The Little Man Computer

The Little Man Computer

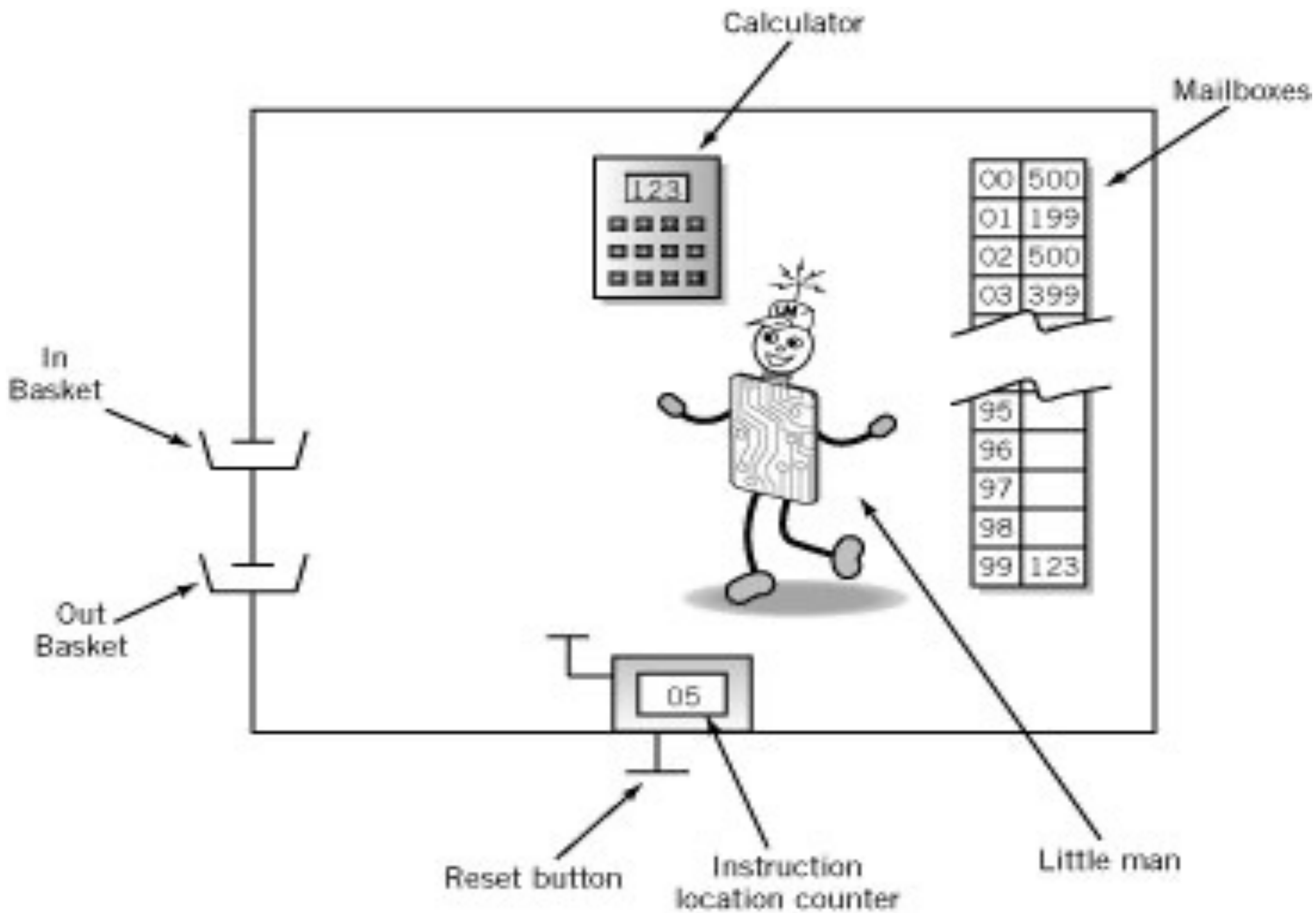
We will review the operations that the computer is capable of performing and look at how those operations work together to provide the computer with its power.

- We will begin by introducing a model of the computer (the LMC); a model that operates in a very similar way to real computers but that is easier to understand.

The Little Man Computer

- Using this model we will introduce:
  - a simplified, but typical, set of instructions that a computer can perform.
  - we will examine how these instructions are executed in the LMC
  - we will examine how these instructions are combined to form programs.

The Little Man Computer



Mailboxes: Address vs. Content

- Addresses are consecutive
- Content may be
  - Data or
  - Instructions

Address	Content

Mailboxes

- At one end of the room, there are 100 mailboxes (memory) numbered 0 to 99, that can each contain a 3 digit instruction or data
- The boss can place instructions in sequence in the mailboxes before the Little Man comes to work
- The Little Man's job is to read these instructions one by one and carry them out

Mailboxes: Content - Instructions

- Op code
  - Operation code
  - Arbitrary mnemonic
- Operand
  - Object to be manipulated
    - Data or
    - Address of data

Address	Content	
	Op code	Operand

Calculator

- The calculator can be used to enter and temporarily hold numbers, and also to add and subtract.
- The display on the calculator is three digits wide.
- For this discussion there is no provision made for negative numbers, or for numbers larger than three digits.

Hand Counter

- There is a two-digit hand counter, the type that you can click to increment the count.

- The reset button for the hand counter is located outside the mailroom.

- We will call the hand counter an instruction location counter. It is used to hold mailbox addresses.

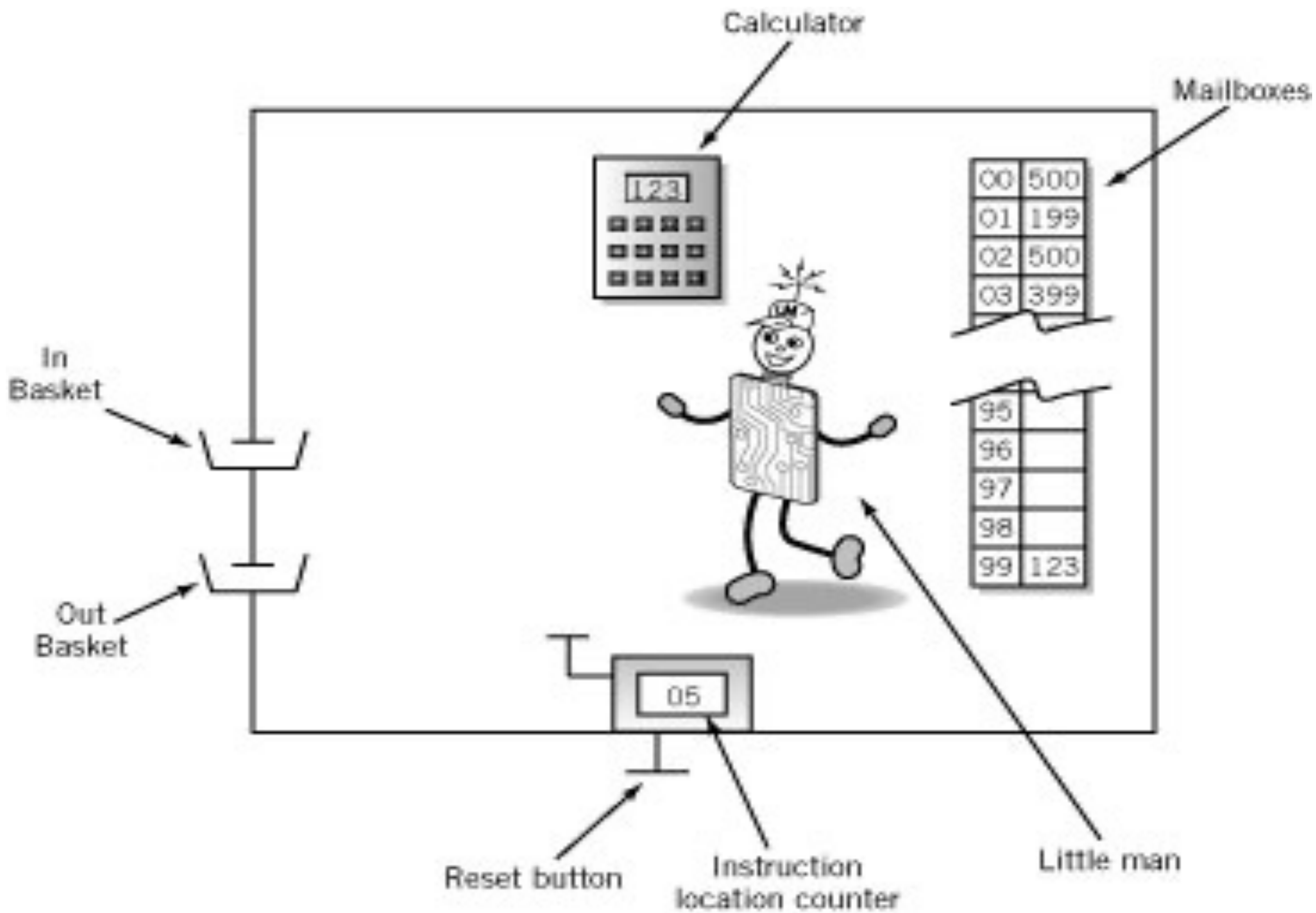
In and Out Basket

- The boss can set the counter at a starting address where the first of a sequence of instructions has been placed.
- The boss can also place data for the Little Man in the In Basket.
- The Little Man can be instructed to place data in the Out basket.
- The boss can send data to the LMC by putting a slip of paper with a three- digit number on it into the basket, to be read by the LMC.
- Similarly, the LMC can write a three-digit number on a slip of paper and leave it in the out basket, where it can be retrieved by the boss.
- Other than setting the first address on the hand counter, the only interaction between the LMC and the outside environment are the In and Out Baskets

Little Man

- Finally, there is the Little Man. His role is to repeatedly:
  - Read the mailbox address which appears in the counter
  - Retrieve the instruction from the mailbox address
  - Carry out the instruction
  - Click the hand counter button which increases its value by 1

The Little Man Computer



LMC – Instruction Set

- We would like the LMC to do some useful work, so we define a small group of 3-digit instructions that he can perform.

- Each instruction will consist of one digit which identifies which operation the LMC should perform – we will use the first of the three digits for this
- This digit is called the operation code or op code

- If the operation requires the LMC to use a particular mailbox we can use the other two digits to specify the mailbox address.

Instruction Set

Arithmetic	1xx	ADD
	2xx	SUBTRACT
Data Movement	3xx	STORE
	5xx	LOAD
Input/Output	901	INPUT
	902	OUTPUT
Machine Control (coffee break)	000	STOP

Input/Output

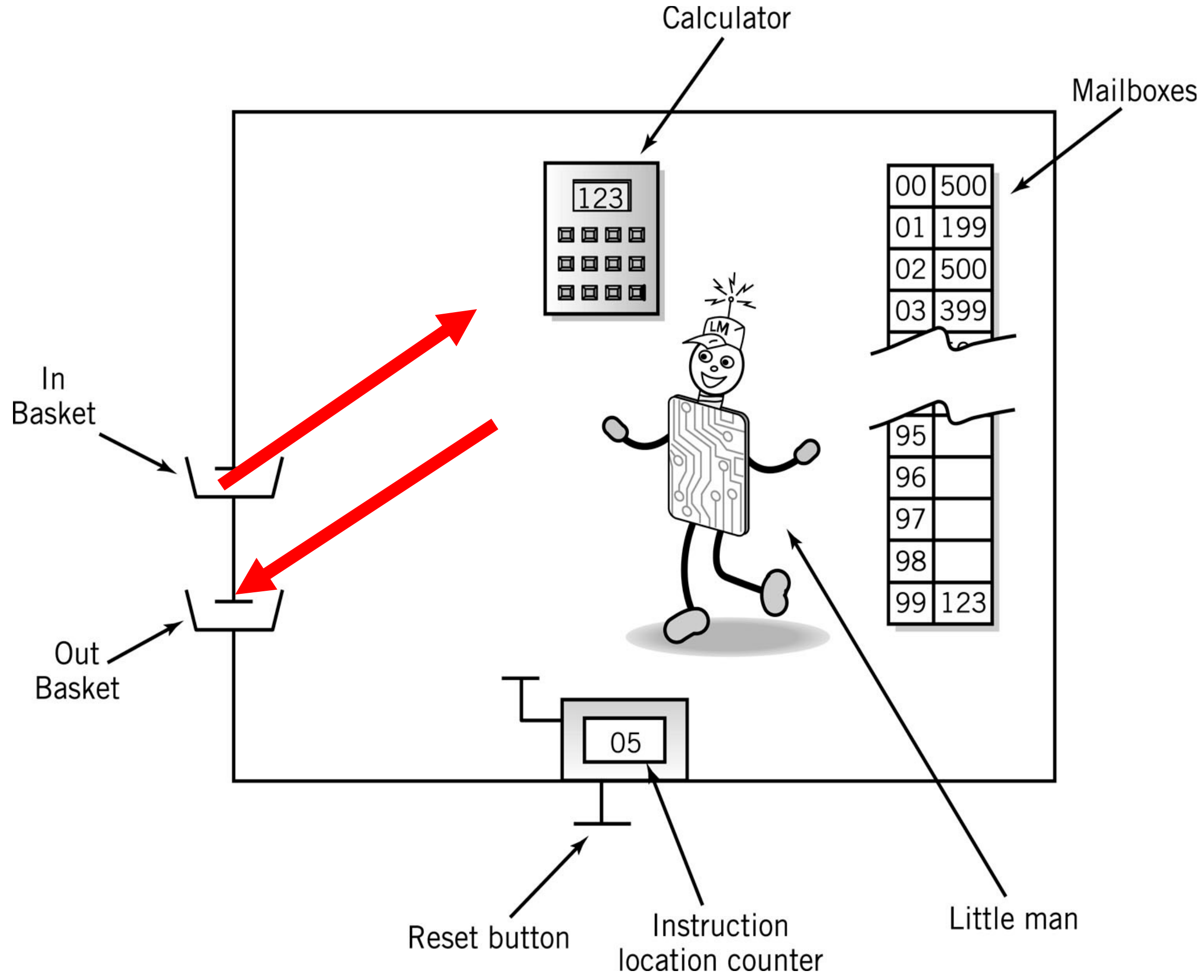
- Move data between calculator and in/out baskets

	Content	
	Op Code	Operand (address)
INP (input)	9	01
OUT (output)	9	02

LMC Input/Output

INP

OUT

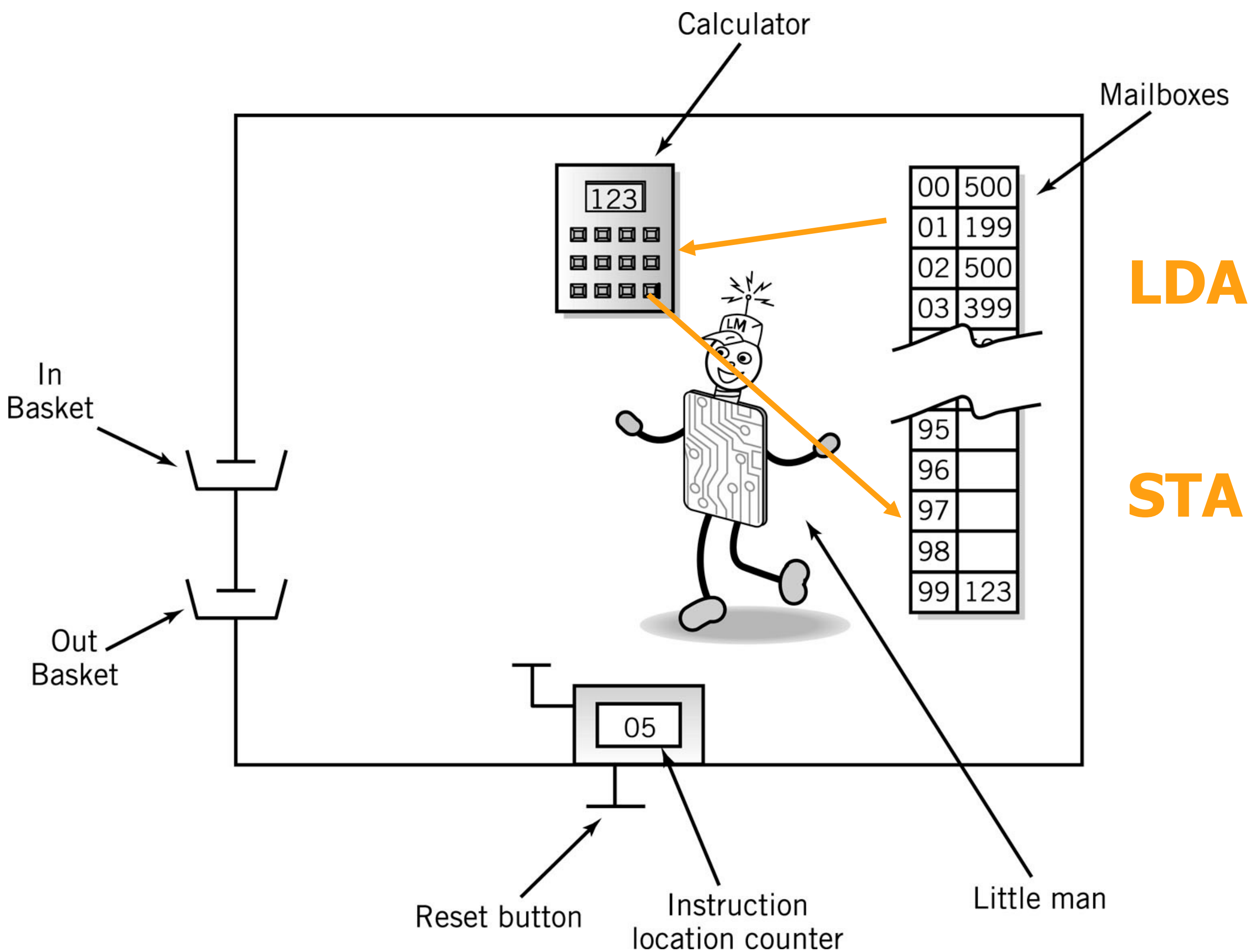


Internal Data Movement

- Between mailbox and calculator

	Content	
	Op Code	Operand (address)
STA (store)	3	xx
LDA (load)	5	xx

LMC Internal Data

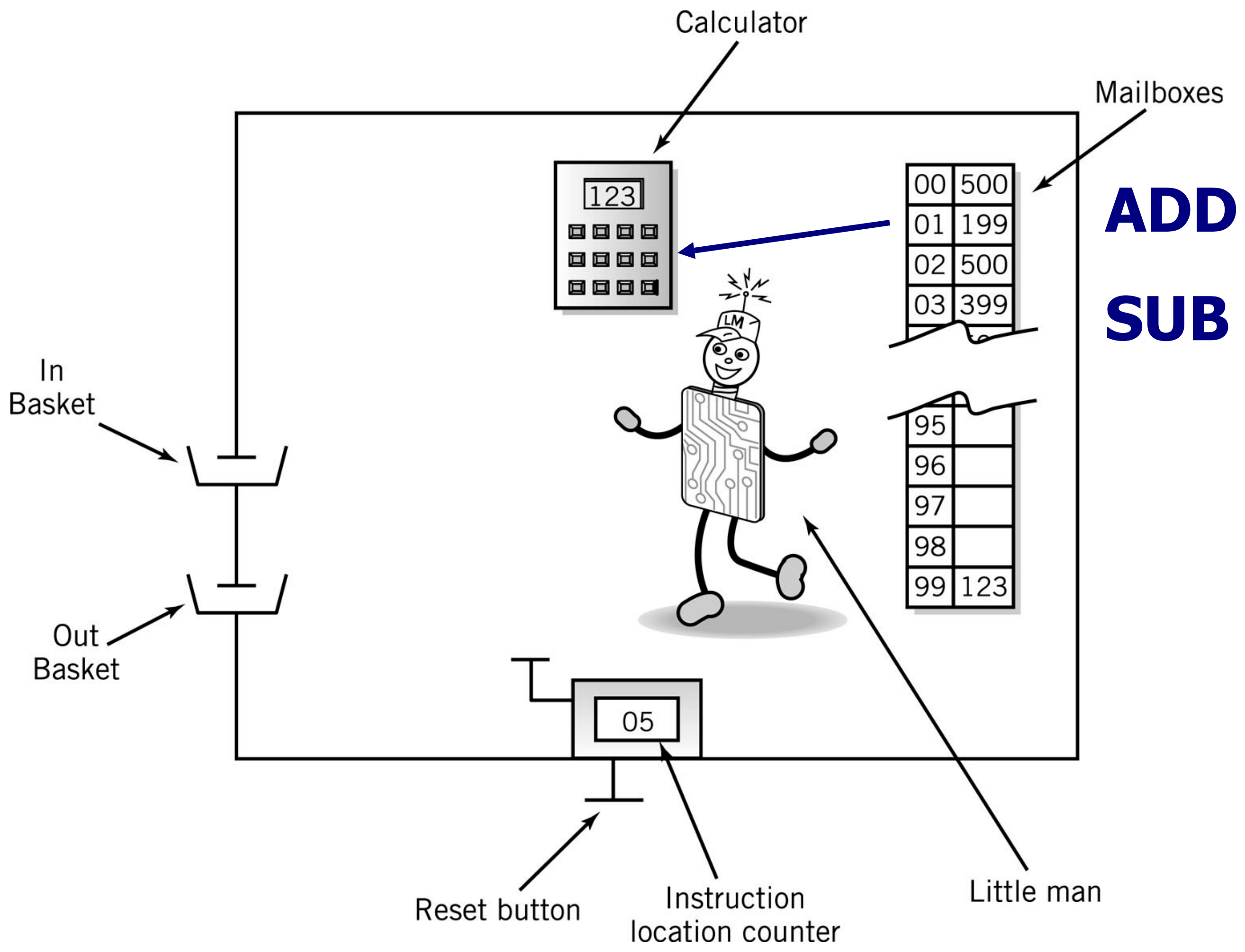


Arithmetic Instructions

- Perform operation in the calculator

	Content	
	Op Code	Operand (address)
ADD	1	xx
SUB	2	xx

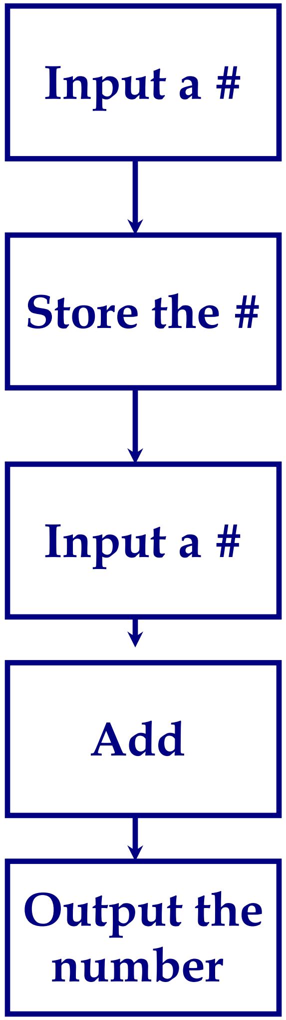
LMC Arithmetic Instructions



Simple Program: Add 2 Numbers

- The logic for adding two numbers in the LMC is given on the right. Write out the series of LMC instructions that implements this logic.

- Assume data is stored in mailboxes with addresses >90



Program to Add 2 Numbers

Mailbox	Code	Instruction Description
00	901	;input 1st Number
01	399	;store data
02	901	;input 2nd Number
03	199	;add 1st # to 2nd #
04	902	;output result
05	000	;stop

Assembler

- The first computers were all programmed with numeric codes
- Binary numbers were used instead of the decimal numbers used in the LMC model
- This can be tedious and error-prone
- Assembler programs were developed to simplify the job of writing code

Assembler

- A mnemonic code is created for each instruction
- A programmer writes the program in assembly language using these mnemonics
- This is much easier as the mnemonics are easy to remember
- The assembly language program is fed into the assembler
- The assembler reads the program and produces the machine code as output

Assembler Advantages

- Programs can be written much more quickly when written in assembler
- Programs written in assembler are much easier to understand and debug
- "Directives" can be used in the assembler such as "DAT" which tells the compiler to reserve a space in memory to be used for data DAT can also tell the assembler to put a constant into that memory address

Assembler advantages

- Most assembly languages allow the use of labels.
- A label is simply a word that is used to either name a memory address where an instruction or data is stored, or to refer to that address in an instruction.
- When a program is assembled.
  - A label to the left of an instruction mnemonic is converted to the memory address the instruction or data is stored at.
  - A label to the right of an instruction mnemonic takes on the value of the memory address referred to above.

Program to Add 2 Numbers: Using Mnemonics

Mailbox	Mnemonic	Instruction Description
00	INP	;input 1^{st} Number
01	STA 99	;store data
02	INP	;input 2^{nd} Number
03	ADD 99	;add 1^{st} # to 2^{nd} #
04	OUT	;output result
05	HLT	;stop

Program to Add 2 Numbers: Using Mnemonics and Labels

Mailbox	Mnemonic	Instruction Description
00	INP	;input 1st Number
01	STA NUM1	;store data in memory labelled NUM1
02	INP	;input 2nd Number
03	ADD NUM1	;add 1st # to 2nd #
04	OUT	;output result
05	HLT	;stop
DAT	NUM1	Labelled data area called NUM1

Assembler advantages

- Labels make writing in assembler much easier as the assembler keeps track of the actual memory address
- If you insert extra code in a program and therefore cause subsequent instructions and data to be moved to a different address, without labels you would have to adjust any memory references accordingly
- With labels, the assembler will automatically re-calculated all labeled addresses when the program is re-assembled.

An Extended Instruction Set

- The instruction set we have specified does not provide any means for branching or looping, both of which are very important constructs for programming.

- We need to extend the instruction set to provide the LMC to carry out more complex programs.

Program Control

- Branching
  - executing an instruction out of sequence
  - Changes the address in the counter
- Halt
  - Stops the work

	Content	
	Op Code	Operand (address)
BR (Jump)	6	xx
BRZ (Branch on Calc=0)	7	xx
BRP (Branch on Calc=+)	8	xx
HLT (stop)	0	(ignore)

Program control

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	.		
	.		
	.		
59	BR 45	645	End of Task; loop to test again
60			Next statement

Extended Instruction Set + Mnemonics

Arithmetic	1xx	ADD
	2xx	SUB
Data Movement	3xx	STORE (Calc Data to Mailbox)
	5xx	LOAD (Mailbox Data to Calc)
BR	6xx	JUMP
BRZ	7xx	BRANCH ON 0
BRP	8xx	BRANCH ON +
Input/Output	901	INPUT
	902	OUTPUT
Machine Control (coffee break)	000	HALT HLT

Find Positive Difference of 2 Numbers

00	INP	901	
01	STA 10	310	
02	INP	901	
03	STA 11	311	
04	SUB 10	210	
05	BRP 08	808	;test
06	LDA 10	510	;if negative, reverse order
07	SUB 11	211	
08	OUT	902	;print result and
09	HLT	000	;stop
10	DAT	000	;used for data
11	DAT	000	;used for data

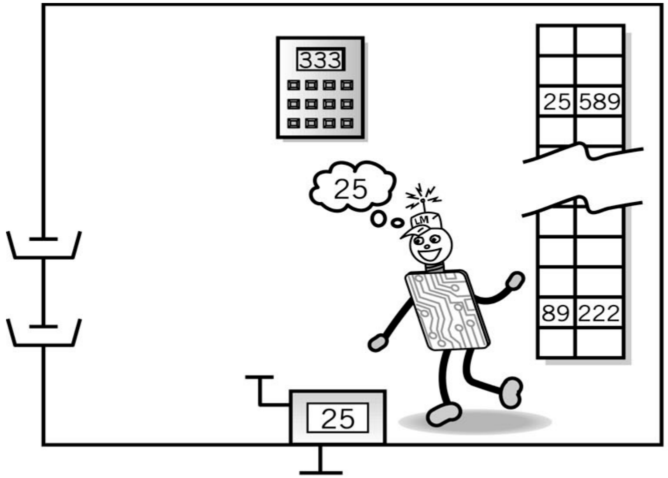
Find Positive Difference of 2 Numbers Using Labels

00	INP	901	
01	STA FIRST	310	
02	INP	901	
03	STA SECOND	311	
04	SUB FIRST	210	
05	BRP OUTPUT	808	;test
06	LDA FIRST	510	;if negative, reverse order
07	SUB SECOND	211	
08	OUTPUT OUT	902	;print result and
09	HLT	000	;stop
10	FIRST DAT	000	;used for data
11	SECOND DAT	000	;used for data

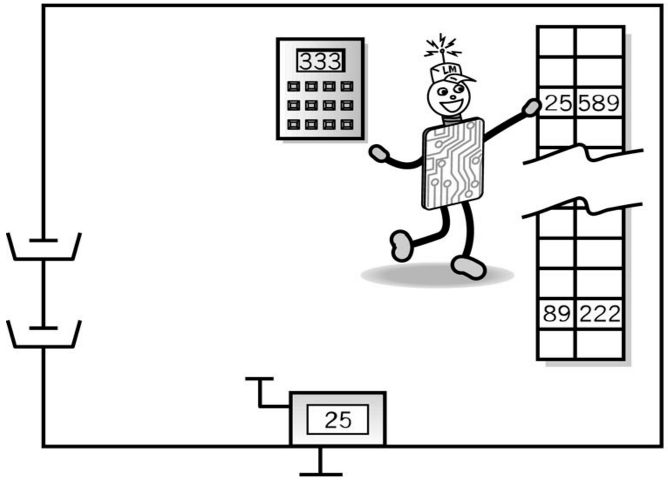
Instruction Cycle

- *Fetch*: Little Man finds out what instruction he is to execute
- *Execute*: Little Man performs the work.

Fetch Portion of Fetch and Execute Cycle

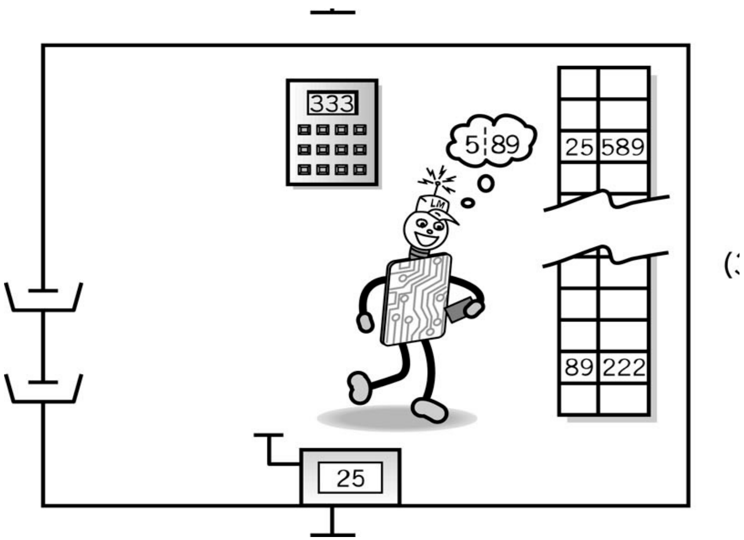


1. Little Man reads the address from the location counter



2. He walks over to the mailbox that corresponds to the location counter

Fetch, cont.

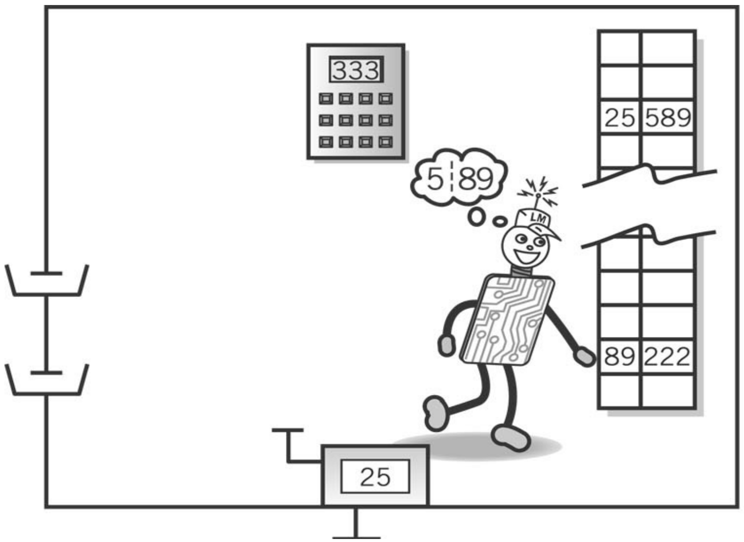


3. And reads the number on the slip of paper (he puts the slip back in case he needs to read it again later)

Execute Portion

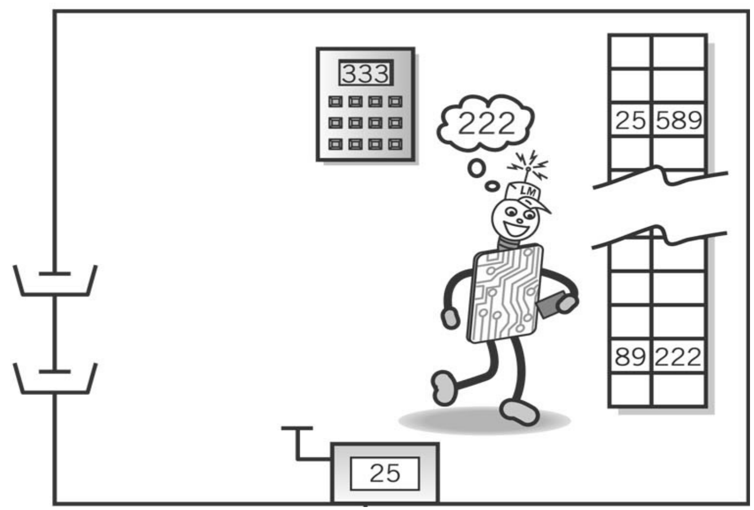
- The execution portion of each instruction is, of course, different for each instruction.
- However, even here, there are many similarities.
- The load instruction (LDA) is typical.

Execute Portion (LDA)

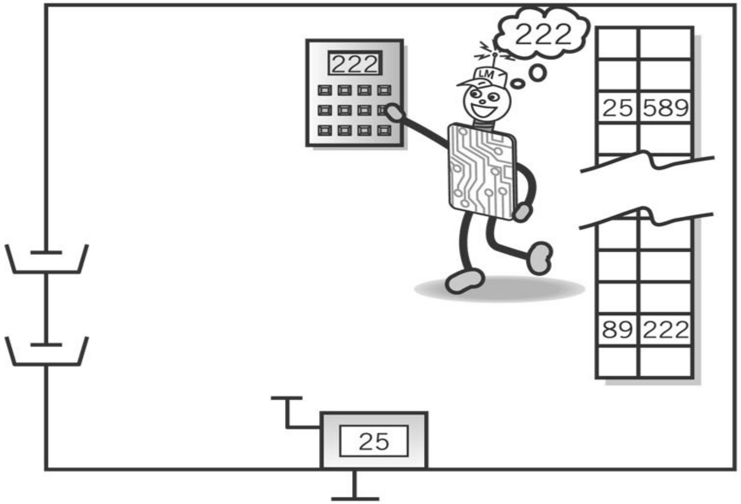


1. The Little Man goes to the mailbox address specified in the instruction he just fetched.

2. He reads the number in that mailbox (he remembers to replace it in case he needs it later).

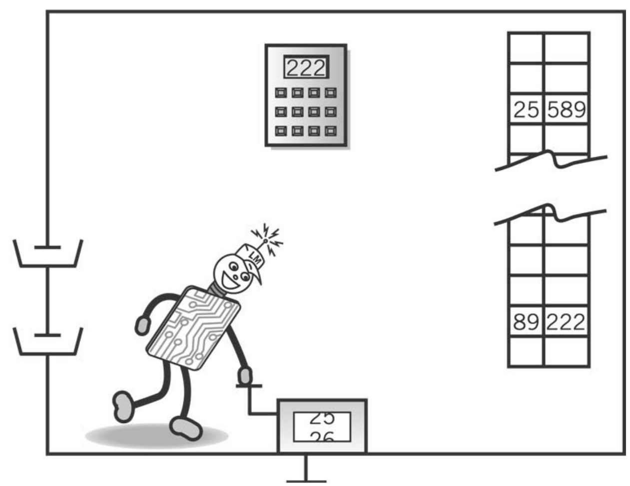


Execute, cont.



3. He walks over to the calculator and punches the number in.

4. He walks over to the location counter and clicks it, which gets him ready to fetch the next instruction.



von Neumann Architecture (1945)

- John Von Neumann is usually considered to be the developer of modern computer architecture.

- The major guidelines that define a Von Neumann architecture are:
  - Stored program concept - memory holds both programs and data.
  - Memory is addressed linearly – i.e address consists of a single number
  - Memory is addressed without regard to content (instruction or data)

Von Neumann Architecture (1945)

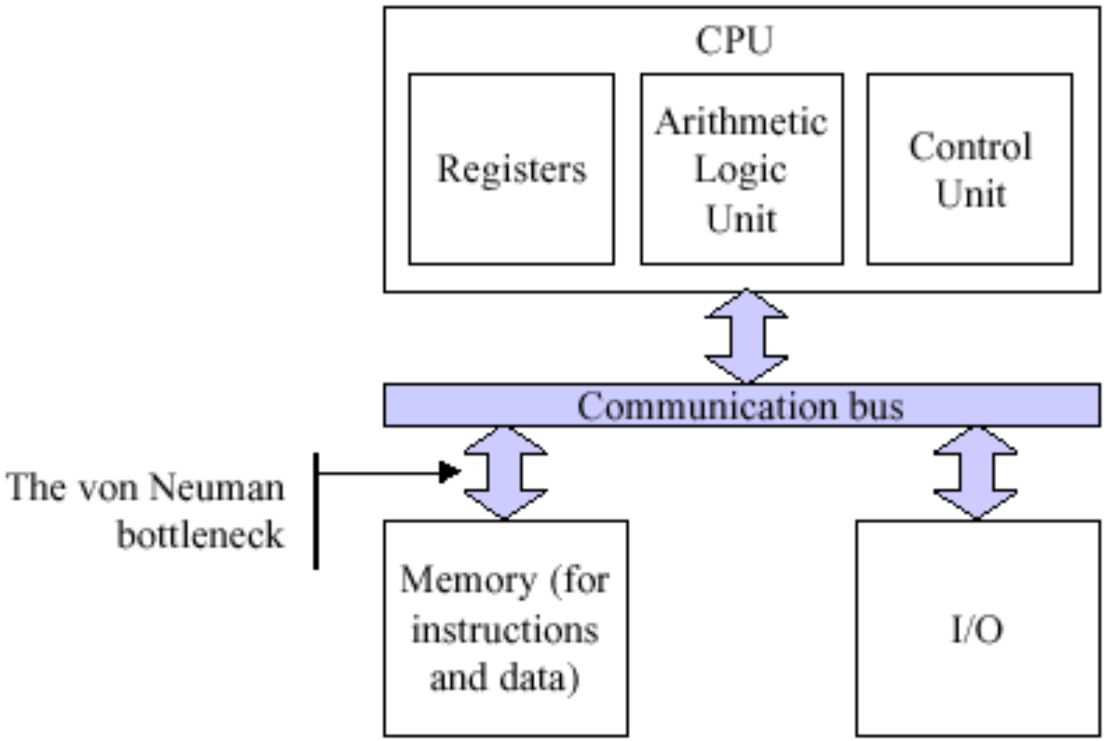
- Instructions are executed sequentially unless an instruction or an outside event cause a branch to occur.

Von Neumann Architecture

- John Von Neumann defined the functional organisation of the computer to be made up of:
  – A control unit that executes instructions
  – An arithmetic logic unit that performs arithmetic and logical calculations,
  – Memory

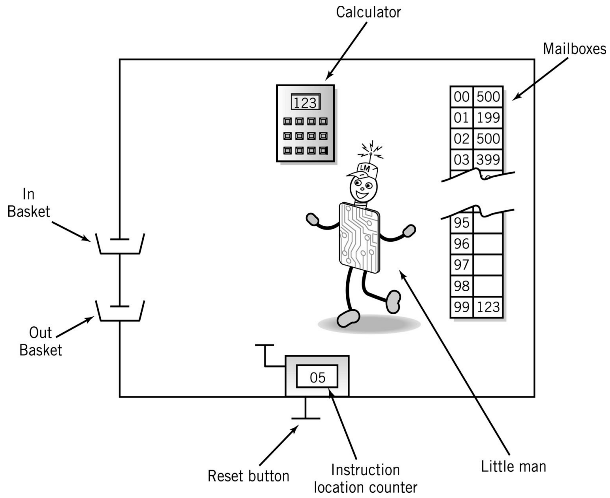
Structure of a Simple Computer

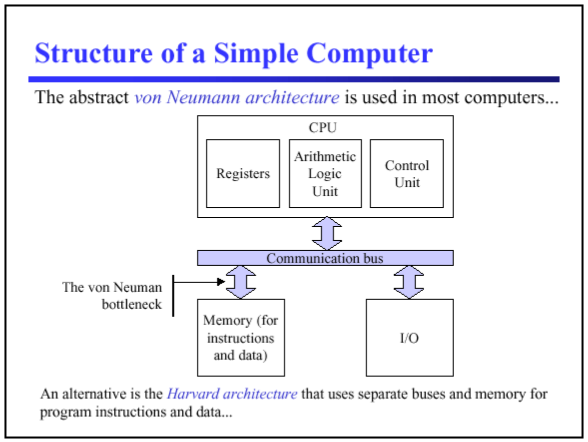
The abstract *von Neumann architecture* is used in most computers...



An alternative is the *Harvard architecture* that uses separate buses and memory for program instructions and data...

LMC & von Neumann





- If you check over the guidelines and organization just described, you will observe that the LMC is an example of a von Neumann architecture :)

LMC & von Neumann

- LMC is an example of a von Neumann architecture
- The Little Man carries out the functions of the control unit fetching, interpreting and carrying out instructions from memory one-at-a-time
- The Little Man also moves the data around which is the function of the communication bus
- Registers are electronic storage internal to the central processing unit and are much faster than main memory storage
- The LMC has 4 registers, the in-tray, the out-tray the counter and an accumulator register (in calculator)
- The mail boxes act as main memory, containers for instructions and also for data

LMC & von Neumann

- The counter acts as a program counter which holds the address of the next instruction to be executed
- The calculator acts as the arithmetic and logical unit which can carry out the actual processing of the data
- The calculator also has the accumulator register which it uses to hold the value of the last calculation as well as one operand of an operation (e.g. add)

Summary

- The LMC provides a model of the workings of a computer.
- The LMC works by following simple instructions, which are described by numbers.
- Some of these instructions cause the LMC to change the order in which instructions are executed.
- Both data and instructions are stored in individual mail slots. There is no differentiation between the two except in the context of the particular operation taking place.
- The LMC follows the fetch-execute cycle.
- Normally the LMC executes instructions sequentially from the mail slots except when he encounters a branching instruction.
- An assembler makes program development and maintenance much easier