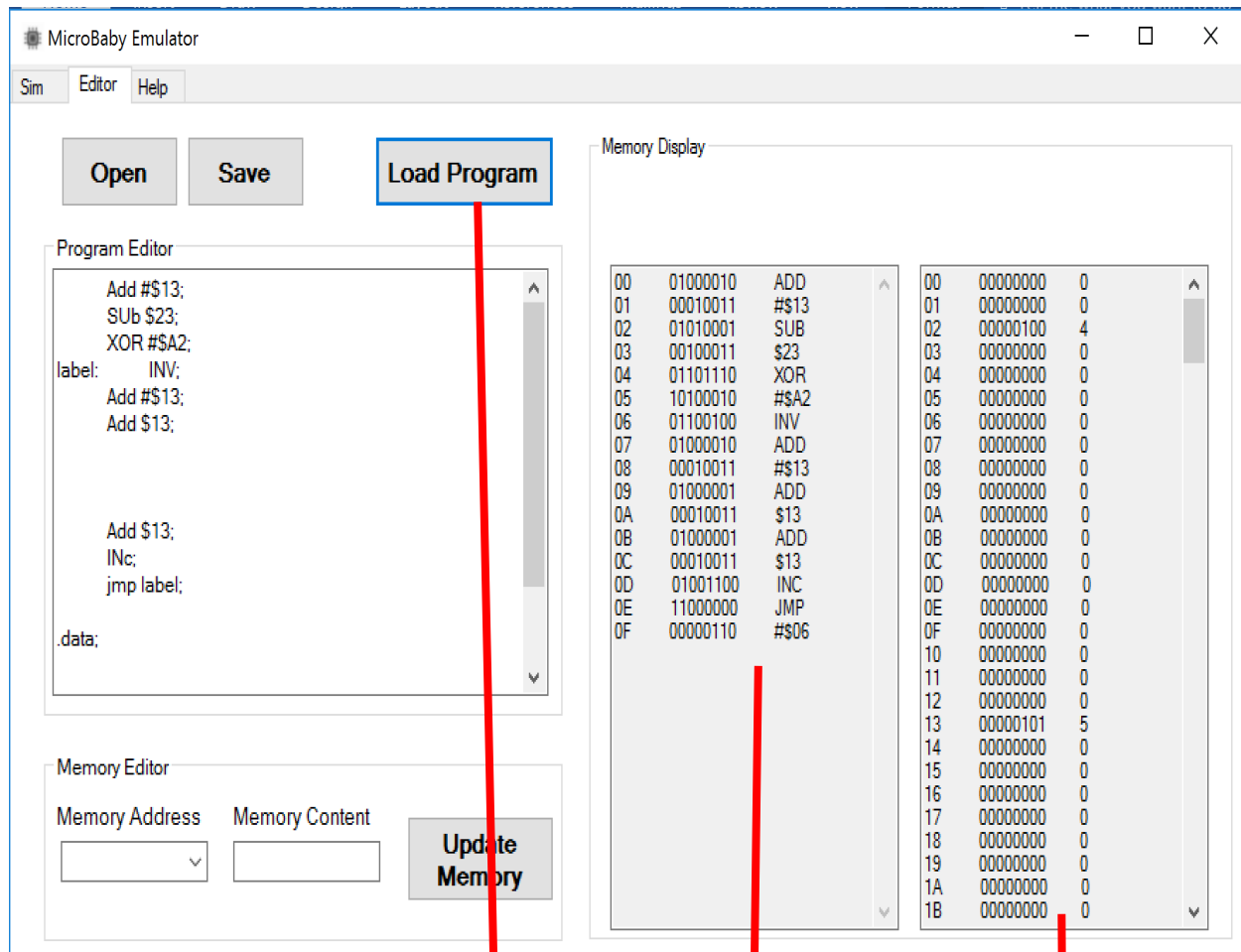
### Step 1: Write or open your assembly code



Enter the assembly code here

Load and save you assembly  
code in .txt format

## Step 2: Load assembly code to the converter

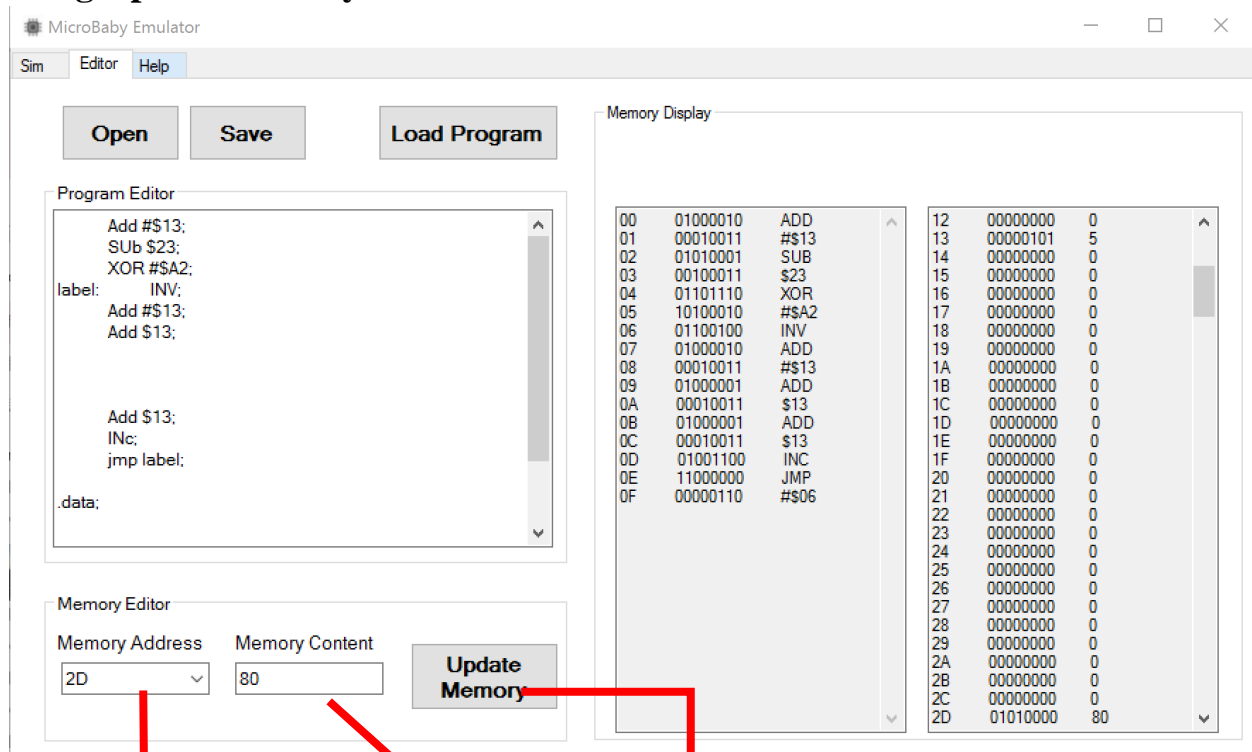


Single Click this  
button to load

Memory display

Instruction  
Opcode

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

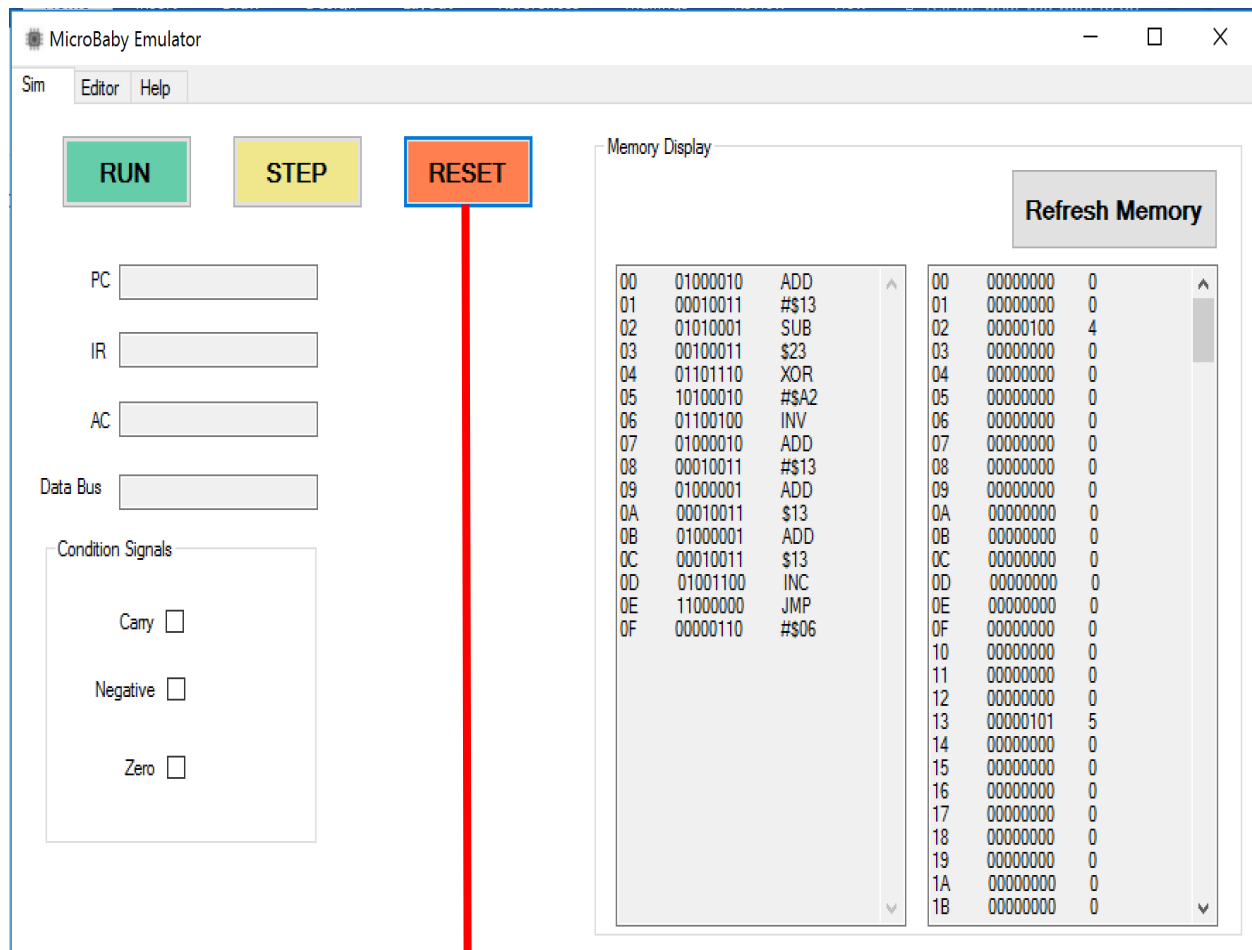


Select memory address

Input value

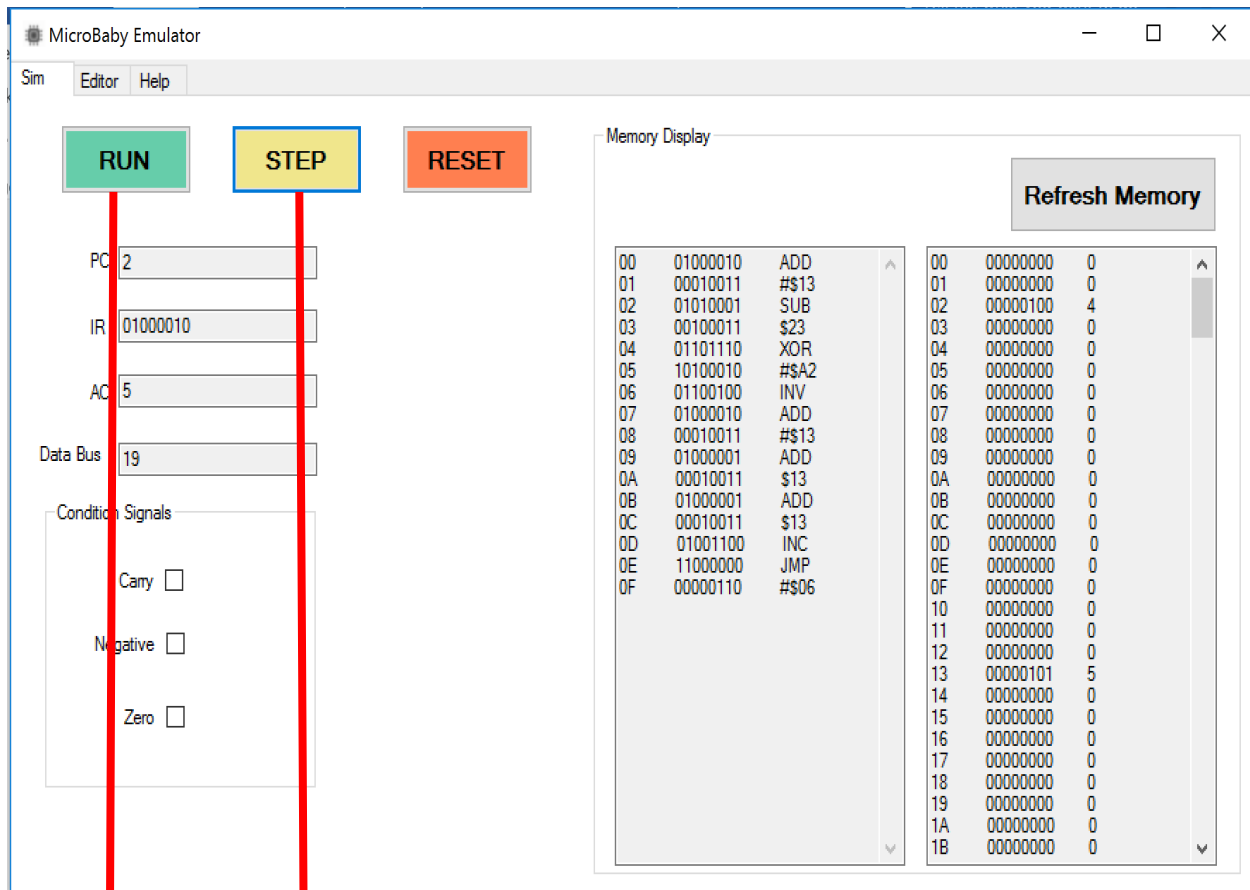
Click this button to redeem new memory content

### 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 MicroBaby Emulator interface is shown with the following components:

- Buttons:** RUN (green), STEP (yellow), RESET (orange).
- Registers:**
  - PC: 2
  - IR: 01000010
  - AC: 5
  - Data Bus: 19
- Condition Signals:**
  - Carry: ☐
  - Negative: ☐
  - Zero: ☐
- Memory Display:**
  - Left pane (addresses 00-0F):

Address	Binary	Operation
00	01000010	ADD
01	00010011	#\$13
02	01010001	SUB
03	00100011	\$23
04	01101110	XOR
05	10100010	#\$A2
06	01100100	INV
07	01000010	ADD
08	00010011	#\$13
09	01000001	ADD
0A	00010011	\$13
0B	01000001	ADD
0C	00010011	\$13
0D	01001100	INC
0E	11000000	JMP
0F	00000110	#\$06
  - Right pane (addresses 00-1B):

Address	Binary	Value
00	00000000	0
01	00000000	0
02	00000100	4
03	00000000	0
04	00000000	0
05	00000000	0
06	00000000	0
07	00000000	0
08	00000000	0
09	00000000	0
0A	00000000	0
0B	00000000	0
0C	00000000	0
0D	00000000	0
0E	00000000	0
0F	00000000	0
10	00000000	0
11	00000000	0
12	00000000	0
13	00000101	5
14	00000000	0
15	00000000	0
16	00000000	0
17	00000000	0
18	00000000	0
19	00000000	0
1A	00000000	0
1B	00000000	0
- Refresh Memory:** A button located above the right memory pane.

The result of register and condition signals can be examined here

Refresh to get newest content of memory