

Practical 02

CPU, Memory and I/O working together

In this practical, you will use a simulator to run the code, observe and appreciate how CPU, Memory and I/O work together.

- A. LMC Simulator
- B. Input data, move the data from accumulator to memory
- C. Input data1, data2 and add two numbers
- D. Input data1, data2 and ADD two numbers, REPEAT infinitely.
- E. Input data1, data2 and ADD two numbers, REPEAT 3 times.
- F. Input data1, data2 and subtract two numbers
- G. Countdown

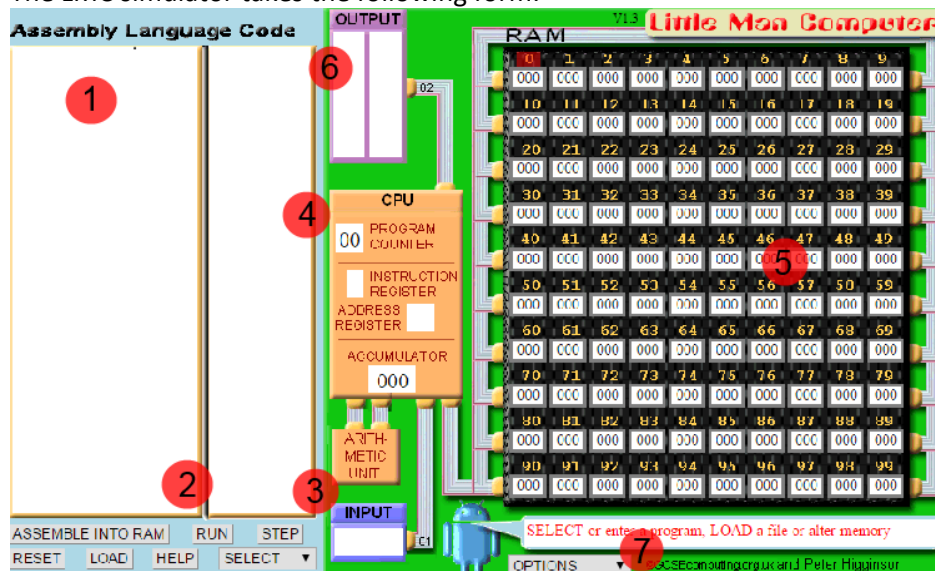
A. LMC Simulator

This practical is based on the excellent LMC simulator provided by Peter Higginson.
(<https://peterhigginson.co.uk/LMC/>)

Little Man Computer Simulation (LMC) in **JavaScript** with Fetch/Execute

Online help for the LMC, go to <https://peterhigginson.co.uk/LMC/help.html>

The LMC simulator takes the following form:



These are main components in the window that are easily recognizable:

1. The window for typing in the code
2. The two buttons - to load the code into memory and then run
3. The window for an input, if any - not necessary
4. An indicator that shows the progress of the code - step by step

5. Memory locations where instructions and data are stored, as specified in von Neumann architecture - 100 cells, from 00 to 99.
6. The window for the output/s during the execution of the code
7. Options for controlling the flow of the execution - slow to fast, etc

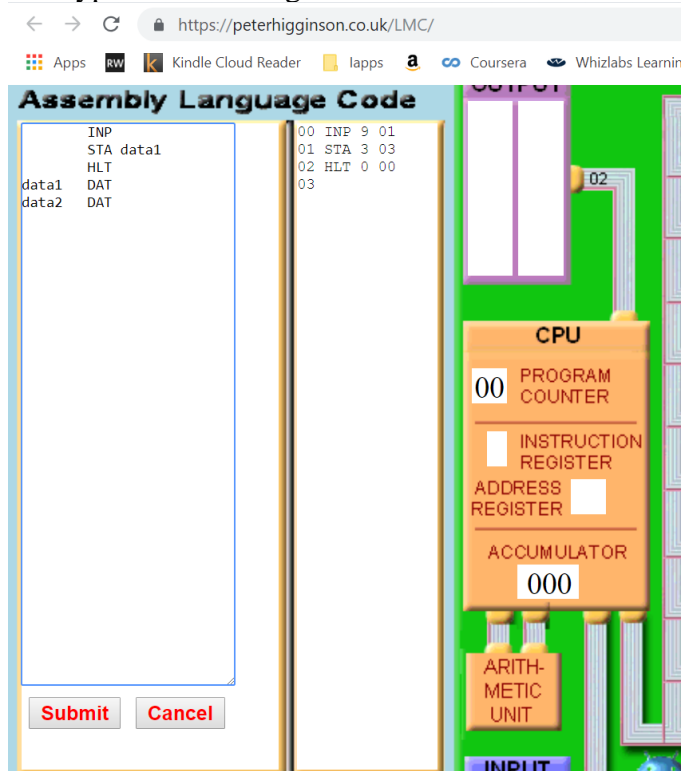
The best way to learn the LMC is running a set of codes, from the simplest to the more advanced gradually, rather than making an effort to understand the simulator fully at first.

You have to be familiar with the set of instructions. There are not many, just 11 of them. They are as follows:

Mnemonic Code	Numeric Code	Instruction
INP	901	Input data
ADD	1XX	Add the contents of the memory address to the Accumulator
SUB	2XX	Subtract the contents of the memory address from the Accumulator
STA	3XX	Store the value in the Accumulator in the memory address given.
LDA	5XX	Load the Accumulator with the contents of the memory address given
BRA	6XX	Branch - use the address given as the address of the next instruction
BRZ	7XX	Branch to the address given if the Accumulator is zero
BRP	8XX	Branch to the address given if the Accumulator is zero or positive
INP	901	Input data
OUT	902	Output data
HLT	HLT	Stop (Little Man has a rest).
DAT		Used to indicate a location that contains data.

B. Input data, move the data from accumulator to memory

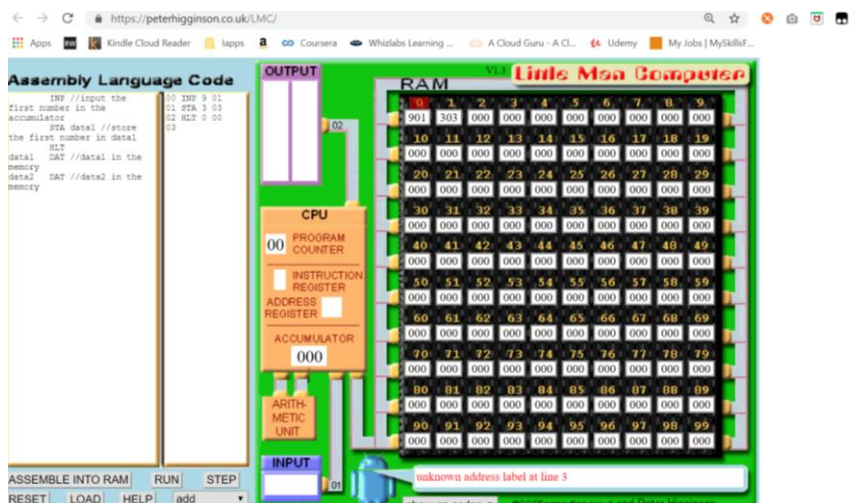
1. Type the following code in the textbox:



```
INP          //input the first number in the accumulator
STA data1    //store the first number in data1

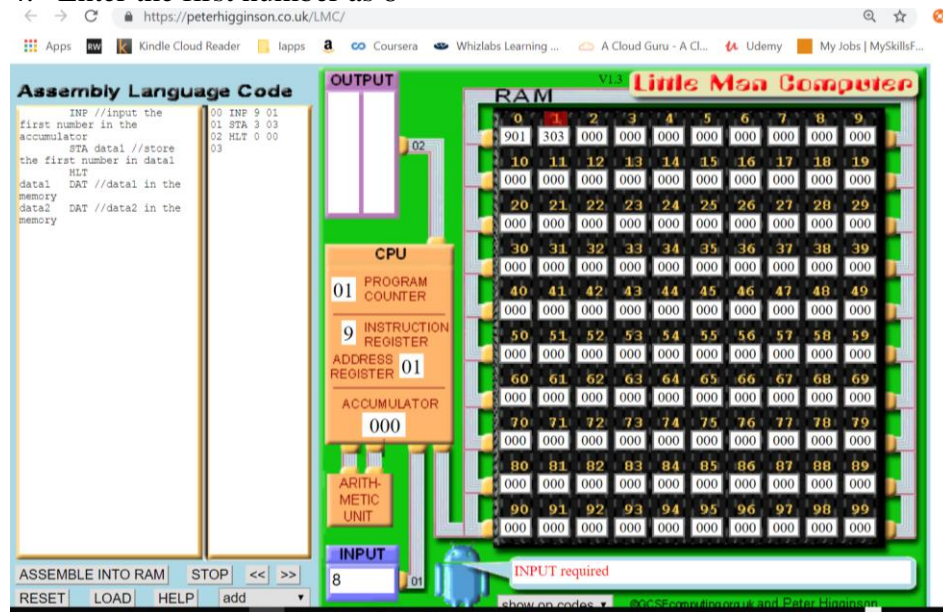
HLT
data1 DAT    //data1 in the memory
data2 DAT    //data2 in the memory
```

2. Observe the numeric code generated and loaded into memory(**click submit button**).



3. Run the code by clicking Step. observe the **red color** is for memory address, and **blue** is for data, or code.

4. Enter the first number as 8



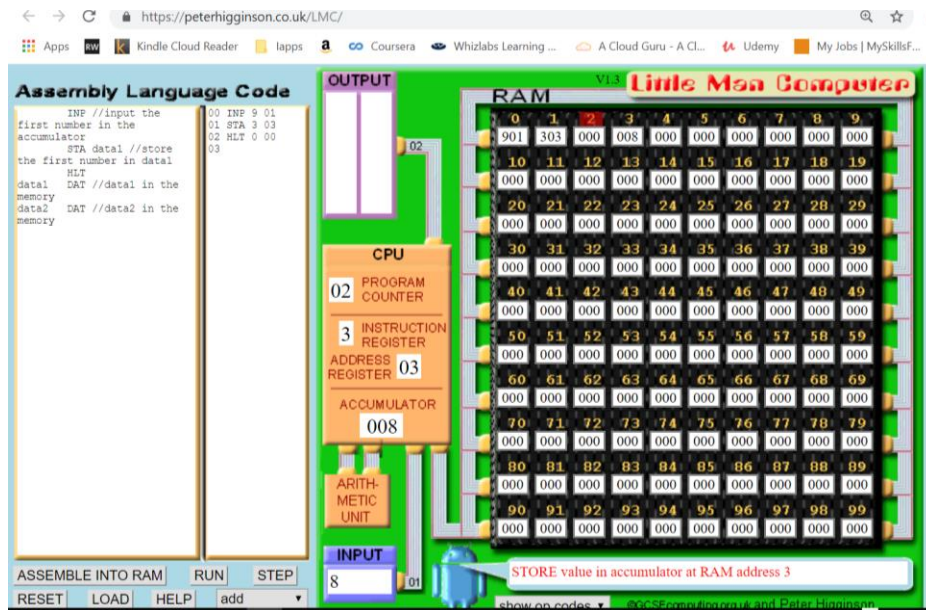
5. After entering the number, what is the value in the accumulator:

Value in the ACCUMULATOR(in decimal)

6. What is the next instruction to be executed by looking at the value of Program Counter?

Value in the Program counter

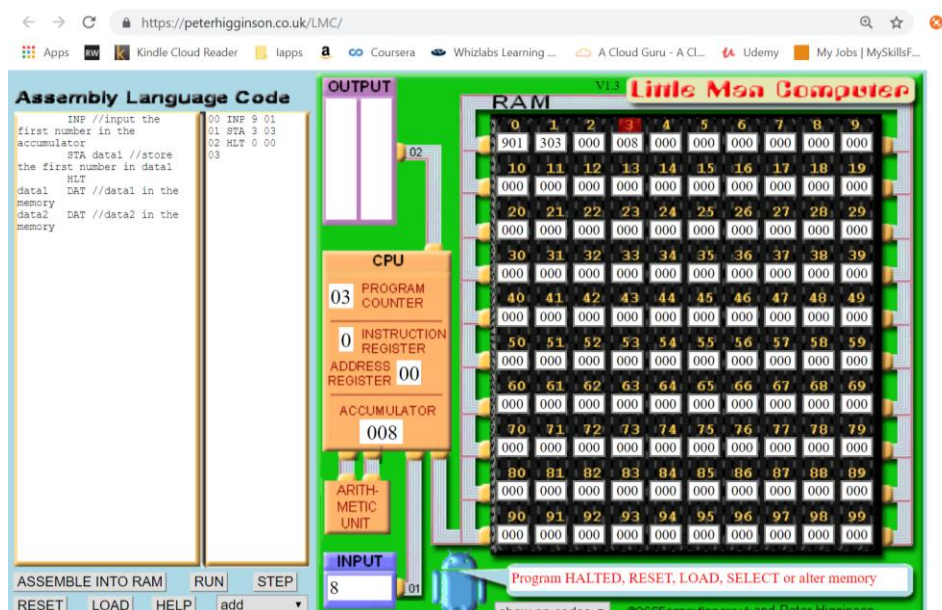
7. Click Step to execute STA (303), to move the first number from accumulator to memory.



Where is the first number stored in the memory?

Memory address for first number	
Value stored in the above memory address	

8. Step again to execute HALT



9. Observe the memory, filling the following table:

Assembly Code	Numeric Code	Address
INP	901	
STA	303	
HLT	000	
data	value	address
Data1	8	

10. Add more code to input data2

```

INP          //input the first number in the accumulator
STA data1    //store the first number in data1

INP          //input the second number in the accumulator
STA data2    //store the second number in data2

OUT          //output the result
HLT

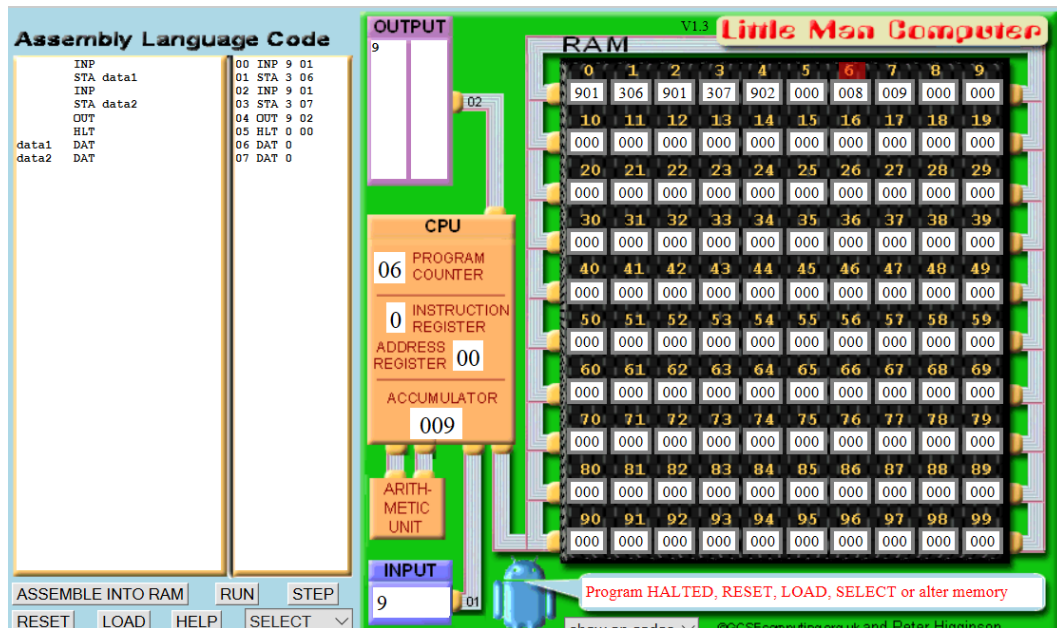
data1 DAT    //data1 in the memory
data2 DAT    //data2 in the memory

```

Run the code.

Enter first number as 8, and second number as 9

11. Observe the memory, filling the following table:



Assembly Code	Numeric Code	Address
INP	901	
STA	306, 307	
HLT	000	
data	value	address
data1	8	
data2	9	

C. Input data1, data2 and add two numbers

- Here is the code for adding two numbers and displaying the sum


```

INP          //input the first number in the accumulator
STA data1    //store the first number in data1

INP          //input the second number in the accumulator
STA data2    //store the second number in data2

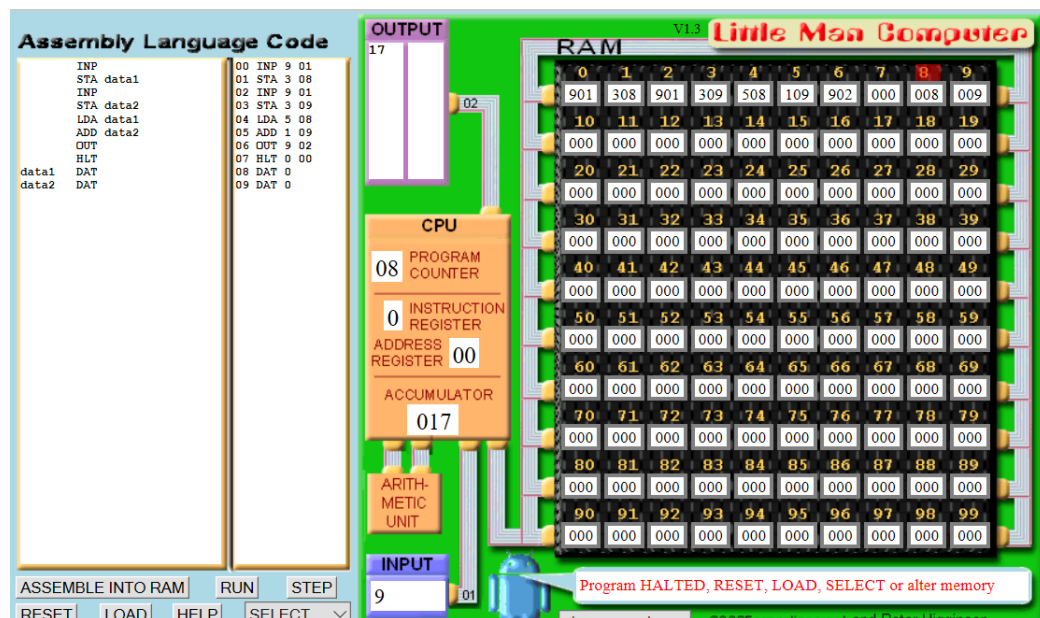
LDA data1    //load the first number in data1 to accumulator
ADD data2    //add the number with data2

OUT          //output the result
HLT

data1 DAT    //data1 in the memory
data2 DAT    //data2 in the memory

```

2. Run the code to test if it works.
3. Observe the memory, register, and output at the end of the execution.



Assembly Code	Numeric Code	Address
INP	901	
STA	308	
INP	901	
STA	309	
LDA data1	508	
ADD data2	109	
OUT	902	
HLT	000	
data	value	Address
data1	8	
data2	9	

4. Initialize data1 with value 10, and data2 with value 11

```

INP          //input the first number in the accumulator
STA data1    //store the first number in data1

INP          //input the second number in the accumulator
STA data2    //store the second number in data2

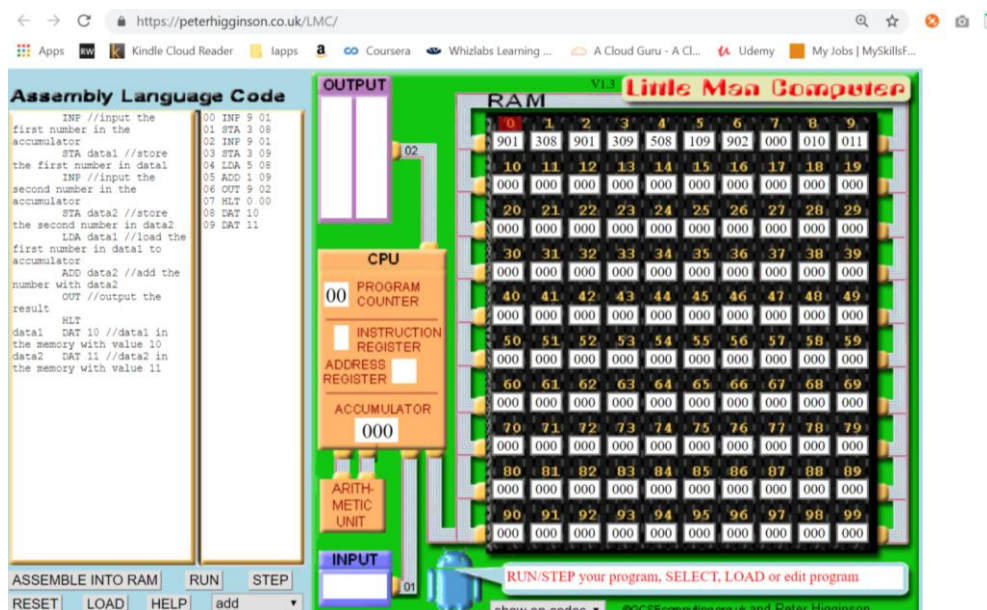
LDA data1    //load the first number in data1 to accumulator
ADD data2    //add the number with data2

OUT          //output the result
HLT

data1 DAT 10 //data1 in the memory with value 10
data2 DAT 11 //data2 in the memory with value 11

```

Submit the code, observe the memory locations for the two variables with initialized value "10", "11"



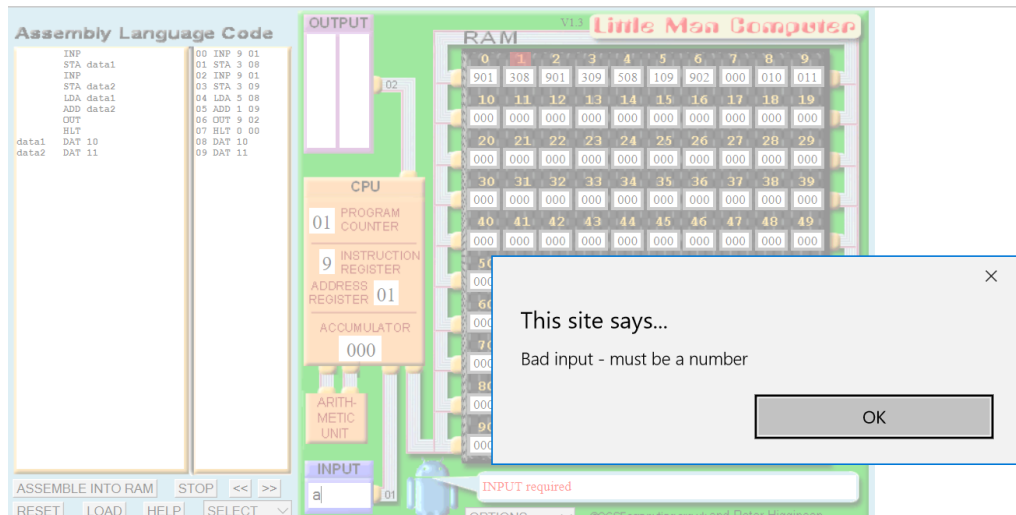
At the end of execution, what are the values in the above two addresses for data1 and data2?

(assuming user input for data1 is "8", and data2 is "9")



data	value
data1	
data2	

5. What if you enter the first number as character "a"?



6. Test and run the program to find the MAXIMUM number which is VALID.

Maximum input value	
---------------------	--

D. Input data1, data2 and ADD two numbers, REPEAT infinitely.

1. Key in the code in simulator

START INP	//input the first number in the accumulator
STA data1	//store the first number in data1
INP	//input the second number in the accumulator
STA data2	//store the second number in data2
LDA data1	//load the first number in data1 to accumulator
ADD data2	//add the number with data2
OUT	//output the result
BRA START	
HLT	
data1 DAT 10	//data1 in the memory with value 10
data2 DAT 11	//data2 in the memory with value 11

2. Test the above code.

E. Input data1, data2 and ADD two numbers, REPEAT 3 times.

```

START INP          //input the first number in the accumulator
      STA data1     //store the first number in data1

      INP           //input the second number in the accumulator
      STA data2     //store the second number in data2

      LDA data1     //load the first number in data1 to accumulator
      ADD data2     //add the number with data2

      OUT          //output the result

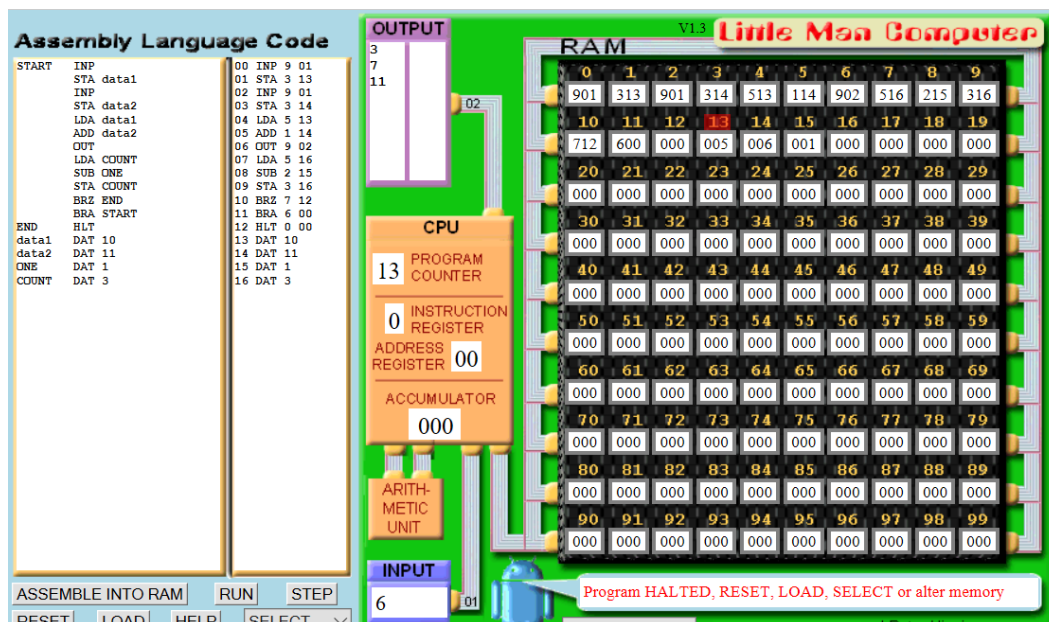
      LDA COUNT     //load the COUNT
      SUB ONE       //Subtract 1
      STA COUNT     //store the result to COUNT
      BRZ END       //if value is zero, branch to END

      BRA START

END    HLT
data1  DAT 10     //data1 in the memory with value 10
data2  DAT 11     //data2 in the memory with value 11
ONE    DAT 1
COUNT DAT 3

```

1. Run and test above code.
2. After loop for three times, the screen shot as follows:



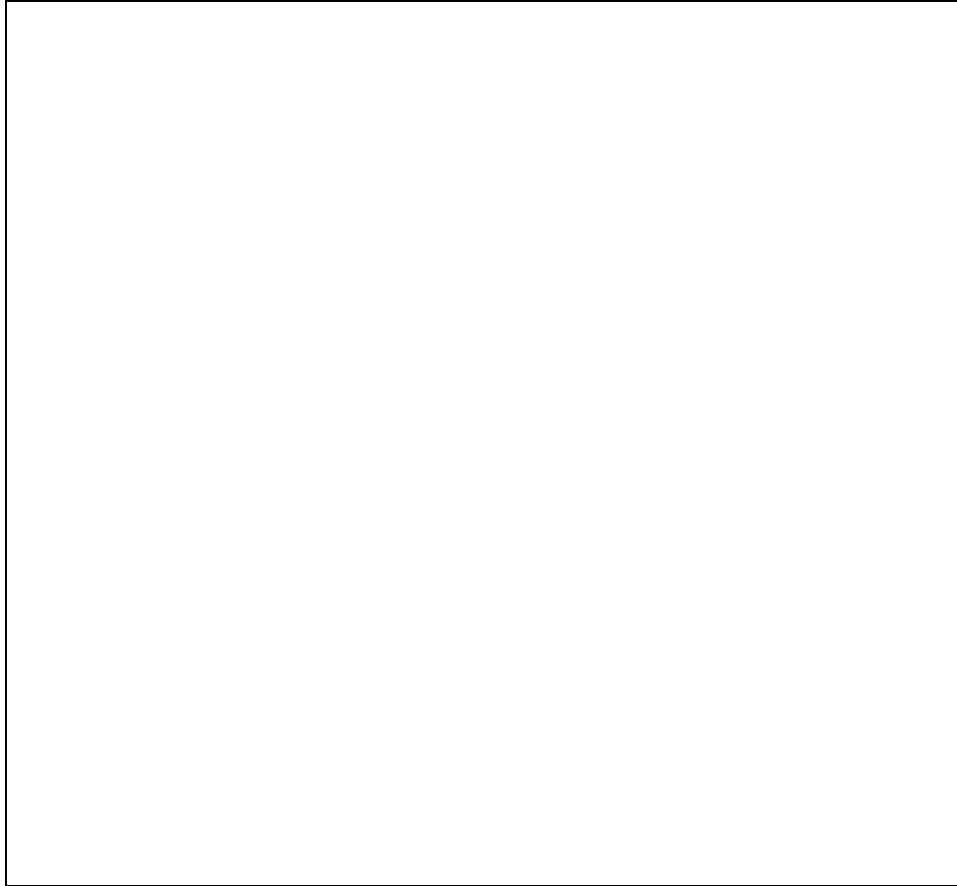
data1 address	
data2 address	
COUNT address	
COUNT value in the end	

In the above code, we use SUB ONE. Change to SUB 1 and test the above program to see if it works.

In your view, which approach is better?

**F. Input data1, data2 and subtract two numbers
(SELF DIRECTED LEARNING)**

My code is shown below:

A large, empty rectangular box with a thin black border, intended for the user to display their code.

G. Countdown (SELF DIRECTED LEARNING)

Write the code to create a countdown and display the following items shown in the OUTPUT window. You should allow the user to input a number between three and seven.

Hint: you can use BRZ (based on value in ACC) and BRA instructions for iteration.

The program ends when the accumulator value is 0



My code is shown below:

Reference:

- (1) https://en.wikipedia.org/wiki/Little_man_computer
- (2) Little man explain on youtube: <https://www.youtube.com/watch?v=kCyyZl1GgsQ>