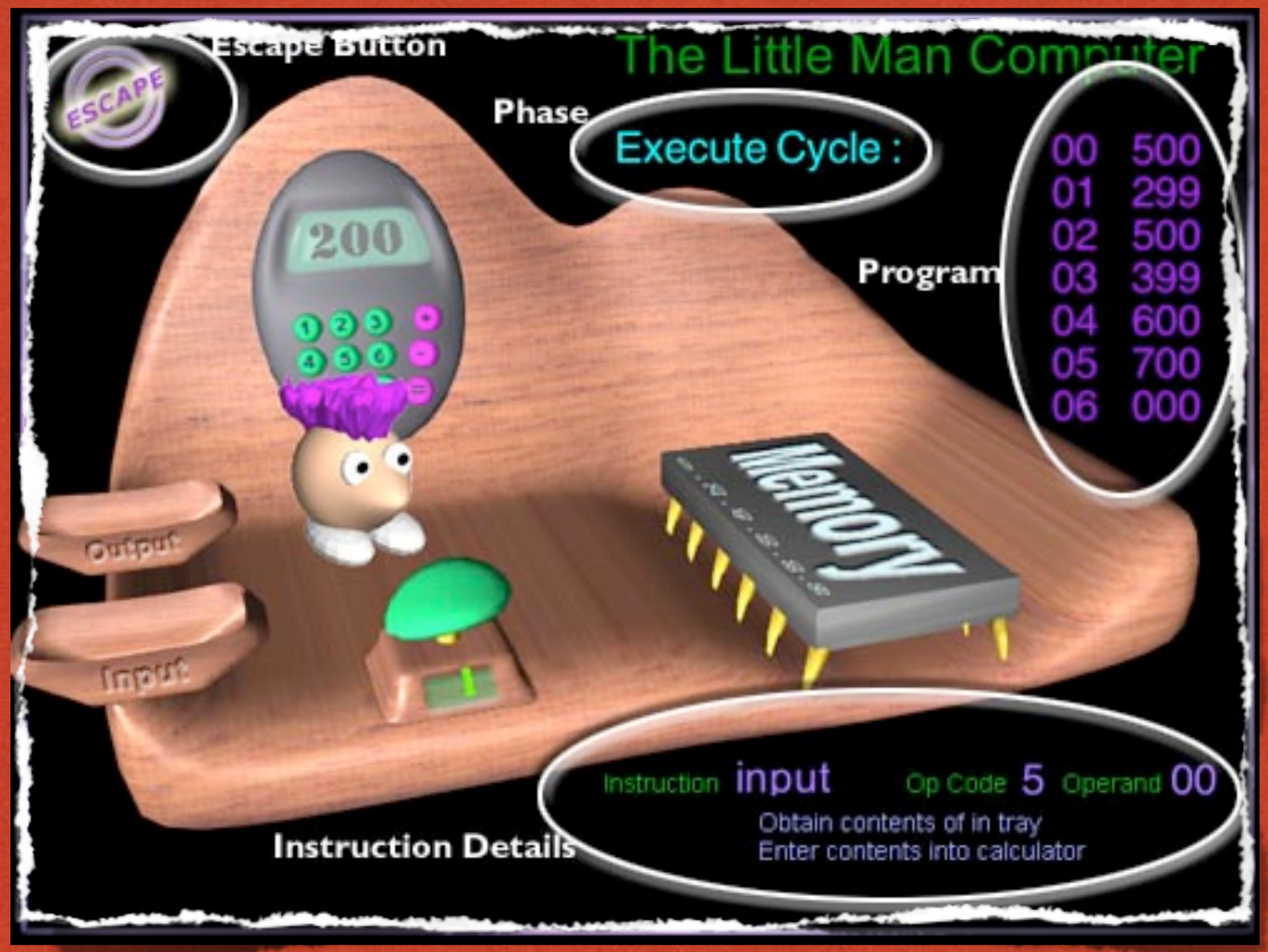


LITTLE MAN COMPUTER

ADMIN ITEMS

16:30 - Introduction

16:45 - Session 1 - Simple Programs

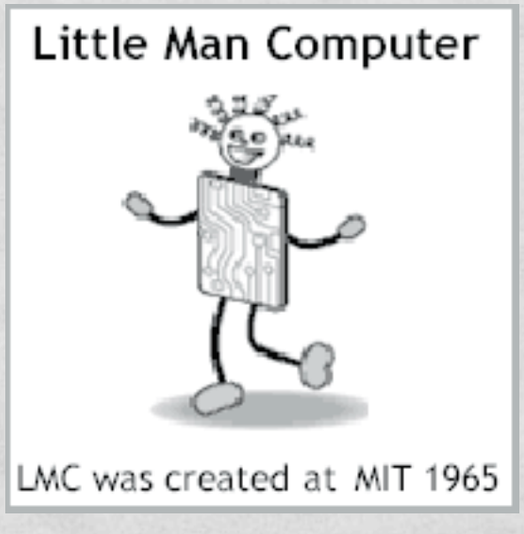
17:15 - Session 2 - Branching

17:45 - Session 3 - Application

18:15 - Plenary

pi.mwclarkson.co.uk

LITTLE MAN COMPUTER



A small man (or woman)

100 empty mailboxes

1 instruction at a time

pi.mwclarkson.co.uk

INSTRUCTION SET

INP	Get some **input** from the user
OUT	**Output** the current value to the user
STA *variableName*	**Store** the current value in *variableName*
LDA *variableName*	**Load** the value in *variableName*
ADD *variableName*	**Add** the value in *variableName*
SUB *variableName*	**Subtract** the value in variableName
HLT	**Halt** (end) the program
variableName DAT	Declare *variableName* as **data**

pi.mwclarkson.co.uk

THE ACCUMULATOR

INP	Input **to the accumulator**
OUT	Output **from the accumulator**
STA *variableName*	Store from **the accumulator** to *variableName*
LDA *variableName*	Load into **the accumulator** from *variableName*
ADD *variableName*	Add *variableName* to **the accumulator**
SUB *variableName*	Subtract *variableName* from **the accumulator**
HLT	**Halt** (end) the program
variableName DAT	Declare *variableName* as **data**

pi.mwclarkson.co.uk

WHAT WILL THIS DO?

Message Box:

```
INP
STA numOne

INP
STA numTwo

LDA numOne
ADD numTwo

STA numThree

OUT

HLT

numOne DAT
numTwo DAT
numThree DAT
```

Input a number & store it as *numOne*

Input a number & store it as *numTwo*

Load *numOne* & add *numTwo* to it

Store the answer as *numThree*

Output the answer & stop

Variable declaration (always goes last)

pi.mwclarkson.co.uk

TEST IT OUT!

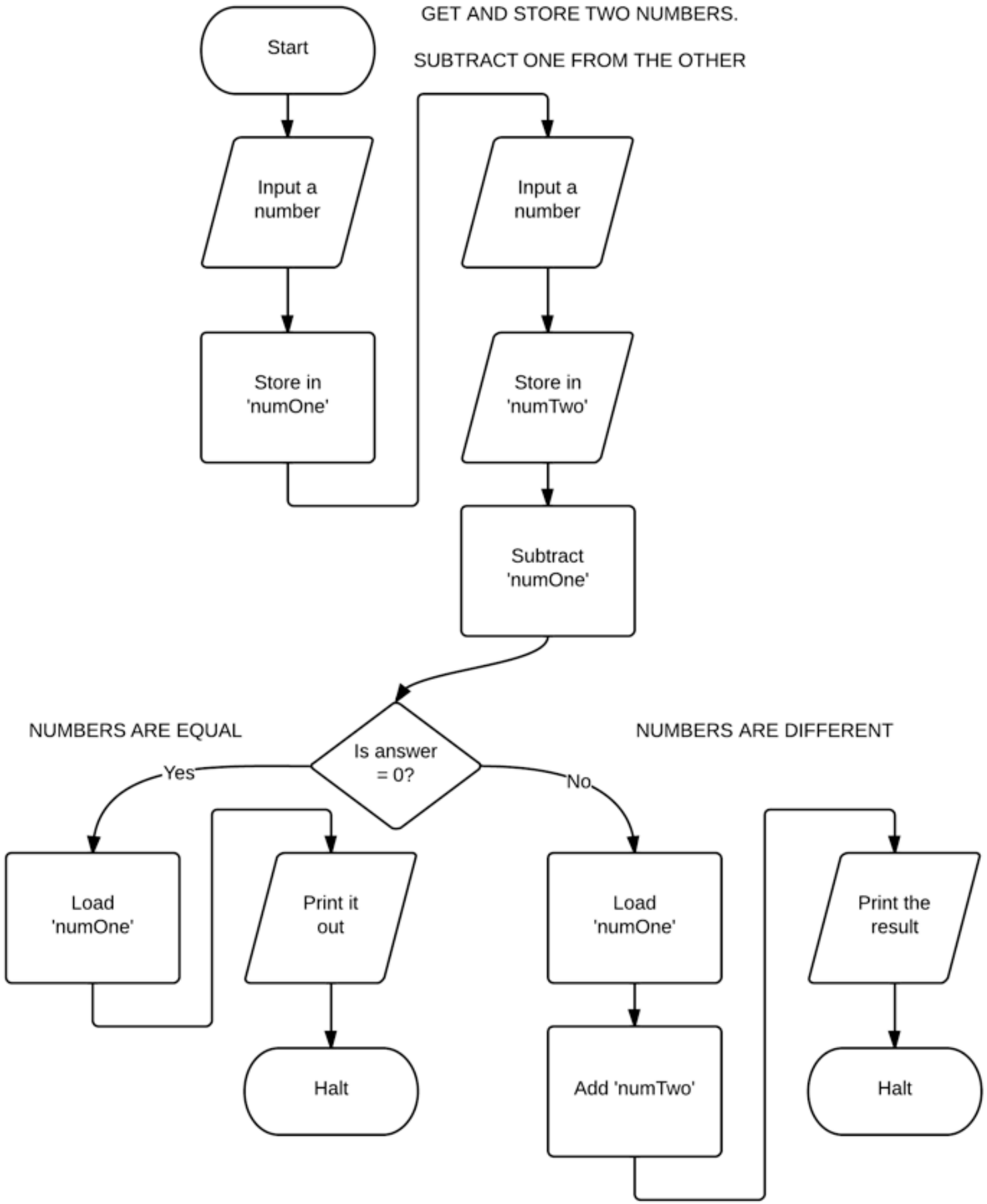
Message Box:

```
INP
STA numOne

INP
STA numTwo

LDA numOne
ADD numTwo

STA numThree

OUT

HLT

numOne DAT
numTwo DAT
numThree DAT
```

pi.mwclarkson.co.uk

BRANCHES

Code	Meaning
BRZ	Branch if zero
BRP	Branch if positive (or zero)
BRA	Branch always (used for looping)


GET AND STORE TWO NUMBERS.
SUBTRACT ONE FROM THE OTHER
Start
Input a number
Store in 'numOne'
Input a number
Store in 'numTwo'
Subtract 'numOne'
Is answer = 0?
NUMBERS ARE EQUAL
Yes
Load 'numOne'
Print it out
Halt
NUMBERS ARE DIFFERENT
No
Load 'numOne'
Add 'numTwo'
Print the result
Halt

APPLICATION TO THEORY

Introduction to programming - Sequencing

Introduction to programming - Variables

Representation of data - Instructions

Hierarchy of programming languages

Translators - Compilers, Interpreters & Assemblers

Fetch-Execute Cycle

pi.mwclarkson.co.uk

SEQUENCING

If you can dodge a wrench, you can dodge a ball

VARIABLES

REPRESENTATION OF DATA

```
----- Translating Mnemonics -----
Line 0 : INP
    Opcode = 901
Line 1 : STA
    Opcode = 3  Address = 08
Line 2 : INP
    Opcode = 901
Line 3 : STA
    Opcode = 3  Address = 09
Line 4 : LDA
    Opcode = 5  Address = 08
Line 5 : ADD
    Opcode = 1  Address = 09
Line 6 : OUT
    Opcode = 902
Line 7 : HLT
    Opcode = 0
Line 8 : DAT
Line 9 : DAT
----- Program Successfully Compiled -----
```

Instruction	Mnemonic	MachineCode
Load	LDA	5xx
Store	STA	3xx
Add	ADD	1xx
Subtract	SUB	2xx
Input	INP	901
Output	OUT	902
End	HLT	000
Branch if zero	BRZ	7xx
Branch if zero or positive	BRP	8xx
Branch always	BRA	6xx
Data storage	DAT	

HIERARCHY OF PROGRAMMING LANGUAGES

High-Level Language e.g. Python / Java / VB	numOne = numOne + numOne
Low-Level Language e.g. Assembly Language	LDA numOne ADD numOne STA numOne
Machine Code	0011 1100 0001 1100 0011 1100

TRANSLATORS

High-Level Language e.g. Python / Java / VB	numOne = numOne + numOne
Low-Level Language e.g. Assembly Language	LDA numOne ADD numOne STA numOne
Machine Code	0011 1100 0001 1100 0011 1100

FETCH-EXECUTE CYCLE

```
INP
STA numOne
INP
STA numTwo
LDA numOne
ADD numTwo
STA numTwo
OUT
HLT
numOne DAT
numTwo DAT
```

pi.mwclarkson.co.uk

PLENARY

When a pupil asks you if they're allowed to program in opcodes instead of mnemonics, you know that they've got it

pi.mwclarkson.co.uk