MODULE 4: VON NEUMANN MACHINE ARCHITECRURE

Learning Outcome:

At the end of the exercise student should:

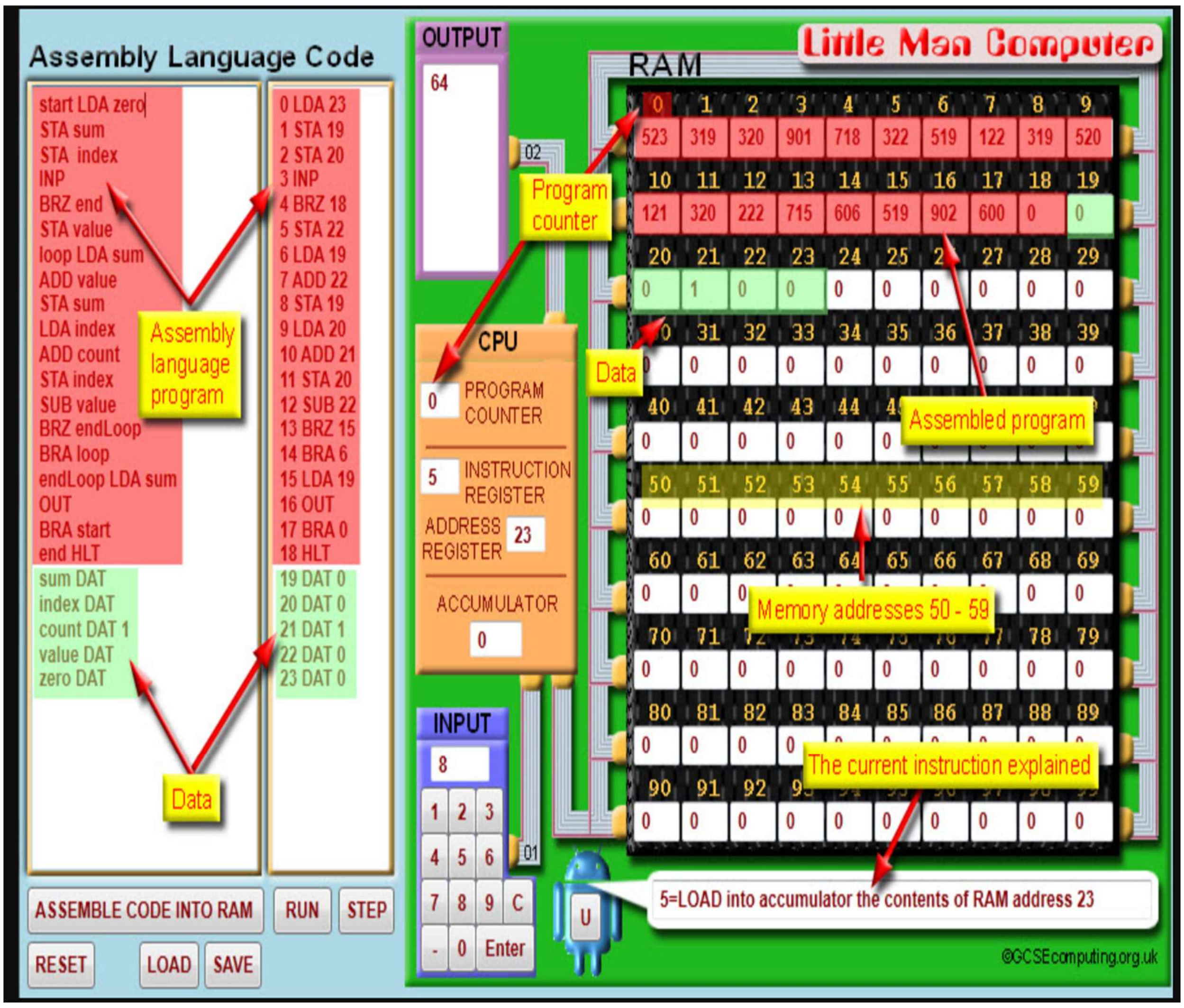
1. Describe Von Neumann Architecture and Littleman Computer
2. Write the mnemonics and program in little man computer
3. Simulate the program and its output

I. Engage Myself

Little Man Computer (LMC) is a simulator that has many of the basic features of a modern computer that uses the Von Neumann architecture (a central processing unit consisting of an arithmetic logic unit and registers, a control unit containing an instruction register and program counter, input and output mechanisms, and RAM to store both data and instructions).

The LMC is based on the idea of a 'Little Man' acting as the control unit of a CPU, fetching instructions from RAM, decoding and executing them as well as managing the input and output mechanisms.

The two versions on this website can be programmed by using a basic set of 10 assembly code instructions which are then assembled into machine code (although in decimal not binary).



```
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
```

OUTPUT: 64

PROGRAM COUNTER: 0
INSTRUCTION REGISTER: 5
ADDRESS REGISTER: 23
ACCUMULATOR: 0
INPUT: 8

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	2	27	28	29
0	1	0	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	0
40	41	42	43	44	4				
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
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	9						
0	0	0	0	0	0	0	0	0	0

5=LOAD into accumulator the contents of RAM address 23

II. Explore it More

Understanding the LMC simulator

The 100 memory addresses in the computer memory are numbered 0 to 99 and can each contain a ‘machine code‘ instruction or data.

Each assembly language instruction is made up of a 3 letter mnemonic (which represents the operation code), usually followed by the memory address of the data the CPU is to act on (this is called absolute memory addressing).

Pressing the Assemble Program button translates the assembly language instructions into ‘machine code‘ and loads them into RAM. it also resets the Program Counter to zero.

The Input box allows the user to enter numerical data (-999 to 999) while a program is running and load it into the accumulator.

The Output box can output the contents of the accumulator while a program is running.

A RAM memory address that is used to store data can be given a meaningful label. Data can also be stored in these named address locations.

The results of any ADD or SUBTRACT instructions are stored in the accumulator .

The Program Counter stores the memory address of the instruction being carried out. It will automatically increment by 1 after each instruction is completed.

If the CPU receives an non-sequential instruction to branch (BRP, BRP or BRZ) then the Program Counter is set to the memory address of that instruction.

Branch instructions are set to branch to a labelled memory location.

To restart a program, the Program Counter is reset to 0.

When assembled, each assembly code instruction is converted into a 3 digit ‘machine code’ instruction (1 digit for the instruction and 2 for the memory address). The 3 digit ‘machine code’ instructions are then loaded into RAM, starting at memory address 0.

Any data is also loaded into memory at the memory address corresponding to the location of the data in the program (i.e. if the 5th line of the assembly language program was data then this data would be loaded into address 4, because memory address start at 0 not 1)

The ‘Little Man’ can then begin execution, starting with the instruction in RAM at memory address 0.

The ‘Little Man’ performs the following steps to execute a program:

1. Check the Program Counter so it knows the memory address to look at.
2. Fetch the instruction from the memory address that matches the program counter.
3. Increment the Program Counter (so that it contains the memory address of the next instruction).
4. Decode the instruction (includes finding the memory address for any data it refers to).
5. If required by the instruction code, fetch the data from the memory address found in the previous step.
6. Execute the instruction and if necessary set the Program Counter to match any branch instructions.

Name:____Score:__________
Course/Sec:____________________Schedule:_______________________Date Submitted:_________________

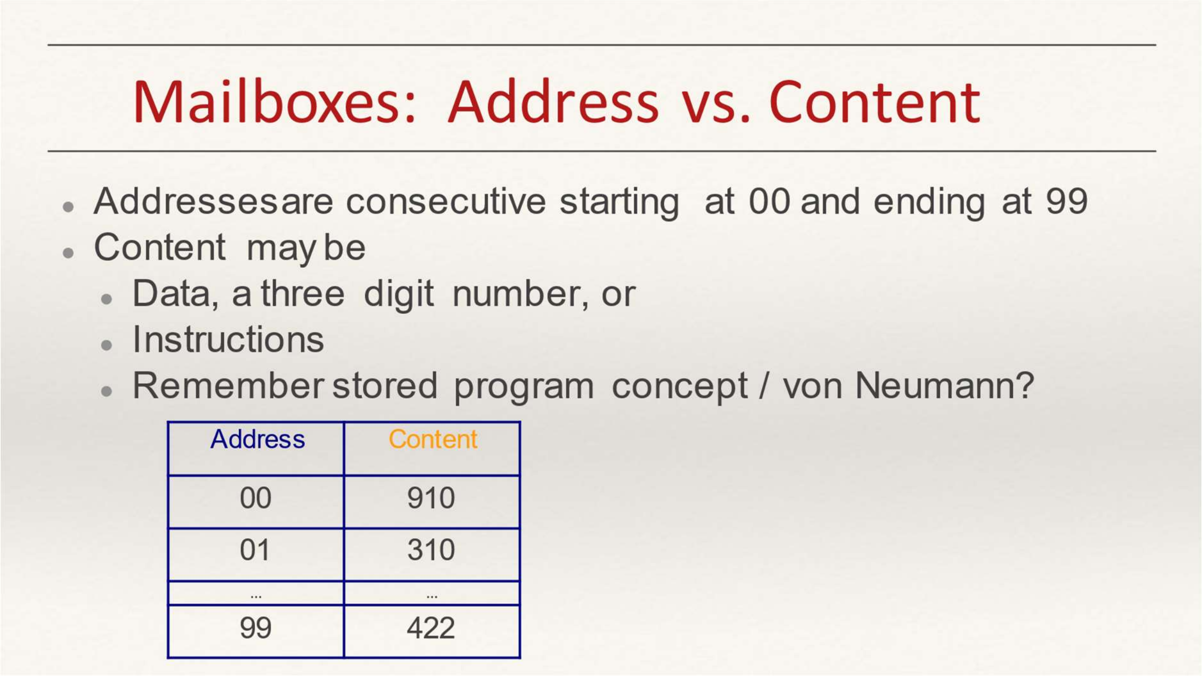
1. Create a program that takes in a number as input and displays in a countdown the number and ends it in 0. (Do While Loop).

2. Create a program that will accept a large number and find and display the sum of its digits.

3. Create the elevator program. Please use your creativity in setting the parameters and specifications.

What to turn In:

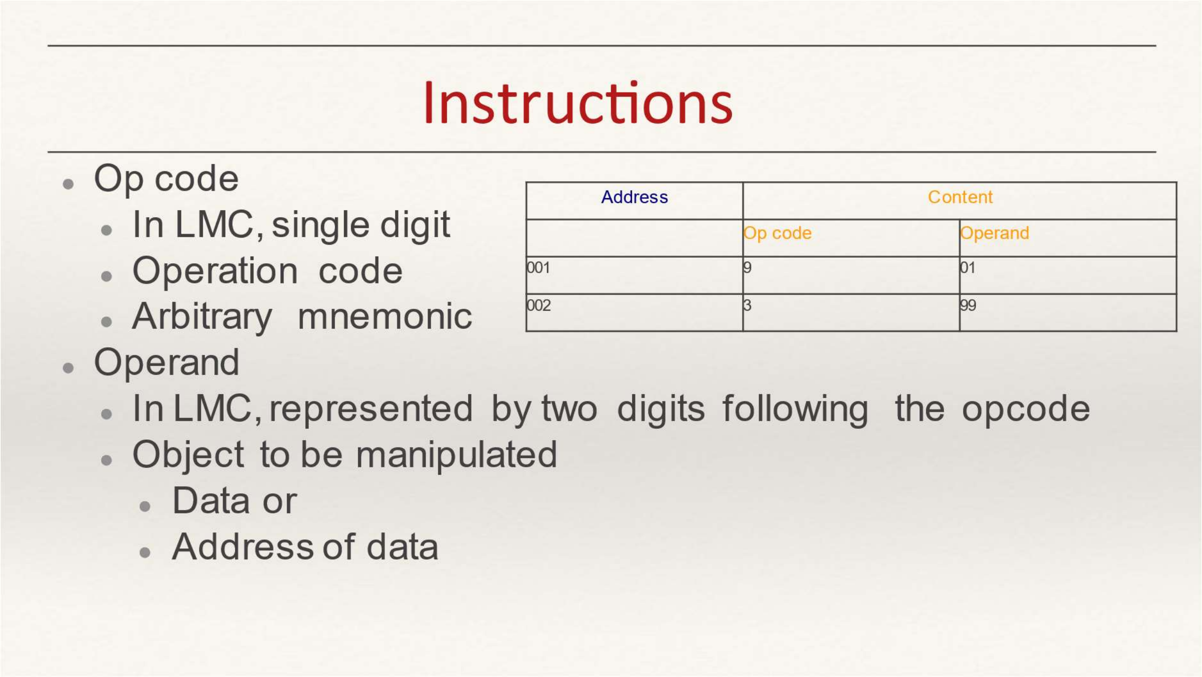
1. Pseudocode AND FLOWCHART
2. Mnemonic assembler
3. LMC
4. Address
5. Screen shot of LMC Simulator Assembler
6. Screenshot of LMC Mailboxes
7. Console Message



Mailboxes: Address vs. Content

- Addressesare consecutive starting at 00 and ending at 99
- Content may be
  - Data, a three digit number, or
  - Instructions
  - Remember stored program concept / von Neumann?

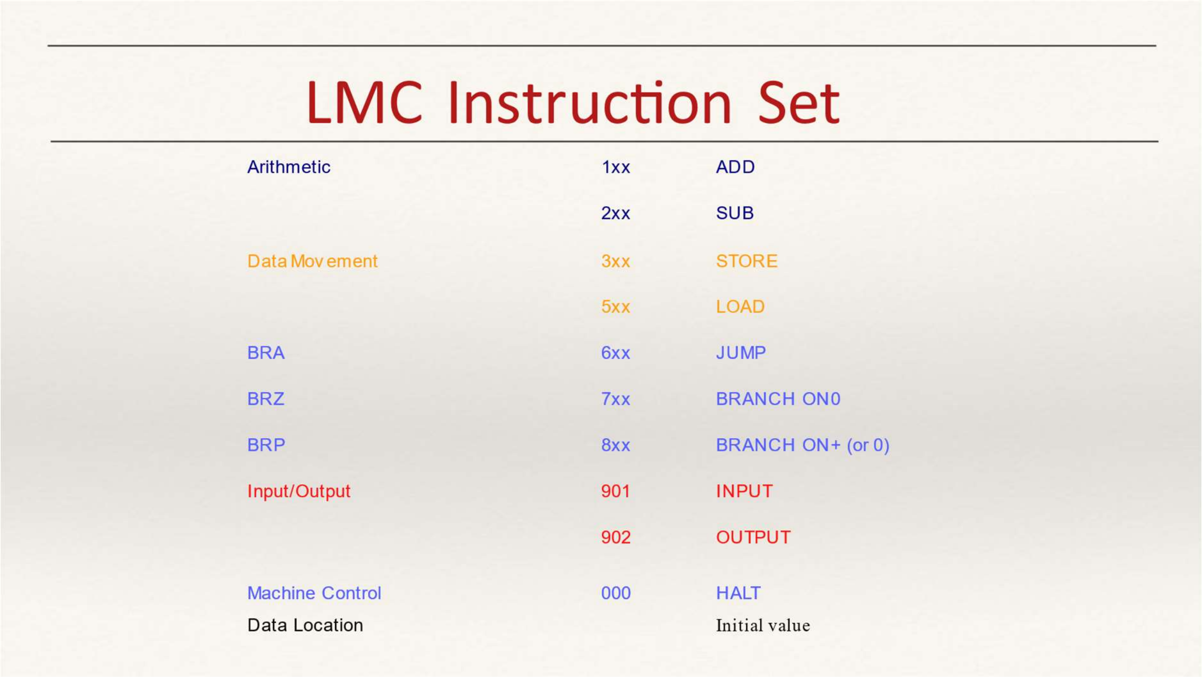
Address	Content
00	910
01	310
…	…
99	422



Instructions

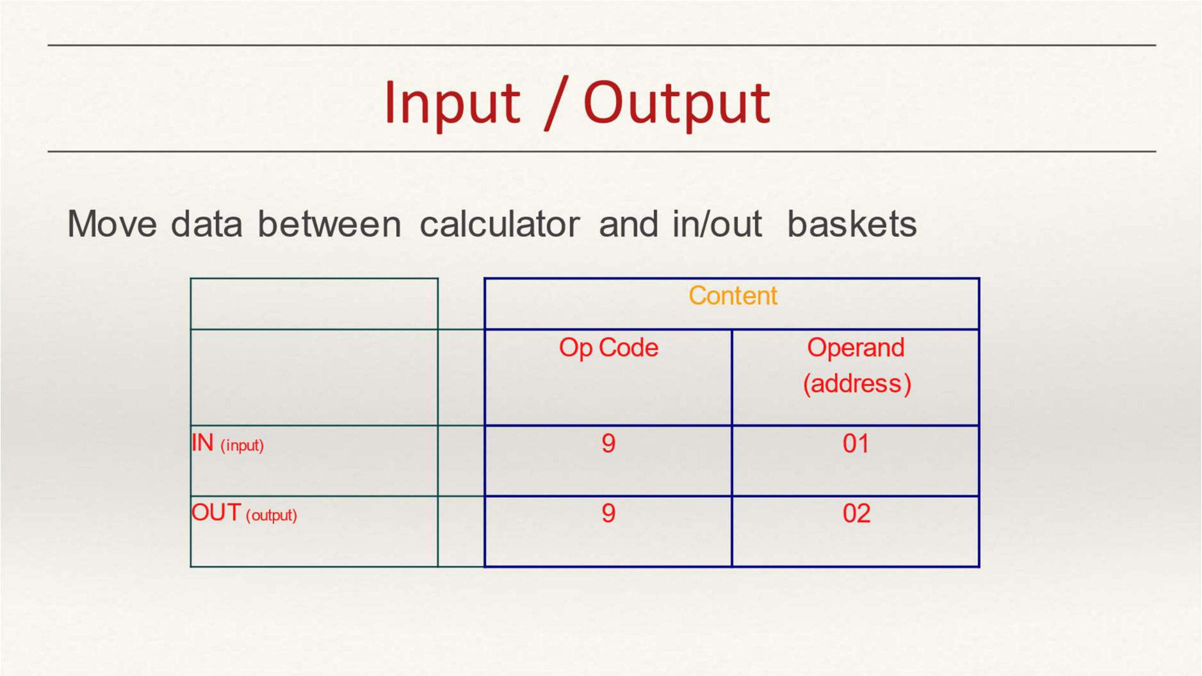
- Op code
  - In LMC, single digit
  - Operation code
  - Arbitrary mnemonic
- Operand
  - In LMC, represented by two digits following the opcode
  - Object to be manipulated
    - Data or
    - Address of data

Address	Content	
	Op code	Operand
001	9	01
002	3	99



LMC Instruction Set

Arithmetic	1xx	ADD
	2xx	SUB
Data Mov ement	3xx	STORE
	5xx	LOAD
BRA	6xx	JUMP
BRZ	7xx	BRANCH ON0
BRP	8xx	BRANCH ON+ (or 0)
Input/Output	901	INPUT
	902	OUTPUT
Machine Control	000	HALT
Data Location		Initial value



Input / Output

Move data between calculator and in/out baskets

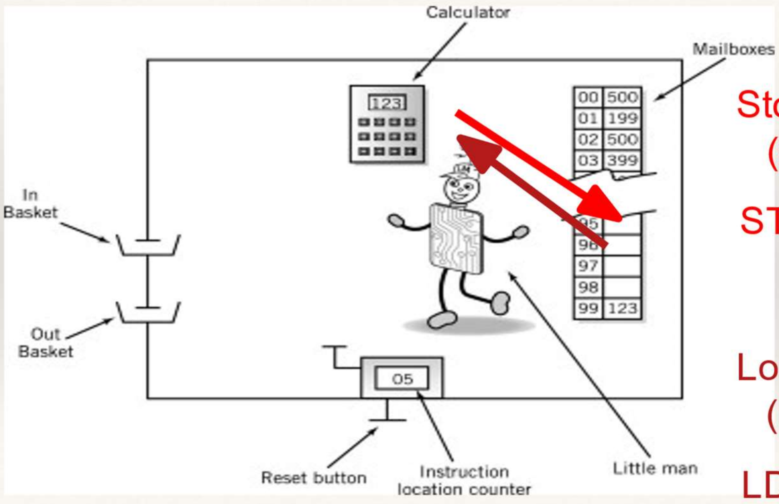
		Content	
		Op Code	Operand (address)
IN (input)		9	01
OUT (output)		9	02

Internal Data Movement

Between mailbox and calculator

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

LMC Internal Data



Store (STO)

STO95

Load (LDA)

LDA96

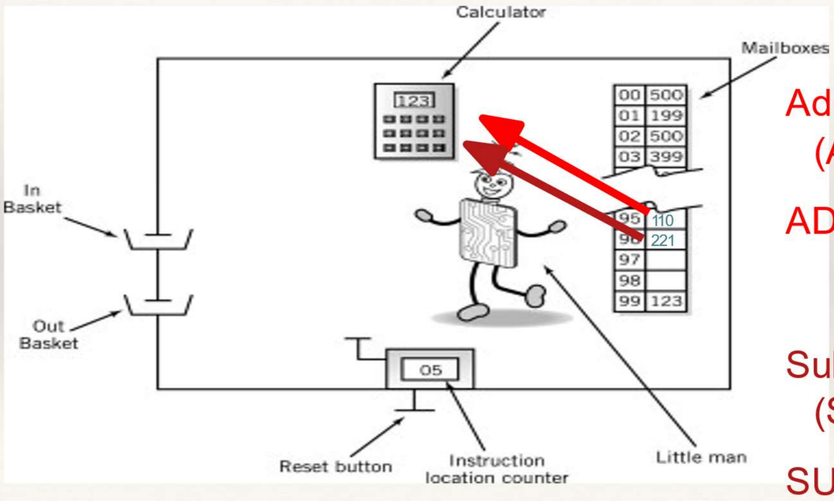
Arithmetic Instructions

Read mailbox

Perform operation in the calculator

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

LMC Arithmetic Instructions



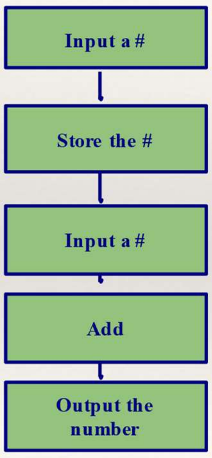
Add
(ADD)

ADD 95

Sub
(SUB)

SUB96

Simple Program: Add 2 Numbers


Input a #

Store the #

Input a #

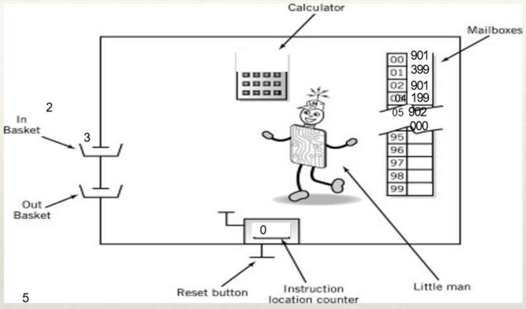
Add

Output the number

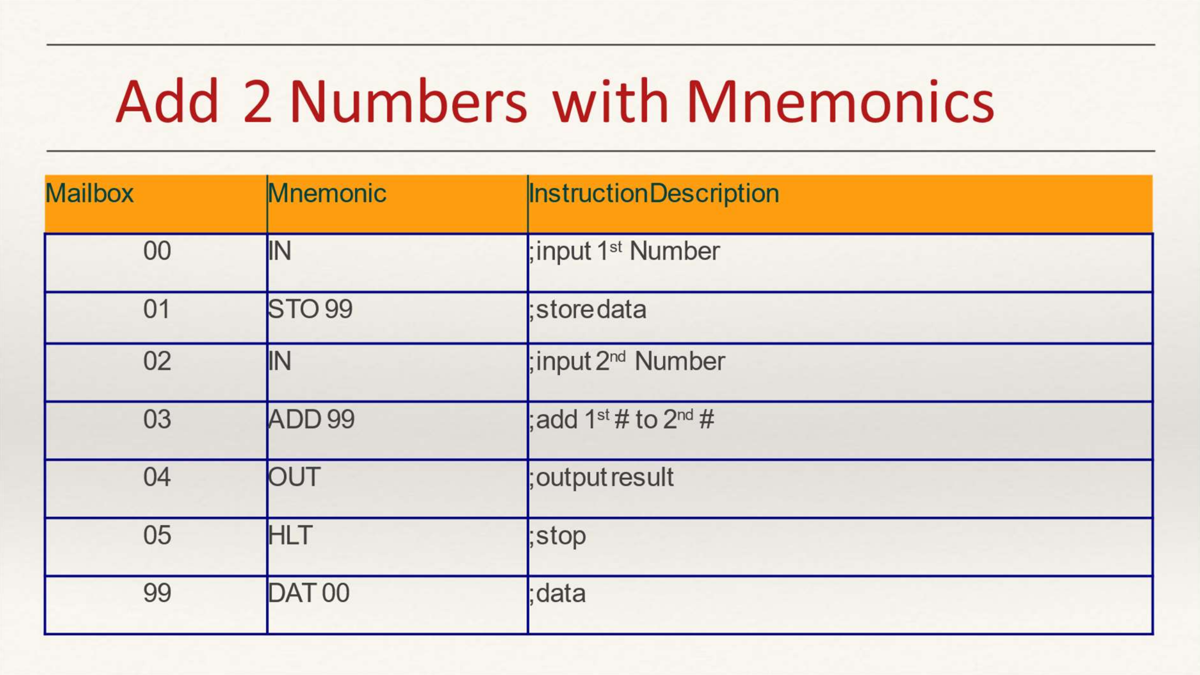
- Assume data is stored in mailboxes with addresses >90
- LMC Instructions

Program to Add 2 Numbers

Mailbox	Code	InstructionDescription
00	901 901	;input 1st Number
01	399	;storedata
02	901	;input 2nd Number
03	199	;add 1st # to 2nd #
04	902	;outputresult
05	000	;stop
99	000	;data

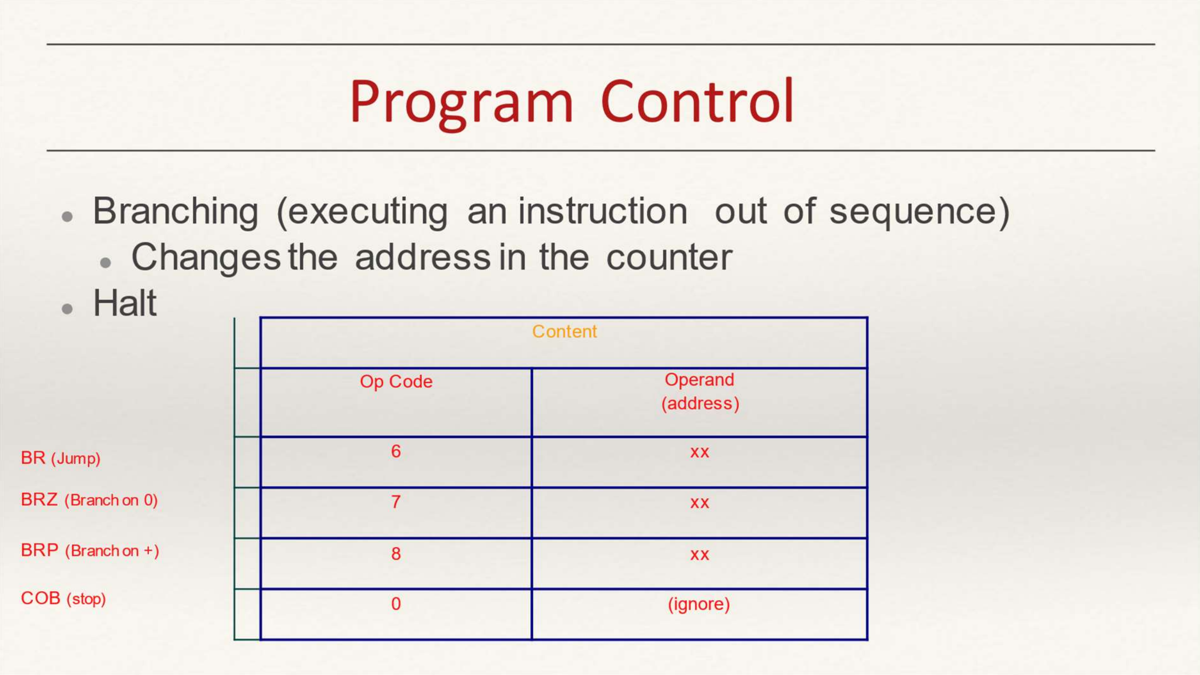


901	1
399	
901	2
199	3
902	4
000	5



Add 2 Numbers with Mnemonics

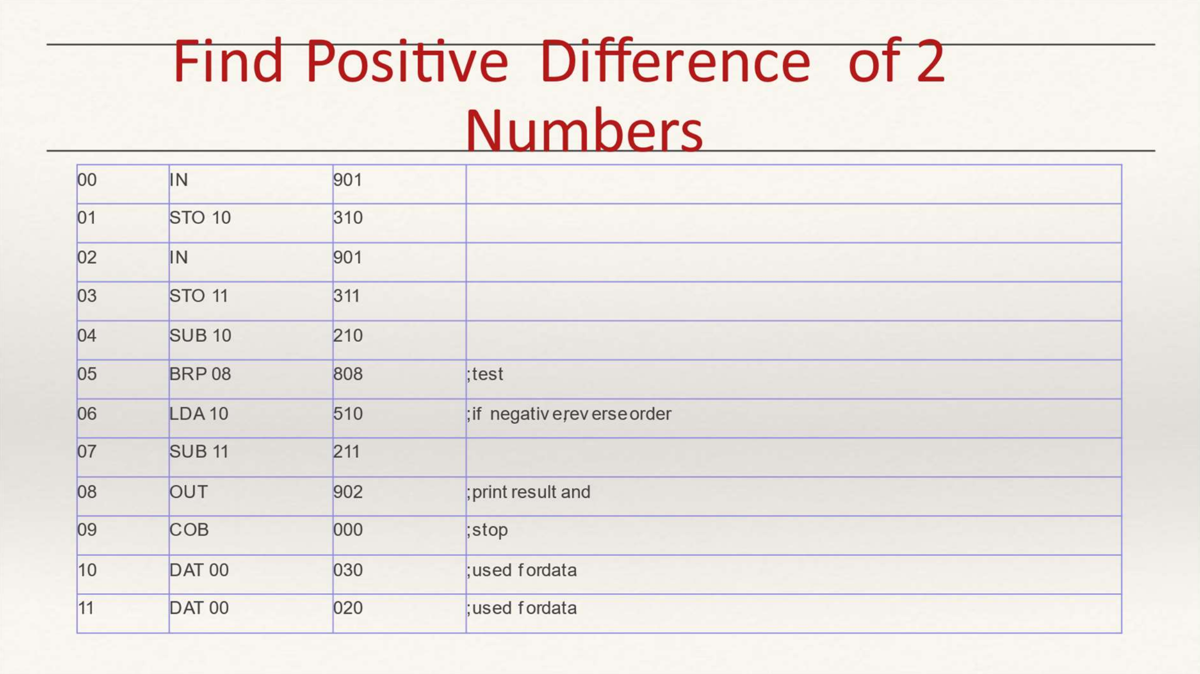
Mailbox	Mnemonic	InstructionDescription
00	IN	;input 1st Number
01	STO 99	;storedata
02	IN	;input 2nd Number
03	ADD 99	;add 1st # to 2nd #
04	OUT	;outputresult
05	HLT	;stop
99	DAT 00	;data



Program Control

- Branching (executing an instruction out of sequence)
  - Changes the address in the counter
- Halt

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



Find Positive Difference of 2 Numbers

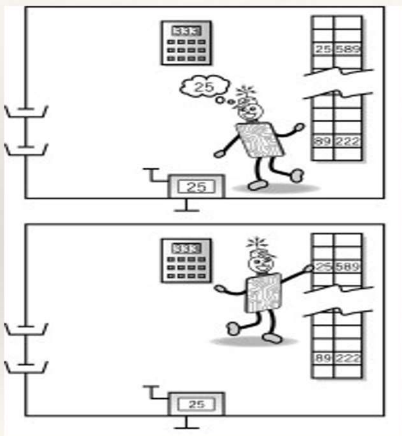
00	IN	901	
01	STO 10	310	
02	IN	901	
03	STO 11	311	
04	SUB 10	210	
05	BRP 08	808	;test
06	LDA 10	510	;if negative,reverseorder
07	SUB 11	211	
08	OUT	902	;print result and
09	COB	000	;stop
10	DAT 00	030	;used fordata
11	DAT 00	020	;used fordata

Instruction Cycle

Fetch: Little Man finds out what instruction he is to execute

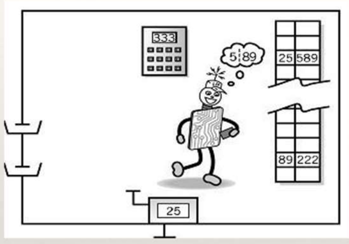
Execute: Little Man performs the work

Fetch Portion



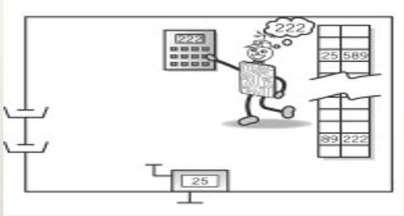
1. Little man reads the address from the location counter
2. He walks over to the mailbox that corresponds to the location counter. (Decode)

Fetch Portion (cont.)

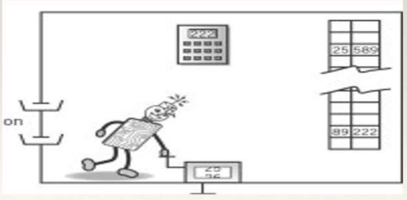


3. And reads the number on the slip of paper (putting the slip back in casehe needs to read it again later)

Execute Portion (cont.)

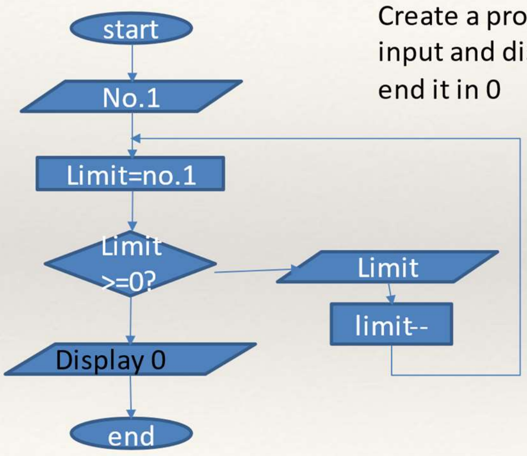


3. He walks over to the calculator and punches the number in.



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

Looping and Iteration



Create a program which takes in a number as input and display in countdown the number and end it in 0

```
Using while loop:
Int limit=input()
while(limit>=0){
output(limit);
limit--;
}
For Loop
Int limit=input();
For (int i=limit; i>=0; i--){
Output(i);
}
```

Countdow n Numbers with Mnemonics

Mailbox	Mnemonic	InstructionDescription	LMC
00	INP	;input 1st Number	901
01	STA limit	;store data	307
02	Loop OUT	;Output the data	902
03	SUB constant1	;decrement limit--	206
04	BRP Loop	;branch on positive	802
05	HLT	;stop	000
06	Constant1DAT 001	;data 001	001
07	Limit DAT	;data 000	000