The Little Man Computer

• Peter Rounce
  p.rounce@cs.ucl.ac.uk

Taken from The Architecture of Computer Hardware and Systems Software by Irv Englander
Then “Little Man Computer” (the name used in the text book) is a reasonable analogy of a computer system. It works in an identical fashion to a computer, although it has less complexity and thus restricted capabilities. The simplicity of this analogy reflects the underlying simplicity of the operation of a computer: the basic operations of a computer are as simple as the basic operation of this analogy. It is the number of operations that are performed and the speed of operation of a computer that produces the complexity of its operation.

Like a computer, the LMC has
- a control unit, the little man, with temporary storage facilities and the capability of executing simple instructions.
- A small set of instructions that allow “programs” to be specified and that can be executed by the little man.
- A calculating unit (calculator) to manipulate/process data.
- General storage (storage boxes) for instructions and data.
- Locations (In-Tray & Out-Tray) where data can be passed into and out of the computer
- An instruction pointer (IP) to indicate where the next instruction is to be read (fetched) from.

Little Man Computer – a simple analogy of a computer

Taken from:–
Irv Englebard
The Architecture of Computer Hardware & Systems Software
Little Man Computer – a simple analogy of a computer

- **In Tray**
- **Out Tray**
- **Calculator**
- **Instruction Pointer (IP)**
- **Increment Button**
- **Reset Button**

**Storage Boxes**

<table>
<thead>
<tr>
<th>Box number</th>
<th>00</th>
<th>01</th>
<th>02</th>
<th>03</th>
</tr>
</thead>
<tbody>
<tr>
<td>Box contents</td>
<td>600</td>
<td>283</td>
<td>500</td>
<td>295</td>
</tr>
</tbody>
</table>

Reset button zeroes IP operated by external agent.
Little Man Computer – a simple analogy of a computer

Storage Boxes

<table>
<thead>
<tr>
<th>Box number</th>
<th>Box contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>600</td>
</tr>
<tr>
<td>01</td>
<td>283</td>
</tr>
<tr>
<td>02</td>
<td>500</td>
</tr>
<tr>
<td>03</td>
<td>295</td>
</tr>
</tbody>
</table>

Instruction Pointer (IP)

| 00       | 00           |

In Tray

Out Tray

Increment Button

Reset Button

Calculator

1 7 5

Little Man Computer
Little Man Computer – a simple analogy of a computer

Taken from:-
Irv Englander
The Architecture of Computer Hardware & Systems Software

In-Tray: numbers for input are placed here

Out-Tray: output numbers are put here by LM

Increment button adds one to number in IP

Reset button zeroes IP

Storage Boxes

<table>
<thead>
<tr>
<th>Box number</th>
<th>Box contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>600</td>
</tr>
<tr>
<td>01</td>
<td>283</td>
</tr>
<tr>
<td>02</td>
<td>500</td>
</tr>
<tr>
<td>03</td>
<td>295</td>
</tr>
</tbody>
</table>

In-Tray:

1 7 5

Calculator

LM types on calculator to enter numbers and perform calculations

LM types on calculator to enter numbers and perform calculations

Instruction Pointer (IP)

0 0

LM on calculator to enter numbers and perform calculations

LM types on calculator to enter numbers and perform calculations

LM types on calculator to enter numbers and perform calculations

In-Tray: numbers for input are placed here

Out-Tray: output numbers are put here by LM

Increment button adds one to number in IP

Reset button zeroes IP

1 7 5

In-Tray: numbers for input are placed here

Out-Tray: output numbers are put here by LM

Increment button adds one to number in IP

Reset button zeroes IP

1 7 5

In-Tray: numbers for input are placed here

Out-Tray: output numbers are put here by LM

Increment button adds one to number in IP

Reset button zeroes IP

1 7 5

In-Tray: numbers for input are placed here

Out-Tray: output numbers are put here by LM

Increment button adds one to number in IP

Reset button zeroes IP

1 7 5

In-Tray: numbers for input are placed here

Out-Tray: output numbers are put here by LM

Increment button adds one to number in IP

Reset button zeroes IP

1 7 5
A storage box holds a 3 digit decimal number, i.e. a value between 000 and 999. This value may represent an item of data, an instruction of a program, or even the address of another storage box. The value stored always has 3 digits, i.e. values like “1”, “20”, etc. are held as “001”, “020”. You should consider that a box always holds a 3-digit number, i.e. a box is never empty.

A storage box has an address to identify it: its box number. In the LMC, an address is a 2-digit decimal number, i.e. a value between 00 and 99, so that there is a maximum of 100 boxes. The storage boxes are equivalent to the memory of a computer.

Again an address always has 2 digits, never more and never less: the 1st box has address “00”, the 2nd “01”...

The Calculator holds a single value, its current value. The Calculator’s current value can be

• Over-written with a new value from the in-tray or a storage box
• Written into a storage box or placed into the out-tray
• Can be modified by using it in a calculation with another value – add, subtract, multiply, etc.

The Calculator is equivalent to the ALU (Arithmetic Logic Unit) in a real computer processor. It holds a 3 digit value, again no more and no less digits: it always has a value, it is never empty.

The Instruction Pointer (IP) holds the address of the storage box from which the next instruction is to be fetched: it is the sole means of identifying the location of the next instruction to execute. It always has 2 digits.

An Instruction is a number with 3 hexadecimal digits, so it can be stored in a storage box. Its first digit defines the instruction. The remaining digits contain information needed by the instruction, as required.

The In-Tray and Out-Tray are holding places for 3-digit data values in transit into and out of the computer. They are ordered in that the little man can only take the top value from the In-Tray and can only put a value at the bottom of the Out-Tray. Values are removed from the Trays when read, so Trays can be empty.

Keywords: address, instruction, data, storage (or memory), instruction pointer, processor or CPU (central processor unit) – the “little man”.

Taken from:-
Irv Engleman
The Architecture of Computer Hardware & Systems Software
The operation of the LMC parallels closely the way a computer operates.

The basic operation is to read and execute an instruction. While the computer is running, this operation is repeated ceaselessly: a computer executes a Fetch-Execute cycle.

The basic operation has 2 stages: the first stage is the same for all instructions and reads the instruction; the second is the execution of the instruction, which is different for each instruction.

The **Fetch stage** – reading the instruction:–

1. Read from the instruction pointer store (IP) the address of the box to read to get the instruction.
2. Increment the address in IP so that it holds (“points at”) the instruction of the immediately following instruction.
3. Read the instruction from a box, selecting the box by using the address read in stage 1.

The **execute stage** varies for each instruction, but the essential feature of all of them is that a value is read from somewhere and placed somewhere else, occasionally with some processing of the data being stored. Instructions are usually very simple: complexity of operation comes from executing very long sequences of instructions.

The Fetch-Execute cycle defined here is essentially all that a processor or CPU does from being switched on until switched off.
Little Man Computer – a simple analogy of a computer

Computer Operation:

Instruction Fetch Phase:

1: LM goes to IP to get number of next storage box to read.
2: LM hits increment button
3: LM goes to storage box and reads contents to get next instruction.
**Little Man Computer – a simple analogy of a computer**

**Computer Operation:**

**Instruction Fetch Phase:**

1: LM goes to IP to get number of next storage box to read.

2: LM hits increment button

3: LM goes to storage box and reads contents to get instruction.

**Instruction Execution Phase:**

LM analyses instruction and performs operation.

*and repeats above.....until it is stopped*
## Instruction Set for Little Man Computer

<table>
<thead>
<tr>
<th>Inst</th>
<th>op-code</th>
<th>operand</th>
<th>operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>5</td>
<td>box number</td>
<td>(load) read value from addressed box; store into calculator</td>
</tr>
<tr>
<td>STO</td>
<td>3</td>
<td>box number</td>
<td>(store) read value from calculator, store into addressed box</td>
</tr>
<tr>
<td>ADD</td>
<td>1</td>
<td>box number</td>
<td>read value from addressed box; add into calculator</td>
</tr>
<tr>
<td>SUB</td>
<td>2</td>
<td>box number</td>
<td>read value from box; subtract from calculator</td>
</tr>
<tr>
<td>IN</td>
<td>9</td>
<td>01</td>
<td>take value from in-tray; type into calculator</td>
</tr>
<tr>
<td>OUT</td>
<td>9</td>
<td>02</td>
<td>read value from calculator, place into out-tray</td>
</tr>
<tr>
<td>BR</td>
<td>6</td>
<td>box number</td>
<td>(branch) put box number into instruction pointer (IP)</td>
</tr>
<tr>
<td>BRZ</td>
<td>7</td>
<td>box number</td>
<td>put box number into IP if last calculator result zero</td>
</tr>
<tr>
<td>BRP</td>
<td>8</td>
<td>box number</td>
<td>put box number into IP if last calculator result positive</td>
</tr>
<tr>
<td>HLT</td>
<td>0</td>
<td>00</td>
<td>stop running: re-start to re-run program by pressing reset</td>
</tr>
</tbody>
</table>
Program to calculate Absolute value of difference between 2 input numbers: \(|X - Y|\)

00  901  IN  # read in first number, X
01  310  STO 10  # store X in box with address 10
02  901  IN  # read in second number, Y
03  311  STO 11  # store Y in box 11
04  210  SUB 10  # subtract X from value in calculator \(\Rightarrow Y - X\)
05  808  BRP 08  # jump to output Y-X if Y-X is positive
06  510  LDA 10  # load X into calculator from box
07  211  SUB 11  # subtract Y from value in calculator \(\Rightarrow X - Y\)
08  902  OUT  # output difference between X and Y
09  000  HLT  # finished program
10  000  # store to save X: can be any value at start
11  000  # store to save Y: can be any value at start

The program above finds the absolute difference between 2 input values, i.e. the answer is a positive number.

The 2 numbers, whose difference is to be calculated, are assumed to be in the In-Tray at the start.

The first instruction reads, and removes, the 1\(^{st}\) value in the In-Tray, storing it in the calculator, and then the 2\(^{nd}\) instruction takes a copy of the value from the calculator and stores it in a box, box number 10.

Thus this pair of instructions gets a value from the In-Tray into a storage location within the computer where it can be accessed later.

The 3\(^{rd}\) and 4\(^{th}\) instructions (in boxes “02” and “03”) get a 2\(^{nd}\) value from the in-tray into box number 11.

At this stage, box 11 hold the 2\(^{nd}\) value, but so does the calculator: storing the value in the calculator into a box does not remove the value from the calculator. Thus the instruction in box “04” subtracts the contents of box number “10” from the value in the calculator to calculate the difference between the input numbers: \(Y - X\).

Let’s see how this is done in detail by executing the program: put 2 numbers “025” and “028” in the in-tray, and push the **Reset** button to let the little man run!
Program to calculate Absolute value of difference between 2 input numbers: \(|X - Y|\)

00  901  IN       # read in first number, X
01  310  STO  10  # store X in box with address 10
02  901  IN       # read in second number, Y
03  311  STO  11  # store Y in box 11
04  210  SUB  10  # subtract X from value in calculator(Y) => Y-X
Little Man (LM) Computer

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>901</td>
<td>IN</td>
</tr>
<tr>
<td>01</td>
<td>310</td>
<td>STO 10</td>
</tr>
<tr>
<td>02</td>
<td>901</td>
<td>IN</td>
</tr>
<tr>
<td>03</td>
<td>311</td>
<td>STO 11</td>
</tr>
<tr>
<td>04</td>
<td>210</td>
<td>SUB 10</td>
</tr>
<tr>
<td>05</td>
<td>808</td>
<td>BRP 08</td>
</tr>
<tr>
<td>06</td>
<td>510</td>
<td>LDA 10</td>
</tr>
<tr>
<td>07</td>
<td>211</td>
<td>SUB 11</td>
</tr>
<tr>
<td>08</td>
<td>902</td>
<td>OUT</td>
</tr>
<tr>
<td>09</td>
<td>000</td>
<td>HLT</td>
</tr>
<tr>
<td>10</td>
<td>000</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>000</td>
<td></td>
</tr>
</tbody>
</table>

In-trays are loaded, but LM is stopped waiting for RESET to be pushed.
Remember the sequence of operations is:-

Fetch:
1. Read box number from IP – address “00” in this case
2. Increment IP – it becomes “01”
3. Read the instruction from the box identified by the address read in stage 1: “901” read from box “00”.

Execute: “901” is an IN instruction
4. read and remove the top value from the in-tray
5. store the value read into the calculator

The computer does very simple operations, putting them together in sequences to perform the more complex operation of fetching and executing one instruction.

Putting instructions into sequences performs the even more complex tasks that the “program” is designed to perform.
Now ready to analyse and execute instruction 901

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>901</td>
<td>IN</td>
</tr>
<tr>
<td>01</td>
<td>310</td>
<td>STO 10</td>
</tr>
<tr>
<td>02</td>
<td>901</td>
<td>IN</td>
</tr>
<tr>
<td>03</td>
<td>311</td>
<td>STO 11</td>
</tr>
<tr>
<td>04</td>
<td>210</td>
<td>SUB 10</td>
</tr>
<tr>
<td>05</td>
<td>808</td>
<td>BRP 08</td>
</tr>
<tr>
<td>06</td>
<td>510</td>
<td>LDA 10</td>
</tr>
<tr>
<td>07</td>
<td>211</td>
<td>SUB 11</td>
</tr>
<tr>
<td>08</td>
<td>902</td>
<td>OUT</td>
</tr>
<tr>
<td>09</td>
<td>000</td>
<td>HLT</td>
</tr>
<tr>
<td>10</td>
<td>000</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>000</td>
<td></td>
</tr>
</tbody>
</table>
Little Man (LM) Computer

```
Address Contents Instruction
00  901  IN
01  310  STO  10
02  901  IN
03  311  STO  11
04  210  SUB  10
05  808  BRP  08
06  510  LDA  10
07  211  SUB  11
08  902  OUT
09  000  HLT
10  000
11  000
```

And execute!

In Tray  Out Tray  Reset Button  Increment Button  Instruction Pointer (IP)  Storage Boxes

Little Man (LM) Computer

0 2 5
After the execution of the instruction at address “00”, IP holds the address “01” and the calculator has the value of “025”.

So now start again by fetching of the next instruction,

1. “01” is read from IP
2. IP contents are changed to “02”
3. The instruction “310” is read from the box with address “01”

And then perform the execution stage (“310” is a store instruction):

4. The value “025” in the calculator is read. [This is a copy operation.]
5. This value is written into box “10” as specified by the last 2 digits of the instruction.

Note: reading the value of the calculator or the value in a box is a copy operation not a removal operation. Only reading from the InTray is a removal operation!!
Little Man (LM) Computer

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>901</td>
<td>IN</td>
</tr>
<tr>
<td>01</td>
<td>310</td>
<td>STO 10</td>
</tr>
<tr>
<td>02</td>
<td>901</td>
<td>IN</td>
</tr>
<tr>
<td>03</td>
<td>311</td>
<td>STO 11</td>
</tr>
<tr>
<td>04</td>
<td>210</td>
<td>SUB 10</td>
</tr>
<tr>
<td>05</td>
<td>808</td>
<td>BRP 08</td>
</tr>
<tr>
<td>06</td>
<td>510</td>
<td>LDA 10</td>
</tr>
<tr>
<td>07</td>
<td>211</td>
<td>SUB 11</td>
</tr>
<tr>
<td>08</td>
<td>902</td>
<td>OUT</td>
</tr>
<tr>
<td>09</td>
<td>000</td>
<td>HLT</td>
</tr>
<tr>
<td>10</td>
<td>000</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>000</td>
<td></td>
</tr>
</tbody>
</table>

Fetch → analyse → execute

In Tray 028
Out Tray
Increment Button
Instruction Pointer (IP)
Calculator
Storage Boxes

Reset Button
Now the instruction at address “02” is fetched.

Another IN instruction so the value “028” is read and removed from the in-tray and stored in the calculator, overwriting the value “025” which has been there since the previous IN instruction.
Little Man (LM) Computer

Calculator

![Diagram of Little Man (LM) Computer]

Address Contents Instruction

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>901</td>
<td>IN</td>
</tr>
<tr>
<td>01</td>
<td>310</td>
<td>STO 10</td>
</tr>
<tr>
<td>02</td>
<td>901</td>
<td>IN</td>
</tr>
<tr>
<td>03</td>
<td>311</td>
<td>STO 11</td>
</tr>
<tr>
<td>04</td>
<td>210</td>
<td>SUB 10</td>
</tr>
<tr>
<td>05</td>
<td>808</td>
<td>BRP 08</td>
</tr>
<tr>
<td>06</td>
<td>510</td>
<td>LDA 10</td>
</tr>
<tr>
<td>07</td>
<td>211</td>
<td>SUB 11</td>
</tr>
<tr>
<td>08</td>
<td>902</td>
<td>OUT</td>
</tr>
<tr>
<td>09</td>
<td>000</td>
<td>HLT</td>
</tr>
<tr>
<td>10</td>
<td>025</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>000</td>
<td></td>
</tr>
</tbody>
</table>
Now fetch and execute the store at address “03” to read the value “028” in the calculator and to store it in address “11”.
Little Man (LM) Computer

Address Contents Instruction
00 901 IN
01 310 STO 10
02 901 IN
03 311 STO 11
04 210 SUB 10
05 808 BRP 08
06 510 LDA 10
07 211 SUB 11
08 902 OUT
09 000 HLT
10 025
11 000
Now, the actual work of the program can be done.

Since the 2nd value read in from the InTray is still in the calculator, the 1st value read from the In-Tray can be subtracted from it to find the difference.

Thus, the next instruction at address “04” is a subtract instruction, so LM does the following:-

1. “04” is read from IP
2. IP contents are changed to “05”
3. The instruction “210” is read from the box with address “04”

And during the execution stage:-

4. The value “025” is read from box “10”: the latter is specified by the last 2 digits of the instruction.
5. This value “025” is subtracted from the current value of the calculator, “028”.
Little Man (LM) Computer

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>901</td>
<td>IN</td>
</tr>
<tr>
<td>01</td>
<td>310</td>
<td>STO 10</td>
</tr>
<tr>
<td>02</td>
<td>901</td>
<td>IN</td>
</tr>
<tr>
<td>03</td>
<td>311</td>
<td>STO 11</td>
</tr>
<tr>
<td>04</td>
<td>210</td>
<td>SUB 10</td>
</tr>
<tr>
<td>05</td>
<td>808</td>
<td>BRP 08</td>
</tr>
<tr>
<td>06</td>
<td>510</td>
<td>LDA 10</td>
</tr>
<tr>
<td>07</td>
<td>211</td>
<td>SUB 11</td>
</tr>
<tr>
<td>08</td>
<td>902</td>
<td>OUT</td>
</tr>
<tr>
<td>09</td>
<td>000</td>
<td>HLT</td>
</tr>
<tr>
<td>10</td>
<td>025</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>028</td>
<td></td>
</tr>
</tbody>
</table>

In Tray

Out Tray

Increment Button

Instruction Pointer (IP)

Storage Boxes

00 901
01 310
02 901
03 311
04 210
05 808
06 510
07 211
08 902
09 000
10 025
11 028

Calculator
Little Man (LM) Computer

And have completed 1st 5 instructions!

Address Contents Instruction
00  901  IN
01  310  STO  10
02  901  IN
03  311  STO  11
04  210  SUB  10
05  808  BRP  08
06  510  LDA  10
07  211  SUB  11
08  902  OUT
09  000  HLT
10  025
11  028

In Tray
Out Tray

Increment Button
Instruction Pointer (IP)

Reset Button
Programme to calculate Absolute value of difference between 2 input numbers: \[ |X - Y| \]

<table>
<thead>
<tr>
<th>Line</th>
<th>Instruction</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>901 IN</td>
<td># read in first number, X</td>
</tr>
<tr>
<td>01</td>
<td>310 STO 10</td>
<td># store X in box with address 10</td>
</tr>
<tr>
<td>02</td>
<td>901 IN</td>
<td># read in second number, Y</td>
</tr>
<tr>
<td>03</td>
<td>311 STO 11</td>
<td># store Y in box 11</td>
</tr>
<tr>
<td>04</td>
<td>210 SUB 10</td>
<td># subtract X from value in calculator(Y) =&gt; Y-X</td>
</tr>
<tr>
<td>05</td>
<td>808 BRP 08</td>
<td># jump to output Y-X if Y-X is positive</td>
</tr>
<tr>
<td>06</td>
<td>510 LDA 10</td>
<td># load X into calculator from box</td>
</tr>
<tr>
<td>07</td>
<td>211 SUB 11</td>
<td># subtract Y from value in calculator (X) =&gt; X-Y</td>
</tr>
<tr>
<td>08</td>
<td>902 OUT</td>
<td># output difference between X and Y</td>
</tr>
<tr>
<td>09</td>
<td>000 HLT</td>
<td># finished program</td>
</tr>
<tr>
<td>10</td>
<td>000</td>
<td># store to save X: can be any value at start</td>
</tr>
<tr>
<td>11</td>
<td>000</td>
<td># store to save Y: can be any value at start</td>
</tr>
</tbody>
</table>
Little Man (LM) Computer

A snap-shot of the computer prior to fetch & execution of instruction from box 05

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>901</td>
<td>IN</td>
</tr>
<tr>
<td>01</td>
<td>310</td>
<td>STO 10</td>
</tr>
<tr>
<td>02</td>
<td>901</td>
<td>IN</td>
</tr>
<tr>
<td>03</td>
<td>311</td>
<td>STO 11</td>
</tr>
<tr>
<td>04</td>
<td>210</td>
<td>SUB 10</td>
</tr>
<tr>
<td>05</td>
<td>808</td>
<td>BRP 08</td>
</tr>
<tr>
<td>06</td>
<td>510</td>
<td>LDA 10</td>
</tr>
<tr>
<td>07</td>
<td>211</td>
<td>SUB 11</td>
</tr>
<tr>
<td>08</td>
<td>902</td>
<td>OUT</td>
</tr>
<tr>
<td>09</td>
<td>000</td>
<td>HLT</td>
</tr>
<tr>
<td>10</td>
<td>025</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>028</td>
<td></td>
</tr>
</tbody>
</table>
At this stage the calculator has the difference between the 2 numbers input from the InTray, as calculated by subtracting the 1st number input from the 2nd number input.

In general, the difference obtained may be positive or negative, since no ordering of the input is specified. So to get the correct answer for output, the program must check the difference, obtained by the program so far, to see if the difference is positive or negative, and to execute different instructions depending on the result.

The instruction at address “05” does this checking, and changes IP or not depending on this result. Thus:-
1. “05” is read from IP
2. IP contents are changed to “06”
3. The instruction “808” is read from the box with address “05” – BRP instruction

And during the execution stage:-
4. The value “003” is read from the calculator and checked to see if it is positive.
5. Since (4) gave a “true” result, the address “08” in the instruction is stored to IP.

This instruction has changed IP so that the next instruction will come not from address “06” but from address “08”. The sequence of instructions has been changed in a data dependant way.

[Note: if stage (4) above had resulted in a “false” (calculator value not positive), then stage (5) would not have been executed and IP would have remained at “06”: the address of the instruction after the BRP instruction.]
Little man fetches and executes the next instruction in the usual way:

Fetch Stage:

1. Read box number from IP – address “08” in this case
2. Increment IP – it becomes “09”
3. Read the instruction from the box identified by the address read in stage 1: “902” read from box “08”

Execute (“902” is an Out instruction):

4. read and take a copy of the value in the calculator, “003”
5. store the value read in 4 in the OutTray

Note: LM has no memory of what he did during the last instruction. When starting on this next instruction, he reads IP, which tells him where to get the next instruction.

IP is the element in the system that remembers where to get the next instruction – changing IP changes the sequence of instructions executed

Control the value IP and you control the instruction sequence.
Conditional branches are key instructions in producing complexity in programs.
Little Man (LM) Computer

Address Contents Instruction
00 901 IN
01 310 STO 10
02 901 IN
03 311 STO 11
04 210 SUB 10
05 808 BRP 08
06 510 LDA 10
07 211 SUB 11
08 902 OUT
09 000 HLT
10 025
11 028
On reading the last instruction the little man finds a halt instruction and stops for a “tea break”.

This is the major difference between the LMC and a real computer: the latter doesn’t need a tea break and only stops executing instructions when its power is turned off, although the instructions it executes are often only just checking whether there is any thing to be done.

A second difference is that computers work in binary: all - values instructions, data and addresses – are in binary not decimal.
Little Man (LM) Computer

Address Contents Instruction
00  901   IN
01  310   STO  10
02  901   IN
03  311   STO  11
04  210   SUB  10
05  808   BRP  08
06  510   LDA  10
07  211   SUB  11
08  902   OUT
09  000   HLT
10  025
11  028
You can go over this program instruction by instruction and micro-step by micro-step in the first LMC exercise within the web-based coursework system.
Programme to calculate Absolute value of difference between 2 input numbers: $|X - Y|$

00 901 IN  # read in first number, X
01 310 STO 10  # store X in box with address 10
02 901 IN  # read in second number, Y
03 311 STO 11  # store Y in box 11
04 210 SUB 10  # subtract X from value in calculator (Y) => Y-X
05 808 BRP 08  # jump to output Y-X if Y-X is positive
06 510 LDA 10  # load X into calculator from box
07 211 SUB 11  # subtract Y from value in calculator (X) => X-Y
08 902 OUT  # output difference between X and Y
09 000 HLT  # finished program
10 025  # store to save X: can be any value at start
11 028  # store to save Y: can be any value at start