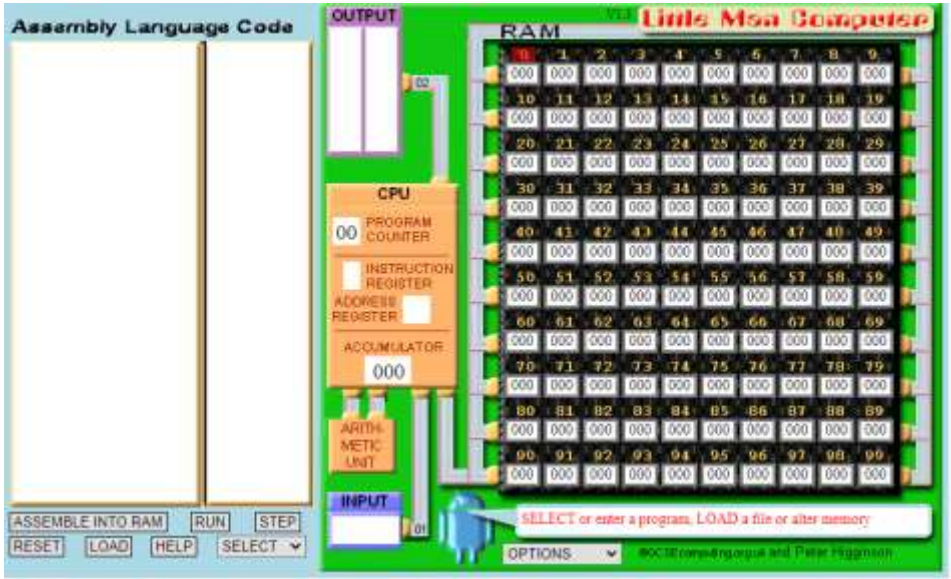
Little man computer
(assembly language)

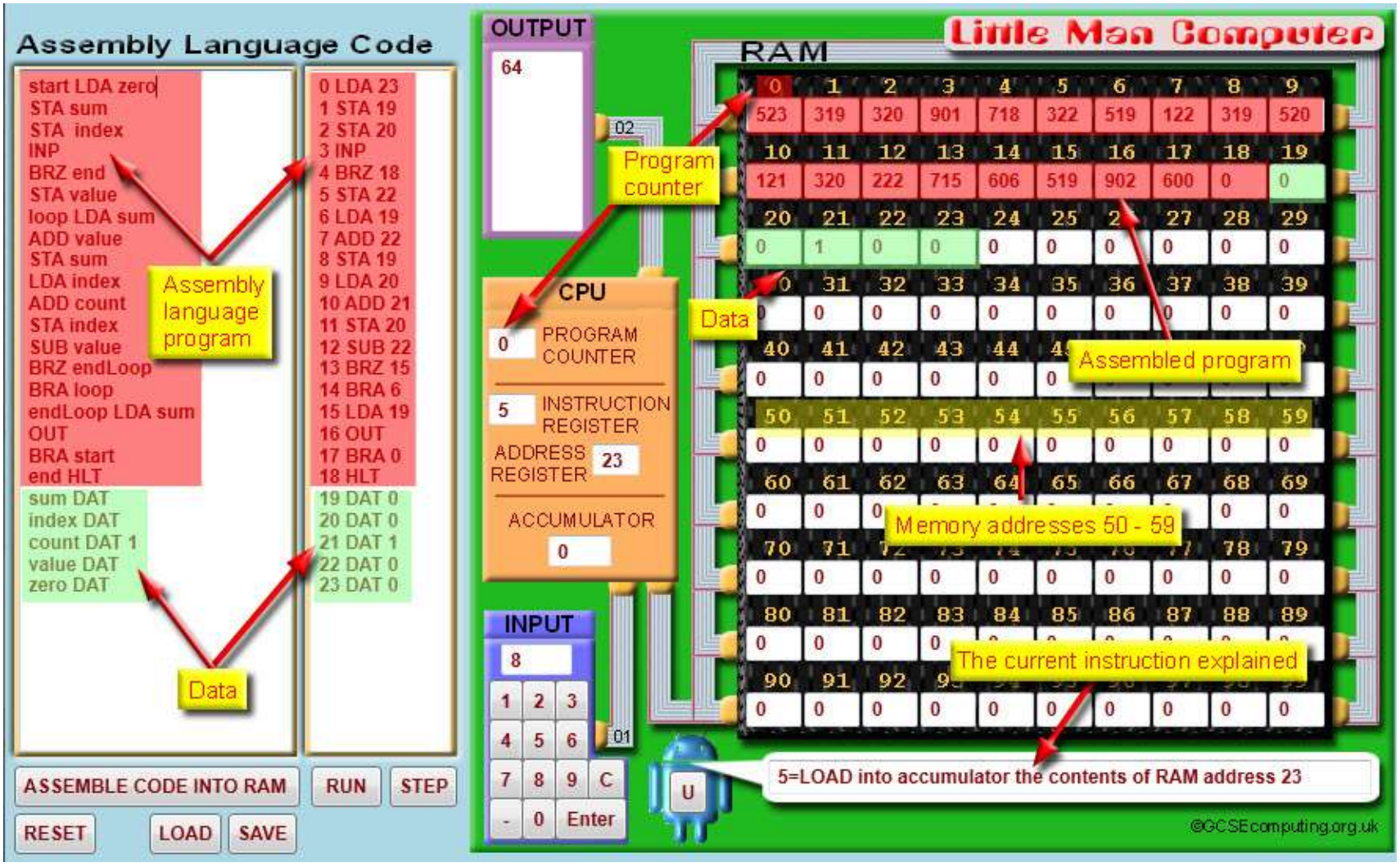
11.5.1.3 analyze a simple program written in the language of assembly

Assessment criteria:

-uses basic instructions of LMC(little man computer)
-knows what is mnemonic
-knows what is operand and opcode
-understand importance of memory in Von Neumann architecture.

Little man computer - simulator of assembly language.




Assembly Language Code
start LDA zero
STA sum
STA index
INP
BRZ end
STA value
loop LDA sum
ADD value
STA sum
LDA index
ADD count
STA index
SUB value
BRZ endLoop
BRA loop
endLoop LDA sum
OUT
BRA start
end HLT
sum DAT
index DAT
count DAT 1
value DAT
zero DAT
0 LDA 23
1 STA 19
2 STA 20
3 INP
4 BRZ 18
5 STA 22
6 LDA 19
7 ADD 22
8 STA 19
9 LDA 20
10 ADD 21
11 STA 20
12 SUB 22
13 BRZ 15
14 BRA 6
15 LDA 19
16 OUT
17 BRA 0
18 HLT
19 DAT 0
20 DAT 0
21 DAT 1
22 DAT 0
23 DAT 0
Assembly language program
Data
ASSEMBLE CODE INTO RAM
RUN
STEP
RESET
LOAD
SAVE
OUTPUT
64
02
Program counter
CPU
0
PROGRAM COUNTER
5
INSTRUCTION REGISTER
ADDRESS REGISTER
23
ACCUMULATOR
0
INPUT
8
1 2 3
4 5 6
7 8 9 C
- 0 Enter
01
U
RAM
Little Man Computer
0 1 2 3 4 5 6 7 8 9
523 319 320 901 718 322 519 122 319 520
10 11 12 13 14 15 16 17 18 19
121 320 222 715 606 519 902 600 0 0
20 21 22 23 24 25 27 28 29
0 1 0 0 0 0 0 0 0 0
31 32 33 34 35 36 37 38 39
0 0 0 0 0 0 0 0 0
40 41 42 43 44
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
70 71 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
90 91 92
0 0 0 0 0 0 0 0 0 0
Data
Assembled program
Memory addresses 50 - 59
The current instruction explained
5=LOAD into accumulator the contents of RAM address 23
©GCSEcomputing.org.uk

LMC Instruction Set

#	Instruction	Mnemonic	MachineCode	Explanation
1	Load	LDA	5xx	Load the contents of the given mailbox onto the accumulator.
2	Store	STA	3xx	Store the contents of the accumulator (calculator) to the mailbox of the given address.
3	Add	ADD	1xx	Add the contents of the given mailbox onto the accumulator (calculator).
4	Subtract	SUB	2xx	Subtract the contents of the given mailbox from the accumulator (calculator).
5	Input	INP	901	Copy the value from the "in box" onto the accumulator (calculator).
6	Output	OUT	902	Copy the value from the accumulator (calculator) to the "out box".
7	End	HLT	000	Causes the Little Man Computer to stop executing your program.
8	Branch if zero	BRZ	7xx	If the contents of the accumulator (calculator) are 000, the PC (program counter) will be set to the given address.
9	Branch if zero or positive	BRP	8xx	If the contents of the accumulator (calculator) are 000 or positive (i.e. the negative flag is not set), the PC (program counter) will be set to the given address.
10	Branch always	BRA	6xx	Set the contents of the accumulator (calculator) to the given address.
11	Data storage	DAT		When compiled, a program converts each instruction into a three-digit code. These codes are placed in sequential mailboxes. Instead of a program component, this instruction will reserve the next mailbox for data storage.

*xx refers to a Mailbox number

Basic instructions of LMC

A+B

```
        INP
        STA 99
              STA - opcode   99 - operand
        INP
              opcode+operand=mnemonic
        ADD 99
        OUT
        HLT
// Output the sum of two numbers
```

http://www.peterhigginson.co.uk/lmc/

For example we have mnemonic **add 45** and **add#45.**

What is difference between **45 and #45?**

Activity 1

1. A+B+C
2. A+C-B(A inputted first, B inputted second, C inputted third)
3. A-B+C

Descriptors:

- **correctly uses instructions of LMC**
- **correctly output result**

BRA,BRZ,BRP,DAT

BRA-Branch - use the address given as the address of the next instruction.
BRP-Branch to the address given if the Accumulator is zero or positive.
BRZ-Branch to the address given if the Accumulator is zero.
DAT-Used to indicate a location that contains data.

Define maximum of two numbers

```
00 INP
01 STA 99
02 INP
03 STA 98
04 SUB 99
05 BRP 10
06 BRA 07
07 LDA 99
08 OUT
09 HLT
10 LDA 98
11 OUT
12 HLT
```

Example of using instruction “DAT”

```
        INP
        STA FIRST
        INP
        ADD FIRST
        OUT
        INP
        SUB FIRST
        OUT
        HLT
FIRST   DAT
// Input three numbers.
// Output the sum of the first two
// and the third minus the first
```

```
00 INP
01 STA 09
02 INP
03 ADD 09
04 OUT
05 INP
06 SUB 09
07 OUT
08 HLT
09 DAT 00
```

Example of using instruction “DAT”

```
INP
SUB ONE  // Subtract the value stored at address ONE from the accumulator
OUT
ONE  DAT 1    // Store the value 1 in this memory address, and label it ONE (variable declaration)
```

Activity 2

- **define maximum of three numbers**
- **a*b**

Descriptors:

- **correctly uses LMC instructions**
- **correctly output result**

1 The code below is part of a program written in assembly language.

```
            mov     dx,1AH
            mov     ax,[loticks]
            sub     dx,ax
            mov     [hiticks],dx
            mov     ax,[hiticks]
            jmp     end
hiticks     dw      0H
loticks     dw      10H
end
```

10
1*16^1+0*16^0=16
1A
1*16^1+10*16^0=26

After this code has run, what values will be stored at memory locations `hiticks`, and `loticks`? Give your answers as **decimal** numbers.

`hiticks` .....

`loticks` ..... [2]

leetcode.com/

Programming Skills
Study Plan