Little Man Computer

Dr. Sapumal Ahangama
Department of Computer Science and Engineering

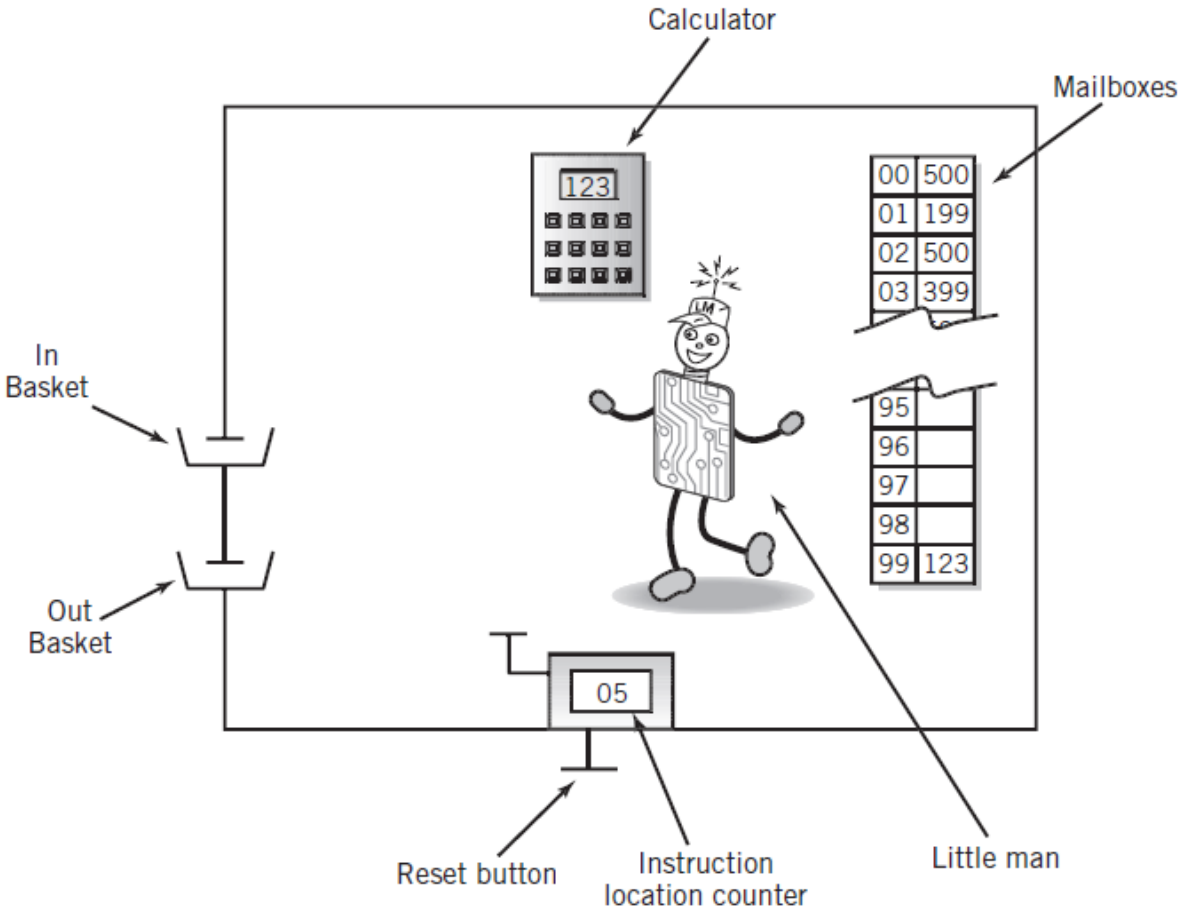
LITTLE MAN COMPUTER (LMC)

- Instruction model of a computer
  - The original LMC was created by Dr. Stuart Madnick at MIT in 1965
  - New version proposed in 1979
  - Models a simple von Neumann architecture computer
  - LMC is generally used to teach students

LITTLE MAN COMPUTER (LMC)

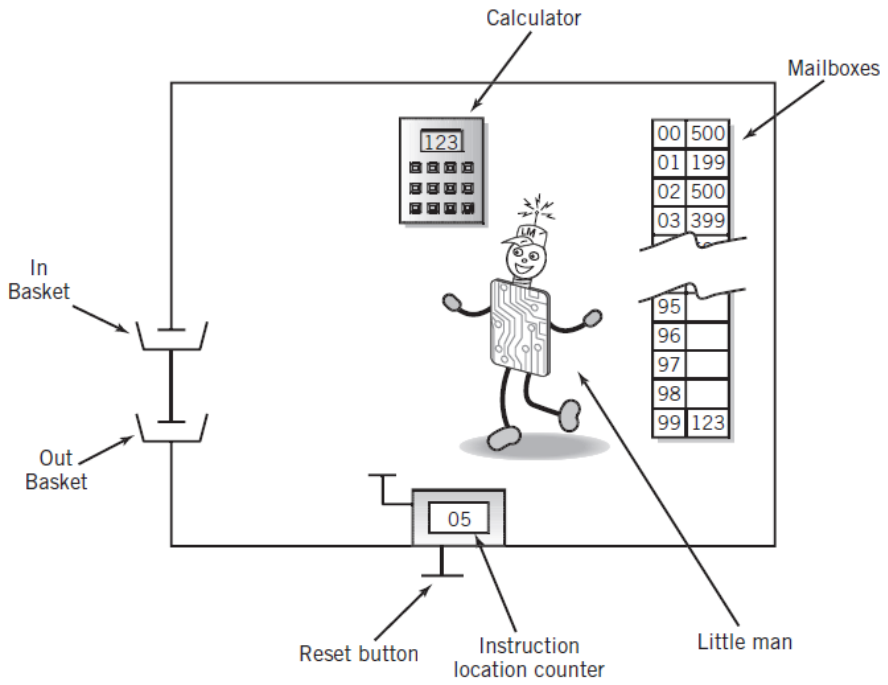
- Consists of,
  - Central Processing Unit (CPU) with Arithmetic and Logic Unit(ALU) and registers
  - Control Unit with Instruction Register and Program Counter
  - Input and Output mechanisms
  - Memory(RAM) to store both instruction and data

LITTLE MAN COMPUTER (LMC)



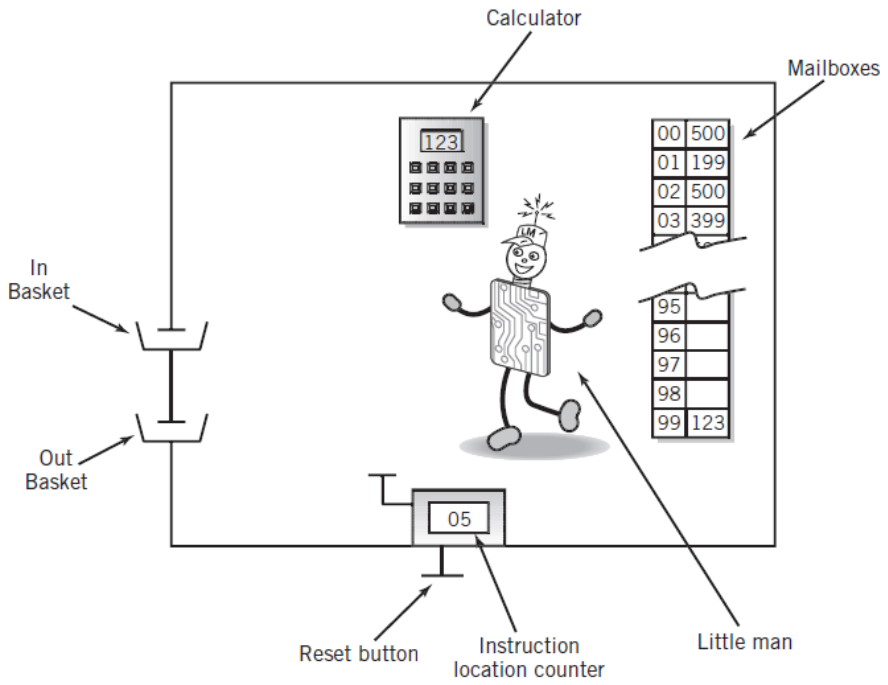
LITTLE MAN COMPUTER (LMC)

- The LMC consists of a walled mailroom. Inside the mailroom are several objects
  - A series of one hundred mailboxes
  - Each numbered with an address ranging from 00 to 99
  - Each mailbox is designed to hold a three-digit decimal number



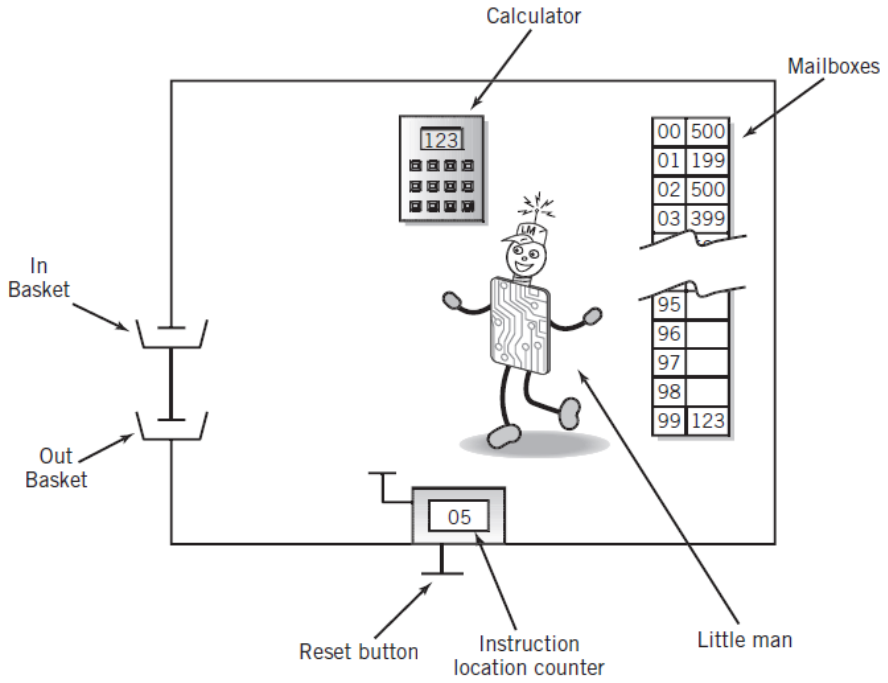
LITTLE MAN COMPUTER (LMC)

- The LMC consists of a walled mailroom. Inside the mailroom are several objects
  - A calculator
  - Can be used to enter and temporarily hold numbers, add and subtract
  - Display on the calculator is three digits wide



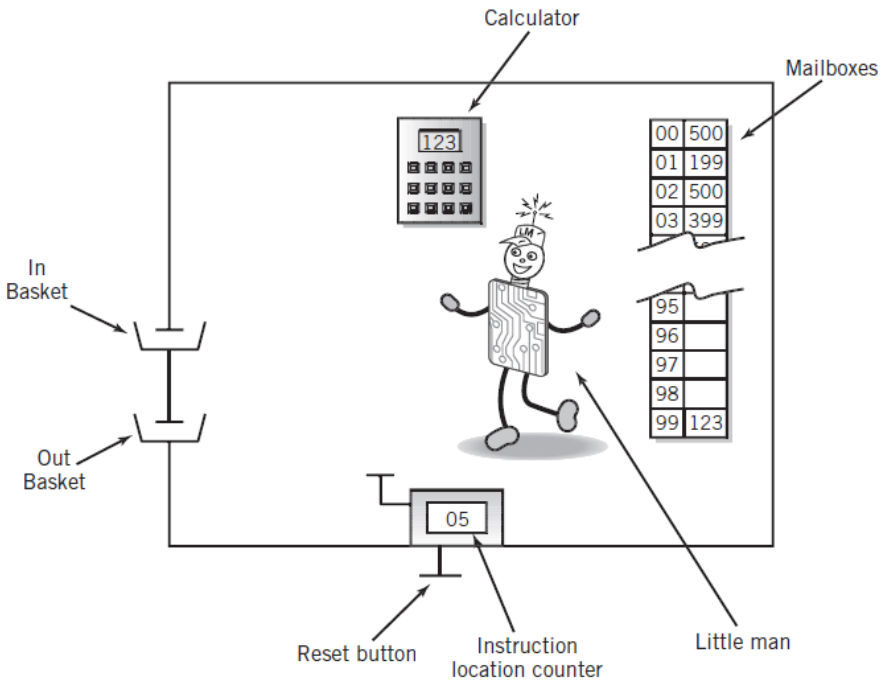
LITTLE MAN COMPUTER (LMC)

- The LMC consists of a walled mailroom. Inside the mailroom are several objects
  - An instruction location counter
  - Two-digit hand counter, click to increment the count.
  - The reset button for the hand counter is located outside the mailroom



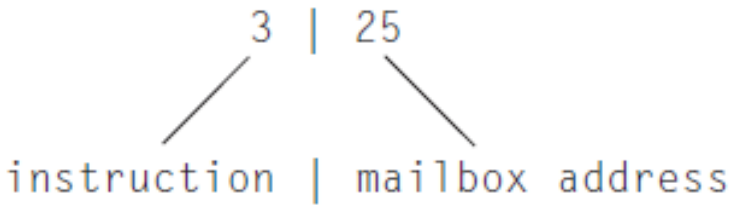
LITTLE MAN COMPUTER (LMC)

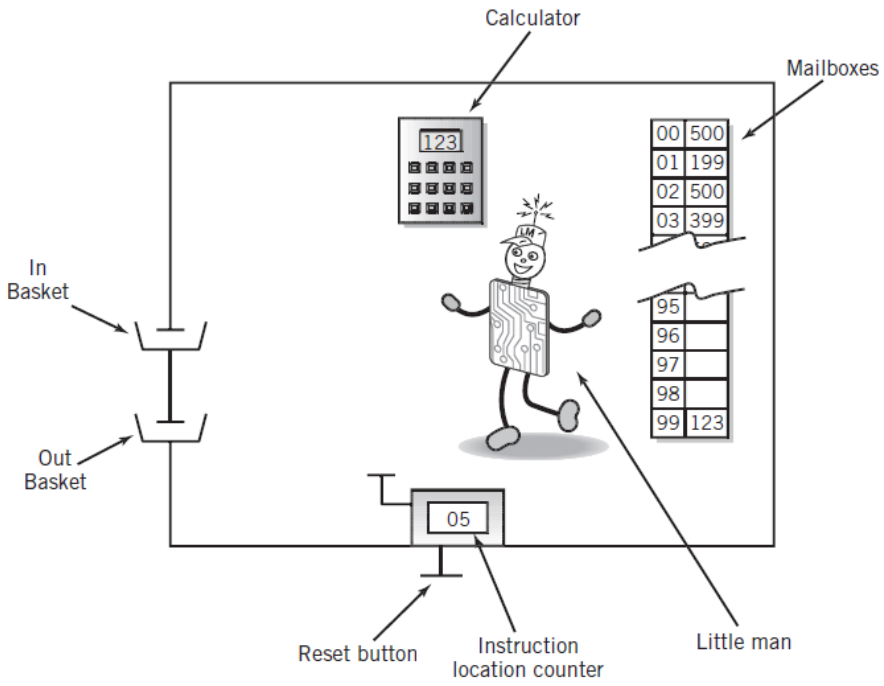
- The LMC consists of a walled mailroom. Inside the mailroom are several objects
  - In basket and an out basket
  - For interaction between the Little Man Computer and the outside environment



INSTRUCTIONS

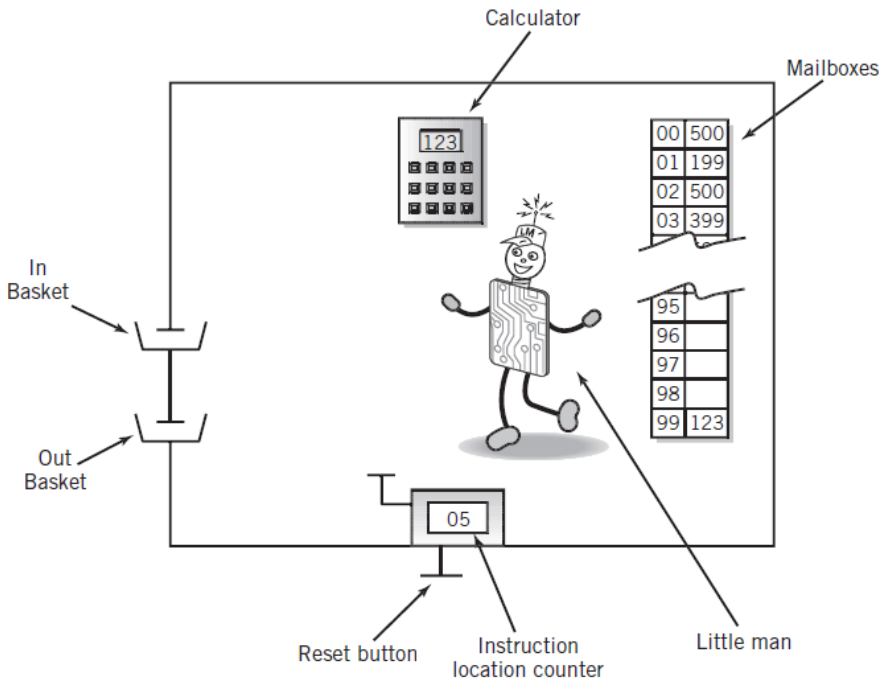
- Each instruction can be represented in a single digit
  - "operation code" or op code for short





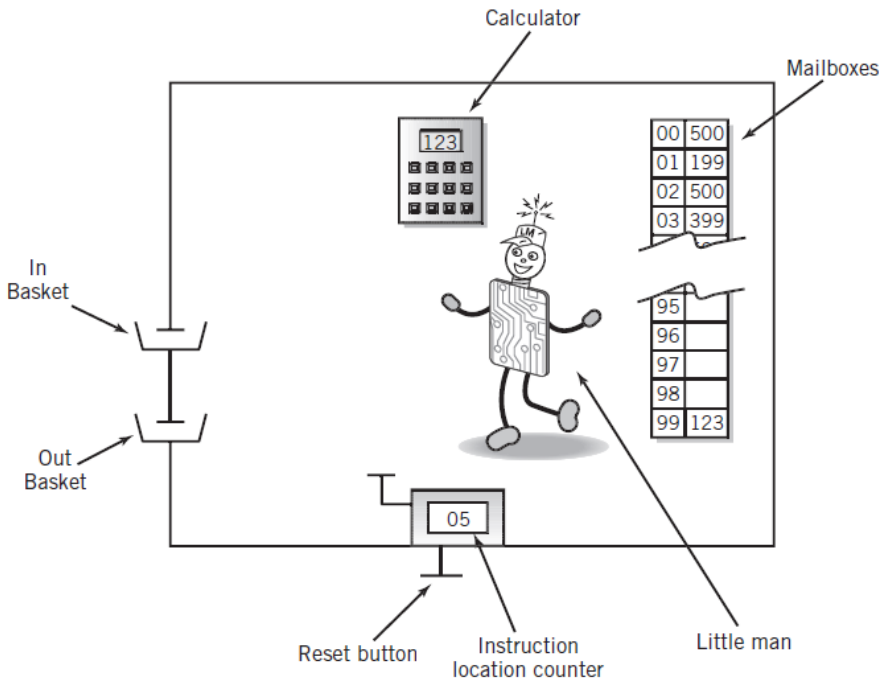
INSTRUCTIONS

- LOAD (LDA)
  - Op code = 5 (5XX)
  - Reads the XX mail box address and loads to the calculator



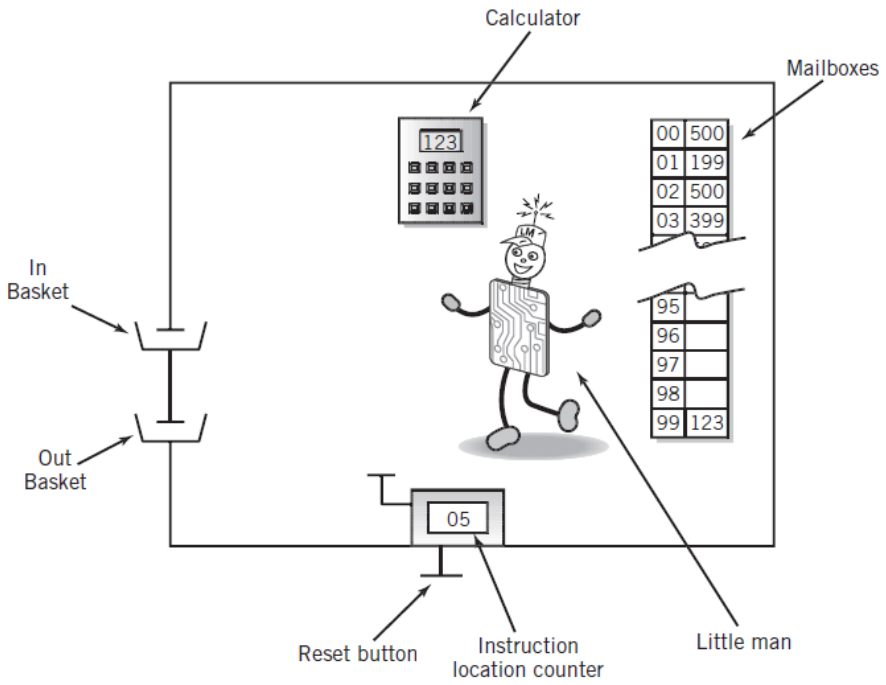
INSTRUCTIONS

- STORE (STO)
  - Op code = 3 (3XX)
  - Opposite of LOAD
  - Reads the number from the calculator and loads to the address



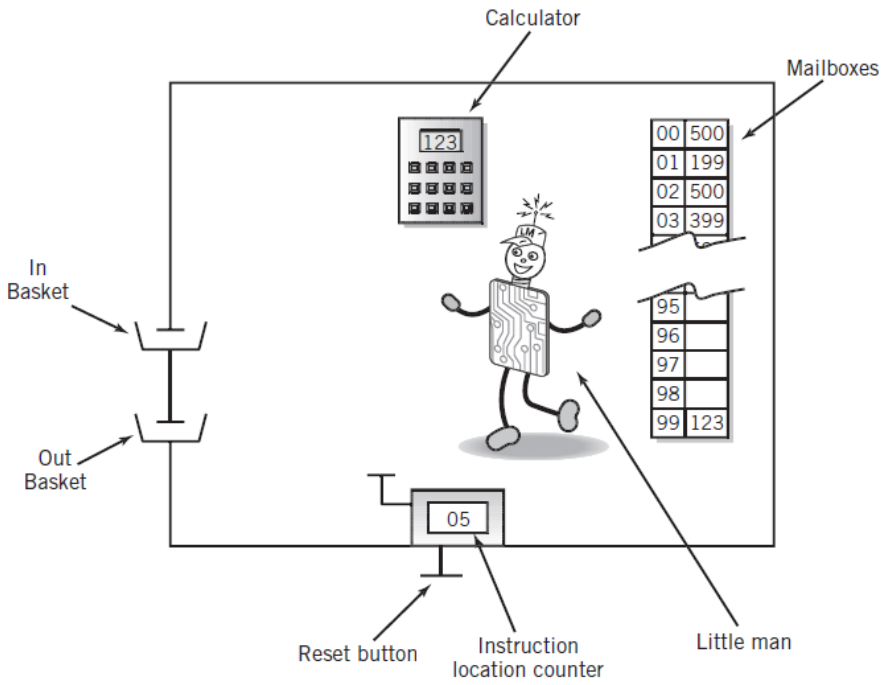
INSTRUCTIONS

- ADD (ADD)
  - Op code = 1 (1XX)
  - Reads the XX mail box address
  - Adds the value to the value already on the calculator



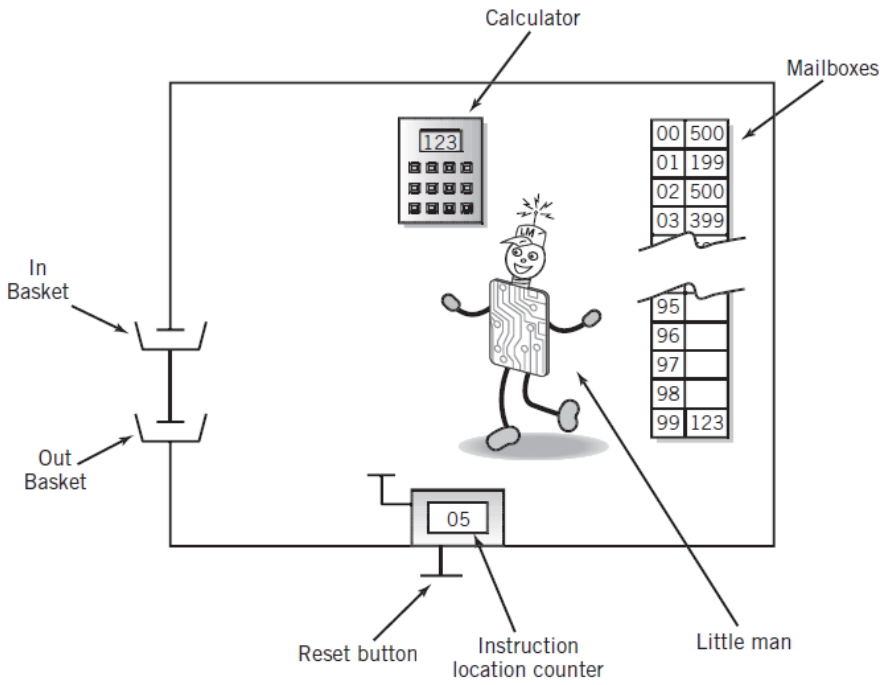
INSTRUCTIONS

- SUBTRACT (SUB)
  - Op code = 2 (2XX)
  - Reads the XX mail box address
  - Subtract the value from the value already on the calculator



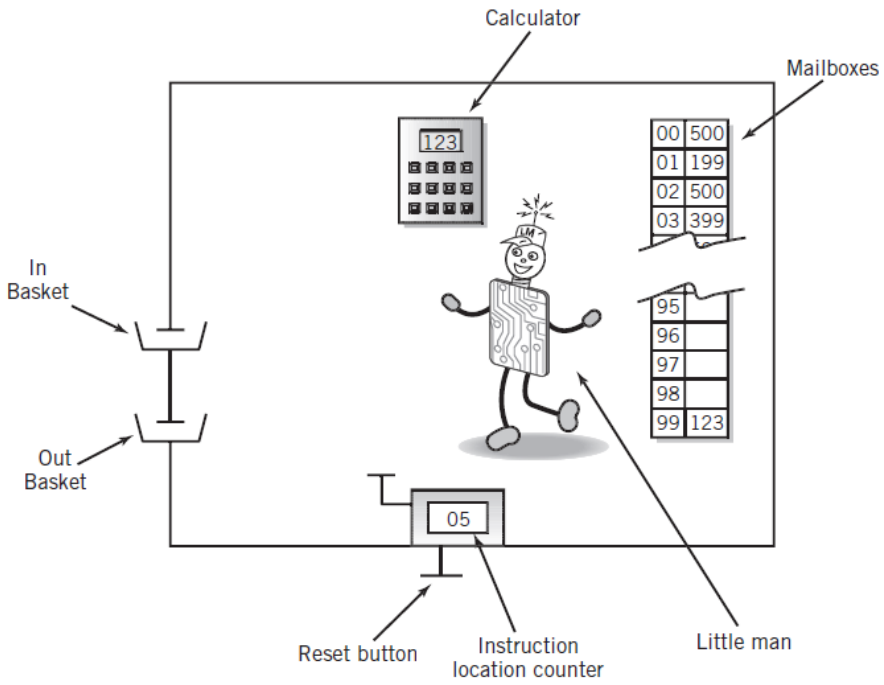
INSTRUCTIONS

- INPUT (IN)
  - Op code = 9 (901)
  - Reads the input basket
  - Input value to the calculator



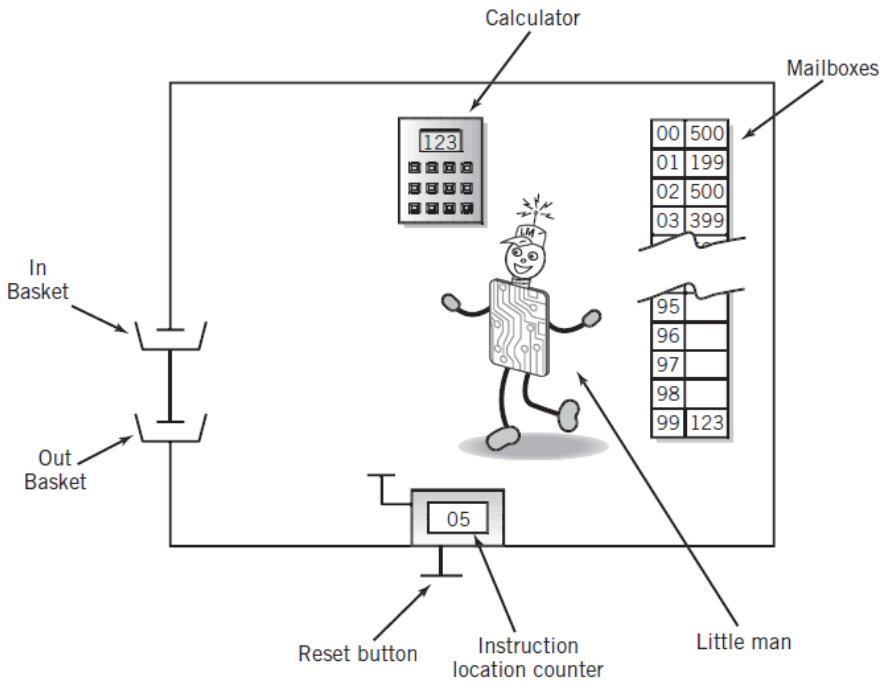
INSTRUCTIONS

- OUTPUT (OUT)
  - Op code = 9 (902)
  - Reads the value from the calculator
  - Outputs the value



INSTRUCTIONS

- COFFEE BREAK or HALT (COB or HLT)
  - Op code = 0 (000)
  - Processing stops
  - Address portion is ignored

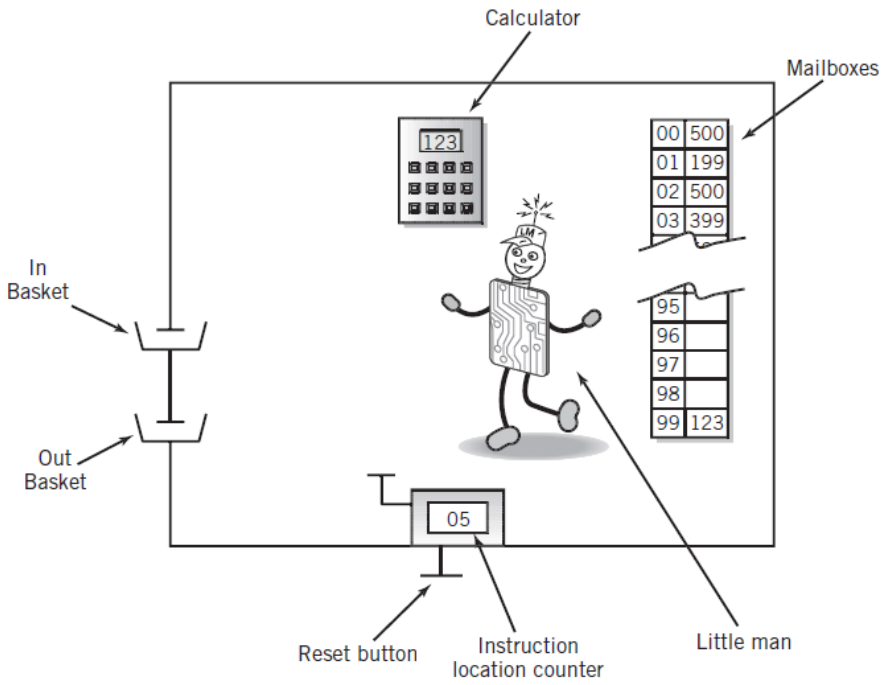


SIMPLE PROGRAM

- Output = Input 1 + Input 2

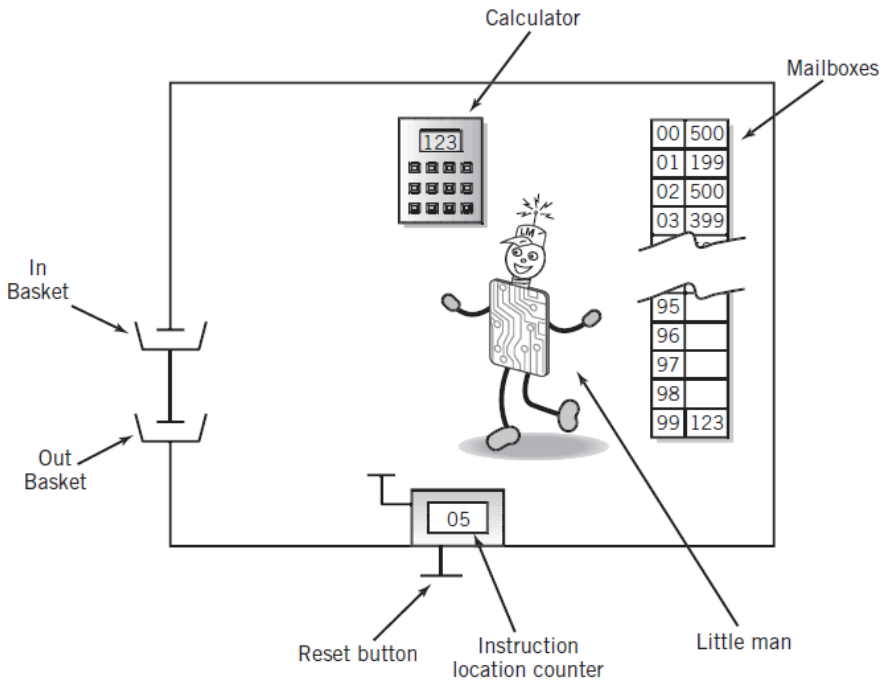
Program to Add Two Numbers

Mailbox code		Instruction description
00	901	INPUT
01	399	STORE DATA
02	901	INPUT 2ND #
03	199	ADD 1ST # TO IT
04	902	OUTPUT RESULT
05	000	STOP
99		DATA



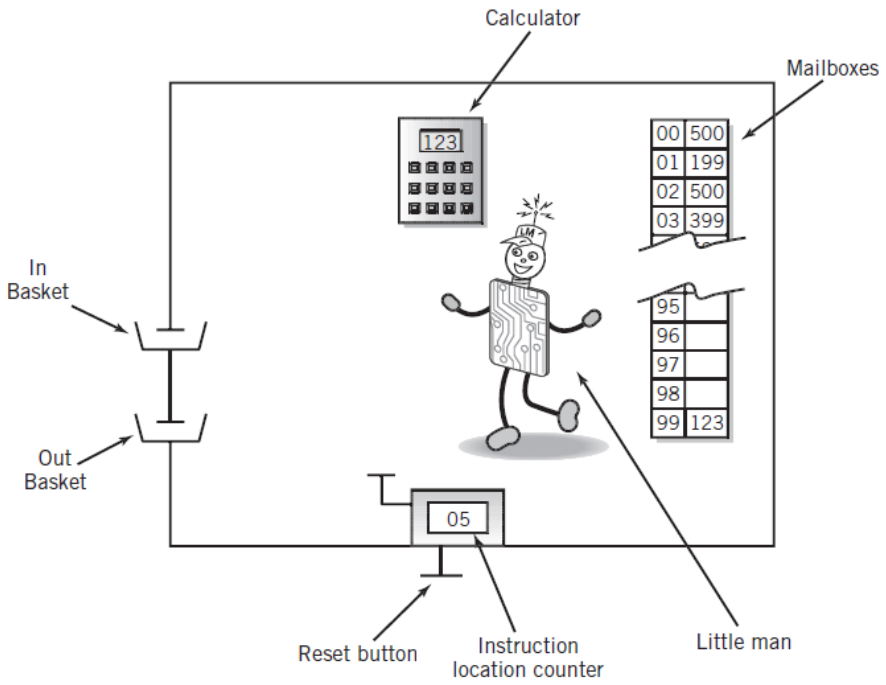
BRANCHING INSTRUCTIONS

- BRANCH UNCONDITIONALLY (BR)
  - Op code = 6 (6XX)
  - Sometimes known as JUMP
  - Change the counter to the location shown in the two address digits of the instruction



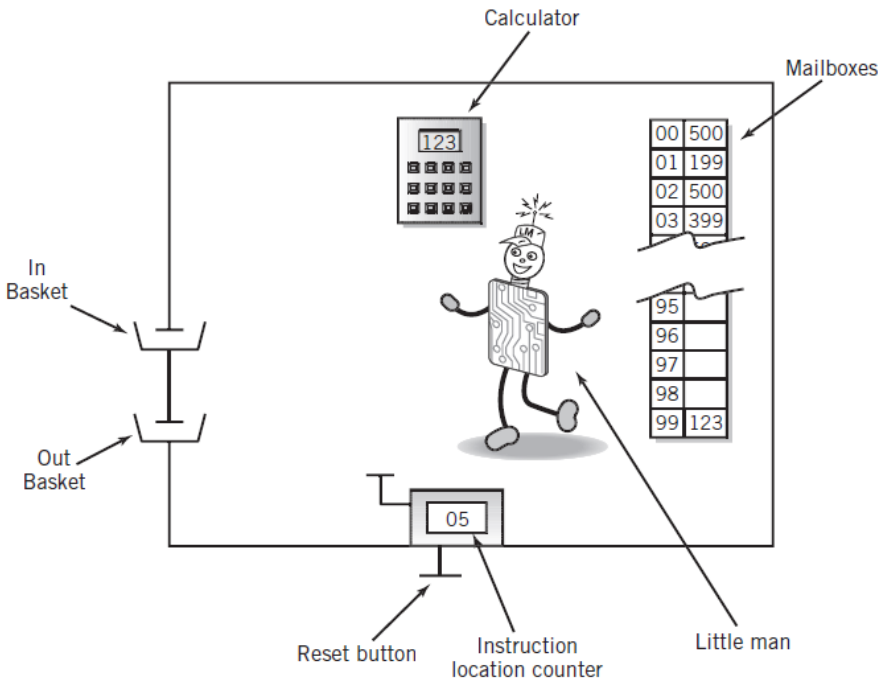
BRANCHING INSTRUCTIONS

- BRANCH ON ZERO (BRZ)
  - Op code = 7 (7XX)
  - If current value at the calculator is 0, modify the instruction counter
  - If not zero, proceed to the next sequence



BRANCHING INSTRUCTIONS

- BRANCH ON POSITIVE (BRP)
  - Op code = 8 (8XX)
  - If current value at the calculator is positive, modify the instruction counter
  - If not positive, proceed to the next sequence
  - 0 is considered as positive



INSTRUCTIONS

LDA	5xx	Load
STO	3xx	Store
ADD	1xx	Add
SUB	2xx	Subtract
IN	901	Input
OUT	902	Output
COB or HLT	000	Coffee break (or Halt)
BRZ	7xx	Branch if zero
BRP	8xx	Branch if positive or zero
BR	6xx	Branch unconditional
DAT		Data storage location

BRANCHING EXAMPLE 1

```
WHILE Value=0 DO
    Task;
NextStatement
```

```
45   LDA 90      590   90 is assumed to contain value
46   BRZ 48      748   Branch if the value is zero
47   BR 60       660   Exit loop; Jump to NextStatement
48     :               This is where the task is located
59   BR 45       645   End to Task; loop to test again
60                     Next statement
```

BRANCHING EXAMPLE 2

- Find "Positive" difference of 2 input numbers

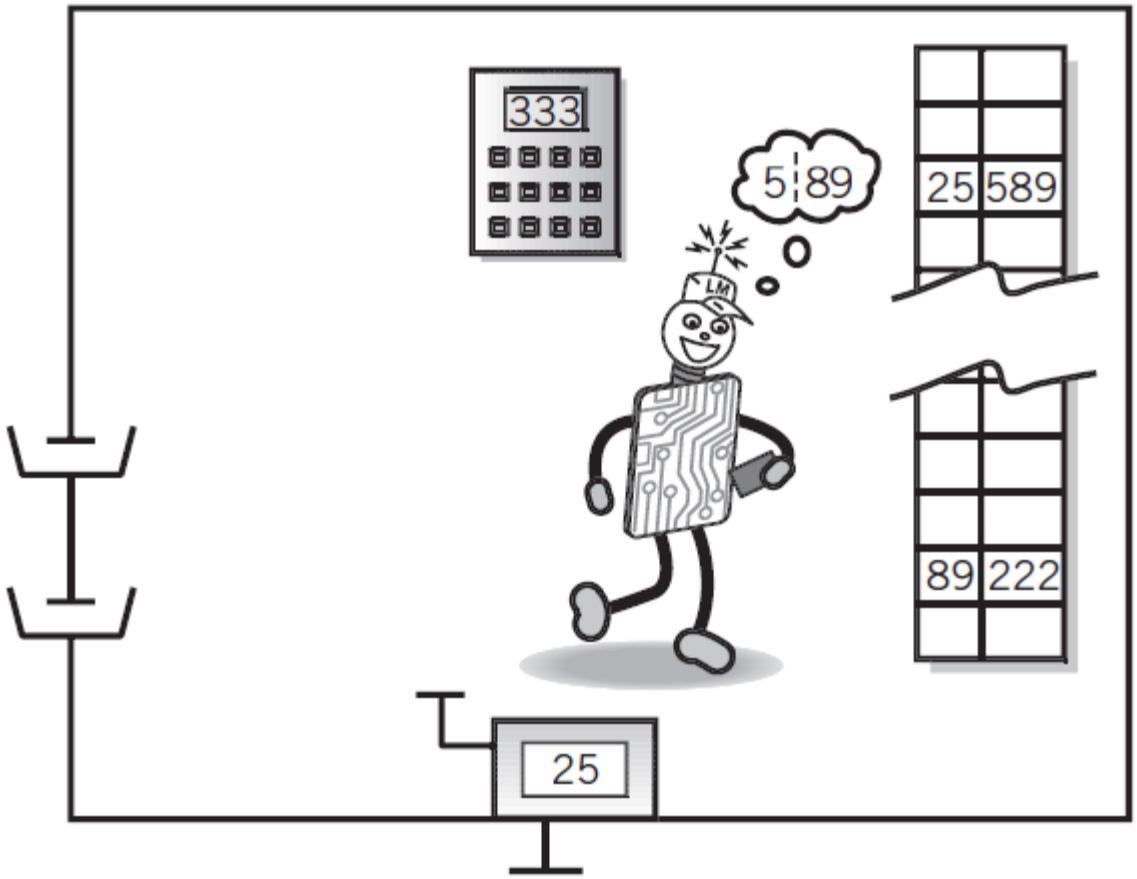
```
00   IN        901
01   STO  10   310
02   IN        901
03   STO  11   311
04   SUB  10   210
05   BRP  08   808   test
06   LDA  10   510   negative; reverse order
07   SUB  11   211
08   OUT       902   print result and
09   COB       000         stop.
10   DAT  00   000   used for data
11   DAT  00   000         "
```

INSTRUCTION CYCLES

- Can be broken into 2 parts
  - Fetch
    - Finds out what instruction is to be executed
  - Execute
    - Performs the work specified in the instruction

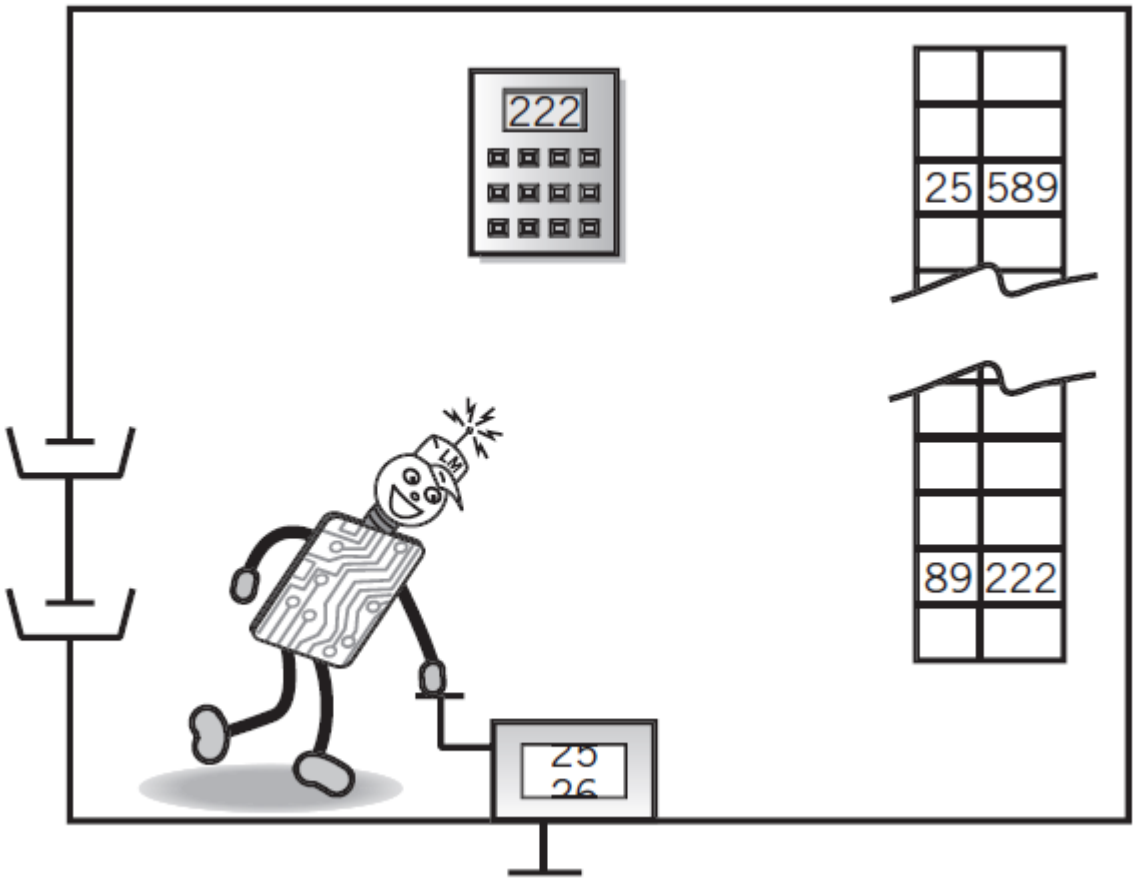
FETCH

- Finds out what instruction is to be executed



EXECUTE

- Performs the work specified in the instruction



INDIVIDUAL EXERCISE

- Translate the following to LMC assembly language

Input X # inputs an integer
Input Y # inputs an integer
if X>Y:
    output X
else:
    output Y
	input X
	while X>0:
	    Output X
	    X = X - 1

THANK YOU

REFERNCES

- Chapter 6: The Little Man Computer - The Architecture of Computer Hardware, Systems Software & Networking: An Information Technology Approach -4th Edition, Irv Englander -John Wiley and Sons