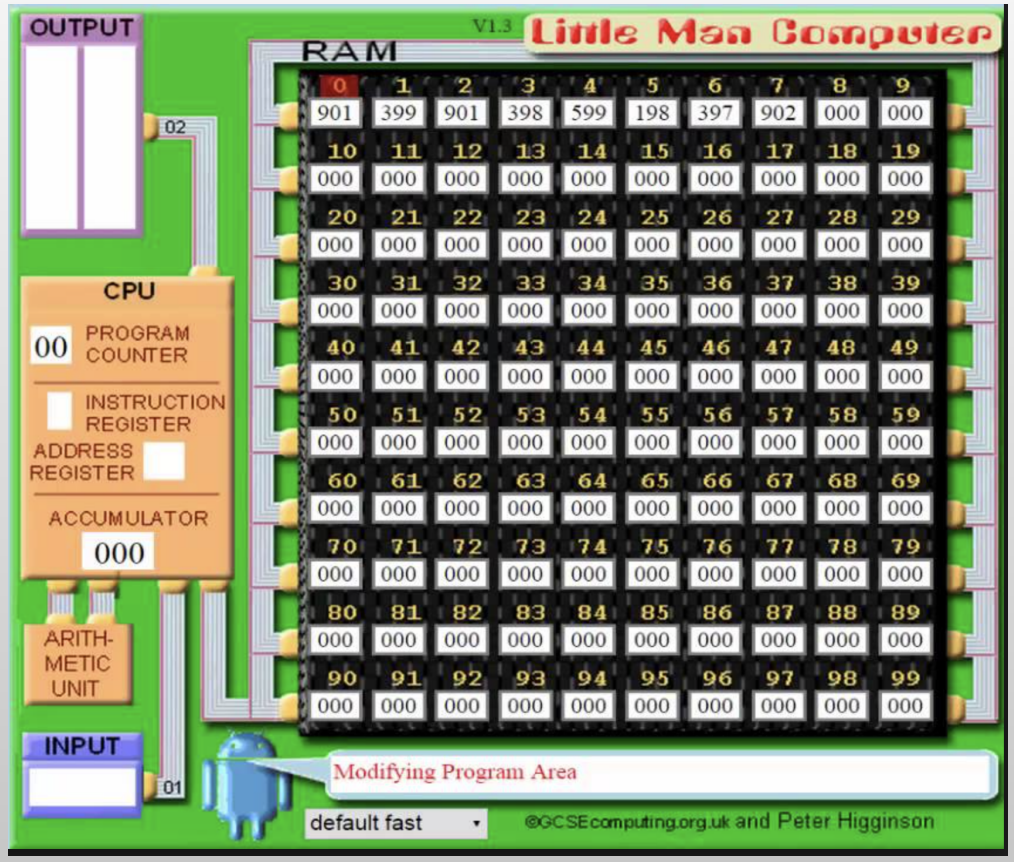
Chapter 6
Excursion:

Little Man Computer



Little Man Computer

- An assembly language simulator
- Shows the packets of information travelling around the machine
- Nice ability to explicitly see the use of the Instruction Register and the Accumulator

The Simulator

- **Accumulator** - The active memory of the simulator. Majority of our instructions will modify contents of the accumulator.
- **Program Counter** - This shows the current memory location that the processor is running.
- **MEM Address** - The current instruction type.
- **MEM Data** - The data being used for the current instruction.

The Simulator

- **In-Box** - The input box. Enter inputs here.
- **Out-Box** - The output box. Observe outputs here.
- **CPU** - The main processing unit of the simulated machine. Carries out the fetching, decoding and executing of the instructions provided.
- **Assembly Language Code -** environment allowing instructions to be entered as LMC assembly to be loaded onto the machine.

Taking Input

Name: Input
Mnemonic: INP
Code: *901*

Description:
The input instruction takes the value in the **In-Box** and puts the value into the **Accumulator**.

Next Action:
After the value has been copied the **Program Counter** will move onto the next (sequential) memory location.

Providing Output

Name: Output
Mnemonic: OUT
Code: *902*

Description:
The output instruction takes the value in the **Accumulator** and puts the value into the **Out-Box**.

Next Action:
After the value has been copied the **Program Counter** will move onto the next (sequential) memory location.

Stopping the Program

Name: Halt
Mnemonic: HLT
Code: *000*

Description:
The halt instruction does not affect any of the memory locations and stops the program.

Next Action:
The execution of the program will stop.

Example Program 1: Specification

- Create a program which:
  - Takes in an input from the user
  - Outputs it back to the user.
- Analyse the memory locations and write down the instruction codes your program generated.

Example Program 1: LMC code

```
___    INP    ___
___    OUT    ___
___    HLT    ___
```

The first line will take an input from the user and place it in the Accumulator

The second line will take the value in the Accumulator and send it to the output.

The third line stops the program.

Storing Data

Name: Store
Mnemonic: STA *variable*
Code: *3 _ _*

Description:
The store instruction will take the data from **Accumulator** and store it into an allocated memory location which will be referred to by the variable name given.

Next Action:
After the value has been copied the **Program Counter** will move onto the next (sequential) memory location.

Retrieving Data

Name: Load
Mnemonic: LDA variable
Code: *5* _ _

Description:
The load instruction will put the value stored at the variable location into the **Accumulator**.

Next Action:
After the value has been loaded into the **Accumulator** ,the **Program Counter** will move onto the next (sequential) memory location.

Data Memory Locations

Name: Data
Mnemonic: *variable* DAT
Code: *(the data)*

Description:
The Data instruction will reserve a memory location to store data. This location can then be referred to by the given name.

Next Action:
After the memory location has been reserved, the **Program Counter** will move onto the next (sequential) memory location.

“Variable”

In the STA, LDA and DAT instructions we see the use of a “variable” argument. This refers to a named memory location.

For example:

one dat 1

will reserve a memory location, named “*one*”, containing the value 1

add one

will add the content of memory location named “one” to the accumulator.

Example Program 2: Specification

- Create a program which:
  - Takes and stores in 2 inputs from the user
  - Outputs the first input followed by the second input.

Example Program 2: LMC Code

```
____   INP    ____
____   STA    var
____   INP    ____
____   OUT    ____
____   LDA    var
____   OUT    ____
____   HLT    ____

var    DAT    ____
```

Line 1 gets the first input.

Line 2 stores this in *var.*

Line 3 gets the next input.

Line 4 prints this back out.

Line 5 loads first value back from *var.*

Line 6 prints this out.

Line 7 stops the program.

What does line 8 do?

Example Program 3: Specification

- Create a program which:
  - Takes and stores 4 inputs from the user
  - Outputs the third input to the user

This is for you to try in your own time!

Addition

Name: Addition
Mnemonic: *ADD variable*
Code: 1 _ _

Description:
The add instruction adds the value stored in the given memory location to the accumulator.

Next Action:
After the value has been loaded into the **Accumulator** ,the **Program Counter** will move onto the next (sequential) memory location.

Subtraction

Name: Subtraction
Mnemonic: *SUB variable*
Code: *2 _ _*

Description:
The subtraction instruction subtracts the value stored in the given memory location away from the accumulator.

Next Action:
After the value has been loaded into the **Accumulator** ,the **Program Counter** will move onto the next (sequential) memory location.

Example Programs 4 and 5: Specification

- Create a program which:
  - Takes and stores in 2 inputs from the user
  - Outputs the sum of them.

- Create a program which:
  - Takes in three numbers and stores them
  - Outputs the sum of the first 2 numbers with the third subtracted. (i.e. A+B-C)

Go To (Branch Always)

Name: Branch Always
Mnemonic: *BRA variable*
Code: *6 _ _*

Description:
Moves the **Program Counter** to the memory location stored within the variable memory location.

Next Action:
After the memory location has been loaded that memory location will be executed.

Go To
(Branch If Zero)

Name: Branch If Zero
Mnemonic: *BRZ variable*
Code: *7 _ _*

Description:
Moves the **Program Counter** to the memory location stored within the variable memory location if the accumulator is equal to zero.

Next Action:
After the memory location has been loaded that memory location will be executed.

Go To
(Branch If Zero or Positive)

Name: Branch If Zero or Positive
Mnemonic: *BRP variable*
Code: *8 _ _*

Description:
Moves the **Program Counter** to the memory location stored within the variable memory location if the accumulator is zero or positive.

Next Action:
After the memory location has been loaded that memory location will be executed.

Little Man Computer Instruction Summary

____	**INP**	____	- Input
____	**OUT**	____	- Output
____	**HLT**	____	- Halt
____	**STA**	***var***	- *Store*
____	**LDA**	***var***	- Load
var	**DAT**	____	- Data
____	**ADD**	***var***	- Addition
____	**SUB**	***var***	- *Subtraction*
____	***BRA***	***var***	- Branch Always
____	***BRZ***	***var***	- Branch If Zero
____	***BRP***	***var***	- Branch If Positive

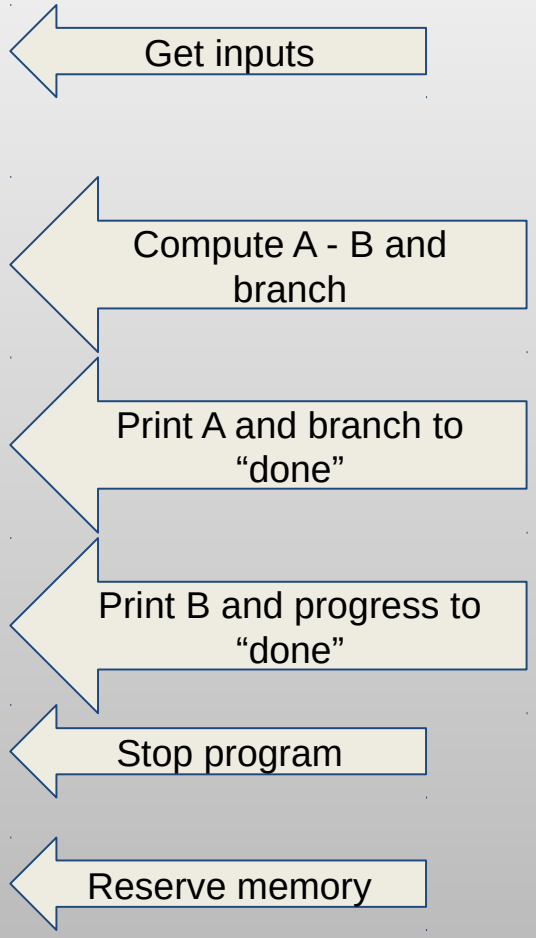
Example Program 6: Specification

- Create a program which:
  - Takes and stores in 2 inputs from the user.
  - Outputs the smallest number.

Hint: If you perform A - B and the result is positive, then A is **bigger** than B

Example Program 6: LMC Code

```
_____   INP  _____
_____   STA  var1
_____   INP  ____
_____   STA  var2

_____   LDA  var1
_____   SUB  var2
_____   BRP  print2

_____   LDA  var1
_____   OUT  _____
_____   BRA  done

print2  LDA  var2
        OUT  _____

done    HLT  _____

var1    DAT  _____
var2    DAT  _____
```



Example Program 7: Specification

- Create a program which:
  - Takes and stores in 2 inputs from the user.
  - Outputs the multiplication of the two numbers.

This is for you to try in your own time!