First generation

The first generation program language is pure machine code, that is just ones and zeros, e.g. 00100100101011111010110.

What are the advantages and disadvantages?

First generation

+ Code can be fast and efficient

+ Code can make use of specific processor features such as special registers

- Code cannot be ported to other systems and has to be rewritten

- Code is difficult to edit and update

Second generation programming

Second-generation programming languages are a way of describing Assembly code which you may have already met.

Assembly Code		Object Code
LDA A ADD #5 STA A JMP #3	-> Assembler ->	000100110100 001000000101 001100110100 010000000011

What are the advantages and disadvantages?

Second generation programming

+ Code can be fast and efficient

+ Code can make use of specific processor features such as special registers

+ As it is closer to plain English, it is easier to read and write when compared to machine code

- Code cannot be ported to other systems and has to be rewritten

Third generation (High Level Languages)

Third generation (High Level Languages) codes are imperative. Imperative means that code is executed line by line, in sequence. For example:

```
1 dim x as integer
2 x = 3
3 dim y as integer
4 y = 5
5 x = x + y
6 console.writeline(x)
```

What are the advantages and disadvantages?

Third generation (High Level Languages)

+ Hardware independence, can be easily ported to other systems and processors

+ Time saving programmer friendly, one line of 3rd gen is the equivalent of many lines of 1st and 2nd gen

- Code produced might not make the best use of processor specific features unlike 1st and 2nd gen

Fourth generation

What are the advantages and disadvantages?

Fourth generation

Fourth-generation languages are designed to reduce programming effort and the time it takes to develop software, resulting in a reduction in the cost of software development.

Declarative languages - describe what computation should be performed and not how to perform it. Not imperative!

Little Man Computer

Learning objective:

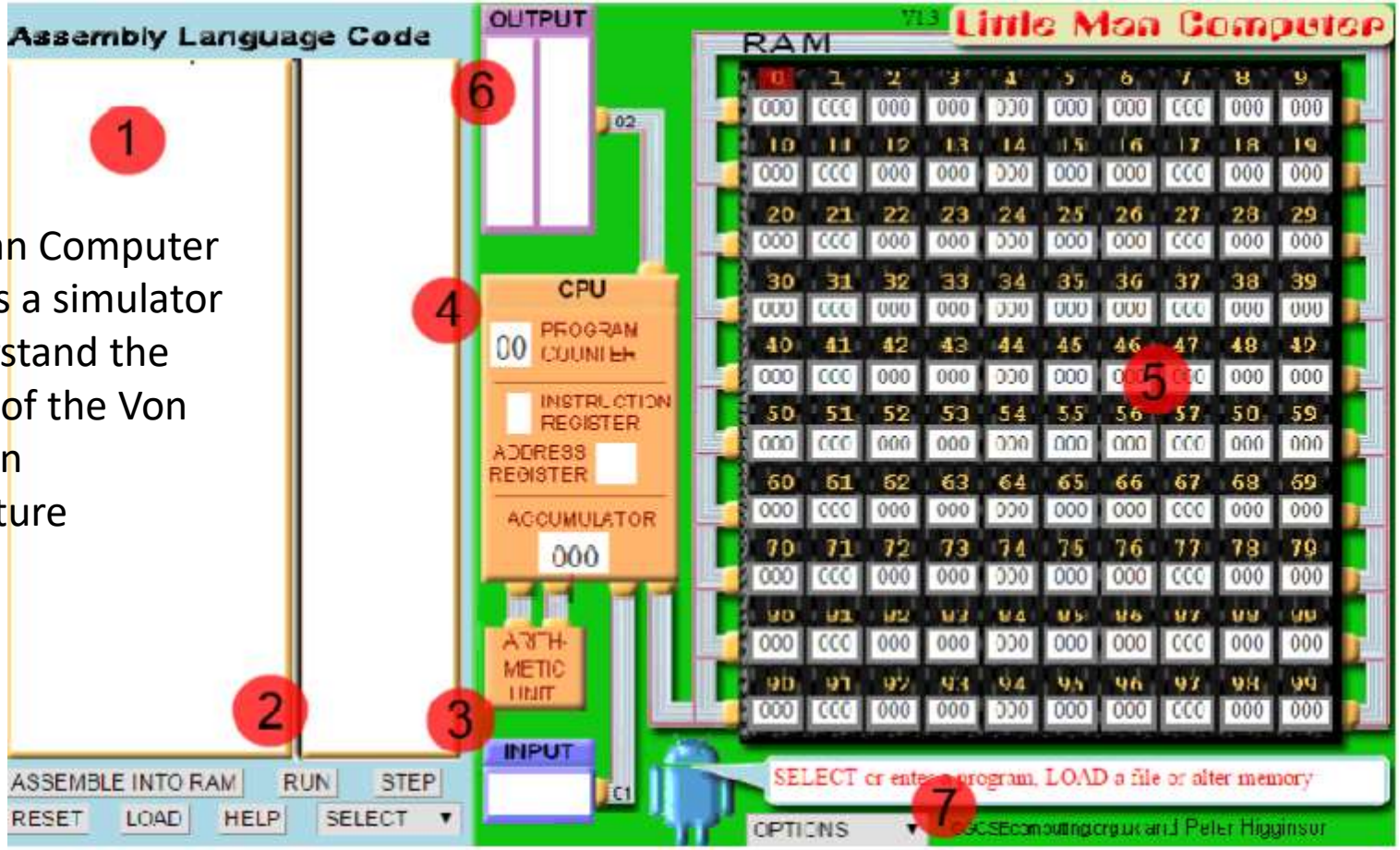
Analyze a simple program written in the language of assembler

Assessment Criteria

- Know the use of LMC commands
- Understand the use of assembly language

- Distinguish the difference between assembly language and others
- Able to use LMC commands to solve problems

Little Man Computer - LMC - is a simulator to understand the working of the Von Neumann Architecture

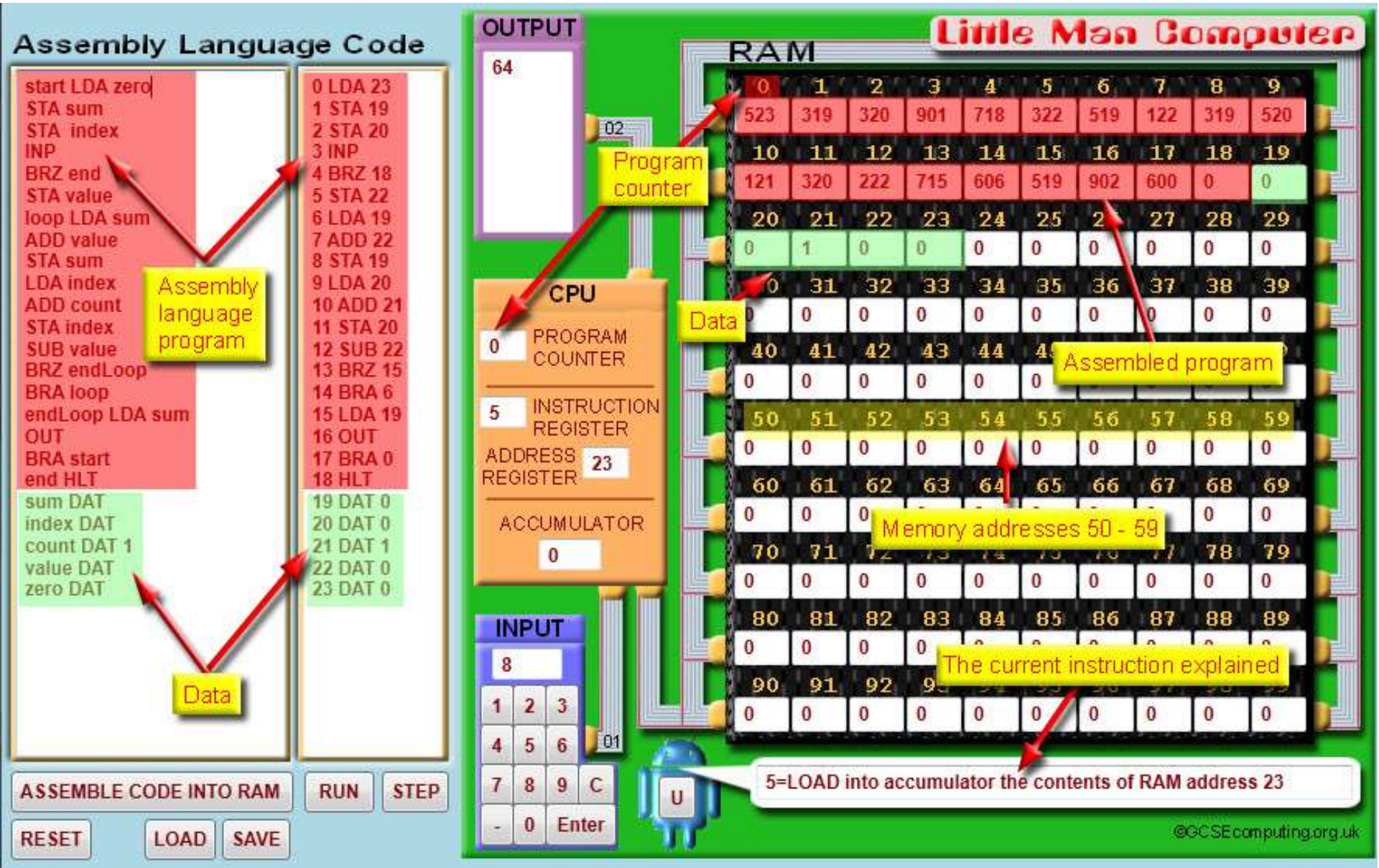


These are main components in the window that are easily recognizable:

http://peterhigginson.co.uk/LMC/

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 Little Man Computer (LMC)



Mnemonic (assembly language)	Numeric (machine code)	Description
INP	901	INPUT
STA FIRST	308	STORE VALUE(FIRST) IN POSITION 8
INP	901	INPUT
STA SECOND	309	STORE VALUE(SECOND) IN POSITION 9
LDA FIRST	508	LOAD VALUE FROM POSITION 8
ADD SECOND	109	ADD VALUE FROM POSITION 9
OUT	902	OUTPUT ANSWER OF SUBTRACTION
HLT	0	HALT (STOP)
FIRST DAT	0	POSITION OF FIRST ITEM OF DATA
SECOND DAT	0	POSITION OF SECOND ITEM OF DATA

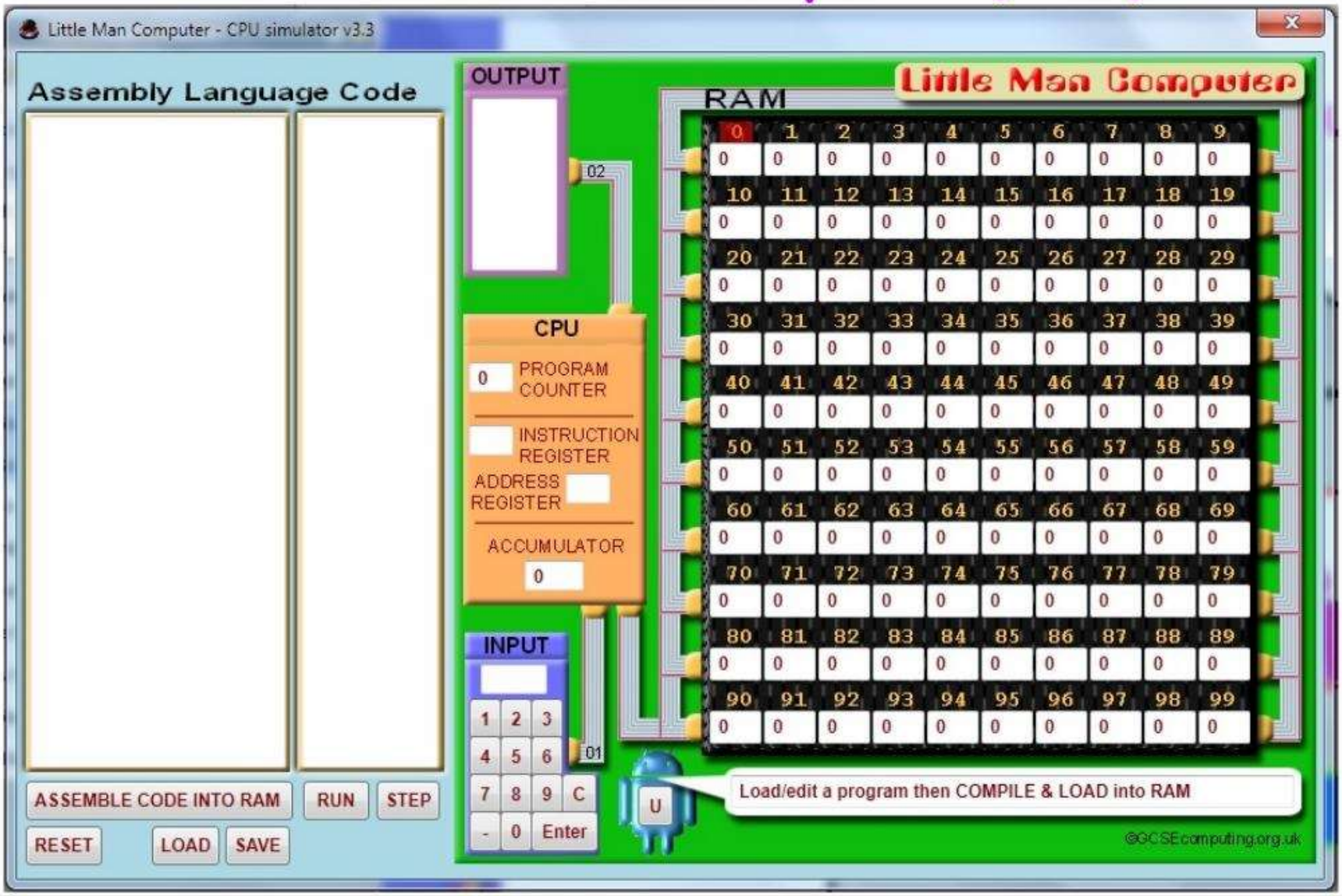
Numeric Code	Mnemonic Code	Instruction
1xx	ADD	ADD
2xx	SUB	SUBTRACT
3xx	STA	STORE
4xx	LDA	LOAD
5xx	BRA	BRANCH (unconditional)
6xx	BRZ	BRANCH IF ZERO (conditional)
7xx	BRP	BRANCH IF POSITIVE (conditional)
901	INP	INPUT
902	OUT	OUTPUT
000	HLT/COB	HALT/COFFEE BREAK
	DAT	DATA

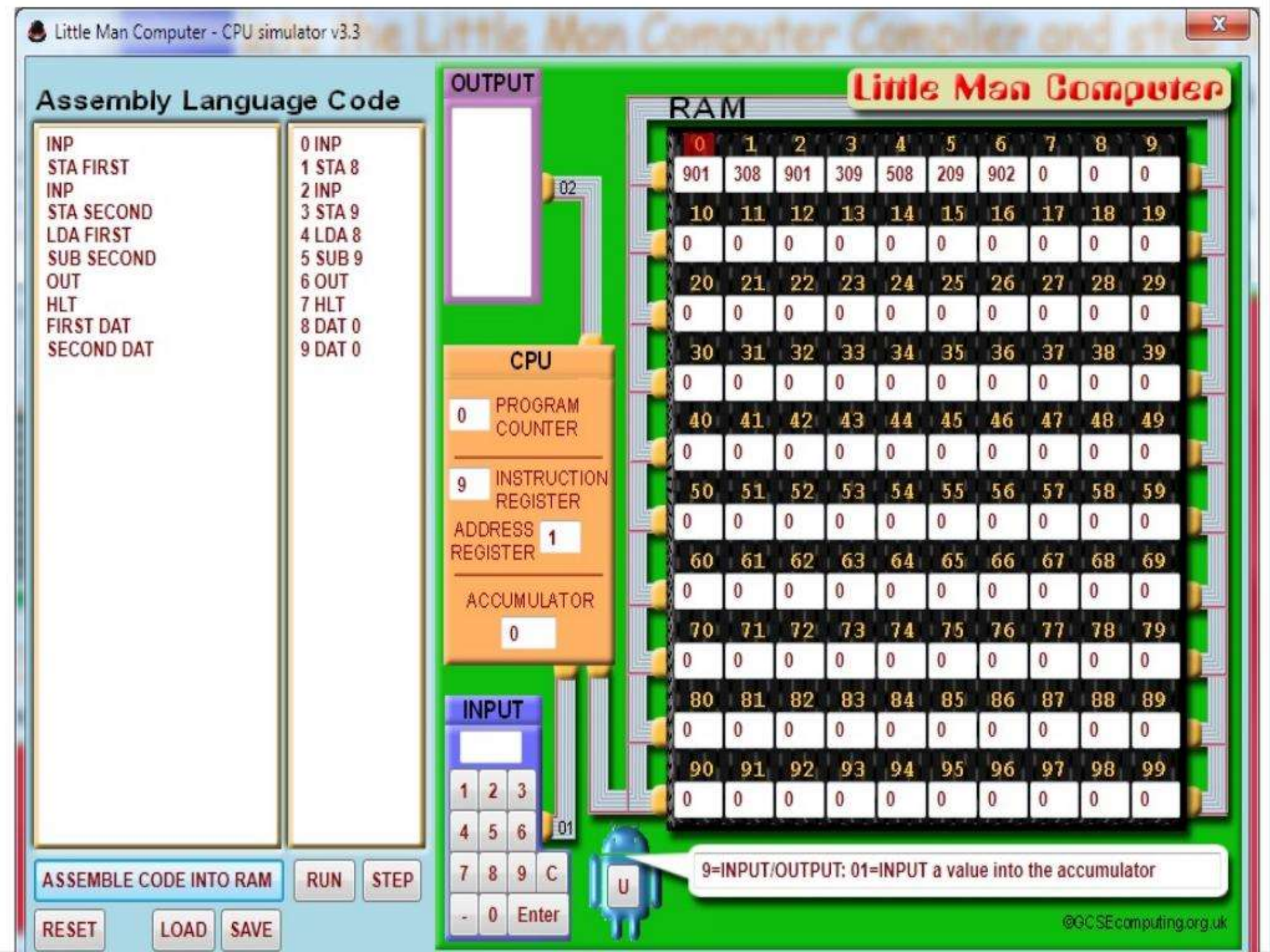
Example 308 means STORE to box 08

and ADD 9 means add the contents of box 9

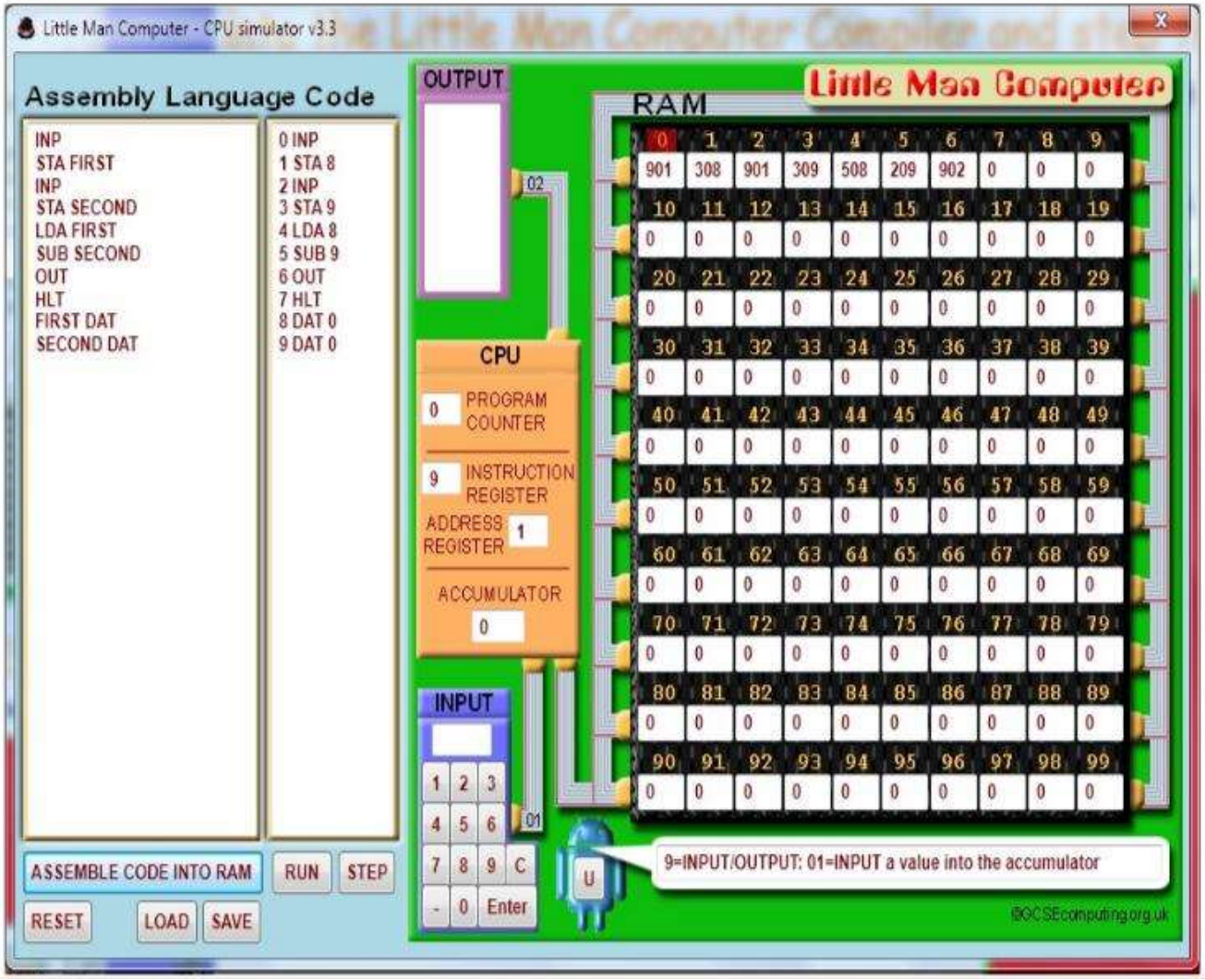
http://peterhigginson.co.uk/LMC/

The Little Man Computer (LMC)



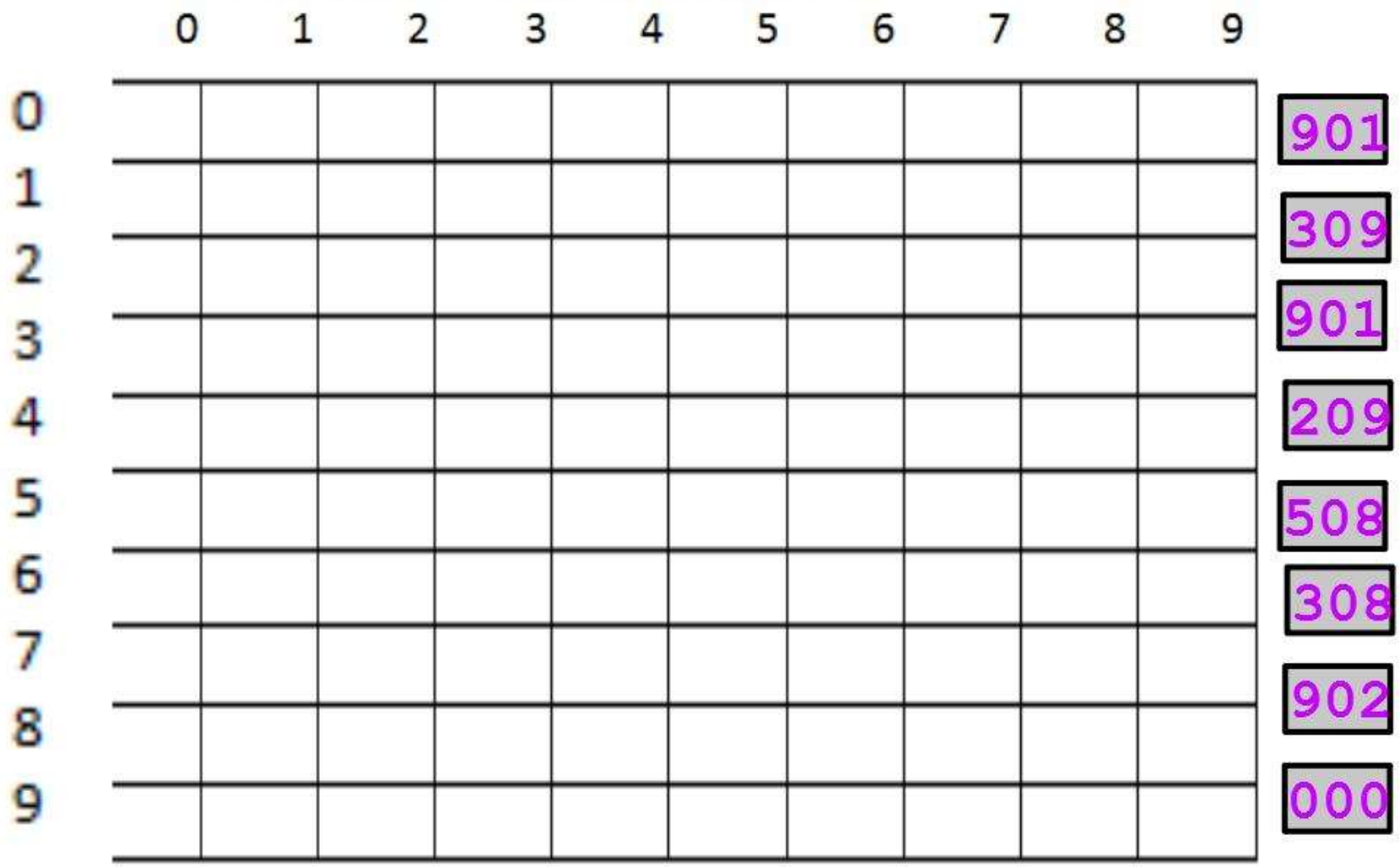

Little Man Computer - CPU simulator v3.3
Assembly Language Code
INP
STA FIRST
INP
STA SECOND
LDA FIRST
SUB SECOND
OUT
HLT
FIRST DAT
SECOND DAT
0 INP
1 STA 8
2 INP
3 STA 9
4 LDA 8
5 SUB 9
6 OUT
7 HLT
8 DAT 0
9 DAT 0
ASSEMBLE CODE INTO RAM
RUN
STEP
RESET
LOAD
SAVE
OUTPUT
02
CPU
0 PROGRAM COUNTER
9 INSTRUCTION REGISTER
ADDRESS REGISTER 1
ACCUMULATOR
0
INPUT
1 2 3
4 5 6
7 8 9 C
- 0 Enter
01
RAM
Little Man Computer
0 1 2 3 4 5 6 7 8 9
901 308 901 309 508 209 902 0 0 0
10 11 12 13 14 15 16 17 18 19
0 0 0 0 0 0 0 0 0 0
20 21 22 23 24 25 26 27 28 29
0 0 0 0 0 0 0 0 0 0
30 31 32 33 34 35 36 37 38 39
0 0 0 0 0 0 0 0 0 0
40 41 42 43 44 45 46 47 48 49
0 0 0 0 0 0 0 0 0 0
50 51 52 53 54 55 56 57 58 59
0 0 0 0 0 0 0 0 0 0
60 61 62 63 64 65 66 67 68 69
0 0 0 0 0 0 0 0 0 0
70 71 72 73 74 75 76 77 78 79
0 0 0 0 0 0 0 0 0 0
80 81 82 83 84 85 86 87 88 89
0 0 0 0 0 0 0 0 0 0
90 91 92 93 94 95 96 97 98 99
0 0 0 0 0 0 0 0 0 0
U
9=INPUT/OUTPUT: 01=INPUT a value into the accumulator
©GCSEcomputing.org.uk

In pairs or individually, Please enter the following program into the Little Man Computer Compiler and step through it, observing what happens.



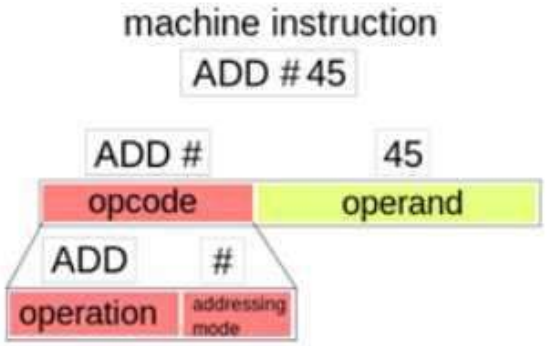
INP
STA FIRST
INP
STA SECOND
LDA FIRST
ADD SECOND
OUT
HLT
FIRST DAT
SECOND DAT

Little Man Computer



Mnemonic (assembly language)	Numeric (machine code)	Description
INP	901	INBOX --> ACCUMULATOR
STA FIRST	308	ACCUMULATOR --> MEMORY[08]
INP	901	INBOX --> ACCUMULATOR
STA SECOND	309	ACCUMULATOR --> MEMORY[09]
LDA FIRST	508	MEMORY[08] --> ACCUMULATOR
SUB SECOND	209	ACCUMULATOR = ACCUMULATOR - MEMORY[09]
OUT	902	ACCUMULATOR --> OUTBOX
HLT	000	HALT/COFFEE BREAK
FIRST DAT	000	FIRST ITEM OF DATA
SECOND DAT	000	SECOND ITEM OF DATA

In pairs, enter the following program into the Little Man Computer Compiler and step through it, writing a description of what happens.




```
INP
STA FIRST
INP
STA SECOND
LDA FIRST
ADD SECOND
OUT
HLT
FIRST DAT
SECOND DAT
```

For the following memory space, what would it look like after executing the assembly code below:

Address	Contents
211	6
212	3
213	78
214	21

```
LOAD #100
STORE 213
LOAD 214
ADD 213
STORE 214
```

When you load the LMC there is already a program in the computer. The program is written out in the table below in machine code. By executing the program and using the list of instructions, work out what the program does.

Address	Instruction	What it does:
00	901	
01	399	
02	901	
03	199	
04	902	
05	000	

Translate the instructions, mnemonic and numeric codes on the worksheets

Translate these instructions into their mnemonic and numeric codes

Instruction	Mnemonic code	Numeric code
INPUT		
STORE		
LOAD		
OUTPUT		
HALT		
DATA		

Translate these numeric codes into their mnemonic and instructions

Numeric code	Mnemonic code	Instruction
1##		
4##		
7##		
901		
902		
5##		

Translate these mnemonic codes into their instructions and numeric codes

Mnemonic code	Numeric code	Instruction
HLT		
OUT		
INP		
LDA		
BRA		

Write programs

A) A-B,

B) A+B-C,

C) A+(B-C)

D) (A-C)+(B-D)

Do it yourself

https://learningapps.org/watch?v=prtz8teoa20

https://www.bzfar.org/publ/algorithms_programming/programming_languages/programming_in_little_man_computer_lmc/42-1-0-47